Project Summary:

- **Project Name:** Plutus
- **Team members:** Lucas Gama (108696592), Felipe Lima (109290055), and Hyden Polikoff (109032718)
- **Project overview:** For this project, we decided to create a mobile (based on React Native) application that will act as a simplified portfolio manager. The application will allow users to create an account, log in, register stocks, REITs and funds, keep in track of each stock weight on their portfolio, visualize the rentability of each stock as well as the average cost per stock. The intention is that this application will be used on a recurring basis by investors, therefore, there isn't a point in which we consider that the application will be "done". However, after a user interaction, we hope to deliver easy and digestible information about the user's investment portfolio.

Project Requirements:

- **Requirements:**
    a. Requirement of save and register user's login information (email and password)
    b. Requirement of register a new stock, fund, and/or REIT's and add it to the current user's portfolio
    c. Requirement of taking user's *'category'* input (category input will allow users to combine a set of stocks into a unique category)
    d. Requirement of calculate new weight portfolio upon user's update on it
    e. Requirement of calculating and creating visual representation of the rentability of each stock, the average stock cost and the rentability of each category created by the user.
- **Responsibilities**
    a. Object responsible for creating profiles and managing the workflow related to that task
    b. Object responsible for storing and manipulating all the profile and stock related data
    c. Method responsible for calculating user's portfolio diversity and rentability.

d. Object responsible for creating graphs and visual information about user's portfolio
e. API responsible for gathering current price on user's stocks.

Users and Tasks: Use Cases

- **Number and types of users & tasks they perform:**
  - For our project we will only have one type of user, the main user. The main tasks to be performed by the user are to create an account within the Plutus system, register investment assets and create categories of types of investment.
- **Use Cases:**

Use Case	Create account
Actor	Main user
Basic Path	1. Access app 2. Tap the “Register account” button 3. Enter requested information a. Email b. Password 4. Click on the "Done" button
Alternative Path	At step 3 of the basic path, if the user enters an invalid email address, prompt for a valid one. At step 3, require the user to input a password with at least 8 characters, 1 upper case, 1 lower case, 1 number and one special character. In case criteria is not met, prompt for a new password.
Extensions	*none*
Goal	Have new account registered on app's database

Use Case	Sign In
Actor	Main user
Basic Path	1. Access app 2. Enter requested information a. Email b. Password 3. Click on the "Sign In" button
Alternative Path	At step 2 of the basic path, if the user enters an invalid email address, prompt for a valid one. At step 2, if the user enters a invalid password, prompt for a valid one.
Extensions	*none*
Goal	Sign in to the application and visualize user's exclusive view

Use Case	Log out
Actor	Main user
Basic Path	Click on “Log out” button
Alternative Path	*none*
Extensions	*none*
Goal	Log out of the application

Use Case	Register new asset
Actor	Main user
Basic Path	1. Click on "Add new Investment" button 2. Enter key information: a. Asset name (stock, REIT, or cryptocurrency name) b. Asset quantity c. Asset current price d. Asset type 3. Click on "Done" button
Alternative Path	At step 2 of the basic path, if the user enters a invalid type for 'Asset Quantity' and/or 'Asset current price', throw an error and prompt for a correct data type. If desired, the user can click on the "Cancel" button and cancel the entire operation
Extensions	At step 2, the user can also add a "Category" for the asset he/she is inputting.
Goal	Have new asset added to the app's database and ability to visualize new asset on app's main screen

Use Case	Check asset
Actor	Main user
Basic Path	1. Click on holding ticker
Alternative Path	*none*
Extensions	1. If desired, user can edit an asset 2. If desired, user can delete an asset
Goal	View an asset information, such as the asset performance

Use Case	Edit Asset
Actor	Main user

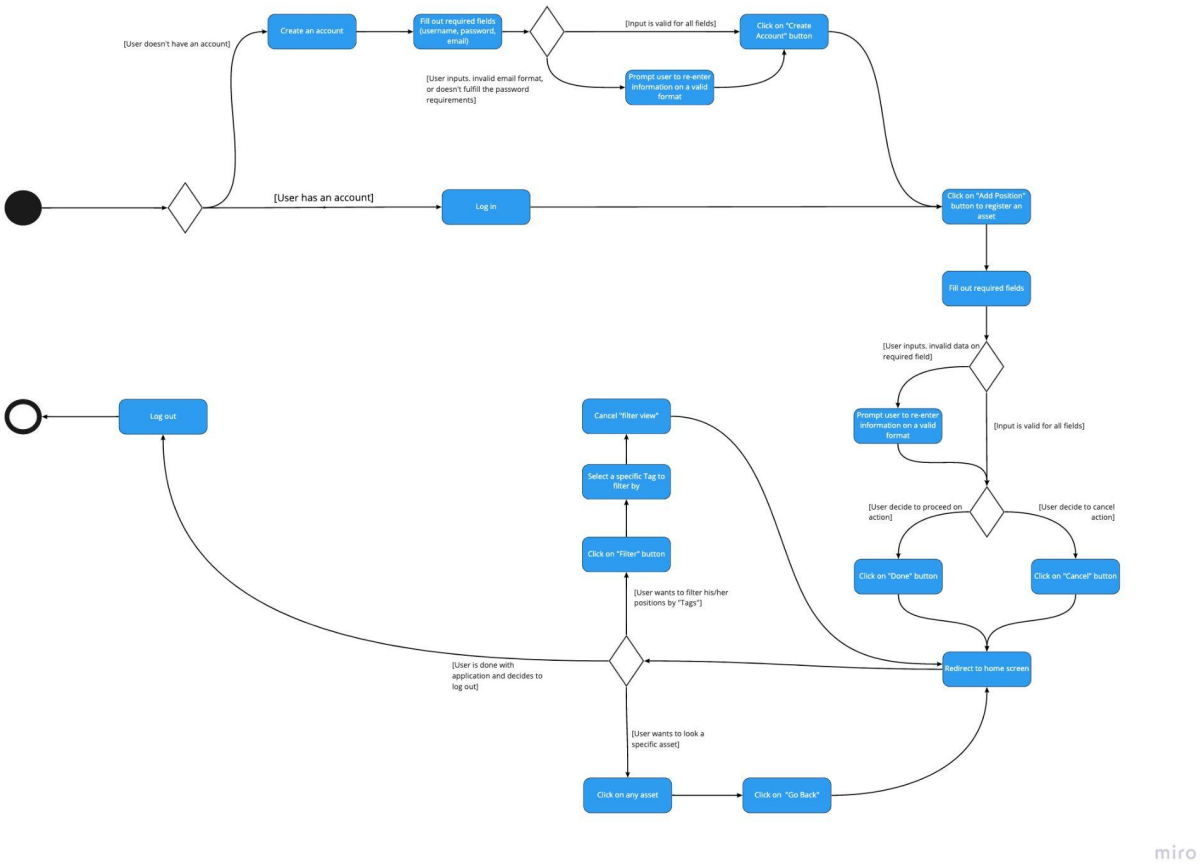
Basic Path	1. Click on the desired holding
	2. On the pop up box click on “Update”
	3. Fill out necessary information
	  a. Bought or sold asset
	  b. # of shares
	  c. Average price
	  d. Tags
	4. Click on “Save”

Alternative Path	

Extensions	1. Deleting an asset
	  a. Click on the desired holding
	  b. On the pop up box click on “Update”
	  c. Click the “Delete” button

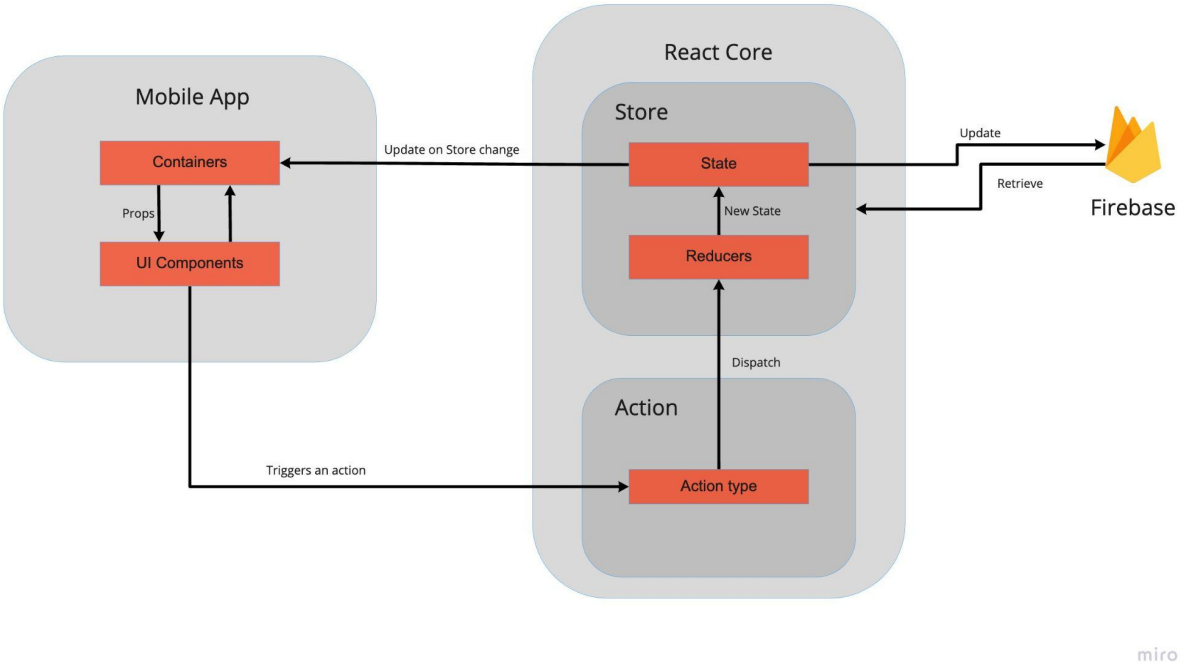
Goal	Edit information on an asset

UML Activity



- Link for the [UML Activity](). Please access it in order to visualize the complete UML Activity diagram.

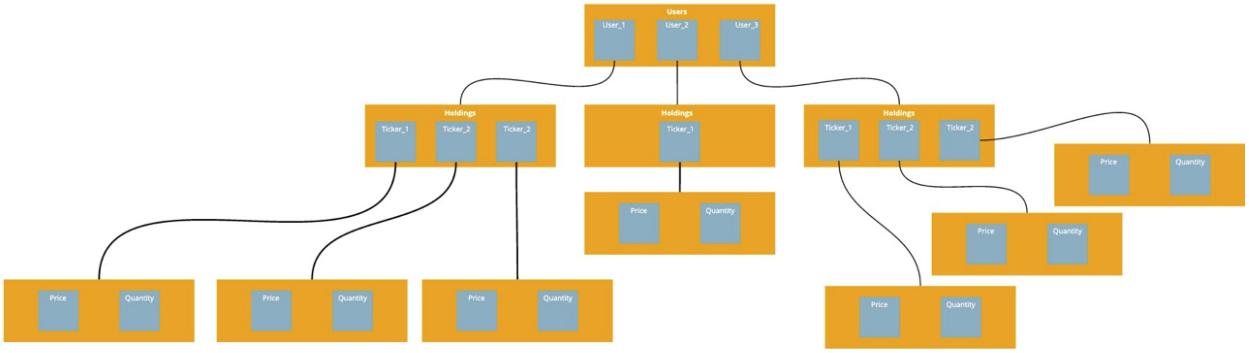
Architecture Diagram



- Link for the [Architecture Diagram](). Please access it in order to visualize the complete UML Activity diagram.

Data Storage

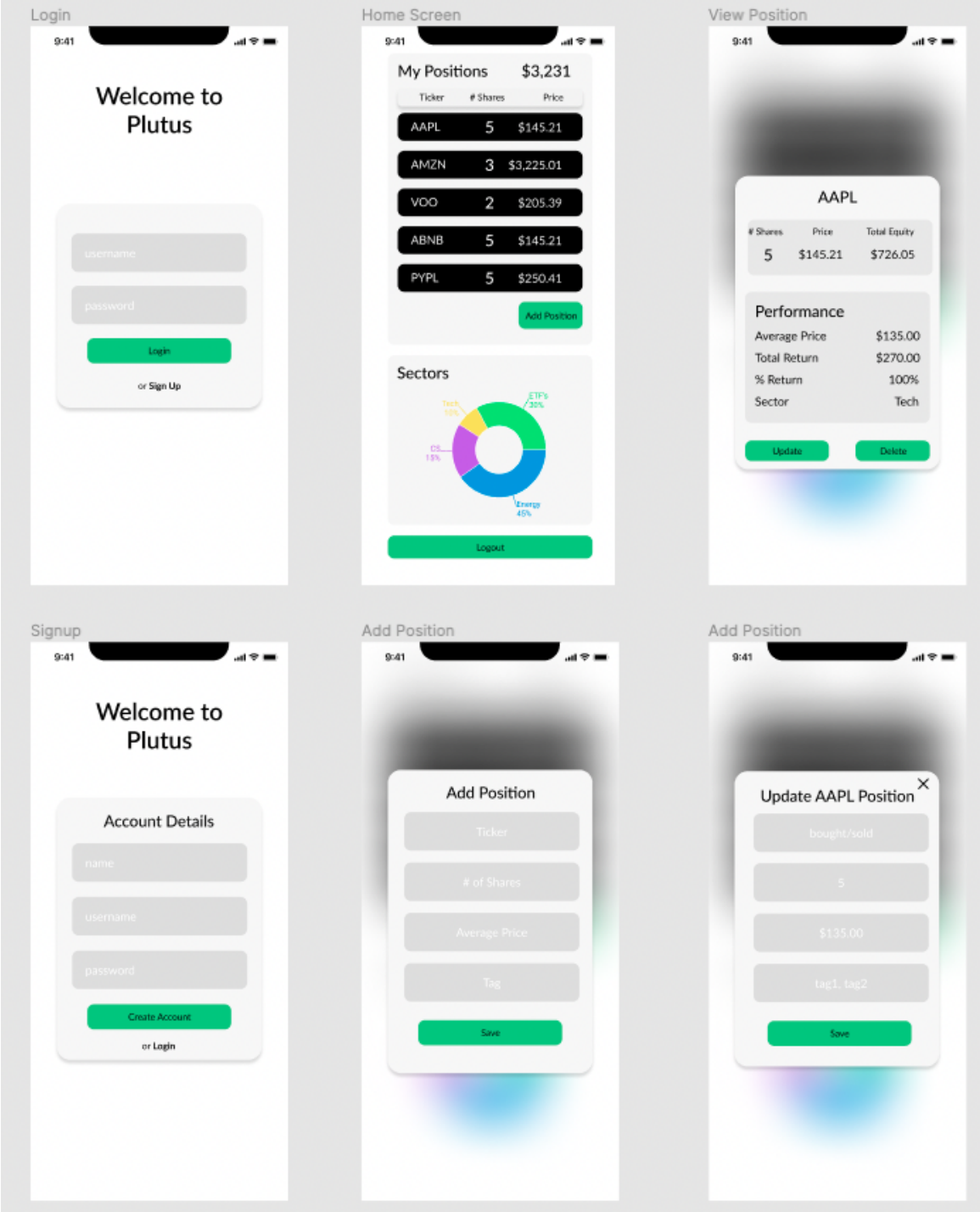
We are planning on using Firebase for all our data and user authentication needs. In order to retrieve data from the database we are using each pertinent class to call for the necessary information on: “User”, “Holdings”.



- Link for the [Data Storage Diagram]

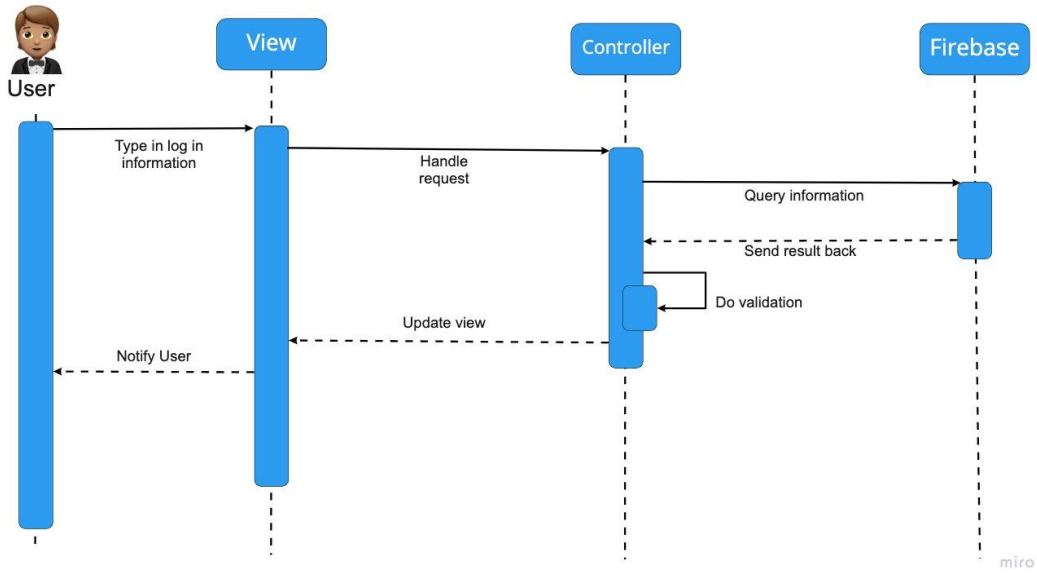
UI Mockups

Here are some previews of the UI as well as [the link] to our Figma file!



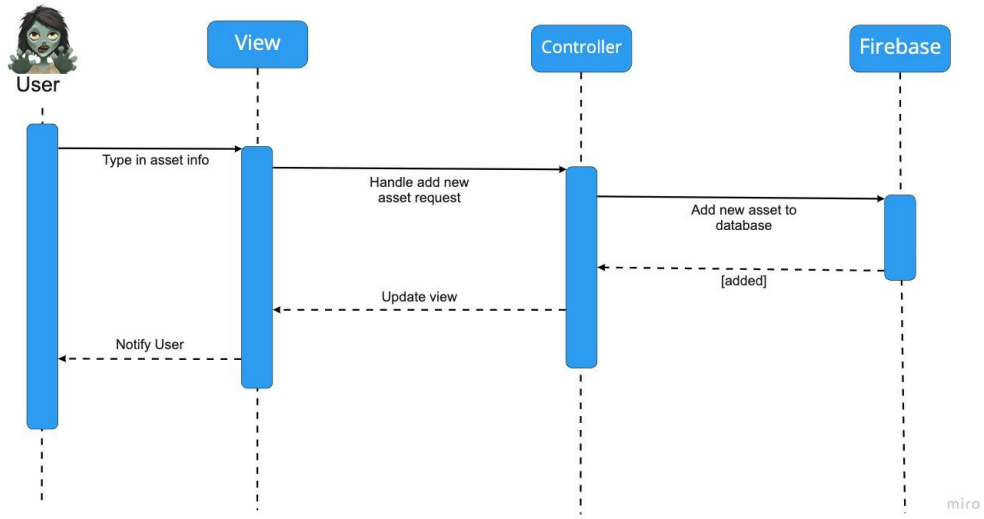
User Interaction/UML Sequence Diagrams

- **Possible user interactions:**
    - **Log in:** On this interaction, the user will simply enter his/her credentials on the view object. These credentials will be sent to the controller so it can make a request to the server on the given credentials. The Firebase will be responsible to find the credentials, and return a message of validity or not. Then the controller will make a request to update the view, taking the user to the home screen.

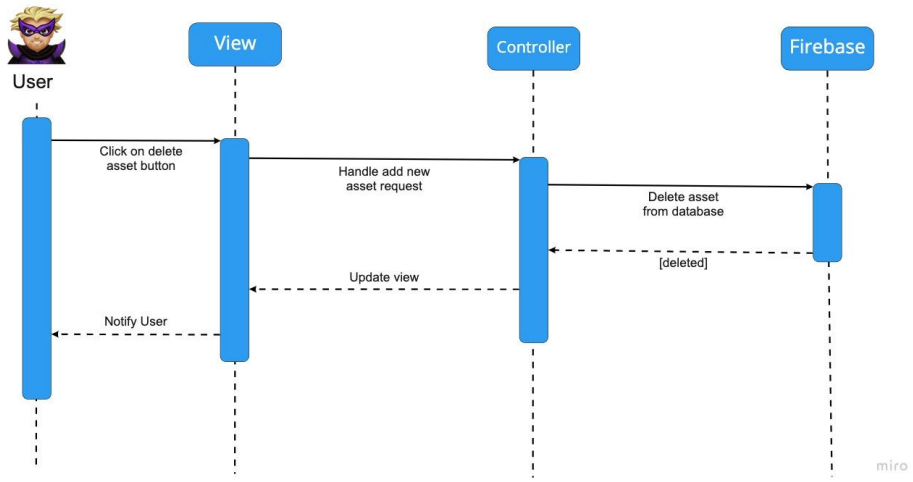


    - **Register a new asset:** To add a new asset, the user will input all needed information on a view object. This information will be sent to the controller so it can make a request to the server to add the new asset. The Firebase will add the new asset to the database. Once done, the

controller will make a request to the view to be updated and the view will notify the user.

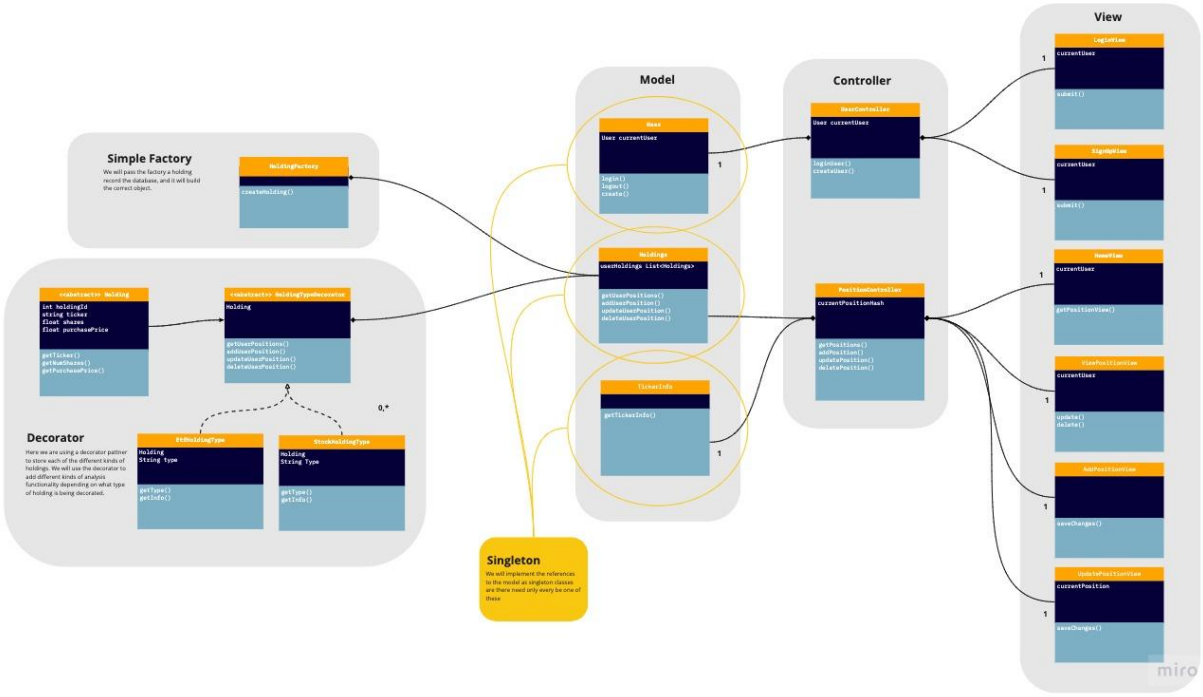


- **Delete an existing asset:** To delete a new asset, the user will click on the delete asset button on a view object. This action will be sent to the controller so it can make a request to the server to delete the specified asset. The Firebase will delete the asset from the database. Once done, the controller will make a request to the view to be updated and the view will notify the user.



All of these diagrams can be seen in a better resolution by accessing [this Miro board](#).

UML Class Diagram



- Link for the [Class Diagram]