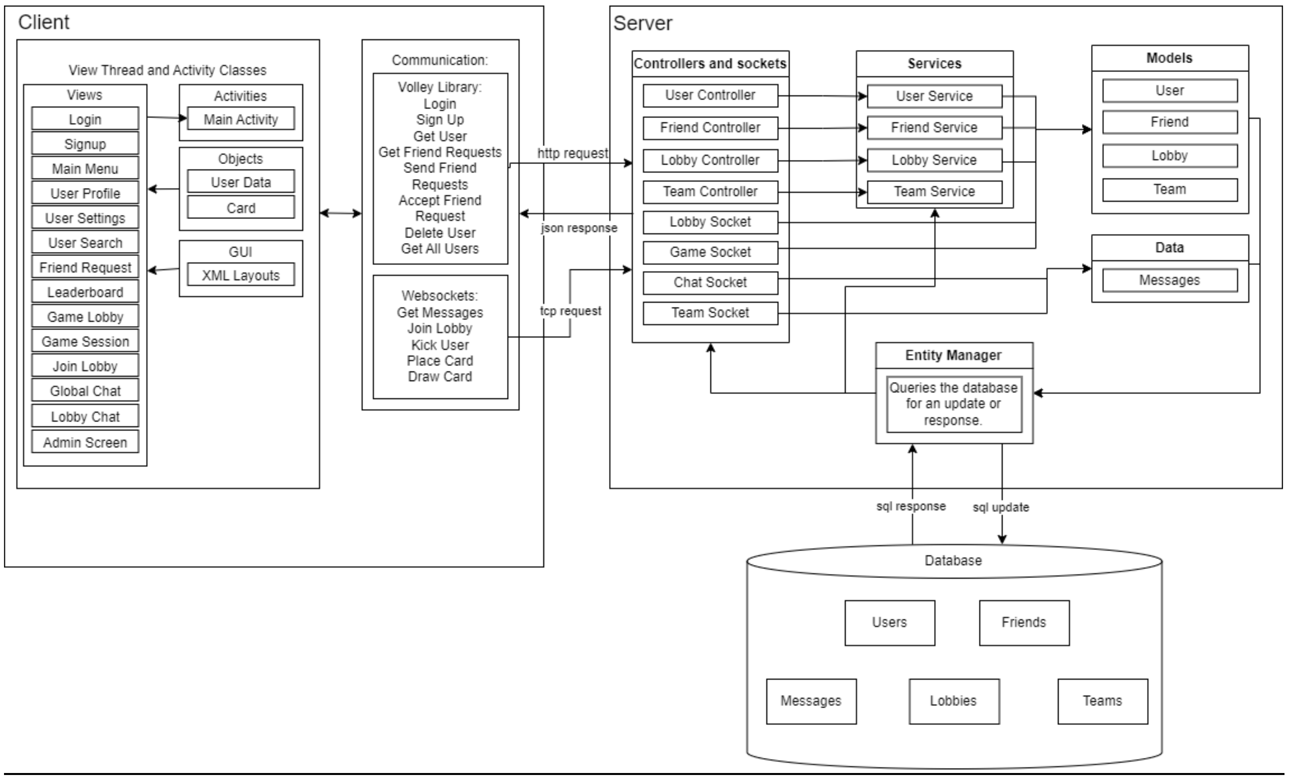
Design Document for UNUS

Group 2_an_5

Sachin Patel, Abram Demo, Isaac Blandin

Block Diagram:



Model

Our application uses the Spring Boot framework for the backend along with MySQL for database storage. Spring Boot provides easy integration with the front end for communication using endpoint parameters that automatically get converted to java objects and types from JSON. It also provides easy management of database tables using Hibernate. With Hibernate tables are created and altered when the application is run. On the frontend, to communicate with the remote server, there is a separate thread being used to run the Volley library which runs http requests and listens for http responses asynchronously from the rest of the application code. Due to this, the UI is generally unaffected by the speed of responses from the server. For the communications which require bi-directionality, web sockets are used. These are managed using Java-WebSocket on the frontend.

Controller

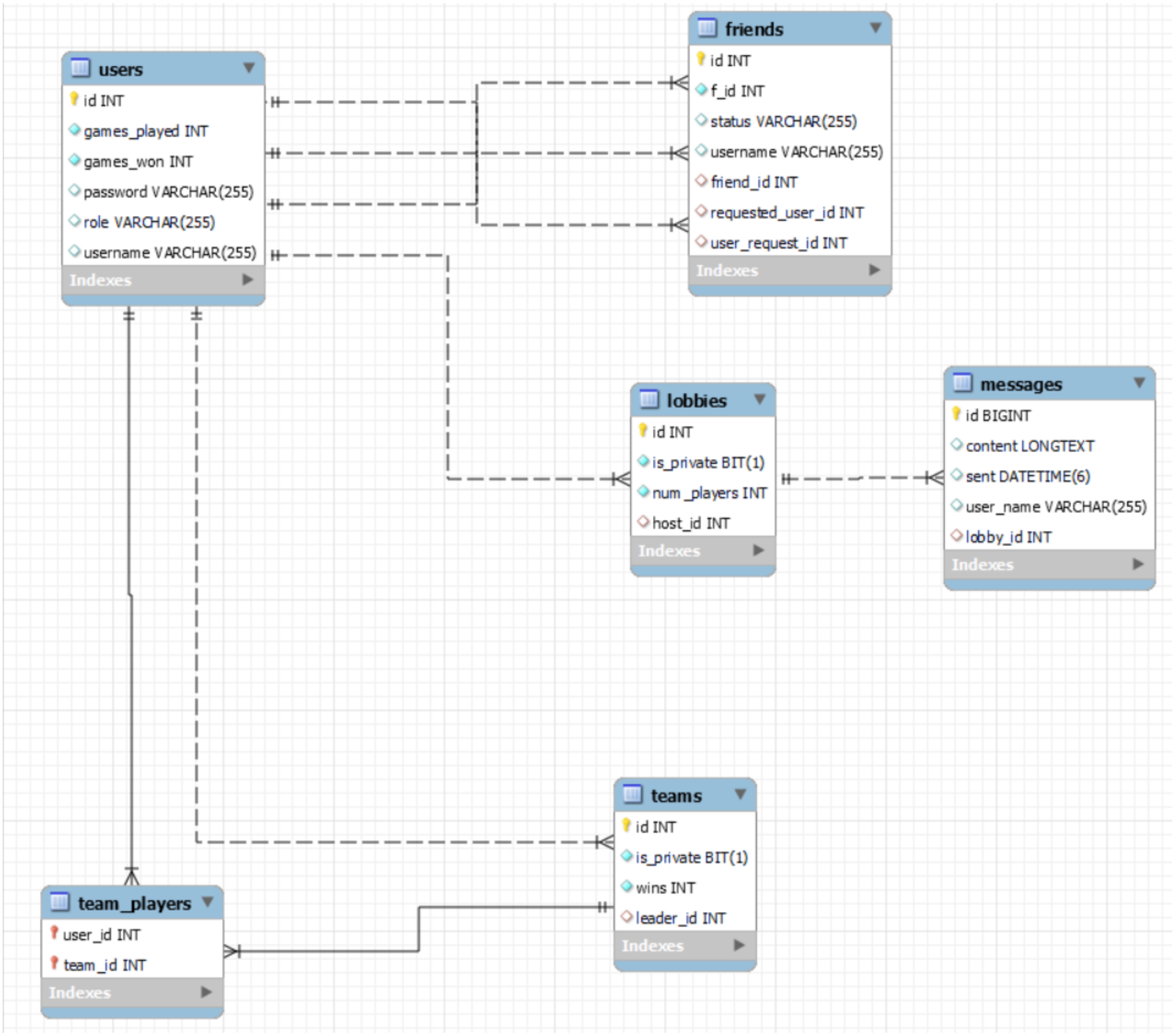
For http requests, the frontend creates requests to access or modify data from the backend for multiple reasons. Requests are created for tasks like login, account creation, accessing friends' data, getting game information, and establishing new friends. On the backend, these requests are used to access, modify, delete, or create new rows in the database tables along with sending a response if necessary.

For web socket tcp requests, the backend declares an endpoint that a user can connect to and stores the user session. Messages sent from the front end to this endpoint will broadcast to other user sessions. We can split up chats (like for teams and lobbies) by storing a map with the lobby and a list of user sessions and only broadcast the messages to sessions associated with that lobby. We can also update any UI based on what response comes from the web socket; for example, we can update the game UI based on what card a user has played.

View

Our application's frontend fully runs on android clients. Due to this, our application's interface is created visually using xml layout files and Java classes/objects to control the logic. Due to the nature of our project being a multiplayer game, the application relies heavily on data retrieved from the server to update the views. The chat, game lobby, and game play all rely on web socket data for the most up to date information on those respective use-cases. Other views are primarily dedicated to social information and account management such as user data and friend information.

Table Relationship Diagram:


users
id INT
games_played INT
games_won INT
password VARCHAR(255)
role VARCHAR(255)
username VARCHAR(255)
Indexes
friends
id INT
f_id INT
status VARCHAR(255)
username VARCHAR(255)
friend_id INT
requested_user_id INT
user_request_id INT
Indexes
lobbies
id INT
is_private BIT(1)
num_players INT
host_id INT
Indexes
messages
id BIGINT
content LONGTEXT
sent DATETIME(6)
user_name VARCHAR(255)
lobby_id INT
Indexes
teams
id INT
is_private BIT(1)
wins INT
leader_id INT
Indexes
team_players
user_id INT
team_id INT
Indexes