Design Document for CyFinance

Group MS3_6

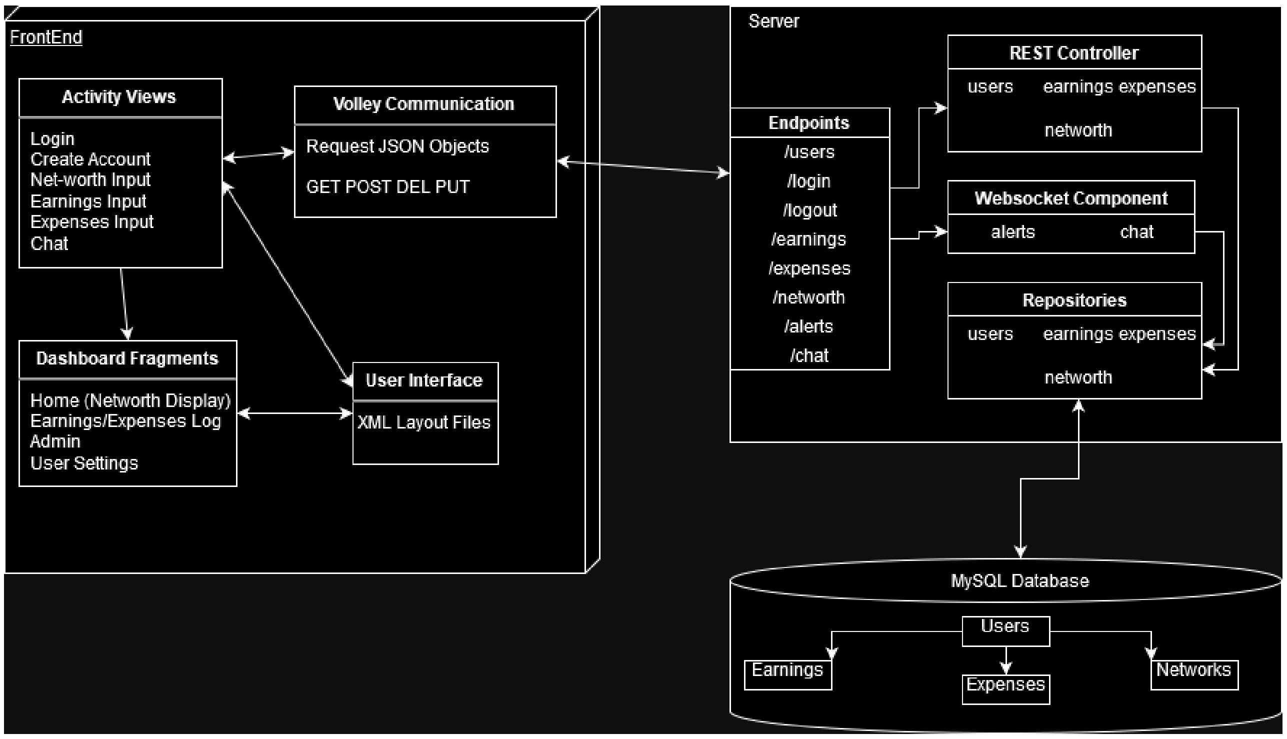
Casper: 25%

Hussein: 25%

James: 25%

Mohamed: 25%

Block Diagram



Frontend

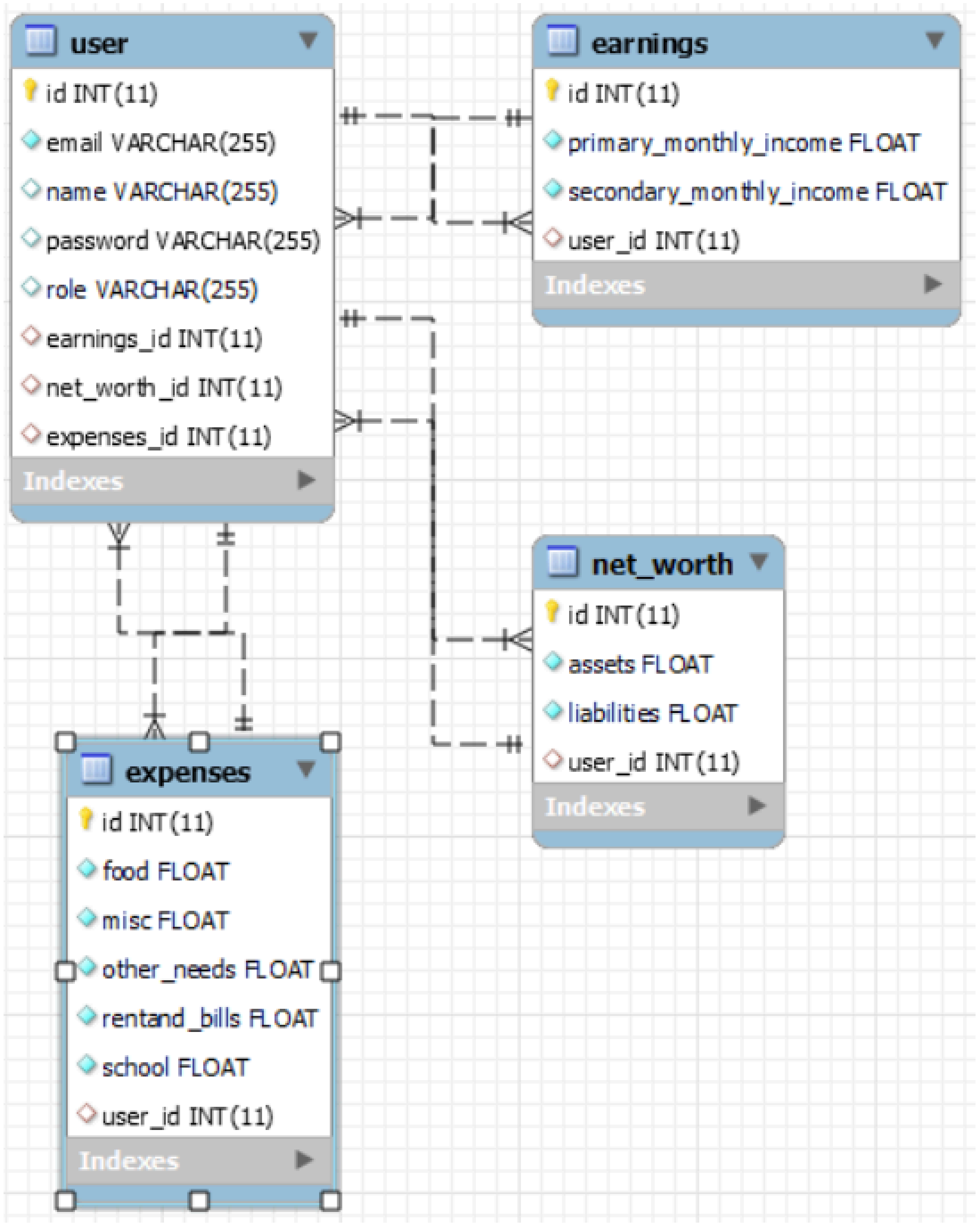
The actor is able to view the UI of the app and interact with it using the provided buttons. The app has many screens that represent whatever view the user picks to go to. Keep in mind that the connection between the frontend and backend that we have is pivotal to allow the user to interact with our application. For example, if we look at the main page, we see 3 options: logging in, signing up, or using the chat feature. Within the main activity page and the login pages we're basically using deserialization (technical term for parsing XML and JSON objects) to communicate with the backend server. If we look at the SignupActivity and the NetWorthActivity: we implemented a volley library to make HTTP requests(we make post requests in both of them). The importance of the file: VolleySingelton is in the fact that it manages Volley-related objects in Android Studio. Basically VolleySingleton gives the singleton for RequestQueue, what that does is ensure the queue isn't attached to an activity and that helps to prevent leaks. To be more exact, the following line of code in signup activity within the postRequest adds the JsonObjectRequest to the volley request queue:
`VolleySingleton.getInstance(getApplicationContext()).addToRequestQueue(createUser);`. In the case of NetworthActivity, the following line within the postRequest as well does the same exact thing like the one in SingupAcitivity:
`VolleySingleton.getInstance(getApplicationContext()).addToRequestQueue(createUser);`. On the other hand, we used websockets for the chat and notifications features. In the case of chat, the String url connected between the frontend and the backend. The java file: ChatConnect aims to handle the connection to the chat server, it also has an XML design that's the intermediate page between what the user sees and when he goes to the chat feature. Moreover, we used the architecture components for Notifications: these Notifications related files aim to display a simple text in the Notifications Fragment using ViewModel. Notifications are on the admin page: the admin sends critical push notifications to users.

One of the most important files is MainActivity.java, which manages the main navigation of the application. All of our pages have an adequate xml design that's connected to the java frontend code using this format: `private Button sendBtn,sendBtn = (Button) findViewById(R.id.sendBtn);` where the string is connected to the id of the xml associated item(Button, Textview or Edittext).

Backend

Our backend uses Spring boot to handle REST API calls with several controllers. The user controller manages the properties of a user as well as letting them login and authenticating their access to data. This controller also sets cookies for a user so that the application can track whether they are logged in or if they have access to the requested service. In addition to the user controller, the net worth, earnings, and expenses controllers all save the provided information in a MySQL database using JPA repositories to manage SQL queries. On top of this, Spring Security is planned to be added for securing access to the API and all the information it holds. Aside from REST controllers, there are also websocket components. The websocket components are used for real-time data transfer in the alert and chat systems of the application. These websocket components also communicate with the JPA repositories for authentication, authorization, and storage of data.

RELATIONSHIPS DIAGRAM



user
id INT(11)
email VARCHAR(255)
name VARCHAR(255)
password VARCHAR(255)
role VARCHAR(255)
earnings_id INT(11)
net_worth_id INT(11)
expenses_id INT(11)
Indexes
earnings
id INT(11)
primary_monthly_income FLOAT
secondary_monthly_income FLOAT
user_id INT(11)
Indexes
net_worth
id INT(11)
assets FLOAT
liabilities FLOAT
user_id INT(11)
Indexes
expenses
id INT(11)
food FLOAT
misc FLOAT
other_needs FLOAT
rentand_bills FLOAT
school FLOAT
user_id INT(11)
Indexes