

Curtin University – Department of Computing

Assignment Cover Sheet / Declaration of Originality

Complete this form if/as directed by your unit coordinator, lecturer or the assignment specification.

Last name:	Maqsood	Student ID:	19970532
Other name(s):	Hamza		
Unit name:	Capstone Computing Project 2	Unit ID:	ISAD3001
Lecturer / unit coordinator:	Mr. Salih Ismail	Tutor:	N/A
Date of submission:	04 August 2022	Which assignment?	Final Submission (Leave blank if the unit has only one assignment.)

I declare that:

- The above information is complete and accurate.
- The work I am submitting is *entirely my own*, except where clearly indicated otherwise and correctly referenced.
- I have taken (and will continue to take) all reasonable steps to ensure my work is *not accessible* to any other students who may gain unfair advantage from it.
- I have *not previously submitted* this work for any other unit, whether at Curtin University or elsewhere, or for prior attempts at this unit, except where clearly indicated otherwise.

I understand that:

- Plagiarism and collusion are dishonest, and unfair to all other students.
- Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
- If I plagiarise or collude, I risk failing the unit with a grade of ANN ("Result Annulled due to Academic Misconduct"), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
- Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
- It is my responsibility to ensure that my submission is complete, correct and not corrupted.

Signature: _____ Hamza _____ Date of signature: _____ 04-08-2022

(By submitting this form, you indicate that you agree with all the above text.)

Curtin University – Department of Computing

Assignment Cover Sheet / Declaration of Originality

Complete this form if/as directed by your unit coordinator, lecturer or the assignment specification.

Last name:	Mehdi	Student ID:	20337270
Other name(s):	Hussain		
Unit name:	Capstone Computing Project 2	Unit ID:	ISAD3001
Lecturer / unit coordinator:	Mr. Salih Ismail	Tutor:	N/A
Date of submission:	04 August 2022	Which assignment?	Final Submission (Leave blank if the unit has only one assignment.)

I declare that:

- The above information is complete and accurate.
- The work I am submitting is *entirely my own*, except where clearly indicated otherwise and correctly referenced.
- I have taken (and will continue to take) all reasonable steps to ensure my work is *not accessible* to any other students who may gain unfair advantage from it.
- I have *not previously submitted* this work for any other unit, whether at Curtin University or elsewhere, or for prior attempts at this unit, except where clearly indicated otherwise.

I understand that:

- Plagiarism and collusion are dishonest, and unfair to all other students.
- Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
- If I plagiarise or collude, I risk failing the unit with a grade of ANN ("Result Annulled due to Academic Misconduct"), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
- Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
- It is my responsibility to ensure that my submission is complete, correct and not corrupted.

Signature: Hussain Mehdi Date of signature: 04-08-2022

(By submitting this form, you indicate that you agree with all the above text.)

Curtin University – Department of Computing

Assignment Cover Sheet / Declaration of Originality

Complete this form if/as directed by your unit coordinator, lecturer or the assignment specification.

Last name:	Sheikh	Student ID:	20203478
Other name(s):	Izaan Zahid		
Unit name:	Capstone Computing Project 2	Unit ID:	ISAD3001
Lecturer / unit coordinator:	Mr. Salih Ismail	Tutor:	N/A
Date of submission:	04 August 2022	Which assignment?	Final Submission (Leave blank if the unit has only one assignment.)

I declare that:

- The above information is complete and accurate.
- The work I am submitting is *entirely my own*, except where clearly indicated otherwise and correctly referenced.
- I have taken (and will continue to take) all reasonable steps to ensure my work is *not accessible* to any other students who may gain unfair advantage from it.
- I have *not previously submitted* this work for any other unit, whether at Curtin University or elsewhere, or for prior attempts at this unit, except where clearly indicated otherwise.

I understand that:

- Plagiarism and collusion are dishonest, and unfair to all other students.
- Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
- If I plagiarise or collude, I risk failing the unit with a grade of ANN ("Result Annulled due to Academic Misconduct"), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
- Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
- It is my responsibility to ensure that my submission is complete, correct and not corrupted.

Signature: _____ Izaan _____ Date of signature: _____ 04-08-2022

(By submitting this form, you indicate that you agree with all the above text.)

Curtin University – Department of Computing

Assignment Cover Sheet / Declaration of Originality

Complete this form if/as directed by your unit coordinator, lecturer or the assignment specification.

Last name:	Dholakia	Student ID:	20397999
Other name(s):	Khushi Paresh		
Unit name:	Capstone Computing Project 2	Unit ID:	ISAD3001
Lecturer / unit coordinator:	Mr. Salih Ismail	Tutor:	N/A
Date of submission:	04 August 2022	Which assignment?	Final Submission (Leave blank if the unit has only one assignment.)

I declare that:

- The above information is complete and accurate.
- The work I am submitting is *entirely my own*, except where clearly indicated otherwise and correctly referenced.
- I have taken (and will continue to take) all reasonable steps to ensure my work is *not accessible* to any other students who may gain unfair advantage from it.
- I have *not previously submitted* this work for any other unit, whether at Curtin University or elsewhere, or for prior attempts at this unit, except where clearly indicated otherwise.

I understand that:

- Plagiarism and collusion are dishonest, and unfair to all other students.
- Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
- If I plagiarise or collude, I risk failing the unit with a grade of ANN ("Result Annulled due to Academic Misconduct"), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
- Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
- It is my responsibility to ensure that my submission is complete, correct and not corrupted.

Signature: Khushi Dholakia Date of signature: 04-08-2022

(By submitting this form, you indicate that you agree with all the above text.)

Final Document

for

Blockchain-Based Polling System

Version 1.0 approved

Prepared by

Hamza Maqsood 19970532

Hussain Mehdi 20337270

Izaan Zahid Sheikh 20203478

Khushi Paresh Dholakia 20397999

Curtin University Dubai – Capstone 2

04/08/2022

Table of Contents

1.Document Scope	3
2.Progress & Product State	3
3.Branching Plan	4
4.Agile Process	4
5.Handover Meeting	5

1. Document Scope

This document outlines the overall progress of the team, the difficulties faced, the final state of the project and the agile methodology followed by the team over the course of three months (Capstone Computing Project 2).

2. Progress & Product State

We created our branching plan based on the priority of the functional and non-functional requirements mentioned in our SRS. After the submission of the branching plan, we started working on the frontend of our website. We did face some difficulties in developing the frontend since React (JavaScript) was a completely new framework/language for us. By the end of Sprint 1, we had developed the skeleton of our frontend.

During Sprint 2, we started with our backend (Nodejs) but soon ran into a challenge of connecting the backend of our website to our database (Mongo DB). We finally did manage to establish a connection between them by doing extensive research on the same. We also developed our Admin dashboard and additional admin functionalities such as view admin, create admin, view users, etc. We were also able to successfully develop our User Registration page, Login page, (including Json Web Token), Navbar, FAQ page, dashboard, user sidebar, footer, contact us page.

During Sprint 3, we coded the solidity contract, connected the contact us page to the backend, updated the admin dashboard with the analytics tab, implemented the admin login and registration functionalities, logout functionality has been coded, the upload document feature was added, the viewing and approval feature for the IDs uploaded was added to the admin dashboard.

During Sprint 4, we did face a challenge with deploying the solidity contract on the blockchain and connecting it to the frontend, which was then resolved by further research and testing. We also implemented the create poll page, explore, active and ended poll pages, forgot password page (which sends an OTP to the user on their registered email), the view voters, end poll (using an end date and a button) and report poll functionality, and the search poll feature. We also added notifications for any error messages, success messages or warnings to be displayed to the users to guide their user experience. By the end of Sprint 4 (Milestone 4) we were able to successfully complete our project and hand it over to our client (Mr. Salih Ismail).

3. Branching Plan

We followed the branching plan we submitted throughout the development of this project and were able to strictly follow it with almost no changes. We maintained one individual branch for each member per Sprint, this was to ensure an agile approach to development and commits. The commit naming convention was the pushers name followed by the features or pages they are pushing and a comment explaining it further. After commits, merging from individual branches to Sprint was only done after the commit was approved by 2 team members, Izaan and Hussain, and merging from a Sprint branch with the main branch was only done after review and approval of 2 more team members, Hamza and Khushi.

4. Agile Process

We followed agile process throughout the project which supported us to have a more productive work environment for the team with less stress. It helped us to have proper progress checks at the right interval and better communication and collaboration which were improved as the sprints went on. It helped us create proper team protocols to track the project progress as well as the coding standards. The following protocols were used to keep track of our Agile process and progress throughout the project duration:

Meetings: We planned on having four sprints (one sprint per milestone) to work on the project. Each sprint was 3 weeks long and we stuck with the 3-week deadlines throughout the development phase. We conducted regular meetings in each sprint to make sure everyone was on track. We conducted an initial meeting at the start of each sprint where we used to plan and divide the sprint among the team members along with a deadline to when the particular task will be completed of each task. Then, we had a meeting where we did a progress check on each other's work and helped each other if anyone was stuck. The meeting timing were adjusted according to everyone's schedule which was a bit of complication. We had a final meeting near the end of each sprint where we finalized the work and pushed the files to the BitBucket repository.

BitBucket: For each sprint, each team member made their own branch where they pushed their relevant files on that milestone. The branches were named after the developer and the summary of the features that were added in that branch. These individual branches were

then pushed to the sprint branch at the end of the sprint. Before merging, the reviewer of respective branches reviewed the work that was being pushed and made sure no conflicts or errors are made.

The finalized bitbucket repository link can be found [here](#).

Teamwork: We made sure that the timing of each meeting was at appropriate time to everyone. At the start of the semester, one of the team members was outside the country so every member made sure to adjust their timings according to match the time zones. To collaborate better on the project, we used a feature called “Live Share” in Visual Studio Code (VSCode) which allowed us to work in the same directory with the same files together and see the changes made in real time inside the shared server. This made collaboration a lot easier and prevented bugs/code collision. The tasks were divided with what the member was comfortable with, and any harder tasks were divided in team or as a whole. All the documents were completed with live collaboration using Microsoft Word sharing feature. This helped us to keep a live track of what the progress of each member was or where they need help.

Toggl: We used Toggl to track the time management of each feature. It helped us track our progress at the end of each milestone as well as helped us in setting deadlines. We added all of the timing to a group project so we could extract a group report. The group report has been attached in the end of the document.

5. Handover Meeting

A meeting with the client was conducted, and an in-depth demo of the website’s functionality was given. During the meeting, the client’s concerns, and suggestions were noted down and all of them were resolved by the end of Sprint 4. After the meeting, a document containing a checklist of all the Functional/Non-Functional requirements and a Software Manual containing a step-by-step guide was compiled. Additionally, a document containing a list of all the changes asked by the client along with screenshots was made and all of these were submitted to the client. Lastly, the Bitbucket repository was finalized, and the files were downloaded for testing.

Software Requirements Specification

for

Blockchain-Based Polling System

Version 2.0 approved

Prepared by

Hamza Maqsood 19970532

Hussain Mehdi 20337270

Izaan Zahid Sheikh 20203478

Khushi Paresh Dholakia 20397999

Curtin University Dubai – Capstone 1

04/08/2022

Table of Contents

1. Introduction	3
1.1 Purpose	3
1.2 Document Conventions	3
1.3 Intended Audience and Reading Suggestions	3
1.4 Project Scope	4
1.5 References	4
1.6 Version History	5
2. Overall Description	6
2.1 Product Perspective	6
2.2 Product Features	6
2.3 User Classes and Characteristics	7
2.4 Operating Environment	8
2.5 Design and Implementation Constraints	8
2.6 User Documentation	8
2.7 Assumptions and Dependencies	8
3. System Features	9
3.1 Account Creation	9
3.2 Normal User Login	10
3.3 Admin User Login	10
3.4 Forgot Password	11
3.5 Create a Poll	12
3.6 Vote on Poll	13
3.7 View Polls	13
3.8 Report Poll	14
3.9 Approve Document	15
4. External Interface Requirement	15
4.1 User Interfaces	15
4.2 Hardware Interface	17
4.3 Software Interface	17
4.4 Communication Interface	17
5. Other Nonfunctional Requirement	18
5.1 Performance Requirements	18
5.2 Safety Requirements	18
5.3 Security Requirements	18
5.4 Software Quality Attributes	19
6. Other Requirements	20
6.1 User Agreement	20
Appendix A: Glossary	20
Appendix B: Analysis Models	21

1. Introduction

1.1 Purpose

The SRS describes a blockchain based website that allows users to create, manage and vote on polls. The primary objective of the Blockchain-Based Polling System is to provide secure and anonymous polling. Poll creators can create polls based on their requirements and receive votes seamlessly. The SRS details all the system features, interface requirements and nonfunctional requirements required by the website.

1.2 Document Conventions

There are no special fonts or highlighting used in our document. Every requirement statement whether functional or non-functional has its own priorities.

Font	Main Heading	Subheading	Body font size	Line spacing
Times New Roman	18	14	12	1.5

1.3 Intended Audience and Reading Suggestions

The intended audience for this document is developers, client, users and testers.

- Developers: The developer must consult all the section to get an insight about the project and update the requirement in the document as needed.
- Client: The client must consult all the section to get an insight about the completion of the project.
- Users: The users can go over the section 1, 2 and 3 so that they can use the website properly.
- Testers: The testers should go over the section 1, 2, 3 and 4 to check if everything has been implement properly as required and to debug the functions.

The document should be read starting with the introduction followed by its purpose and description. No section should be skipped as it provides you the necessary information about the project.

1.4 Project Scope

A Blockchain-Based Polling System is a system that allows the users to create, view and vote on polls with complete anonymity.

There are two primary users:

- Users: Poll creators, voters
- Admins

The system will allow users to accomplish the following goals:

- A decentralized system that allows voters to cast votes anonymously
- The users can create an account which is then verified by the administrator
- Allow the users to create and vote on polls after authentication
- Allow the users start creating polls using the create poll form
- The system stores the user ID, vote and polls on the blockchain to maintain their integrity
- Allow the admins to review the reported polls and act accordingly

The scope of this project is as follows:

- Allow users to anonymously create and vote on polls of any magnitude with complete transparency
- Enhance security using blockchain technology

1.5 References

- MongoDB Documentation: <https://www.mongodb.com/>
- React Documentation: <https://reactjs.org/>
- Node JS Documentation: <https://nodejs.org/en/docs/>

1.6 Version History

Change	Date	Version
Added blockchain information to the project description, updated the frontend decision, updated the backend decision, updated the database decision, removed the payment gateway	22 nd - February - 2022	1.1
Added meta mask functionality, discussed the advantages of using a blockchain in a polling system	7 th - March - 2022	1.2
Updated the assumptions and dependencies	5 th - April - 2022	1.3
Updated the Use-case Diagram, updated the development end for software interfaces	5 th - April - 2022	1.4
Updated the overall document details according to the developed project	4 th – August - 2022	2.0

2. Overall Description

2.1 Product Perspective

The Blockchain-based Polling System developed using React, MongoDB and Ethereum Blockchain is a new, self-contained product. The objective of this project is to enable the users to vote for polls as well as create their own polling booths where they can customize their polls. This system uses Blockchain technology to store the polling booth, the votes of the voters, the timestamp of the vote, and the voter's unique ID which is assigned to each user after they are verified.

The advantage of using blockchain technology for storing votes is: -

- Increase in security and transparency
- Traceability and authenticity of the votes
- Anonymity of voters
- Decentralized Structure (No centralized authority who has complete control over the blockchain)
- Automated Operations (Lesser chances of errors occurring)

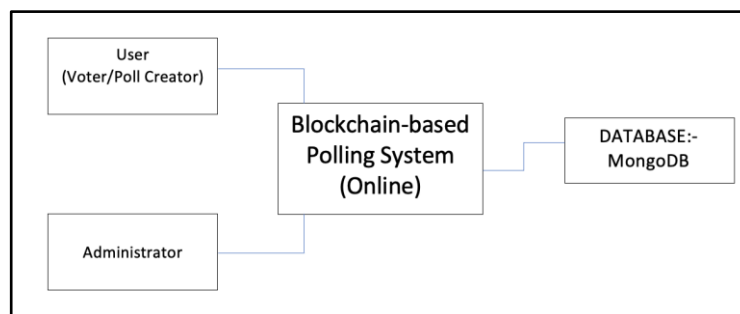


Figure 1: Major Components of the System

2.2 Product Features

This web-based system will enable registered users to create and vote for polls online. The users will be able to search for existing polls and vote in them. The votes casted by the users will be stored on the Ethereum blockchain to ensure their security and integrity. The users will also be able to view their poll history which will consist of all the polls they have voted for. Furthermore, users will be able to manage their profile and change their password using an email-based OTP system.

The poll creators will be able to check the authenticity of the votes casted. They will also be given the option to set an end date for their polls. The blockchain of choice is Ethereum Blockchain which is an open source and decentralized system which allows transparency of votes.

This system will also have a login portal for admins who will be responsible for validating the identification proof uploaded by the users during their registration. Validated users can use their Metamask Wallet to cast their votes, the votes will be stored on the basis of their Metamask wallet ID. Due to this, the identity of the voters will remain anonymous. The admin will also be in charge of moderating the reported polls.

2.3 User Classes and Characteristics

Voters

They will be able to:

- Create polls
- Filter for polls based on different categories such as Politics, Sports, Entertainment, etc or use the search bar to search for specific keywords
- Vote for existing polls
- View their vote history.
- Report Polls based on different categories

To be able to vote, the users need to upload images of their ID proof which must be verified by the admin. Additionally, voters must have a connected Metamask Wallet ID. Unverified users cannot vote for polls.

Poll Creators

They will be able to:

- Create polls
- Set an end date
- Add a description
- View the votes, voters' id with timestamp
- Select a specific category for their poll
- View their created polls and end them at any given time

To be able to create polls, the users need to upload images of their ID proof which must be verified by the admin. Unverified users cannot create polls.

Admin

They will be able to:

- View the user details
- Validate the user based on the identity proof provided.
- Can view reported polls and moderate accordingly
- Create and view new admins
- Receive messages from Contact Us Page
- View website analytics

2.4 Operating Environment

Browser: Chrome, Safari, Firefox

Operating System: Windows, Mac, Linux

Database: MongoDB

Platform: React/JavaScript

Technology: Ethereum Blockchain

2.5 Design and Implementation Constraints

- The verification of the user's ID proof will be done manually
- Only English language will be supported by the polling system
- The reported polls will be manually checked for offensive content

2.6 User Documentation

A FAQ page will also be available which will answer most of the commonly asked questions and will also provide details on the different aspects of creating a poll and voting for polls.

A Contact Us page will be provided on the website that allows users to send in their queries.

Users will have the facility to email the admin for queries.

2.7 Assumptions and Dependencies

Assumptions made are: -

- We assume that the users have a minimum internet connection bandwidth of 2 Mbps.
- We assume that the users have a basic knowledge of using a computer and the internet.
- One of the dependencies of our project is that the user should have their own MetaMask wallet prior to registering

3. System Features

3.1 Account Creation (ID: SF-1)

3.1.1 Description: Allowing the user to register and create an account on the website using a valid email and password. This is high priority as the user cannot create or vote on polls without creating an account first.

3.1.2 Stimulus/Response Sequences:

1. Users will visit the website
2. Click on create an account
3. Enter their email, username, first name & last name
4. Create Password
5. Re-enter Password

Errors:

- If username field is empty.
- If email field is empty.
- If email field matches an existing email.
- If password field is empty.
- If password and confirm password fields do not match

Error handling: In all these cases the program shall display an error message, such as “This” field cannot be empty.

3.1.3 Functional requirements **Dependencies**: This is not dependent on any other system requirement.

3.2 Normal User Login (ID: SF-2)

3.2.1 Description: This allows users to log in to their account in order to create or vote on polls. This has high priority as this is the only way the user can interact with the website.

3.2.2 Stimulus/Response Sequences:

1. User is prompted to enter their email and password
2. User is taken to the dashboard if login was successful

Errors:

- If email or password fields are empty.
- If email or password are incorrect

Error Handling: In both cases an appropriate error message will be displayed if the credentials are incorrect.

3.2.3 Functional requirement (Dependencies): SF-1.

3.3 Admin User Login (ID: SF-3)

3.3.1 Description: This allows users to log in with admin permissions to perform high level tasks such as document approval. This has high priority as it is the only way admin users can interact with the website.

3.3.2 Stimulus/Response Sequences:

1. Admin user is prompted to enter their email and password

Errors:

- If email and/or password are empty
- If email and password do not match

Error Handling: In all cases an appropriate error message will be displayed if the credentials are incorrect.

- 3.3.3 Functional Requirement (Dependencies): This has no requirements as admin accounts cannot be created by users, they are given by management to specific individuals.

3.4 Forgot Password (ID: SF-4)

- 3.4.1 Description: This allows users to change their account passwords if they have forgotten it and cannot log into their account. This is high priority as if the user cannot log into their account they cannot view or create polls.

- 3.4.2 Stimulus/Response Sequences:

1. User selects forgot password
2. User is prompted to enter their email
3. User is sent an OTP on their email
4. User is prompted to enter their OTP
5. User is prompted to enter their new password
6. User is prompted to confirm new password
7. User is notified of the password change

Errors:

- If email field is empty
- If email entered is not a valid entry
- If the OTP is invalid
- If the OTP field is empty
- If the new password field is empty
- If confirm password and password field do not match.

Errors handling: In all cases an appropriate error message will be displayed if the credentials are incorrect or if the passwords do not match each other

Functional Requirement (Dependencies): SF-1

3.5 Create a Poll (ID: SF-5)

3.5.1 Description: This allows the user to create a poll and post it on the website for users to vote on. This is high priority as it is a polling website.

3.5.2 Stimulus/Response Sequences:

1. User clicks on create poll
2. User enters title for the poll
3. User chooses a category for the poll
4. User enters short description for the poll
5. User enters the four options
6. User selects the end date
7. User clicks the Submit button

Errors:

- Poll title is empty
- The options are empty
- Category is empty
- End date is not selected
- Metamask wallet is not connected
- User is not verified
- User rejected transaction
- Lack of ether in account

Error handling: In each error case an error message will be displayed specifying the reason for the error to the user.

3.5.3 Functional requirement (Dependencies):

- SF-1
- SF-2

3.6 Vote on Poll (ID: SF-6)

3.6.1 Description: This allows users to vote on the polls that are posted on the website. This is also high priority as this is necessary for users to vote on the polls.

3.6.2 Stimulus/Response Sequences:

1. User visits the explore or active polls page
2. User can then select an option to vote on

Errors:

- User's internet times out before their option is submitted
- Metamask not connected
- User rejected transaction
- Lack of ether in account

Error Handling: An appropriate error notification will be displayed notifying that the vote has not been casted

3.6.3 Functional Requirement (Dependencies):

- SF-1
- SF-2
- SF-7

3.7 View Polls (ID: SF-7)

3.7.1 Description: This allows users to view the polls that are posted on the website, or the ones they voted on or created, either by searching by name or by category. This is also high priority as this is necessary for users to see the polls posted on the website.

3.7.2 Stimulus/Response Sequences:

1. User visits the explore, active or ended polls from the sidebar
2. Users can view their own polls by visiting my polls page from the sidebar
3. User can choose a category or search for a poll by name

Error:

- If the user leaves search and category field empty

Error Handling: It will continue with the search and all results will show.

2. Functional requirement (Dependencies):

- SF-1
- SF-2

b. Report Poll (ID: SF-8)

1. Description: This feature allows users to report polls that are offensive so that such polls can be taken down. This is low priority as it is not integral to other system features.

2. Stimulus/Response Sequences:

1. User chooses the report option on the poll
2. User chooses the reason from the dropdown menu provided
3. User clicks the Send button

Errors:

The user does not choose a reason from the dropdown

Error Handling: The report will not submit and display an error message that says an option was not chosen.

3. Functional requirement (Dependencies):

- SF-1
- SF-2
- SF-7

c. Approve Document (ID: SF-9)

1.Description: This will allow admin users to approve documents posted by normal users to get their accounts verified. This is high priority as it is necessary to get approval to create polls and vote.

2.Stimulus/Response Sequences:

1. Admin User visits the users page from the admin sidebar
2. Admin User is then presented a list of current users in tabular form along with their verification status and uploaded document
3. Admin User can then click on the document link to view the ID
4. Admin verifies the user based on the document uploaded

Error:

Admin's option to approve or reject is not uploaded due to connection timeout

Error Handling: An error message is displayed specifying the error due to timeout

3.Functional Requirements (Dependencies): SF-3

4. External Interface Requirements

4.1 User Interfaces

The platform is to be designed as a Web-Based Application; therefore, the GUI must be compatible with all devices. The website will be designed and built using React and relevant libraries will be used. The website will consist of multiple pages, and the interface will be different for each user. The website will have a landing page which will have two radio buttons for users to navigate to their relevant portals. Voters can select 'Sign up' and poll creators will have a button called 'Create Polls'. Both the buttons will lead to register and login pages respectively.

- Admin Interface:
 - Admin has access to all aspects of the application
 - The screen will display an admin login portal
- User Registration Interface:
 - This interface will allow the users (voters/ poll creators) to create an account. The "Sign up" button on the homepage will link to this portal. Users will have to enter:

- Username
 - Password
 - Valid Email ID
 - Full Name
- Login Interface:
 - This interface will allow users to login to an existing account. The “Create a poll now” button on homepage will link to this portal. Users will have to enter:
 - Email ID
 - Password
- Poll Creating Interface:
 - This interface will allow users to create polls according to their preferences. The user will select “Create Poll”. The user will enter the following details in the create poll form:
 - Poll Title
 - Category
 - Short Description
 - Selectable Options
 - End date
- Voting Interface:
 - This interface will allow users to explore live polls based on categories, or through direct search. They can then select a specific poll to vote on. Users will select one of the available choices and vote.

4.2 Hardware Interfaces

- Supported Device Types: Mobile phones, laptops, Personal Computers and tablets
- Database: MongoDB database version 8.0.28
- Screen Resolution: A Screen resolution of at least 800x600 is required for adequate functioning and display of the website across all platforms.
- Peripheral devices: Devices such as keyboards, mouse or touch screen are required in order to interact with the website.

4.3 Software Interfaces

- Client : Web Browser, Windows Series OS, React
- Database Server : MongoDB Atlas
- Development End : Ubuntu (Linux Distro), Node.js, JavaScript (Express.js, ethers.js, d3.js, chart.js, crypto.js, HTML, Solidity, Tailwind CSS, MongoDB
- Blockchain : Ethereum Blockchain, Metamask, Remix IDE, Hardhat

4.4 Communications Interfaces

- Client will be using HTTP/HTTPS protocol to access website
- Firewall is required to secure the web server
- TCP/IP protocol is used on client end
- JWT Token is used for generating a cookie upon user authentication
- All passwords stored on database will be encrypted using BCrypt library

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The system must show a meaningful error message when a problem occurs
- The system must show a notification when a vote is successfully casted
- The system must show all backend responses in a simple language in the front end without any delay

5.2 Safety Requirements

- Appropriate error messages should be displayed when an error is encountered, in order to guide the users to recover accordingly. The standard time for the system to recover from an error should be no longer than 20 seconds
- Additionally, all data input from the user's end must be validated. All users must be at least age 18 or older

5.3 Security Requirements

- The system must store the user ID, votes, timestamp and the polls on blockchain to maintain integrity
- The admin must authenticate the user by verifying the uploaded ID before the users can create or vote on polls
- The system must maintain the anonymity of the voters
- A Metamask wallet with a unique ID must be connected in order to vote on or create polls

5.4 Software Quality Attributes

- Adaptability/Compatibility:
 - The system must make sure that the website runs successfully on any platform or device

- Availability:
 - The system should be available for 99.9% of the time
- Correctness:
 - The system must prevent poll creators from exceeding word count
- Flexibility:
 - The system must allow the user to customize polls. They can select an **end date** and pick a **relevant category**
- Interoperability:
 - The system must work with the back-end database and blockchain to provide users with a good and safe experience
- Reliability:
 - The system must store the important information such as poll votes and **Metamask** ids on the blockchain which will help in preventing double or false votes and maintain the integrity of the polls
- Usability:
 - The system must provide the user with a **form** when creating a poll which will help them create a poll quickly
 - The system must provide appropriate instructions as well where needed
 - The system must show a meaningful error message when an error occurs

6. Other Requirements

6.1 User Agreement

The user must agree to the terms and conditions during the signup and must adhere to the privacy policy. The users should avoid illegal rules and protocols. Neither admin nor member should cross the rules and regulations of their country.

Appendix A: Glossary

Serial No.	Acronyms/Abbreviations	Description
1.	GUI	Graphical User Interface
2.	FAQ	Frequently Asked Questions
3.	Mbps	Megabits per second
4.	ID	Identity Document
5.	AI	Artificial Intelligence
6.	HTML	Hypertext Markup Language
7.	CSS	Cascading Style Sheets
8.	SF	System Features

Appendix B: Analysis Models

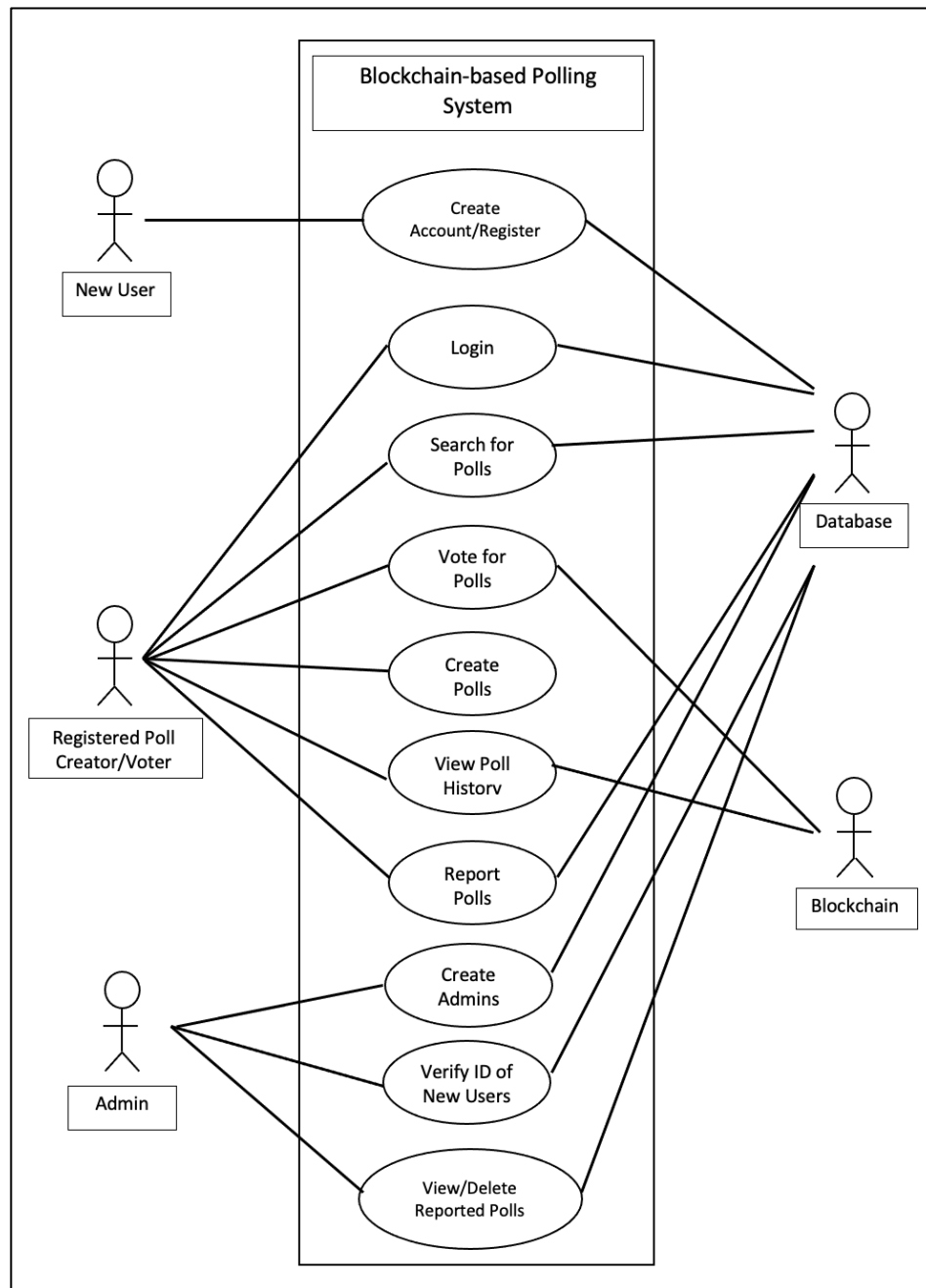


Figure 2: Blockchain-Based Polling System Use Case Diagram Updated

- New User: They can create an account/register on the website
- Registered User (Verified & Metamask Wallet Connected): They can be voters or poll creators. They have to login to continue. They can search, vote and create polls. If they

find any offensive polls, then they can report them

- Admin: They verify the ID proof uploaded by the new users while creating an account. They also decide if the reported polls have any offensive content in them. Additionally, the admin can create new admins and receives contact us messages
- Database: The Mongo DB database that stores the user's personal details
- Blockchain: The Ethereum Blockchain stores the poll creator and voter's wallet ID's, the votes casted by the user and the poll details

Branching Plan

for

Blockchain-Based Polling System

Version 1.0 approved

Prepared by

Hamza Maqsood 19970532

Hussain Mehdi 20337270

Izaan Zahid Sheikh 20203478

Khushi Paresh Dholakia 20397999

Curtin University Dubai – Capstone 2

26/05/2022

Table of Contents

1.Introduction	3
2.Branching Structure	3
2.1 Individual Branch	3
2.2 Sprint Branch	3
2.3 Develop Branch	4
2.4 Master Branch	4
3.Branching Standards & Procedure	4
3.1 Merges	4
3.2 Code review & Approval	5
3.3 Error Detection	5
3.4 Timelines	6
3.5 Commit Rules	6
3.6 Responsibilities	7

1. Introduction

This document provides the client information about the branching plans and strategies used and how the changes will be applied while developing the Blockchain Based Polling System.

2. Branching Structure

2.1 Individual Branch

Each member working on the project will have their own separate branch known as the individual branch which they will commit to. If a member needs to make changes to the project, then those change would be made on the individual branch instead of directly making them on the master branch. At the end of a sprint, once all the changes have been committed a pull request will be made that would have to be reviewed and accepted by two students. When accepted (i.e. there are no issues found), the individual branch will be merged into the sprint branch.

This branch's name convention will be: *<Name_Of_Member>_<Feature_edited>*

2.2 Sprint Branch

For each sprint, we will be creating a new branch called the sprint branch, that will be taken from the previous sprint. During each sprint, the changes, and updates we have made to the code will be pushed to that branch. At the end of each sprint, we will do a sprint review and test the features that have been implemented in that sprint before we push it on to the develop branch. This will help prevent complications and overwriting as each member will have their own copy of the branch that they can work on before submitting a merge request once they are done with their changes. The team members that will be reviewing the commits from the individual to sprint branch will be Hussain and Izaan.

When the changes have been made to the branch, before merging with the develop branch, the code is reviewed by a member or members of the team to ensure the code is up to standard and functions as intended. After the review, the sprint branch can be merged into the develop branch, then the sprint branch is deleted to prevent cluttering and confusion.

This branch's name convention will be: *Sprint_< Number >*

2.3 Develop Branch

This branch will serve as a connection point between sprint and master branches. The sprint branch will be integrated into the develop branch after all the features have been merged into the current active sprint branch. This will allow the reviewers to perform one more check on which task has been completed and which task is due for the next sprint before publishing the system's stable version. The team member that will be committing to this branch will be Hussain. There will be no review for this commit as it has already been reviewed in develop to master. This branch's name convention will be: *develop*

2.4 Master Branch

The project contains one main Master Branch. All sprints will move from the sprint branch to the develop branch. After the Develop Branch has been approved, the branch will be pushed to the master branch. Initially, the master branch will only contain the file structure required by the end application. Developers will clone this branch and get a copy of the file structure. The Master Branch will be the latest stable version of the application. The team members that will be reviewing the commits from the develop to master branch will be Khushi and Hamza.

This branch's name convention will be: *master*

3. Branching Standards & Pro

3.1 Merges

When a merge needs to be made to a branch, it will have to go through specific steps and procedures put in place to avoid merging of incomplete or erroneous changes. These steps are:

1. A pull request will be made to review the changes that are going to be merged.
2. The pull request will be reviewed by two reviewers.
3. The reviewers will have to give their approval of the merge request to the repository manager before merging with the branch.
4. A final approval meeting will be held, with all team members present, to review the code before merging with the master branch.

3.2 Code Review & Approval

Before any feature branches are merged into the sprint branch, a thorough code review will be completed. All types of pull requests will be communicated to the whole team and any member of the team may review it. Additionally, before a member makes any modifications to the code, the developer must consult with the rest of the team. It is also important that the developer will not be part of the review team for their own code in order to maintain quality. After the review, if any modifications are done, the push request must be examined and approved by at least two reviewers. Small modification to the code such as adding, deleting, formatting, comments and so on changes can be assessed by one of the reviewers.

Following the review and approval of push requests, a weekly meeting is arranged to discuss the final modifications, after which the repository manager merges the develop branch into the master branch. Any merge conflicts must be resolved by the repository manager and the developer in the meeting itself. In case if a reviewer takes too much time to review the code, the developer is allowed to approve the code themselves if the approval is obstructing their progress.

The reviewers for:

- Individual to Sprint Branch: Hussain and Izaan
- Develop to Master: Khushi and Hamza

3.3 Error Detection

Error detection is an integral part of the branching plan. The primary goal of error detection is to mitigate any merge conflicts or vulnerable/unstable code from making it to the master branch.

Any code added to a branch by the developers will have to go through a series of general checks. After which, it must be compiled without any errors or major warnings. At the initial stage minor warnings can be ignored however, before the merge into the Master Branch, all warnings should be rectified.

Additionally, once the code is running, test cases must be run in order to point out any issues in the functionality of the code. Primarily white-box testing must be used.

3.4 Timelines

Code Review Timeline

The reviewers will be allotted a time of 2 days (maximum) to review the code to be branched to the Master Branch. If they exceed the 2 day timeframe then the student who created the code to be merged will be allowed to approve the code.

Merge Timeline

Individual branches can be created any time and do not have a set date to be pushed to the repository. The Sprint branch, however, will be merged a day before the last day of submission and the develop branch will be merged on the last day of submission.

3.5 Commit Rules

Commits are used to push changes to each branch. These need to be maintained by each team member. Commits should be done after small variations and in high frequency so in case of any errors, the program can be reverted to a few instances before the error rather than starting the entire feature.

The commits must include short meaningful descriptions detailing the features and changes added in the specific commit. The following rules must be followed while writing commit messages:

- The Commit must include a Subject Line and body description separated by a space
- The Subject will be limited to 60 characters
- The subject line must be kept brief and include the task number and name of feature
- The body of the commit will be limited to 80 characters long. It will be used to explain the details of each commit. However, will be kept short and to the point.
- People in charge of:
 - Commit to Master Branch – Khushi Dholakia
 - Commit to Develop Branch – Hussain Mehdi
 - Commit to Sprint Branch – Izaan Sheikh
 - Commit to Individual Branch – Respective team member

3.6 Responsibilities

We plan to work as a team rather than just making that particular section responsibility of one person instead. We have divided the project into the following categories among ourselves. The distribution of responsibilities is as follows:

Front End – Khushi Dholakia, Izaan Sheikh

Back End – Hussain Mehdi, Hamza Maqsood

Middleware – Hamza Maqsood

Blockchain – Khushi Dholakia, Hussain Mehdi

Database – Izaan Sheikh

Software Manual

for

Blockchain-Based Polling System

Version 1.0 approved

Prepared by

Hamza Maqsood 19970532

Hussain Mehdi 20337270

Izaan Zahid Sheikh 20203478

Khushi Paresh Dholakia 20397999

Curtin University Dubai – Capstone 2

02/08/2022

Table of Contents

1.Introduction	3
2.Document Scope	3
3.Document Conventions	3
4.Technology Stack (MERN)	3
4.1 MongoDB	3
4.2 Express.js	3
4.3 React	3
4.4 Node.js	3
4.5 Ether.js	4
4.6 Contract Testing (.sol)	4
4.7 Deployment Environment Testing	4
5.Pre-requisite	4
6.Software Requirement	4
7.Application Setup	4
8.Features	7
8.1 User Registration	7
8.2 Normal User Login	7
8.3 Admin User Login	7
8.4 Admin User Registration	7
8.5 View Admin User	7
8.6 Forgot Password	7
8.7 Create a Poll	8
8.8 Vote on Poll	8
8.9 My Polls	8
8.10 View Voters	8
8.11 View Polls	9
8.12 Report Poll	9
8.13 Approve Document	9

1. Introduction

Blockchain based polling system is a Decentralized Application (DApp) developed to allow users to create and vote on polls with complete anonymity and transparency. This file contains the detailed steps on how to deploy and use the web-application. The link to the Bitbucket repository can be found [here](#).

2. Document Scope

The scope of this document includes a step-by-step guide on how to set up and use the Blockchain Based Polling System. The access of this document is directed towards the client, admins and testers who will be using this website. The features pertaining to the website admins have a role header that says admin (Eg: AdminLogin).

3. Document Conventions

Font	Heading	Sub-heading	Body font size	Line spacing
Times New Roman	16	14	12	1.5

4. Technology Stack (MERN)

4.1. MongoDB

MongoDB is an open-source document-oriented database system. It is being used to store user information, admin information, contact messages and reports.

4.2. Express.js

Express.js is a web application framework for Node.js which acts as a backend. It is being used to connect frontend with database.

4.3. React

ReactJS is a free and open-source JavaScript front-end library for creating user interfaces with UI components. We have used react as our frontend.

4.4. Node.js

Node.js is an open-source JavaScript runtime environment for backend.

4.5. Ethers.js

Ethers.js is a library used to interact with the Ethereum Blockchain and its ecosystem.

4.6. Contract Testing (.sol)

The Solidity contract was tested on Remix and compiled/ deployed using Hardhat.

Local Hardhat testing was also done prior to deployment.

4.7. Deployment Environment Testing

The website was deployed to a virtual machine and tested extensively using both Windows and Linux VMs.

5. Pre-requisites

- A working PC with internet connection
- npm package manager should be installed

```
npm install npm@latest
```

6. Software Requirements

Operating System: Windows, Linux

7. Application Setup

- Cloning the Repository

```
git clone https://bitbucket.org/blockchain-based-polling-system/bbps.git
```

```
(kali㉿kali)-[~/Desktop]
$ git clone https://bitbucket.org/blockchain-based-polling-system/bbps.git
Cloning into 'bbps' ...
Receiving objects: 100% (513/513), 15.66 MiB | 9.98 MiB/s, done.
Resolving deltas: 100% (242/242), done.
```

• Installing the dependencies

- Client (bbps/client)

```
npm install --legacy-peer-deps
```

```
(kali@kali) [~/Desktop/bbps/client]
$ npm install --legacy-peer-deps
npm WARN deprecated stable@0.1.8: Modern JS already guarantees Array#sort() is a stable sort, so this library is deprecated. See the compatibility table on MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort#browser_compatibility
npm WARN deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated
npm WARN deprecated mkdirp-promises@0.1: This package is broken and no longer maintained. 'mkdirp' itself supports promises now, please switch to that.
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated source-map-resolve@0.6.0: See https://github.com/lydell/source-map-resolve#deprecated
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated source-map-resolve@0.5.3: See https://github.com/lydell/source-map-resolve#deprecated
npm WARN deprecated chokidar@2.1.8: chokidar 2 does not receive security updates since 2019. Upgrade to chokidar 3 with 15x fewer dependencies
npm WARN deprecated querystring@0.2.0: The querystring API is considered legacy. New code should use the URLSearchParams API instead.
npm WARN deprecated multicodec@1.0.4: This module has been superseded by the multiformats module
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated multibase@0.6.1: This module has been superseded by the multiformats module
npm WARN deprecated multibase@0.7.0: This module has been superseded by the multiformats module
npm WARN deprecated multicodec@0.7.2: This module has been superseded by the multiformats module
npm WARN deprecated svgo@1.3.2: This SVGO version is no longer supported. Upgrade to v2.x.x.
npm WARN deprecated c16@0.7.5: This module has been superseded by the multiformats module
npm WARN deprecated core-js@1.2.7: core-js@3.23.3 is no longer maintained and not recommended for usage due to the number of issues. Because of the V8 engine whims, feature detection in old core-js versions could cause a slowdown up to 100x even if nothing is polyfilled. Some versions have web compatibility issues. Please, upgrade your dependencies to the actual version of core-js.

added 2844 packages, and audited 2845 packages in 48s

258 packages are looking for funding
  run `npm fund` for details

37 vulnerabilities (10 moderate, 7 high)

To address all issues possible (including breaking changes), run:
  npm audit fix --force

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.
```

- Server (bbps/server)

```
npm install
```

```
(kali@kali) [~/Desktop/bbps/server]
$ npm install
npm WARN old lockfile
npm WARN old lockfile The package-lock.json file was created with an old version of npm,
npm WARN old lockfile so supplemental metadata must be fetched from the registry.
npm WARN old lockfile
npm WARN old lockfile This is a one-time fix-up, please be patient...
npm WARN old lockfile

added 133 packages, and audited 134 packages in 12s

11 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

- **Running the servers**

- Frontend

```
npm start
```

- Backend

```
nodemon app.js
```

```
(kali@kali)-[~/Desktop/bbps/server]
$ nodemon app.js
[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
server is running at port no 3001
connection success
```

8. Features

8.1 User Registration

Instructions: Users can register on the website by navigating to the Register page using the “Sign Up Now!” button visible on the landing page. The “Create new Account” form is displayed, and user must fill in all the details. Once the details are filled, and the “Sign up” button is clicked, the request will be processed, and user will be added to the backend (MongoDB Database).

8.2 Normal User Login

Instructions: Normal Users can login via the “Create your own poll now” button displayed on the landing page in addition to navigating from the register page. Once on the login page, the users are asked to enter an Email ID and password which were entered during user registration. The form will query the backend (MongoDB Database) to check whether the user details are valid. Once the user clicks “Login”, the user will be redirected to the user dashboard.

8.3 Admin User Login

Instructions: Admin users can login via the “Staff Portal” button placed in the footer of the website. The button navigates the user to the Admin Portal, where they are asked to enter a valid username and password. Once the admin user enters their credentials, the database is queried to check validity of credentials and the user is redirected to the Admin Dashboard.

8.4 Admin User Registration

Instructions: Admin users can register additional admins using the “Manage Admins” tab on the Admin Dashboard. Once the admin is logged in, they can click on “Create” on the admin sidebar and navigate to the “Register a new Admin” Form. They can add the relevant details and click register.

8.5 View Admin Users

Instructions: Admins can view the list of registered admins under the “View” tab on the admin dashboard. The username, name and role are displayed in a table format.

8.6 Forgot Password

Instructions: Users can navigate to the forgot password page either from the login page (by clicking forgot password) or from the Profile page (by clicking change password). The user must enter their valid email ID and click Send OTP. The user email will be validated with the user database and will display an error message if the email is not valid or a success message if the email is valid and will then send an email to the user using nodemailer. The user will receive an email with the OTP on their valid email address. The user must then enter the OTP along with the new password click

reset password. The new user password will then be hashed and stored in the backend database (MongoDB)

8.7 Create a Poll

Instructions: Once the user is logged in, verified and has connected their Metamask Wallet, they can navigate to the create poll page from the sidebar to create a poll. The user can then enter the poll title, poll description and options and select the poll category and end date. Once all the fields are filled the user can click the Submit button which will process the request and pop up the Metamask account to confirm or reject the transaction. Once the user selects the confirm option a loader for poll is being created is displayed and once the transaction has been processed, a notification that says your poll has been created is displayed.

8.8 Vote on Poll

Instructions: Once the user is logged in, verified and has connected their Metamask Wallet, they can navigate to the explore or active poll page from the sidebar to view the polls. The user can then select one of the four options displayed on a particular poll which will first check if the user has voted before on this particular poll. If the user has previously voted, a notification saying You have already voted is displayed. If the user is voting for the first time, the Metamask transaction pops up with a reject or confirm option. Once the user selects the confirm option and the transaction has been processed, a notification that says your vote has been casted is displayed.

8.9 My Polls

Instructions: Once the user is logged in, verified and has connected their Metamask Wallet, they can navigate to the my polls page from the sidebar to view the polls that they have created. The user can also view the voters who have voted on their poll and their votes. The user can end the poll manually (before the poll's end date) by clicking the End Poll button for that poll. The Metamask transaction pops up with a reject or confirm option. Once the user selects the confirm option and the transaction has been processed, a notification that says poll has been ended is displayed.

8.10 View Voters

Instructions: Once the user is logged in, verified and has connected their Metamask Wallet, they can navigate to the explore or active poll page from the sidebar to view the polls. The user can then click on the view voters on a particular poll. The voter ID, option selected and the voting timestamp is displayed.

8.11 View Polls

Instructions: Once the user has logged in, they can navigate to the Explore section of the user side bar which will be present on the left side of their screen at all times. The explore page can be used to view all polls that are available on the website, they can see the voters of the poll, the end time and the number of votes on each option. There is also a search bar present on that page that can be used to search for key words that the user is looking for in a poll. The Explore section on the sidebar also has two subsections, Active and Ended Polls. These two pages have the polls divided according to their activeness, Active Polls has the polls that are still on going and can be participated in and Ended Polls has those polls that have their results finalized and cannot be participated in, the user can only view them. Both these pages also have search bars in them.

8.12 Report Poll

Instructions: Once the user is logged in, verified and has connected their Metamask Wallet, they can navigate to the explore page from the sidebar to view all the polls. The user can report a particular poll by clicking the Report Poll button displayed on that poll. Once selected, a dropdown will be displayed and the user can choose a reason for reporting the poll. Once selected, the user can then click the Send button and the reported poll along with the reason and user information is stored in the backend (MongoDB) and the administrator can go and review these polls when they log in and navigate to the Reported Poll section.

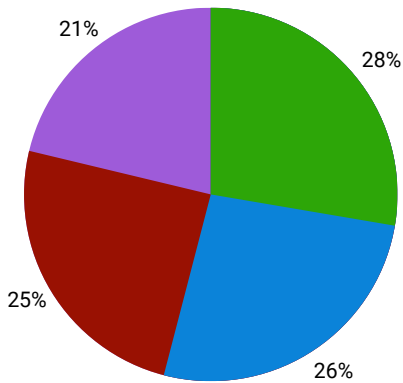
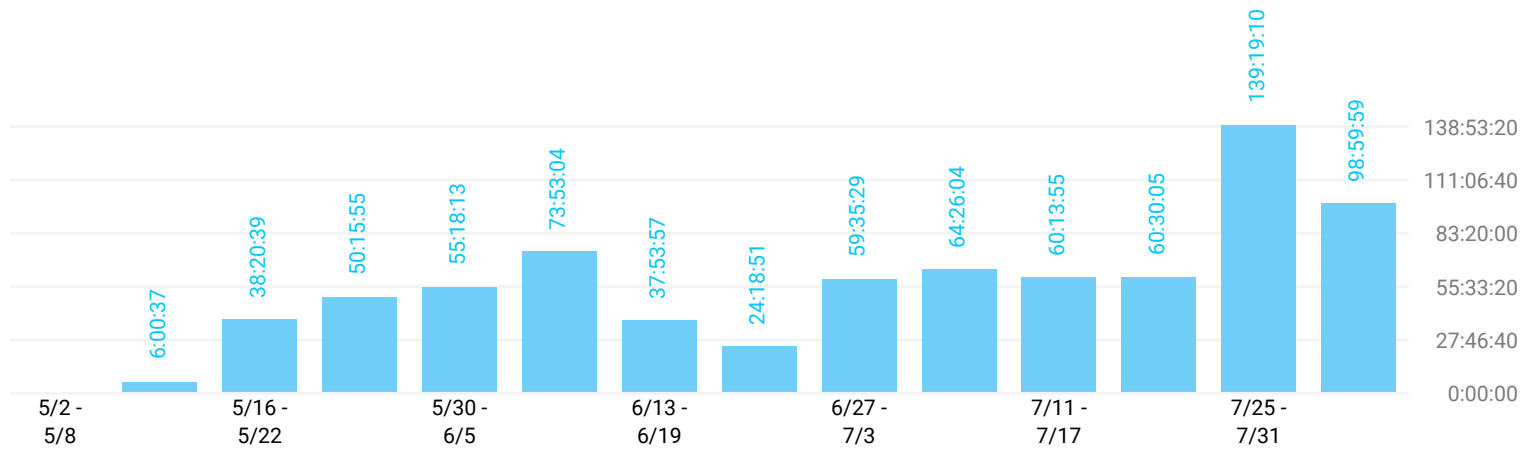
8.13 Approve Document

Instructions: Users have to provide some sort of identification to verify themselves. The document is uploaded to the backend (MongoDB) where it is reviewed by the administrator and verified accordingly. The user is displayed with proper messages as if the user has not uploaded any document, a message is displayed on the Dashboard displaying “You are not verified!! Please upload an ID on Profile page.” with profile page being hyperlinked. After the document has been uploading, the message is changed to “Verification Status: Pending”. After the verification, the message disappears. These relevant messages are also displayed to administrator on their portal where they can see who is waiting for authentication or has not uploaded the document.

Summary Report

05/02/2022 – 08/07/2022

TOTAL HOURS: 769:05:58

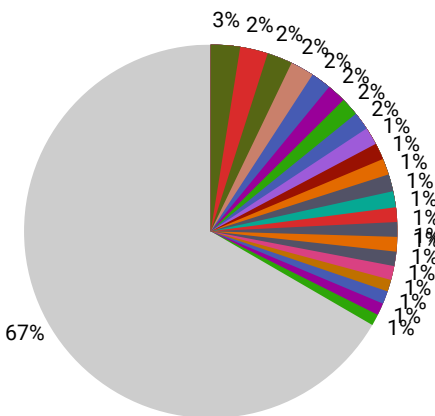


USER

- HM Hussain Mehdi
- KD Khushi Dholakia
- IS Izaan Sheikh
- HM Hamza Maqsood

DURATION

- 212:49:03
- 202:31:24
- 190:18:36
- 163:26:55



TIME ENTRY

- Milestone 5: Working on Presentation
- Handover (Meeting): Handover Discussion
- Milestone 4 (Meeting): Sprint 4 Discussion
- Milestone 3 (Meeting): Sprint 3 Discussion
- Milestone 4 (Meeting): Progress Check
- Milestone 5: Fixing Final Repo Bugs
- Milestone 4: Editing solidity, implementing vote, view votes, end poll, view my polls
- Milestone 3 (Meeting): Sprint 3 Research
- Milestone 4: Fixing Blockchain Connectivity
- Handover (Meeting): Handover Finalization
- Milestone 5: Working on the presentation, and making notes for presentation
- Milestone 2 (Meeting): Finalizing Sprint 2
- Milestone 2: Navbar, Page Linking & CSS
- Milestone 4: Storing wallet ID in database and Report Poll functionality
- Milestone 2 : Adding the Backend, connecting mongodb and Register page revamp

DURATION

- 19:53:32
- 18:27:08
- 16:17:05
- 15:44:37
- 13:51:55
- 12:46:27
- 12:10:15
- 11:51:10
- 11:29:12
- 11:13:45
- 11:01:52
- 10:47:49
- 10:27:00
- 10:12:34
- 10:04:00

● Milestone 4: Adding notifications, enhancing UI, fixing blockchain bugs and assisting with backend functionality	9:34:00
● Milestone 4: Displaying meta text, adding notifications, assisting with SOL	9:33:09
● Milestone 2: Optimizing css/grid issue and making a drop down menu	8:42:34
● Milestone 2: Working on Admin Dashboard	8:42:22
● Milestone 2 (Meeting): Sprint 2 Research	8:18:46
● Milestone 4: Bug fixing and minor updates	7:52:20
● Milestone 4: Implementing forgot password and otp, Editing the milestone 4 document	7:44:28
● Other time entries	512:19:58


USER - TIME ENTRY

	DURATION	PERCENTAGE
Hamza Maqsood	163:26:55	21.25%
Branching Plan: Merge	0:35:12	0.08%
Branching Plan: Sprint branch	1:09:48	0.15%
Handover (Meeting): Handover Discussion	4:01:15	0.52%
Handover (Meeting): Handover Finalizing	3:48:00	0.49%
Handover (Meeting): Working on Handover	1:30:29	0.2%
Handover: Working on the document	1:05:50	0.14%
Meeting with Client: Final Presentation	0:46:00	0.1%
Meeting: Branching Plan	1:04:06	0.14%
Meeting: Branching Plan Discussion	1:42:04	0.22%
Meeting: Branching Plan Finalizing	1:14:49	0.16%
Meeting: Branching Plan Review	0:56:01	0.12%

USER - TIME ENTRY	DURATION	PERCENTAGE
Milestone 1 (Meeting): BitBucket and Jira	0:45:53	0.1%
Milestone 1 (Meeting): Bug Fixing	0:42:40	0.09%
Milestone 1 (Meeting): Finalizing Sprint	3:56:01	0.51%
Milestone 1: Sprint 1	0:55:38	0.12%
MileStone 2 (Meeting): Finalizing Sprint 2	5:22:00	0.7%
Milestone 2 (Meeting): Sprint 2 Backend	0:37:00	0.08%
Milestone 2 (Meeting): Sprint 2 Discussion	1:34:00	0.2%
Milestone 2 (Meeting): Sprint 2 Research	1:28:00	0.19%
Milestone 2: Finalizing Individual Work	2:05:23	0.27%
Milestone 2: Making changes to Welcome Page	1:38:22	0.21%
MileStone 2: Sidebar Formatting	1:03:53	0.14%
Milestone 2: User Dashboard Avatar Formatting	1:30:15	0.2%
Milestone 2: User Dashboard Avatar Implementation	2:11:55	0.29%
Milestone 2: User Dashboard Avatar Research	2:21:15	0.31%
MileStone 2: User Dashboard Grid Alignment	2:16:33	0.3%
Milestone 2: User Dashboard icons	2:13:27	0.29%
Milestone 2: User Dashboard Layout	2:07:11	0.28%

USER - TIME ENTRY	DURATION	PERCENTAGE
Milestone 2: User Profile Page	1:06:00	0.14%
MileStone 2: User Sidebar Formatting	2:50:04	0.37%
Milestone 2: User Sidebar Implementation	2:39:01	0.34%
Milestone 2: User Sidebar Research	3:17:23	0.43%
Milestone 3 (Meeting): Sprint 3 Discussion	3:54:00	0.51%
Milestone 3 (Meeting): Sprint 3 Progress Check	0:46:04	0.1%
Milestone 3 (Meeting): Sprint 3 Research	3:55:00	0.51%
Milestone 3: Adding User information to Profile Page	1:53:10	0.25%
Milestone 3: Debugging the Upload Document Feature	2:42:00	0.35%
Milestone 3: Finalizing Upload Feature	4:42:09	0.61%
Milestone 3: Individual Research	2:24:30	0.31%
Milestone 3: Linking Upload Feature to Back end	1:42:10	0.22%
Milestone 3: Logout Function Creation	1:48:28	0.24%
Milestone 3: Logout Function Debugging	2:23:45	0.31%
Milestone 3: Logout Function Implementation	2:11:38	0.29%
Milestone 3: Logout Research	2:13:32	0.29%
Milestone 3: Profile Page New Format	2:03:02	0.27%


USER - TIME ENTRY	DURATION	PERCENTAGE
Milestone 3: Upload Document Implementation	2:12:47	0.29%
Milestone 3: Upload Document Page Creation	1:48:12	0.23%
Milestone 3: Upload Document Research	2:24:00	0.31%
Milestone 3: Upload Progress Page Creation	1:50:00	0.24%
Milestone 4 (Meeting): Meeting with client	2:12:24	0.29%
Milestone 4 (Meeting): Milestone 4(Meeting): Finalizing Sprint 4 and Document	2:08:48	0.28%
Milestone 4 (Meeting): Preparing for demo with client	1:46:00	0.23%
Milestone 4 (Meeting): Sprint 4 Discussion	3:39:58	0.48%
Milestone 4 (Meeting): Sprint 4 Progress Check	4:15:45	0.55%
Milestone 4: Active Page Research	2:53:00	0.37%
Milestone 4: Active Polls page Debugging	2:10:33	0.28%
Milestone 4: Active Polls page Implementation	2:41:29	0.35%
Milestone 4: Client Feedback Implementations	1:27:38	0.19%
Milestone 4: Ended Polls Implementations	3:25:42	0.45%
Milestone 4: Ended Polls Page Debugging	2:47:28	0.36%
Milestone 4: Ended Polls Research	2:48:10	0.36%
Milestone 4: Search Bar Debugging	3:06:15	0.4%

USER - TIME ENTRY	DURATION	PERCENTAGE
Milestone 4: Search Bar Implementation	5:08:35	0.67%
Milestone 4: Search Bar Research	2:05:09	0.27%
Milestone 4: Sprint 4 Research	2:39:20	0.35%
Milestone 5 (Meeting): Final Document Discussion	1:41:00	0.22%
Milestone 5 (Meeting): Updating SRS	1:41:49	0.22%
Milestone 5: Working on Final Submission	1:28:26	0.19%
Milestone 5: Working on Presentation	1:45:27	0.23%
Milestone1 (Meeting): Finalizing Milestone 1 Document and Code	3:27:00	0.45%
Milestone1: Finalizing Home Page	1:52:00	0.24%
Milestone1: Home Page	2:01:19	0.26%
Milestone1: Learning React	2:43:45	0.35%
 Hussain Mehdi	212:49:03	27.67%
Branching Plan: Structure and Procedure	4:31:00	0.59%
Handover (Meeting): Handover Discussion	4:46:00	0.62%
Handover (Meeting): Handover Finalization	3:45:00	0.49%
Handover (Meeting): Working on Handover	1:44:00	0.23%
Handover: Docker Research	1:08:00	0.15%

USER - TIME ENTRY	DURATION	PERCENTAGE
Handover: Editing the document	1:33:00	0.2%
Meeting with Client: Demo	0:38:49	0.08%
Meeting with Client: Final Presentation	0:51:00	0.11%
Meeting: Branching Plan	1:01:00	0.13%
Meeting: Branching Plan Discussion	1:42:38	0.22%
Meeting: Branching Plan Finalizing	1:15:40	0.16%
Meeting: Branching Plan Review	0:57:54	0.13%
Milestone 1 (Meeting): Bitbucket & Jira	0:44:38	0.1%
Milestone 1 (Meeting): Bug Fixing	0:43:00	0.09%
Milestone 1 (Meeting): Finalising Milestone 1 document and Code	3:44:00	0.49%
Milestone 1: Building File System & Template	1:34:49	0.21%
Milestone 1: Editing Navbar	2:02:47	0.27%
Milestone 1: Refining File Structure	1:48:00	0.23%
Milestone 1: Researching React	3:47:00	0.49%
Milestone 2 (Meeting): Sprint 2 Discussion	1:35:16	0.21%
Milestone 2 (Meeting): Finalizing Sprint 2	4:50:31	0.63%
Milestone 2 (Meeting): Sprint 2 Backend	0:33:07	0.07%

USER - TIME ENTRY	DURATION	PERCENTAGE
Milestone 2 (Meeting): Sprint 2 Research	3:55:00	0.51%
Milestone 2: Building Admin Dashboard	6:01:59	0.78%
Milestone 2: Editing pages and CSS	3:20:12	0.43%
Milestone 2: Importing User Data from Database into Admin Dashbaord	2:50:03	0.37%
Milestone 2: Navbar, Page Linking & CSS	10:27:00	1.36%
Milestone 2: Optimizing Admin Dashboard	0:37:45	0.08%
Milestone 2: Working on Admin Dashboard	8:42:22	1.13%
Milestone 3 (Meeting): Finalizing Sprint 3 and Document	2:23:00	0.31%
Milestone 3 (Meeting): Finalizing Sprint 3	1:22:26	0.18%
Milestone 3 (Meeting): Sprint 3 Discussion	3:58:18	0.52%
Milestone 3 (Meeting): Sprint 3 Progress Check	0:54:00	0.12%
Milestone 3 (Meeting): Sprint 3 Research	4:01:02	0.52%
Milestone 3: Adding content to Welcome page	3:50:29	0.5%
Milestone 3: Fixing admin dashboard and login	4:24:47	0.57%
Milestone 3: Fixing Blockchain bugs	4:54:00	0.64%
Milestone 3: Fixing Pages bugs	4:36:28	0.6%
Milestone 3: Implementing Blockchain	7:10:41	0.93%


USER - TIME ENTRY	DURATION	PERCENTAGE
Milestone 3: Sprint 3 Working on Admin Login	3:41:00	0.48%
Milestone 4 (Meeting): Finalizing Sprint 4 and Document	2:00:00	0.26%
Milestone 4 (Meeting): Preparing for demo with client	2:00:00	0.26%
Milestone 4 (Meeting): Progress Check	4:33:00	0.59%
Milestone 4 (Meeting): Sprint 4 Discussion	4:31:03	0.59%
Milestone 4: Adding BrowsePolls page and getting data from Blockchain	3:09:29	0.41%
Milestone 4: Adding notifications, enhancing UI, fixing blockchain bugs and assisting with backend functionality	9:34:00	1.24%
Milestone 4: Adding UI/UX Features and removing errors from console	1:26:37	0.19%
Milestone 4: Deploying onto Blockchain	4:05:08	0.53%
Milestone 4: Displaying meta text, adding notifications, assisting with SOL	9:33:09	1.24%
Milestone 4: Editing Single Poll page and displaying content from blockchain	1:20:27	0.17%
Milestone 4: Fixing Blockchain Connectivity	11:29:12	1.49%
Milestone 4: Fixing up loader	0:39:17	0.09%
Milestone 4: Fixing vote on poll functionality	5:19:32	0.69%
Milestone 4: Sprint 4 Research	1:41:10	0.22%
Milestone 4: Updating SOL contract and blockchain functionality	2:44:00	0.36%
Milestone 4: Working on creating new page with blockchain hash	3:54:47	0.51%

USER - TIME ENTRY	DURATION	PERCENTAGE
Milestone 4: Working on displaying blockchain data	6:46:35	0.88%
Milestone 5 (Meeting): Final Submission Discussion	1:36:00	0.21%
Milestone 5 (Meeting): Updating the SRS	2:01:00	0.26%
Milestone 5: Working on Final Submission	1:59:28	0.26%
Milestone 5: Working on Presentation	7:41:28	1.0%
Milestone1: Building Navbar and Fixing Bitbucket ssh	2:16:00	0.29%
 Izaan Sheikh	190:18:36	24.74%
Branching Plan: Code Review & Approval	0:49:58	0.11%
Branching Plan: Develop Branch	0:57:32	0.12%
Branching Plan: Responsibilities	0:45:27	0.1%
Handover (Meeting): Handover Discussion	4:51:34	0.63%
Handover (Meeting): Handover Finalization	3:42:53	0.48%
Handover (Meeting): Working on Handover	1:43:47	0.22%
Handover: Working on Handover	0:57:32	0.12%
Meeting with Client: Demo	0:38:25	0.08%
Meeting with Client: Final Presentation	0:52:13	0.11%
Meeting: Branching Plan	1:05:40	0.14%

USER - TIME ENTRY	DURATION	PERCENTAGE
Meeting: Branching Plan Discussion	1:44:24	0.23%
Meeting: Branching Plan Finalizing	1:17:05	0.17%
Meeting: Branching Plan Review	0:56:26	0.12%
Milestone 1 (Meeting): BitBucket and Jira	0:44:26	0.1%
Milestone 1 (Meeting): Bug Fixing	0:42:34	0.09%
Milestone 1 (Meeting): Finalizing document and Code	3:32:20	0.46%
Milestone 1: Forgot Password Page	0:59:24	0.13%
Milestone 1: Learning React	3:56:43	0.51%
Milestone 1: Learning to merge the js	2:34:58	0.34%
Milestone 1: Login Page	2:28:45	0.32%
Milestone 1: Registration Page	1:44:01	0.23%
Milestone 2 (Meeting): Finalizing Sprint 2	5:57:18	0.77%
Milestone 2 (Meeting): Sprint 2 Backend	0:36:45	0.08%
Milestone 2 (Meeting): Sprint 2 Discussion	1:34:40	0.21%
Milestone 2 (Meeting): Sprint 2 Research	1:27:46	0.19%
Milestone 2: Database	2:06:46	0.27%
Milestone 2: Database connectivity research	4:52:18	0.63%

USER - TIME ENTRY	DURATION	PERCENTAGE
Milestone 2: Footer and Login Page Integrated	1:40:31	0.22%
Milestone 2: Footer, Buttons, Linking	6:36:10	0.86%
Milestone 2: Optimizing css/grid issue	1:33:35	0.2%
Milestone 2: Optimizing css/grid issue and making a drop down menu	8:42:34	1.13%
Milestone 2: Registration Page CSS	0:37:50	0.08%
Milestone 2: Registration Page Integrated	1:16:22	0.17%
Milestone 2: Troubleshooting backend and Hashing Password	2:46:24	0.36%
Milestone 3 (Meeting): Finalizing Sprint 3	1:21:00	0.18%
Milestone 3 (Meeting): Finalizing Sprint 3 and Document	2:21:39	0.31%
Milestone 3 (Meeting): Progress Check	0:53:58	0.12%
Milestone 3 (Meeting): Sprint 3 Discussion	3:58:41	0.52%
Milestone 3 (Meeting): Sprint 3 Research	3:55:08	0.51%
Milestone 3 : Fixing Backend and other Bugs	4:58:17	0.65%
Milestone 3 : Milestone 3 Document	1:14:01	0.16%
Milestone 3: Admin Login Troubleshooting	1:23:14	0.18%
Milestone 3: Admin Registration	1:53:03	0.24%
Milestone 3: Approval Document	4:26:44	0.58%

USER - TIME ENTRY	DURATION	PERCENTAGE
Milestone 3: Connecting Contact us to Database	0:56:34	0.12%
Milestone 3: CSS for Contact Us Admin side	0:38:48	0.08%
Milestone 3: CSS for Login, Register, Forgot	0:33:26	0.07%
Milestone 3: CSS for Users	1:11:25	0.15%
Milestone 3: Displaying admin data from MongoDB	2:46:30	0.36%
Milestone 3: Displaying contact us data to Admin	0:58:20	0.13%
Milestone 3: Displaying user data from MongoDB	3:26:00	0.45%
Milestone 3: Renaming image for storage in MongoDB	0:36:19	0.08%
Milestone 3: Retrieving data from MongoDB	1:05:39	0.14%
Milestone 3: Storing image in MongoDB	2:13:16	0.29%
Milestone 3: Verifying the users in the Database	1:45:34	0.23%
Milestone 4 (Meeting): Finalizing Sprint 4 and Document	1:48:21	0.23%
Milestone 4 (Meeting): Preparing for demo with client	1:47:23	0.23%
Milestone 4 (Meeting): Progress Check	4:42:03	0.61%
Milestone 4 (Meeting): Sprint 4 Discussion	3:34:37	0.47%
Milestone 4: Adding pending status and Text area for description	4:28:19	0.58%
Milestone 4: Bug fixing and minor updates	7:52:20	1.02%

USER - TIME ENTRY	DURATION	PERCENTAGE
Milestone 4: CSS for Explore and Fixing dropdown for create poll	2:59:26	0.39%
Milestone 4: CSS for Explore page	1:01:22	0.13%
Milestone 4: CSS for UserSidebar and Explore	5:57:47	0.78%
Milestone 4: Report Schema	1:35:52	0.21%
Milestone 4: Storing wallet ID in database	2:12:32	0.29%
Milestone 4: Storing wallet ID in database and Report Poll functionality	10:12:34	1.33%
Milestone 5 (Meeting): Final Submission Discussion	1:25:56	0.19%
Milestone 5 (Meeting): Updating the SRS	2:01:24	0.26%
Milestone 5: Fixing Final Repo Bugs	6:32:29	0.85%
Milestone 5: Working on Final Submission	2:06:52	0.27%
Milestone 5: Working on Presentation	10:26:37	1.36%
 Khushi Dholakia	202:31:24	26.33%
Branching Plan: Research on Branching Plan	1:45:00	0.23%
Branching Plan: Working on the document (Individual Branch)	1:14:25	0.16%
Handover (Meeting): Handover Discussion	4:48:19	0.62%
Handover (Meeting): Handover Finalization	3:45:52	0.49%
Handover (Meeting): Working on Handover	1:44:49	0.23%

USER - TIME ENTRY	DURATION	PERCENTAGE
Handover: Adding screenshots	0:52:09	0.11%
Meeting with Client: Demo	0:39:04	0.08%
Meeting with Client: Final Presentation	0:51:00	0.11%
Meeting: Branching Plan	1:04:51	0.14%
Meeting: Branching Plan Discussion	1:50:56	0.24%
Meeting: Branching Plan Finalising	1:17:48	0.17%
Meeting: Branching Plan Review	1:07:00	0.15%
Milestone 1 (Meeting): BitBucket and Jira	0:46:29	0.1%
Milestone 1 (Meeting): Bug Fixing	0:34:00	0.07%
Milestone 1 (Meeting): Finalising Milestone 1 Document and Code	3:43:17	0.48%
Milestone 1: Bug Fixing the About Us Page	1:30:00	0.2%
Milestone 1: Creating the About Us Page	2:35:00	0.34%
Milestone 1: Creating the Individual Submission	0:08:07	0.02%
Milestone 1: Learning CSS	1:52:00	0.24%
Milestone 1: Learning React	1:33:53	0.2%
Milestone 2 (Meeting): Sprint 2 Discussion	1:35:00	0.21%
Milestone 2 (Meeting): Finalizing Sprint 2	5:10:00	0.67%

USER - TIME ENTRY	DURATION	PERCENTAGE
Milestone 2 (Meeting): Sprint 2 Backend	0:37:00	0.08%
Milestone 2 (Meeting): Sprint 2 Research	1:28:00	0.19%
Milestone 2 : Adding the Backend, connecting mongodb and Register page revamp	10:04:00	1.31%
Milestone 2: Adding JW Tockens	1:30:26	0.2%
Milestone 2: Adding the FAQ page (Collapse functionality)	2:04:26	0.27%
Milestone 2: Create Dashboard and mongo db account	1:14:00	0.16%
Milestone 2: Editing Login backend	2:46:00	0.36%
Milestone 2: Editing the About Us page, Designing Button and adding the Contact Us page	6:50:15	0.89%
Milestone 2: Fixing errors while posting registration data to MongoDB	0:59:12	0.13%
Milestone 2: Implementation of a search bar and basic dashboard page layout	1:31:02	0.2%
Milestone 2: implementation of search bar	3:10:28	0.41%
Milestone 2: Login backend and Hashing the password stored when user registers	1:47:00	0.23%
Milestone 2: Research on Backend and Database Connectivity	3:17:00	0.43%
Milestone 2: Revamping the contact us page	0:24:00	0.05%
Milestone 2: Troubleshooting FAQ collapse	2:20:10	0.3%
Milestone 2: Working on viewing dashboard only after logging in and dynamic name	0:35:44	0.08%
Milestone 2: Writing my individual report	0:37:35	0.08%

USER - TIME ENTRY	DURATION	PERCENTAGE
Milestone 3 (Meeting): Finalizing Sprint 3	1:16:23	0.17%
Milestone 3 (Meeting): Finalizing sprint 3 and Document	2:31:43	0.33%
Milestone 3 (Meeting): Sprint 3 Discussion	3:53:38	0.51%
Milestone 3 (Meeting): Sprint 3 Progress Check	0:53:26	0.12%
Milestone 3 (Meeting): Sprint 3 research	3:53:00	0.5%
Milestone 3: Bug fixes and Learning to Implement a smart contract	0:59:00	0.13%
Milestone 3: Coding the solidity contract	1:44:00	0.23%
Milestone 3: Coding the solidity contract	4:49:07	0.63%
Milestone 3: Debugging the solidity contract	0:42:03	0.09%
Milestone 3: Implementation of Smart Contract	5:24:13	0.7%
Milestone 3: Research on Blockchain	5:35:44	0.73%
Milestone 3: Working on connecting smart contract to frontend	4:54:29	0.64%
Milestone 3: Working on Smart Contract	3:10:19	0.41%
Milestone 3: Working on the blockchain	0:32:00	0.07%
Milestone 4 (Meeting): Finalizing Sprint 4 and Document	2:08:00	0.28%
Milestone 4 (Meeting): Preparing for demo with client	1:47:16	0.23%
Milestone 4 (Meeting): Progress Check	4:36:52	0.6%

USER - TIME ENTRY	DURATION	PERCENTAGE
Milestone 4 (Meeting): Sprint 4 Discussion	4:31:27	0.59%
Milestone 4 : Adding the FAQ text	0:48:50	0.11%
Milestone 4: Editing Solidity contract	1:10:33	0.15%
Milestone 4: Editing solidity, implementing vote, view votes, end poll, view my polls	12:10:15	1.58%
Milestone 4: Getting User input from Creat poll front end and research on connecting blockchain	1:01:13	0.13%
Milestone 4: Getting User input from Create poll front end and research on connecting blockchain	3:13:05	0.42%
Milestone 4: Implementing ending poll according to end date	2:13:20	0.29%
Milestone 4: Implementing forgot password and otp, Editing the milestone 4 document	7:44:28	1.01%
Milestone 4: Implementing Voting functionality	1:09:06	0.15%
Milestone 4: Storing end date in sol contract and giving a select date option on create poll page	1:38:00	0.21%
Milestone 4: Trying NULL value for end Date	2:25:54	0.32%
Milestone 4: Updating Create poll	1:09:00	0.15%
Milestone 4: Working on displaying voter ids on polls	2:28:27	0.32%
Milestone 4: Working on vote on poll	4:20:47	0.57%
Milestone 5 (Meeting): Final Submission Discussion	1:35:03	0.21%
Milestone 5 (Meeting): Updating the SRS	2:04:47	0.27%
Milestone 5: Finalising the individual submission document	0:16:14	0.04%

USER - TIME ENTRY	DURATION	PERCENTAGE
Milestone 5: Fixing Final Repo Bugs	6:13:58	0.81%
Milestone 5: Working on Final Submission	1:18:38	0.17%
Milestone 5: Working on the Handover document	0:46:08	0.1%
Milestone 5: Working on the presentation	2:33:49	0.33%
Milestone 5: Working on the presentation, and making notes for presentation	11:01:52	1.43%