Name of the Project: Plutus

Team members: Lucas Gama , Hyden Polikoff , Felipe Lima

Demo Video URL:

https://drive.google.com/file/d/1CRsIoIa6tQ5lKH5lBNxu9AQ30LFuxG5D/view?usp=sharing

Final State of System Statement:

We are really proud of the results the whole team was able to achieve with our application, Plutus. In the last few weeks, we were able to implement almost all features promised in project 5. The implemented features are the following:

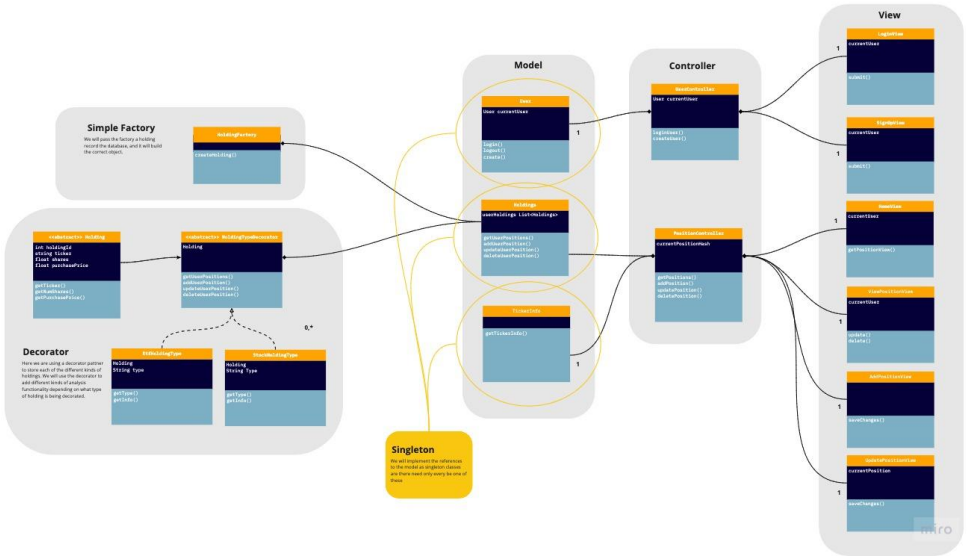
- Login/create account
- Ability to register stocks, REITs and/or funds
- Edit the total number of shares and tags of a current holding
- Delete a current holding
- Display data about user's assets
- Logout from the app

The features the team wasn't able to implement in time are the summary graph of user's holdings, display the weight that each asset represents in the total user's portfolio and display the overall user rentability. Those were secondary features, therefore, we focused on the implementation of the most important features first. We plan to implement the rest of these features in the future. Other than that, the only things that changed from project 5 and 6 to the final project results are the app style choices, the class diagram and the usage of design patterns (the last two changes will be better explained in other parts of this document).

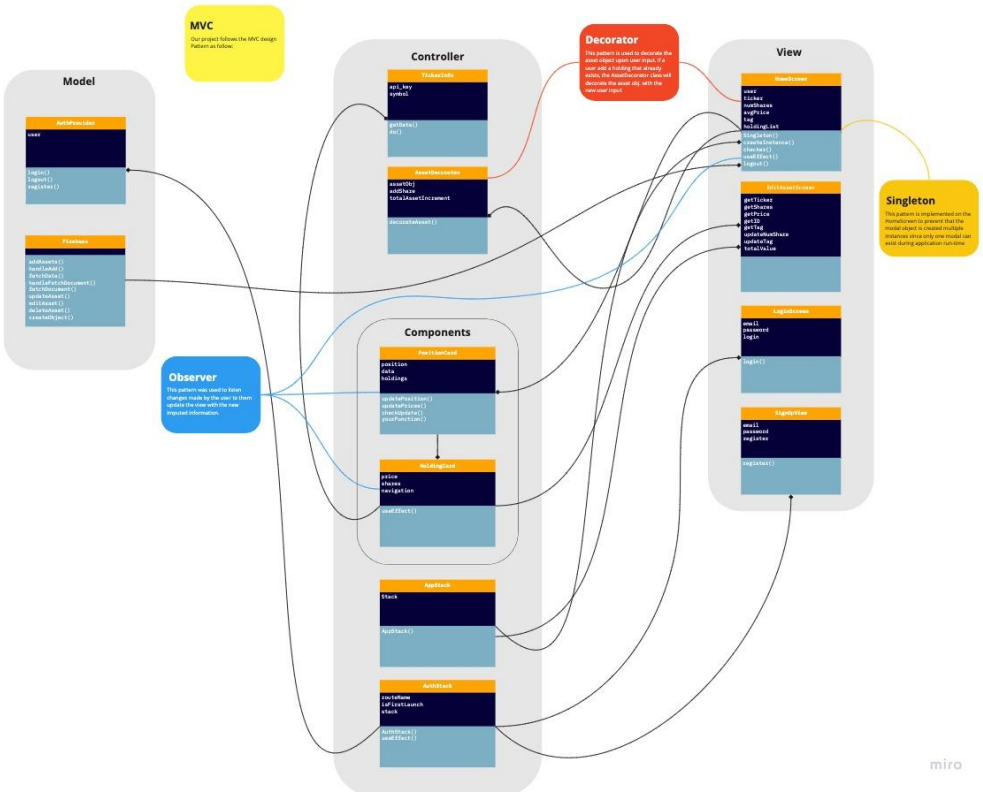
Final Class Diagram and Comparison Statement:

[Link for class diagram and more!](#) (final class diagram is at the right button of the miro board named after *Class Diagram FINAL - Project 7*

Class Diagram Project 5:



New class diagram:



A lot has changed from project 5 diagram to the final class diagram. The main changes are the class names, its functions, and the flow in which each class communicates with each other. For example, we have decreased the number of views on the last deliverable, model classes and added more classes acting as controllers. These changes were key for Plutus application to work as expected. In addition, we decided not to move forward with the Factory Pattern. We kept the Singleton and Decorator Pattern, however, we implemented them in different classes and for different purposes then from what was outlined in Project 5. More details on how these Patterns are being used can be find in the class diagram in Miro.

Third-Party code vs. original code Statement:

For this section it is important to say that less than 20% of the whole project code is third-party code. The places that have some usage of third party code are some parts of the login/create account screens and logic flow, some parts of the navigation component, some parts of the add/edit/delete firebase doc flow, and

lastly, some parts of the observer pattern logic implementation. The resources used are the following:

- https://javascript.plainenglish.io/react-hooks-and-the-observer-pattern-1e4274f0e5f5
- https://blog.logrocket.com/navigating-react-native-apps-using-react-navigation/
- https://www.codegrepper.com/code-examples/javascript/firestore+delete+document+by+id
- https://reactnative.dev/docs/intro-react
- https://firebase.google.com/docs/firestore/manage-data/add-data
- https://firebase.google.com/docs/auth/web/start
- https://github.com/jeremybarbet/react-native-modalize

Statement on the OOAD process during Semester Project:

- UML diagrams before-hand: Even though these might be helpful in designing a good program and serve as a guideline for the design patterns to be implemented, in our case it didn't help us much. Since we were almost completely new to React Native and the React framework in general, our initial conception of how to program would look ended up being very different from the final product, and the UML diagram along with the patterns changed quite a lot. I wouldn't say it was a negative experience per se, but it just wasn't much help.
- Design patterns: Given the chosen framework of React Native the implementation of the design patterns wasn't the easiest. A lot of the functionality and usefulness of implementing these patterns are covered automatically by React, which we could spot in the code, so we had to work around this to implement the proposed design. It was an extra challenge, but in the end it helped understand the functionality and the different ways of implementing them for different languages. We would say it was a positive learning experience.
- MVC: The usage of Firebase framework was a great choice made by the team because it allowed us to focus on the key features of the application. With Firebase handling both authentication and data storage we had more time to

work on the actual application and the different functionalities we wanted to implement (such as the design patterns for example). Really positive experience on this side.