Interm Report

Online Voting System

Aodan Hardy B00826427

Table of Contents

1. Introduction

1.1 Problem Statement

Concerns over the security and integrity of voting systems has led to low participation rates and trust in many organisations' elections. This system aims to address these issues by guaranteeing each voter that their vote will remain anonymous and will be accurately counted. Blockchain technology will be used to store each vote across distributed ledgers, ensuring that the process remains secure, transparent, and resistant to interference.

The system is designed to be versatile, with customisable election settings that allow it to be adapted for use across a wide variety of organisations and voting scenarios.

1.2 Project Aim

To develop a secure and transparent online voting system using blockchain technology to ensure the integrity and accuracy of elections.

1.3 Project Objectives

1. To design a system that enables election organisers to create election profiles, manage participants, and invite voters through an email system containing unique voting links.

2. To implement Ethereum blockchain technology for processing and securely storing votes, ensuring transparency and immutability throughout the election process.

3. To develop a mechanism for automatically retrieving and tallying votes from the blockchain after the voting period ends, providing accurate and verifiable results to the election organisers.

2. Literature Review

2.1. Introduction to Voting Systems

In this section I will present my findings from research I've conducted on the different types of voting that will be integrated into my voting system. It's important to note that only the effects of one-off elections are relevant to this project. This means I won't be covering how different voting types affect the outcome of elections with multiple constituencies. Instead, I will focus on how each individual constituencies select their representatives, as well as how these systems apply to presidential elections and referendums.

First Past the Post

The first past the post voting system is the simplest form of multi-candidate voting. It is used in parliamentary elections in the UK, Canada and Australia and used in US presidential elections by individual states. In this system, each voter selects one candidate, and the candidate with the most votes wins, a majority is not necessary. (Baalman, 2018)

Ranked-Choice Voting (RCV)

RCV is a system where voters rank candidates by preference instead of choosing just one. The selection process is processed in rounds. If no candidate wins an outright majority in the first round, the candidate with the fewest votes is eliminated, and their votes are reallocated to the voters next choice. This repeats until a candidate gets a majority of votes. RCV can also be used to select more than one winner. All that needs to change is the number of votes needed to win, which is calculated by the number of winning spaces plus one, divided by the total number of votes, plus one. The process concludes when all spaces are filled. RCV helps promote fairer representation, it reduces the impact of "spoiler" candidates, and improves voter satisfaction, as they will feel more able to vote for who they truly want rather than voting strategically (FairVote, 2024).

Approval Voting

Approval voting is a much less common system where voters can vote for as many candidates as they want, without ranking them. The candidate with the most votes wins. This method ensures that the most approved of candidate wins, resulting in an increase of voter satisfaction by allowing voters to support multiple candidates without worrying about how other people are voting, reducing strategic voting. By widening the field of viable candidates, approval voting results in more representative outcomes and incentivises candidates to appeal to a wider range of voters. (Electionscience.org, 2024)

2.2. Traditional vs Digital Voting Systems

Traditional paper based voting systems have been foundational to the democratic process. These involve a person voting at physical polling stations, where voters mark their choices on paper ballots, which are then manually counted. The simplicity of this

method ensures that voting remains accessible to a wide range of citizens and allows for a straightforward counting process, promoting trust among the electorate.

Since this has been the process of voting for a long time, there are many rules in place to ensure the integrity of the voting process. One key measure is the requirement for voter identification. Voters are often asked to provide ID to prevent impersonation at the polling station. In some countries, this is strictly enforced, while in others, it's a matter of verifying identity through other methods, like government provided voting cards. Another important security feature is the sealing of ballot boxes once the voting ends, preventing tampering during the transportation process. Representatives from political parties are normally allowed to monitor the entire process, including the counting of votes. Paper ballots are also often stored in a secure location to allow for recounts in the event of a dispute.

This manual counting process remains transparent throughout, with officials physically counting and verifying the results, while observers from the public or representatives from parties, ensuing there is not vote tampering. This process, while costly and time consuming, offers a level of transparency and accountability that is valued by many voters. However, the manual counting process also makes paper-based voting more vulnerable to human error or delays, which can be remedied by using more modern technologies like electronic voting.

Digital voting systems have been considered for many years, especially as technology continues to advance. Despite potential benefits, such as increased accessibility and convenience for voters, they are not yet as widespread as traditional paper-based systems. One major reason for this reluctance is trust. Voters need assurance that their vote will be counted correctly. Many people are wary of the possibility of hacking, vote tampering, or technical failures that could undermine the integrity of the election. Ensuring the security of a digital system is an ongoing challenge, with the risk of cyber-attacks posing a significant threat to public confidence in the voting process (CORDIS, 2021).

Another concern is anonymity. In traditional voting systems, voters physically cast their ballots, ensuring that their choices remain confidential. However, in a digital system, there is an inherent risk that voter anonymity could be compromised. While encryption and other security measures can be used to protect voter identities, the fear remains that digital systems could expose voters. (Cetinkaya, 2015).

Lastly, there are concerns over security. Digital voting systems need to be protected against a wide range of risks, including hacking, malware, and data breaches. As INRIA (2024) explains for these systems to gain widespread acceptance, they must offer guarantees that they are as secure as, or more secure than, traditional voting methods. The reliability of digital systems in handling elections, particularly large-scale ones, is still questionable, as even small technical issues could lead to major disruptions.

Overall, while digital voting systems have clear advantages, the combination of concerns about security, trust, and voter anonymity has hindered their widespread adoption.

2.3. Blockchain Voting Systems

Blockchain technology has gained a significant amount of attention for its potential to revolutionise industries, including voting systems. The main appeal of blockchain in voting its ability to offer transparency, security, and immutability of vote data. In blockchain-based voting systems, votes are recorded as transactions in a decentralised ledger, making it very difficult to tamper with the voting data once it is recorded. This ensures that all votes are securely stored and that the integrity of the election process is preserved.

According to Built (2023), “Blockchain works to be inherently secure and ensure original information is unchangeable, providing a single-source-of-truth and acting as a tracking method for any data recorded. When applied to voting, this could mean the opportunity for unaltered votes, voting transparency, increased online accessibility and more organised voting operations.”
This highlights the advantage blockchain technology has which, unlike traditional digital systems, stores data in a decentralised ledger addressing security concerns and improving voter trust.

Real-world applications of blockchain technology in voting systems have already been trialled in several places, including West Virginia and Estonia. West Virginia conducted a pilot in 2018, where overseas military people used a blockchain-based mobile voting app to cast their ballots in the midterm elections. This initiative aimed to improve accessibility and security for voters in remote locations. The use of blockchain technology ensured that votes could be securely cast and verified without the risk of interference or manipulation (Miller, 2018).

Similarly, Estonia has been at the forefront of digital voting, having introduced its i-Voting system in 2005. While Estonia's system does not use blockchain, there are ongoing discussions about how blockchain could further enhance the security of online voting. The use of blockchain in such systems could potentially offer an extra layer of security and anonymity for voters, while maintaining integrity of the voting process (PwC, 2019).

2.4. Challenges and Limitations of Blockchain Voting

Blockchain voting offers significant potential for security, transparency, and reliability in voting systems. However, this technology is not without its challenges. From scalability to legal and regulatory issues, blockchain-based voting systems face multiple limitations.

Double Spend Problem and Consensus Protocol
In blockchain-based voting systems, the "double spend" problem is a potential threat,

where a voter could vote more than once by reusing the same transaction ID. Blockchain consensus protocols, Proof of Work (PoW) and Proof of Stake (PoS), are typically used to secure transactions and prevent double-spending by validating entries across networks. When it comes to voting, these protocols may struggle with scalability and energy efficiency, making them less ideal for vote validation. Additionally, deciding who controls the validation process, whether through decentralised miners or selected validators presents challenges, for example, what is to stop a powerful foreign country, or an organisation with a stake in an election, taking control of a large portion of miners, and changing the vote data while its being stored. Even if this was recognised, it could destroy public trust in the voting process. Different consensus protocols also come with trade-offs. PoW, while highly secure, requires significant energy consumption, while PoS offers greater energy efficiency but can be more susceptible to centralisation, raising concerns about voter anonymity and public trust of the election (Maxie, 2018).

Anonymity

Ensuring voter privacy while maintaining transparency is a key challenge in blockchain-based voting. Blockchains use pseudonymity, assigning voters unique, encrypted IDs instead of personal information. However, achieving complete privacy, where no voter can be linked to their vote, requires additional encryption. One solution is Zero-Knowledge Proofs (ZKPs), which verify votes without revealing voter details. This ensures votes are securely recorded while preventing traceability, striking a balance between voter privacy and election integrity (Fazekas, 2022).

Scalability

One of the primary challenges facing blockchain voting systems is scalability. Blockchain networks that rely on “proof of work” consensus protocols, can experience significant performance drops as the number of participants increase. The time required to process and verify transactions can become a bottleneck, especially during large-scale elections:

“their systems [current blockchain voting systems] are not efficient for the national level to handle millions of transactions because they use current blockchain frameworks” (Jafar, Aziz and Shukur, 2021)

Furthermore, as the blockchain grows, maintaining efficiency while ensuring high levels of security and decentralisation becomes more complex. To address these concerns, various techniques like sharding, off-chain transactions, and layer-two solutions are being explored to enhance the scalability of blockchain-based voting systems. These solutions aim to minimise transaction costs and improve processing times, making the system more feasible for larger elections (Jafar, Aziz and Shukur, 2021).

Energy Consumption

A significant limitation of blockchain voting systems is the high energy consumption associated with transaction validation and block creation. In PoW-based systems, miners must solve complex cryptographic puzzles to validate transactions, which requires substantial computer power. This results in increased energy consumption, which has raised concerns about the environmental impact of large-scale blockchain

networks. According to recent studies, while some newer consensus protocols, like proof-of-stake (PoS), have been developed to reduce energy usage, the energy demands of PoW still pose a challenge for blockchain applications like voting, where processing and scalability are critical. Furthermore, the energy inefficiency of PoW-based systems makes them less sustainable for widespread use in democratic elections where millions of votes would need to be processed. (Sedlmeir et al., 2020).

Legal considerations
Legal considerations for implementing a blockchain voting system must address compliance with data protection laws, such as the Data Protection Act 2018 and GDPR. These regulations require safeguarding voter privacy, voter personal data, and ensuring transparency in data processing. (UK Data Service, 2021). Election systems must also follow electoral laws, guaranteeing secure, anonymous voting while preventing fraud. Additionally, international usage of the system should account for differences in electoral and data protection laws.

2.5. Conclusion

These findings highlight how blockchain technology can improve the security, transparency, and efficiency of digital voting systems. Features like immutable records and pseudonymous voter ID can reduce risks like fraud and tampering. Additionally, techniques such as Zero-Knowledge Proofs, can enhance voter privacy while preserving the transparency required for a trusted process.

Based on these findings, my project will use blockchain technology to construct an online voting system for small-scale elections, such as organisational or community voting. These situations allow for controlled testing of the system's features, ensuring it meets its objectives of secure and reliable voting.

On the other hand, it is essential to recognise the limitations of this system. Blockchain voting, in its current form, may not be suitable for public elections due to challenges like scalability and legal compliance. Thus, it is my conclusion that this project would not be suitable for public elections, due to the high stakes and vulnerability to motives for vote tampering. The mere suspicion of which could destroy public trust in the system. Instead, the scope of this project will be limited to privately organised elections, with lower stakes but also high demand for security and reliability.

3. Project Plan and Requirement Specification

3.1. Stakeholder Identification

For an online voting system, it is crucial to identify the stakeholders to ensure that the system's functionality aligns with the needs of each type of user. The following stakeholders have distinct roles within the voting process:

- **Voters**: Voters are the end-users responsible for casting ballots in elections. They require a secure, straightforward interface to authenticate their identities, view relevant election details, and submit their votes. Each voter is uniquely identified to ensure that they can cast only one vote per election, protecting the integrity of the voting process. Voters also expect privacy, knowing that their vote remains confidential and tamper-proof.
- **Customer Organisations**: These are the organisations conducting the elections. They oversee the general setup and management of elections, define voting rules, and ensure the overall success and integrity of the voting process. While they do not directly interact with the voting system, they provide the framework and parameters for each election.
- **Election Managers:** Election managers are individuals within the organisations who actively interact with the voting system through the dashboard app. They handle the configuration of each election, including adding candidates, defining voting types, and managing voter lists. Additionally, they monitor voter participation and, when needed, provide technical support to voters. Election managers need to have access to real-time data to ensure a smooth voting process. Their role is crucial in handling the election setup with the organisation's election goals.

3.2. Requirement Prioritisation Strategy Justification

The MoSCoW method will be used as the primary strategy for classifying and prioritising system requirements in this project. MoSCoW is effective for projects with limited resources or strict timelines, as it helps focus on delivering core functionalities first. By categorising requirements into levels of necessity, this method will support clear prioritisation and resource allocation. (Product Plan, 2024)

3.2.1. MoSCoW Prioritisation

The MoSCoW project priorisation method classifies requirements into “Must have”, “Should have”, “Could have”, and “Won't have”. This approach is suitable for the voting system, which must balance essential security and usability features with optional ones.

1. **Must have**: These requirements will include critical features that the system cannot operate without. The “Must have” category defines the minimum features that the project needs to have. An example of a “Must Have” would be a graphic showing the election managers the results of the election.
2. **Should have**: This category will include important but non-critical features. While not necessary for the functioning of the system, these features will enhance usability for election managers and voters. The inclusion of these items in future releases may increase user satisfaction. An example of a “Should Have” would be the ability for election organisers to view who has voted.
3. **Could have**: These desirable but non-essential features will aim to improve the user experience, for example an advanced results analysis tool that breaks down voters by characteristics such as gender, age etc.
4. **Won't have**: These requirements represent features unnecessary for the current scope of the project, for example a Machine learning model that predicts election results.

The decision to use the MoSCoW method comes from its simplicity and adaptability, especially for projects that involve security-sensitive applications like online voting. In summary, the MoSCoW method will ensure efficient resource allocation and project focus.

3.3. System Requirement Specification

3.3.1. Functional Requirements

ID	Feature	Description	MoSCoW
F1	Public website avaliable	There is a informational website that is avaliabe to everyone	S
F2	User Registration	Users can register to create and manage elections.	Mo
F3	User Login	Secure login for registered users.	Mo
F4	User Dashboard Access	Only logged in users have access to dashboard	Mo
F5	User Specific Dashboard	User will only have access to their dashboard	Mo
F6	Dashbord will list elections	Dashbord will list all elections that the organiser has made	Mo
F7	Create Election	Organisers can create elections.	Mo
F8	Chose to use Blockchain	Organisers have the option to use blockchain or not	S
F9	Add Ballots	Organisers add multiple ballots on the one election.	S
F10	FPTP Ballot	Organisers can add FPTP ballot	Mo
F11	RCV Ballot	Organisers can add RCV ballot	S
F12	YES/NO Ballot	Organisers can add YES/NO ballot	S
F13	Approval Ballot	Organisers can add approval ballot	Co
F14	Add Candidates	Add candidates	Mo
F15	Upload Voter List	Organisers can upload a CSV of voters.	Mo
F16	Voters can have 'weight'	voters can each have their own vote value instead of 1	C
F17	Start/Stop Election	Control to start and stop elections, recording timestamps.	Mo
F18	Vote tallying	Organisers can see vote tally during election	S
F19	Voter has UUID	UUIDs for enhanced security in voter table.	Mo
F20	Voting Link	A link containing UUID leads to ballot	Mo
F21	Link is emailed	Each voter gets an email with their unique link	S
F22	Vote Summary	A summary is presented to voter before ballots	S
F23	Unique Ballot for Vote type	A specific ballot interface for each voting type	Mo
F24	Submit Vote	Voters submit for each ballot.	Mo
F25	Progress through Ballots	Voters navigate through ballots.	Mo
F26	Blockchain Integration	Record votes on the blockchain.	Mo
F27	Local Database Storage	Store votes in PostgreSQL if blockchain is not used.	S
F28	Vote Summary	Display a summary to voters after submission.	S
F29	Voters can only vote once	Voter table has boolean value hasVoted	Mo
F30	Result Generation	Fetch and store votes in the election table after voting ends.	Mo
F31	Display Results	Organisers can view a breakdown of results by ballot.	Mo
F32	Invalid ID Handling	Show errors for invalid voter IDs or expired links.	S
F33	Invalid Ballot Submission	Prevent incomplete or invalid ballots from being submitted.	Mo
F34	Manage Elections	Organisers can edit and delete their elections.	S
F35	Result Access	Organisers can access election results.	Mo
F36	Responsive Design	Ensure the voting platform is mobile-friendly.	S
F37	User friendly Interface	Provide a clear, accessible interface for organizers and voters.	S

3.3.2. Non-Functional Requirements

ID	Feature	Description	MoSCoW
NF1	Performance	operations load quickly, even with high traffic.	Mo
NF2	Scalability	The system supports high volume of users and voters.	Mo
NF3	Fault Tolerance	The system should handle failure.	Mo
NF4	Data Integrity	Ensure that data is accurately stored.	Mo
NF5	Accessibility	Design the system to be accessible to users with disabilities.	S
NF6	User-Friendly Interface	interface should Minimize confusion and making voting straightforward.	S
NF7	Multilingual Support	Offer multiple language options for users.	C
NF8	Data Privacy	Ensure that personal information is kept confidential and secure.	Mo
NF9	Voter Authentication	Ensure that only authorized voters can access their specific elections.	Mo
NF10	Immutability of Votes	Ensure votes cannot be altered once cast.	Mo
NF11	Code Documentation	Provide detailed documentation for the code.	S
NF12	Logging and Monitoring	Implement logging to monitor key processes, errors, and security events.	S
NF13	Blockchain Compatibility	Design the system to be compatible with Ethereum.	Mo
NF14	Database Integration	Ensure the system is compatible with PostgreSQL.	Mo
NF15	Optimized Resource Usage	Optimize the system to minimise resource usage.	S
NF16	High Uptime	Ensure the system is available and operational for the entire voting period.	S
NF17	Regular Data Backup	Implement regular backups for all data on posgres.	S

3.4. Selected Software Lifecycle Methodology Justification

For this online voting system project, the Agile methodology has been selected as the preferred software development lifecycle approach.

3.4.1 Agile Methodology Overview

The Agile approach is based on short development cycles, known as “sprints,” which focus on incremental development. Each sprint delivers a functional part of the system, allowing for early testing, feedback, and adjustments. Agile’s emphasis on stakeholder collaboration aligns well with this project’s need to respond to feedback from both election organisers and voters. This iterative approach ensures that security, usability, and performance are addressed early in the process, allowing the project team to adapt the design as needed to meet evolving requirements. (Brush and Silverthorne, 2022)

3.4.2 Justification for Using Agile

- **Flexibility for Changing Requirements**: In a project like this, where secure voting systems and user needs may shift based on testing and feedback, Agile’s iterative approach supports continuous refinement.

- **Early and Frequent Testing**: With Agile, each sprint includes testing, which is critical for identifying security vulnerabilities early. Since this online voting system will require a great deal of security, including encryption and blockchain validation, frequent testing will be key to ensuring that each iteration addresses potential risks.

3.4.3 Comparison with Other Methodologies

While the Waterfall model is useful for projects with fixed requirements, its linear approach would not provide the flexibility needed to respond to security or usability testing feedback quickly. The Agile model’s iterative nature supports continuous improvement and allows for adjustments, making it a more suitable choice for this project.

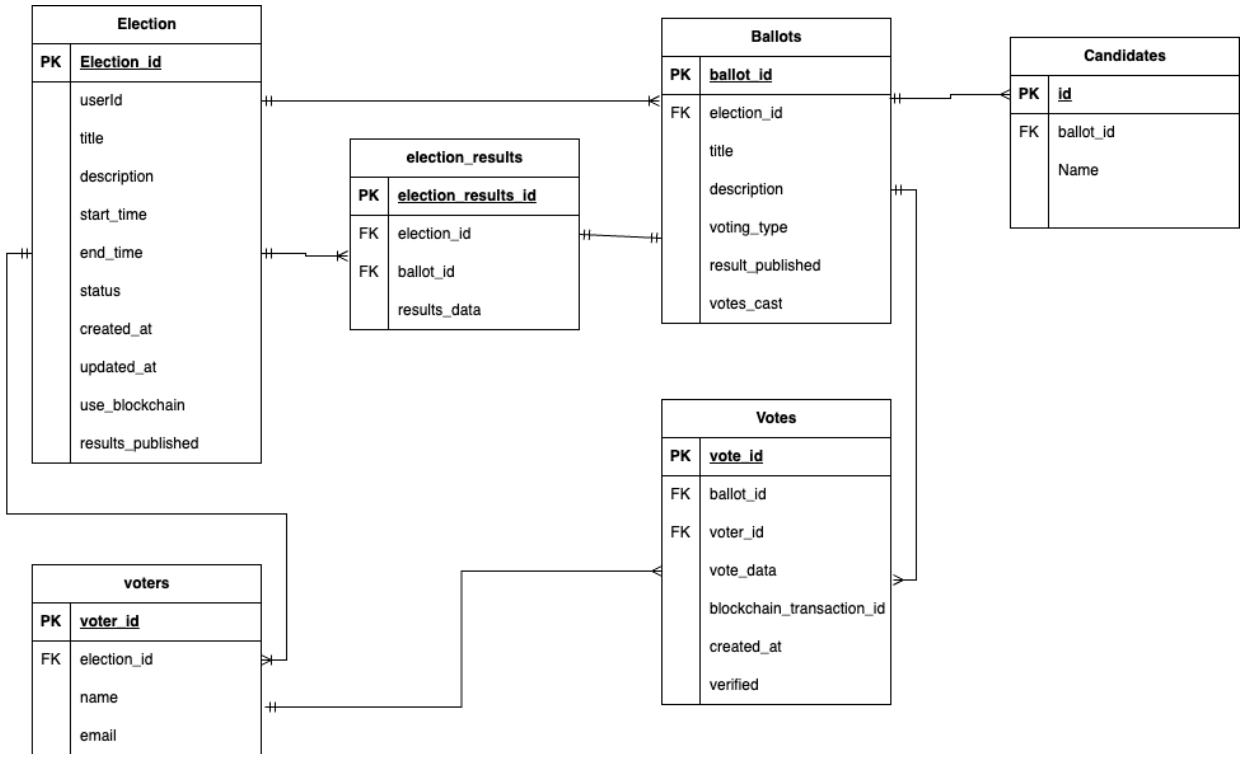
3.5. Implementation Plan

3.5.1. Work Breakdown Structure and Effort Estimate

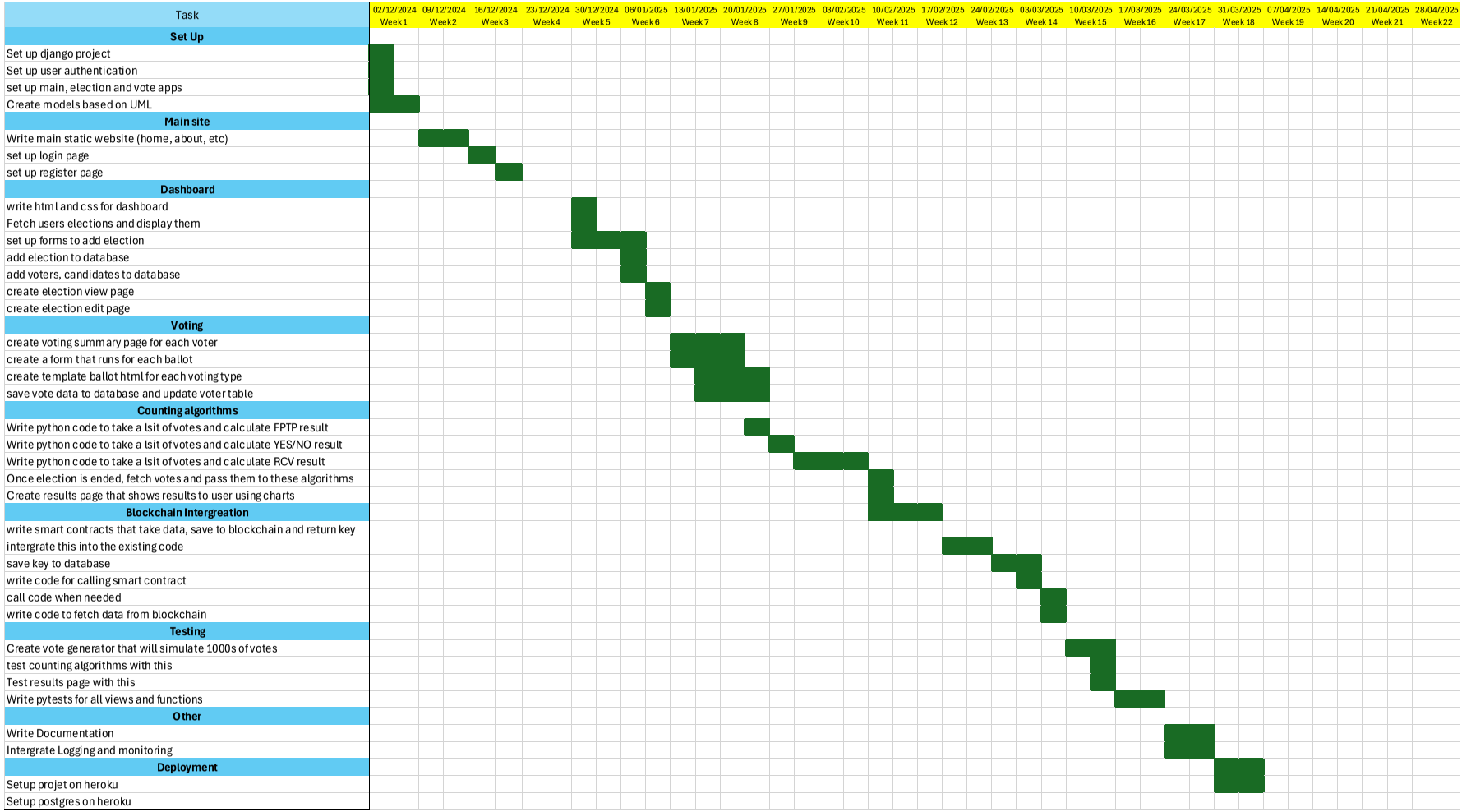
Task	Time (Hours)
Set Up	
Set up Django project	1
Set up user authentication	1
Set up main, election and vote apps	1
Create models based on UML	3
Set Up Total	6
Main site	
Write main static website (home, about, etc)	10
Set up login page	3
Set up register page	3
Main site total	16
Dashboard	
Write html and CSS for dashboard	8
Fetch users' elections and display them	2
Set up forms to add election	15
Add election to database	2
Add voters, candidates to database	1
Create election view page	3
Create election edit page	3
Dashboard Total	34
Voting	
Create voting summary page for each voter	2
Create a form that runs for each ballot	8
Create template ballot html for each voting type	3
Save vote data to database and update voter table	3
Voting Total	16
Develop counting algorithms for each voting type	
Write python code to take a list of votes and calculate FPTP result	5
Write python code to take a list of votes and calculate YES/NO result	3
Write python code to take a list of votes and calculate RCV result	20
Once election is ended, fetch votes and pass them to these algorithms	10
Create results page that shows results to user using charts	25
Counting Algorithms Total	63
Blockchain Integration	
Write smart contracts that will take data, save to blockchain and return key	20
Intergrate this into the existing code	15
Save key to database	2

Write code for calling smart contract	5
Call code when needed	1
Write code to fetch data from blockchain	15
Blockchain Intergreation Total	58
Testing	
Create vote generator that will simulate 1000s of votes	10
Test counting algorithms with this	10
Test results page with this	10
Write pyTests for all views and functions	20
Testing Total	50
Other	
Write Documentation	10
Integrate Logging and monitoring	15
Other Total	25
Deployment	
Setup project on Heroku	15
Setup Postgres on Heroku	10
Deployment Total	25
Project Estimate Total (Hours)	293

3.5.2. Database Schema UML



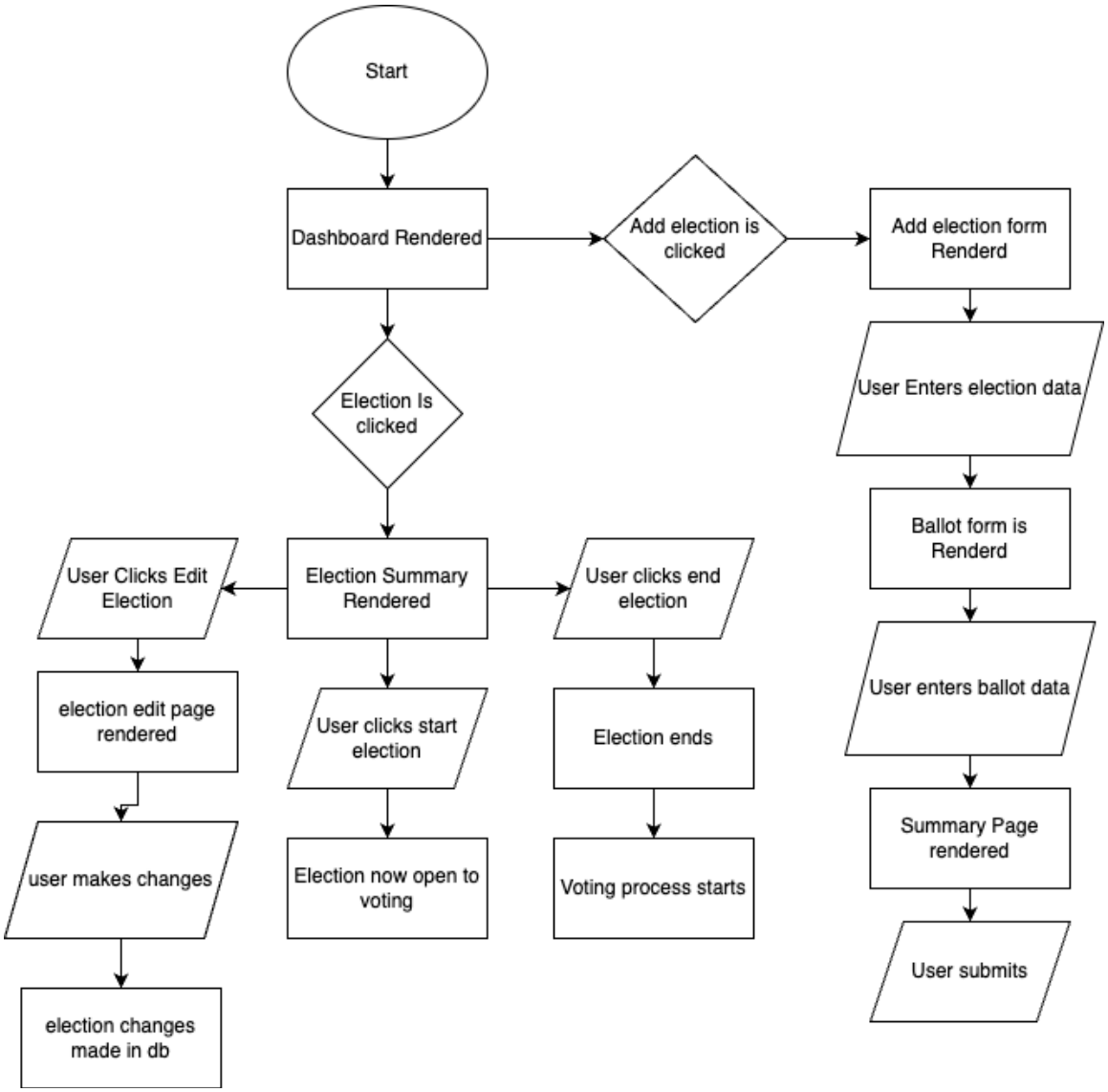
3.5.3. Gantt Chart



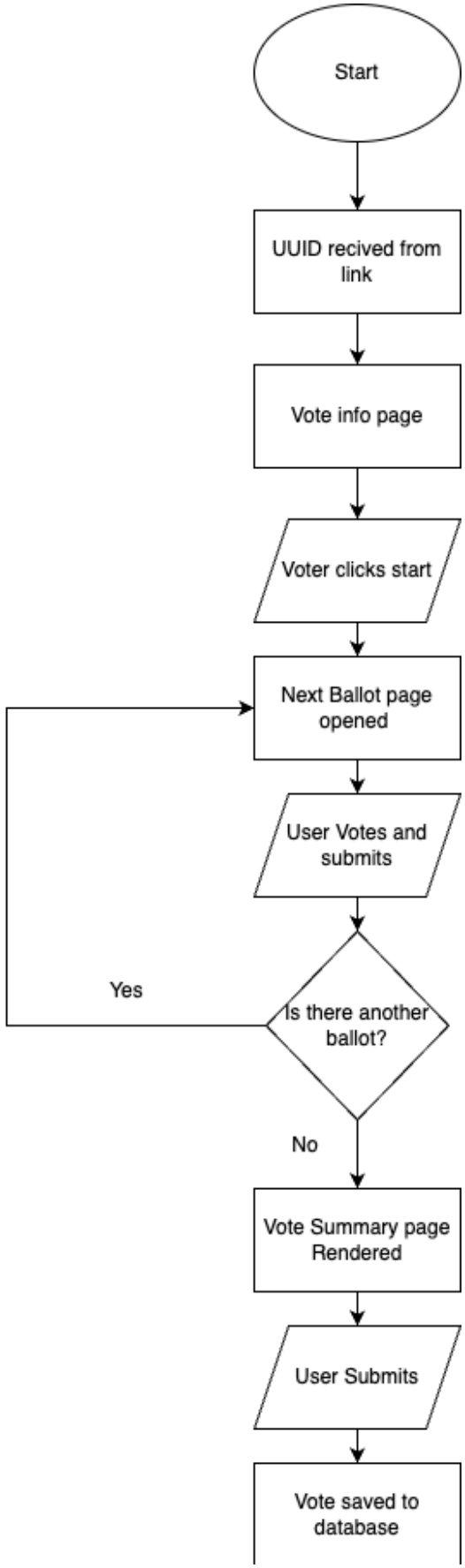
Task	02/12/2024 Week 1	09/12/2024 Week 2	16/12/2024 Week 3	23/12/2024 Week 4	30/12/2024 Week 5	06/01/2025 Week 6	13/01/2025 Week 7	20/01/2025 Week 8	27/01/2025 Week 9	03/02/2025 Week 10	10/02/2025 Week 11	17/02/2025 Week 12	24/02/2025 Week 13	03/03/2025 Week 14	10/03/2025 Week 15	17/03/2025 Week 16	24/03/2025 Week 17	31/03/2025 Week 18	07/04/2025 Week 19	14/04/2025 Week 20	21/04/2025 Week 21	28/04/2025 Week 22
Set Up																						
Set up django project																						
Set up user authentication																						
set up main, election and vote apps																						
Create models based on UML																						
Main site																						
Write main static website (home, about, etc)																						
set up login page																						
set up register page																						
Dashboard																						
write html and css for dashboard																						
Fetch users elections and display them																						
set up forms to add election																						
add election to database																						
add voters, candidates to database																						
create election view page																						
create election edit page																						
Voting																						
create voting summary page for each voter																						
create a form that runs for each ballot																						
create template ballot html for each voting type																						
save vote data to database and update voter table																						
Counting algorithms																						
Write python code to take a lsit of votes and calculate FPTP result																						
Write python code to take a lsit of votes and calculate YES/NO result																						
Write python code to take a lsit of votes and calculate RCV result																						
Once election is ended, fetch votes and pass them to these algorithms																						
Create results page that shows results to user using charts																						
Blockchain Intergreation																						
write smart contracts that take data, save to blockchain and return key																						
intergrate this into the existing code																						
save key to database																						
write code for calling smart contract																						
call code when needed																						
write code to fetch data from blockchain																						
Testing																						
Create vote generator that will simulate 1000s of votes																						
test counting algorithms with this																						
Test results page with this																						
Write pytests for all views and functions																						
Other																						
Write Documentation																						
Intergrate Logging and monitoring																						
Deployment																						
Setup projet on heroku																						
Setup postgres on heroku																						

3.5.4 Flow Chart

Dashboard Flow Chart

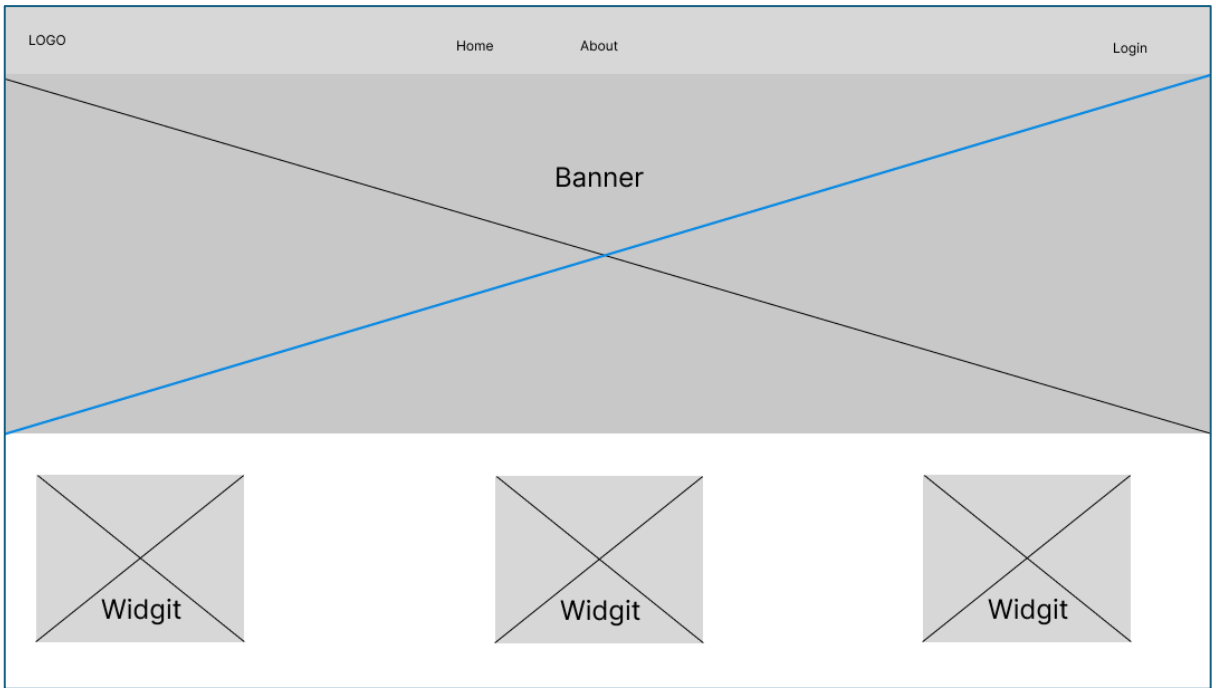


Voting Flow Chart

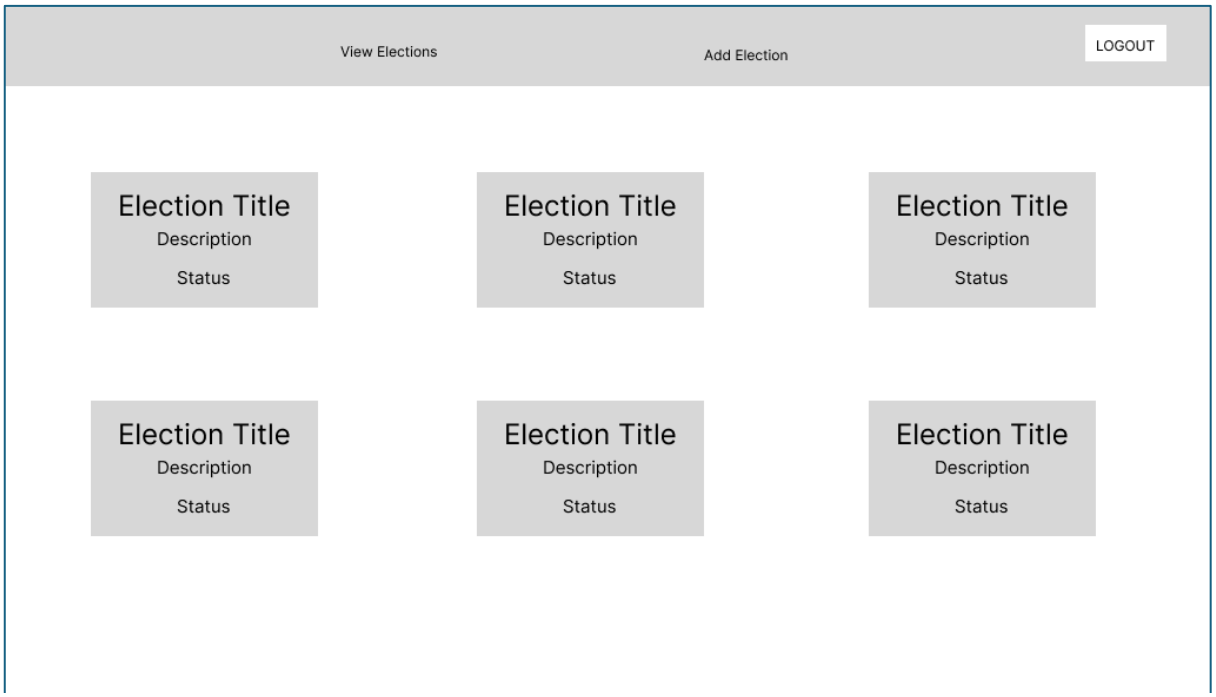


3.5.5. Wireframes

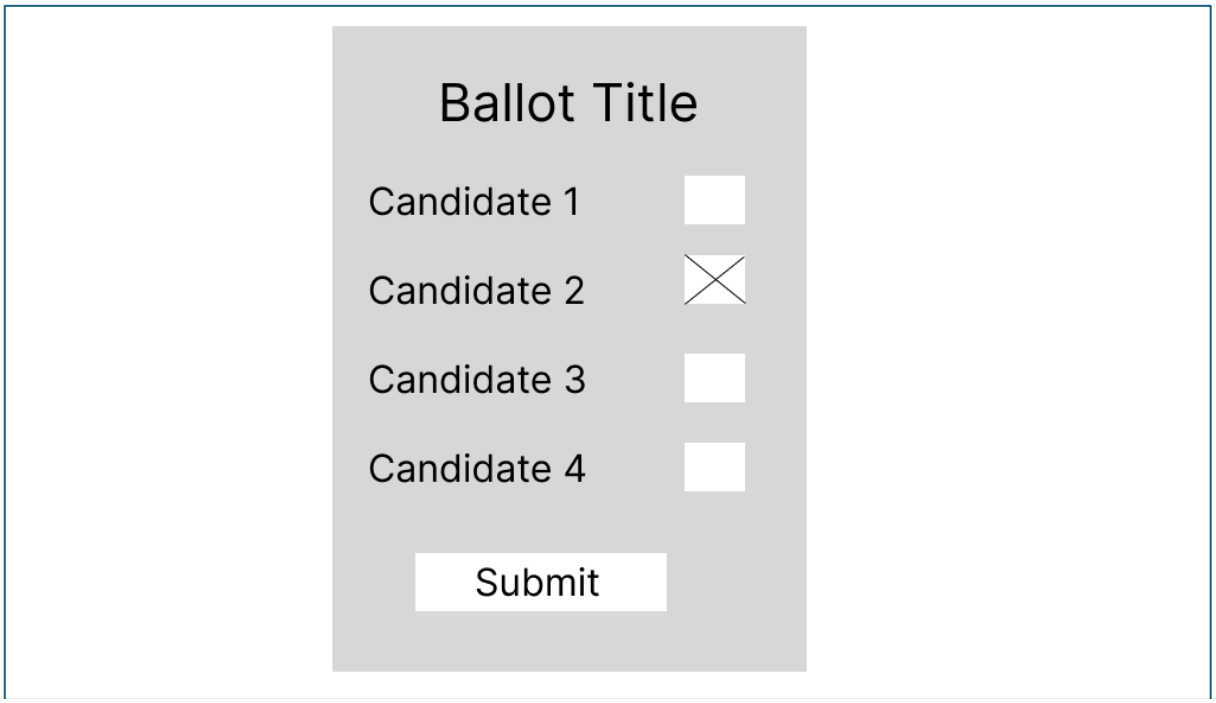
Home



Dashboard



Ballot


Ballot Title
Candidate 1
Candidate 2
Candidate 3
Candidate 4
Submit

3.6. Verification Plan

“The verification plan identifies the procedures and methods to be used for verification, including the development of test benches and automation. The verification plan is usually distinct from the Verification Tests themselves.” (www.sciencedirect.com, n.d.)

Since this project is adopting the Agile development methodology, at the end of each iteration, the new features should be vigorously tested before merged with the main branch. This includes writing unit tests, which test individual components (classes and functions) and integration testing, testing how these new components work with other components of the project.

3.6.1. Unit Testing

Unit testing will ensure that individual components of the system function correctly and independently. Tests will focus on core functions and logic without dependencies on other parts of the system. In my personal professional experience I have carried out industry standard testing on major software projects.

Examples of unit tests include:

- Vote Submission Validation: Ensure votes with valid and invalid formats are handled correctly.
- Authentication: Verify login and registration functions using correct and incorrect credentials.
- Blockchain Interactions: Test whether vote data is properly formatted before being sent to the blockchain.

Testing will be conducted using tools such as pyTests, and mock data will simulate different scenarios to ensure reliability. (GeeksforGeeks, 2017)

3.6.2. Integration Testing

Integration testing will ensure that various components of the system work together. It focuses on verifying the flow of data and interactions between different parts of the project (www.fullstackpython.com, n.d.).
Key areas for testing include:

- Frontend and Backend Communication.
- Backend and Blockchain Integration.
- Database Interactions.

3.6.3. Acceptance Testing

Creating vote generator

Acceptance testing will validate that the system meets user requirements and performs as expected (GeeksforGeeks, 2019). It will involve creating realistic election mocks. To do this I will need to create a vote generator. Key test scenarios include:

- Mock Elections: Conduct a full election process, from setting up an election to submitting votes and viewing results.

- Error Handling: Validate that the system handles invalid inputs or unexpected actions gracefully.

Feedback from these tests will guide final adjustments.

References

Baalman, M. (2018). *First-past-the-post: a rogue's practice?* [online] On Elections. Available at: https://onelections.net/2018/07/31/first-past-the-post-a-rogues-practice/ [Accessed 13 Nov. 2024].

FairVote (2024). *Ranked Choice Voting*. [online] FairVote. Available at: https://fairvote.org/our-reforms/ranked-choice-voting/.

Electionscience.org. (2024). *What is Approval Voting? | Center for Election Science*. [online] Available at: https://electionscience.org/education/approval-voting.

CORDIS. (2021). Anxieties over internet voting reflect wider social concerns. CORDIS Results. [online] Available at: https://cordis.europa.eu/article/id/448713-anxieties-over-internet-voting-reflect-wider-social-concerns [Accessed 13 November 2024].

Cetinkaya, D. (2015). Anonymity in eVoting protocols. ResearchGate. [pdf] Available at: https://www.researchgate.net/profile/Deniz-Cetinkaya/publication/250698361_ANONYMITY_IN_EVOTING_PROTOCOLS/links/54ddfdc70cf2814662ec34dc/ANONYMITY-IN-EVOTING-PROTOCOLS.pdf [Accessed 13 November 2024].

INRIA. (2024). Security of electronic voting: Digital security and confidentiality. INRIA. [online] Available at: https://www.inria.fr/en/security-electronic-voting-digital-security-confidentiality [Accessed 13 November 2024].

Built In (2023) Blockchain Voting: How It Could Secure the Future of Elections. Available at: https://builtin.com/blockchain/blockchain-voting-future-elections (Accessed: 13 November 2024).

PwC (2019) Estonia: The digital republic secured by blockchain, PwC. Available at: https://www.pwc.com/gx/en/services/legal/tech/assets/estonia-the-digital-republic-secured-by-blockchain.pdf (Accessed: 13 November 2024).

Miller, M. (2018) West Virginia becomes first state to test mobile voting by blockchain in a federal election, Government Technology. Available at: https://www.govtech.com/biz/west-virginia-becomes-first-state-to-test-mobile-voting-by-blockchain-in-a-federal-election.html (Accessed: 13 November 2024).

Maxie, E. (2018). *Pros and Cons of Different Blockchain Consensus Protocols*. [online] Very. Available at: https://www.verytechnology.com/iot-insights/pros-and-cons-of-different-blockchain-consensus-protocols.

Fazekas, L. (2022). *How to create an anonymous, unhackable voting system on the blockchain*. [online] Geek Culture. Available at: https://medium.com/geekculture/how-to-create-an-anonymous-unhackable-voting-system-on-the-blockchain-ff6f932727b8.

Jafar, U., Aziz, M.J.A. and Shukur, Z. (2021). Blockchain for Electronic Voting System—Review and Open Research Challenges. *Sensors*, 21(17), p.5874. doi:https://doi.org/10.3390/s21175874.

Sedlmeir, J., Buhl, H.U., Fridgen, G. and Keller, R. (2020). The Energy Consumption of Blockchain Technology: Beyond Myth. *Business & Information Systems Engineering*, [online] 62(6), pp.599–608. doi:https://doi.org/10.1007/s12599-020-00656-x.

UK Data Service (2021). *The Data Protection Act and GDPR*. [online] UK Data Service. Available at: https://ukdataservice.ac.uk/learning-hub/research-data-management/data-protection/data-protection-legislation/data-protection-act-and-gdpr/.

Product Plan (2024). What Is MoSCoW Prioritization? | Overview of the MoSCoW Method. [online] Productplan.com. Available at: https://www.productplan.com/glossary/moscow-prioritization/.

Brush, K. and Silverthorne, V. (2022). *What is Agile Software Development (Agile Methodologies)?* [online] TechTarget. Available at: https://www.techtarget.com/searchsoftwarequality/definition/agile-software-development.

www.sciencedirect.com. (n.d.). *Verification Plan - an overview | ScienceDirect Topics*. [online] Available at: https://www.sciencedirect.com/topics/computer-science/verification-plan.

GeeksforGeeks. (2017). *Unit Testing in Python - Unittest*. [online] Available at: https://www.geeksforgeeks.org/unit-testing-python-unittest/.

www.fullstackpython.com. (n.d.). *Integration Testing*. [online] Available at: https://www.fullstackpython.com/integration-testing.html.

GeeksforGeeks (2019). *Acceptance testing | software testing - geeksforgeeks*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/acceptance-testing-software-testing/.