Design Document for MoneyFlow (Formerly MoneyFlow Insight)

Group TA4_1

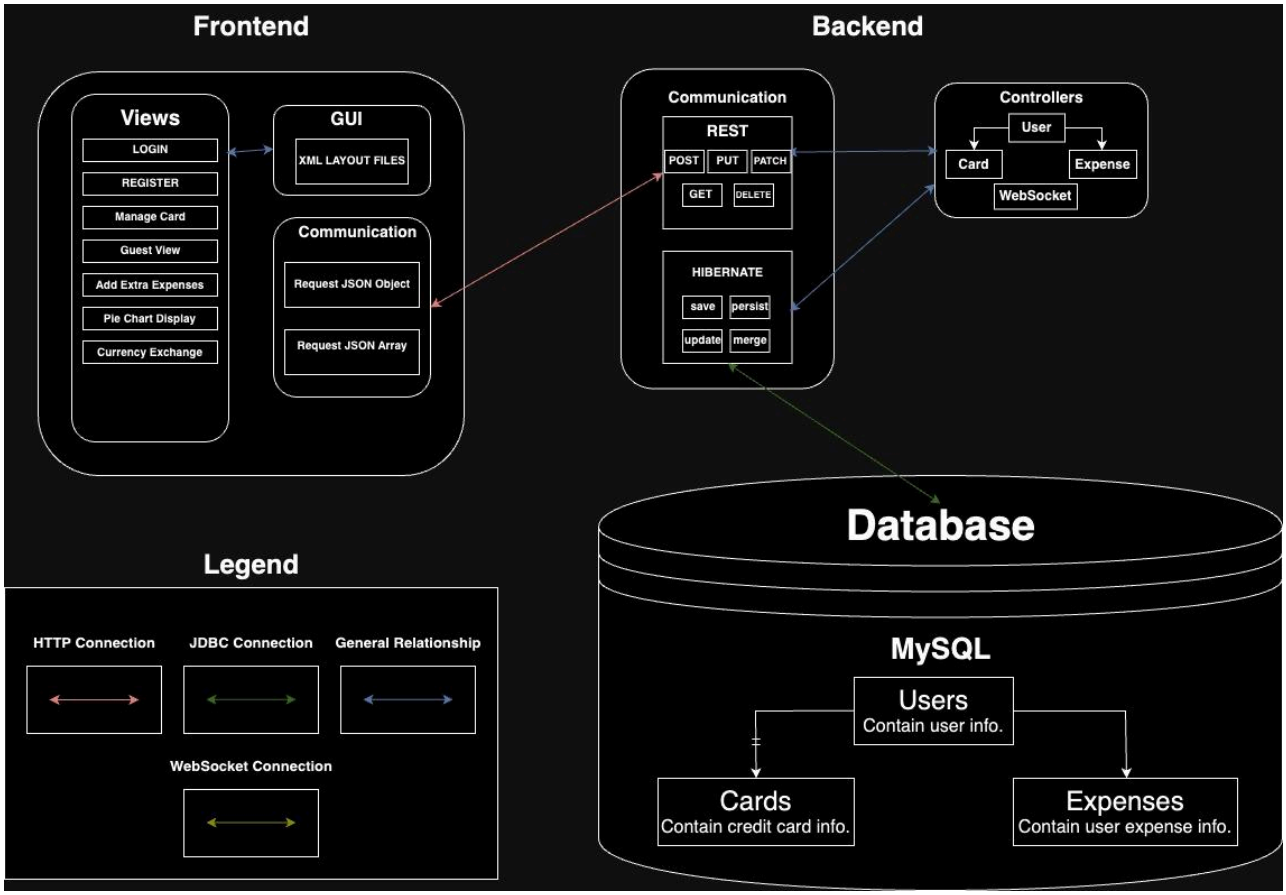
Prakeerth Regunath: 25 % contribution

Onur Onal: 25% contribution

Kemal Yavuz: 25% contribution

Erroll Barker: 25% contribution

PUT THE BLOCK DIAGRAM PICTURE ON THIS PAGE! (Create the picture using pencil or drawIO)



Use this third page to describe complex parts of your design.

Frontend

The frontend designs the main frame of the app using XML layout, as well as setting up different windows that redirect based on what button the user inputs on.

Login

When loading up the app, the user is greeted with a login, signup, and a guest view button. Upon clicking the guest view, the user will be redirected to a screen in which data has not been entered as the user needs to either create an account or log in. Another circumstance is when the user logs in, the user is greeted with a pie chart with information about different categories and a Dropdown menu with different options.

Main Screen/Pie Chart

The Main screen of the program has a pie chart with the following categories: Personal, Work, Home, and Other. In the top right of the screen there will be a dropdown menu that has different options: Currency Exchange, as well as Add Extra Expenses.

Currency Exchange

This window, when clicked on from the dropdown menu, will display Various kinds of currencies the user can look towards whenever they want to deal with converting cash along the travel, for example.

Add Extra Expense

This window, when clicked on from the dropdown menu, will show the user's current values and a prompt for the user to choose which expense they would like to add. Upon request, the user will need to add a certain amount, and after inputting the amount, the user's pie chart will be updated along with the total amounts from the categories.

Backend

The backend uses mappings to update the database based on information sent to the given mappings' URLs. GET, POST, PUT, PATCH, and DELETE methods are used in our three controllers for different purposes. These purposes are described for each table below:

Users

For users we use GET requests to either retrieve all users, retrieve users of a specific type (regular, guest, premium), or retrieve a single user. Our login, login guest, and signup methods are also in this controller, both of which are PUT requests that return the unique ID of a user to the frontend. Once logged in, the proceeding requests are made with that unique ID in the path, essentially enabling us to set up our user account logic. There are PUT and PATCH methods to update a user's credentials. The PUT method can update all credentials, while the PATCH method only updates the income of the user. We also have a GET request to generate financial reports, which returns a

double value of the users income minus their expenses. Finally, we have a DELETE method to subtract users from the database.

Expenses

Users have one set of expenses, therefore users and expenses have a one-to-one relationship. In our expenses controller, we have a GET method to retrieve the expenses of a user, and a POST method to create a set of expenses for a user. In our app, expenses are primarily collected through bank statement uploads by our users, however we still want users to be able to enter extra expenses that accumulate day to day. For this, we have a PUT method for adding extra expenses to users. Lastly, we have a DELETE method to remove all the expenses for a user. All expense methods are called with a user ID in the path, allowing us to link expenses to users.

Cards

A user can have multiple cards, meaning that users and cards have a one-to-many relationship. We have GET methods to retrieve all cards of all users, all cards of a specific user, a selected card of a user, and the default card of a user. Credit cards are used in our app to upgrade from a regular user to a premium user. Each time a new card is created with our createCard POST request, that card is automatically set as the default card of the user. The default card is used in our autofill card feature when users are renewing their premium status. We also have a PUT method to update the information of a user's card, and a DELETE method to subtract a card from the user. If the default card is deleted, the user's oldest remaining card will be set to the default card, however this functionality will be changed such that the default card cannot be deleted in the first place. All user card methods are called with a user ID in the path, allowing us to link cards to users.

PUT THE TABLE RELATIONSHIPS DIAGRAM on this fourth page! (Create the picture using MySQLWorkbench)