Design Document for MartSmart

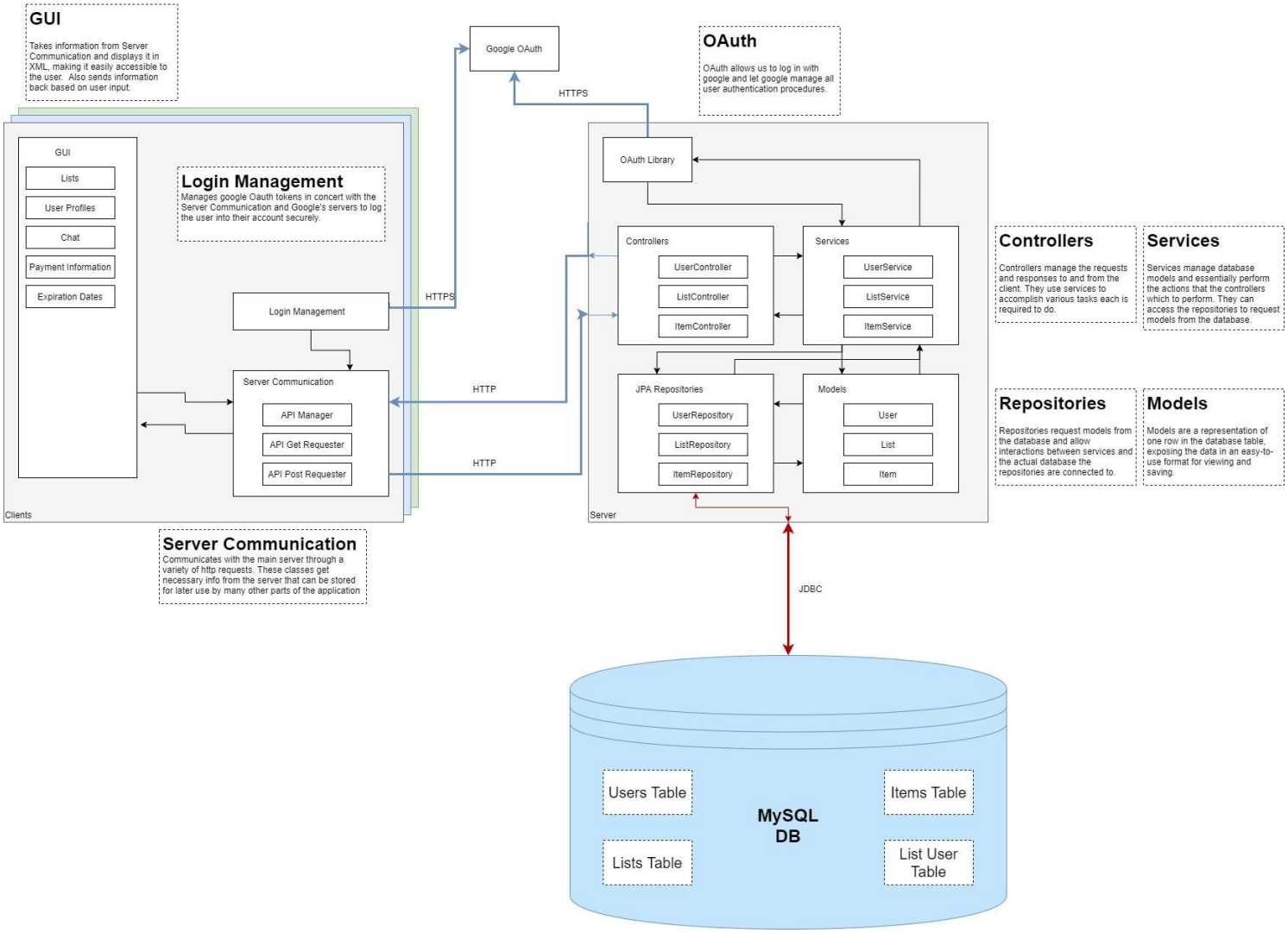
Group SR-05

Gregory Ling: 25% contribution

Fahmi Rafie: 25% contribution

Matt Karmelich: 25% contribution

Miles Hankins: 25% contribution



* Note: User, List, and Item are functional.

Overall Communication Flow
The flow starts with the login screen of the android app. When the screen loads, Android checks if there is a user present, and performs the OAuth flow described below. Once the user is logged in, one example of communication would be loading the lists screen. The Android Fragment for the lists screen asks the *APIManager* to send a request to the backend. It does and appends the id token described below to the headers of the request. A Controller on the backend receives the request, asks services to perform functions, which communicate through models with the repositories, and returns a response. The response is returned from the APIManager to the requesting fragment, and one request loop has been completed.

OAuth on Frontend
Our application allows the user to login with Google rather than depending on our own proprietary login system. To do so, the user must first choose a google account, which we then use to communicate with the server using the id token of the account generated by Google. This id token is sent to the backend server, which then also communicates with Google to verify the authenticity of the token from the frontend. After that process is complete, the application can use the authenticated id token in requests to communicate with the server and receive user specific information securely.

OAuth on Backend
The login system in our project is rather complex. After the frontend has logged in and has acquired the JWT id token from google, it will pass that token to the server in the header of every http request. This JWT token contains information about the user and a signature from Google to validate the data. During each request, the server takes that token, confirms the signature is valid, and uses this data to login the user. Since this token contains a name and email, the token's data can be used to create a user instance if the user does not currently exist in the database. The user model is then requested from the user repository and passed to the appropriate controller after verification. This removes the need for a dedicated sign up page, and only a single sign in button is required on the frontend side. If the id was invalid, a *UserAuthenticationException* is thrown with a message for the frontend to display detailing what failed during the login process.

*Note: User, List, and Item are functional. Created using Jetbrains' DataGrip