Design Document for Film Finder Project

By Group 2_UG_2

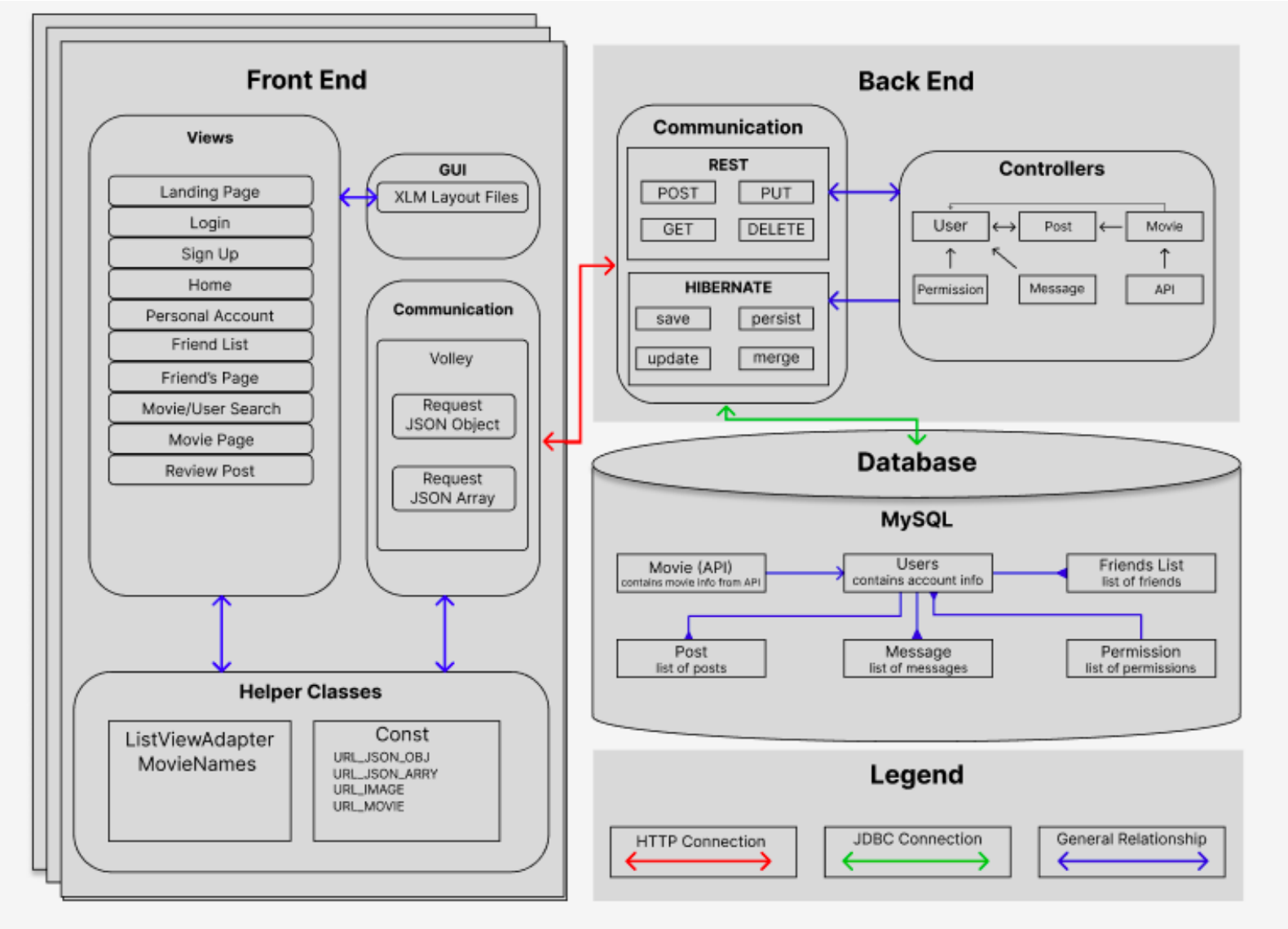
Haris Khan

Sabrina Francis

Sonali Malhotra

Wonjun Choi

BLOCK DIAGRAM



Frontend (currently implemented)

Sign Up (User)

- Sign Up generates a page with the following elements:
    - EditText: Full Name
    - EditText: Email
    - EditText: Username
    - EditText: Password
    - Button: Sign Up
    - Button: Back Button
- Upon clicking the sign up button, if none of the values are empty, the values of Full Name, Email, Username, and Password are sent as a POST request to the server.

Login (User)

- Login generates a page with the following elements:
    - EditText: Username
    - EditText: Password
    - Button: Login
    - Button: back button
- Login Screen takes user input of their username and password and authenticates it by sending a GET request to the server. If it is valid, it will send them to the home screen where they are logged in as that user.

Personal Account (User)

- Personal Account generates a page with the following elements:
    - ImageView: userPic
    - EditText: Username
    - Button: Friends List
    - TableLayout:
        - ImageView: userPic
        - textView: post content
        - ImageView: star rating out of 5
- Loading this page after clicking the button from the home page completes a GET request from the server into the specific user's table to get some of their information, along with their dynamic 'Post' table.

Friends List (User)

- Friends List generates a page with the following elements:
    - TableLayout:
        - ImageView: friends' userPics
        - Button:
            - TextView: friends' usernames
- After loading this page by clicking on the 'Friends List' button on the Personal Account page, it will complete a GET array request from the server to retrieve the dynamic array of the specific user's friends.

Friend's Page (User)

- Friend's Page generates a page with the following elements:
    - ImageView: friend's userPic
    - EditText: friend's Username
    - TableLayout:
        - ImageView: friend's userPic
        - textView: friend's post content
        - ImageView: friend's star rating out of 5
- After clicking on one of the friend's buttons from the Friends List page, it will load that specific friend's page and will make a GET request to get that friend's information.

Post Page (User)

- Post Page generates page with the following elements:
    - TextView: "Rating:"
    - TextView: "Review:"
    - EditText: enter out of 5
    - EditText: enter here
    - Button:
        - TextView: Back
        - TextView: Post
- After clicking the post button it will make a POST request to the server

Movie Page (User)

- Movie Page generates a page with the following elements:
    - TextView: movieTitle
    - TextView: movieYear
    - TextView: movieGenre
    - TextView: movieOverallRating
    - TextView: movieActor
    - TextView: movieDirector

        - Button:
            - TextView: Make Post
- After clicking the make post button it will take you to the make post page

Backend

Communication

The backend utilizes mappings to alter information in the database. The information sent to the mappings' URLs determines what will happen to the information in the database. These include:

- Get: To read/view information from the database
- Post: To create information in the database
- Put: To update information in the database
- Delete: To delete information from the database

Controllers

The controllers have mappings that enable communication between the frontend and the database. These mappings consist of the following:

- User: Contains all of the above mappings to create a user with a one-to-many relationship with posts, a one-to-many relationship with messages, and a many-to-one relationship with permissions. Form a GetMapping for the posts, messages, and permissions associated with the user.
- Movie: API to create a MovieService that gets movie information from MovieDB. Create a getMapping that can use MovieService to get movie information.
- Post: Contains all the mappings to create a post that contains a many-to-one relationship with the user. Creates a JsonProperty to store information about the movie_id, but no relation because it doesn't store any information about the movie.
- Message: Contains all the mappings to create messages with a many-to-one relationship with the user.
- Permission: Contains all the mappings to create Permission with a one-to-many relationship with the User.

DATABASE SCHEMA

Description of Movie Database: Not storing movie information, but fetching it on demand via the API. Therefore, there is no need to create and store objects, and as a result, the relationship between objects is also lost, so only the movie data is created separately.