Product Design

Team < Team No : 10 ; Rohit Gowlapalli ,Vempati Siva Koti Reddy , Mourya Losetti , Nirmala Kadali >

Introduction

- **System Overview:**

The system overview section of the Design Document provides a high-level description of the system's functionality, features, and benefits. It also describes the user groups that will be involved in the project, the data sources and outputs, and the overall system architecture.

- **Design Overview (Architectural Design):**

The design overview section provides an in-depth description of the system's architecture, including the hardware and software components, network topology, data flow, and system integrations. This section also describes the different layers of the system, such as the presentation layer, application layer, and data layer.

- **System Interfaces (User Interface):**

The system interfaces section of the Design Document describes the user interfaces that will be used in the system, including their functionality, design, and layout. It also includes details on the usability requirements, accessibility, and responsiveness of the user interfaces.

- **APIs:**

The APIs section of the Design Document describes the application programming interfaces (APIs) that will be used in the system. This section includes details on the types of APIs, their functionality, and their integration with other systems.

- **Model:**

The model section of the Design Document provides a description of the data model that will be used in the system, including its structure, entities, and relationships. It also includes details on the data sources and the data processing and analysis requirements.

- **Sequence Diagram:**

The sequence diagram section of the Design Document provides a visual representation of the system's interactions, including the data flow, the different system components, and the user interactions. This diagram helps to visualise the system's functionality and to identify any potential bottlenecks or issues.

- **Design Rationale:**

The design rationale section of the Design Document provides a justification for the design decisions made during the system's development. It describes the reasons behind the chosen architecture, user interfaces, data model, and other design elements. This section should also provide an explanation of how the design decisions align with the project's objectives and goals.

Overall, the purpose of the Design Document is to provide a comprehensive plan for the project's development. The document includes all the technical and functional aspects of the project, including the system overview, design overview, system interfaces, APIs, data model, sequence diagram, and design rationale. The document serves as a reference for all stakeholders involved in the project, enabling everyone to be on the same page and to ensure the project is delivered on time, within budget, and with the desired level of quality.

System Overview

The Airdrop Insurance application is a software system designed to offer insurance options to farmers and automatically pay them out in the event of natural disasters such as floods. The system uses weather APIs to track weather patterns and issue payouts based on predetermined criteria. Additionally, the system incorporates reinsurance from speculators who bid for insurance options taken by farmers, taking a major share of the premiums and paying out claim amounts when the claim criteria are met.

This system can be accessed through a mobile platform, allowing businesses and residents to purchase personal and crop insurance. The vouchers can be redeemed for the local digital currency or other goods. The program also leverages collective risk to gain stronger bargaining power in reinsurance negotiations.

In real-life scenarios, farmers can use the Airdrop Insurance application to protect themselves against financial loss due to natural calamities. By applying for insurance options, farmers can receive a payout in case of a natural disaster such as floods. This can help them cover the costs of any damages or losses they may suffer, allowing them to continue farming without significant financial burden.

For example, a farmer in a flood-prone area can use the Airdrop Insurance application to apply for insurance options that cover potential flood damage. If a flood occurs, and the predetermined criteria for a payout are met, the system will automatically pay out the claim amount to the farmer, enabling them to recover from any losses they may have suffered.

Another example could be a group of farmers who collectively pool their resources to apply for insurance options. This can allow them to reduce the overall cost of insurance and provide a greater level of coverage. If a natural disaster occurs, the system will automatically pay out the claim amount to the group, allowing them to distribute the funds among themselves and continue farming without significant financial loss.

Design Overview

For the diagram, refer to the “Sequence Diagram” section.

1. Architectural design

We are making our app by using flutter as our frontend and firebase as our backend and database

Flutter Frontend

Flutter is a free and open-source mobile UI framework created by Google and released in May 2017. In a few words, it allows you to create a native mobile application with only one codebase. This means that you can use one programming language and one codebase to create two different apps (for iOS and Android). Flutter consists of two important parts:

- An SDK (Software Development Kit): A collection of tools that are going to help you develop your applications. This includes tools to compile your code into native machine code (code for iOS and Android).

- A Framework (UI Library based on widgets): A collection of reusable UI elements (buttons, text inputs, sliders, and so on) that you can personalise for your own needs.

Firebase backend and database

Google Firebase is a Google-backed application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiments. Firebase offers a number of services, including:

- Analytics
- Authentication
- Crashlytics
- Performance
- Test lab

Subsystems:

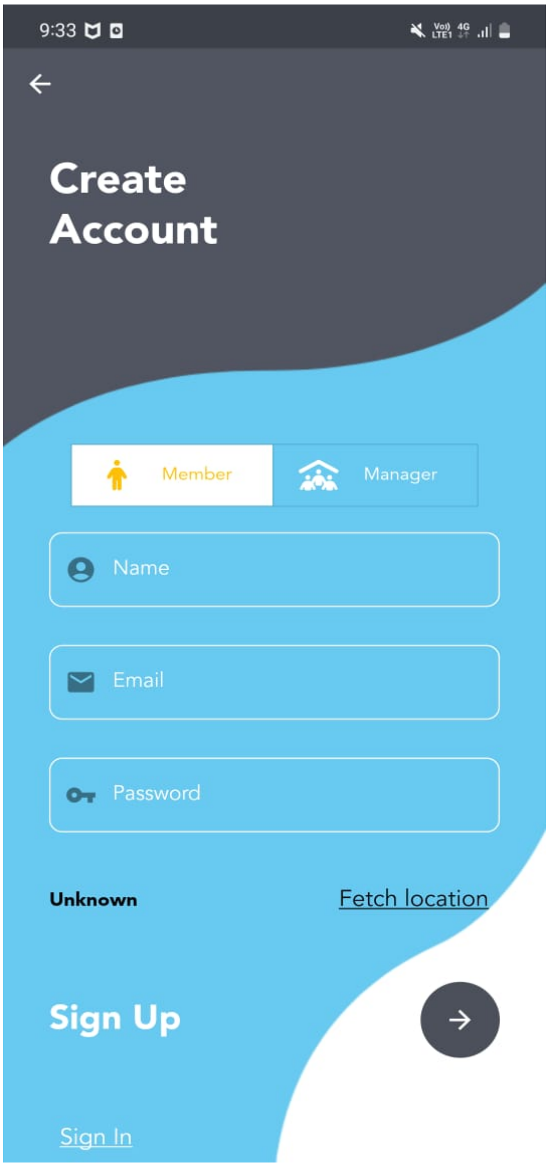
- ➔ **User Authentication:** Four types of users can register to the app: Members , Community-Managers , Admins and Speculators .Once they're registered, the users can login through the login portal and access their dashboards based on their roles.
- ➔ **Reinsurance Embodiment (Insurance Claim)** : It is the process of carrying out and putting into action the use of reinsurance in a particular insurance program or system. It involves incorporating reinsurance as a risk management strategy. It aims to enhance the stability and sustainability of an insurance program. A speculator bids & cashes in the members' insurances and takes on the responsibility of managing payouts to the users while also assuming the risk involved
    1. Members can apply for Claims by Clicking on "Insurance Claims" grid on Dashboard and selecting the respective Insurance option
    2. Members issue Insurance Claim requests when criteria for the claim is met
    3. Speculators payout corresponding amounts for a particular Insurance option after the claim is verified

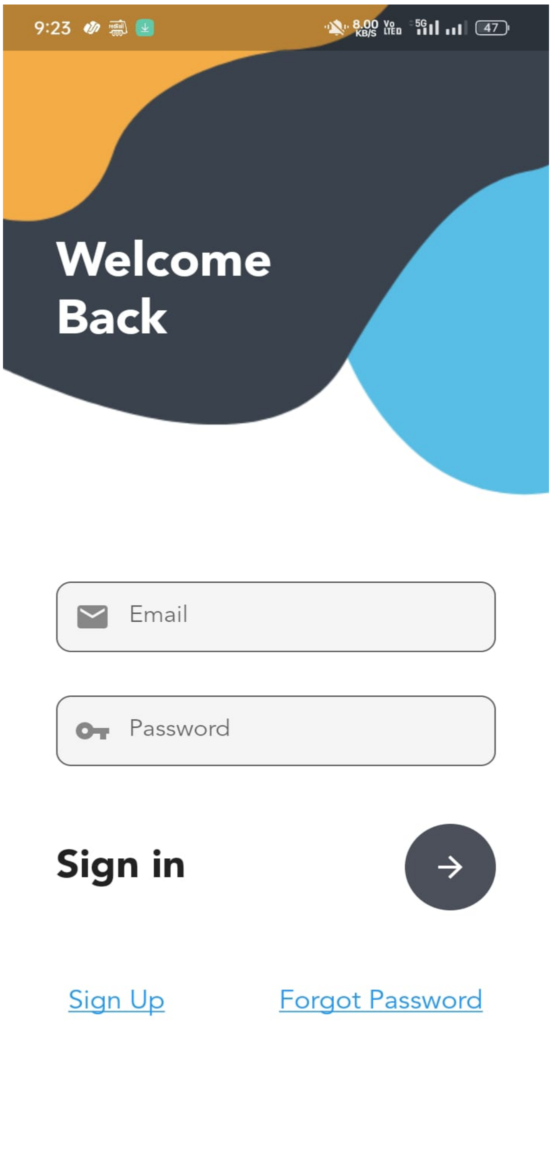
- ➔ **Privacy Protection:** By obtaining only the essential permissions from the user, we can ensure compliance with Google Play Store's terms and conditions, preventing potential issues during app deployment. This approach helps to avoid any violations of the store's policies.
- ➔ **Creating a Poll:**
  1. The users can avail this service of creating a poll by Clicking on “Polls” grid on the Dashboard.
  2. A poll is created only after the approval by a manager.
  3. Polls are commonly used to gather feedback, opinions, and preferences from users. They can be an effective way to engage with app users and get a sense of what they want and need from the app.
- ➔ **Initiating Digital-Token Transactions:** Members can exchange tokens from their wallet to another user of the Application via Meta-mask Wallet Address
  1. Users click on “Wallet” grid on the Dashboard
  2. The user then enters Meta-mask wallet address of the recipient of the transaction and also the amount of tokens to be sent as part of the transaction
  3. Balance of the sender’s wallet is deducted and the recipients is incremented by the amount processed during transaction if the transaction is successful
- ➔ **Bid for Insurance Options:**
  1. Speculator bids for an insurance option and takes on the responsibility of managing payouts to the users while also assuming the risk involved
  2. Speculator is handed over major share of the premiums received from the Insurance option he is taking risk for and is expected to payout to the members when they claim it back

2. *System interfaces*

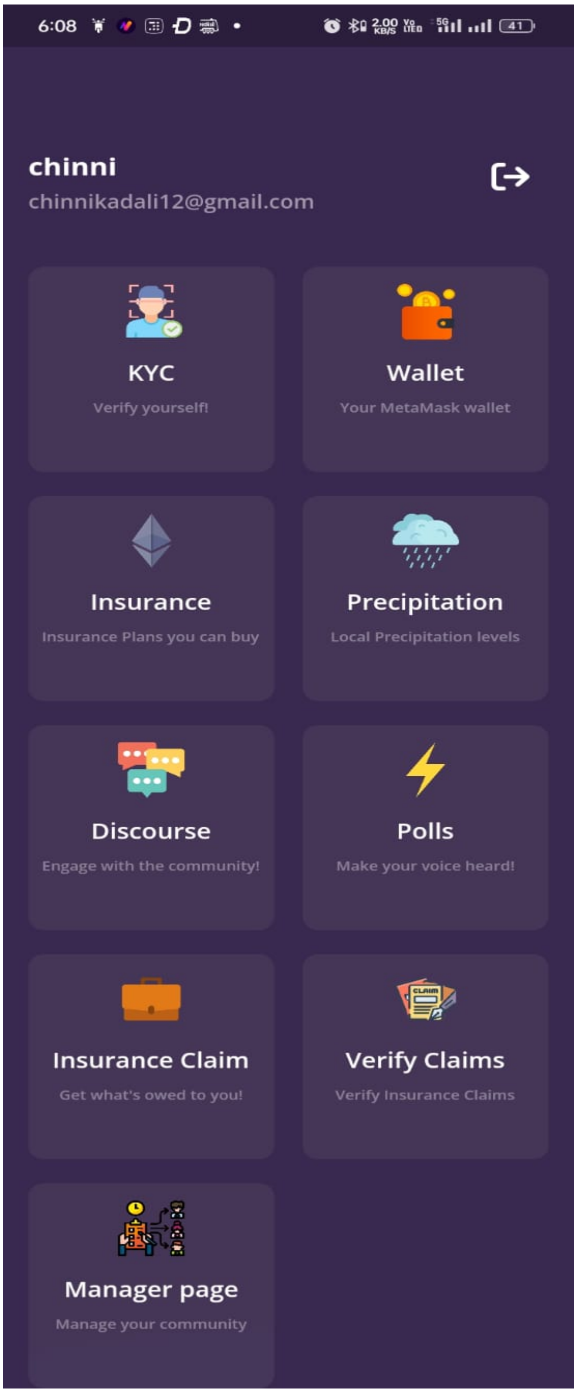
1. User Interface:

- ❖ Users who are new to the application can register by selecting the SignUp option and specifying their particular roles. On the other hand, existing users can log in to the application interface to gain access.



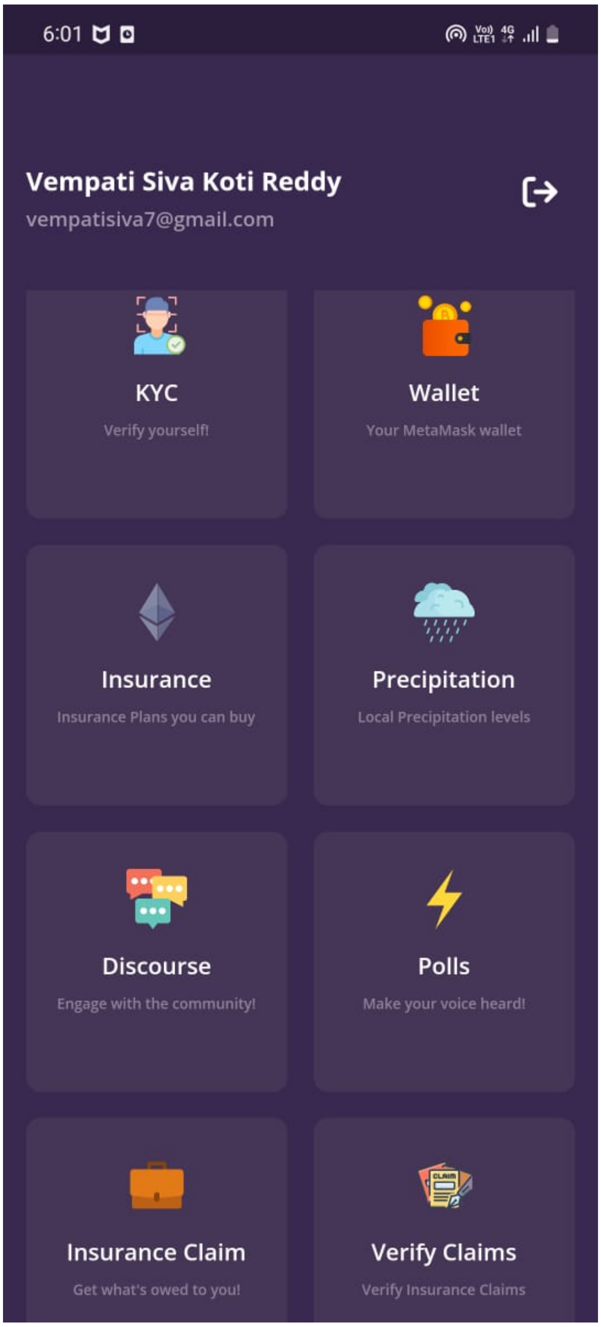


- Based on their roles , Users are redirected to their respective dashboards.
- **Admin:**



a. Admin has the power to select or remove community managers.
b. Admin also adminstrates the Application and creates Insurance options in the Application

- **Community Manager:**



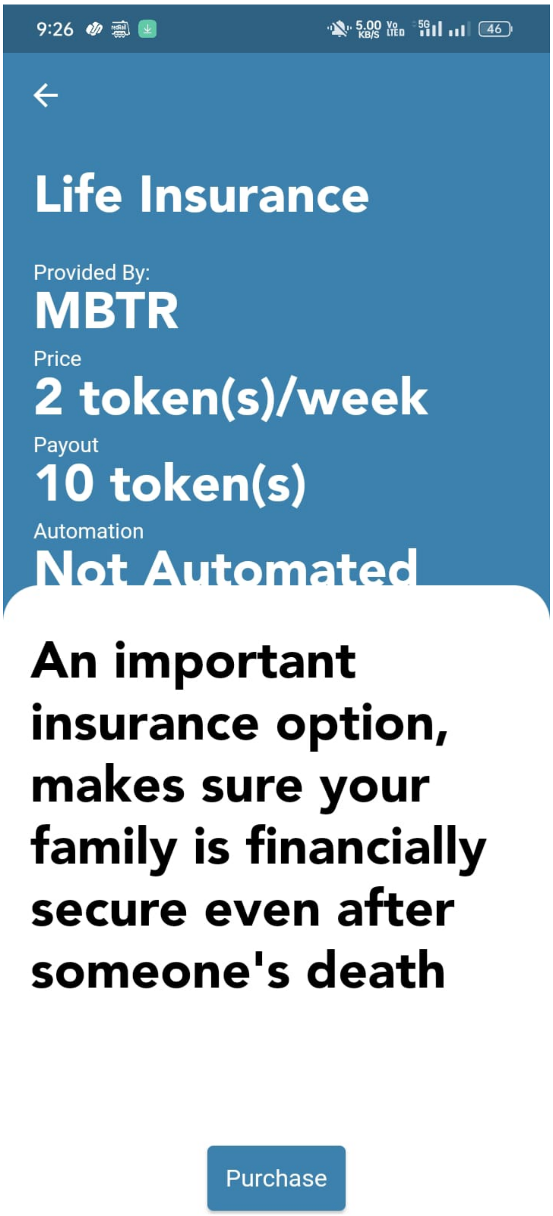
a. Verifies official documents and audio recordings to grant User the access to the Airdrop Application.

- **Speculator:**

  Bids & cashes in the members' insurances and takes on the responsibility of managing payouts to the users while also assuming the risk involved

- Members of the Application buy Insurance options and pay the premium in a recurring fashion

1. Member clicks on the “**Insurance-Option**” grid in the Dashboard and select respective Insurance option of interest
2. Member clicks on “**Purchase**” option on the screen and is expected to pay premiums from start-date of the Insurance option in a recurring fashion



Life Insurance

Provided By:
MBTR

Price
2 token(s)/week

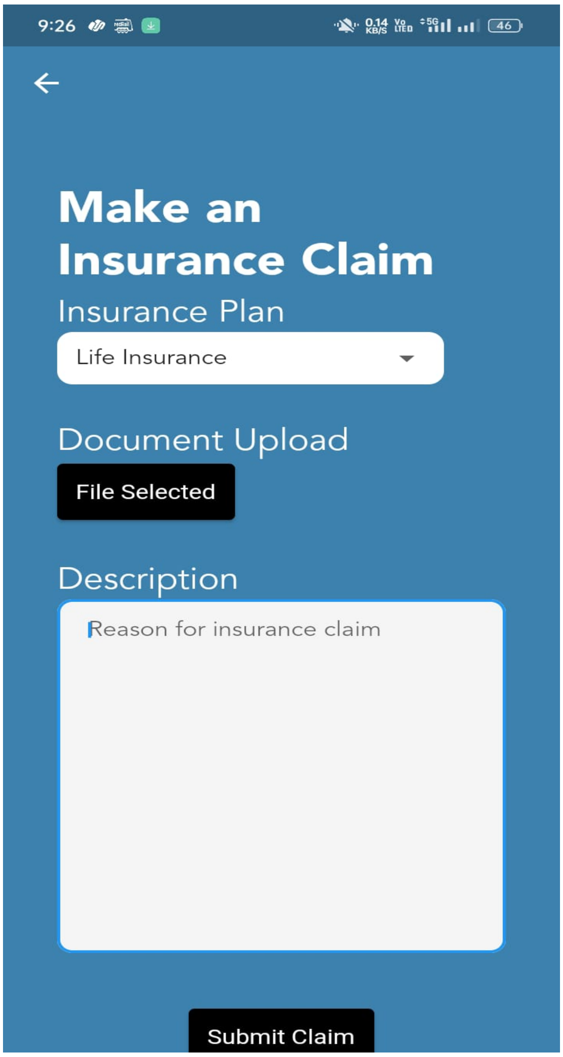
Payout
10 token(s)

Automation
Not Automated

An important insurance option, makes sure your family is financially secure even after someone's death

- Members of the Application can issue Insurance claims when the criteria for the claim is met
  1. Member clicks on **“Insurance-Claim**” grid on the Dashboard and are redirected to another view showing Insurance options purchased by the member

2. Member selects respective Insurance option of choice and Submits document for Claim verification and also provided description for Insurance Claim
3. Member Clicks on “**Submit Claim**” option
4. The Claim is reviewed and if Accepted , member is paid out respective amount corresponding to the Insurance option by the speculator



Make an Insurance Claim

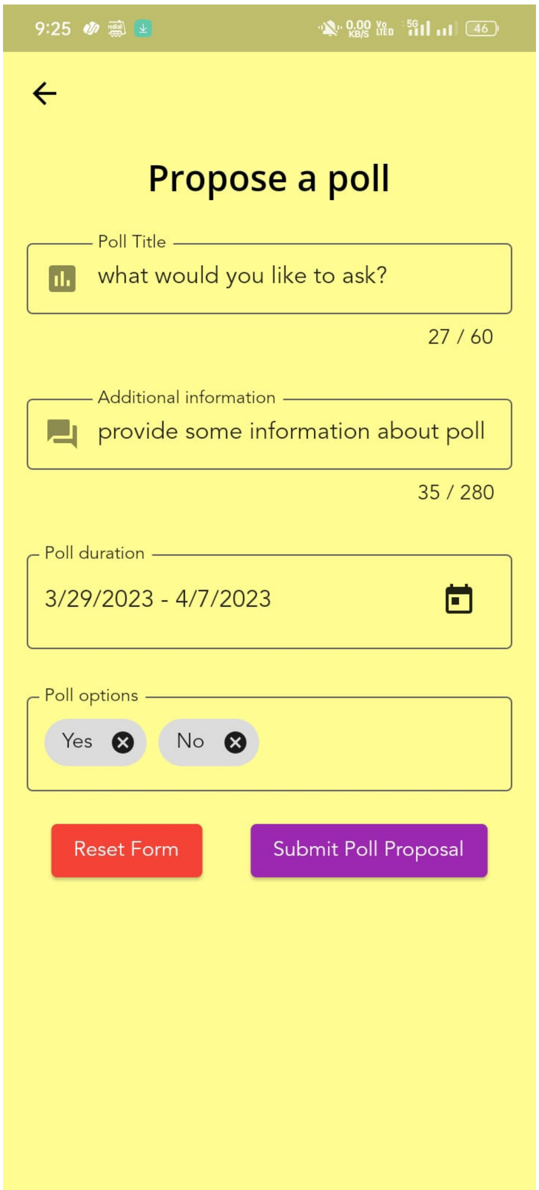
Insurance Plan: Life Insurance

Document Upload: File Selected

Description: Reason for insurance claim

- Members of the Application can also generate polls as a part of discussion or to elect a New Manager
    1. Member clicks on **Polls** grid on dashboard to create polls to generate discussions or to promote themselves to the role of a Manager

2. Member provides Poll Title,Additional Information,Poll Duration , Poll Options
3. Member clicks on “**Submit Poll Proposal**” option
4. The poll is ready to take opinions now !!



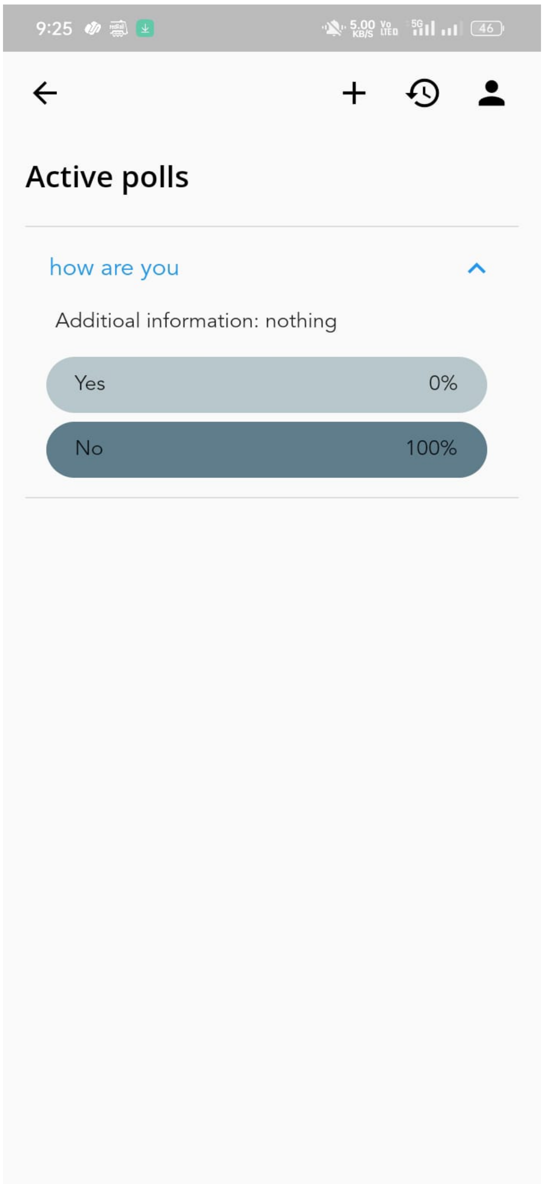
Propose a poll

Poll Title
what would you like to ask?
27 / 60

Additional information
provide some information about poll
35 / 280

Poll duration
3/29/2023 - 4/7/2023

Poll options
Yes No



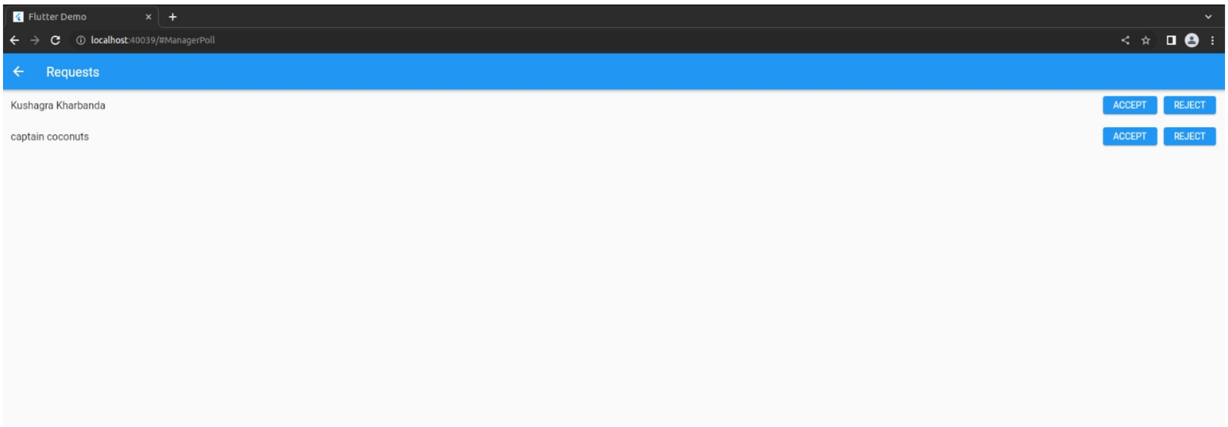
Active polls

how are you

Additioal information: nothing

Yes	0%
No	100%

❖ **Manager-Poll:** An electoral survey aimed at determining whether to elevate a user to the managerial position on the grounds of collective balloting.



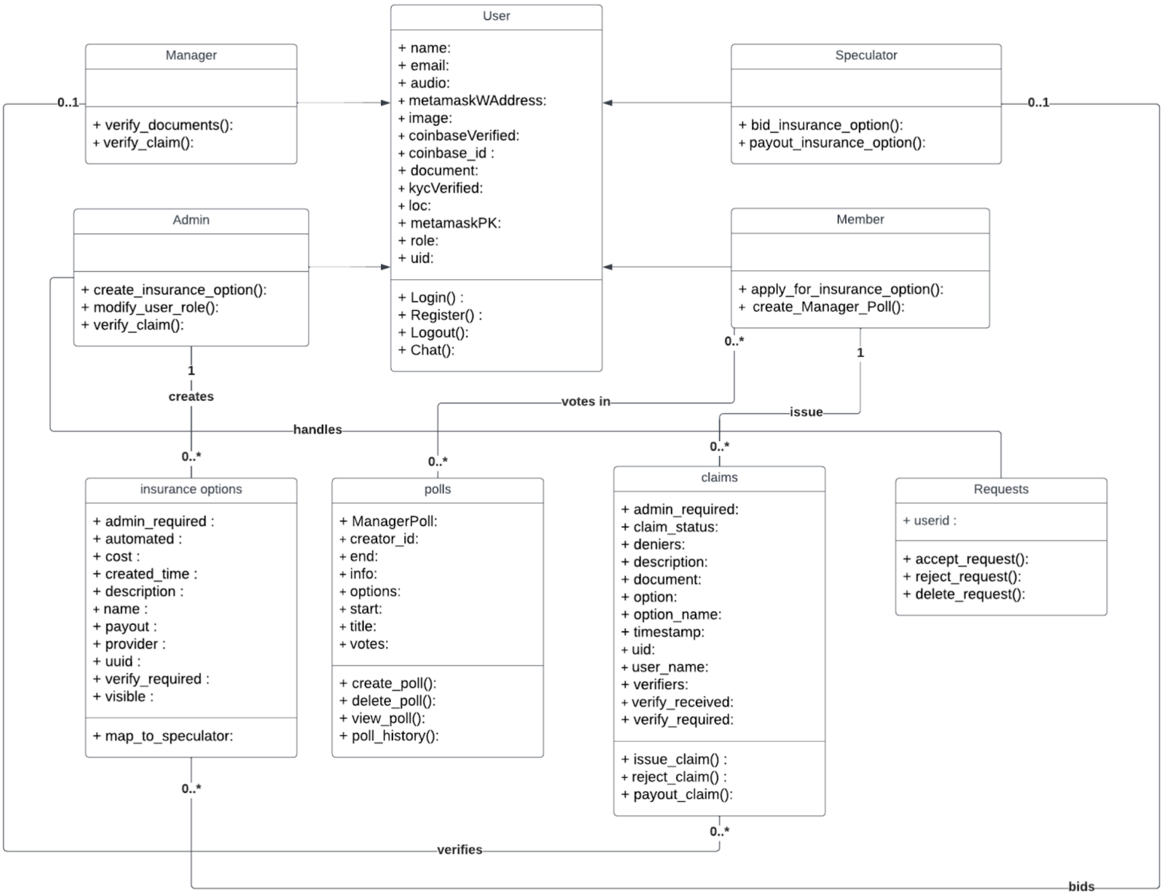
Kushagra Kharbanda

captain coconuts

2. APIs:

Nothing as of now.

3. Model



User	Class state ❖ **name**:Name of the user ❖ **email:**Email of the user ❖ **audio:** Audio of the user used for verification ❖ **metamaskWAddress:**Metamask Address of the user ❖ **image:** Image used for verification ❖ **coinbaseVerified:** variable to check whether if user is verified with his coinbase account ❖ **coinbase_id :** coinbase account id of the user ❖ **document:** Document used for verification ❖ **kycVerified:** variable to check whether if user's documents are verified ❖ **loc:** Location of the user ❖ **metamaskPK:** privatekey of user's metamask wallet ❖ **role:** Role of the user (member , admin , manager , speculator) ❖ **uid:** unique id for user document in firebase ❖ **options:** LIst of all options voted by the user Class behaviour ❖ **Login() :** User logs in to the Application ❖ **Register() :** User registers in the Application ❖ **Logout():** User logs out of the Application ❖ **Chat():** Users interact with each other in Discourse section
Admin	Class state ❖ We will have the same information as the user ❖ **role : "admin"** Class behaviour ❖ **create_insurance_option():** Admin creates insurance options for members to purchase ❖ **modify_user_role() :** Admin can accept or decline the request of a User to be promoted to the role of a Manager ❖ **verify_claim():** Admin can verify an particular Insurance claimed by members
Speculator	Class state ❖ We will have the same information as the user

	❖ **role : “speculator”** Class behaviour ❖ **bid_insurance_option():**speculator bids for a particular insurance option ❖ **payout_insurance_option():**speculator pays out Insurance amount to the member
Manager	Class state ❖ We will have the same information as the user ❖ **role : “manager”** Class behaviour ❖ **verify_documents():**Manager verifies documents of existing members of the Application ❖ **verify_claim():** Manager verifies Insurance option claimed by member
Member	Class state ❖ We will have the same information as the user ❖ **role : “member”** Class behaviour: ❖ **apply_for_insurance_option():** Member apply for an Insurance option and payout premium in a recurring fashion ❖ **create_Manager_Poll():** Members create a Manager_Poll to promote themselves to the role of a Manager based on public opinion
insurance options	Class state ❖ **admin_required :** variable to check whether if admin is required to verify claim ❖ **automated :** variable to check whether if Insurance schema is automated ❖ **cost :** Insurance premium to be paid in a recurring fashion ❖ **created_time :** Time of creation of the Insurance option ❖ **description :** Description of the Insurance Option ❖ **name :** Name of the Insurance Option ❖ **payout :** Amount paid to the member when Insurance is claimed ❖ **provider :** provider of the Insurance option

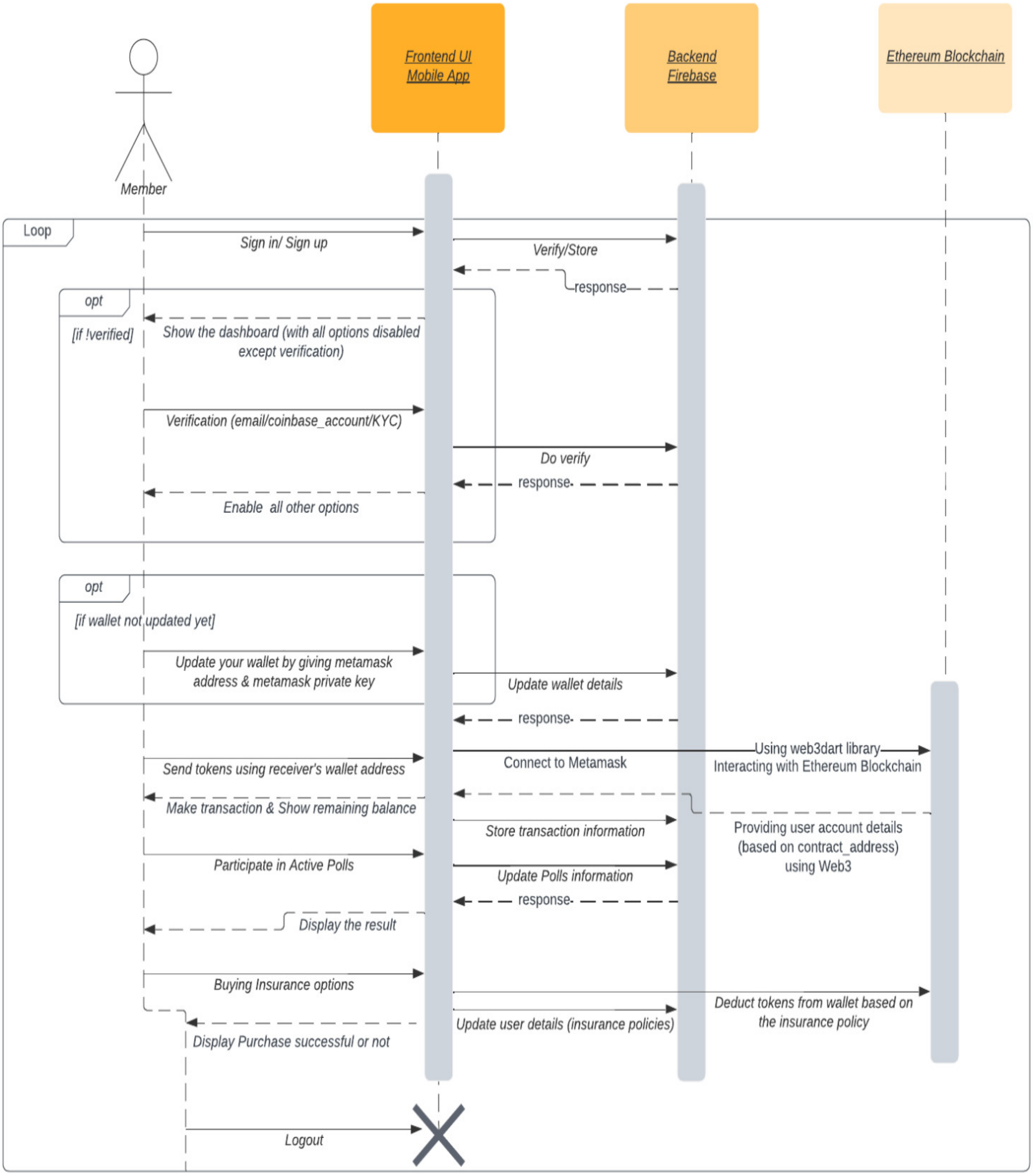
	❖ **uuid :** unique id of the document in firebase ❖ **verify_required :** variable to check whether if Insurance claim needs to be verified ❖ **visible :** variable to display the Insurance option Class behaviour ❖ **map_to_speculator:** Assign the Insurance option to a speculator for reinsurance
claims	Class state ❖ **admin_required:** variable to check whether if admin is required to verify claim ❖ **claim_status:** Number to specify whether if the Claim is successful ❖ **deniers:** List of Manager who have denied the Insurance claim ❖ **description:** Description of the INsurance Claim ❖ **document:** Document used for verification ❖ **option:** Insurance option chosen ❖ **option_name:** Name of the Insurance option ❖ **timestamp:** Timestamp of the Insurance claim ❖ **uid:** unique id for the document in firebase ❖ **user_name:** Name of the user ❖ **verifiers:** List of Managers who have verified the Insurance claim ❖ **verify_received:** Number of Managers who have verified the Insurance Claim ❖ **verify_required:** Minimum Number of Managers required for the Claim to be valid Class behaviour ❖ **issue_claim() :** Claim request is sent for review to the authorities of Application ❖ **reject_claim() :** Claim is rejected based on community guidelines ❖ **payout_claim():** Claim is paid out to the respective member
polls	Class state ❖ **ManagerPoll:** variable to check whether if a poll is to elect a new manager

	❖ **creator_id:** uid of the creator the poll ❖ **end:** end-date of the poll ❖ **info:** information regarding the poll ❖ **options:** options for the poll ❖ **start:** start-date of the poll ❖ **title:** Title of the poll ❖ **votes:** List of votes received for the poll Class behaviour ❖ **create_poll():** Users create new polls ❖ **delete_poll():** Users can delete polls ❖ **view_poll():** Users view polls with detailed view ❖ **poll_history():** History of all polls created by a particular user
Requests	Class state ❖ **userid:** uid of the member who is keen to be promoted to the role of a Manager and also has considerable support from other members Class behaviour ❖ **accept_request():** Accepting request promotes the particular user corresponding to the request to the role of a Manager ❖ **reject_request():** Rejecting request promotes the particular user corresponding to the request to the role of a Manager ❖ **delete_request():** Accepting/Rejecting requests deletes the request after being processed

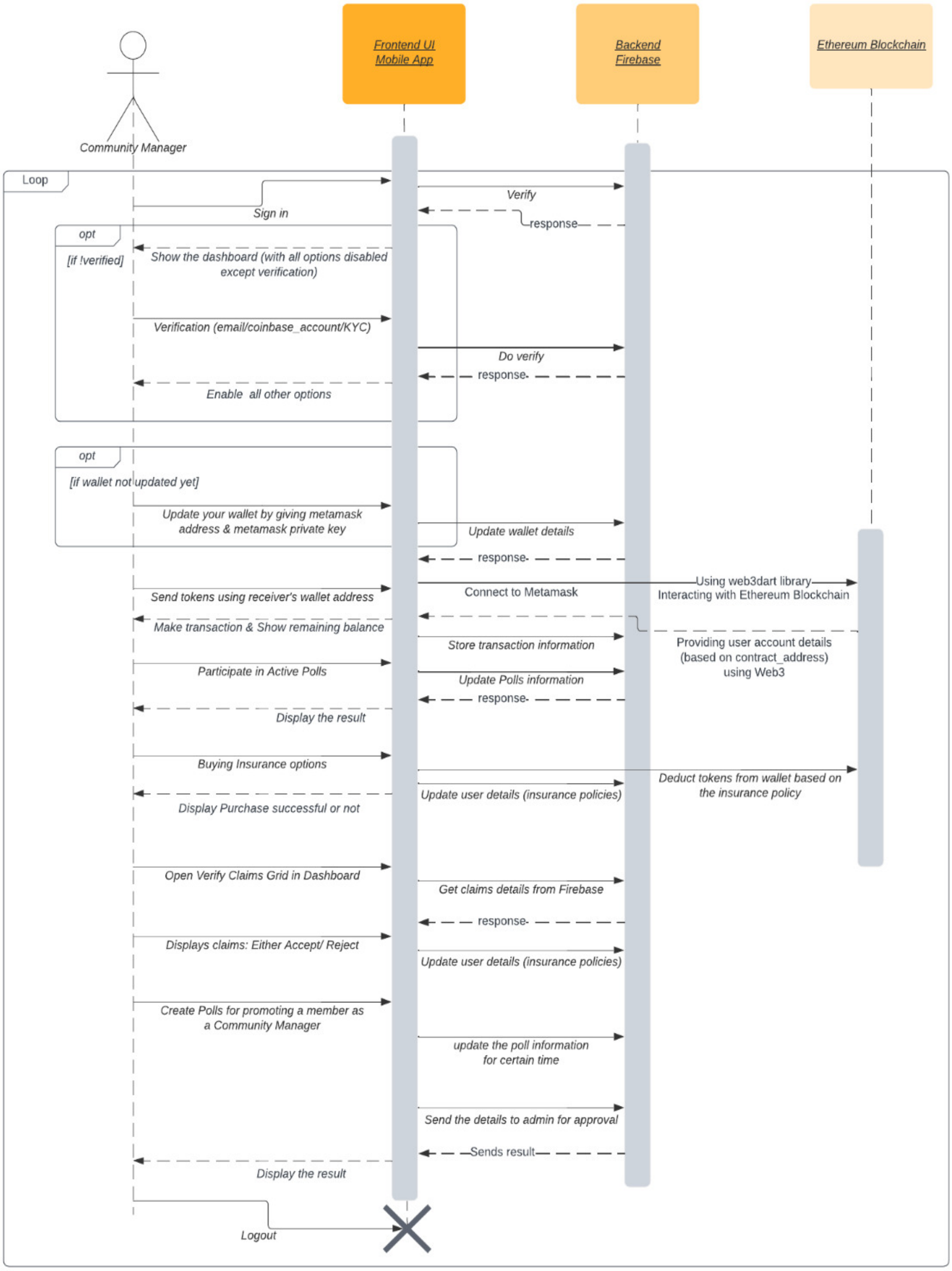
Sequence Diagram(s)

- Sequence diagrams below depict User journey for various roles in Airdrop Insurance Application namely Member , Community-Manager , Admin & Speculator

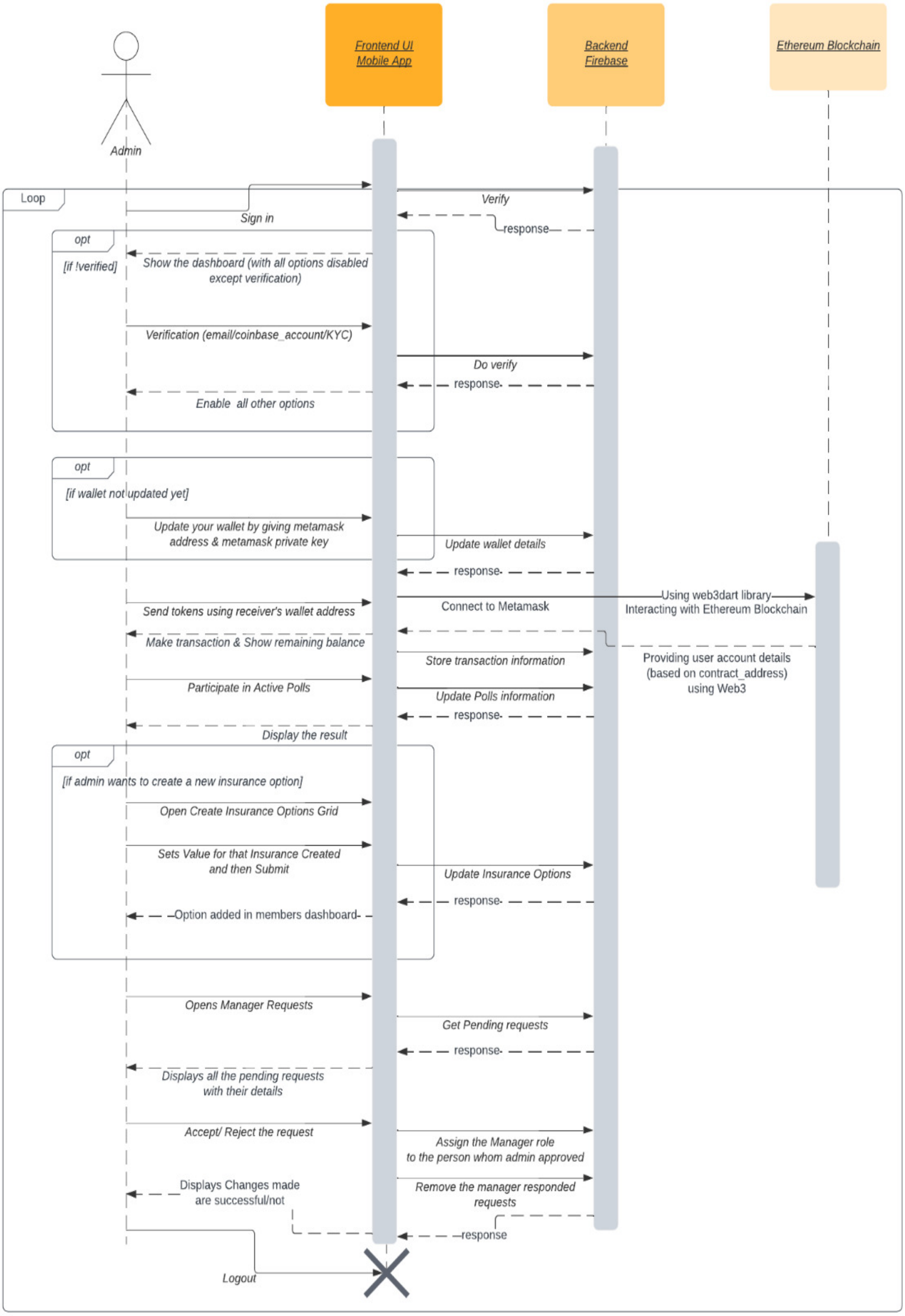
Member
Sequence Diagram


Frontend UI Mobile App
Backend Firebase
Ethereum Blockchain
Member
Loop
Sign in/ Sign up
Verify/Store
response
opt
[if !verified]
Show the dashboard (with all options disabled except verification)
Verification (email/coinbase_account/KYC)
Do verify
response
Enable all other options
opt
[if wallet not updated yet]
Update your wallet by giving metamask address & metamask private key
Update wallet details
response
Using web3dart library
Interacting with Ethereum Blockchain
Connect to Metamask
Send tokens using receiver's wallet address
Make transaction & Show remaining balance
Store transaction information
Providing user account details (based on contract_address) using Web3
Participate in Active Polls
Update Polls information
response
Display the result
Buying Insurance options
Deduct tokens from wallet based on the insurance policy
Update user details (insurance policies)
Display Purchase successful or not
Logout

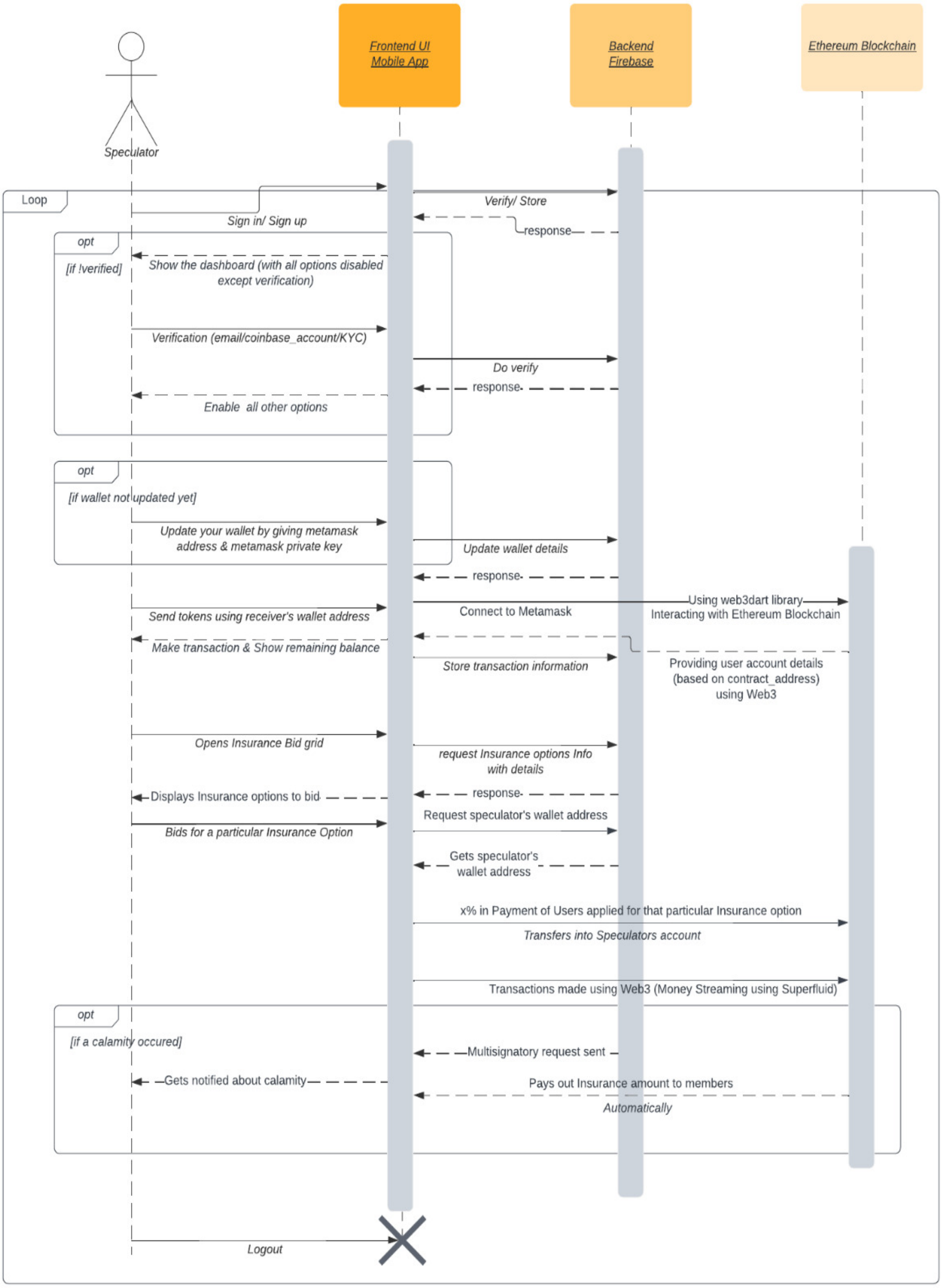
Community Manager Sequence Diagram


Frontend UI Mobile App
Backend Firebase
Ethereum Blockchain
Community Manager
Loop
Sign in
Verify
response
opt
[if !verified]
Show the dashboard (with all options disabled except verification)
Verification (email/coinbase_account/KYC)
Do verify
response
Enable all other options
opt
[if wallet not updated yet]
Update your wallet by giving metamask address & metamask private key
Update wallet details
response
Send tokens using receiver's wallet address
Connect to Metamask
Using web3dart library Interacting with Ethereum Blockchain
Make transaction & Show remaining balance
Store transaction information
Providing user account details (based on contract_address) using Web3
Participate in Active Polls
Update Polls information
response
Display the result
Buying Insurance options
Deduct tokens from wallet based on the insurance policy
Update user details (insurance policies)
Display Purchase successful or not
Open Verify Claims Grid in Dashboard
Get claims details from Firebase
response
Displays claims: Either Accept/ Reject
Update user details (insurance policies)
Create Polls for promoting a member as a Community Manager
update the poll information for certain time
Send the details to admin for approval
Sends result
Display the result
Logout

Admin Sequence Diagram


Frontend UI
Mobile App
Backend
Firebase
Ethereum Blockchain
Admin
Loop
Sign in
Verify
response
opt
[if !verified]
Show the dashboard (with all options disabled
except verification)
Verification (email/coinbase_account/KYC)
Do verify
response
Enable all other options
opt
[if wallet not updated yet]
Update your wallet by giving metamask
address & metamask private key
Update wallet details
response
Send tokens using receiver's wallet address
Connect to Metamask
Using web3dart library
Interacting with Ethereum Blockchain
Make transaction & Show remaining balance
Store transaction information
Providing user account details
(based on contract_address)
using Web3
Participate in Active Polls
Update Polls information
response
Display the result
opt
[if admin wants to create a new insurance option]
Open Create Insurance Options Grid
Sets Value for that Insurance Created
and then Submit
Update Insurance Options
response
Option added in members dashboard
Opens Manager Requests
Get Pending requests
response
Displays all the pending requests
with their details
Accept/ Reject the request
Assign the Manager role
to the person whom admin approved
Displays Changes made
are successful/not
Remove the manager responded
requests
response
Logout

Speculator Sequence Diagram


Frontend UI
Mobile App
Backend
Firebase
Ethereum Blockchain
Speculator
Loop
Sign in/ Sign up
Verify/ Store
response
opt
[if !verified]
Show the dashboard (with all options disabled except verification)
Verification (email/coinbase_account/KYC)
Do verify
response
Enable all other options
opt
[if wallet not updated yet]
Update your wallet by giving metamask address & metamask private key
Update wallet details
response
Send tokens using receiver's wallet address
Connect to Metamask
Using web3dart library
Interacting with Ethereum Blockchain
Make transaction & Show remaining balance
Store transaction information
Providing user account details (based on contract_address) using Web3
Opens Insurance Bid grid
request Insurance options Info with details
Displays Insurance options to bid
response
Request speculator's wallet address
Bids for a particular Insurance Option
Gets speculator's wallet address
x% in Payment of Users applied for that particular Insurance option
Transfers into Speculators account
Transactions made using Web3 (Money Streaming using Superfluid)
opt
[if a calamity occured]
Multisignatory request sent
Gets notified about calamity
Pays out Insurance amount to members
Automatically
Logout

Design Rationale

- ★ Our secondary goal in our Initial phases of the Project was to generate a Monte-Carlo Simulation to predict weather patterns based on given weather-data ; but we were told to go ahead with Reinsurance Money streaming and Implementation of Speculator role in our Application instead by our Client
- ★ Our client stated that the current state of the UI/UX design to be rudimentary and lacking in professionalism and hence , we have decided to improvise design by providing more symbols so that it is easier to understand the Dashboard
- ★ To create new digital tokens to test the app whether transactions are happening or not, we went through different resources and tried testnets like Sepolia faucet, Goerli faucet to generate tokens but finally Polygon (Mumbai testnet) worked as it is cheaper and we can generate tokens easily.
- ★ Our attempts to deploy our app for closed testing on the Google Play Store were repeatedly rejected, with the same error message appearing every time. It took us a significant amount of time to realise that the issue was caused by the Google Play Console failing to properly remove the first version we had released. This meant that even when we were deploying error-free versions of the app, the Play Store was still using the first version. In the end, we had to create a new app deployment to successfully pass the testing phase.