



# UNIVERSITY OF SURREY

Faculty of Engineering and Physical Sciences

Department of Computing

Final Year Project Report

Title: A Secure E-voting Web Application for AGM Elections

Student: Ayoolamide Oye-dada

URN: 6487008

Supervisor: Dr Steve Schneider

Date: 18/05/2021

## Declaration of Originality

---

*The work submitted in this report is my own. Where work from others is used, it has been clearly identified and referenced using IEEE referencing standards. The University of Surrey may seek to confirm this by means using a plagiarism detection service such as Turnitin UK. Lastly any penalised occurred in the assessment of this project as a result of plagiarism are fully accepted.*

## Acknowledgments

---

*I would like to dedicate this work to my mother, for her love and financial support that has allowed this project to be carried out. Her unwavering faith in my abilities during this academic year has been my light in moments of darkness.*

*My adoration and appreciation goes out to my supervisor, Dr Steve Schneider, who has gone above and beyond to provide the guidance required to complete this project. In addition, I would like to sincerely thank Alex Harden, Theo Donnell, Katherine Phillips for agreeing to be interviewed and demonstrating how the USSU's existing E-Voting System works. Their cooperation has significantly contributed to this project and stands as a testament to what can be achieved when stakeholders contribute to the development of software. Lastly, I would also like to thank my friends and support network who have motivated and encouraged me throughout the completion of this project.*

## Abstract

---

Data from the most recent NSS and Pulse survey show that trust and satisfaction in the University of Surrey Students' Union is at an all-time low. Many students do not believe that the USSU represents their academic interests. This is a direct result of the low turnout seen in recent USSU elections. In addition to this the existing E-Voting system cannot be used in small scale AGM elections. This combined with the Covid-19 pandemic meant that AGMs, which are normally carried out in person, were conducted virtually using Surveys in Microsoft Teams (an insecure system) using Surveys in Microsoft Teams (an insecure system). To solve these problems this project details the, design, implementation and testing of a new E-Voting System (a web application) which will replace the existing E-Voting solution used by the USSU. The hope is that the new system will boost student trust and participation in elections, making the USSU more representative.

Additionally, the rise in distributed ledger technology and its properties make it ideal for implementing a secure E-Voting system. More specifically, blockchain, a technology traditionally associated with cryptocurrencies, can be used to implement a secure E-Voting system. Therefore, the E-Voting solution implemented in this project leverages private blockchain technology, which differs from the traditional approaches to E-Voting systems which rely on centralised databases and public blockchains.

## Table of Contents

---

<b>Declaration of Originality</b> .....	<b>2</b>
<b>Acknowledgments</b> .....	<b>3</b>
<b>Abstract</b> .....	<b>4</b>
<b>Statement of Ethics</b> .....	<b>9</b>
<b>Chapter 1: Introduction</b> .....	<b>13</b>
<b>2.1 Project Overview</b> .....	<b>14</b>
<b>2.2 Background</b> .....	<b>14</b>
<b>2.3 Project Motivation</b> .....	<b>16</b>
<b>2.4 Project Aims &amp; Objectives</b> .....	<b>17</b>
<b>2.5 Success Criteria</b> .....	<b>18</b>
<b>2.6 Methodology</b> .....	<b>18</b>
2.6.1 Chosen Methodology.....	21
<b>2.7 Project Plan</b> .....	<b>21</b>
<b>2.8 Structure of report</b> .....	<b>21</b>
<b>Chapter 2: Research</b> .....	<b>23</b>
<b>Literature Review</b> .....	<b>23</b>
<b>3.1 What is E-voting?</b> .....	<b>23</b>
<b>3.2 The security problems with E-Voting</b> .....	<b>25</b>
<b>3.3 What makes a good E-Voting system?</b> .....	<b>26</b>
3.3.1 Evaluation .....	28
<b>3.4 Types of E-Voting System</b> .....	<b>29</b>
3.4.1 Centralised Database with Cryptographic Techniques .....	29
3.4.2 Blockchain.....	31
3.4.3 Selecting an Approach - Blockchain .....	33
<b>3.5 Permissionless and Permission Blockchain</b> .....	<b>33</b>
3.5.1 Permissionless Blockchains.....	35
3.5.2 Permissioned .....	36
3.5.3 Evaluation .....	36

3.6	Conclusion .....	38
<b>Technology Review.....</b>		<b>39</b>
3.7	<b>Native, Hybrid, Progressive and Cross platform .....</b>	<b>39</b>
3.7.1	Native.....	39
3.7.2	Hybrid .....	40
3.7.3	Cross-Platform .....	40
3.7.4	Evaluation .....	41
3.7.5	Selecting an Approach (PWA) .....	41
<b>Chapter 3: Problem Analysis .....</b>		<b>42</b>
4.1	<b>Interviews, Focus groups and Surveys .....</b>	<b>42</b>
4.2	<b>Interviews .....</b>	<b>42</b>
4.2.2	Interview Analysis .....	43
4.3	<b>Focus Groups.....</b>	<b>44</b>
4.3.1	Focus Group Ethical Consideration .....	45
4.3.2	Focus Group Results .....	45
4.4	<b>Surveys .....</b>	<b>45</b>
4.4.1	Survey Objectives.....	46
4.4.2	Survey Target Audience .....	46
4.4.3	Survey Tools and Distribution .....	46
4.4.4	Survey Design.....	47
4.4.5	Survey Ethical Consideration .....	47
4.4.6	Survey Response Analysis .....	48
4.5	<b>4.1.4 The Current System .....</b>	<b>52</b>
4.5.1	Mi-Voice.....	52
4.5.2	AGM Election System.....	54
<b>Chapter 4: Requirements .....</b>		<b>59</b>
5.1	<b>Overall Description.....</b>	<b>59</b>
5.1.1	Product Perspective and Scope .....	59
5.1.2	Product Features.....	60
5.1.3	User Classes and Characteristics.....	61
5.1.4	Operating Environment .....	62
5.1.5	Constraints.....	62
5.1.6	Assumptions and Dependencies.....	62

<b>5.2</b>	<b>System Requirements .....</b>	<b>62</b>
5.2.1	Functional Requirements.....	63
5.2.2	Non-Functional Requirements.....	65
<b>Chapter 5: Design .....</b>		<b>66</b>
<b>6.1</b>	<b>System Architecture .....</b>	<b>66</b>
<b>6.2</b>	<b>React Web Application Design.....</b>	<b>67</b>
6.2.1	Use Case Diagram .....	68
6.2.2	Activity diagrams .....	72
6.2.3	Sequence diagrams.....	75
6.2.4	User Interface Design.....	76
<b>6.3</b>	<b>Node.js Server Design .....</b>	<b>80</b>
6.3.1	API Design .....	81
<b>6.4</b>	<b>IBM Blockchain Cloud Design .....</b>	<b>83</b>
6.4.1	Database Design .....	84
6.4.2	Smart Contract Design .....	88
<b>6.5</b>	<b>Design Challenges .....</b>	<b>88</b>
<b>Chapter 6: Security Analysis.....</b>		<b>89</b>
<b>7.1</b>	<b>Introduction .....</b>	<b>89</b>
<b>7.2</b>	<b>Blockchain Security .....</b>	<b>90</b>
7.2.1	Public Key Infrastructure .....	90
<b>7.3</b>	<b>Risk Assessment .....</b>	<b>91</b>
7.3.1	Risk Identification .....	91
<b>7.4</b>	<b>Risk Analysis.....</b>	<b>94</b>
<b>7.5</b>	<b>Risk treatment .....</b>	<b>96</b>
<b>7.6</b>	<b>Security Design.....</b>	<b>97</b>
7.6.1	Authentication .....	97
7.6.2	Authorisation .....	98
7.6.3	Web Tokens .....	99
<b>Chapter 7: Implementation .....</b>		<b>100</b>
8.1.1	Repository Overview.....	100
<b>8.2</b>	<b>Important Aspects.....</b>	<b>101</b>

<b>8.3</b>	<b>Deployment .....</b>	<b>103</b>
<b>8.4</b>	<b>Technical Challenges .....</b>	<b>103</b>
<b>Chapter 8: System Testing.....</b>		<b>105</b>
<b>9.1</b>	<b>Unit Testing.....</b>	<b>105</b>
9.1.1	Node.js Sever .....	106
9.1.2	Chaincode (Smart Contracts) .....	108
<b>9.2</b>	<b>Cross-Device Testing .....</b>	<b>110</b>
<b>9.3</b>	<b>Requirements Testing .....</b>	<b>111</b>
9.3.1	Functional Requirements Test Matrix.....	111
9.3.2	Non-Functional Requirements Test Matrix.....	114
9.3.3	Validation Testing .....	115
9.3.4	Security Testing.....	116
<b>9.4</b>	<b>User Acceptance Testing .....</b>	<b>118</b>
9.4.1	AGM Election Summary.....	118
9.4.2	UAT Findings .....	119
9.4.3	UAT Suggested Improvements .....	120
<b>Chapter 9: Evaluation &amp; Conclusion .....</b>		<b>121</b>
<b>10.1</b>	<b>Introduction .....</b>	<b>121</b>
<b>10.2</b>	<b>Evaluation of project against objectives .....</b>	<b>121</b>
<b>10.3</b>	<b>Evaluation of E-Voting System.....</b>	<b>124</b>
<b>10.4</b>	<b>Future Work.....</b>	<b>126</b>
<b>10.5</b>	<b>Conclusion .....</b>	<b>127</b>
<b>References.....</b>		<b>128</b>
<b>Appendices .....</b>		<b>132</b>
<b>12.1</b>	<b>Appendix A: Consent form .....</b>	<b>132</b>
<b>12.2</b>	<b>Appendix B: Gantt Chart.....</b>	<b>133</b>
<b>12.3</b>	<b>Appendix C: E-Voting System Survey .....</b>	<b>133</b>
<b>12.4</b>	<b>Appendix D: E-Voting System Survey.....</b>	<b>135</b>
<b>12.5</b>	<b>Appendix E UI Validation.....</b>	<b>144</b>



## Statement of Ethics

---

The project was carried out following the legal, ethical, social and professional standards outlined in the British Computing Society Code of Conduct (BCOC), the Computer Misuse Act (1990), the Data Protection Act (2018) and the Copyright, Designs and Patents Act (1988). These standards were upheld in all stages of the project, that is: research, design, implementation, and testing.

### **Do No Harm**

The E-Voting system's primary users are University of Surrey students and Student Union Staff members, who oversee elections. Students and staff members from other universities will need to be considered, as the E-Voting System will be scaled up to include other universities across the country in further iterations of the E-Voting System (beyond this project). Therefore, the welfare of students, staff and the wider higher education community was protected throughout this project.

### **Data Confidentiality**

The E-Voting system will need to collect, manage and process personal and sensitive data from students and staff. This data includes student identification numbers and emails, which the system uses to determine eligible voters and track how students voted. To adhere to the Data Protection Act (2018), Microsoft Office 365 APIs were leveraged to check eligible voters. This ensured that the data collected is accurate and up-to-date with university records.

A private Blockchain was used to implement the system and ensure that the data was processed and stored securely. This means only valid data controllers that have access to the nodes within the network will access the personal and sensitive data of users. Furthermore, valid data controllers who will have full access to the system's data include myself (for development purposes) and staff members responsible for overseeing the Students Union elections. More information about the security controls that were implemented to protect user data can be found in **Chapter 6: Security Analysis** of this report.

The Survey conducted in the Problem Analysis section of this report also requires the collection of personal and sensitive data. This data includes student: emails, ethnicity, gender, year of study, course and views on the existing E-Voting System. Emails were collected to ensure that respondents were students who attend the University of Surrey. Respondent's ethnicity and gender, which is considered sensitive data, was collected

to identify any biases in the survey's results. A justification of why each piece of data in the survey was collected can be found in this report's Problem Analysis section (under ethical considerations). Additionally, several things were done to ensure the Survey adhered to the Data Protection Act (2018). Firstly, before completing the survey students were asked for their consent to use their responses in this report. Secondly, the data collected was only used in this report (as stated) and respondents were under no obligation to complete the Survey once they had begun it. Thirdly, student responses were aggregated and kept anonymous to ensure that their views were kept confidential. Fourthly, the responses were stored safely in a password protected Microsoft Teams account that can only be accessed by myself. This also means that the data was kept within the EEA, in line with the Data Protection Act (2018). Lastly, once this project was completed, the survey responses were deleted to ensure that the data was not kept longer than necessary.

### **Informed Consent**

In line with the Data Protection Act (2018), focus group and interview participants were asked to sign a consent form (see appendix A), and their responses were kept anonymous. The Computer Misuse Act (1990) is not relevant to developing the E-Voting system in this project because there is no 'ethical hacking' or 'accessing of unauthorised material'. However, the Computer Misuse Act (1990) provided a framework in which security threats to the E-Voting system could be assessed (see **Chapter 6: Security Analysis**).

To ensure that the E-Voting system is ethical, users must know what personal data is being used by the system, for example the system stores how they voted in the election. Therefore, before a user can use the E-Voting system, they must agree to the application's terms of use. Without agreeing to the terms of use, the user will not be able to use the E-Voting system.

### **Social Responsibility**

The E-Voting system in this project was developed to improve the security of the existing system, the student's voting experience, and the convenience of running student elections at the University of Surrey. An E-Voting System with a better voting experience is likely to boost student turnout, making the USSU more representative of student views. Furthermore, improved security will also increase the trust that students have in the democratic processes at the USSU. The development of this system also raises some equality issues.

Due to time constraints, the final version of E-Voting System produced in this project does not cater to students who have a disability that may inhibit them from using the application. This is a huge problem that

will be addressed in future iterations of the system, because disabled students being unable to vote in USSU elections violates their human rights. The Evaluation Section outlines ways in which accessibility features could be implemented in future iterations of the system.

Another issue is that the time gained from using the E-Voting System could make staff members at the Students Union redundant. Currently, three members of staff at the USSU oversee student elections, the new E-Voting system could mean a reduction in the human resources required to oversee the election or could reduce the workload (and salary) of all three staff members. Alas, the benefits of the E-Voting System's welfare on students and staff outweighs the possibility of job losses. This is because it is unlikely that staff members will be made redundant due to adopting a better E-Voting system. A more likely outcome is that the time gain will allow staff to focus on the Student Union's other priorities. In addition, because the new E-Voting system will require some technical expertise to manage, staff members may be given extra education and training, which will improve their human capital.

## **Conduct**

The principles outlined in the BCS's Code of Conduct (BCOC) were used to maintain professionalism throughout this project. The first principle of the BCOC: make IT for everyone, is demonstrated throughout this project. This is because the project's main aim is to leverage technology to make the democratic processes at the USSU better for student and staff. To add, the majority of the research conducted throughout this project can be from reputable sources (books, journals, research papers and other advanced material), ensuring the integrity of the information collected. In addition, all sources were referenced appropriately using IEEE standards.

Learning about Blockchain technology and the other technologies to implement the E-Voting system adheres to the second principle of the BCOC: show what you know and learn what you don't. In addition, preparation time prior to the interviews with staff members was done to ensure that USSU staff members' time was not wasted.

Additionally, the question asked in the Survey and focused groups were respectful and appreciative. Both of the points align with the BCOC, which states that developers should respect the organisations and individuals they work with. Last, to ensure that the project was managed professionally, a realistic Gantt chart was produced, and backup copies of this report were maintained on the cloud (see ***Project Plan in Chapter 1***).

## **Intellectual Property and Licencing**

Intellectual Property in the UK is governed by the Copyright, Designs and Patents Act (1988). Furthermore, intellectual property has been defined as property that relates to the creation of the mind and can only exist if it is an artefact. The E-Voting system developed in this project is a new IP. As a result, a decision on licensing will have to be agreed with the Student Union. The most pragmatic agreement, which was reached with the USSU, is that in a future iteration of the E-Voting System management of the blockchain nodes used by the USSU will be handed over. The USSU will also have ownership of the web application produced (free of charge). However, the background technology's license (the design and architecture of the system) and the overall blockchain will remain mine. This will allow for work on the E-Voting to continue, with the possibility of expanding to other universities.

Many things were done to ensure that the agreement with the USSU was fair and ethical. Firstly, the USSU will be expected to host the web application on a domain they own after this project is completed. This ensures the full ownership of the web application is given to the USSU, preventing the possibility of the web application being shut down by myself if a dispute arises. Secondly, the web application will have no obfuscation, ensuring that other developers can read the code and continue development if the USSU no longer wishes to work with me. Furthermore, it was decided that the system would be made open source in a future iteration. This will allow other developers to highlight and fix vulnerabilities.

# Chapter 1: Introduction

---

The main objective of this project is to produce a secure Electronic Voting (E-voting) system that can be used at student elections at the University of Surrey.

The University of Surrey Student's Union is responsible for the administration of all student elections at the University of Surrey. Therefore, they will be the primary stakeholder of this project. The second stakeholder will be students at the University of Surrey, as they will be the primary users of the proposed solution. As a result, members of staff responsible for the administration of elections at the Students' Union as well as students who attend the University of Surrey will be interviewed and surveyed throughout the project to ensure the solution produced is suitable for the end-users.

To ensure an optimal solution is achieved, this report will discuss and moreover, justify the incorporation of highly sophisticated technologies, such as blockchain that can be used in the proposed solution. In addition, analysis of the Students' Union's existing E-voting system will be conducted in order to find security flaws and features that can be improved. Once stakeholders are satisfied with the proposed solution, a requirements specification of a tangible technical solution will be proposed to address issues discovered within the existing E-voting system. The proposed solution will then be designed, implemented and evaluated against the project's aims and objectives.

## 1.1 Project Overview

This chapter will provide background information into the projects two stakeholders: University of Surrey students and the University of Surrey's Students' Union (USSU). This section will also provide a brief explanation of the types of student elections held by the USSU and how they are conducted. Lastly, the project's aims, objectives and success criteria will also be specified in this chapter.

## 1.2 Background

Most UK universities will have an associated Students' Union. Most Student Unions are registered to the UK charity commission (the UK's charity regulator) and must comply with its rules and regulations. Similar to other Students' Union in the UK, the University of Surrey's Students' Union (also referred to as USSU) is a student-led charitable organisation.

The USSU exist to represent the interests of students at the University of Surrey to Senior University management. The USSU is led by five full-time student Sabbatical Officers, who are elected by Surrey University students each year. The annual election, known as Surrey Decides, occurs every year in the month of February.

The elected full-time positions include a President and four Vice-Presidents. Within the USSU, there exists four different zones: voice, activity, support and community. Each zone focuses on a different aspect of student life and is overseen by one of the four elected Vice-Presidents. The USSU also has 21 elected student part-time officers (PTOs), five in each zone. One PTO is the Union Chair who is responsible for holding all elected officers to account.



*Figure 1: The University of Surrey Students' Union Building*

## Students' Union Officer Team 2020/21

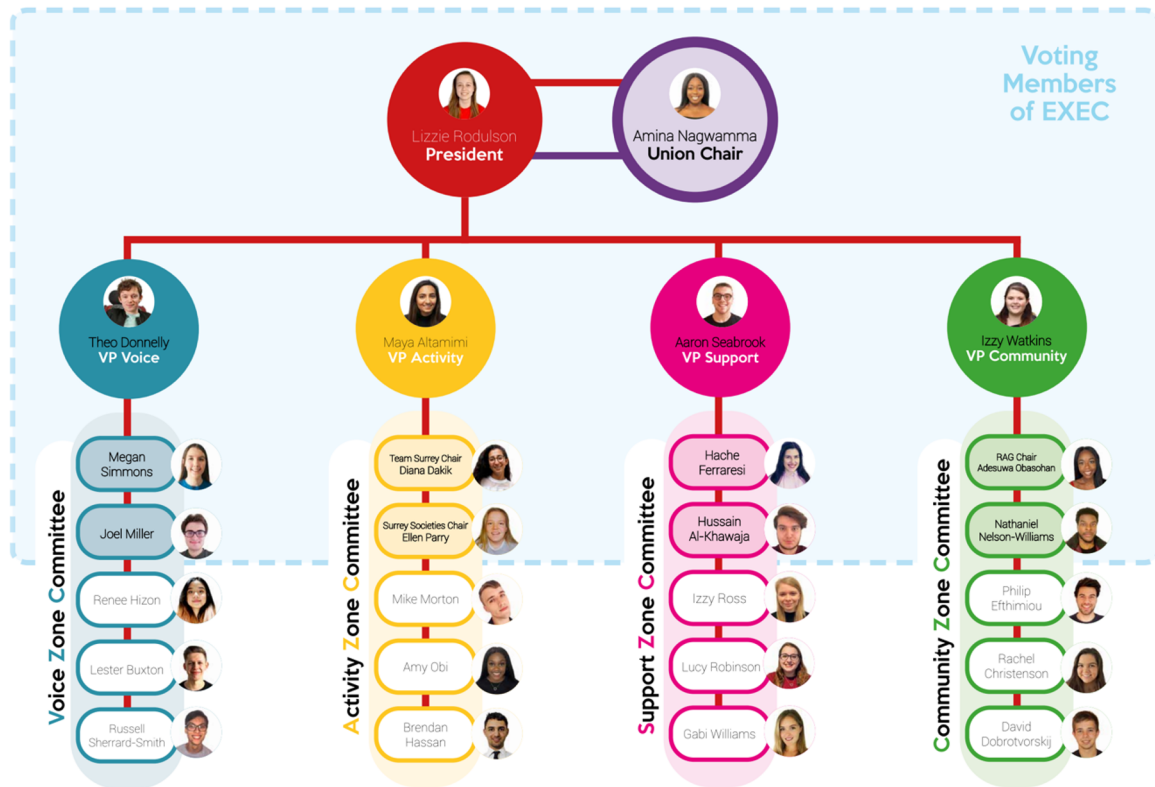


Figure 2: Structure of Surrey Students' Union and Elected Officers for 2020/21

The Voice Zone's primary responsibility is to uphold the election processes of student elections at the University. Therefore, attempts will be made to regularly engage the VP Voice and Voice Zone Committee members during the gathering of requirements, design, development and testing of the proposed solution. Feedback from other officers and students will also be welcomed and sought out.

The USSU also employ non-elected full-time staff members. The staff members of relevance to this project include the Digital Experience Manager (Alex Harden), who is responsible for the day-to-day operations of the USSU, and the Voice Zone manager (Katherine Phillips), who is responsible for overseeing the democratic processes of the USSU. Feedback from the Digital Experience Manager and Voice Zone manager will be sought out and welcomed throughout this project.

The USSU is also responsible for the running of Course Representative elections and society AGMs. Course Representative elections are held during the start of the academic year; thus, it would be impractical to test the proposed solution during these elections.

Society AGMs are the formal elections that take place within a university society. Society AGMs are held from the start of April till May hence, a working version of the proposed solution could be trialled during these elections. The societies that could be interested in trialling the proposed solution include ACS, HackSurrey, WOKESurrey and CompSoc. The current committee of these societies and other societies that wish to use the proposed solution to run their AGM will be consulted throughout the development of the solution.

## 1.3 Project Motivation

In recent years student engagement and satisfaction at the University of Surrey has fallen significantly.

The Office for Student's, National Student Survey (NSS), a survey completed by students in their final year of University, shows that the University of Surrey is now ranked 126, down from rank 20 in 2016. In four years, the University of Surrey has fallen 106 places. A breakdown of the 2020 NSS results shows that 42.37% of Surrey students believe the Students' Union does not effectively represent student academic interests [1].

Turnout for Surrey Decides, the annual Student Union election, has also fallen significantly in recent years. In the 2020 Surrey Decides election, the most recent election, only 8% of Surrey students voted [2]. Student elections held in the same year at the London School of Economics and Royal Holloway had a turnout of 16% [3] and 30.6% [4] respectively. This data shows that at other similar UK Universities, turnout in student elections is more than double when compared to recent elections held at the University of Surrey. Robust voter turnout is fundamental to a healthy democracy, as low turnout is usually attributed to political disengagement and the belief that voting for one candidate/party or another will do little [5]. Therefore, voting in Surrey Decides is a crucial indicator of student engagement with the Students' Union, because students who are engaged with the USSU will want to influence its governance.

Many reasons could explain why student turnout and engagement with the USSU has fallen significantly. The primary reason that relates to this project is that students do not trust or have faith in the current election process. Evidence of this exists in the pulse survey conducted by Alterline on behalf of the USSU between 2016 to 2020. The pulse survey showed that 68% of students do not think they 'can influence decisions made by the SU' [6]. The same survey also revealed that 68% of students do not know 'who the USSU Sabbatical Officer team are' [6]. The fact that the majority of students do not know who the Sabbatical Officers are explains why 42.37% [1] of Surrey students believe the USSU does not effectively represent student interests.



## Students' perceptions of their ability to influence the SU

Proportion of students who selected 'agree' or 'strongly agree' in response to 'I can influence decisions made by the SU', by academic year, Pulse period and month. Answers exclude students who selected 'I don't know'.

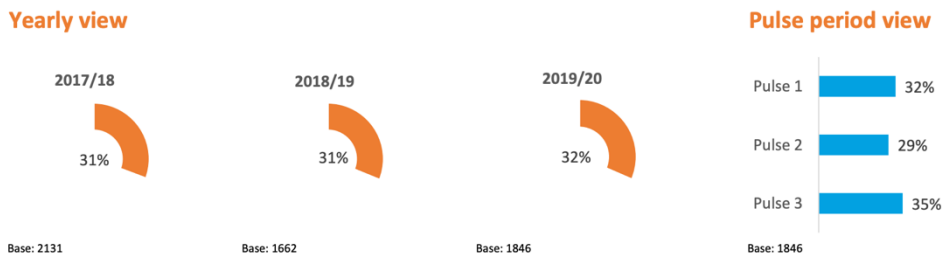


Figure 3: Data from the Pulse Report 2019/20 showing student apathy

Students' selecting yes to 'Please provide a 'yes' or 'no' to the following statements' this Pulse period and by month.

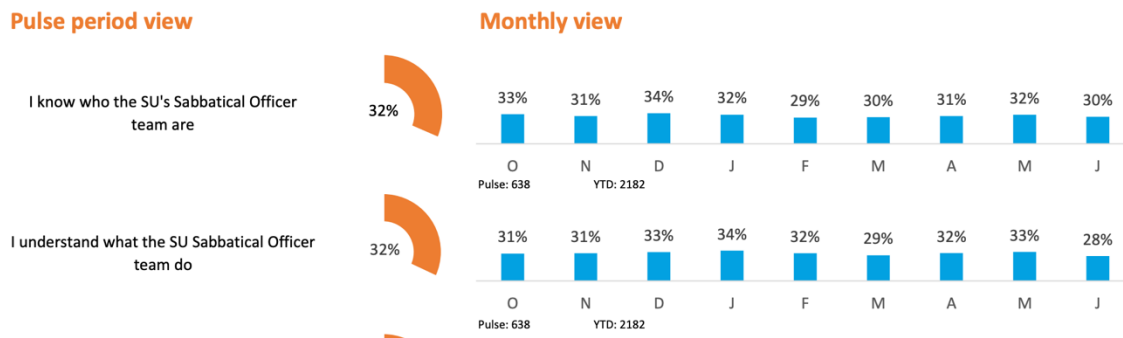


Figure 4: Data from the Pulse Report 2019/20 showing lack of student awareness and engagement

Therefore, as an individual who was an elected member of the USSU in 2018, the main motivation behind this project is to improve the democratic processes, outlined above, at the University of Surrey. The hope is that by improving these democratic processes, the USSU will restore the trust of students, leading to a better NSS score for the University of Surrey. Additionally, as this is a software engineering project the secondary motivation, is to acquire key software development and programming skills that can be used in the real world.

## 1.4 Project Aims & Objectives

The primary aim of this project is to produce a new secure e-voting system that can be used to conduct AGM elections at the University of Surrey.

The project objectives that will allow me to achieve this aim are as follows:

- Explore and understand the attributes and vulnerabilities of an effective E-Voting system.

- Review technologies and frameworks that can be used to implement an E-Voting system that functions on IOS and Android devices.
- Gather Requirements through the use of stakeholder interviews, surveys and focus groups.
- Analyse and evaluate the current E-Voting system used by the Student Union to conduct AGMs.
- Conduct a risk assessment to identify threats to the proposed E-Voting system and propose controls to mitigate them.
- Iteratively design and implement a new E-Voting system that can be used on IOS and Android.
- Perform unit, functional, non-functional, cross device and user acceptance testing to ensure that the system functions as expected.
- Execute beta testing, by trialling the application in one or more university election to demonstrate the utility of the new system.
- Gather feedback and recommendations from Beta users, that will be implemented in future iterations of the application.

## 1.5 Success Criteria

The success of this project will be evaluated based on the project's aims and objectives outlined above. Additionally, the effectiveness of the produced E-Voting system will be assessed using the attributes that are considered paramount for an E-Voting system to have. These attributes are explained in Section: **2.3 What makes a good E-Voting system?** Furthermore, the project will be considered a success if the produced E-Voting system is able to fulfil the majority of these attributes.

## 1.6 Methodology

A software development lifecycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software [7]. Every SDLC consists of the following stages:

- Software specification, where customers and engineers define the software that is to be produced and the constraints on its operation [8].
- Software development, where the software is designed and programmed [8].
- Software validation, where the software is checked to ensure that it is what the customer requires [8].
- Software evolution, where the software is modified to reflect changing customer and market requirements [8].

It is important that an appropriate SDLC is chosen for the development of the proposed system. This will ensure a well-structured flow of phases that allow me to quickly produce high-quality software which is well-tested and usable. The two most notable SDLCs are, the waterfall model and agile development.

The waterfall model is split into the following phases:

**Requirement's analysis** – in this stage, the requirements for the system are identified and a requirement specification is produced.

**Design** – in this stage, how the system will function will be layout. Design diagrams and UI mock-ups are often produced during this stage.

**Implementation and Testing** – in this stage the system is developed and tested. Testing documentation is often used to keep track of the working functionalities.

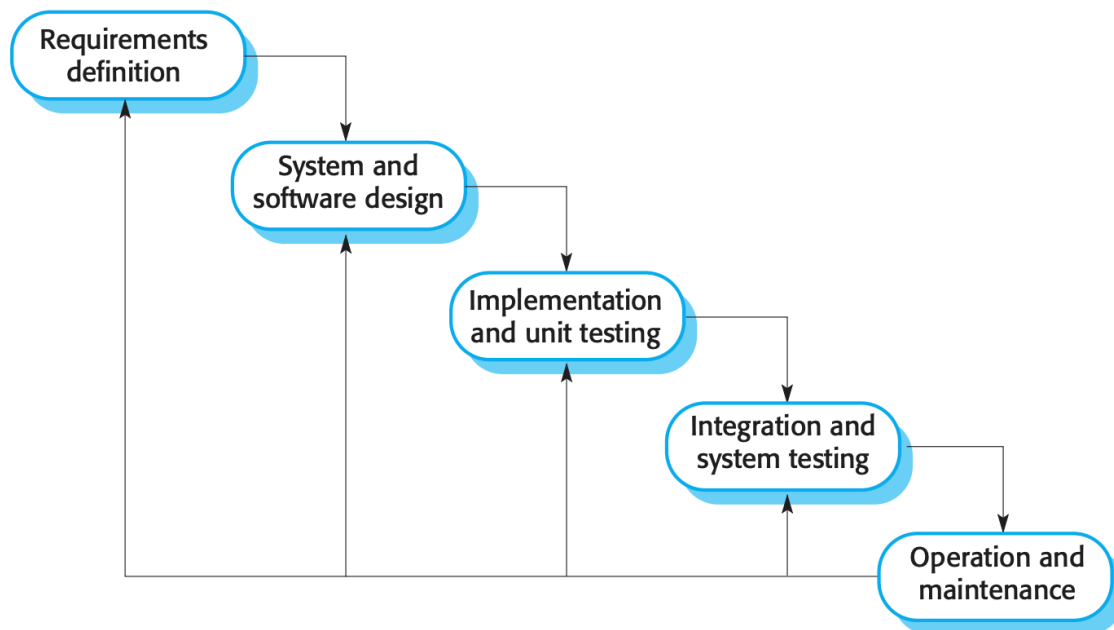


Figure 5: The waterfall model

The advantage of using the waterfall model is that its linear step-by-step well-defined phases make it simple to follow. However, the main disadvantage of the waterfall model is that it is rigid [9]. If a new requirement is recommended by stakeholders in the development phase it will not be possible to add the new feature to the requirement specification. This is because a change in the requirement specification will also require a change in the design documentation of the proposed system. This inflexibility to make changes to requirements in an ad-hoc manor will mean that stakeholder feedback will be limited.

Agile development (also known as incremental development) is based on the idea of developing an initial implementation, getting feedback from users and others, and evolving the software through several versions until the required system has been developed [10]. An agile approach to incremental development means that early increments are identified but the development of later increments depends on stakeholder feedback.

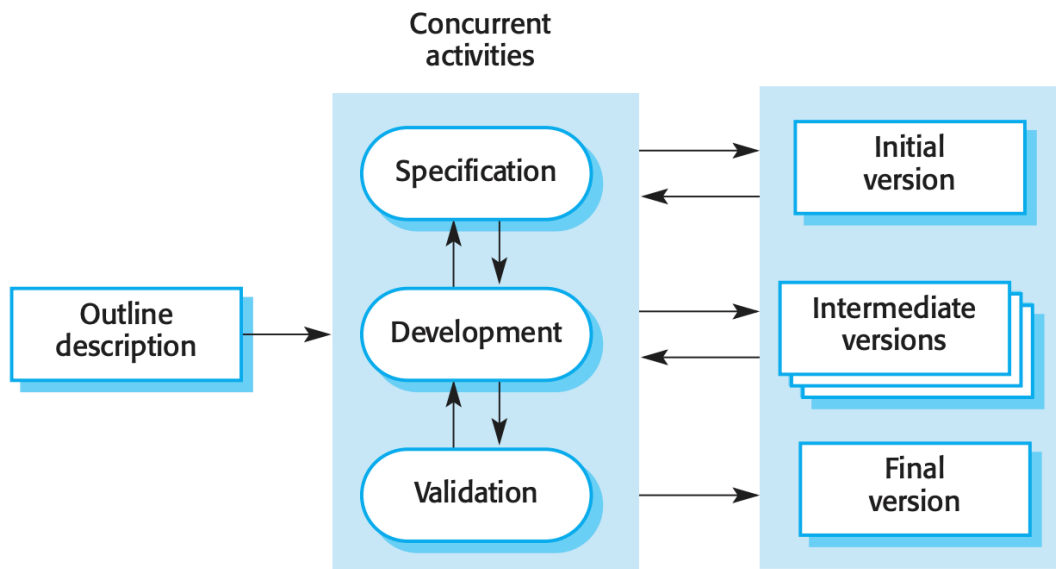


Figure 6: Incremental development

The main advantage of incremental development is it will reduce the amount of documentation that has to be redone if new requirements are recommended by stakeholders [10]. Additionally, using the agile model will allow for stakeholder feedback to be incorporated into the proposed system at the next iteration of the application. To add on, the agile approach to development has become widely used in the software development industry, thus using it in this project will facilitate the development of skills that can be used in the real world.

## 1.6.1 Chosen Methodology

On balance, the best SDLC for this project is the agile approach. Although it is a more complex mythology, as stated by Ian Sommerville, “agile development methods are better than the waterfall for systems whose requirements are likely to change during the development process” [10]. As stated previously the development of the proposed system will be influenced heavily by stakeholder feedback. Therefore, using the agile approach will ensure that stakeholder feedback is incorporated into the application during the development process. In addition, the time constraints of the project mean that the core functionalities can be implemented in the first iterations of the system. Additionally, desirable but non-essential features can be implemented in later iterations of the system if time permits.

## 1.7 Project Plan

This project will span from the 21/09/20 (the start of the academic year) until the 18/05/21 (the deadline of the project). All the tasks and milestones of this project have been mapped out in the Gantt chart displayed in **Appendix B**. However, it is important to note that the timeframes and durations outlined in the Gantt chart are estimations. Therefore, while an effort will be made to stick to these dates, the actual completion dates are likely to deviate from the intended completion dates.

## 1.8 Structure of report

The project report will be split into the following chapters:

**Chapter 2: Research (Literature and Technology Review)** – This section includes a summary, analysis and evaluation of all the initial research conducted in this report. The first half of the chapter, the literature review, focuses on E-Voting and Blockchain literature. The latter half, the technology review focuses on the technologies and frameworks that can be used to implemented the E-Voting System.

**Chapter 3: Problem Analysis** – This chapter will provide insight into the problems facing the USSU and Surrey Students when using the current E-Voting system. This will be done through stakeholder interviews, focus groups and a survey. Additionally, an evaluation of the system currently used by the USSU will be conducted.

**Chapter 4: Requirements** – This chapter will outline what the final version of the E-Voting system produced in this project is expected to do.

**Chapter 5: Design** – This chapter will contain the design documentation that will describe how the final version of the E-Voting System produced in this project will function.

**Chapter 6: Security Analysis** – This chapter will present the risk assessment conducted on the E-Voting system as well as the necessary security controls that will be implemented to ensure the E-Voting system is not susceptible to attacks.

**Chapter 7: Implementation** – This chapter will provide an overview of the code base of the application and briefly discuss how the application was deployed.

**Chapter 8: System Testing** – This chapter will document the results from unit, security, functional, cross device and user acceptance tests. As part of user acceptance testing the outcomes and results of the E-Voting application on a real AGM election, and will discuss the improvements and recommendations made by participants.

**Chapter 9: Evaluation & Conclusion** – This section will evaluate the overall success of the project based on the project objectives outlined above. Future enhancements to the system will also be discussed in this section. Finally, this section will include a conclusion summarising this project.

## Chapter 2: Research

---

This chapter contains a summary, analysis and evaluation of all the research conducted in this project. The research in this chapter is split into two sections, the Literature Review and the Technology Review. The literature review discusses what an E-Voting system is, the security problems associated with E-Voting, what makes a sound E-Voting System, the types of E-Voting Systems, and permissionless and permissioned blockchains. The Technology Review discusses the technologies and frameworks that can be used to develop the new E-Voting System. Throughout this chapter, justifications are provided when an approach is selected over an alternative.

### Literature Review

---

In this section, relevant E-Voting literature will be analysed and evaluated in order to select the best approach for the proposed E-Voting System. Additionally, when one approach is chosen over another, a suitable justification will be provided.

#### 2.1 What is E-voting?

From hand raising to physically moving to one side of an auditorium, since the inception of democracy, voting has been conducted in a multitude of ways. Today, voting via a secret ballot is the most common method of voting. This method of voting guarantees the two pillars of a free and fair election, anonymity and trust [11]. Anonymity ensures that an individual's vote cannot be discovered by someone else. This prevents individuals from being coerced to vote in a particular way. Further, the UK's voting system upholds anonymity by discarding ballots that contain anything that identifies the caster, such as a signature. Additionally, the banning of cameras and phones in polling stations prevents people from being able to prove how they voted. The second pillar, trust, ensure that people have faith in the voting process and will respect the outcome [12]. For there to be trust, the voting process must be transparent and understood by everyone. In the UK, trust in the voting process is upheld through the use of, counting observers from both opposing parties, and the electoral commission – a transparent and impartial body that oversees UK elections.

An Electronic voting (E-voting) system is a voting system in which the election data is recorded, stored and processed primarily as digital information [13]. Instead of crossing an 'X' on a piece of paper, an electronic voting machine, home computer, mobile device or telephone can be used to cast a vote. Compared to conventional voting systems, E-Voting is quicker, less expensive and generally more convenient for voters and authorities. Further, there are two types of E-Voting Systems:

- Voting Machines – This type of voting is similar to ballot voting in that it requires voters to go to a polling station to cast their vote. However, instead of crossing an X on a piece of paper, voters push a button on a voting machine [13].
- I-Voting – This type of voting allows people to vote remotely using an application on a device that is connected to the internet [13], as shown in **Figure 7**. This is the kind of System that will be developed in this project.

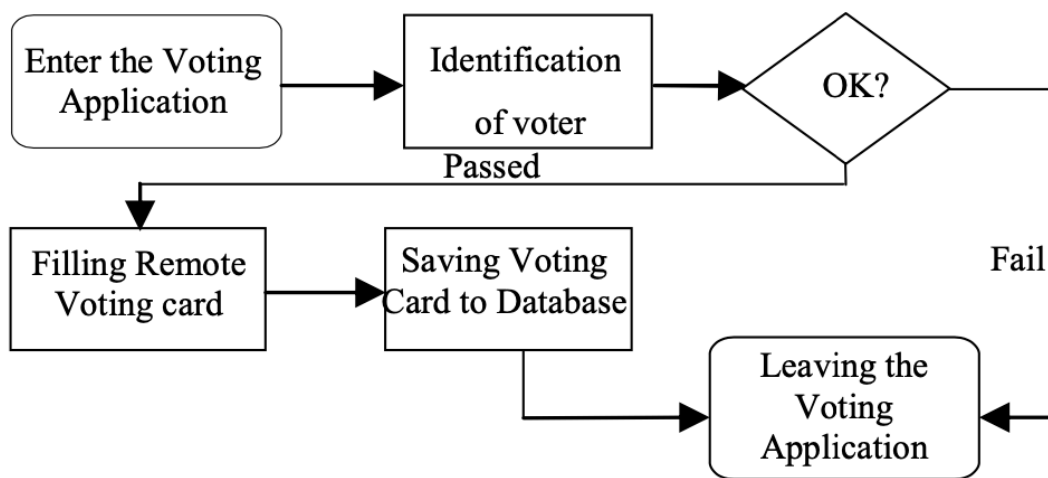


Figure 7: High-level view of an I-Voting System

In line with the guidance given by The Organization for the Advancement of Structured Information Standards (OASOS), the process of E-Voting can be split into three sequential phases.

The Pre-Voting phase – includes the candidate nomination process, in which candidates put themselves forward to stand for the election [14]. The nomination process can take many forms, for example, USSU bylaws state that 'candidates require the nomination of 5 other students' to stand in the election [15]. Candidates who fail to do this will not appear on the ballot. Additionally, this phase also includes the Voter Registration process, in which eligible individuals will have to register to vote before polling day [14]. In line with USSU bylaws, all Surrey University students are automatically registered to vote [15]. At the end of this phase, there should be a list of all eligible candidates and voters.

The Voting phase – during this phase, all eligible voters will be able to vote for eligible candidates, using an electronic channel such as a smartphone [14]. The new E-Voting system will allow students to vote using their smartphone or computer.



The Post-Voting phase – this is the most important phase as it covers counting the votes and publishing the results [14]. The way in which votes are counted must be clear and transparent. The E-Voting system should also allow recounting so that administrators can reproduce the same results if the election results are challenged. In addition, this phase also includes an analysis of the election in the form of a report. This analysis can range from voter turnout to a transcript detailing how votes were counted (this can be especially crucial when complex voting systems like STV are used).

## 2.2 The security problems with E-Voting

While E-Voting can improve the efficiency and convenience of an election, there many security problems that arise when using an E-Voting system:

**Denial of Service Attack (DoS)** – The purpose of this attack is to overwhelm the system by flooding the E-Voting system with packets until it can no longer accept legitimate connections [16]. This type of attack could have a particularly high impact, given the short timescales of USSU elections (3 days) [17]. It is unlikely that this attack will be effectively carried out by a student with a stake in the election. This is because, denial of service attack requires advance knowledge of networks and specialist software, which most students do not possess. Having said this, the Student Union has 17,000 eligible voters, and while it is highly unlikely that students will all vote at the same time, the E-voting system should be able to handle a substantial volume of requests.

**Viruses & Worms** – Viruses and Worms are programs that can reproduce and cause harm to computers. The difference between them is that worms do not need to change existing programs to spread, while viruses do [18]. If a student has a virus or worm on their device, it should not be possible for them to overwrite files and change election results. Indeed, the virus might change the vote of the individual student but the overall integrity of the election should be protected. According to a study conducted by Sophos, 48% of UK organisations were hit by a malicious attack in 2020 [19], so ensuring that security controls are put in place to counter this problem is paramount.

**Internal Attacks** – Attacks by people with privileged access to the System, for example, system developers or system administrators could alter votes [17]. But it is safe to assume that the Students' Unions election officer would not be able to favour one candidate over another. This is due to, the election officer being accountable to the democracy committee which ensures that the power election officer is checked and balanced. Additionally, the Students' Union will have multiple administrators with privileged access, making

this kind of attack more difficult. Lastly, the System could restrict administrators to change votes or allocate votes to candidates which will significantly reduce their ability to influence the election results.

Another problem with using an E-voting system is that trust and anonymity cannot be guaranteed. E-Voting systems are complex, which means that stakeholders in an election may struggle to trust the integrity of the software being used [12]. One solution to this problem is to publish an anonymised list of how people voted. Each voter would be given a unique ID so that they can confirm that their vote was counted correctly. The issue with this is that in an attempt to make the system more trustworthy, anonymity is sacrificed. For example, a coercer could demand an individual's unique ID in order to check that they voted in a particular way, breaking the anonymity of votes.

While these security flaws present valid threats to E-Voting systems, the E-Voting system proposed in this project is operating in a low threat environment. This is because unlike national elections, Students' Union elections are not high stake. Furthermore, it is unlikely that professional hackers will target a University of Surrey election; therefore, we can create the E-Voting system on the bases that all attacks, if any, will be low level. This does not mean security will be ignored before development begins, a security analysis of the proposed design will be carried out, and the necessary controls will be implemented.

## 2.3 What makes a good E-Voting system?

There have been a number of proposals for electronic voting systems. Across all these proposals there is a set of desirable attributes that make an E-Voting system secure and effective. These attributes include:

**Accuracy** – For an E-Voting system to be considered accurate, three conditions must be met. Firstly, it must not be possible to alter or discard a vote during its transmission or during the counting of votes. Secondly, it must be impossible for invalid votes to be counted [20]. Thirdly, votes from ineligible voters should not be counted [20]. This is the most important attribute of an E-Voting system, without accuracy, the election results could be brought into question.

**Verifiability** – For an E-Voting system to be considered verifiable voters should be able to independently check that their vote was recorded correctly [20]. This can be done by publishing an anonymised list of how people voted so that voters can confirm that their vote was counted correctly. Additionally, administrators should also be able to check that the System has not been compromised, this can be achieved through the use of a checksum.

**Democracy** – An E-Voting system is considered democratic if it permits only eligible voters to vote, only once [20]. This is achieved through a combination of verifiability and accuracy. Further, the System could authenticate students by using Microsoft's Office 365 login (their outlook login) which uses two-step verification.

**Privacy** – An E-Voting system upholds privacy if a third party cannot determine how an individual voted in the election [21]. This prevents voter coercion as individuals have no way of proving how they voted to the coercer. While this is an important attribute, as mentioned previously, it is impossible to have privacy and total verifiability. This is because, for voters to be able to verify their vote, they must be able to check how they voted, which could be shown to the coercer. Having said this, security controls can be implemented to reduce the possibility of coercion. One example is to prevent users from screenshotting their phones while using the E-Voting system.

**Convenience** – An E-Voting system is considered to be convenient if it allows users to quickly and easily vote, with minimal technical skill [20]. A user-friendly and intuitive user Interface will ensure this attribute can be achieved.

**Flexibility** – For flexibility to be upheld, the administrator should be able to change election rules such as eligible voters, a student's ability to change their vote, the start and end date of elections, and the electoral System being used, i.e., switching between FPTP and STV.

**Mobility** – An E-Voting system is considered mobile if there is no restriction on the location from which an eligible voter can vote [13]. This is important as the University of Surrey has 10,000 students that live off-campus, therefore to ensure that every student has the opportunity to vote the proposed System should allow all students with an internet connection to cast a vote.

**Reliability** – An E-Voting system is considered reliable if it continuously performs its function as expected [13]. More specifically the E-Voting system should allow all eligible voters to vote for the duration of the election. Further, the creation of backups could also ensure that important data isn't lost during the election.

**Social Acceptance** – An E-Voting system is considered to have social acceptance if it is perceived as being an effective system by the majority of students [22]. Overlooking this attribute will have dire consequences. This

is because If the majority of students do not like or trust the E-Voting system, they will not use it. For this reason, student feedback will be sought throughout the development of the System.

### 2.3.1 Evaluation

Developing an E-Voting system in such a short time frame means that attributes will have to be prioritised, with the goal of implementing the most important attributes first. It is, therefore, necessary to ranking the attributes above from most important to least important. The hope is that the E-Voting system produced will contain all the attributes listed above. However, the attributes ranked higher will be implemented in earlier sprints. Below are the ranks for each attribute along with a justification for the rank:

1. Democracy (accuracy and verifiability) – This is the most important attribute and is a combination of **accuracy** and **verifiability**. This is because, an E-Voting system is simply a tool that makes democratic elections easier for voters to vote and overseers to manage. Furthermore, if the System cannot accurately count the votes cast, and produce the correct election results, the System is useless. Another reason why this attribute is important is because, if the System does not allow for students to confirm how their vote was counted, trust in the System will fall, reducing turnout. The final reason why this is an important attribute is because, nobody will stand or vote in an election that is not fair. Furthermore, if an election used an E-Voting system that allowed some individuals to vote multiple times, the election would not be considered fair.
2. Reliability – This is ranked second because, without reliability, the System could stop functioning during the election. The E-Voting system going down could prevent students from voting or registering as a candidate before the deadline. This would have devastating consequences on USSU elections as a student only have a short time window to vote (3 days).
3. Semi-Privacy – As mention previously, privacy and verifiability cannot simultaneously be achieved in an E-Voting System. Therefore, the proposed E-Voting System will have full verifiability and semi-privacy. Semi-Privacy is defined as an E-Voting system having security controls that reduce the likelihood of coercion. Ranking third, this is an important attribute because to get a truly authentic election result, students must be able to conceal how they voted.
4. Mobility – This is ranked fourth because if the system didn't allow students outside of campus to vote, over 10,000 students that commute and live off-campus would not have the same equal opportunity to vote.
5. Convenience – This is ranked fifth because students will be more likely to use the E-Voting system if they like its UI, and if the voting process is as easy as possible. The reason why this isn't ranked higher

is the attributes above focus on the functionality of the E-Voting System. Further, if the E-Voting System does not function as intended, nobody will use it irrespective of how good the UI is.

6. Flexibility – This is an attribute that would make the E-Voting System more convenient for administrators who oversee the election. However, it is a desirable attribute, the priority is to develop a system that allows students to vote in student elections using the standard USSU election rules. Giving the E-Voting System more flexibility in the event of rule changes, will be implemented in later sprints of this project, hence its low rank.
7. Social Acceptance – While it is important that students accept the E-Voting system, this attribute was given the lowest rank because if the System meets all the attributes above it will be perceived as an effective system by the majority of students.

## 2.4 Types of E-Voting System

There are a multitude of architectures and technologies that can be used to create an effective E-Voting system. The two most prominent are Blockchain and a centralised database that uses cryptographic techniques. Both of these approaches have advantages and disadvantages. Further, a decision on which approach will be selected for the proposed solution will be determined by weighing up and evaluating the pros and cons of both technologies against the objectives of the project and application.

### 2.4.1 Centralised Database with Cryptographic Techniques

A centralised database is a single location in which information is stored. It can be accessed through the internet or a private LAN connection. Many organisations use this architecture to store their information, this includes, colleges, companies and banks.

There are a multitude of cryptographic techniques that are used to store and share data in a database. These cryptographic techniques prevent adversaries from being able to decipher messages in transmission. They also stop adversaries from getting useful information if they successfully hack into the database.

One of these techniques is encryption which is the process of translating data from understandable plaintext, to ciphertext. This ciphertext cannot be understood by anyone, and it can only be translated back into plaintext with the secret key. There are many types of encryption algorithms each providing their own level of security; however Advanced Encryption Standard (AES) is the algorithm trusted as the standard by the US government [23].

Another cryptographic technique is hashing. A hash is the result of a one-way mathematical hash function that takes in a single value. A hash function is generally considered secure when it is collision resistant meaning that it is computationally infeasible to find 2 values that produce the same hash. SHA-3 is the standard currently specified by Federal Information Processing Standards (FIPS) [23]. Lastly cryptographic techniques also include security protocols used to send messages over a communication channel. These protocols provide a high-level of security across a channel if certain conditions are met. Furthermore, these protocols include: Mixnets, Blind signature, Homomorphic secret sharing schemes and Diffie-Hellman Key Exchange with RSA.

### 2.4.1.1 Application

The best way to demonstrate how a centralised Database with cryptographic techniques can be used to produce a secure E-Voting System is to discuss an example of such a system. The article A Light-Weight e-Voting System with Distributed Trust produced by Aneta Zwierko and Zbigniew Kotulski [24] is a practical example of how this kind of E-Voting System can be implemented.

Below is a high-level overview of the E-Voting system proposed in the article.

The System is comprised of five parties:

The trusted authority – who is responsible for creating a list of users who are eligible to participate in the election and for authentication data that allows them later to vote. This is denoted as TA.

The Mix – who is responsible for protect the voters' privacy. This is denoted as  $TA_0$ .

The Counter – who is responsible for collecting the votes, counting all valid votes and then publishing the votes for verification. This is denoted as AC.

The Voters (users) – the individuals who are eligible to vote in the election. The number of voters is denoted as  $N_u$ .

The voting channel – this is the in which user can use to vote in the election, for example a mobile web application. This is denoted as AV.

The basic steps of the election are:

- TA creates the set of credentials and the list of registered users and sends them to the mix agent AM.
- For each voter requesting credentials, the mix agent AM creating credentials from data received from T A. First, it checks if a user has the right to vote.

- The user sends its vote along with credentials to the counter agent AC: the counter checks the credentials with TA and if they are proper the counteragent AC sums up the vote, publishing a proof attached to the vote.

The scheme makes use of two cryptographic primitives: the secure secret-sharing scheme and the Merkle's puzzles. Details of how these cryptographic primitives function as well as a formal specification for this E-Voting scheme can be found in section 3 and section 5 of the article respectfully [24].

### 2.4.1.2 Evaluation

Centralised databases are the most common form of storage in today's world. However, as the volume of transactions and data has increased the integrity of these centralised systems have been brought into question. As well as having a single point of failure centralised databases rely on an authority to maintain a record of all transactions. The first problem with this is that when multiple records of a transaction are recorded, there is no way of ensuring that all parties have recorded the transaction correctly. For example, when Bear Stearns investment bank failed in 2008 and was completely acquired by JP Morgan Chase, the number of shares offered to the acquirer was larger than the shares outstanding in the books of Bear Stearns [25]. It was not possible to clarify the accounting errors and JP Morgan Chase had to bear the damage from excess (digital) shares [25]. The problem here is that a single point of failure in a centralised database means that an adversary can launch DoS attacks, which will prevent users from being able to communicate with the database. This could have a huge impact on the election as voting is only open for a limited time. The second problem is that a centralised database system gives a small number of institutions a significant amount of power, which can lead to corruption [26]. For example, in the UK Banks have the power to temporarily freeze bank accounts and hand over customer transactions on the behest of the authorities. Additionally, the proposed E-Voting system in A Light-Weight e-Voting System with Distributed Trust [24] also place a significant amount of trust in the hand of a few authorities. For the System to function we assume that, the trusted authority, the mix, and the counter reliably carry out their responsibilities and that they do not collude to influence the election results. In addition to this no cryptographic protocol is full proof if the administrator does not keep important credential secret. For example, if an adversary was able to gain access to an administrator's password, they would be able to compromise the election. This means that administrator will be required to have a level of information security training in order to use the E-Voting system securely.

### 2.4.2 Blockchain

There are two types of blockchain: permissionless and permissioned, their differences are discussed in the proceeding section. In order to effectively compare blockchain with a centralised database no distinction will be made between the two types.

Blockchain is a distributed and immutable ledger that allows tangible and intangible assets to be tracked [26, 27]. The distributed nature of blockchain means that every node in a network has a record of every transaction [27]. The immutable attribute of blockchain is achieved through the use of connected blocks. Each block consists of: all the transactions that occurred when the block was created, a hash which is a unique alphanumeric string, and the hash of the previous block in the chain. What makes this chain of connected blocks tamper resistant is that the hash of a block is determined by the block's transactions. Therefore, if someone was to change the transactions of a block its hash would also change. This would make the next block which contains the previous hash, invalid.

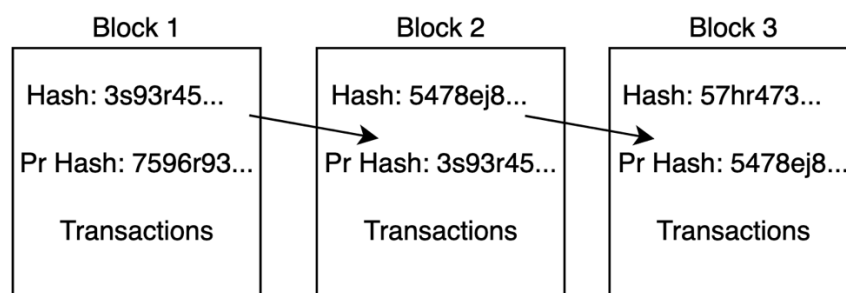


Figure 8: Example of blocks in a Blockchain

### 2.4.2.1 Advantages

Using Blockchain for E-Voting has many advantages. Firstly, because of the complex cryptographic proofs among nodes in the network transactions on a blockchain can be trusted [27]. This is especially useful in E-Voting as the System can be sure that a vote has come from a valid node. Secondly, Blockchains are **immutable** meaning it is computationally infeasible to change or delete records made [26]. This allows for the election process to be audited and for the election results to be recomputed if there is a dispute. Thirdly, the **decentralised** nature of the blockchain means that there is no single point of failure that adversaries can attack [25]. Further, this removes the need to trust a single authority, instead students can have faith in a transparent system. Lastly, blockchain allows for every node in the network to use a unique ID as opposed to their real name. This keeps people's transactions anonymous which can be especially useful in an E-Voting system as the secrecy of ballots casts is paramount [25].



### 2.4.2.2 Evaluation

One of the potential vulnerabilities of Blockchain is that its performance can be significantly reduced when more nodes become active [28]. An example of this can be seen with Bitcoin which saw a significant reduction in performance when its price increased to \$20,000 in December 2017. This could be a huge issue for an E-Voting system if multiple users are trying to vote at the same time. In addition, because Blockchain is a new and complex architecture, administrators managing the blockchain will need to have a good technical skill [29]. Furthermore, without effective governance of the blockchain administrators will be unable to properly oversee the election process, which could bring the integrity of the election into question. Additionally, irrespective of a Blockchain's data integrity a blockchain based E-Voting system is still susceptible to viruses, worms and internal attacks. Lastly, another issue with Blockchain is its close association with cryptocurrencies. It is no secret that cryptocurrencies have revolutionised the black-market as they allow for untraceable transactions to be carried out. Therefore, some individuals may not trust the technology as it is closely associated with scams and illegal activity. This is a huge problem because as stated previously without social acceptance of the technology being used, users will have no faith in the E-Voting system, which will lead to lower turnout.

### 2.4.3 Selecting an Approach - Blockchain

After assessing the pros and cons of a Blockchain approach against a Centralised Database with cryptographic techniques, Blockchain will be used to develop the E-Voting System in this project. This is because the **immutability** and anonymity guaranteed by a blockchain will ensure that votes cannot be changed or traced once cast. Additionally, because blockchain uses a distributed network there is no single point of failure that an adversary can exploit. Along with this, because blockchain does not rely on a few central authorities, the possibility of collusion is significantly reduced. Lastly, Blockchain is a new and innovative technology which will allow for the acquisition of new skills one of the success criteria for this project.

## 2.5 Permissionless and Permission Blockchain

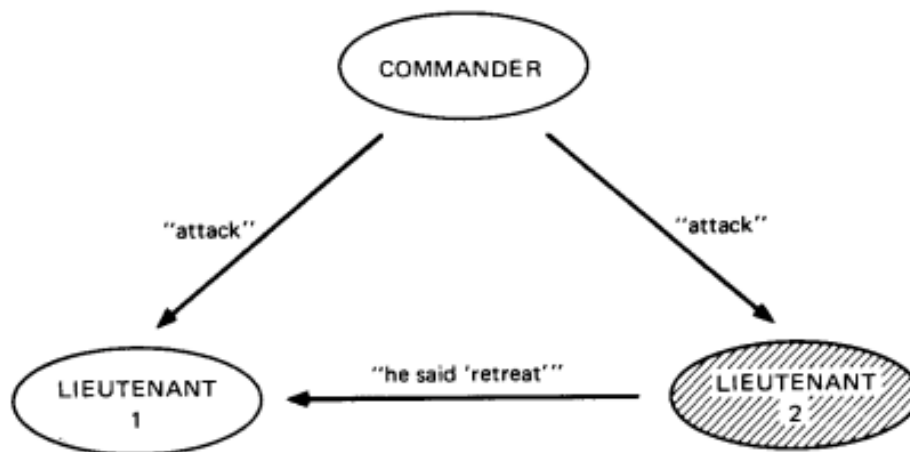
There are two types of Blockchain, permissionless and permissioned, which based have their own characteristics and different use cases.

Permissionless and Permissioned blockchains also differ on the consensus algorithm they use. The distributed nature of information on the network means that Blockchain relies on a consensus algorithm to ensure an

agreement is reached on the state of data among distributed nodes. One of the properties that consensus algorithms must achieve is Byzantine Fault Tolerance [30].

To explain what the Byzantine Fault Tolerance is the Byzantine general's problem must be understood. The Byzantine general's problem was first articulated by Lamport, Shostak and Pease in 1982 [30]. The best way to understand the problem is to imagine a Byzantine army surrounding a city. The army is divided into groups each lead by a general. Furthermore, for an attack on the city to be successful all generals must agree to attack at the same time. However, generals can only communicate by sending messages to each other. This problem lies in the fact that generals can act as traitors meaning they can send false messages to other generals in order to prevent consensus. Additionally, it is possible for messages to get lost or corrupted further adding to the problem.

Furthermore, it is impossible to solve the Byzantine general's problem when  $\frac{1}{3}$  of the generals are traitors. In **Figure 9**, **Lieutenant 2** acts as a traitor sending a false retreat message to **Lieutenant 1**. Similarly, in **Figure 2**, the **Commander** acts as a traitor and sends an attack message to **Lieutenant 1** and a retreat message to **Lieutenant 2**. In both scenarios **Lieutenant 1** will not be able to determine which general is the traitor and thus will not know whether to attack or retreat. Therefore, a consensus algorithm can only provide Byzantine Fault Tolerance to a distributed network if the number of potential traitor generals (malicious nodes) is less than  $\frac{1}{3}$ .



*Figure 9: Lieutenant 2 is the traitor*

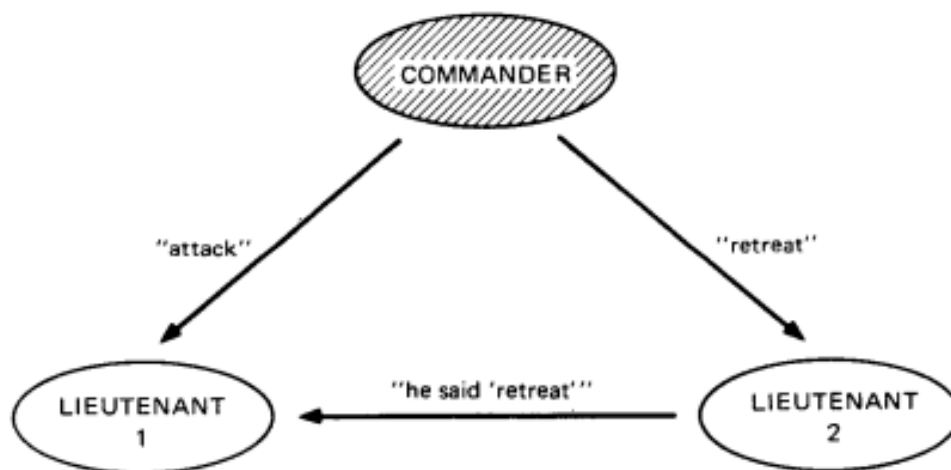


Figure 10: Commander is the traitor

## 2.5.1 Permissionless Blockchains

Permissionless Blockchain allows for anyone to join and leave the network as a reader or writer at any time [26]. There is no central entity which controls the membership of the blockchain. This means that at any time someone can see a list of all transactions that occurred on the Blockchain. The two most popular implementations of a permissionless blockchain is Bitcoin and Ethereum.

Permissionless blockchains use the following two consensus algorithms:

**Proof of Work (PoW)** – Proof of work was first presented as an idea to combat email spam by Dwork and Naor in 1993 [31]. The proof of work algorithm requires nodes to function as a prover or verifier. All nodes will attempt to solve a resource-intensive computational puzzle, once a node has found a correct solution the other nodes that will verify the solution [32]. The node that solves the puzzle correctly will broadcast their new block to the network. It is important to note that the verification process undertaken by the losing nodes requires significantly less computational power than solving the puzzle. The idea behind PoW is that a successful attack would require a lot of computational power and time to solve the puzzle. Further, since the incurred cost is greater than the potential rewards attacking the network is inefficient and, in many cases, infeasible. PoW is used as the consensus algorithm in Bitcoin where miners act as provers and verifiers. Furthermore, under Bitcoin there is an incentive to become a miner as successful miners are rewarded with Bitcoins [32].

**Proof of Stake (PoS)** – Proof of Stake was first presented in the Bitcointalk forum 2011 [33], to deal with the energy consumption issues in PoW. PoS algorithm uses a pseudorandom election process to select a node as a validator, this node will then validate the next block in the blockchain [32]. In order to be eligible to become

a validator, the node must own a certain number of assets in the blockchain [32]. In other words, the bigger the stake of an individual in the blockchain the higher the chances that they will be selected as a validator. To mitigate the algorithm only favouring the wealthy other factors such as, randomness and the age of coins are used. As a reward for validating the next block the node receives a transaction fee associated with adding the block. This consensus algorithm provides security because it disproportionately selects individuals with a high stake in the network to act as validators. Further, if the network detects a fraudulent transaction the node that verified the block will be penalised financially and will lose their right to be a verifier in the future. The only way for an attacker to approve fraudulent transactions is for them to own 51% of the assets being traded on the blockchain, something that is often infeasible. Peercoin, a cryptocurrency, was the first implementation of PoS [34].

## 2.5.2 Permissioned

This type of Blockchain only allows an authorised group of readers and writers to join the network [26]. Under this implementation a central entity grants rights to allow individuals to read and/or write to the Blockchain. The most widely known instance of permissioned blockchains are Hyperledger Fabric and R2 Corda. Permissioned Blockchains use the following consensus algorithms:

### **Practical Byzantine Fault Tolerance (pBFT)**

The pBFT was first proposed by, Miguel Castro and Barbara Liskov, to deal with the Byzantine general's problem. Practical Byzantine Fault Tolerance works by ordering nodes in the System sequentially, with one node acting as the leader and the others acting as backup nodes. All nodes in the System communicate with each other in hope that all honest nodes will come to an agreement of the state of the System using majority rule. This only works if malicious nodes do not equal or exceed 1/3 of all nodes in the System. Further, it can be said that as the number of nodes in the pBFT increases so does the security.

## 2.5.3 Evaluation

Permissionless Blockchains are true decentralised as anyone is able to access the Blockchain. This means that it is a fully transparent network that cannot be controlled by a single authority. USSU elections would benefit greatly from this transparency as students will have more faith in a system that is not centrally controlled. However, because by default everyone has access to the network extra configuration work will have to be dedicated to ensuring only eligible voters can vote in the election. Additionally, the open nature of permissionless blockchain make them highly secure, as there is a positive correlation between the number of

nodes and the security of a blockchain. While this is a positive attribute USSU elections will not benefit greatly from having a more secure network, as the E-Voting system will be operating in a low threat environment. Another, drawback of a permissionless blockchain is that its open nature makes the overall speed and performance of the System is slower. Slow performance will diminish the overall voting experience of students, which could reduce voting turnout in USSU elections.

In addition to this the, permissionless system uses proof of work and proof of stake consensus algorithms which are impractical for an E-Voting System. Firstly, because each student is only allowed to vote once, there is no incentive, in the form of a reward, for students to act as a prover. Further, without provers in the network the blockchain will fail to function as intended. The proof of stake algorithm is equally impractical. This is because, every voter has an equal stake in the election represented in the form of a vote, which means the algorithms pseudorandom election will fail to elect a node with a significant stake in the election.

Nodes in a Permissioned Blockchain must be given access by the network administrators before they can make transactions on the Blockchain. This authentication process can take many forms, for example, nodes may have to undergo an authentication process before they can make transactions on the Blockchain. If a permissioned Blockchain was adopted, only allowing University of Surrey students to vote would be straight forward, saving development time. Additionally, because only a limited number of people can access the network permissioned blockchains are much faster and energy efficient than permissionless blockchains. This can make voting for students quicker, making the voting process more convenient which will increase turnout. The drawback of permissioned Blockchains is that they are governed by a central actor, which increases the risk of collusion. Having said this, other security controls such as having multiple actors who oversee the network can solve this problem. Another issue with permissioned blockchains is that they are less transparent than permissionless blockchains, which could reduce the trust students have in the election. This issue can be mitigated by publishing an anonymised list of how people voted after the election.

	<b>Permissionless Blockchain</b>	<b>Permissioned Blockchain</b>
<b>Throughput</b>	Low	High
<b>Latency</b>	Slow	Medium
<b>Number of Readers</b>	High	Medium
<b>Number of Writers</b>	High	Low
<b>Centrally managed</b>	No	Yes

On balance, using a permissioned Blockchain is more suitable for this project. This is because as outlined above many of the issues with a permissioned Blockchain can be mitigated. Additionally, the consensus algorithms that permissionless Blockchains are not fit for purpose. Lastly, the low threat environment that the application will function in means that the high level of security that permissionless blockchains provide is unnecessary.

## 2.6 Conclusion

To conclude, this literature review has examined the fundamentals that make a good E-Voting system, as well as the approaches that will be taken to develop the new E-Voting system. After evaluating the security vulnerabilities of E-Voting systems, it was concluded that a security analysis will be conducted before development begins. In addition, the attributes that determine a good E-Voting system were identified and ranked based on their importance. It was concluded that the attributes ranked highest will be implemented in the earlier sprints, and the attributes ranked lower in the later sprints. Further, upon the completion of this project the System produced will be evaluated against the identified attributes. Lastly after considering a variety of options, it was decided that the new E-Voting System should be implemented using permissioned Blockchain technology.

# Technology Review

---

This section will discuss the various frameworks and technologies that can be used to develop the system. When one approach is chosen over another, a suitable justification will be provided.

## 2.7 Native, Hybrid, Progressive and Cross platform

To ensure that all students at the University of Surrey can vote in a student election, the system must function on Android and IOS devices. This is because 90% of individuals own either an Android or IOS device, thus by ensuring the system functions on both device all students will be able to vote.

There are three technologies that can be used to ensure the system functions on Android and IOS.

### 2.7.1 Native

A native application is developed exclusively for a single platform [35]. This means that it is written in one programming language for a particular operating system [36]. Native applications for Android devices are written in Java or Kotlin and native applications for IOS devices are written in Objective-C or Swift.

Native applications perform faster due to the fact that the code written is less complex [36]. Additionally, Native applications function well in offline environments as internet connectivity is not required to render UI components [37].

The user experience for Native applications is much better because Native applications are a nuanced version of their device's default apps [38]. Furthermore, when a user performs some functions, he quickly understands the natural flow of the application because it is similar to apps already on the device [38]. Lastly Native applications are more secure. This is due to Native applications non-reliance on web technologies which often have security flaws. Therefore, developing a Native mobile app is a great way to guarantee the protection of user data [38].

The main drawbacks of building a Native application is that multiple code bases will have to be maintained and worked on separately, if the application is being used on IOS and Android [38]. Therefore, more knowledge is required for the development of native IOS and android applications, as expertise in two different programming languages (Java and Objective-C) is required.

## 2.7.2 Hybrid

Hybrid applications have been described as the golden mean between native and web applications [36]. Hybrid apps are deployed in a native container that uses a mobile WebView object, giving users the look and feel of a Native application [39]. It does this by taking advantage of web technologies such as JavaScript, HTML and CSS.

Hybrid applications running on different devices will use the same backend code. This allows for hybrid applications to be developed fast [40]. Additionally, because only one codebase is needed for the application to run on Android and IOS, maintenance of the application is much easier. Both of these feats reduce the development cost of hybrid application, especially when the application is intended for multiple platforms.

The web-based nature of hybrid applications means that the application will not work without internet connection. To add on, the performance of the application will be dependent on the user's internet connectivity [36]. Another disadvantage is that despite sharing a codebase certain features may be supported by Android and not by IOS [37]. This could lead to display inconsistencies and bugs that will have to be caught with additional testing.

## 2.7.3 Cross-Platform

Cross-Platform applications are similar to hybrid applications, in the sense that they allow for a single codebase to develop applications on IOS and Android. However, Cross-Platform applications are different from hybrid applications as they do not rely on web technologies. Instead, code written in Cross-Platform application are compiled to native element giving the user the look and feel of a native application. The three most popular Cross-Platform technologies are: Flutter, React Native and Xamerin.

One advantage of Cross-Platform applications is the reduced development time, as only one code base will need to be maintained [37]. Additionally, the applications non-reliance on web technologies means that it can function without internet connection. Furthermore, while the overall performance of Cross-Platform applications is slightly slower than Native applications, it performs much better than Hybrid applications [41]. Also, because the code written is rendered to native code, Cross-Platform applications will give a native experience to IOS and Android users.



The main disadvantage of Cross-Platform applications is the level of skill required to be able to develop an application that is adapted to different platforms. A developer will need to keep all the little differences between operating system and the hardware they run [36].

## 2.7.4 Evaluation

In order to determine which development approach will be taken, the above approaches must be measured against the needs of the proposed E-voting system.

If the e-voting system was implemented as a Native application, two separate codebases would need to be maintained, in order to allow for IOS and Android users to use the application. Although a Native application would allow for the highest possible performance and better security (an important characteristic for an e-voting system), the time constraints of the project means that it would be impractical for two separate versions of the application to be developed simultaneously.

While hybrid applications allow for a single codebase to be maintained, it does this at the cost of performance and user experience. Problems with the UI will occur as a result of students having phones of various screen sizes. Even if an iPhone and Android device are used to test the UI of the application, it would be infeasible to test all possible screen-sizes. Emphasis has been placed on improving the student voting experience, therefore developing a hybrid application will not allow this goal to be achieved.

Cross-Platform applications are slower in comparison to Native applications. While they provide a better user interface than hybrid applications, developing a Cross platform application require specialist technical skills. Furthermore, extra work will be requiring to configure the application function as indented on both IOS and Android users.

## 2.7.5 Selecting an Approach (PWA)

Like hybrid applications progressive web application are built using web technologies. A progressive web application is best described as a mobile responsive website that acts and feels like an app. Frameworks such as Angular and React can be used to build these kinds of applications. Further, Progressive web applications, offers the benefits of a mobile app and web application with minimal downside. Firstly, because only a single codebase is required the application can be developed faster than a native application. Secondly, because the application requires an internet connection, users will always be using the latest version of the application, increase the secure of the system. Thirdly, because the application is operating on the web, the system can

generate a link to allow students to access the election. This is something that cannot be done easily with native and cross platform applications as user will have to first download the app before they are able to vote. The impact on turnout could be significant as having to download an application is an additional step that lengthens the process of voting. This is in line with study that show approximately 20% of users are lost during the onboarding stage of a native app [42]. In addition, students will be able to vote on any device that has a browser. By increasing the number of devices students can vote on, it is logical to assume this will increase the likelihood of them voting. Lastly as an individual who wants to work in web development after university, selecting this approach will allow the acquisition of vital web development skills that can be used in the real world. **Therefore, the e-voting system will be implemented as a progressive web application.**

## Chapter 3: Problem Analysis

---

This chapter will offer an analysis of the problems the USSU face with their existing E-Voting System, which will inform the requirements specification of the new E-Voting system. More specifically this chapter will comprise of an analysis of, the USSU's existing E-Voting System and the findings from interviews, focus groups and surveys.

### 3.1 Interviews, Focus groups and Surveys

The best way to ensure a system meets the needs of users, is to ask users for their opinions. While there are many techniques that can be employed to find out what users require, interviews, focus groups and surveys were used in this project. This section will present and analyse the opinions of the Surrey Student and USSU staff.

### 3.2 Interviews

Interviews are one on one meetings, in which questions are asked by the interviewer and answered by the interviewee. Interviews are an ideal way to find out what specific individuals want to see in the new E-Voting system. Further, interviewees can share their experiences when using the existing E-Voting system and spotlight flaws such as a poor UI. Additionally, interviewees can provide innovative ideas and suggestions that can solve these issues. However, the quality of the information gained from an interview depends solely on the person being interviewed. Due to this, interviews were only conducted on the members of USSU staff that use the system. This is because, their experience as Administrators of the current E-Voting system mean they, have a more technical understanding of the current system when compared to students. This makes

staff at the USSU more suitable for highlighting the shortcomings of the current system and suggesting ideas and improvements.

### ***3.2.1.1 Interview Ethical Consideration***

The interview was conducted with staff members at the USSU. In compliance with the Data Protection Act (2018) and this project Statement of Ethics, interviewees were asked to sign a consent form (see appendix A) so that they can be named and quoted in this report.

## **3.2.2 Interview Analysis**

A total of three meetings were held over the course of this project and the key takeaways of each meeting can be found below. Furthermore, lockdown rules meant that interviews had to be conducted via zoom. To ensure interviews were as productive as possible, questions were prepared beforehand and each interview had a predefined objective. The objective and analysis of the outcome of each meeting held with staff at the USSU can be found below.

### ***3.2.2.1 Meeting 1***

The main focus of this meeting was to inform the USSU that the development of a new E-Voting System was underway. Additionally, getting the VP Voice (Theo) to agree to meet upon request was another objective. Lastly, ascertaining the key issues with the existing E-Voting System and ideas that could be added to the new E-Voting System was of paramount importance.

The main outcome of the meeting was as follows:

- The VP Voice (Theo Donnelly) who is responsible for overseeing elections at the USSU agreed to meet regularly and provide feedback on the progress of my application.
- It was made clear that there are three types of elections that the USSU run, SurreyDecides, AGMs and Executive resolutions.
- The voting system used by the USSU is provided by a third party MiVocie.
- The Coronavirus pandemic has meant that AGMs which are normally conducted in person are being conducted virtually using surveys created in Microsoft Teams, an insecure system the USSU does not wish to repeat.
- The ability to change your vote until the election officially closes, while this idea was welcomed there was a general feeling that such a rule would have to be decided upon by the USSU's democracy

committee. This was due to the moral question it raises about people losing votes. It was agreed that if this feature was implemented there would need to be an option to turn it off.

- The ability to allow QR codes to be scanned by users and then to be taken to the voting page was also a feature that the was appreciated.

Taking into consideration that the USSU does not have an E-Voting system that can be used for AGMs, the first two iterations of the new E-Voting system will focus on catering to AGM elections. The last iteration of the E-Voting system will focus on SurreyDecides elections. Further, a system that caters to executive resolutions is beyond the scope of this project. This is because executive resolutions are voted on by 12 elected students. Therefore, it would be futile to create a system that will only serve a small number of students.

Another feature that has been adopted following the meeting was a scannable QR code that allows students to access an AGM. This ensure that only students in attendance of the AGM can vote in it (in line with the USSU election policy).

### **3.2.2.2 Meeting 2**

The main purpose of this meeting was to find out more about the USSU existing E-Voting systems. In the meeting a walkthrough of the USSU current E-Voting system for AGM elections and SurreyDecides election was given. Details of the walkthrough can be found later in this chapter under ***the current system***.

In addition, it was originally intended for the application server to send authentication requests to Surrey365. However, for security reasons the IT department at the University of Surrey were unable to grant my application access to Surrey365. In the meeting the USSU agreed that once the application had demonstrated core functionality, they would facilitate a meeting with the IT department so that an agreement can be reached. Therefore, the feature to use Surrey365 to authenticate students and to get access to students' personal information will be implemented after this project is completed.

## **3.3 Focus Groups**

Focus Groups are group discussions in which participants are given a topic or question to discuss. One advantage of using focus groups is that they offer an opportunity for new ideas to be formulated as participants can bounce ideas off one another. Furthermore, focus groups provide insights into the minds of end-users, which will help in the prioritisation of system requirements. Additionally, focus groups can be

used as a way to ask for user feedback, after each sprint, to ensure the system developed is liked by students. However, the quality of focus groups is dependent on getting a diverse range of participants to take part. Without a diverse range of participants everyone will express similar opinions which could mean that the requirement specification (for the new E-Voting system) alienates a demographic of end-users. Therefore, to ensure the information gathered from focus groups is truly representative of student opinions a diverse range of participant based on gender, race, year of study and course was selected.

### 3.3.1 Focus Group Ethical Consideration

In compliance with the Data Protection Act (2018) and this project Statement of Ethics, participants will be asked to sign a consent form (see appendix A) so that focus group sessions can be recorded. The recording of these session will allow for more time to be spent interacting with the participants, as there is no pressure to take notes during the session. The recordings were deleted upon the completion of this project, and were stored safely on a password protected MacBook. In addition, student ideas and opinions were kept anonymous in this report ensuring that the confidentiality of students is respected. Lastly it was made clear to participants of the focus group that they could leave the session at any time, if they left uncomfortable (with the questions being asked) or wanted to take a break.

### 3.3.2 Focus Group Results

After the User Acceptance Testing was run, a focus group consisting of University of Surrey students were asked a series of questions about their experience when using the application. As mentioned above, due to legal reasons, the audio recording during the focus group was deleted after the completion of this report, therefore it cannot be included. The findings from the focus group can be found in **Chapter 8: System Testing** in section **8.4.2 UAT Findings**.

## 3.4 Surveys

Surveys are a set of questions that are filled out by participants. The main advantage in using surveys is that they can be used to gather responses from a large amount of people. However, the issue with Surveys is that because they are completed independently, unlike focus groups and interviews, clarifications and follow-up questions cannot be asked. The proceeding section discuss how the survey used in this project was designed and distributed.

### 3.4.1 Survey Objectives

Before formulating the questions that will go into the Survey it is important to establish the intended learning objectives from gathering the data. The main objectives of the survey are:

- To find out what students think about the current USSU E-Voting System
- To find out the most important aspects of an E-Voting System to students
- To find out what kind of devices students want to use to vote on
- To find out if students have any concerns about using an E-Voting system
- To find out if students have any suggestions or ideas that can improve the current USSU E-voting system

### 3.4.2 Survey Target Audience

The proposed system is being designed for students at the University of Surrey; therefore, it is important that the views of as many Surrey students is taken into consideration. To ensure that the responses come from University of Surrey Students, they will be asked for their university email (as Surrey Student emails end with @surrey.ac.uk).

### 3.4.3 Survey Tools and Distribution

Microsoft Forms will be used as a tool to create and distribute the Survey. Microsoft Forms was selected for the following reasons:

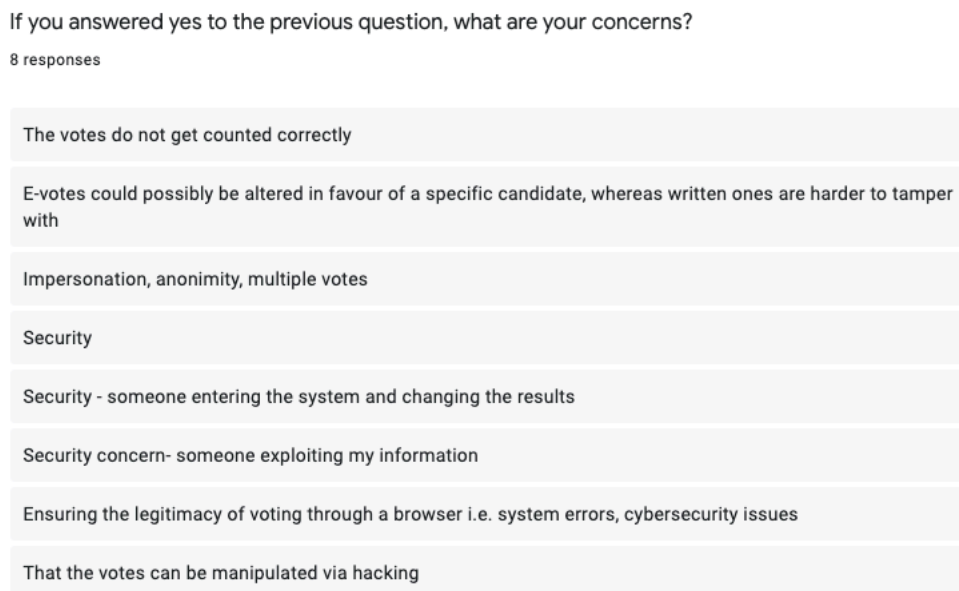
- It is a convenient way for students to complete the Survey as it can be completed via a link on their phones and laptops.
- It will provide quick and easy access to student responses as responses in Microsoft Forms are presented in a clear dashboard.
- Updates to the Microsoft Forms dashboard are instant meaning responses from students can be ascertained in real time.
- It allows for responses to be looked at in a granular way, allowing for deeper insights into student responses if required.

In order to get the survey completed by as many students as possible the survey link will be sent to those in my computer science cohort via social media, email and the CompSci WhatsApp group chat. Additionally, friends (that attend the Surrey of University) will be asked to send the Survey to two other people upon completion. In line with the project plan, the Survey will open from November 20<sup>th</sup> 2020 to January 1<sup>st</sup> 2021, which is enough time for a large sample of responses to be gathered.

### 3.4.4 Survey Design

The survey was kept as short as possible (consisting of 10 questions), as a way of ensuring that students completed the Survey in its entirety. Additionally, to make the completion time of the Survey quick the majority of question asked were closed questions with the flexibility for the respondent to expand upon their answer. For example, as shown in **Figure 11** below, the question allows the respondent to provide a justification for the answer. This allows for students to expand their answer and offer more clarity if they desire.

It was also important to record the age, gender, ethnicity, course and year of study of each respondent so that biases are identified.



*Figure 11: Example of Open Question*

Since the completion of this section the online Survey has been removed (to prevent late respondents). However, the full Survey can be found in the Appendices.

### 3.4.5 Survey Ethical Consideration

In compliance with the Data Protection Act (2018) and this project's Statement of Ethics, survey participants were asked for their informed consent to use their responses in this report. All responses were kept in a password protected Microsoft Teams account, and were deleted upon this report's completion. The survey

asked students for information that would be classified as sensitive personal data. For example, the ethnicity and gender of each respondent was collected. This was done to allow for a robust evaluation of the data collected (at a later stage). Indeed, it is important to note that students had the option to opt out of these personal questions by selecting 'prefer not to say'. Lastly, in the Survey description it was made clear that students were under no obligation to complete the survey once they had started it.

### 3.4.6 Survey Response Analysis

Below is a summary of the survey responses along with how these responses will inform the requirements of the system.

It is important to note that at the time of the survey's launch 1<sup>st</sup> year students have not yet voted in a USSU election, so have not used the existing E-Voting system. This means responses from students in 1<sup>st</sup> year on the existing E-Voting system will be disregarded. However, their contribution to the other questions in the survey will be treated with the same validity as other year groups.

A total of 93 students with a valid University of Surrey email completed the survey.

The majority of students who answered the Survey were from African descent (62%) and 58.1% of respondents were female. Year groups were better represented with no single year group having more than 29% of respondents. There was also a diverse range of courses that participated with the most popular course being computer science (28%). A full evaluation of the sample and how biases could have influenced the result of the survey can be found in the evaluation chapter.

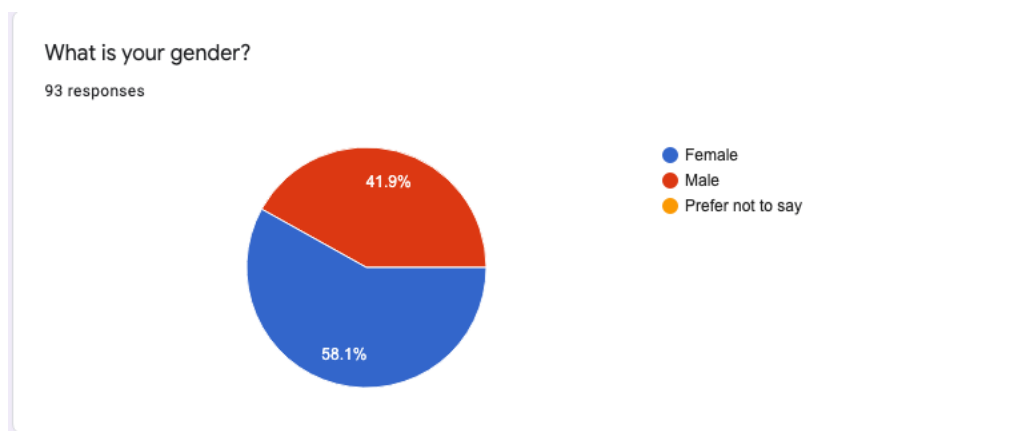


Figure 12: Percentage of Male and Female that took the survey



Approximately 2/3 of the respondents said that they had voted in a student's union election. Furthermore, of the people who have voted using the USSU existing E-Voting system 65% did not believe their voting experience was good, with 59.3% saying their experience was neutral and 5.9% saying their experience was bad or very bad. This demonstrates that there is a need for a better E-Voting system that students can be enthusiastic about.

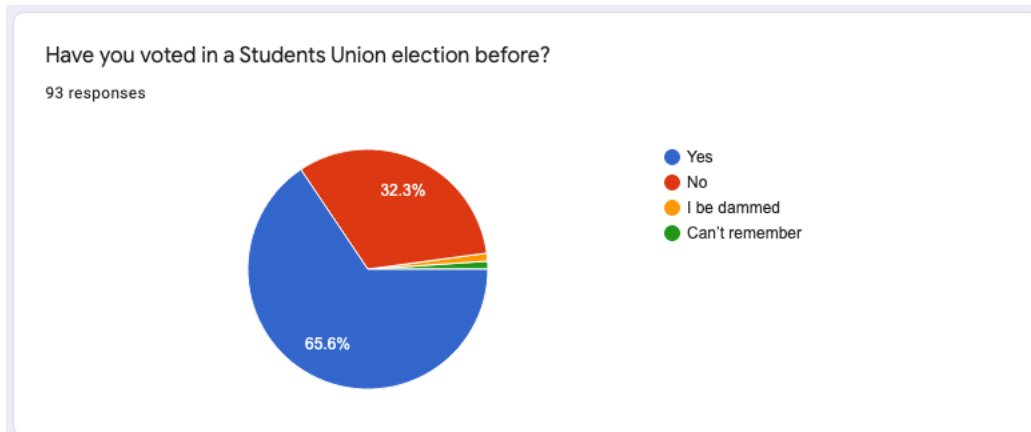


Figure 13: Percentage of Student who have voted in USSU election

A small minority of students do not trust the USSU to oversee and manage student elections (6.5%). This means that the concerns raised in the literature review about a private blockchain being susceptible to internal attacks, can be deprioritised as students trust USSU staff to oversee elections.

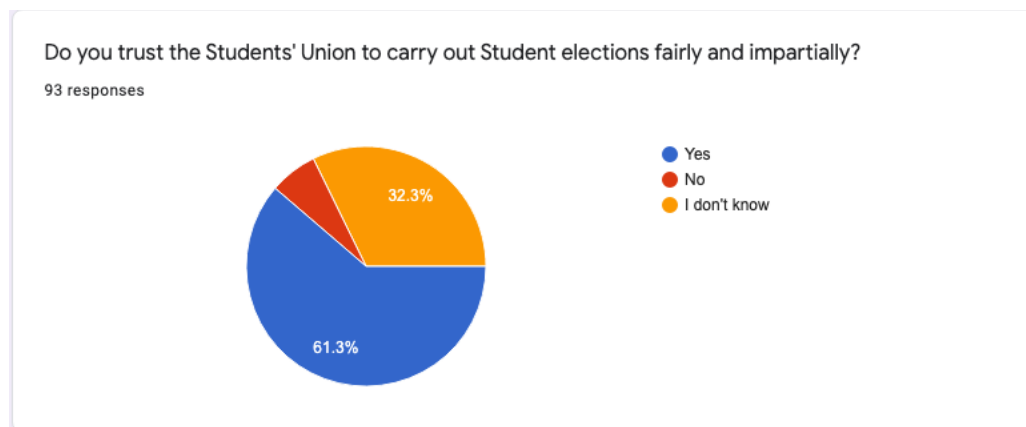


Figure 14: Percentage of Students who trust the USSU to carry out election

The survey also revealed that the majority of students would want to vote via their iPhone or Android (88%). The second most popular method of voting is via a laptop, which concurs with the literature review which asserted that a web application is the most convenient solution (as it allows student to vote via their Android and iPhone).

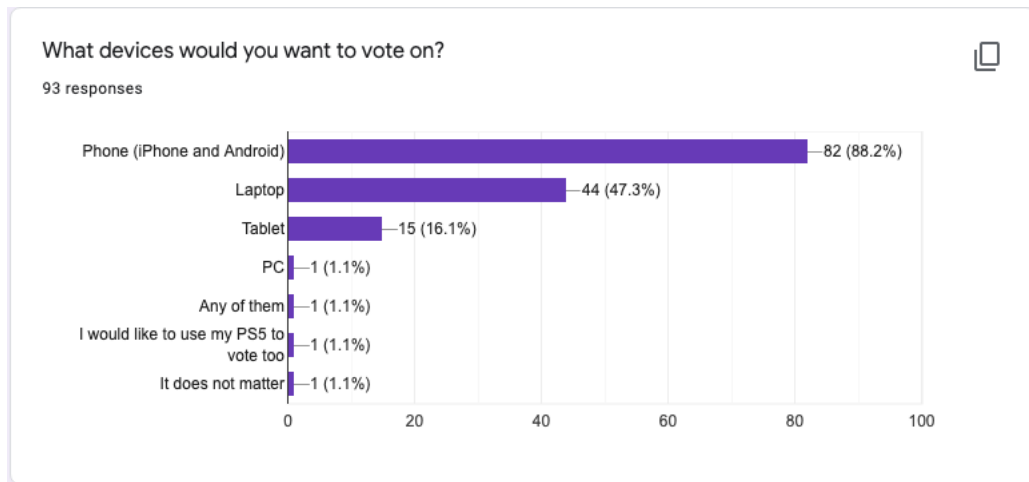


Figure 15: Popularity of Devices for Voting App

Additionally, the survey showed that students rank democracy, security, usability and reliability as the most important properties in an E-Voting system. This concurs with the conclusion drawn in the literature review which ranks democracy as the most important property of an E-Voting system. It is also important to note that many students do not view the privacy of their vote as important, reinforcing the idea that semi-privacy should be pursued instead.

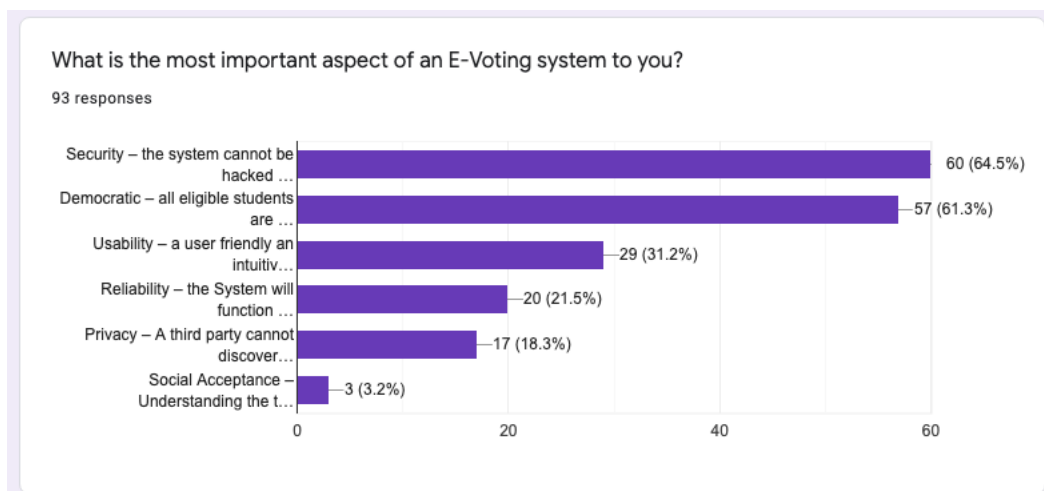


Figure 16: Most important aspects of E-Voting to Students

It was also shown that the majority of students do not have any concerns about voting using a web application, with 85% answering no. Those that answered yes sighted 'hacking' and 'multiple voting' as the cause for concern. However, despite students viewing E-Voting web application as trustworthy, only 58% of students trust the USSU E-Voting system to properly record and count all votes. From this we can deduce that 26% of students trust the principle of E-Voting but do not trust the system being used by the USSU. This

is a significant number of students who perceive the E-Voting system used by the USSU to be insecure, highlighting the need for a better system.

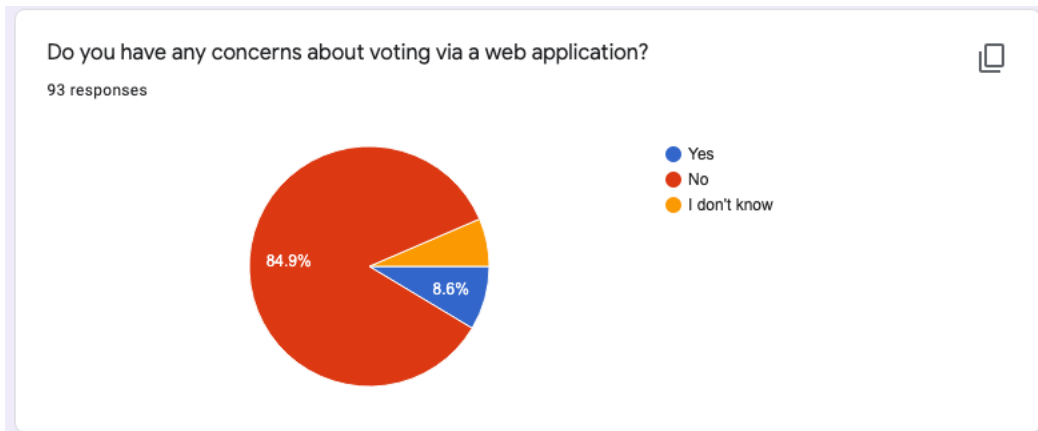


Figure 17: Percentage of Students concerned about using a Web app for Voting

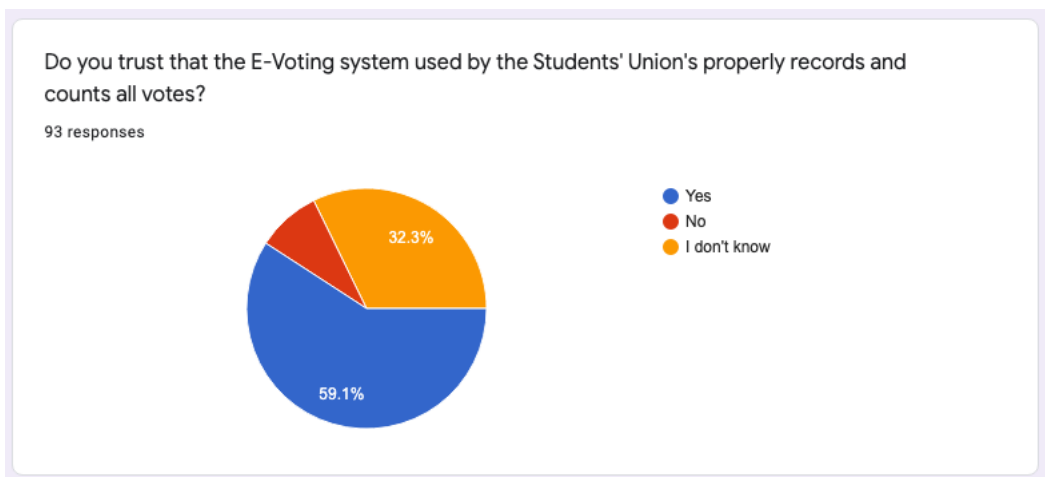


Figure 18: Number of Student who trust the USSU records votes properly

Lastly the final question which asked students for their suggestions on features the E-Voting system could incorporate had three noteworthy responses.

The first feature was for the system to send automatic email receipts after students cast their vote. The logic here is that students will be put at ease, as the email receipt allow students to verify that their vote was counted correctly. However, this feature would breach the important concept of privacy which was discussed in the literature review. With that said, verification is important so, instead of sending an automatic email receipt, an anonymous list of how each student voted will be publish such that students check the list to verify their vote (semi-privacy). For security purposes only administrators can publicise the list after the election is completed.

The second suggested feature was to include the biography of candidates within the E-Voting application. This would only be possible in SurreyDecides elections where candidates are known in advance of the election. Implementing this feature in AGM election is infeasible because USSU bylaws allow for any student to run on the day of the AGM. In addition, AGMs are carried out in a short time window, therefore it is unlikely that students at the AGM will be able to read the biographies of all candidates.

The third feature was to show the election results in real time. This was a good idea in theory, however revealing this information before ballots closed could influence the election outcome.

To conclude the Survey conducted reaffirmed ideas established in the literature review and confirmed that students are not satisfied with the existing E-Voting system used by the USSU.

## 3.5 4.1.4 The Current System

This section will discuss and evaluate the two existing E-Voting Systems used by the USSU.

### 3.5.1 Mi-Voice

Mi-Voice is the name of the E-Voting System used for SurreyDecides elections. This system was managed and developed by a third-party organisation (called Mi-Voice).

#### 3.5.1.1 Overview of Mi-Voice

Mi-Voice allows administrators to set up events which can consist of multiple contests (elections). In the USSU case SurreyDecides is the election event and the contests are the various elections within SurreyDecides, for example the election of the USSU President. Every contest requires a name, description and can be customised to suit the election being held. These customisation options range from allowing open responses to only the election system (FPTP or STV). Once the contest is created, the name, photo, description and other information of each candidate has to be manually entered by the administrator. Once all candidates have been inputted, administrators can start, end and release the results of the election. Administrators also have access to a live dashboard which shows key stats about the election, such as student turnout and the vote tally.

In regards to students, once the election has been set up and made live by the administrator student will be able to vote, via a unique link (generated for each student). Students can access this link via their email, or the USSU website which will redirect them to their unique link once they log in (see diagram below).

### ***3.5.1.2 Evaluation of Mi-Voice***

The main problems identified when using Mi-Voice for SurreyDecides are as follows.

**Labour intensive:** One of the main problems with Mi-Voice is that candidates in the election have to be manually entered. This is a long-winded and tedious process that requires a lot of work as SurreyDecides elections normally consist of 70 candidates. Addressing this issue will free up the time of USSU staff members allowing them to focus on other USSU priorities. Furthermore, one way in which this issue could be solved is by allowing each candidate to upload their own details and manifesto. This self-service feature will reduce the likelihood of candidate details being entered incorrectly.

**Single use:** During the demonstration, it was made clear that Mi-Voice can only be used by the USSU to run SurreyDecides elections. AGMs could not be run using the system because Mi-Voice requires the candidates to be known in advance of the election (something which cannot be done for AGMs). Furthermore, having a single application in which all elections were conducted would improve the voting experience for students and management of elections for USSU staff. In addition, it will also mean that less time will be spent having to understand and maintain two different systems. Indeed, the best solution to this problem is to allow administrators to create elections in which the candidates are unknown. Overseers of the AGM will then be able to start the election and share the voting link with attendees via a barcode. Information detailing how this feature will function can be found in the Requirement and Design section below.

**Defragmented process:** Immersion when students use Mi-Voice system is lost early on. One reason for this is, the candidate nomination process is conducted using another system. Instead of candidates nominating themselves via the USSU website, candidates should be able to self-nominate via same E-Voting system that will be used to conduct the election. Similarly, other students should be able to endorse candidates who have nominated themselves, until the number of endorsements the candidates need exceeds the specified threshold. This feature would streamline the E-Voting process bringing together many defragmented components, which will improve the overall experience of student elections.

## 3.5.2 AGM Election System

### 3.5.2.1 Overview of AGM E-Voting System

AGMs are normally conducted in person; however, the Covid-19 pandemic has meant that student elections have been moved online. As stated in meeting 1, in the last academic year (2019/2020) the USSU used surveys powered by Microsoft teams to conduct AGMs. Furthermore, the USSU were not keen to repeat this insecure method of voting so created their own inhouse E-Voting system that leverages power automate and logic apps. Below is a step-by-step walkthrough of how election using this system work.

Once an AGM is started, the supervisor of the AGM will be presented with the below Power Automate card, prompting them to start the election.

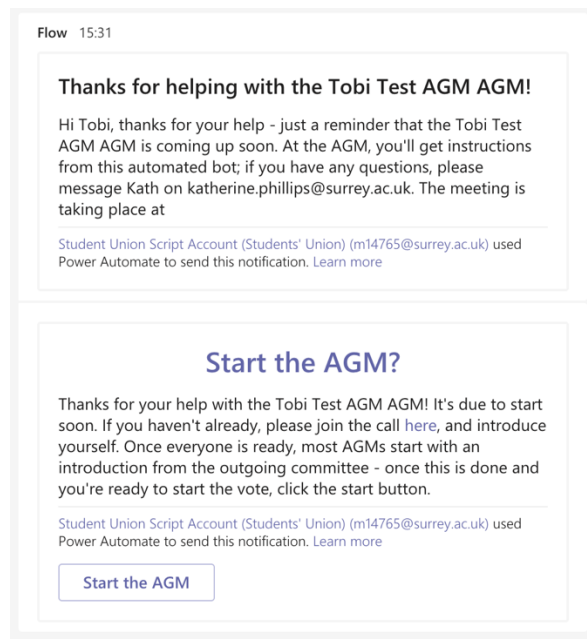


Figure 19: Launch AGM Using Power Automate

Once the start button is clicked by the supervisor, another card will popup asking for the name of the position being contested and for the name of the candidates running in the election.

Flow 19:54

**What position would you like to elect?**

Before we start the next vote, we just need to know which position it's for, and who the candidates are.

Please enter the title of the position below, and the names of any candidates that wish to stand. You should also ask each candidate to say a few words (if they wish), and answer any questions from the group if appropriate.

Position Title (eg. President)

Candidates:

Student Union Script Account (Students' Union) (m14765@surrey.ac.uk) used Power Automate to send this notification. [Learn more](#)

Figure 20: Entering Candidate in AGM Elections

Once the supervisor has added all the candidates and contests name (position title), they can start the vote. The card below shows the link that attendees can use to vote in the AGM. The card also shows all of the candidates can be voted for. Note that RON and Abstain were add by the system automatically because these options are required in all USSU elections.

Flow 20:00

**Voting Instructions**

To get voting, just share the link below in the meeting chat - on that page, a ballot will be generated for the candidates below.

The ballot will include abstain (meaning the voter does not want to vote for any particular candidate), and RON or 'Re-Open Nominations' (meaning that the voter would like different candidates to come forward).

Voting Link: [https://www.ussu.co.uk/\\_layouts/authenticate-ussu.aspx?source=https://www.ussu.co.uk/voice/Pages/Vote.aspx?group=Tobi%20Test%20AGM](https://www.ussu.co.uk/_layouts/authenticate-ussu.aspx?source=https://www.ussu.co.uk/voice/Pages/Vote.aspx?group=Tobi%20Test%20AGM)

Please copy and paste the link above into the meeting chat.

Candidates:

- Tobi
- Sam
- Bob
- Alice
- Abstain
- RON (Re-Open Nominations)

**Ready to continue?**

When everyone has voted, use the button below to get the results of the election. Careful - once you start the count, we will not accept any other votes.

Student Union Script Account (Students' Union) (m14765@surrey.ac.uk) used Power Automate to send this notification. [Learn more](#)

Figure 21: Voting In Process

Students will be prompted to enter their *Surrey365* credentials and will then be redirected to a SharePoint poll.

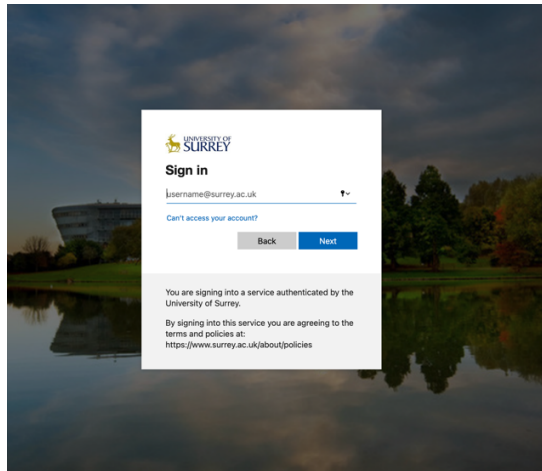


Figure 22: Surrey365 Login Screen

Students will then be able to select which candidates they wish to vote for. For the purposes of this tutorial a vote was given to Alice. Once the student clicks 'submit your vote' their vote will be saved and they will be unable to vote again.

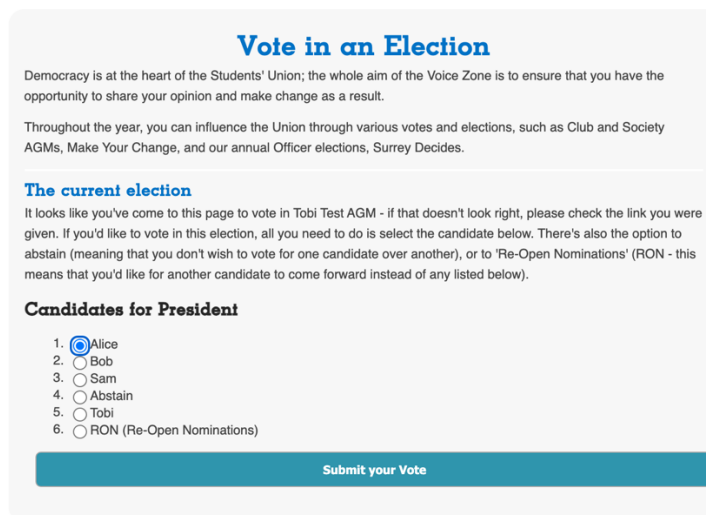


Figure 23: Vote in Election



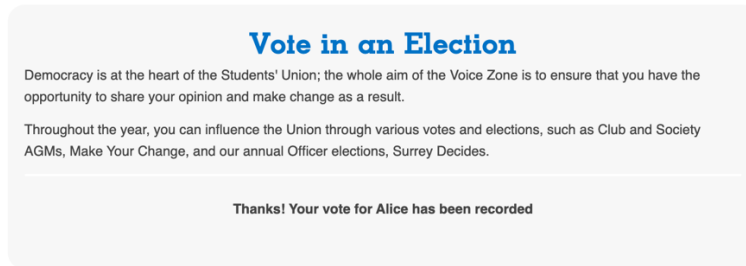


Figure 24: Vote Thank you message

Once all attendees have finished voting the supervisor can click the 'count the votes' button. The results of the election will then be counted and presented to the supervisor. The supervisor will then have to enter the name and email of the winning contestant. Furthermore, the supervisor can choose to run more contests for other positions or can end the AGM if there are no more contests.

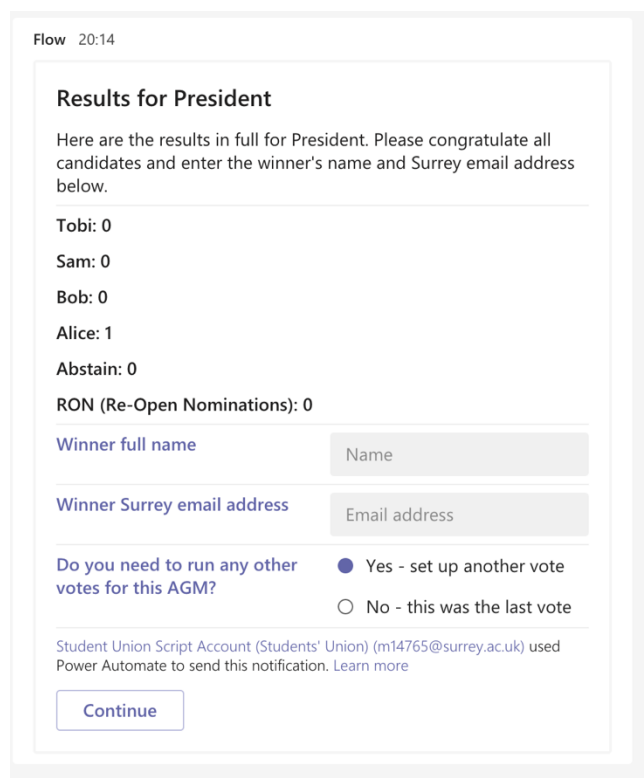


Figure 25: Results of Election

### *3.5.2.2 Evaluation of the AGM E-Voting System*

The main problems identified when E-Voting used for AGM are as follows.

**System security:** The biggest concern is that the technologies being used by the system (logical apps, SharePoint and power automate) weren't designed to support applications that require high levels of security. While USSU election are conducted in low threat environments, the system's ability to maintain the integrity of votes questionable. To add on the application was created by a non-technical developer who has not carried out a security analysis on the system. To resolve this, the E-Voting System designed in this project will undergo a security analysis to find and mitigate vulnerabilities (see design section).

**Lack of testing:** From questioning the internal developer, only alpha testing has been conducted on the system. This in combination with the fact that the developer is non-technical means that the E-Voting system may not function as intended. Unlike the USSU existing system the E-Voting system developed in this project will undergo alpha and beta testing to ensure all requirements function correctly.

**Dependencies:** Another issue is that the system leverages technologies the USSU are not directly in control of. In an event were SharePoint stops working, resolving the issue will require the assistance of other system administrators at the university. Furthermore, while the E-Voting system relies on some dependencies (IBM Hyperledger Cloud) the system developed as a standalone system with containerised components. Therefore, in the unlikely event that IBM's Hyperledger Cloud stopped functioning, the private blockchain could be replace with a backup or another distributed database.

**Virtually orientated:** The system was created for the AGMs taking place this academic year, which are all virtual (due to the covid-19 pandemic). This means that the USSU will revert back to in person paper ballot elections next year, which presents new challenges (explored in the literature review). One issue is that attendees will have to trust the AGM's supervisor to count all votes with integrity and accuracy. With the E-Voting system developed in this project attendees do not have to put their trust in a stranger, instead they will put their trust in a robust and accurate E-voting system.

**Single use:** Another issue is that the system can only be used for AGM elections. Furthermore, using two different systems for elections creates a multitude of issues (which have already been explained in Mi-Voice evaluation).

## Chapter 4: Requirements

This section brings together the information gathered in the Literature review and Problem analysis section, to provide a description of the new E-Voting System that has been developed in this project. The IEEE requirement specification template was used as a guide to ensure requirements and features were presented in a clear way.

### 4.1 Overall Description

The proceeding section will discuss the E-Voting system's: full scope, expected functions, intended operating environments, end users, constraints, functional dependencies and assumptions.

#### 4.1.1 Product Perspective and Scope

As stated in **Chapter 3: Problem Analysis** the USSU currently uses two different E-Voting Systems, one for AGM Elections and another for SurreyDecides elections. The time constraints of this project, mean that it would be infeasible to attempt to implement a replacement for both E-Voting Systems. **Therefore, the new E-Voting system will be designed for the conduction of AGM Elections.** The ability to conduct SurreyDecides elections will be added after the completion of this project (see **Chapter 10 Evaluation**).

The new E-Voting system will seek to expand upon many of the features that exist on the current system, as well as introduce new features that improve the system. With that said, the new E-Voting system will be a standalone application, that will interact with the blockchain database via a Node.js server.

As shown in figure 26, the E-Voting system will consist of three main components: the web application (client side), the Node.js server (server-side) and the Hyperledger Blockchain (distributed database).

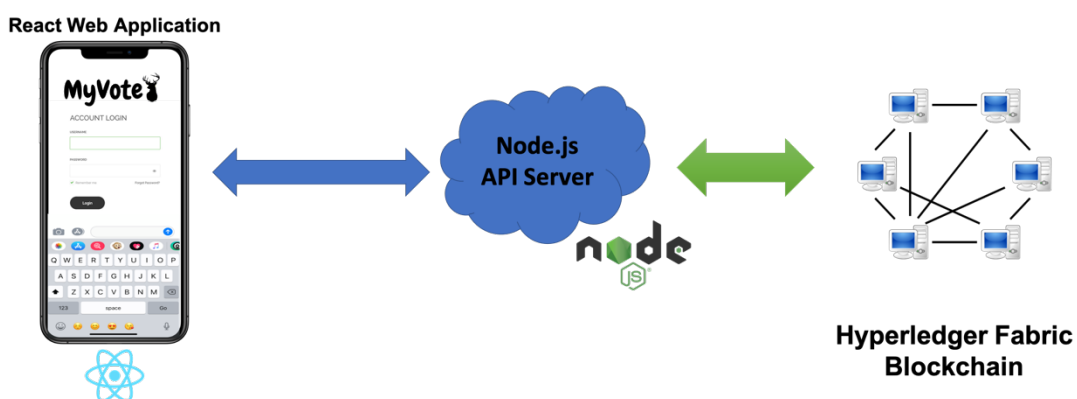


Figure 26: Diagram showing High-level overview of the E-Voting System

Information about how each component will communicate and interact with each other can be found under System Architecture in **Chapter 5: Design**.

## 4.1.2 Product Features

Below is a high-level summary of the major features that the E-Voting System will allow users (students and staff) to perform.

The E-Voting system will allow students:

- To log in to their accounts using their credentials
- To log out of their account
- To vote in the elections in line with the election rules, once logged in.
- To view the biographies of election candidates
- To view the election results once they are made visible by the administrator
- To use their device to scan a QR code to access AGM elections
- To verify that their vote was counted correctly, if made possible by administrators
- To register as a candidate in the election
- To nominate candidates who wish to stand in the election

The E-Voting system will allow administrators:

- To log into the E-Voting system using their Surrey365 credentials
- To log out of their account
- To grant administrative privileges to other USSU staff members
- To create SurreyDecides and AGM elections with customize rules, for example choosing an election system (STV or FPTP)
- To delete created elections
- To edit created SurreyDecides and AGM election, for example adding and removing candidates
- To start SurreyDecides and AGM elections, making them available for students to vote in
- To view election results before they are made visible to students
- To end elections and make election results visible
- To publish an anonymized list of how each student voted in the elections
- To approve student who register as candidates in the election

### 4.1.3 User Classes and Characteristics

As stated throughout this report the main users of the new E-Voting System will be University of Surrey Students and USSU staff members. Furthermore, USSU staff members can be divided into ordinary staff members and election officers. These types of users will differ in regards to what they are able to do on the system. In addition, AGM officers (who can be student or staff members) also have a specific way in which they will use the system. Below is a description of each user, and how their interaction with the system is expected to differ.

#### 4.1.3.1 Ordinary Students

Students are the primary users of the E-Voting system and their opinions on the E-Voting system will determine the success of this project. Students are expected to use the E-Voting system primarily to vote in elections held by the USSU. Students are also expected to use the system to view the election's result, when released, and to verify that their vote was counted correctly.

#### 4.1.3.2 Part-time Student Officers

As stated in the problem analysis section, part-time elected student union officers (PTOs) are used to supervise AGMs. As a result, PTOs will be expected to use the E-Voting system to start and manage AGM elections. Furthermore, it is important to note that candidates for an AGM are unknown until the AGM starts, therefore PTOs will be expected to use the E-Voting system to add candidates that put themselves forward. PTOs are still students therefore they will be given the lowest level administrative privileges.

#### 4.1.3.3 Election Officers

The election officers are the most senior members of staff responsible for overseeing the democratic process at the University of Surrey. They will be expected to use the system to manage and oversee all elections at the University of Surrey and will be given the highest administrative privileges. This will include, managing the accounts and privileges of other administrators as well as creating, editing, starting, ending and deleting elections.

#### 4.1.3.4 USSU Staff

USSU staff work at the USSU but are not necessarily responsible for overseeing elections. They will be expected to interact with the system to perform tasks requested by the election officer, for example creating AGM elections. However, because they are not responsible for managing elections, their ability to manage

election will be restricted. For example, USSU staff will not be able to delete or edit elections. Too add on, the privileges of USSU staff members can be changed by election officers.

#### 4.1.4 Operating Environment

React along with node.js was used to create the E-Voting system as a web application. This allows for the E-Voting system to function on mobile phones and computers that can run a web browser. More specifically the device must be able to connect to the internet. According to react official docs, react application will function on all modern browsers, including Internet Explorer 9+, chrome or Firefox.

#### 4.1.5 Constraints

The Hyperledger Blockchain that the E-Voting system will leverage will be set up using IBM Cloud (view Technology Review in **Chapter 2: Research**). This was selected as it allows for the blockchain to be deployed and utilised freely for a one-month period. Upon the completion of this project the USSU will be asked to continue paying any fees associated with running the Blockchain on the IBM Hyperledger Cloud. Due to this, the E-Voting system will be implemented using three nodes. If the USSU wish to continue using the system more nodes will be required to guarantee security.

The E-Voting System produced will be deployed using Netlify and Heroku, which allow for free deployment. Further, the USSU will be advised to use the University's deployment own deployment servers if the wish to continue using the system.

#### 4.1.6 Assumptions and Dependencies

This requirement specification was produced under the assumption that IBM Hyperledger Blockchain will continue to be free for a one-month period. Additionally, as this E-Voting System was developed using react (a Facebook framework) it is assumed that Facebook will continue to keep react available and open-source.

### 4.2 System Requirements

The proceeding section details the codified requirements for the new E-Voting System proposed in this project. The requirements have been split into two categories: functional requirements and non-functional requirements. Functional requirements describe what the system is expected to do. Meanwhile, non-functional requirements describe the systems attributes which include: performance, accuracy, security, software quality and communication.

## 4.2.1 Functional Requirements

As mentioned in **Chapter 1: Introduction** the system was developed using an agile approach. This means that, a set of requirements were specified at the beginning of each sprint of the application. The requirements deemed most important were identified and implemented in earlier sprints. These requirements will be used to conduct functional testing (see **Chapter 8: System Testing**).

For the purposes of this report each functional requirement has been given a:

1. Unique identifier – a unique numerical key that will be used to refer to the requirement in the proceeding sections of this report.
2. Synopsis – a short and relevant title for the requirement.
3. User Class – the user(s) to use the requirement (all users are identified in User Classes and Characteristics).
4. Sprint – the sprint that the requirement will be completed in.
5. Description – a description of the requirement use case.

ID	Synopsis	User Class	Sprint	Description
FR1	Log in to account	All Users	1	All users of the application must be able to log into their account using valid credentials.
FR2	Log Out of account	All Users	1	All users of the application must be able to log out of their account
FR3	Create Student Account	Election Officers	1	Election Officer(s) must be able to create new Student accounts, with a first name, last name, email, password and privilege.
FR4	Create Admin Account	Election Officers	1	Election Officer(s) must be able to create new Admin accounts, with a first name, last name, email, password and privilege.
FR5	Change Password	All Users	1	All users must be able to change their own password.
FR6	Create AGM Election	Election Officers, USSU Staff	1	Election Officer(s) and USSU staff must be able to create elections with an election name, description, start and end date.
FR7	Edit AGM Created Elections	Election Officers, USSU Staff	1	Election Officer(s) and USSU staff must be able to edit the attributes (name, description and date) of a created AGM election.

FR8	Delete AGM Elections	Election Officers	1	Election Officer must be able to delete any election on the system.
FR9	View all Elections	Election Officers, USSU Staff	1	Election Officer(s) must be able to view all created elections on the system
FR10	Create AGM contest	Election Officers, USSU Staff, PTOs	1	Election Officers, USSU staff and PTOs must be able to create AGM contests. Each contest must have a contest name, description(optional), list of candidates.
FR11	Edit AGM contest	Election Officers USSU Staff	1	Election Officers, USSU staff and PTOs must be able to edit the attributes (name, description and list of candidates) of a created AGM contest.
FR12	Start AGM Election Contest	Election Officers, USSU Staff, PTOs	1	Election Officer(s), USSU staff and PTOs must be able to start the contest, making it available for students to vote in.
FR13	Generate QR code	System	2	The system must be able to generate a unique QR code for each AGM contest
FR14	Generate Voting link	System	2	The system must be able to generate a unique voting link for each AGM contest
FR15	Access AGM via QR code	Students	2	Students must be able to access the election by using their device to scan the QR code of the election or by using the voting link
FR16	Vote in AGM Election contest	Students	2	Students must be able to cast a vote for a candidate in the AGM election.
FR17	Record Votes to Blockchain	System	2	Votes casted in the election by students must be stored in the system's Hyperledger blockchain
FR18	Tally all votes correctly	System	2	The system must be able to count all votes for each candidate correctly.
FR19	Change Admin Privileges	Election Officers	2	Election Officer(s) must be able to add and remove administrative privileges from USSU staff and PTOs.



FR20	End AGM Contest	Election Officers, USSU Staff, PTOs	2	Election Officer(s), USSU staff and PTOs must be able to end AGM contests, stopping new students from voting.
FR21	Make AGM Result Visible	Election Officers, USSU Staff, PTOs	3	Election Officer(s), USSU staff and PTOs must be able to make the results of an AGM contests, visible to students
FR22	View AGM Results	All Users	3	All users must be able to view the result of an AGM contest.
FR23	Verify Vote	Students	3	Students must be able to verify that their vote was counted correctly by the system

## 4.2.2 Non-Functional Requirements

Below is a table showing the non-functional requirements for the purposes of the report each requirement has been given:

- Unique identifier – a unique numerical key that will be used to refer to the requirement in the proceeding sections of this report
- Type – the category the non-functional requirement falls into (performance, accuracy, security, software quality, reliability, semi-privacy and communication)
- Description – a description of the requirement use case

ID	Type	Description
NFR1	Performance	Request from users to the server must be carried out in a reasonable amount of time. Additionally, the application should use loading icons when pages are being loaded.
NFR2	Accuracy	User-input must be validated by the system to ensure the data submitted is in the correct format, for example: name fields can only accept an input consisting of letters.
NFR3	Software quality	The UI must be intuitive and user friendly. The UI has a constant theme, style and colours. For example, the buttons and fonts are the same across the application.
NFR4	Software quality	The UI must response to invalid inputs into the system in a clear way.

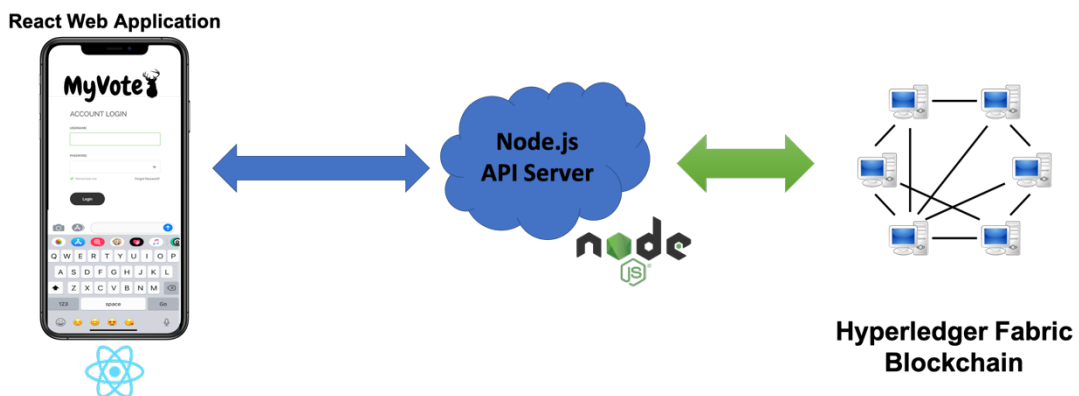
NFR5	Security	The system must comply with the data protection act (2018) and all the security controls outlined in <b>Chapter 6: Security Analysis</b> must be implemented.
NFR6	Security	Only the specified User Class (with the correct privileges) should be able to perform the functional requirements assigned ( <b>Chapter 6: Security Analysis</b> ).
NFR7	Reliability	All users should be able to use the system at any hour in the day, 7 days a week.
NFR8	Semi-privacy	All student votes and personal information should be kept confidential(full-privacy). If the Election officer decides enable verification, students should be able to verify how they voted(semi-privacy).
NFR9	Communication	All three components of the system must be able to communicate with each other. More specifically, the React web application must be able to send requests and receive responses from the node.js server, and the node.js server must be able to send requests and receive responses from the Hyperledger Blockchain.

## Chapter 5: Design

This Chapter will focus on how the new E-Voting system will function to ensure that the requirement specified are met. Furthermore, a relevant set of diagrams were used for each component to provide a visual representation of the system.

### 5.1 System Architecture

As shown in **Figure 26** which was shown in **Chapter 4: Requirements**, the E-Voting system is made up of four main components. Below is a brief description of how each of the components will function.



**React Web Application** – The react web application will act as the systems front-end, allowing users to interact directly with the application. The react application will send requests to the node.js server to retrieve and store data in the blockchain. The react application was further broken down into four subsystems. Use Case diagrams were produced for each subsystem. Following this, UI Mock Ups, activity diagrams and sequence diagrams were produced as design documentation to support the development of the application.

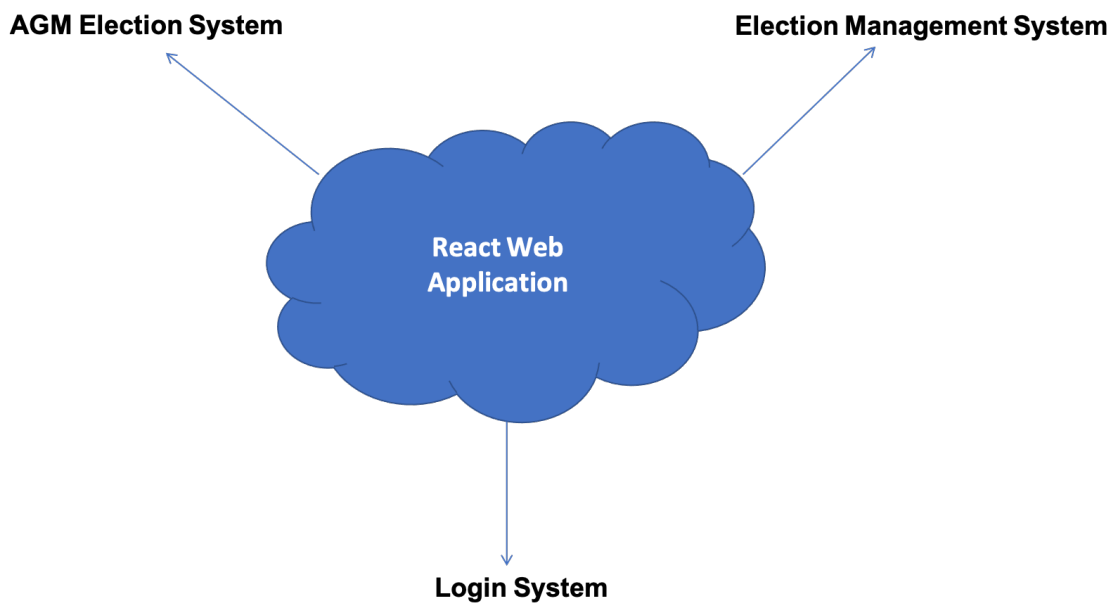
**Node.js Server** – The node.js server will act as the systems back-end, receiving and responding to requests sent by the react web application. The node.js server will contain most of the business logic of the application, and will query the Hyperledger blockchain when necessary. A table showing the endpoints that the node.js server can be found in API design.

**Hyperledger Fabric Blockchain** – The Hyperledger fabric blockchain will act as the systems primary database. It will store user credentials, elections, election contests and votes of students in the election. The architecture as well as a brief description of the blockchain was produced to show the blockchain's functionality. An entity relationship diagram was produced to show the entities, attributes and relationships that will be stored and maintained by the blockchain.

## 5.2 React Web Application Design

This section explains how the react web application component of the E-Voting system functions.

As shown in **Figure 27**, the Web Application was broken down into three subsystems.



*Figure 27: React Web Application Subsystems*

#### **Login subsystem**

The login subsystem, allows all users to login to the application. This is an important subsystem because in order to use the functionalities of the E-Voting system users must log in to their account. This component will also encompass the Administrators being able to change the credentials of other users.

#### **Election management subsystem**

The Election management subsystem, allows administrators to create, edit and delete AGM elections. This subsystem will also allow administrators to release the results of the election. In addition, the admin will be able to reveal to students how their vote was counted in the AGM election, allowing students to verify their vote.

#### **AGM election subsystem**


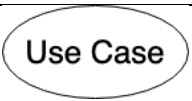




The AGM election subsystem, allows administrators to configure AGM elections, students to vote in them and view the results once they are released.

### **5.2.1 Use Case Diagram**

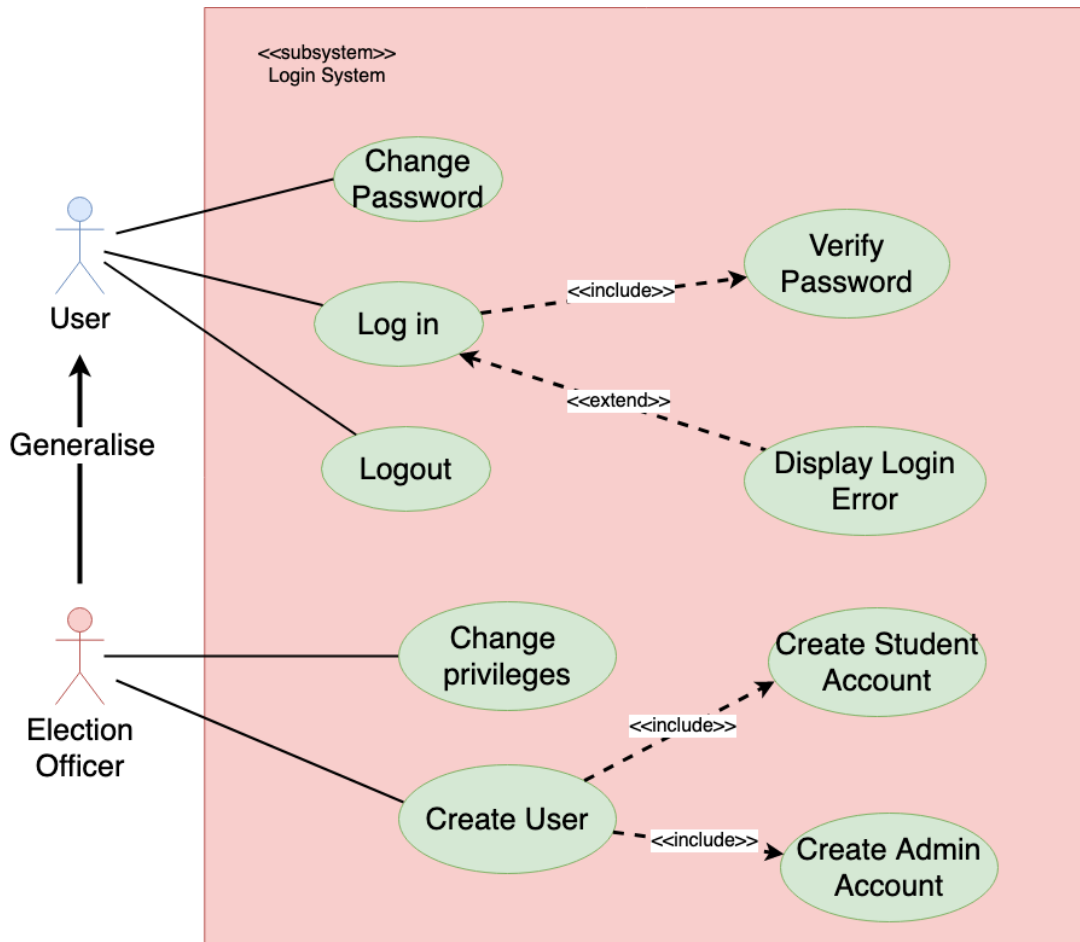
Use Case diagrams are behavioural UML diagrams that are used to summarise details of a system. They are ideal for showing how entities (known as actors) will interact with the system in order to perform a specific action. The actors used in the use case diagram correlate with the user classes mentioned in **Chapter 4:**

**Requirements.** In the design of the proposed E-Voting system, use case diagrams were chosen to demonstrate how the users will interact with the different subsystems in the web application.

The table denotes what the symbols in a Use Case diagram mean:

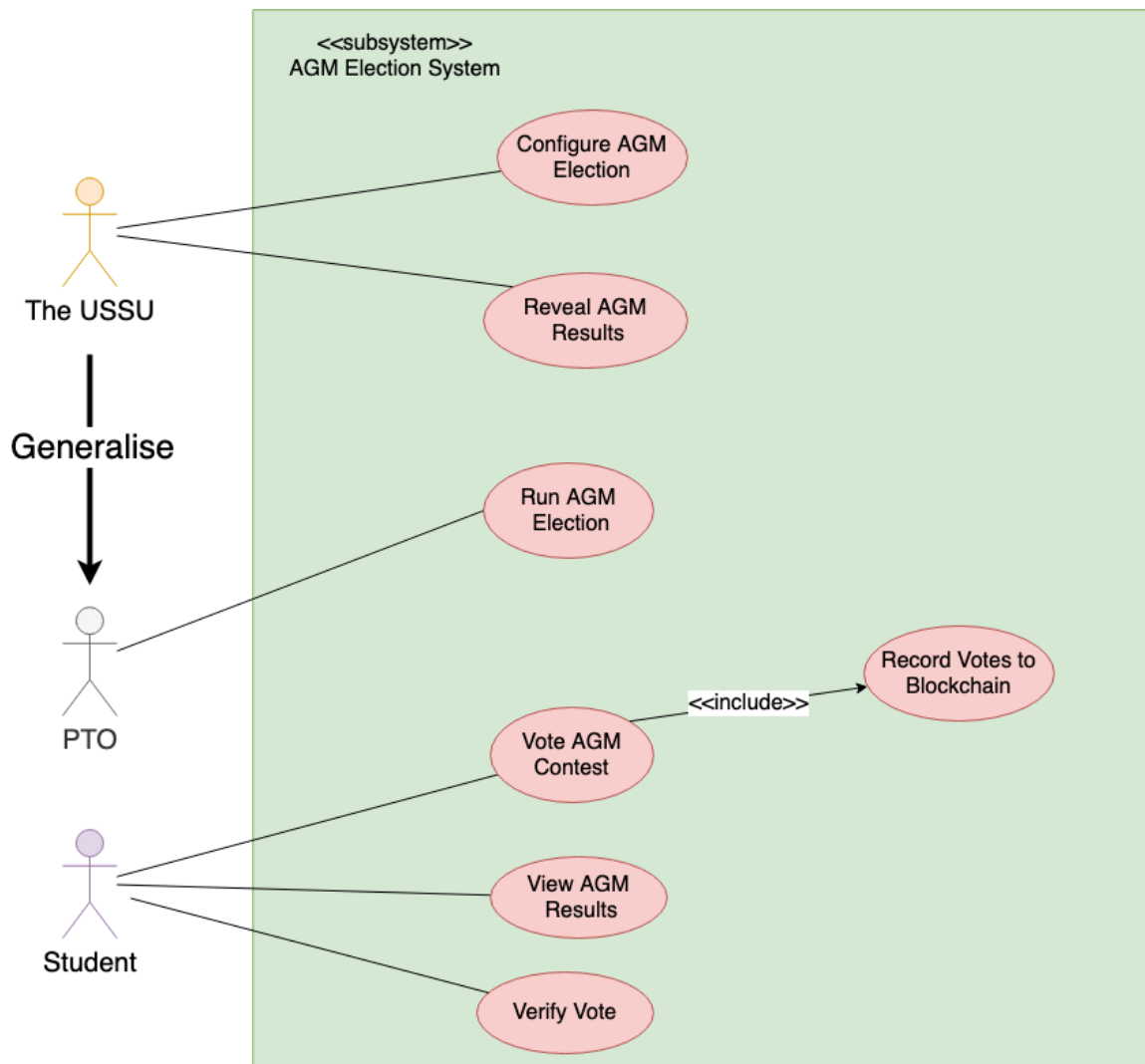
Symbol	Name	Description
 <p style="text-align: center;">Actor</p>	Actor	The person or entity that wants to perform or fulfil the associated use cases.
 <p style="text-align: center;">Use Case</p>	Use Case	The action/functionality that the system will allow associated actors to perform.
	Associate	Used to connects an actor to a use case.
	Include	Used to connect two use cases, every time the base use case is executed the include use case is executed as well.
	Extend	Used to connect two use cases, every time the base use case is executed the extend use case will happen sometimes.
	Generalise	Used to connect two actors or two use cases. The general use case acts as a parent and its children (known as specialised use case) inherit all its behaviours. Specialise use case have their own behaviour in addition to the behaviour they inherit.

**Figure 28, 29** and **30** below provide a high-level overview of all the use cases that can be performed by the different actors using the system. The actors used in the diagrams have already been identified and explained in **Chapter 4: Requirements** under user classes. Each diagram represents a different subsystem of the web application.



**Figure 28: Use Case Diagram for Login Subsystem**

As shown in **Figure 28**, the user actor which represents all users of the system, can log in and logout of the system. When a user attempts to log in the system will always verify their password, which is represented by the includes arrows. Furthermore, on some occasions (when the user incorrectly enters their password) the system will display a login error, which is represented by the extend arrow. Election officers are children of the User actor, inheriting all of its behaviours and is represented by the generalise arrow. Election officers have the additional behaviour of being able to change privileges of other users in the system. In addition, Election officers will have the ability to create new user, which can either be students account or admin accounts.



**Figure 29: Use Case Diagram for AGM Election Subsystem**

As shown in **Figure 29**, the AGM Election subsystem has three main actors: the USSU, PTO, Students. The actor ‘the USSU’ refers to, Election officers and USSU staff. These two actors are described in **Chapter 4: Requirements**. PTOs are low level administrators with privileges that allow them to run already configured elections. Unlike the USSU actor, PTOs cannot configure AGM elections. This relationship between PTOs and the USSU was modelled by assigning the shared behaviours to the PTO actor and then allowing the USSU actor to inherit these behaviours. Students are able to use the AGM subsystem to vote in AGM election. Their vote will immediately be stored in the blockchain by the system which is represented by the includes arrow. Once the USSU has decided to release the results of the AGM Election students will be able to view the results.

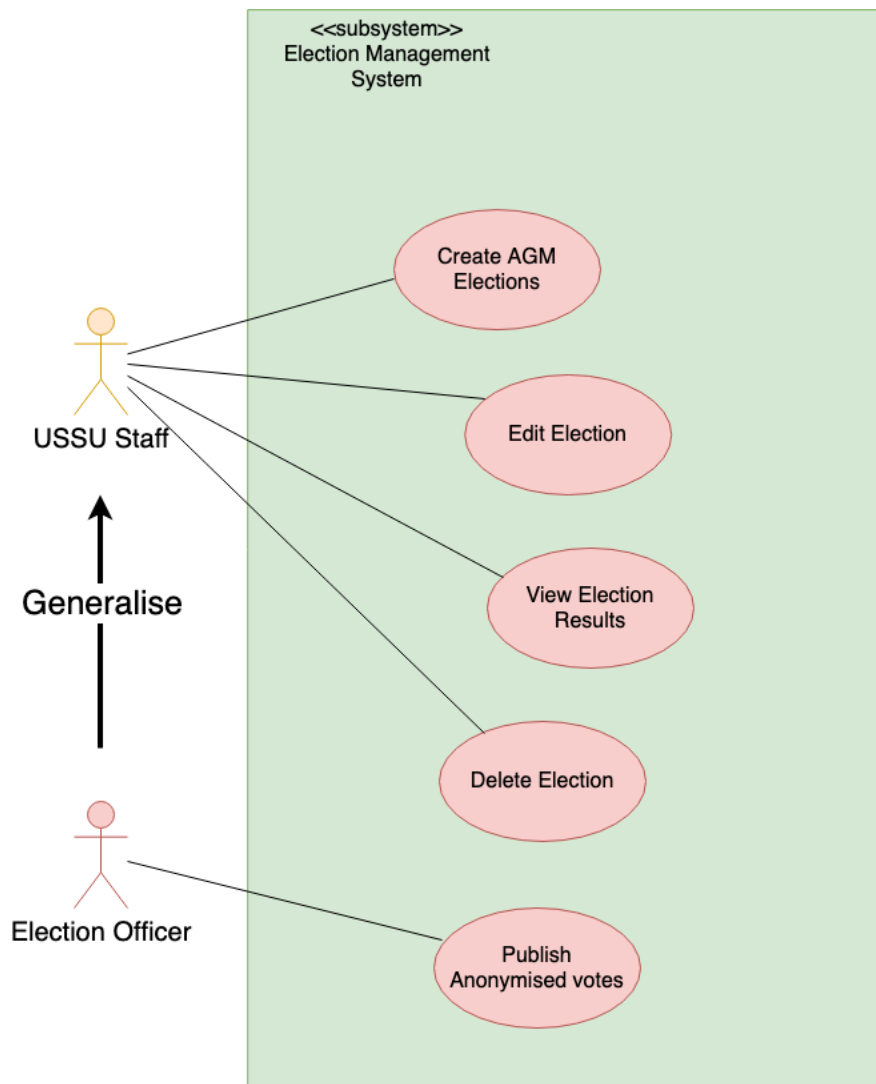


Figure 30: Use Case Diagram for Election Management Subsystem

As shown in **Figure 30**, the Election management subsystem has two actors: Election Officer and USSU staff. These two actors are described in **Chapter 4: Requirements**. Election officers have special privileges that allow them to reveal how a student voted in the election. USSU Staff can create, edit, delete and view elections.

## 5.2.2 Activity diagrams

The Use Case diagram above only offers a high-level overview of how users can interact with the system. In addition, it is still unclear how AGM elections will be carried out in this proposed E-Voting System. **Chapter 1: Introduction** and **Chapter 4: Requirements** have only briefly discussed what AGM. Therefore, to better understand how the E-Voting system proposed in this project will fulfil its most important use case, the facilitation of elections, activity diagrams produced. Activity diagrams are behavioural diagrams used to



describe aspects of a system. They are similar to flowcharts in the notation they use, and allow for the modelling of advanced behaviours. More specifically the activity diagrams below show, how the E-Voting system will facilitate, the creation, configuration and running of AGM.

As shown in **Figure 31** AGM Elections can be broken down into sequential steps. The first step which can be carried out by Election Officers or USSU staff is to create the AGM Election. When AGM elections are created the system will check that the parameters required to create an election have been entered correctly by the user. Once this is done, the election can be started, by the election overseer (typically a PTO). Upon starting the AGM Election, the individual overseeing the election will need to create an election contest. An AGM contest is an election in which students can vote for specified candidates. Under AGM elections a contest can only be created once the election has been started. The election overseer will be required to enter the name of the contest as well as the names of the candidates that have put themselves forward. If these parameters are entered correctly the election overseer can proceed to start the AGM contest. A voting QR code will then be generated by the system. This will allow students in the room to use their phones to scan the QR code to gain access to the AGM Contest. When the election overseer is satisfied that all students have voted they will be able to end the AGM Contest and display the results. The election overseer will then have two options to continue the AGM with a new AGM contest to elect another student, or end the AGM Election. If the former is chosen the Overseer will have to create another AGM contest and follow all the proceeding steps. If the latter is chosen the AGM election ends.

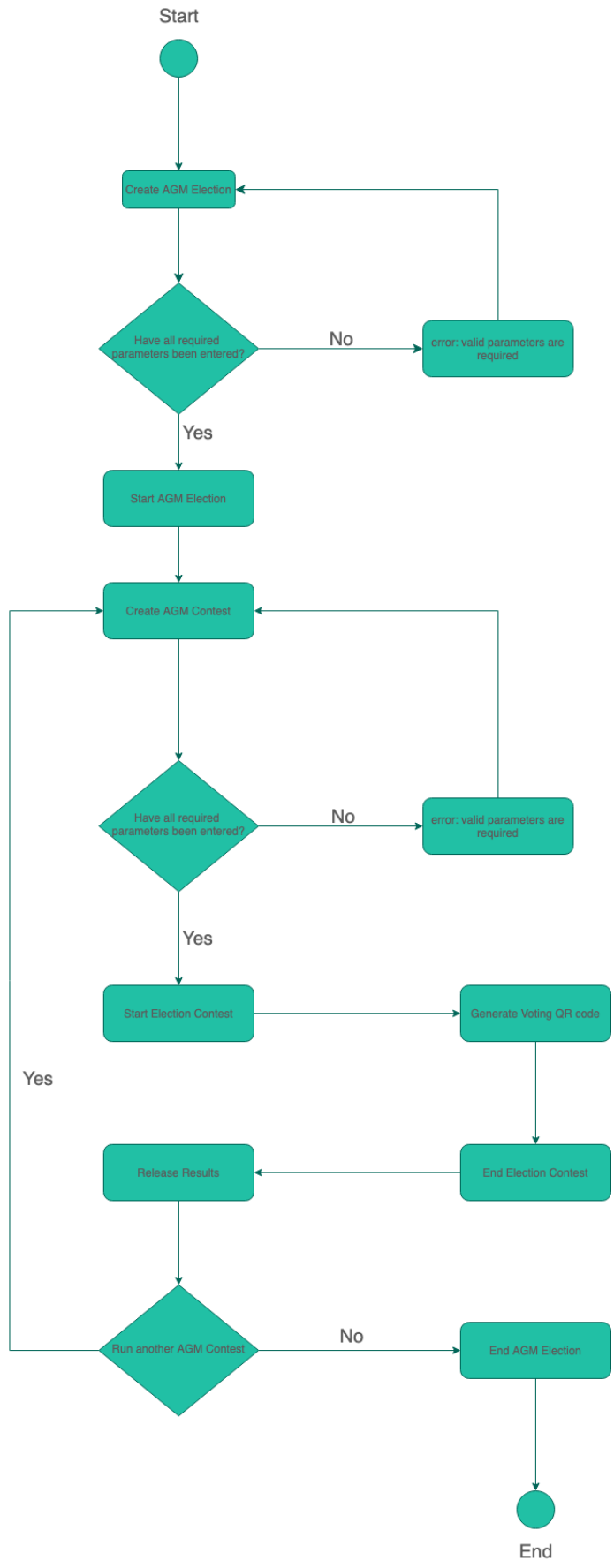


Figure 31: Activity Diagram for AGM Elections

### 5.2.3 Sequence diagrams

The activity diagrams produced, provide a walkthrough of how AGM elections will function on the proposed E-Voting system. However, it is still unclear how actions initiated on the client will be communicated to the server-side and database of the E-Voting system. For instance, what messages are sent between the react web application, the node.js server and the Hyperledger blockchain database, when the user clicks a button. To make it clear how actions triggered by users are completed from end to end by the system, sequence diagrams were produced. Sequence diagrams are another behavioural UML diagram that details how operations are carried out between different components within a system.

A total of two sequence diagrams were produced. The first details how elections are created and stored in the Hyperledger Blockchain and the second details how students can login and vote in elections (with their votes being stored in the Hyperledger Blockchain).

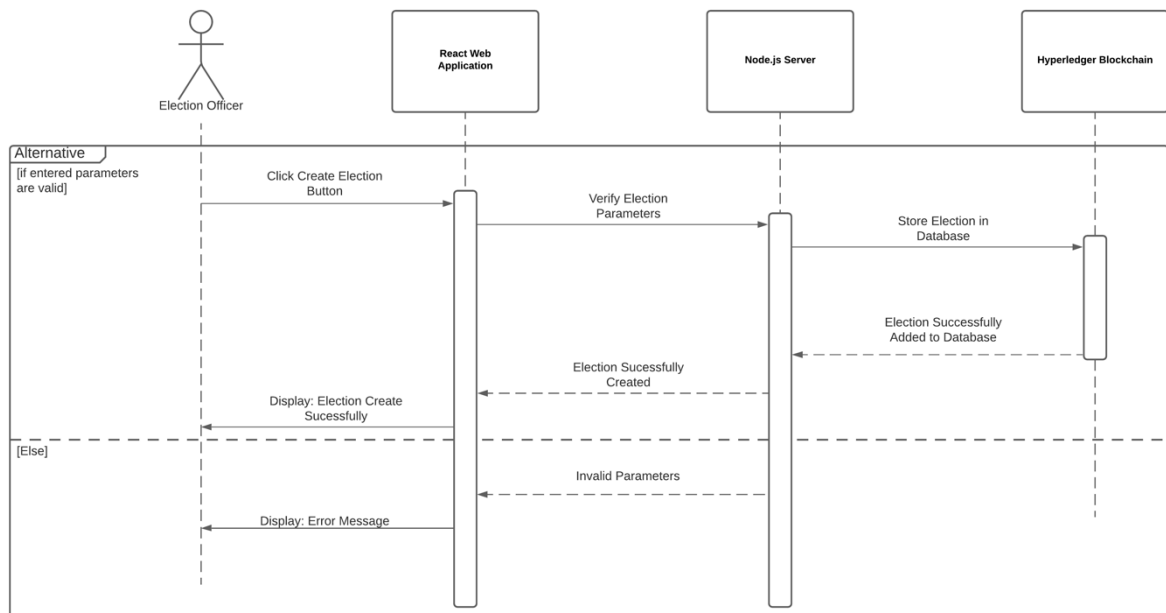
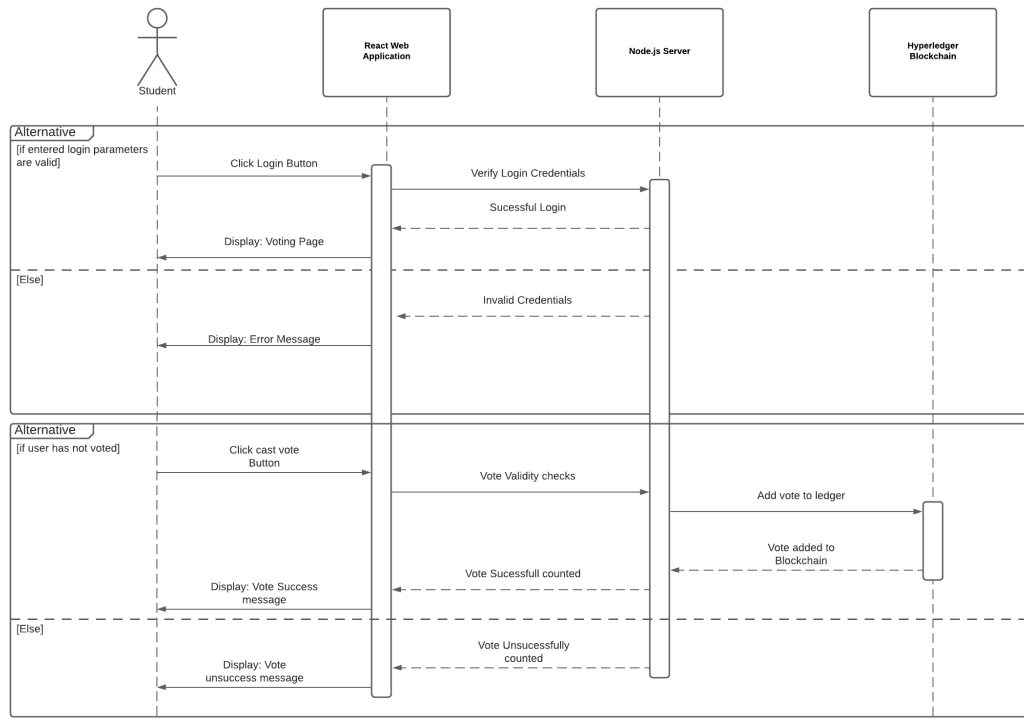


Figure 32: Sequence diagram showing students vote

As shown in **Figure 32** when the user clicks the create election button the react application calls the createElection endpoint in the node.js server. The node.js server will verify that the parameters are valid and then store the created election in the Hyperledger Blockchain database. If the node.js considers the parameters to be invalid then it will send an error message back to the react application. The react application will then display the error message to the user. For more information on the createElection endpoint and the parameters it will accept see API design.



**Figure 33: Sequence diagram showing students vote**

As shown in **Figure 33** when the user clicks the login button the react application calls the login endpoint in the node.js server. The node.js server will verify that entered login credentials are valid. If so, the user will be redirected to the voting page. If the node.js deems the credentials to be invalid then it will send an error message back to the react application. The react application will then display a login error message to the user. For more information on the login endpoint and the parameter it will accept see API design. Once logged in the user can select the candidate they wish to vote for and click the cast vote button. When the button is clicked the castVote API will be called and the node.js server will verify that the user has not voted before. If the validity check is successful the vote will be added to the Hyperledger fabric blockchain ledger. If validity check is unsuccessful the node.js server will display send an error message back to the react application, which will be displayed to the user.

## 5.2.4 User Interface Design

The user interface is extremely important, as it will be the part of the application users directly interact with. Below details how each page in the system will look, as well as the colour scheme chosen for the application.

### 5.2.4.1 Logo

**Figure 34** is the logo that was design for my application. The Stag icon was used in the logo because it is the mascot of the University of Surrey and the USSU.



*Figure 34: Logo for the E-Voting System*

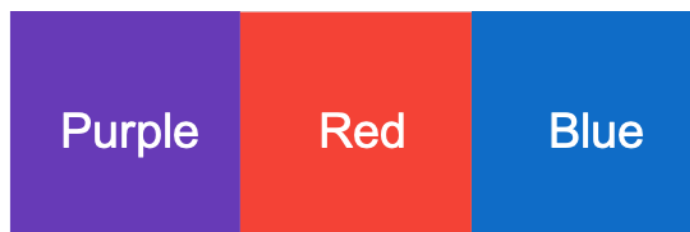
### 5.2.4.2 Colour Scheme

While it would have been appropriate to ask users during focus groups what colours they would like the E-Voting system to be, it is likely to yield a negative result. This is because students are likely to respond with their favourite colours as opposed to colours suitable for an E-Voting Application. Therefore, the application will consist of colours commonly used on the University of Surrey and the Students Unions websites. These colours include:

Purple: which is commonly associated with royalty and nobility will be the primary colour of the application.

Blue: which is a conversative colour associated with security will be the secondary colour of the application.

Red: which represents danger and aggression, this will be used to highlight important messages, error messages and any problems within the application.



*Figure 35: Colours Scheme used for the E-Voting system*

### 5.2.4.3 UI Mock-ups

UI mock-ups were produced for the E-Voting system. The mock ups below only show what the application would look like on a mobile phone. This is because as shown in the survey conducted in **Chapter 4: Requirements** most people will vote via their mobile devices. Having said this the web application is responsive so will scale up and still look aesthetically pleasing on a laptop and other devices. Figures below illustrate the UI mock-ups.

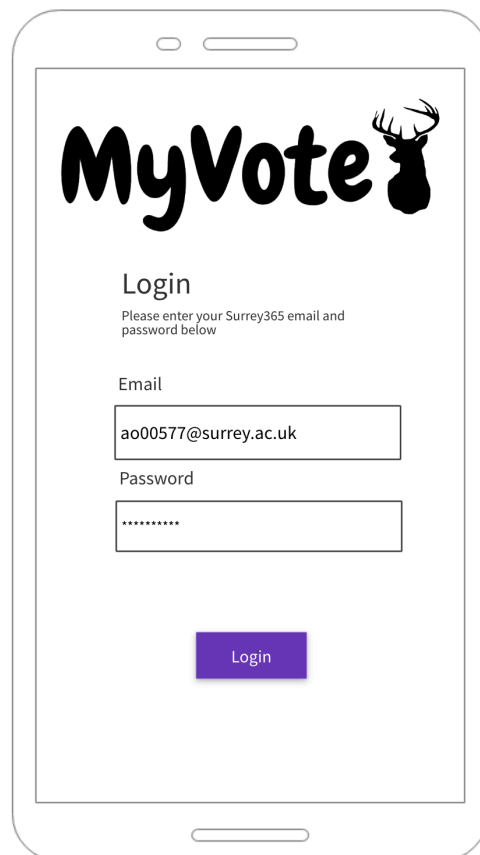


Figure 36: Mock Up of the Login Page

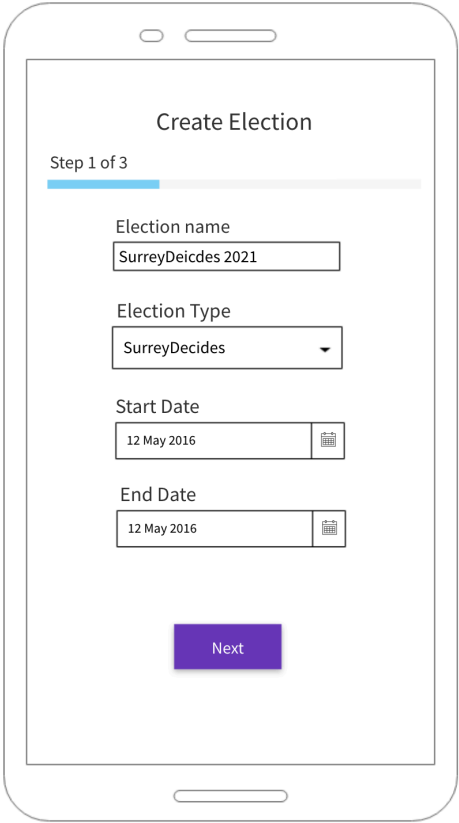


Figure 37: Mock Up of Create election Page

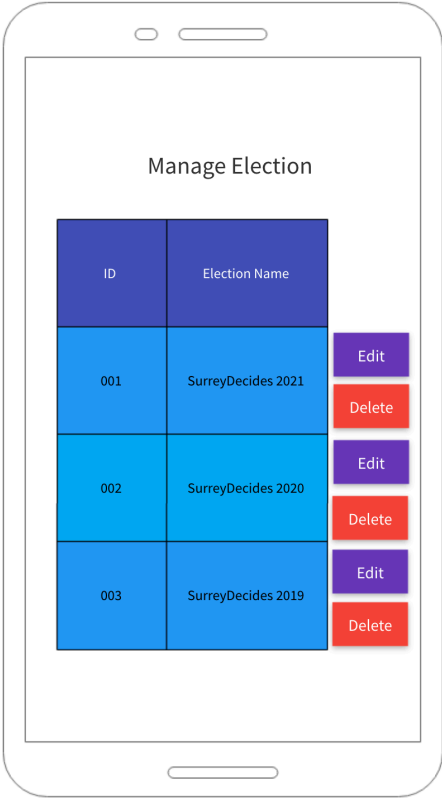


Figure 38: Mock-up of Manage Election Page

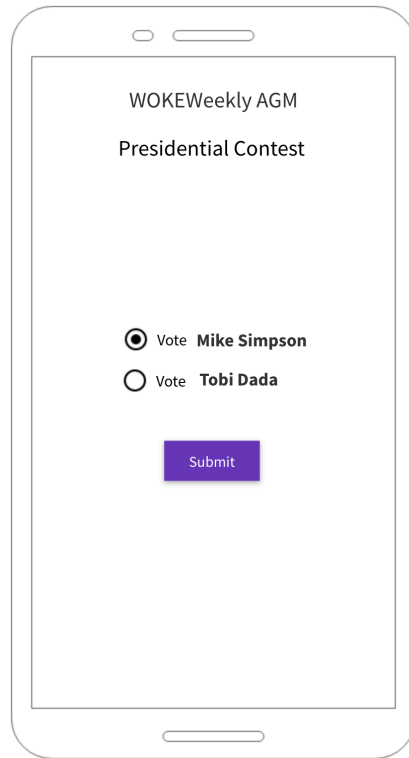


Figure 39: Mock-up of AGM Voting Page

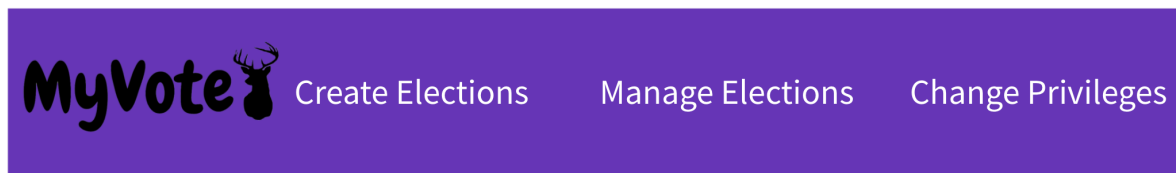


Figure 40: Mock-up of Header/Navbar of E-Voting System

The UI is subject to change during the development process of the application. The actual UI is can be found the appendixes.

## 5.3 Node.js Server Design

This section will explain the how the node.js server will interact with the react web application. For details on how the node.js server will interact with the Blockchain ledger see public key infrastructure. An express node.js server was used to develop the systems server-side. Furthermore, the sequence diagrams in the proceeding section showed that the react application will send messages to the node.js server. These messages will reach endpoints that will be located in the express node.js server. **Section 5.3.1 API Design** offers a description of each endpoint along with their parameters.



### 5.3.1 API Design

Below is a table showing the API endpoints that will need to be implemented in the node.js server in order for the E-Voting system to function. Because an agile approach was taken, all the designed endpoints were implemented.

Each endpoint has a:

**Name** – the name of the endpoint

**Parameters** – the parameters that the endpoint will need to receive from the client

**Type** – the type of RESTFUL request (GET, POST, PUT, DELETE)

**Description** – short explanation of what the endpoint does

Name	Type	Parameters	Description
<b>getLogin</b>	GET	–	Returns details of the user currently logged in.
<b>login</b>	POST	email, password	Checks if the users entered credential match what is stored on the blockchain, if not an error message is return. If they are correct access token is set back (as a HTTP cookie).
<b>logout</b>	GET	–	Logs a user out by removes the users access token.
<b>createElection</b>	POST	electionName, electionType, startDate, endDate, electionDescription	Creates an election with the specified parameters and store it in the Blockchain. Return a message denoting whether or not the election has been created successfully.
<b>createAGMContest</b>	POST	electionID, contestName, contestDescription, candidates	Creates an AGMContest for the specified election and with the specified parameters. Return a message denoting whether or not the election contest has been created successfully.
<b>updateContest</b>	PUT	newValue, attribute, electionID, contestID	Updates a specified electionContest with new values. Returns a message denoting whether or not the election contest has been updated successfully.

<b>updateElection</b>	PUT	newValue, attribute, electionID	Updates a specified election with new values. Returns a message denoting whether or not the election has been updated successfully.
<b>changeElectionStatus</b>	PUT	electionID, contestID, newStatus	Changes the election status from (from closed to open and from open to ended) Returns a message denoting whether or not the election has been updated successfully.
<b>showAGMResult</b>	GET	electionID, contestID,	Calculates the results by counting all the votes in the ballot box and returns the election results to the client.
<b>getElectionContest</b>	GET	electionID, contestID,	Retrieves the specified election contest from the blockchain ledger and returns it to the client side.
<b>getAGMResults</b>	GET	electionID, contestID,	Retrieves the ballot box from the blockchain ledger and then Calculates the results by counting all the votes in the ballot box and returns the election results to the client.
<b>getHasVoted</b>	GET	electionID, contestID, studentEmail	Returns whether or not a specified student has voted or not.
<b>deleteElectionContest</b>	DELETE	electionID, contestID,	Deletes election contest from blockchain.
<b>deleteElection</b>	DELETE	electionID	Deletes election from blockchain.
<b>getAllElections</b>	GET	–	Returns all elections from the blockchain.
<b>getElection</b>	GET	electionID	Retrieves the specified election from the blockchain ledger and returns it to the client side.
<b>castVote</b>	POST	electionID, contestID, ballot (studentEmail and candidateId)	If the user is a valid voter, and has not previously voted. The vote will be sent to the blockchain, a message denoting whether the vote has been counted will be returned to the client.

<b>createUser</b>	POST	email, firstName, lastName, password, privilege, userType	Creates a user with the specified parameters and stores it in the Blockchain. Return a message denoting whether or not the user has been created successfully.
<b>updateUser</b>	PUT	newValue, attribute, userEmail	Updates a specified user in the blockchain with new values. Returns a message denoting whether or not the election has been updated successfully.
<b>deleteUser</b>	DELETE	userEmail	Deletes user from blockchain
<b>getAllUsers</b>	GET	–	Returns all users from the blockchain.

## 5.4 IBM Blockchain Cloud Design

As stated in **Chapter 2: Research** it was decided that a private Blockchain would be used to implement the E-Voting system. Furthermore, the chapter also explains why IBM’s Blockchain platform was chosen over alternatives to create and manage the Hyperledger Fabric blockchain for the application.

**Figure 41** below shows the architecture of the blockchain ledger.

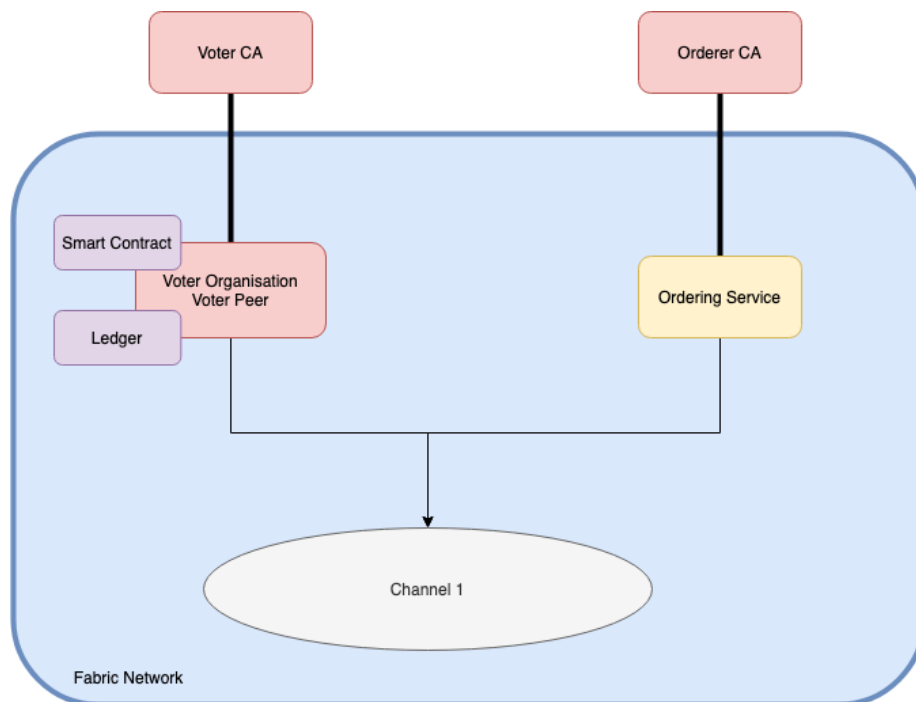


Figure 41: Hyperledger Blockchain Architecture

**Voter CA** – The Voter CA is the certificate authority that is responsible for enrolling new identities (users) onto the blockchain. Once enrolled, users can request transaction certificates from the voter CA. These certificates can be used to invoke chaincode transactions on the blockchain. Enrolment and verification of certificates is executed using a public key infrastructure (which is explained in **Chapter 6: Security**).

**Orderer CA** – The orderer CA is the certificate authority that is responsible for the enrolment of the orderer server.

**Ordering Service:** The orderer node is used to decide the correct order of transaction within a blockchain. This is important because relies on a deterministic consensus algorithm, meaning that ledger cannot fork the way many permissionless blockchain networks do.

**Voter Peer:** The voter peer is a node on the blockchain that is used to execute Chaincode transactions and record transactions. The voter peer also stores the smart contract deployed onto the blockchain and the world state (ledger).

**Channel 1:** Is a private subnet of communications between two or more specific network members. These members include: all the peers and ordering services in the network, as well as the blockchains ledger and Chaincode. Each transaction on the network is executed on a channel. The parties that are allowed to execute Chaincode on the channel are determined by the Voter CA.

**Smart Contract** – See Smart Contract Design below.

**Ledger** – See Database Design below.

## 5.4.1 Database Design

As stated above the ledger where data is stored within a blockchain. The ledger in Hyperledger Fabric consist of two parts:

The world state – A database that holds the current values of a set of ledger states. The world state makes it easy for the Node.js server to directly access and appended data to the blockchain.

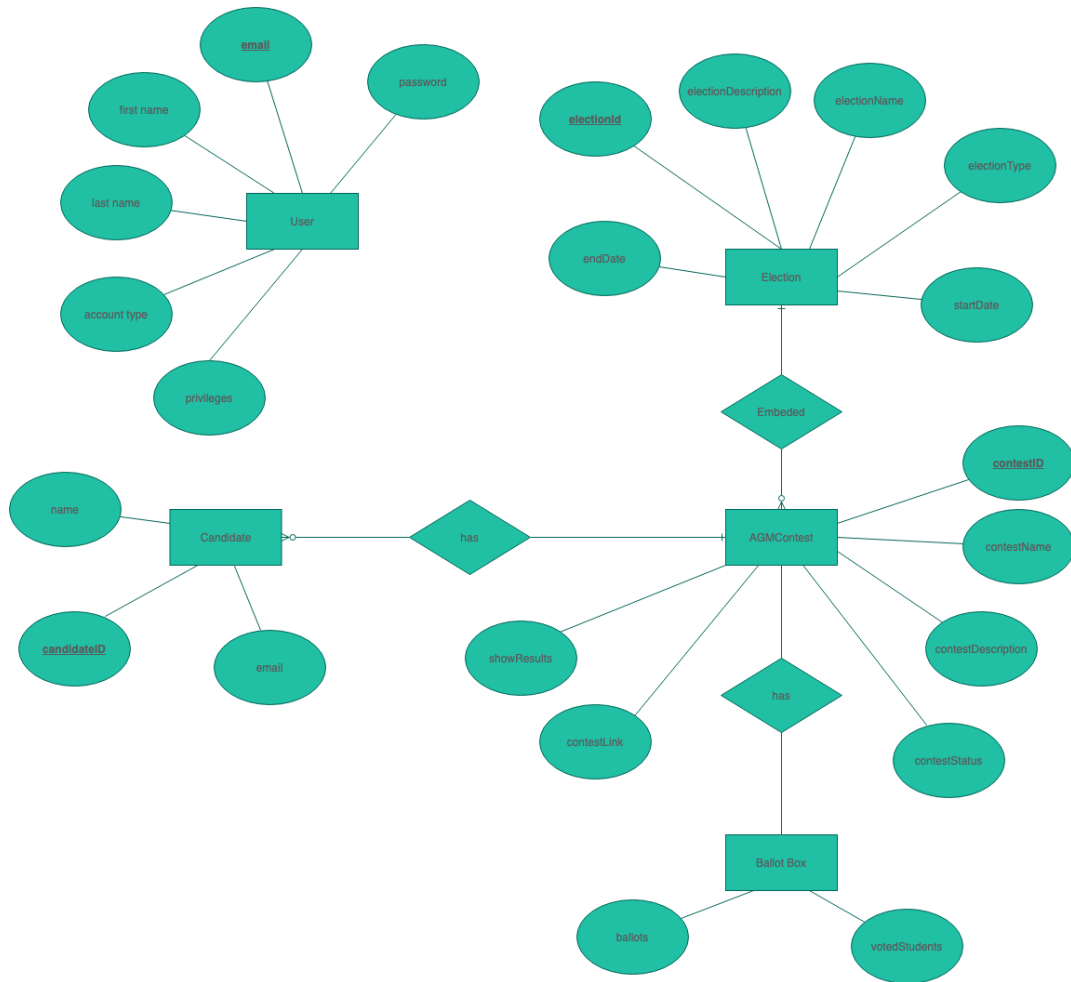
Transaction Log – The history of transaction that resulted in the current world state current values.

Hyperledger Fabric expresses the data in the ledger as key-value pairs. IBM Blockchain gives us the option to either use LevelDB or CouchDB, both of which have pros and cons.

LevelDB is NoSQL database that stores data as key-values. LevelDB allows for basic operation such as get, put and delete to be performed on the blockchain. Keys and values can be any byte array and not just strings. By default, LevelDB is used by the blockchain and is appropriate when ledger states are simple key-value pairs.

CouchDB is also a NoSQL database that stores data as a JSON documents. This is more suitable for my application as data will be sent from the react web application to the node.js server as JSON object. Furthermore, Node.js makes working with JSON object simple. In addition to this CouchDB supports the rich queries and update of richer data types often found in business transaction. Implementation-wise, IBM Blockchain Platform allows for CouchDB can be enabled as the ledger’s database.

**Figure 42** is a diagram that represents the objects stored the CouchDB database.



**Figure 42: Diagram showing Objects in CouchDB Database**

As shown in **Figure 42**, the CouchDB database will store two types of main objects:

**User** – The purpose of the User object is to store the login credentials and personal information of the users using the application, such as their email, password, first name, last name, account type (Student or Admin) and privileges (1,2 or 3). A description of the User Object’s attributes are as follows:

- **Email:** is the unique ID for a User Object, and will be required to log the user into the system. The email attribute must be a valid email address.

- **Password:** is a secret string known only to the owner of the account. It will be required to log the user into the system.
- **FirstName:** is a string containing letters only which holds the account owners first name.
- **LastName:** is a string containing letters only which holds the account owner's last name.
- **AccountType:** is a string that can either be Student or Admin. The account type will determine what pages on the system the user has access to (see **Chapter 5 Security Analysis**).
- **Privileges:** is an integer that can either be 1,2 or 3. The privilege of an account will determine what the user will be able to do on the system, for example an admin with a privilege of 3 will be able to delete accounts. (See **Chapter 5 Security Analysis**).

Election – The purpose of the Election object is to store information about elections created by Admins. This information includes: electionId, electionName, electionDescription and election type, start date and end date. The Election object will also consist of multiple AGMContest objects (see below). A description of the Election object's attributes are as follows:

- **ElectionId:** is the unique ID for an Election Object. The electionID is generated by the system.
- **ElectionName:** is a string containing letters. It will hold the name of the election, which will be displayed on the User's voting screen.
- **ElectionDescription:** is a string containing letters and spaces. It will hold the name of the election short description of the election.
- **ElectionType:** is a string that can only be set as AGM. This attribute was created because in future iterations of the application other types of election can be implemented.
- **StartDate:** is a string in the format DD-MM-YYYY. It is used to hold the intended start date of the election.
- **EndDate:** is a string in the format DD-MM-YYYY. It is used to hold the intended end date of the election.
- **AGMContests:** See below

**Figure 42** shows that the CouchDB database also stores three types of embedded objects (objects that are stored within other objects). This includes:

**AGMContest** – The purpose of the AGMContest object is to store details about an election contest. These details include: contestID, contestName, contestDescription, contestLink, candidates, ballotBox(see below), showResults and electionStatus. The AGMContest is an embedded object as it is stored within the Election object. An AGMContest can only belong to one Election object. A description of the AGMContest object's attributes are as follows:

- **ContestId:** is the unique ID for a Contest Object. The electionID is generated by the system.
- **ElectionId:** is the election that the ballotBox embedded in.
- **ContestName:** is a string containing letters. It will hold the name of the election contest, which will be displayed on the User's voting screen.
- **ContestDescription:** is a string containing letters and spaces. It will hold the name of the election short description of the election contest.
- **ContestLink:** is a string denoting the URL link to the election contest, this is a combination of the electionId and contestName. This will be used to generate a barcode of an election.
- **Candidates:** is a list of candidates in the election contest. Each candidate will have a name, email and id. A student will be able to cast their vote for one of the three candidates.
- **ShowResults:** is a Boolean value that is either true or false. If it is false that means that only the administrator can see the results of the election. If it is true that the election results are available to all users.
- **ElectionStatus:** is a string that can either be closed, open or ended. When an Election contest is created, it is set to closed by default. This means students cannot vote in the election or view the results. If the electionStatus is set to open then students are permitted to vote in the election contest. If the electionStatus is set to ended, the student will not be permitted to cast anymore votes, but they will be able to see the results of the election if showResults is set to true.
- **Ballotbox:** See below.

**BallotBox** – The purpose of the BallotBox object is to store details about votes cast in an AGMContest. These details include: electionId, contestId, votedStudents and ballots. The ballotBox is an embedded object as it is stored within the AGMContestObject. A Ballot Box can only belong to one AGMContest object. A description of the BallotBox object's attributes are as follows:

- **ElectionId:** is the election that the ballotBox embedded in.
- **ContestId:** is the AGMContest that the ballotBox embedded in.

- **VotedStudents:** a list containing the emails of the student who have voted in the election. This is used to keep track of all the student who have voted, which prevent students from voting more than once.
- **Ballots:** A list of ballots cast in the AGMContest. Each Ballot has a caster (the email of the person who cast the vote) and a candidateld (the id of the candidate the student voted for).

## 5.4.2 Smart Contract Design

Smart contracts are transactions that can be invoked to perform actions on the blockchain. In other words, it is the business logic that allows the Node.js server to execute request and append data to the blockchain ledger. As shown in **Figure 41**, the Smart Contract will run on the Voter Peer. In Hyperledger multiple smart contracts can written, however they will have to be packaged into chain code before they can be deployed onto the application. JavaScript was selected as the language in which the smart contract would be written this is because the Node.js server is also written in JavaScript, allowing for an easier learning curve.

To keep things simple all functions that queried and appended data to the blockchain were written in a single file. In addition, four classes representing the objects that would be stored in the CouchDB database will be produced. More information about how the smart contracts were produced can be found in the implementation section.

## 5.5 Design Challenges

Aspects of designing the E-Voting system proved more challenging than others. These challenges include:

**Structuring of JSON Objects:** As explained above CouchDB will be used to store data on the Blockchain and will therefore act as the E-Voting Systems primary database. Because CouchDB is a noSQL database that uses stores data in JSON documents it was difficult to model the various objects that the Blockchain would need to store and the relationships between these objects. After reading CouchDB documentation, it appeared that relationships could be established between two objects by embedded one object within another. Once this had been discovered modelling the database become easier.

**Designing Security Controls:** The security controls outline in **Chapter 6: Security Analysis** were originally were difficult to come up with. It was only after conducting a risk assessment of the proposed design that security flaws were identified. With that said, it is difficult to mitigate all against possible attacks to the system because attacks can change and become or sophisticated overtime. Indeed, there are simply too many unknown, unknowns to deal implement an application with no security flaws. Furthermore, it is



through the application been breached by an attacker that can allow for better security controls to be developed. However, taking into consideration that the E-Voting system is operating in a low threat environment the security control outlined in **Chapter 6: Security Analysis** protect against common attacks to web applications.

**Inability to use Surrey365:** It was originally indented for all users of the application to be able to authenticate themselves using Surrey365, a Microsoft service that allows students and staff members at the university of surrey to login application such as Outlook and SurreyLearn with a single set of login credentials. This would have transferred the responsibility of securely storing user credentials to the University. In addition, using Surry365 would have mean that a user management system would not have needed allow for more time to be spent working on other aspects of the system. Unfortunately, the IT Department would not grant my application access to Surrey365, meaning that a user management system had to be implemented with necessary security criteria.

## Chapter 6: Security Analysis

---

This chapter seeks to analyse the security risks and threats the proposed E-Voting system is vulnerable to, as well as detail how security will be built into the application.

### 6.1 Introduction

While the E-Voting Application will function in a low threat environment, it is still important to identify any potential security threats that could compromise the system. The security goals of this project are based on the CIA triad:

- **Integrity:** is achieved when data remains consistent, unaltered and uncorrupted while being transmitted, processed or stored. The most valuable asset that the E-Voting system will transmit is how a student has voted. If vote can be altered during transmission the whole election will be liable to disputes by disgruntled candidates. This makes integrity the most important security goal.
- **Confidentiality:** is achieved when data stored and processed by the E-Voting system is protected from disclosure or exposure to unauthorized individuals. The E-Voting system will store and process users' personal information such as their email, full name, passwords and casted vote. Furthermore, as stated in the literature review a fundamental principle of free and fair elections is that votes are confidential. Other sensitive information such as the

election results must also be kept confidential by the system until the election officer decides to reveal the results. To achieve confidentiality users' personal information will be stored in an encrypted database, a hierarchy of access privileges will be established and JSON Web Tokens will be implemented. This means that only those authorised will be able to access certain information stored by the system. This is the second most important security goal because if the system lacks confidentiality, then students are unlikely to use the application in fear that their personal information and casted vote will be exposed.

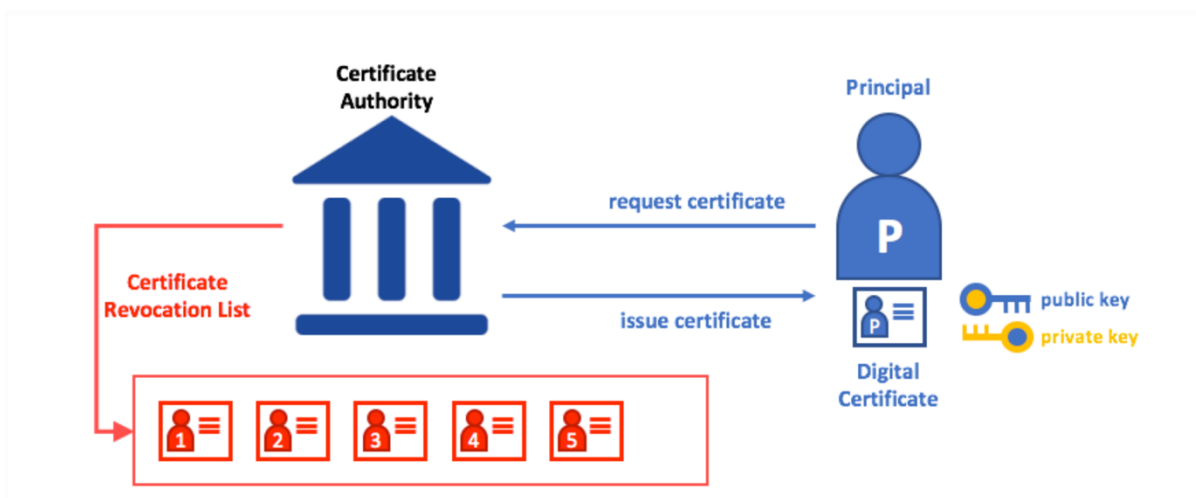
- **Availability:** is achieved when the E-Voting application is available at all times. Furthermore, data must also be accessible and correctly formatted for use without interference or obstruction. The E-Voting system must allow students to login and vote whenever they want, as long as the election is still open.

## 6.2 Blockchain Security

As stated in **Chapter 2: Research** using a blockchain to store data provides a multitude of security properties. For example, using a blockchain guarantees integrity as data is stored in sequential blocks meaning that data in the ledger cannot be changed without detection. In addition to this because Hyperledger is a private Blockchain, a public key infrastructure is used to ensure request to the Blockchain are from validate parties.

### 6.2.1 Public Key Infrastructure

Hyperledger Fabric uses a public key infrastructure to communicate with authorised parties. The Blockchain will only accept transactions invoked by requests that have been signed by a valid private key. The certificate authority is responsible for enrolling new identities into the blockchain. As shown in **Figure 43** a certificate consists of a private a public key.



*Figure 43: Diagram of Certificate Authority*

In the E-Voting application when a new user is created a script is run that asks the certificate authority to create a private and public key for the users. These keys will be added to a wallet stored on the server. Whenever the server wishes to invoke a transaction onto the blockchain, the transaction will need to be signed by the private key of the user, which will be stored in the wallet directory. If the signature is not valid then the blockchain will reject the transaction.

In addition, because HTTPS is used to send data between the blockchain database and the node.js server an adversary will have a difficult time intercepting data transmitted.

## 6.3 Risk Assessment

Risk assessments are a great way to identify potential risks to the system before development begins so that necessary controls can be built in to the system to mitigate the risks. The risk assessment below will focus on ensuring data sent from the react web application to the node.js server is secure.

The methodology used to conduct risk assessment in this project is the BS ISO/IEC 27005:2011. The latest version released in ISO/IEC 27005:2018 was not available without paying a fee.

Furthermore, the risk assessments will have three stages:

1. Risk identification – identifying risks to the system
2. Risk assessment – calculating the chance and gravity of each identified risk
3. Risk control – finding solutions that can be implemented to mitigate the identified risks

### 6.3.1 Risk Identification

To correctly identify the risk to the application the applications assets, threats and vulnerabilities must first be identified.

#### 6.3.1.1 Asset Identification

The table below shows a list of assets the application is anticipated to have.

Ref	Asset	Type	Description
A1	User credentials	Information	The users' credentials such as user passwords which are stored in a Hyperledger blockchain
A2	Casted Votes	Information	The candidate students have voted which is stored in a Hyperledger blockchain

<b>A3</b>	Election Results	Information	The results of the election which is stored in a Hyperledger blockchain
<b>A4</b>	Node.js Server	Network	The Server where API requests sent from the react application are sent to the Hyperledger blockchain
<b>A5</b>	React Web Application	Software	The client-side of the application

### 6.3.1.2 Identification of Threats

A threat is an unwanted incident, which may result in harm to an application. The table below lists the potential threats to the E-Voting application (identified above).

Ref	Threat	Description
<b>T1</b>	Data theft from Blockchain	An attack agent could gain access to data stored within the blockchain such as user passwords and casted ballots. This could happen in a multitude of ways for example by intercepting data in transmission.
<b>T2</b>	Endpoint Attack	An attack agent could invoke an endpoint if the endpoint URL is known allowing them to perform CRUD operations on the blockchain without the necessary permissions.
<b>T3</b>	DOS attacks	The server may be brought down due to high volumes of network traffic.
<b>T4</b>	Semantic URL attack	An attack agent that is a user could gain access to data and restricted pages of the application by altering the URL in their web browser.
<b>T5</b>	Packet Sniffing	An attack can intercept data sent between the client react application and the node.js server.

### 6.3.1.3 Identification of Vulnerabilities

A vulnerability is a weakness of an asset that can be exploited by a threat. Since the goal of the project is allow the USSU to conduct elections, a vulnerability is anything that hinders this.

The table below shows the vulnerability of each identified asset and the consequences to the system.

Ref	Vulnerability	Incident Scenario	Consequences
<b>V1.</b>	Un-hashed/unencrypted sensitive data	An attacker gains access to blockchain database.	Sensitive data such as election results, a user's password and ballots can be obtained by an attacker, bring the integrity of the election into question.
<b>V2.</b>	Exposed Endpoints in node server	An attacker uses the API endpoints URL to access confidential resources.	One potential consequence is that election results, which are sent to the react application can be obtained before they are released which can influence the election.
<b>V3.</b>	No contingency to handle network server downtime	An attacker conducts a DOS attack on network servers	Some students may not be able to vote in the election, which will bring the fairness of the election into question.
<b>V4.</b>	Unprotected Routes	Students logs in and changes URL in order to get access to pages restricted to administrators	Unauthorised users will be able to perform administrative tasks that can influence the outcome of the election.

V5.	Insecure Channel	An attacker can intercept data being sent from the React Web Application to the Node.js server	Sensitive data such as, a user's password and ballots can be obtained by an attacker, bring the integrity of the election into question.
-----	------------------	--	--

## 6.4 Risk Analysis

In the absence of quantitative data, a qualitative risk assessment will be conducted to assess the level of impact the identified vulnerabilities will have on the E-Voting system's ability to run elections.

The table below shows the impact of each identified vulnerability if it is successfully exploited along with the likelihood of it occurring. Since this is a qualitative risk assessment a justification is provided as to why a vulnerability was given a level. The likelihood and impact are ranked on a scale of (LOW, MEDIUM and HIGH). **Figure 44** is the matrix that shows the overall risk severity based on the impact and likelihood.

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

Figure 44: Probability Impact Chart (OWASP, 2016)

Ref	Impact	Likelihood	Justification	Overall Risk
V1.	HIGH	LOW	Impact – A high level of impact was given because if an adversary is able to get access to the blockchain they would be	MEDIUM

			<p>able to obtain a user's credentials allowing them to log in and vote on their behalf. This defeats the whole point of the election as every student should be given a single vote.</p> <p>Likelihood – A low likelihood was given because the Hyperledger Blockchain is a private Blockchain meaning that it is extremely difficult/ impossible to breach.</p>	
<b>V2.</b>	MEDIUM	LOW	<p>Impact – A Medium impact level was given because while third gain access to the election results is not ideal, it is unlikely to influence voting behaviour on a massive scale.</p> <p>Likelihood – A low likelihood was given because an attack is unlikely to know the names of the endpoints.</p>	LOW
<b>V3.</b>	MEDIUM	LOW	<p>Impact – A medium impact level was given because a DOS attack would not alter the votes that have already been casted in the election. However, it will cause disruptive and frustration to votes. Therefore, not having a contingency plan to handle network server downtime will mean some students will be unable to vote.</p> <p>Likelihood – A low likelihood was given because the E-Voting system is operating in a low threat environment. Elections at the USSU are low stake therefore it is unlikely someone will execute a DOS attack.</p>	LOW
<b>V4.</b>	HIGH	MEDIUM	<p>Impact – A High impact was given because if the student is able to gain access pages only meant for administrators, they will be able to make changes ongoing election and cause damage to the system.</p> <p>Likelihood – A medium likelihood was given because although a user is unlikely to know the URL of an admin webpage it will not be difficult for them to guess.</p>	HIGH
<b>V5.</b>	HIGH	Low	<p>Impact – A High impact was given if an attacker is able to get access to data being sent between the react web application and the node.js server, the attack would be able to obtain the passwords of student and administrators. This will give them unauthorised access to the system, in which they will be able to do significant damage.</p>	MEDIUM

			Likelihood – A low likelihood was given because the E-voting system is operating in a low threat environment it is unlikely that student trying to influence the election would have access to the tools required to intercept data in transmission.	
--	--	--	--	--

## 6.5 Risk treatment

With all the vulnerabilities identified and ranked in terms of overall risk, it is now possible to propose ways in which to tackle the identified risks. Generally, there are four ways risk can be treated:

- Risk modification – eliminate the risk through controls.
- Risk retention – choosing to accept the consequences of the risk.
- Risk avoidance – Removing the asset entirely so that the risk does not continue to exist.
- Risk sharing – outsourcing the risk so that it become the responsibility of a third party.

### Risk Controls

The table below shows the controls that will be implemented to eliminate all the vulnerabilities.

Ref	Control	Strategy	Description
<b>C1</b>	JSON Web Tokens (cookies)	Risk modification	Json Web Tokens will be used to ensure that only though with authorised can access data from endpoints.
<b>C2</b>	Protected Routes in React	Risk modification	Protect Routes is a library in react that can be used to restrict users without authorisation from certain pages. This can be to prevent student users from access pages restricted to administrators.
<b>C3</b>	Hashing of Password	Risk modification	Passwords will be hashed using library BCrypt when new users are registered.
<b>C4</b>	Third party hosting and maintenance	Risk sharing	The React Web Application will be deployed using Netlify and the server will be deployed using Heroku. Heroku has its own



			contingency plan to handle network server downtime as well as DOS protection.
<b>C5</b>	Using HTTPS	Risk modification	The domain for both the client and node.js server use HTTPS to transfer data. This makes it more difficult for attackers to intercepted data.
<b>C6</b>	Using CORS in node.js server	Risk modification	CORS which stands for Cross-Origin Resource Sharing is a JavaScript library that can be used to set a specific domain that the server will accept requests from. This will prevent requests to the server that do not come from the domain of the react application.

The controls outlined in the table above, provide a sufficient level of security to the system and ensures CIA is not breached. In **Chapter 8: System Testing** these security controls used to test against the produced E-Voting system to ensure the system has a good level of security.

## 6.6 Security Design

This section will discuss how authentication, authorisation and access tokens will be implemented to ensure the system has a good level of security.

### 6.6.1 Authentication

Before being able to use the E-Voting system, the user must log into the system with a created account. As explained in **Chapter 3: Problem Analysis** the IT department have not granted this project access to Surrey365, which is used to authenticate students at the University of Surrey. Furthermore, the USSU have agreed to lobby the IT department, once a functioning system is up and running. Therefore, election officers will be required to create accounts on the system, this is to prevent individuals from outside the university creating accounts and voting in the elections. Once the IT department grants the E-Voting system access to Surrey365, it will be used to authenticate users.

To create an account successfully the election officer will need to enter:

- The User's first name
- The User's last name
- The type of User (Student or Admin)
- The user's Email, which must be a valid email and cannot already exist on the system.
- The user's Password. Passwords are required to be at least 10 characters long with at least one uppercase key and one symbol. A 10-character password provides a good level of security as it takes over 500 days to crack. This ensures that passwords are not susceptible to brute force attacks by an adversary. In addition to this, passwords will be salted and hashed before being stored onto the blockchain making it more difficult for an adversary to discover a user's password. To prevent the election officer from knowing the password of all created accounts, users of the application will be able to change their password once logged in by entering their existing password followed by their new password.
- The user's Privileges from 1 to 3 (see **6.6.2 Authorisation**).

## 6.6.2 Authorisation

All users will need to login with their correct email and password to use the system. If the user is a student, then they will be restricted from viewing pages meant only for admins. Likewise, admin users will not be able to access the voting page (as they are not students and should not be allowed to vote in elections).

While administrators will have access to all admin pages, the E-Voting system will have a hierarchy of access privileges. Without a certain privilege level administrators will be unable to perform some actions. The following details what an administrator can do at each privilege level (the actions outlined below corresponds to the Synopsis in the requirement specification):

- **Level 1:** Users with this privilege level will be able to Create AGM contests, Edit AGM contests, Start AGM Election Contests, End AGM Contests and Make AGM Results Visible. This is the lowest level an administrator can have and will be expected to be given to PTOs.
- **Level 2:** In addition to everything mentioned in level 1 users with this privilege level will be able to Create AGM Elections, Edit Created AGM Elections. This level of privilege will be expected to be given to USSU staff members.
- **Level 3:** In addition to everything mentioned in level 1 and level users with this privilege level will be able to Create Student Account, Create Admin Account, Delete AGM Elections, Change Admin

Privileges and Enable Student verification. This is the highest level an administrator can have and will be expected to be given to the election officer.

If the user attempts to perform an action and do not have the correct privilege the server will unauthorised user error. This can be done in an Node.js server by verifying requests before executing an endpoint.

In addition to this, all student account will be restricted from access administrative pages. This will include all pages that are used to create and configure elections. The application will know what privilege the user has based on their access token created by the user when they logged in (see Access token section for more information). Restricting certain pages from students can be done using protected routes in react.

### 6.6.3 Web Tokens

Tokenisation is the process of transforming data into random strings of characters that cannot be decoded by an adversary. JSON Web Tokens (JWT) can be used in order to grant authorisation to users.

When a user logs into their account a new JWT will be generated on the node.js server. This token will be sent back to the client as a HTTP cookie. Whenever the user makes a REST call to the server the HTTP only cookies will be included in the request. Using HTTP only cookies over normal cookies provide an extra level of security. This is because unlike normal cookies, HTTP cookies cannot be accessed by the client-side using JavaScript. Without this, an adversary could launch a XSS attack using “document.cookie” to get a list of stored cookies. When a request arrives at the node.js server, the endpoint will check that a valid token with the correct privileges has been sent with the request. This will be done by utilising node.js middleware which allows for functions to be run before an endpoint executed.

JWT and Cookies must have a set expiration date. After the set expiration date, the JWT will fail the verification process carried out by the node.js server. This will make it more difficult for an adversary to abuse the system in the event that they gain access to a JWT. In addition, when the cookie expires the user will be automatically logged out of the E-Voting system. They will need to login again to gain access to a validate JWT. Furthermore, due to the sensitivity of the system expiration duration of 30 minutes was chosen. This means that every 30 minutes users will be required re-login to their accounts to re-authenticate themselves.

The use of HTTP only cookies means that the E-Voting system is susceptible to **cross-site request forgery attacks (CSRF)**. A CSRF attack is one in which a user is duped into performing some action in an app that they

are currently logged into. If an attacker is able to get the user to make a request to that app (often without the user knowing it), the browser will automatically send its cookies, and thus the attack will be very possible. The best way to mitigate against this attack is to set up an endpoint that sends a csrf token to the client as a cookie. The client will then need to add the csrf token to headers of requests sent to the node.js server.

## Chapter 7: Implementation

---

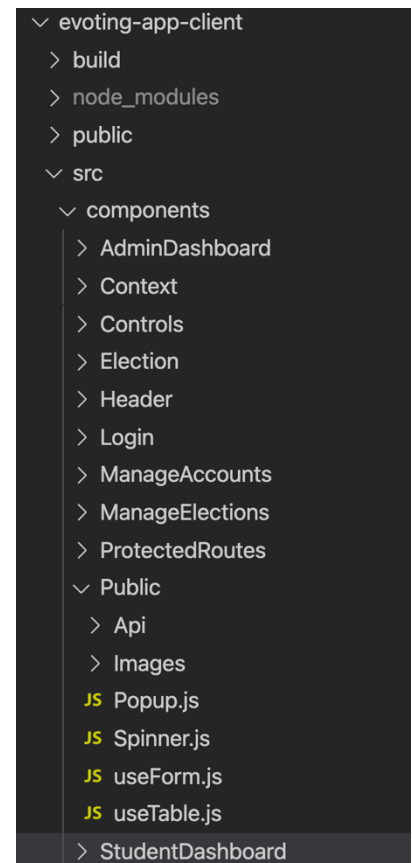
This section will give an overview of the E-Voting system produced in this project. A demonstration of the working solution can be found in the video presentation submitted with this report.

### 7.1.1 Repository Overview

The entire code for this application was submitted along with the project. Below is a short description of the contents of each directory.

**Figure 45** shows the directory if the React Web Application.

- **AdminDashboard** – contains the code for the Admin Dashboard Page
- **Context** – contains the code for the UserContext that is wrapped around most components of the react application (see high order components).
- **Election** – contains the code for the Pages relating to Voting in elections
- **Header** – contains the code for the navigation bar
- **Login** – contains the code for the login Pages
- **ManageAccount** – contains the code for the Pages relating to managing account in the system
- **ManageElection** – contains the code for the Pages relating to managing elections in the system
- **ProtectedRoutes** – contains the logic behind Protected Routes in the application (see high order components)
- **Public** – contains the components and functions that will be used by other components in the application
- **Api** – contains the functions that will call the endpoints in the node.js server
- **Images** – contains the images that will be used in the application



**Figure 45: React Web Application Repository**

Figure 46 shows the directory of the React Web Application.

- **Election** – contains the endpoints relating to voting in and managing elections.
- **HyperledgerFabric** – contains the code that connects the node.js server to the Hyperledger blockchain.
- **Login** – contains the endpoints required to login to the System.
- **Middleware** – contains code that will be run before an endpoint is executed.
- **User** – contains the endpoints relating to voting in and managing elections.
- **Wallet** – contains the public and private keys of users in the system, which will be used to connect the node.js server to the Hyperledger blockchain.

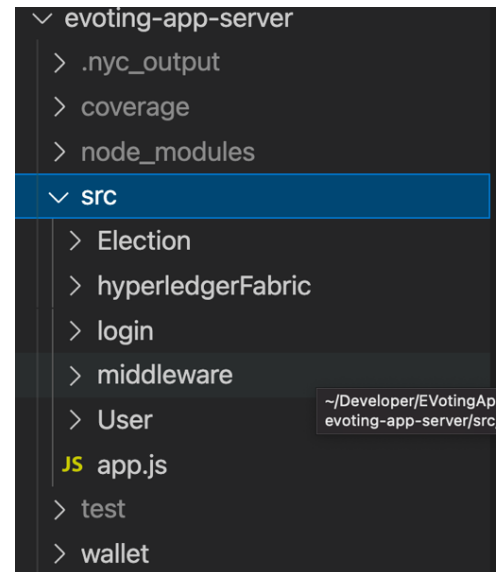


Figure 46: Node.js Repository

## 7.2 Important Aspects

This section draws attention to the main aspects of the system. These aspects include:

**Chain code queries:** Since data from the blockchain is utilised through the E-Voting application, it was important that a generic set of functions were produced to allow other functions in the chain code to easily request data from the blockchain. Figure 47 below shows the reusable code that can be used to query the two types of objects (User and Election) stored on the Blockchain. Since CouchDB stores data in JSON documents the SELECTOR function was taken advantage of.

**Vote Storage and Counting:** A BallotBox is an object that is embedded into the Election object. Within the BallotBox object there exists a list of ballots. Each ballot will contain a user's email along with who they voted for. Figure 48 is an example of a BallotBox object (stored in a JSON format). In addition to this, the BallotBox object will keep a list of students that have voted, to prevent students from voting more

```

async queryByObject(ctx, args) {
  const { isInternal } = args
  const queryData = (isInternal) ? args : JSON.parse(args)
  const { objectType } = queryData
  let queryString = {}

  if (objectType == 'User') {
    queryString = {
      selector: {
        objectType: objectType
      }
    };
  }

  else if (objectType == 'Election') {
    queryString = {
      selector: {
        objectType: objectType
      }
    }
  }

  let queryResults = await this.queryWithQueryString(ctx, JSON.stringify(queryString));

  return queryResults;
}

```

Figure 47: Chaincode Query Function

```

"ballotBox": {
  "ballots": [
    {
      "casterEmail": "TOBIDADA17@GMAIL.COM",
      "voteId": 1
    }
  ],
  "contestId": "7jss3tirrgi7yezwx89",
  "electionId": "dd56osfycjhlh29k61kb",
  "votedStudents": [
    "TOBIDADA17@GMAIL.COM"
  ]
}

```

Figure 48: BallotBox stored in JSON

than once. If a user requests to view the results of an election and are authorised to do so, the BallotBox object for the AGM Contest will be sent to the node.js server. The server would then add up the ballots given to each candidate sending the results back to the react web application to be displayed. **Figure 49** shows the code used to count the ballots in a BallotBox.

```
const getAGMResults = async (electionId, contestId, userType) => {
  const {electionContest} = await getElectionContest(electionId, contestId)
  const {ballotBox, candidates, contestName, showResults} = electionContest
  if (!showResults && userType=='Student'){
    return ({contestName: contestName, message: 'Results have not been released' })
  }
  const {ballots} = ballotBox
  const votes = candidates

  votes.forEach( (vote) => {
    vote['votes'] = 0
  })

  ballots.forEach( (ballot) => {
    votes[ballot['voteId']]['votes'] = votes[ballot['voteId']]['votes'] + 1
  })
  console.log(votes)
  return ({contestName: contestName, result: votes })
}
```

*Figure 49: Function that tallies the votes in the BallotBox*

**Protected Routes:** Protected Routes were implemented to prevent students from gaining access to admin pages by changing the URL of the web page. The react web application has three types of protected Routes, StudentRoutes which are routes meant only for student users, AdminRoute which are routes meant only for administrators, and LoginRoute routes which can only be accessed if the user is not logged in. In the event that a user tries to gain access to a restricted page they will be redirected to their dashboard (if they are logged in) or the login page if they are not logged in. **Figure 50** is an example of a component wrapped in AdminRoute.

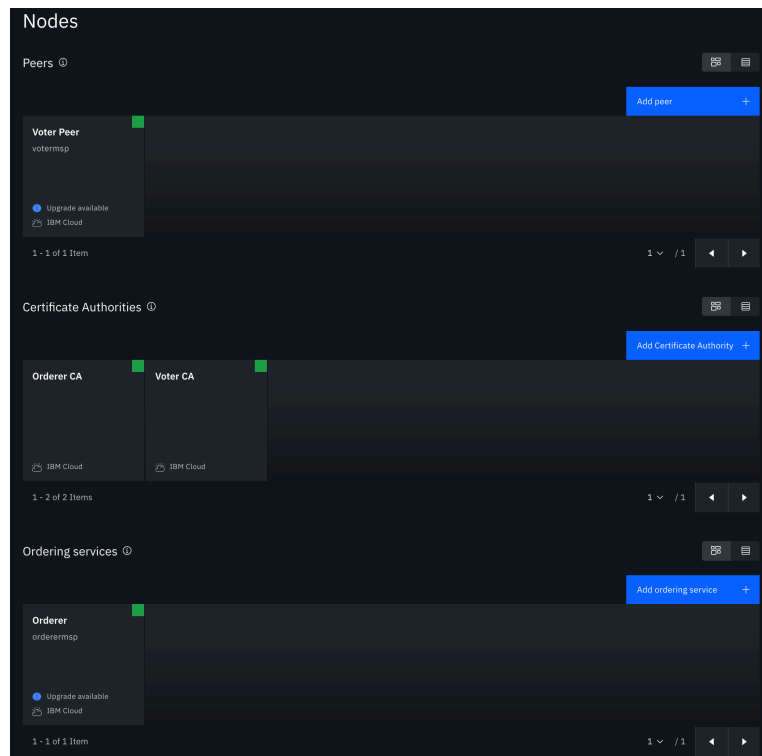
```
<ProtectedAdminRoute
  path="/manageElections"
  component={ManageElectionPage}
/>
```

*Figure 50: ManageElection Page wrapped in Protected Routes*

**Node.js Structure:** The node.js server was implemented in a way to allow for logic across the server to be shared. This was done by producing a Utilities file for every route file. Endpoints in a route file will call functions in their respective Utilities. This mimics the model-view-controller architecture in terms of providing a clear code structure. Furthermore, structuring the express server this way allowed for the code based to be better navigated, as the majority of logic for each route file was constrained in one file.

## 7.3 Deployment

The application was deployed using Netlify (for the react web application) and Heroku (for the node.js application). In addition the domain name myvote.code was obtained freely and used for the application. Netlify and Heroku both provided SSL certificates allowing the web application transmit data using HTTPS. The smart contracts were deployed using IBM Hyperledger Blockchain. **Figure 51** is an overview of the Blockchain network.



*Figure 51: Hyperledger Fabric Blockchain in IBM Cloud*

## 7.4 Technical Challenges

This section highlights the main technical challenges incurred when creating the E-Voting Application.

**Chaincode & Hyperledger Fabric:** Writing Chaincode was one of the most challenging aspects of implementing the system. Although it was written in JavaScript, the same language as my node.js server there were a limited number of examples available demonstrating how Chaincode should be written. In addition, learning the intricacies of Hyperledger Fabric such that its full capabilities could be harnessed ate into development time. This was mainly due to Hyperledger Fabric being a niche project within blockchain.

**Protected Routes:** Implementing protected routes was challenging as it required an understanding of complex react concepts. More specifically understanding how high-order components worked in react took a while to grasp. However, thanks to react being a popular framework there were many examples and tutorials out there showing how protected routes could be implemented.

**Code Quality:** When implementing a web application rarely is there a single way to achieve a desired outcome. Therefore, due to being new to react and node.js, many components and functions that work correctly could have been implemented in a way that is considered bad practise. In addition, the time constraints of this project meant that good practises such as code commenting and documentation were deprioritised over completing all of the functional requirements. This will have to be address after the project is completed, as time will no longer be an issue.



## Chapter 8: System Testing

---

During the development of the system tests were written to ensure that the system worked as intended. In addition to this a real AGM election was run using the produced application (UAT Testing). This section will cover all the aspects of the testing that was conducted. This will include: unit Testing, cross-device testing, functional testing, non-functional testing, security testing and user acceptance testing.

### 8.1 Unit Testing

Unit Tests are used to test individual methods. It is important that all the lines of code within the method are tested. This can be measured by looking at the test coverage of the application. NYC, a node package, was used in this project to measure the test coverage of the application.

There are three main approaches to unit testing:

**White Box Testing** – The goal of white box testing is to test all statements and branches. The main advantage of this approach is that the system is thoroughly tested. However, the main disadvantage is that the tests take longer to write, and require more knowledge of the innerworkings of the code base.

**Black Box Testing** – Involves testing the outputs of the methods, as opposed to the innerworkings of methods. The goal of these tests is to ensure that the correct values are outputted given the appropriate set of input values. Furthermore, tests can be written with little knowledge of how the code inside the method works. The main advantage of this approach is that black box tests are less complex to write when compared to white box testing, and they take less time to write. The main disadvantage of these tests is that it is easy to miss edge cases.

**Grey Box Testing** – Is a hybrid of the two approaches explained above. Under this approach Black box tests are written with an understanding of the interworking of the method. With a proper understanding of the method being tested the developer will be able to test that the edge case is being missed in Blackbox testing.

White box testing was used to test the system developed in this project. Due to the time constraints of this project and the lack of experience writing tests this was the most comprehensive approach. The following libraries were employed to write the tests:

- Mocha – is a feature-rich JavaScript test framework, making it easier to write asynchronous testing
- Sinon – is a standalone testing framework that allows test spies, stubs and mocks for JavaScript.
- Chai – is an assertion library for node that can be paired with any JavaScript testing framework

## 8.1.1 Node.js Sever

Unit tests were written for every function within the Node.js API in order to achieve 100% test coverage.

**Figure 52** is an example of a unit test that was written for the node.js server. The unit test was written to check that the `isPasswordCorrect` function works as intended. The `isPasswordCorrect` function returns true if a password in plain text is equal to a hashed password, and false if otherwise.

```
Run Test | Debug Test
describe('Test isPasswordCorrect', () =>{
  Run Test | Debug Test
  it("Test isPasswordCorrect Valid Password", () =>{
    const enteredPassword = 'password1234';
    const salt = bcrypt.genSaltSync(10);
    const password = bcrypt.hashSync(enteredPassword, salt);
    console.log('waste man')
    expect(loginUtilities.isPasswordCorrect(enteredPassword, password)).to.be.equal(true);
  });

  Run Test | Debug Test
  it("Test isPasswordCorrect invalid Password", () =>{
    const enteredPassword = 'password1234';
    const correctPasswordPlainText = 'password12345';
    const salt = bcrypt.genSaltSync(10);
    const password = bcrypt.hashSync(correctPasswordPlainText, salt);
    expect(loginUtilities.isPasswordCorrect(enteredPassword, password)).to.be.equal(false);
  });
});
```

*Figure 52: Unit Test Example for PasswordIsCorrect Function*

**Figure 53** shows the test running successfully, as you can see all 53 tests pass.

```
✓ Test isPasswordCorrect invalid Password (360ms)
Test getUser empty user error
✓ Test getUser no user
Test getUser connection error
✓ Test getUser network error
networkObj { error: 'some error' }
Test getUser user not found in database
✓ It should return an error
networkObj { connection: 'Successful Connection' }
Test getUser user found in database
✓ It should return an error
networkObj { connection: 'Successful Connection' }

Test JWT
  Test createToken
    ✓ Test createToken

32 passing (41s)
```

*Figure 53: Results of running all server-side unit tests*

Test coverage is split into four different categories, with each category representing a different aspect of unit testing. These categories include:

- Statements (Stmts) – Measures the percentage of statement that have been executed by the tests
- Branch – Measures the percentage of paths in a control statement that have been executed. For example, have both true and false paths been executed by the tests.
- Function (Funcs) – Measure the percentage of function that have been tested.
- Lines – Measures the percentage of lines that have been executed by the tests.

As shown in **Figure 54** the average percentage of statement coverage (Stmts) is 90% with the lowest file network.js being 82%. The main reason why 100% statement coverage could not be achieved is because certain function cannot be invoked within a testing environment. For instance, statements in the network.js file are wrapped in a try catch error, in the event that there is an error when trying to connect to the blockchain. This is not something that can be replicated in a testing environment, for two reasons. Firstly, the error/problems that can occur when connecting to the blockchain are vast and uncontrollable. Secondly, there is no way to intentionally throw an error when connecting to the blockchain ledger.

File	% Stmts	% Branch	% Funcs	% Lines
All files	90.02	63.89	95.83	90.17
src	100	75	100	100
app.js	100	75	100	100
src/Election	90.67	56.25	96.43	90.67
ElectionUtilities.js	93.41	61.11	100	93.41
electionRoutes.js	88.24	50	93.75	88.24
src/User	92.54	50	100	92.54
UserRoutes.js	91.49	50	100	91.49
UserUtilities.js	95	50	100	95
src/hyperledgerFabric	82.29	75	100	82.29
network.js	82.29	75	100	82.29
src/login	92.54	78.57	85.71	93.85
loginRoutes.js	94.87	83.33	100	97.3
loginUtilities.js	89.29	75	66.67	89.29
src/middleware	94.74	100	100	94.74
JWT.js	94.74	100	100	94.74

*Figure 54: Test Converge of Server-side code*

**Figure 54** also shows the function coverage and the line coverage which both averaged 95% and 90% respectively. The LoginUtilities had a function coverage of 66.7% an anomaly in comparison to the other file. The cause of this low score is due to a bug with the function bcrypt.compareSync which is called within the isPasswordCorrect function. The NYC package counts bcrypt.compareSync as a function but fails to track its execution. This issue has been raised on several forum such a stack overflow [43]. Therefore, the function coverage of loginUtilities is actually 100%. With that being said the function coverage and Line coverage still

fall below 100%. This is because the code within the catch branches is not being executed (for the same reasons explained above).

Lastly **Figure 54** shows the branch coverage is 63.89%. This is incredibly low, and can mostly be attributed to the code within catch branches not being executed. In addition, time constraints meant that it was difficult to write tests that covered all branches of code.

### 8.1.2 Chaincode (Smart Contracts)

Unit tests were also written for the chain code (smart contracts), to ensure that they function as intended.

**Figure 55** shows an example of a test that was written to test the `createUser` within the smart contract of the E-Voting application. The test checks that the smart contract can successfully create a user given the necessary parameters. The function is expected to return a message and the email of the newly created election.

```
Run Test | Debug Test
describe('test createUser function', () => {
  Run Test | Debug Test
  it('createUser should function correctly', async () => {
    const args = {
      email: "TOBIDADA17@GMAIL.COM",
      firstName: "Tobi",
      lastName: "Dada",
      password: "password1234",
      userType: "Admin",
      privilege: 3
    }
    const actualResult = await ctx.contract.createUser(ctx, JSON.stringify(args));
    actualResult.should.have.own.property('email')
    actualResult.should.have.own.property('message')
  });
});
```

*Figure 55: Chaincode Test example for a Creating User*

As shown in **Figure 56**, all thirty tests that were run for the smart contract pass successfully.

```
test createElection function
  ✓ createElection should function correctly
test createUser function
  ✓ createUser should function correctly (209ms)
test createUser function
  ✓ createUser should function correctly (196ms)
test createAGMContest function
  ✓ createAGMContest should function correctly
test changeElectionStatus function
  ✓ changeElectionStatus should function correctly CLOSED
  ✓ changeElectionStatus should function correctly OPEN
  ✓ changeElectionStatus should function correctly ENDED
test deleteElection function
  ✓ deleteElection should function correctly
test deleteUser function
  ✓ deleteUser should function correctly
```

*Figure 56: Test Coverage of Chaincode*

As shown in Figure 57 the average percentage of statement coverage (Stmts) is 98% with the lowest file smartContract.js being 93%. The reason why 100% coverage was not achieved for the smartContract.js file is because code within a catch statement is not being executed.







File ▲		Statements ⇅	
AGMContest.js		100%	14/14
Ballot.js		100%	7/7
BallotBox.js		100%	8/8
Election.js		100%	12/12
User.js		100%	15/15
smartContract.js		93.33%	70/75

Figure 57: Evidence of Chaincode Test coverage

Figure 58 shows that every function within the smart contract except 1 is covered by the tests written. The function that has not been covered is called updateContest. Unfortunately, it cannot be tested because in order to run successfully it must be able to query the blockchain (which cannot be done in a test environment).

Functions ⇅		Lines ⇅	
3	100%	1/1	100% 14/14
5	100%	1/1	100% 7/7
3	100%	1/1	100% 8/8
2	100%	1/1	100% 12/12
2	100%	1/1	100% 15/15
2	90%	9/10	93.33% 70/75

Figure 58: Evidence of Chaincode Test coverage (Functions and Lines)

Lastly, Figure 59 shows that branches medium level of branch coverage. The main reason for a low level of branch coverage is that certain aspects of the smart contract cannot be tested in a testing environment. In addition to this time constraints did not allow for alternative ways to test the various branches of the smart

contract to be explored. UAT testing and requirements testing will be able to cover the areas missed by the unit tests.

Branches ◆

61.54%	8/13
50%	3/6
75%	6/8
75%	9/12
58.33%	7/12
72.73%	16/22

*Figure 59: Evidence of Chaincode Test coverage (Branches)*

In summary, a high level of test coverage has been reached for both the node.js server and the chain code (smart contracts). This means that the Node.js server as well as the chain code can be trusted to function as intended. The code that could not be covered by the unit tests will be covered during requirements testing and UAT testing. Indeed, it would be sensible to explore ways to test the uncovered branches at a later date.

## 8.2 Cross-Device Testing

The purpose of Cross-Device testing is to check that the application looks and runs properly on all intended platform.

As stated in **Chapter 5: Design** the application needs to be mobile responsive as users are expected to use the application via their smartphones.

The following figures show the main pages of the application on both androids, iPhone and desktop browsers. As shown in **Figures 60** the application looks consistent across all platforms. In addition, users can use all the features of the application regardless of the platform.



Figure 60: Admin Dashboard on iPhone(right), desktop(middle) and Android(left)

## 8.3 Requirements Testing

The purpose of requirements testing is to test that the application produced meets the criteria set out in the requirements specification **Chapter 4: Requirements**. This includes both functional and non-functional requirements.

The tables below show the results of testing the application against the functional and non-functional requirements. The application was able to meet all of its functional and non-functional requirements. A working solution of the application was demonstrated to the supervisor of this project. He can testify to the accuracy of the testing matrix below. In addition to this a working solution of the application was demonstrated in the video presentation submitted with the project.

### 8.3.1 Functional Requirements Test Matrix

ID	Synopsis	Description	Status		
			Android	IOS	Desktop
FR1	Log in to account	All users of the application must be able to log into their account using valid credentials.	PASS	PASS	PASS
FR2	Log out of account	All users of the application must be able to log out of their account	PASS	PASS	PASS

<b>FR3</b>	Create Student Account	Election Officer(s) must be able to create new Student accounts, with a first name, last name, email, password and privilege.	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR4</b>	Create Admin Account	Election Officer(s) must be able to create new Admin accounts, with a first name, last name, email, password and privilege.	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR5</b>	Change Password	All users must be able to change their own password.	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR6</b>	Create AGM Election	Election Officer(s) and USSU staff must be able to create elections with an election name, description, start and end date.	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR7</b>	Edit AGM Created Elections	Election Officer(s) and USSU staff must be able to edit the attributes (name, description and date) of a created AGM election.	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR8</b>	Delete AGM Elections	Election Officer must be able to delete any election on the system.	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR9</b>	View all Elections	Election Officer(s) must be able to view all created elections on the system	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR10</b>	Create AGM contest	Election Officers, USSU staff and PTOs must be able to create AGM contests. Each contest must have a contest name, description(optional), list of candidates.	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR11</b>	Edit AGM contest	Election Officers, USSU staff and PTOs must be able to edit the attributes (name, description and list of candidates) of a created AGM contest.	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR12</b>	Start AGM Election Contest	Election Officer(s), USSU staff and PTOs must be able to start the contest, making it available for students to vote in.	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR13</b>	Generate QR code	The system must be able to generate a unique QR code for each AGM contest	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR14</b>	Generate Voting link	The system must be able to generate a unique voting link for each AGM contest	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR15</b>	Access AGM via QR code	Students must be able to access the election by using their device to scan the QR code of the election or by using the voting link	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>



<b>FR16</b>	Vote in AGM Election contest	Students must be able to cast a vote for a candidate in the AGM election.	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR17</b>	Record Votes to Blockchain	Votes casted in the election by students must be stored in the system's Hyperledger blockchain	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR18</b>	Tally all votes correctly	The system must be able to count all votes for each candidate correctly.	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR19</b>	Change Admin Privileges	Election Officer(s) must be able to add and remove administrative privileges from USSU staff and PTOs.	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR20</b>	End AGM Contest	Election Officer(s), USSU staff and PTOs must be able to end AGM contests, stopping new students from voting.	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR21</b>	Make AGM Result Visible	Election Officer(s), USSU staff and PTOs must be able to make the results of an AGM contests, visible to students	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR22</b>	View AGM Results	All users must be able to view the result of an AGM contest.	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>
<b>FR23</b>	Verify Vote	Students must be able to verify that their vote was counted correctly by the system	<b>PASS</b>	<b>PASS</b>	<b>PASS</b>

### 8.3.2 Non-Functional Requirements Test Matrix

ID	Type	Description	Status
NFR1	Performance	Request from users to the server must be carried out in a reasonable amount of time. Additionally, the application should use loading icons when pages are being loaded.	Pass
NFR2	Accuracy	User-input must be validated by the system to ensure the data submitted is in the correct format, for example: name fields can only accept an input consisting of letters.	Pass
NFR3	Software quality	The UI must be intuitive and user friendly. The UI has a constant theme, style and colours. For example, the buttons and fonts are the same across the application.	Pass
NFR4	Software quality	The UI must response to invalid inputs into the system in a clear way.	Pass (see validation testing)
NFR5	Security	The system must comply with the data protection act (2018) and all the security controls outlined in <b>Chapter 6: Security Analysis</b> must be implemented.	Pass (see security testing)
NFR6	Security	Only the specified User Class (with the correct privileges) should be able to perform the functional requirements assigned ( <b>Chapter 6: Security Analysis</b> ).	Pass (see security testing)
NFR7	Reliability	All users should be able to use the system at any hour in the day, 7 days a week.	Pass
NFR8	Semi-privacy	All student votes and personal information should be kept confidential(full-privacy). If the Election officer decides enable verification, students should be able to verify how they voted(semi-privacy).	Pass
NFR9	Communication	All three components of the system must be able to communicate with each other. More specifically, the React web application must be able to send requests and receive responses from the node.js server, and the node.js server must be able to send requests and receive responses from the Hyperledger Blockchain.	Pass

### 8.3.3 Validation Testing

The purpose of Validation Testing is to ensure that the UI only acceptance valid inputs from the user.

The application has a total of three forms, one for creating new users, one for creating elections and one for creating AGM contests. **Figure 61**, is an example of what happens if the user attempts to create an election with invalid inputs. Screenshots showing this for the Create AGM contest page and create User page can be found in Appendixes under validation test screenshots.

*Figure 61: Example of invalid fields for creating an Election*

The table below shows the outcome of each validation test (Appendixes for evidence).

Field Name	Validity conditions	Status?
<b>Create Election Form</b>		
<b>Election Name</b>	Letter only with spaces allowed	Pass
<b>Election Description</b>	Letter only with spaces allowed Field can be left blank	Pass
<b>AGM Election</b>	AGM Election is the only option	Pass
<b>Start Date</b>	Must be in format DD-MM-YYYY and cannot exceed the end date	Pass
<b>End Date</b>	Must be in format DD-MM-YYYY and cannot precede the end date	Pass
<b>Create User Form</b>		
<b>First Name</b>	Letter only no spaces	Pass
<b>Last Name</b>	Letter only no spaces	Pass

Email	Valid email address cannot already exist on the system	Pass
Password	Must be at least 10 characters long	Pass
User Type	Options are limited to Admin or Student	Pass
Privilege	Options are limited to 1,2,3	Pass
<b>Create AGM Contest</b>		
Contest Name	Letter only with spaces allowed	Pass
Contest Description	Letter only with spaces allowed	Pass
Candidate Name	Letter only with spaces allowed	Pass
Candidate Email	Valid email address	Pass

### 8.3.4 Security Testing

The purpose of security testing is to ensure that the application has met all the criteria set out in **Chapter 6: Security Analysis**. Indeed, this section aims to check that the risk controls specified in the risk assessment have been implemented and work as expected.

**Figure 62** below is a code snippet showing the verifyToken function. This function is call before every endpoint in the node.js server. If the privilege required to access the endpoint is higher than the privilege stored in the access token, a 400-status code will be sent back to the client-side. This ensures that only users with the correct privileges can access certain aspects of the system.

```

const verifyToken = (privilege) => {
  return (req, res, next) => {
    const accessToken = req.cookies["access-token"]
    if(!accessToken) {
      return res.status(400).send({error: "User is not Authenticated"})
    }
    try {
      verify(accessToken, config.security.jwt.secretkey, (error, decodedToken) =>{
        if(error) {
          throw new Error(error)
        }
        else {
          if( parseInt(decodedToken.privilege) >= parseInt(privilege) ){
            req.userEmail = decodedToken.userEmail;
            next();
          }
          else{
            return res.status(400).send({error: "User is not Authenticated"})
          }
        }
      })
    }
    catch(error){
      res.send(error)
    }
  }
}

```

Figure 62: VerifyToken function in Node.js Server

The table below shows the security controls that were implemented. Evidence of these security controls can be found within the code submitted with this report. Furthermore, these security controls were demonstrated to the supervisor of this project.

Ref	Control	Description	Implemented?
C1	JSON Web Tokens (cookies)	Json Web Tokens will be used to ensure that only those with authorised access can access data from endpoints.	YES
C2	Protected Routes in React	Protect Routes is a library in react that can be used to restrict users without authorisation from certain pages. This can be used to prevent student users from accessing pages restricted to administrators.	YES
C3	Hashing of Password	Passwords will be hashed using library BCrypt when new users are registered.	YES
C4	Third party hosting and maintenance	The React Web Application will be deployed using Netlify and the server will be deployed using Heroku. Heroku has its own contingency plan to handle network server downtime as well as DOS protection.	YES
C5	Using HTTPS	The domain for both the client and node.js server use HTTPS to transfer data. This makes it more difficult for attackers to intercept data.	YES
C6	Using CORS in node.js server	CORS which stands for Cross-Origin Resource Sharing is a JavaScript library that can be used to set a specific domain that the server will accept requests from. This will prevent requests to the server that do not come from the domain of the react application.	YES

## 8.4 User Acceptance Testing

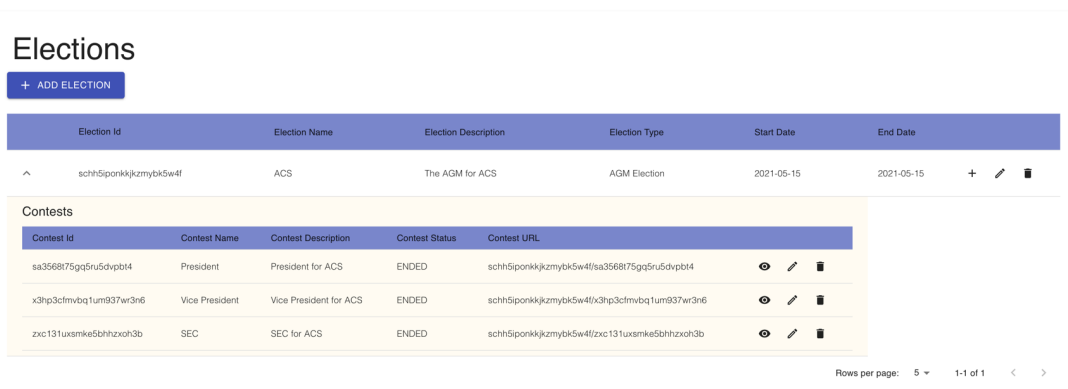
User acceptance testing is an important pillar of developing any piece of software. The main purpose of UAT is to check that an application as a whole is in an acceptable condition for users. To do this the application was used to run a real AGM election at the University of Surrey, where all actors including the admin was a third party. Below is a description of how the election went and the feedback from users.

### 8.4.1 AGM Election Summary

The society that agreed to use the system for their AGM was called ACS. This is a small cultural society at the university and was an ideal society to test the produced application on. The election was host on a zoom call and had five main stages:

**Configuration of Accounts:** Accounts had to be made for all attendees of the AGM. A total of 6 people were expected to come to the AGM, so a total of six accounts was created by the administrator. Users were then sent their credentials via email before the AGM started.

**Configuration of Elections:** An Election had to be created on the system for the AGM. As there were three positions up for grabs a total of three AGM contests had to be created. All three contests had two valid candidates. **Figure 63** shows all three contests that were created for the ACS election.



The screenshot shows a web interface titled 'Elections'. At the top left, there is a '+ ADD ELECTION' button. Below it is a table with the following columns: Election Id, Election Name, Election Description, Election Type, Start Date, and End Date. The table contains one row for an election named 'ACS' with description 'The AGM for ACS' and type 'AGM Election', starting and ending on 2021-05-15. Below the election table is a 'Contests' section with a table containing three rows. Each row has columns for Contest Id, Contest Name, Contest Description, Contest Status, and Contest URL. The contests are for 'President', 'Vice President', and 'SEC' for ACS, all with a status of 'ENDED'. Each contest row also includes a set of icons for viewing, editing, and deleting.

Election Id	Election Name	Election Description	Election Type	Start Date	End Date
schh5iponkkjkzmybk5w4f	ACS	The AGM for ACS	AGM Election	2021-05-15	2021-05-15

Contest Id	Contest Name	Contest Description	Contest Status	Contest URL
sa3568f75gq5ru5dvpbt4	President	President for ACS	ENDED	schh5iponkkjkzmybk5w4f/aa3568f75gq5ru5dvpbt4
x3hp3cfmvbq1um937wr3n6	Vice President	Vice President for ACS	ENDED	schh5iponkkjkzmybk5w4f/x3hp3cfmvbq1um937wr3n6
zxc13luxsmke5bhzhxoh3b	SEC	SEC for ACS	ENDED	schh5iponkkjkzmybk5w4f/zxc13luxsmke5bhzhxoh3b

**Figure 63: Election and Contests for ACS AGM**

**Voting Process:** After the election had been configured the users were asked to use their votes to scan the QR code and vote in the election. Users that were using their desktops were sent the links to each contest via zoom. Users were then given 10 minutes to cast the vote before the contests were close and no more votes could be cast. **Figure 64** shows the QR code for the presidential contest.

## Link for President contest

<https://myvote.codes/AGMvote/schh5iponkkjkzmybk5w4f/sa3568t75gq5ru5dvpbt4>



*Figure 64: QR code for Presidential Contest*

**Release Results:** Once the election had been ended by the admin, they released the results. Figure 65 shows the outcome of the presidential AGM contest. The election results were consistent with how students said they voted.

## Results for President contest

Khalil Russal (Khalil@gmail.com) : 6

Gbemi (Gbemi@gmail.com) : 1

RELEASE RESULTS

*Figure 65: Results of Presidential Contest*

**Verify Vote:** After the results had been released the user was able to verify that their vote had been counted correctly.

### 8.4.2 UAT Findings

The trial election was a huge success, all users were able to log into the application and vote in the three AGM contests. Furthermore, after conducting the AGM election users were asked to give feedback of their experiences. The feedback was mostly positive with a few recommendations (see suggested improvements below).

The majority of users liked the UI describing it as “professional”, “simple” and “easy to use”. In addition, users found the colours of the application blue, white and red to blend well together, giving the application

an aesthetically pleasing feel. In addition, users liked the logo of the application especially the use of a stag which is the mascot of the University of Surrey. On the voting process users found the application to be intuitive, as it only required a voting link. Users who used the application on their phone appreciated the ability to access the election via a QR barcode. When asked if they trusted the application, students felt that the ability to verify how they voted as well as the professional look of the application made the system trustworthy.

Admin users also found the ability to manage the system intuitive and easy to use as it required little technical knowledge. In addition to this the admin appreciated the fact that AGM contests were embedded into an overall election, as it makes the management of the AGM process much easier.

In addition to running the election users were asked to “play around” with the application. This was done to see if there were any flaws within the application that an adversary could exploit. Gladly users were unable to access restricted pages or perform tasks that they were not authorised to do.

One observation of the UAT conducted is that it did not test how the application would perform with a significant number of users. Furthermore, the application performed well with only 7 users which is typical of an AGM election. However, if multiple AGMs are being conducted at the same time it is unclear if there will be problems with the applications performance. To ascertain whether performance is an issue with large volumes of users, load balance tests will have to be conducted.

### 8.4.3 UAT Suggested Improvements

After the completion of the election users were asked to give constructive criticism of the application and any features they would like to see in the future. These features included:

**Clearer results:** Currently the results show the total number of votes received by each candidate in the election. Users thought that it would be better for the results to also display the winner of the election as opposed to just the tally of the votes. In addition to this, users thought it would also be beneficial if the total voter turnout was displayed on the results screen as well.

**Option for Preferential voting:** Currently the system only supports first past the post voting, in which a candidate can only vote for one candidate. Users expressed the desire to be able to vote using a preferential voting system such as AV and STV (in which voters rank candidates).

**Help Page:** Although users found the application intuitive, users thought that it would be beneficial to have a help page. This page would graphically demonstrate how users could use the application to vote.

**Confirmation Messages:** When performing important and unchangeable action such as voting and deleting elections, the application is unforgiving. This is because there is no pop-up message that allows users to confirm if they want to perform the action. This could potentially lead to students accidentally voting for a



particular candidate and admins accidentally deleting elections. Therefore, it was suggested that a pop-up message asking the user to confirm irreversible actions.

**Vote Success Confirmation:** Currently, after a student votes in an AGM election they are taken back to the student dashboard page. Users found this confusing as there was nothing confirming that their vote had been successfully counted. Therefore, it was suggested that a success message be displayed to students after their vote had been successfully counted.

**More Spinners:** Currently when a user clicks a button and data is being fetched there is no indication that the application is fetching the data. This meant that at times many users were unsure if their click was picked up by the system. It was therefore suggested that spinners be added underneath buttons while data is being fetched from the server.

## Chapter 9: Evaluation & Conclusion

---

### 9.1 Introduction

This chapter will evaluate the work produced in this project against the aims and objectives outlined in **Chapter 1: Introduction**. In addition, while the previous section focused on testing the application this section will focus on evaluating the overall effectiveness of the E-Voting system produced. The overall effectiveness of the application will be measured against the criteria of what makes a good E-voting system outline in **Chapter 2: Research**. This section will also discuss the future development of the application, and how the principles learnt while undertaking this project can be applied in the real world.

### 9.2 Evaluation of project against objectives

As seen in **Chapter 8: System Testing** the E-Voting Application produced in this project meets all functional and non-functional requirements outlined in the **Chapter 4: Requirements** section. Additionally, the same can be said about the security controls outlined in **Chapter 5: Security Analysis** which have all been implemented. However, these requirements do not cover the project's objectives. Therefore, to evaluate the project as a whole we must consider the objectives stated in **Chapter 1: Introduction**. The table below shows whether each objective has been met.

Objectives	Description	Achieved? (Yes/No)
Objective 1	Explore and understand the attributes and vulnerabilities of an effective E-Voting system.	Yes
Objective 2	Review technologies and frameworks that can be used to implement an E-Voting system that functions on IOS and Android devices.	Yes
Objective 3	Gather Requirements through the use of stakeholder interviews, surveys and focus groups.	Yes
Objective 4	Analyse and evaluate the current E-Voting system used by the Student Union to conduct elections.	Yes
Objective 5	Conduct a risk assessment to identify threats to the proposed E-Voting system and propose controls to mitigate them.	Yes
Objective 6	Iteratively design, implement and deploy a new E-Voting system that can be used on IOS and Android.	Yes
Objective 7	Perform unit, functional, non-functional, cross device and security testing to ensure that the system functions as expected.	Yes
Objective 8	Conduct UAT testing, by trialling the application in one or more AGM elections to demonstrate the utility of the new system.	Yes
Objective 9	Gather feedback and recommendations from users, that will be implemented in future iterations of the application.	Yes
Objective 10	Evaluate the effectiveness of the E-Voting system produced and propose improvements for future iterations.	Yes

Evidence of objective 1 and 2 can be found in **Chapter 2: Research**. The first half of the chapter, the literature review, examined the fundamentals that make a good E-Voting system, as well as the approaches that will be taken to develop the new E-Voting system. After evaluating the security vulnerabilities of E-Voting systems, it was concluded that a security analysis will be conducted before development begins. In addition, the attributes that determine a good E-Voting system were identified and ranked based on their importance. It was also concluded that the attributes ranked highest will be prioritised during the implementation of the application. It was also decided that upon the completion of this project the System produced will be evaluated against the identified attributes. Lastly after considering a variety of options, it was decided that the new E-Voting System should be implemented using permissioned Blockchain technology. The second half of the chapter provided a justification for the technologies that were used to develop the application. After looking at a multitude of web technologies, react was used to implement the

UI of the application and node.js was used to implement the server-side. Upon reflection although a comprehensive literature review was produced, aspects of the literature review were excessive and unnecessary to the development of the application.

Evidence of Objective 3 and 4 can be found in **Chapter 3: Problem Analysis**. The chapter provides an analysis of the meeting held with the Students Union to identify issues they were having with their current voting system and to ask what features they would like to see in a new E-Voting system. Additionally, the results of the survey that was carried out on student is presented and analysis to ascertain the kind of features would like the E-Voting system to have. The analysis obtained from meeting and survey were used to inform the requirements in **Chapter 4: Requirements**. In addition, the latter half of the chapter breaks down and explains the USSU current E-Voting system for both AGM election and SurreyDecides elections. The systems were then analysed to highlight important aspect that of the system that needed to be implemented in the new E-Voting system. Additionally, the analysis of the system also allows for security flaws and areas that need to be improved to be identified. In reflection, the scope of the E-Voting application should have been mapped out prior to the Problem analysis being conducted. This is because, it was later decided in **Chapter 4: Requirements** that system was going to replace the USSU's AGM election system and not their SurreyDecides E-Voting system. This would have saved time that could have been spent on development as the SurreyDecides system would not have needed to be analysed.

**Chapter 6: Security Analysis** acts as the evidence for Objective 5. In this chapter a full risk assessment was conducted to assess how secure the E-Voting system designed in **Chapter 5** was. The assessment began by seeking to identify all risks to the system (no matter how small). After this was done, it was then necessary to calculate the likelihood and impact of each identified risk. The last part of the risk assessment proposed solutions to mitigate the identified risks. Furthermore, section **6.5 Security Treatment** then explained how the security controls proposed could be built into the system. Conducting the risk assessment proved to be worthwhile as it is much easier to build an application with security controls, than to implement security controls after the application has been built. However, in hindsight it would have been better to have conducted a security analysis prior to designing the application, as it could have saved time.

**Chapter 5: Design, Chapter 7: Implementation** as well as the finished version of the application stands as evidence for Objective 6. In Chapter 6, a variety of diagrams were produced to represent how the application will function. On balance the chapter did a great job at explaining the systems overall architecture and how different components would interact in order to perform tasks for users. Having said this, this chapter could have referenced common design patterns, as it would have led to a more robust system being produced.

**Chapter 7: Implementation** highlights the important functions within the application and discusses the technical challenges during the development process. The chapter also details how the application was deployed such that it can be accessed by users to carry out an AGM Election. Requirements were split up to different sprints allowing for an iterative approach to be taken to develop the application. Having said this, it was difficult to stick to the principles of agile development as many of the principles work best in small teams of developers.

Evidence of objectives 7, 8 and 9 can be found in **Chapter 8: System Testing**. In this chapter rigorous testing was conducted on the final version of the application. The application including the smart contracts had a test coverage of over 90%. In addition, all functional and non-functional requirements were successfully implemented. Further, as part of UAT the application was used to conduct an AGM election at the university of surrey. The application was able to run the AGM election from end to end with no major issues. Afterwards feedback, which was mostly positive, was gathered and will be used to plan the next phase of development for the E-Voting system. Although the testing of the system was robust and of high quality, further testing should include load balancing testing. This will ensure that the E-Voting system performs well when there is a large volume of users.

Objective 10 which evaluates the effectiveness of the E-Voting system and discusses further work can be found in section **9.3 Evaluation of E-Voting System** and **9.4 Future Work**.

## 9.3 Evaluation of E-Voting System

According to academic literature there are a comment set of attributes that make an E-Voting system effective. These attributes were discussed in section **2.3 What makes a good E-Voting system?** and will be used to evaluate the effectiveness of the E-Voting system produced. The table below show whether an attribute has been met and provides a justification.

Attribute	Description	Achieved? (Yes/No)
Accuracy	Votes cannot be altered during transmission or during the counting of votes, and it is impossible for ineligible voters to vote and for invalid votes to be counted.	Yes
Verifiability	Voters should be able to independently check that their vote was recorded correctly.	Yes

<b>Democracy</b>	Combination of accuracy and verifiability. E-Voting system permits only eligible voters to vote, only once.	Yes
<b>Reliability</b>	An E-Voting system is considered reliable if it continuously performs its function as expected.	Yes
<b>Semi-Privacy</b>	An E-Voting system upholds privacy if a third party cannot determine how an individual voted in the election.	Yes
<b>Mobility</b>	An E-Voting system is considered mobile if there is no restriction on the location from which an eligible voter can vote.	Yes
<b>Convenience</b>	An E-Voting system is considered to be convenient if it allows users to quickly and easily vote, with minimal technical skill.	Yes
<b>Flexibility</b>	The administrator should be able to change election rules such as eligible voters, a student's ability to change their vote, the start and end date of elections, and the electoral System being used, i.e., switching between FPTP and STV.	Partially
<b>Social Acceptance</b>	An E-Voting system is considered to have social acceptance if it is perceived as being an effective system by the majority of students.	Yes

Below is an explanation of how each attribute is met.

**Accuracy:** The system prevents users from being able to vote by keeping a record of students that have already voted. Additionally, all students at the university of surrey are eligible to vote in AGM elections. And because only students will have access to the system, only eligible voters will be able to vote. Lastly the system prevents votes from being altered during transmission by using HTTPS when data is sent between the React Web Application, Node.js server and the Blockchain.

**Verifiability:** As shown in *Chapter 8 System Testing*, Students are able to verify their vote once the election results have been released by the admin.

**Democracy:** Both accuracy and verifiability are met, and the system only allows users to vote for a single candidate, once.

**Reliability:** As demonstrated in UAT the application has been deployed and can be used to run AGM elections from start to finish with no major issues.

**Semi-Privacy:** How a student voted is securely stored on the blockchain and a third party cannot find out how a user has voted (not even the admin). In addition, students will only be able to find out how they voted after the election results have been released.

**Mobility:** As demonstrated in cross device testing the application can be used by anyone with a modern web browser and an internet connection.

**Convenience:** After the UAT was conducted students described the application as being simple and easy to use. Furthermore, the application requires no technical skills to be able to use the system to vote.

**Flexibility:** While admins are able to delete students from the system (revoking their voting eligibility) they are unable to change the electoral system used to conduct the election. It is for this reason that this attribute has been partially met. Having said this, this feature will be implemented in a future iteration of the application.

**Social Acceptance:** Feedback taken from users after the UAT was conducted shows that students trust the application and will therefore trust the election results it produces.

## 9.4 Future Work

After demoing my application to USSU staff many of them were impressed by what the application could do, in such a relatively short amount of time. They expressed an interest in the continual development of the application, and asked if they could receive regular updates on the progress of the application going forward. While this project was a huge success the E-Voting system in its current form allows AGM election to be conducted at a basic level. This section will build on from section **8.4.3: UAT suggested recommendations** and discuss the additional functionalities that will be implemented in future versions of the application. These features will include:

**Feedback from UAT:** First and foremost, the recommendations gathered from the UAT Tests should be implemented as a top priority. As these suggestions have come directly from users, implementing these features will improve the overall user experience of the E-Voting Application.

**SurreyDecides Elections:** *Chapter 3: Problem analysis* included an analysis of the issues that the USSU were experiencing with their existing E-Voting system, Mi-Voice. However, due to the time constraints of this project it was not possible to produce a new system that could be used for both AGM and SurreyDecides elections. Therefore, it would be appropriate to extend the E-Voting system to allow for SurreyDecides Elections to be run using the system. This would mean that Election officers would gain the ability to create, edit and delete SurreyDecides elections. In addition, students would gain the ability to nominate themselves as a candidate in SurreyDecides Elections, as well as upload their own candidate information. This candidate information will include their manifesto commitments and a profile picture of the candidates. Giving the E-Voting system the ability to conduct SurreyDecides elections as well as AGM elections will mean that the two most important elections at the university can be run using the same application. This will give election at the University of Surrey an ecosystem student and admins will be familiar with.

**Society Self Registration:** Another feature that would make running elections at the University of Surrey easier is the ability to allow society leaders to self-register their society for an AGM. Admins would then have the option to approve or discard the AGM election. This would save the staff at the USSU time, allowing them to focus on other tasks.

**Load Balance Testing:** As stated in *Chapter 8: System Testing*, load balance testing was not carried out during the development of the application. This will need to be done in order to ascertain if the system can support a large volume of users.

**Surrey365:** If this application is to be adopted by the University of Surrey, it will need to be integrated with the Universities single sign in system. Unfortunately, this could not be done in this project as the IT department refused to grant access to it. However as stated previous the USSU has agreed to lobby the IT department. Therefore, future versions of the application should allow users to login to their account using the same credentials they use for outlook and SurreyLearn.

## 9.5 Conclusion

In summary, completing this project was heart-warming; being able to produce a system that leveraged technologies to improve the election process at the University of Surrey was truly fulfilling and rewarding. When measuring the project against its aim, which was to produce a new secure e-voting system that can be used to conduct AGM elections at the University of Surrey, the project has been a huge success. As shown in this chapter the projects have been able to meet all of its objectives, and can be considered an effective E-Voting system in line with attributes gathered from academic literature. Additionally, important skills such as researching and understanding academic literature, full stack development using react and node.js, blockchain development, project management and system testing have been gained from undertaking this project. The hope is that these skills can be used to solve real world problems and continue the development of the E-Voting system.

## References

---

- [1] Office for Students, "National Student Survey - NSS," Office for Students, 24 09 2020. [Online]. Available: <https://www.officeforstudents.org.uk/advice-and-guidance/student-information-and-data/national-student-survey-nss/get-the-nss-data/#datafiles>. [Accessed 08 10 2020].
- [2] Surrey Students' Union, "Surrey Decides 2020 Elections Summary," Surrey Students' Union, 07 03 2020. [Online]. Available: [ussu.co.uk/voice/Elections/Pages/SurreyDecide.aspx](http://ussu.co.uk/voice/Elections/Pages/SurreyDecide.aspx). [Accessed 09 10 2020].
- [3] The London School of Economics Students' Union, "LSE Students' Union Election 2020 Stats," The London School of Economics Students' Union, 2020. [Online]. Available: <https://www.lsesu.com/democracy/elections/election-stats/>. [Accessed 09 10 2020].
- [4] Royal Holloway Students' Union, "Royal Holloway Election 2020 Stats," Royal Holloway Students' Union, 2020. [Online]. Available: <https://www.su.rhul.ac.uk/voice/elections/lead/>. [Accessed 09 10 2020].
- [5] Fairvote, "Voter Turnout 101," Fairvote, [Online]. Available: [https://www.fairvote.org/voter\\_turnout#voter\\_turnout\\_101](https://www.fairvote.org/voter_turnout#voter_turnout_101). [Accessed 09 10 2020].
- [6] Surrey Students' Union, "Pulse Report," Alterline, 01 07 2020. [Online]. Available: <https://www.ussu.co.uk/yourunion/Pulse%20Reports/2019/Student%20Life%20Pulse%20-%20The%20University%20of%20Surrey%20Students'%20Union%20-%20Pulse%203%2019-20.pdf>. [Accessed 09 10 2020].
- [7] A. Altvater, "what-is-sdlc," 8 04 2020. [Online]. Available: <https://stackify.com/what-is-sdlc/>. [Accessed 1 10 2020].
- [8] I. Sommerville, "Professional Software Development," in *Software Engineering, Global Edition*, Pearson Education Limited, 2016, pp. 22-24.
- [9] Lucidchart Content Team, "Pros and cons of waterfall methodology," 5 10 2017. [Online]. Available: <https://www.lucidchart.com/blog/pros-and-cons-of-waterfall-methodology>. [Accessed 1 10 2020].
- [10] I. Sommerville, "Software process models," in *Software Engineering Global Edition*, Pearson Education Limited, 2016, pp. 45-51.
- [11] D. Gritzalis, "Principles and requirements for a secure e-voting system," *Computers & Security*, vol. 21, no. 6, pp. 539-556, 2002.
- [12] T. Scott, Director, *Why Electronic Voting Is Still A Bad Idea*. [Film]. YouTube, 2019.



- [13 S. Talab and A. Al-Ameen, "The Technical Feasibility and Security of E-Voting," *The International Arab Journal of Information Technology*, vol. 10, no. 4, 2013.
- [14 OASIS, "Election Markup Language," OASIS, 2011.
- [15 The University of Surrey Students' Union, "The University of Surrey Students' Union Byelaws," 2019. [Online]. Available: <https://www.ussu.co.uk/yourunion/Governing%20Documents/Constitution%20and%20Byelaws/Byelaws.pdf>. [Accessed 01 11 2020].
- [16 E. M. and F. S., "Internet-Based Security Incidents and the Potential for False Alarms," *Electronic Networking Applications and Policy*, vol. 10, no. 3, pp. 238-245, 2000.
- [17 The Electoral Commission, "Electronic voting May 2007 electoral pilot schemes," The Electoral Commission, 2007.
- [18 S. Gupta, "Types of Malware and its Analysis," *International Journal of Scientific & Engineering Research*, vol. 4, no. 1, 2013.
- [19 Sophos, "THE STATE OF RANSOMWARE 2020," Sophos, 2020.
- [20 Washington University Computer Science, "Design and Implementation of a Security-Conscious Electronic Polling System," Washington University Computer Science, 1996.
- [21 E.-S. A, "Fast Cryptographic Privacy Preserving Association Rules Mining on Distributed Homogenous Database," *The International Arab Journal of Information Technology*, vol. 7, no. 2, pp. 152-153, 2010.
- [22 P. A and K. S, "How to Improve Security in Electronic Voting," *Ubiquity Information Everywhere*, vol. 8, no. 6, pp. 1-7, 2007.
- [23 "FIPS 197, Advanced Encryption Standard (AES)," Federal Information Processing Standards Publications, 2001.
- [24 A. Zwierko and A. Zwierko, "A Light-Weight e-Voting System with Distributed Trust," *Electronic Notes in Theoretical Computer Science*, p. 109–126, 2007.
- [25 M. Nofer, P. Gomber, O. Hinz and D. Schiereck, "Blockchain," *Business & Information Systems Engineering*, vol. 59, pp. 183-187, 2017.
- [26 K. Wüst and A. Gervais, "Do you need a Blockchain?," *2018 Crypto Valley Conference on Blockchain Technology*, pp. 45-54, 2018.
- [27 Z. Zheng, H.-N. Dai and S. Xie, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services*, vol. 14, no. 4, p. 352, 2018.

- [28] "RISKS AND OPPORTUNITIES OF BLOCKCHAIN BASED ON E-VOTING SYSTEMS," *2019 16th International Computer Conference on Wavelet Active Media Technology and Information Processing*, pp. 365-368, 2019.
- [29] S. Park, M. Specter, N. Narula and R. L. Rivest, "Going from Bad to Worse: From Internet Voting to Blockchain Voting," MIT & Harvard, 2020.
- [30] L. LAMPORT, R. SHOSTAK and M. PEASE, "The Byzantine Generals Problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382-401, 1982.
- [31] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," *Annual International Cryptology Conference*, pp. 139-147, 1992.
- [32] M. S. Ferdous, M. J. M. Chowdhury, M. A. Hoque and A. Colman, "Blockchain Consensus Algorithms: A Survey," Cornell University, 2020.
- [33] QuantumMechanic, "Proof of stake instead of proof of work," 11 07 2011. [Online]. Available: <https://bitcointalk.org/index.php?topic=27787.0>. [Accessed 11 05 2019].
- [34] Peer Coin, "Peer Coin," [Online]. Available: <https://peercoin.net>. [Accessed 24 07 2019].
- [35] S. Kidecha, "native-vs-hybrid-vs-cross-platform," 27 05 2020. [Online]. Available: <https://dzone.com/articles/native-vs-hybrid-vs-cross-platform-how-and-what-to>. [Accessed 06 11 2020].
- [36] S. Sharma, "Native-vs-hybrid-vs-cross-platform-what-to-choose," 09 08 2020. [Online]. Available: <https://medium.com/flutterdevs/native-vs-hybrid-vs-cross-platform-what-to-choose-3221130f7cc5>. [Accessed 06 11 2020].
- [37] S. Korolev, "Mobile App Development Approaches Explained," 19 06 2020. [Online]. Available: <https://railsware.com/blog/native-vs-hybrid-vs-cross-platform/>. [Accessed 06 11 2020].
- [38] U. Khan, "The Pros and Cons of Native Apps," 18 06 2018. [Online]. Available: <https://clutch.co/app-developers/resources/pros-cons-native-apps>. [Accessed 06 11 2020].
- [39] StarDust, "Hybrid-apps," [Online]. Available: <https://www2.stardust-testing.com/en/blog-en/hybrid-apps>. [Accessed 06 11 2020].
- [40] I. Vyas, "native-vs-hybrid-vs-cross-platform-best-option-for-mobile-app-in-2020," 20 06 2020. [Online]. Available: <https://citrusbug.com/article/native-vs-hybrid-vs-cross-platform-best-option-for-mobile-app-in-2020>. [Accessed 06 11 2020].

- [41 M. Jonez, "Cross-Platform Mobile Application Development- Benefits and Drawbacks," 20 01 2020.  
] [Online]. Available: <https://martha-7987.medium.com/cross-platform-mobile-application-development-benefits-and-drawbacks-b8982095beb0>. [Accessed 06 11 2020].
- [42 G. Cselle, 31 07 2018. [Online]. Available: <https://medium.com/gabor/every-step-costs-you-20-of-users-b613a804c329>. .
-

## Appendices

---

### 11.1 Appendix A: Consent form

#### E-Voting System for the USSU: Consent Form

Ayoolamide Oye-dada (Tobi Dada) is a final year student at the University of Surrey. As a final year Computer Science student Tobi Dada is required to complete a project. Tobi has decided to produce a new E-Voting system that can be used by the University of Surrey Student Union to hold elections. Tobi's final submission will need to include the functioning E-Voting System as well as a report detailing how the system was developed from end-to-end. Once submitted the report will become the property of the University of Surrey and will be marked by two University of Surrey lecturers. Furthermore, the University of Surrey will have the prerogative to share the report with others.

I \_\_\_\_\_

hereby give consent for Tobi Dada to use quotes, statements and observations made by myself in his report, on the bases that I can be directly named.

I \_\_\_\_\_

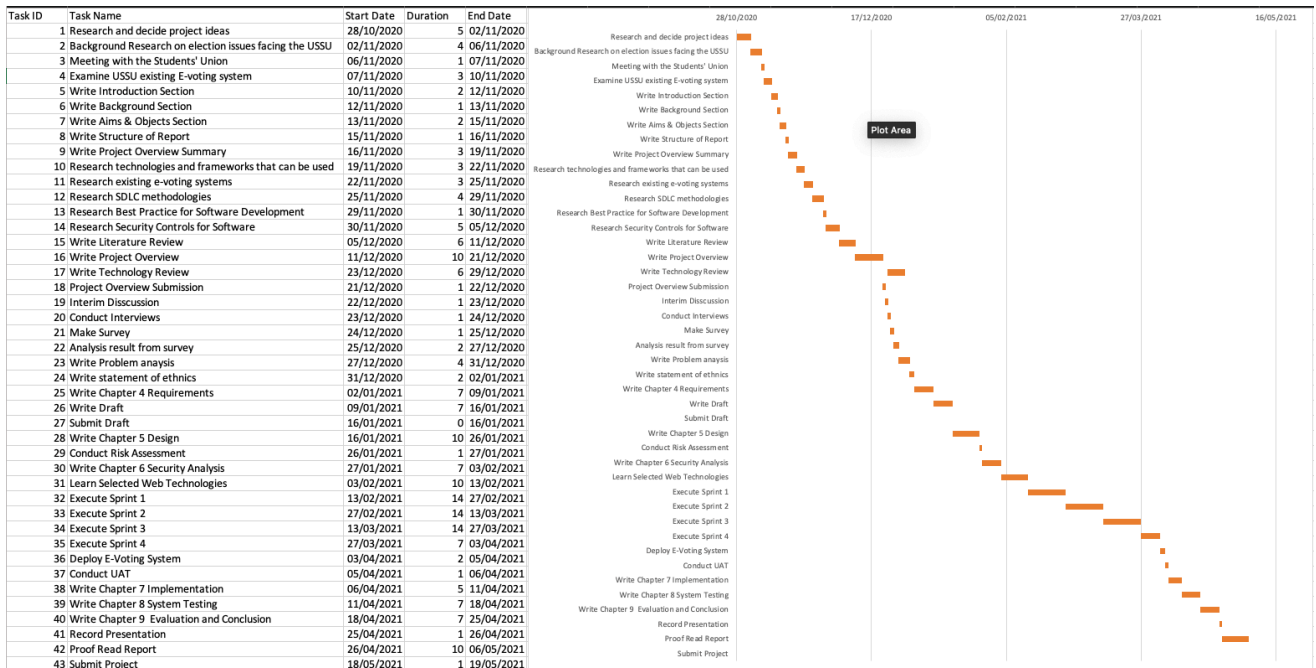
hereby give consent for my quotes, statements and observation to be shared with and used by the University of Surrey.

Sign here:

Date:

---

## 11.2 Appendix B: Gantt Chart



## 11.3 Appendix C: E-Voting System Survey

Please provide your University of Surrey email.

Open Response

**What is your ethnicity?**

White

Mixed or Multiple ethnic groups

Asian or Asian British

Black, African, Caribbean or Black British

Prefer not to say

**What is your year of study?**

First Year

Second Year

Third Year

Fourth Year

Other...

**What course do you study?**

Open Response

**What is your gender?**

Male

Female

Other

Prefer not to say

**Have you voted in a Students Union election before?**

Yes

No

**Can you rate your overall experience when using the Students' Union's Voting system?**

1 – Very Bad

- 2 – Bad
- 3 – Neutral
- 4 – Good
- 5 – Very Good

**Do you trust that the E-Voting system used by the Students' Union's properly records and counts all votes?**

- Yes
- No
- I don't know

**Do you trust the Students' Union to carry out Student elections fairly and impartially?**

- Yes
- No

**What devices would you want to vote on(select 2)?**

- Phone (iPhone and Android)
- Laptop
- Tablet

**What is the most important aspect of an E-Voting system to you?**

Security – the system cannot be hacked by a third party in order to influence the election.

Democratic – all eligible students are able to vote once for their preferred candidate(s).

Usability – a user friendly an intuitive UI

Reliability – the System will function as indented for the entirety of the election.

Privacy – A third party cannot discover how an individual voted in the election.

Social Acceptance – Understanding the technology being used by the E-Voting System

**Do you have any concerns about voting via a web application?**

- Yes
- No
- I don't know

**If you answered yes to the previous question, what are your concerns?**

Open response

**Are there any features or ideas that you would like to see added to the Students' Union's existing E-Voting System?**

Open response

## 11.4 Appendix D: E-Voting System Survey

### SAGE-HDR

---

Response ID	Completion date
640816-640807-73194089	13 May 2021, 01:53 (BST)

<b>1</b>	<b>Applicant Name</b>	Ayoolamide Oye-dada
<b>1.a</b>	<b>University of Surrey email address</b>	ao00577@surrey.ac.uk
<b>1.b</b>	<b>Level of research</b>	Undergraduate
<b>1.b.i</b>	<b>Please enter your University of Surrey supervisor's name. If you have more than one supervisor, enter the details of the individual who will check this submission.</b>	Steve Schneider
<b>1.b.ii</b>	<b>Please enter your supervisor's University of Surrey email address. If you have more than one supervisor, enter the details of the supervisor who will check this submission.</b>	s.schneider@surrey.ac.uk
<b>1.c</b>	<b>School or Department</b>	Computer Science
<b>1.d</b>	<b>Faculty</b>	FEPS - Faculty of Engineering and Physical Sciences Sciences

<b>2</b>	<b>Project title</b>	A Secure E-voting Web Application for AGM Elections
----------	----------------------	---

<b>3</b>	<p><b>Please enter a brief summary of your project and its methodology in 250 words. Please include information such as your research method/s, sample, where your research will be conducted and an overview of the aims and objectives of your research.</b></p>	<p>Data from the most recent NSS and Pulse survey show that trust and satisfaction in the University of Surrey Students' Union is at an all-time low. Many students do not believe that the USSU represents their academic interests. This is a direct result of the low turnout seen in recent USSU elections. In addition to this the existing E-Voting system cannot be used in small scale AGM elections. This combined with the Covid-19 pandemic meant that AGMs, which are normally carried out in person, were conducted virtually using Surveys in Microsoft Teams (an insecure system). To solve these problems this project details the, design, implementation and testing of a new E-Voting System (a web application) which will replace the existing E-Voting solution used by the USSU. The hope is that the new system will boost student trust and participation in elections, making the USSU more representative.</p> <p>Additionally, the rise in distributed ledger technology and its properties make it ideal for implementing a secure E-Voting system. More specifically, blockchain, a technology traditionally associated with cryptocurrencies, can be used to implement a secure E-Voting system. Therefore, the E-Voting solution implemented in this project leverages private blockchain technology, which differs from the traditional approaches to E-Voting systems which rely on centralised databases and public blockchains.</p>
----------	--	---



4	<b>Are you making an amendment to a project with a current University of Surrey favourable ethical opinion in place?</b>	NO
---	--	----

5	<b>Does your research involve any animals, animal data or animal derived tissue, including cell lines?</b>	NO
---	--	----

7	<b>Does your project involve any of the following: human participants (including human data and/or any human tissue*); or is your project linked to engineering and/or the physical sciences?</b>	YES
---	---	-----

13	<p><b>Does your project involve any type of human tissue research? This includes Human Tissue Authority (HTA) relevant, or irrelevant tissue (e.g. non-cellular such as plasma or serum), any genetic material, samples that have been previously collected, samples being collected directly from the donor or obtained from another researcher, organisation or commercial source.</b></p>	NO
----	--	----

14	<p><b>Does your research involve exposure of participants to any hazardous materials e.g. chemicals, pathogens, biological agents or does it involve any activities or locations that may pose a risk of harm to the researcher or participant?</b></p>	NO
----	---	----

15	<p><b>Will any activities in your research take place in the Surrey Clinical Research Building (CRB)?</b></p>	NO
----	---	----

16	<p><b>Will you be accessing any organisations, facilities or areas that may require prior permission? This includes organisations such as schools (Headteacher authorisation), care homes (manager permission), military facilities etc. If you are unsure, please contact RIGO.</b></p>	NO
----	--	----

17	<p><b>Will you be working with any collaborators or third parties to deliver any aspect of the research project?</b></p>	NO
----	--	----

18	<p><b>Is your project a service evaluation or an audit?</b></p>	NO
----	---	----

19	<p><b>Does your funder, collaborator or other stakeholder require a mandatory ethics review to take place at the University of Surrey?</b></p>	NO
----	--	----

20	Are you undertaking security-sensitive research, as defined in the text above?	NO
----	--	----

21	<p>Does your project process personal data<sup>1</sup>? Processing covers any activity performed with personal data, whether digitally or using other formats, and includes contacting, collecting, recording, organising, viewing, structuring, storing, adapting, transferring, altering, retrieving, consulting, marketing, using, disclosing, transmitting, communicating, disseminating, making available, aligning, analysing, combining, restricting, erasing, archiving, destroying.</p>	YES
21.a	Please ensure that you adhere to the data protection guidance	I am an UG or PGT student and I understand that I have to abide by the 'Data protection and security for undergraduate and postgraduate taught student projects' policy found at <a href="https://research.surrey.ac.uk/ethics">https://research.surrey.ac.uk/ethics</a>

22	Does your project require the processing of special category <sup>2</sup> data?	YES
----	---	-----

22.a	<b>Please ensure that you adhere to the data protection guidance</b>	I am an UG or PGT student and I understand that I have to abide by the 'Data protection and security for undergraduate and postgraduate taught student projects' policy found at <a href="https://research.surrey.ac.uk/ethics">https://research.surrey.ac.uk/ethics</a>
------	--	--

23	<b>Are you using a platform, system or server<sup>3</sup> to collect, process and/or store any personal and/or special category data?</b>	YES
23.a	<b>Please list the platforms, systems and/or servers that you are using in your research.</b>	Google Drive
23.b	<b>You must delete all research data including personal information from the external platform, system or server you are using upon completion of your research. Students: Your supervisor must support the use of the external platform, system or server and must agree to taking on the responsibility to ensure you delete the data upon completion of your study.</b>	Students: My supervisor has approved the platform, system or server I am using and I will delete all data from external platforms, systems or servers used upon completion of my research and my supervisor agrees to take on the responsibility to ensure this.

24	Does your research involve any of the above statements? If yes, your study may require external ethical review or regulatory approval	NO
----	---	----

25	Does your research involve any of the above? If yes, your study may require external ethical review or regulatory approval	NO
----	--	----

26	Does your project require ethics review from another institution? (For example: collaborative research with the NHS REC, the Ministry of Defence, the Ministry of Justice and/or other universities in the UK or abroad)	NO
----	--	----

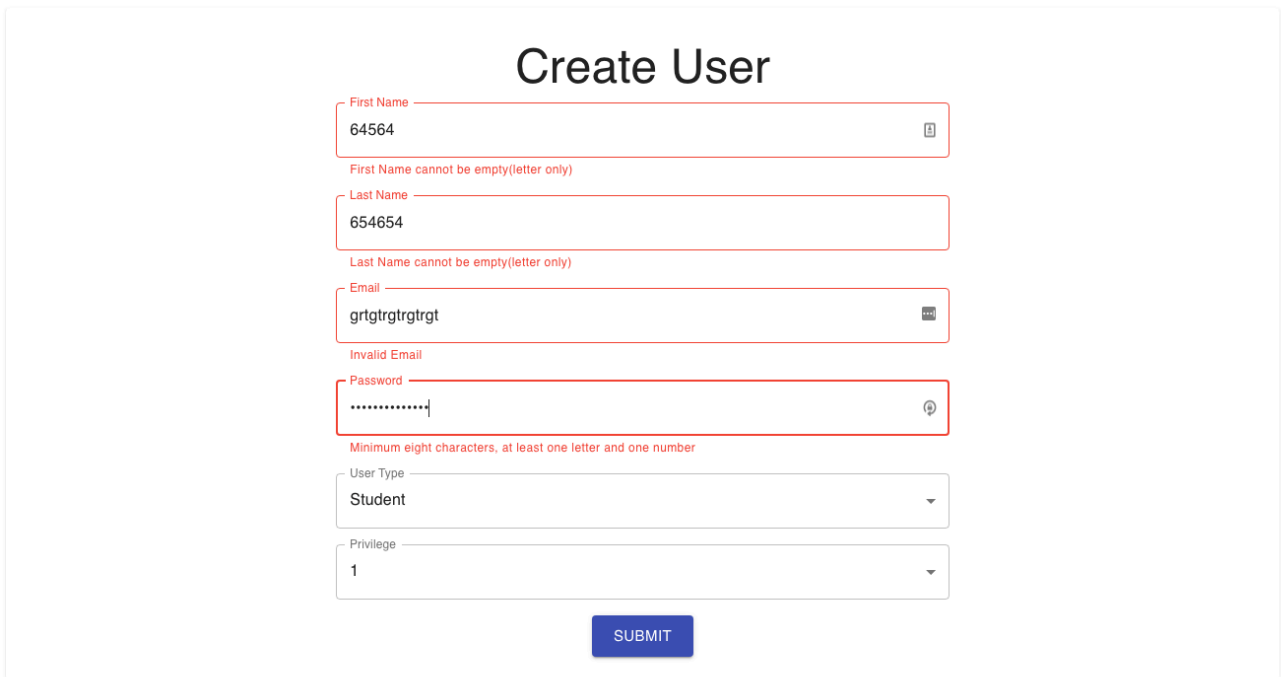
30	Declarations	<ul style="list-style-type: none"> <li>• I confirm that I have read the University's Code on Good Research Practice and ethics policy and all relevant professional and regulatory guidelines applicable to my research and that I will conduct my research in accordance with these.</li> <li>• I confirm that I have provided accurate and complete information regarding my research project</li> <li>• I understand that a false declaration or</li> </ul>
----	--------------	--

providing misleading information will be considered potential research misconduct resulting in a formal investigation and subsequent disciplinary proceedings liable for reporting to external bodies

- I understand that if my answers to this form have indicated that I must submit an ethics and governance application, that I will NOT commence my research until a Favourable Ethical Opinion is issued and governance checks are cleared. If I do so, this will be considered research misconduct and result in a formal investigation and subsequent disciplinary proceedings liable for reporting to external bodies.
- I understand that if I have selected any options on the higher, medium or lower risk criteria then I MUST submit an ethics and governance application (EGA) for review before conducting any research. If I have NOT selected any of the higher, medium or lower risk criteria, I understand I can proceed with my research without review and acknowledge that my SAGE answers and research project will be subject to audit and inspection by the RIGO team at a later date to check compliance

31	<b>If I am conducting research as a student:</b>	<ul style="list-style-type: none"><li>• I confirm that I have discussed my responses to the questions on this form with my supervisor to ensure they are correct.</li><li>• I confirm that if I am handling any information that can identify people, such as names, email addresses or audio/video recordings and images, I will adhere to the security requirements set out in the relevant Data protection Policy</li></ul>
----	--	--

## 11.5 Appendix E UI Validation



The screenshot shows a 'Create User' form with the following fields and validation messages:

- First Name:** Input field contains '64564'. Validation message: 'First Name cannot be empty(letter only)'. A red border highlights the input field.
- Last Name:** Input field contains '654654'. Validation message: 'Last Name cannot be empty(letter only)'. A red border highlights the input field.
- Email:** Input field contains 'grtgrtgrtgrt'. Validation message: 'Invalid Email'. A red border highlights the input field.
- Password:** Input field contains '.....'. Validation message: 'Minimum eight characters, at least one letter and one number'. A red border highlights the input field.
- User Type:** Dropdown menu with 'Student' selected.
- Privilege:** Dropdown menu with '1' selected.

A blue 'SUBMIT' button is located at the bottom center of the form.

Figure 66: User Form Validation Evidence



# Create Contest for WOKEWeekly

## Candidates

There are currently no candidates, add some to start the election

### Contest Details

Contest Name

Contest name field cannot be empty(letters only)

Contest Description

Contest Description cannot be empty(letters only)

[CREATE CONTEST](#)

### Add Candidate

Candidate Name

Candidate Name field cannot be empty(letters only)

Candidate Email

Invalid email

[ADD CANDIDATE](#)

Figure 67: User Validation Evidence

## 11.6 Appendix F UI Pages Final Version

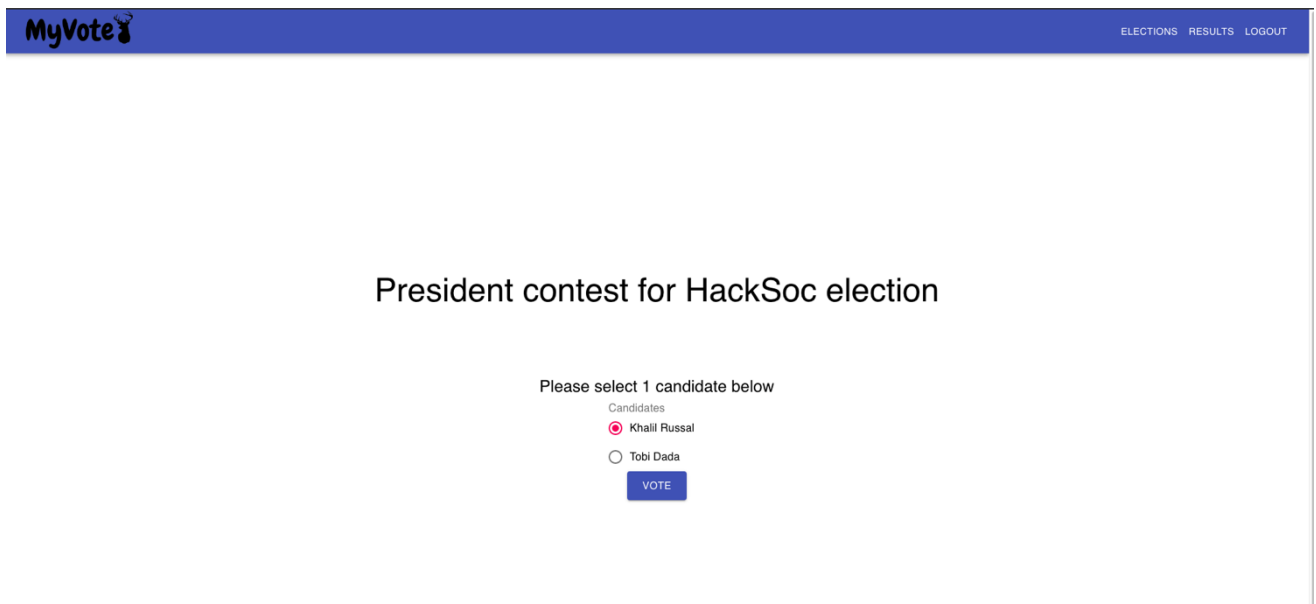


Figure 68: Voting Page Desktop



Hi Tobi Dada please enter the Election Id and Contest Id

GO TO CONTEST

Figure 69: Go to Contest Page Desktop



Welcome Tobi Dada what would you like to do

VOTE IN ELECTION

Figure 70: Student Dashboard

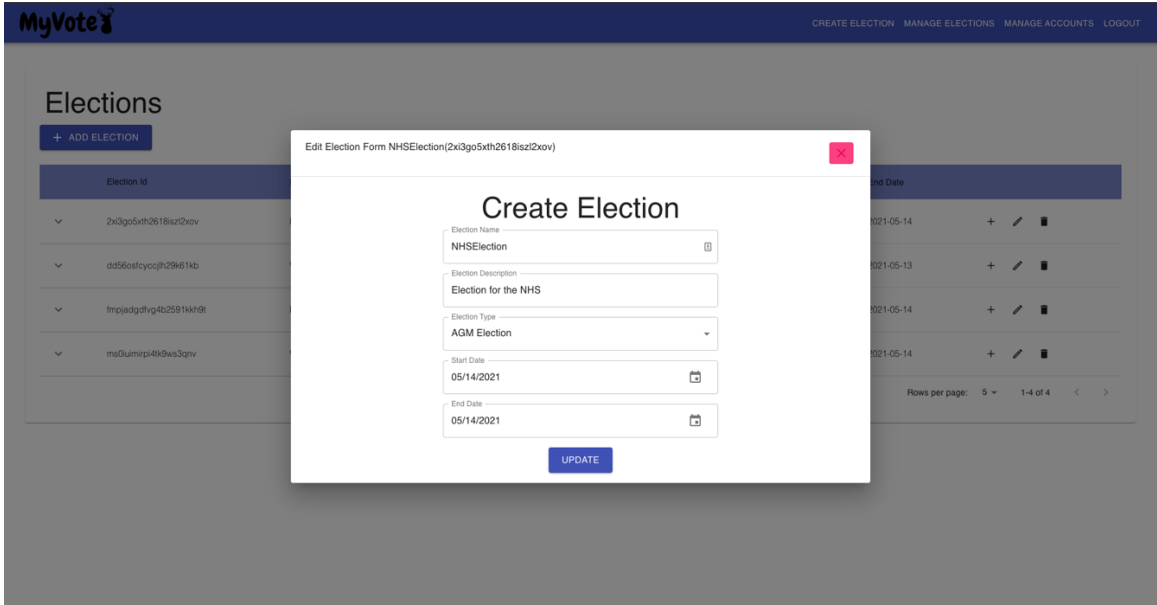


Figure 71: Edit Election Page

## Create Contest for NHSElection

### Candidates

There are currently no candidates, add some to start the election

Contest Details	Add Candidate
<input type="text" value="Contest Name"/>	<input type="text" value="Candidate Name"/>
<input type="text" value="Contest Description"/>	<input type="text" value="Candidate Email"/>
<input type="button" value="CREATE CONTEST"/>	<input type="button" value="ADD CANDIDATE"/>

Figure 72: Create Contest Page









### Create User

<input type="text" value="First Name"/>
<input type="text" value="Last Name"/>
<input type="text" value="Email"/>
<input type="text" value="Password"/>
User Type Student
Privilege 1
<input type="button" value="SUBMIT"/>

Figure 73: Create Election Page

## Users

+ ADD USERS












Email	First Name	Last Name	User Type	Privilege	
AO00577@SURREY.AC.UK	Tobi	Dada	Admin	3	 
BOLADALE@GMAIL.COM	Bola	Dale	Student	1	 
KHALIL@GMAIL.COM	Khalil	Russal	Student	1	 
TOBIDADA17@GMAIL.COM	Tobi	Dada	Student	1	 

Rows per page: 5 ▾ 1-4 of 4 &lt; &gt;

Figure 74: User Account Table

## Elections

+ ADD ELECTION

Election Id	Election Name ↑	Election Description	Election Type	Start Date	End Date		
2xi3go5xth2618iszl2xov	NHSElection	Election for the NHS	AGM Election	2021-05-14	2021-05-14	+  	
<b>Contests</b>							
Contest Id	Contest Name	Contest Description	Contest Status	Contest URL			
p89krn4j8cthrzdmecj	Matron	Matron Election	OPEN	2xi3go5xth2618iszl2xov/p89krn4j8cthrzdmecj	  		
dd56ocfycyccjh29k61kb	WOKEWeekly	The AGM for WOKEWeekly		AGM Election	2021-05-13	2021-05-13	+  
fmpjadgdtvg4b2591kh9t	HackSoc	HackSoc Election		AGM Election	2021-05-14	2021-05-14	+  
ms0uimrpi4k9ws3gnv	WOKEWeekly	AGM for WOKEWeekly		AGM Election	2021-05-14	2021-05-14	+  

Rows per page: 5 ▾ 1-4 of 4 &lt; &gt;

Figure 75: View Election Table

### Create Election

AGM Election

05/15/2021

05/15/2021

Figure 76: Create Election Form



Welcome Tobi Dada what would you like to do

- 
- 
- 

Figure 77: Admin Dashboard Page