Design Document for Trivia Game

Group 3_UG_1

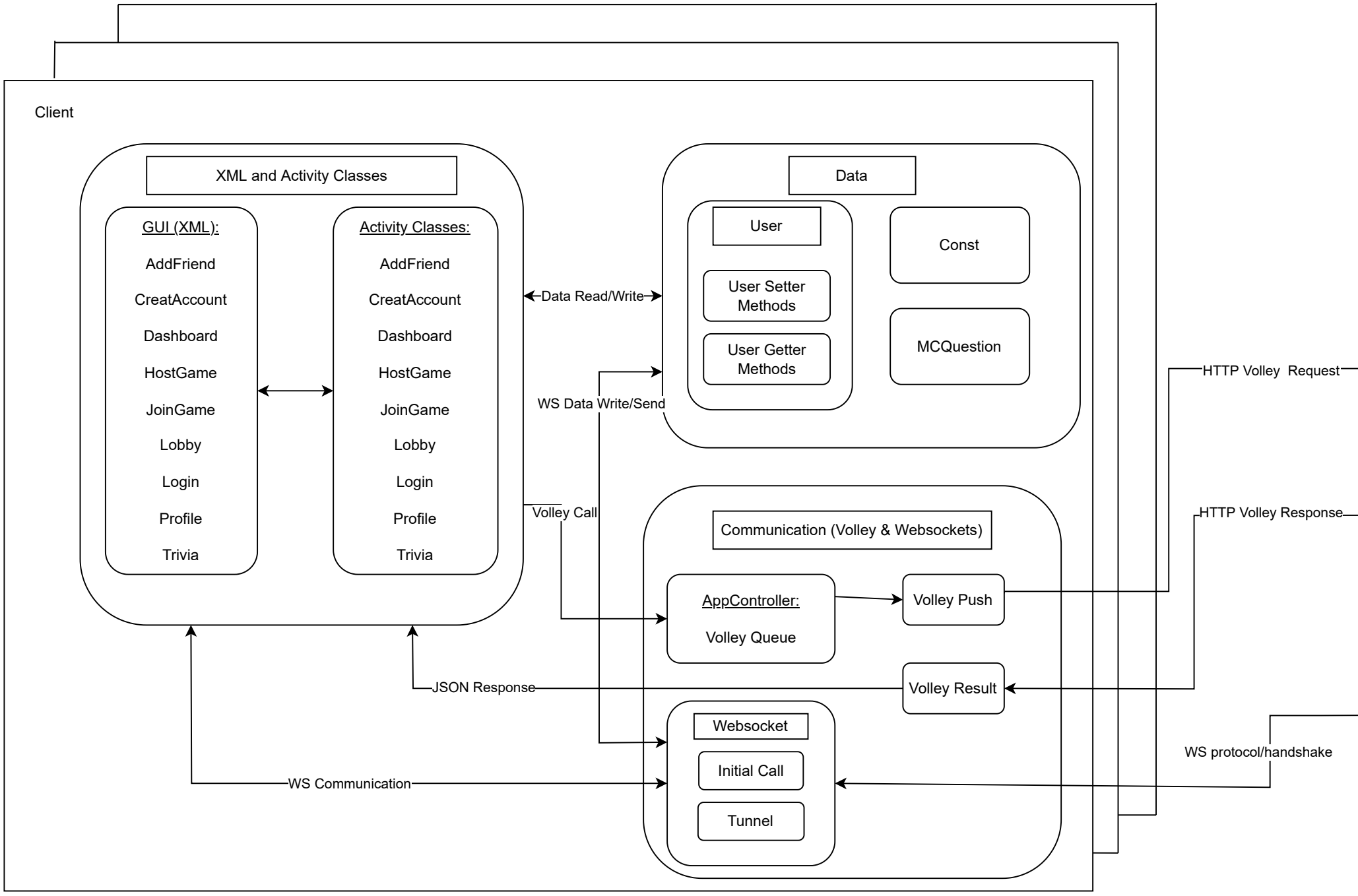
Mason: 25%

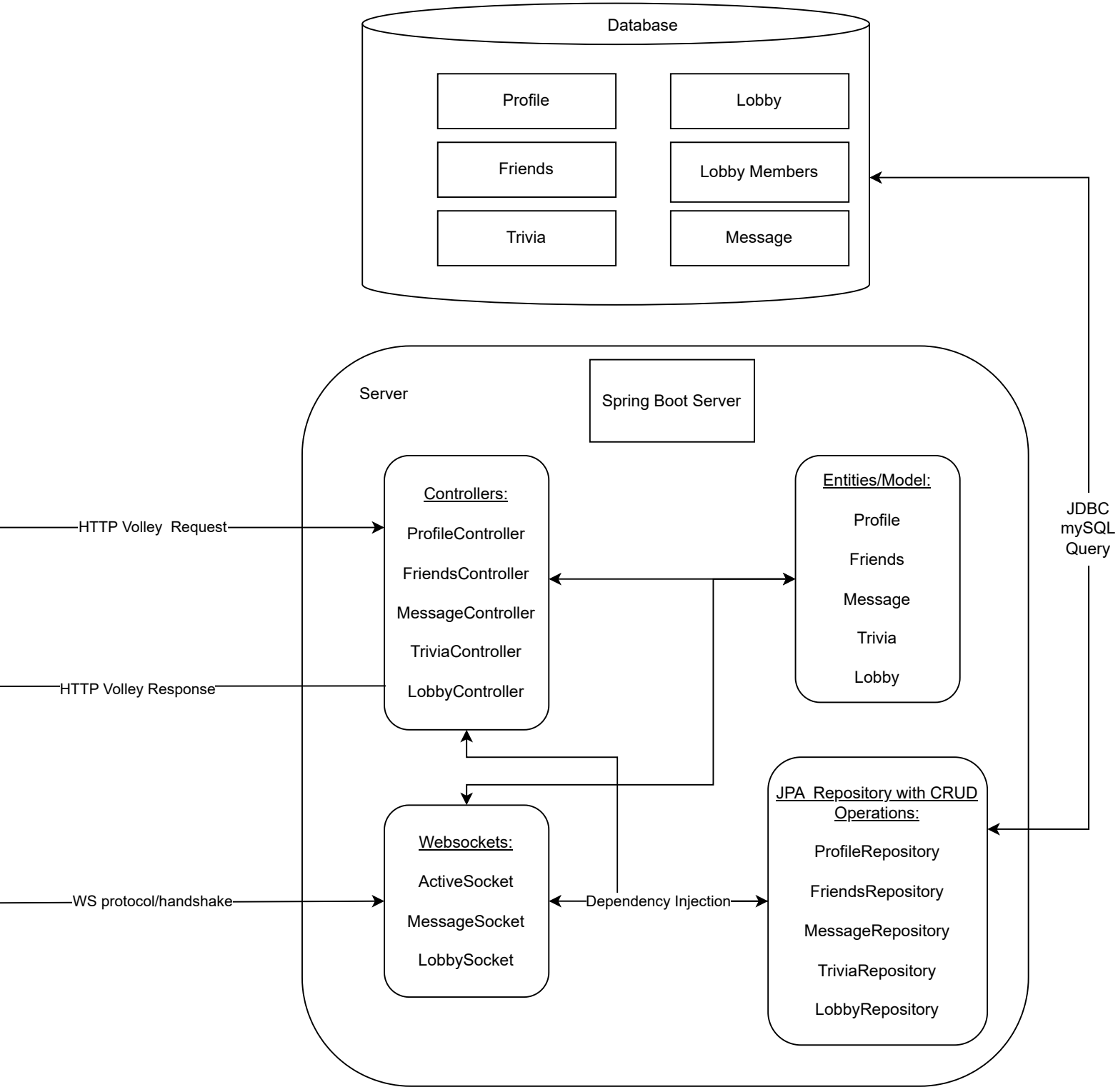
Sid: 25%

Lance: 25%

Isaac 25%

Block Diagram
Team: 3_ug_1
Project Name: Trivia




Database
Profile
Lobby
Friends
Lobby Members
Trivia
Message
Server
Spring Boot Server
Controllers:
ProfileController
FriendsController
MessageController
TriviaController
LobbyController
Entities/Model:
Profile
Friends
Message
Trivia
Lobby
JDBC
mySQL
Query
HTTP Volley Request
HTTP Volley Response
Websockets:
ActiveSocket
MessageSocket
LobbySocket
WS protocol/handshake
Dependency Injection
JPA Repository with CRUD Operations:
ProfileRepository
FriendsRepository
MessageRepository
TriviaRepository
LobbyRepository

Executive Design Description

Our application serves as a space where users can create custom lobbies where they can invite their friends to participate in live trivia. The user is able to create a lobby and define aspects such as the number of people who are allowed to join, trivia theme, and trivia type. For users joining, they are able to select what lobby to join out of a list of all active lobbies. Following a completed trivia game, the user is shown their stats, and then taken back to the dashboard where they are able to select what they wish to do next.

Android GUI

The actor will interact with our application through a series of android activities. These activities will allow the user to navigate their way through creating a lobby where they can participate in trivia with their friends. The actor is also able to join a lobby if they have the host-defined password. The host then activates the trivia, and all the participants within the lobby will be served all the trivia questions by the GUI. During this time, the user also has the ability to chat with friends via a live websocket. After a round of trivia is completed, the user is taken back to the dashboard. The actor is also able to view their profile to see more information about themselves, add/remove friends, and view the chat logs between your friend(s) and yourself.

Android DATA/CONTROLLER

Upon the actor logging in or creating an account, our android User class will collect all the data the application needs for the current session. This information will be referenced as they navigate through various activities, and will be updated only after a trivia game, after logging in, or when the user has closed the app. When the user activates a JSON request, the volley call is pushed through the app controller, then to the backend. When calls are returned, they will either change something on display to the actor, and/or update the User data class.

Server

We are using Spring Boot with Apache Tomcat as our server. Our server contains the Controller, Entity/Model, Repository and Websockets. The Controller allows for the frontend to make calls to the server to send or retrieve data. The Entity/Model are the persistence objects stored as a record in the database. The Repository communicates between the Database and the Server to send or receive data. Websockets also utilize JPA Repositories to make changes to the database. In addition, websockets keep track of open websocket Sessions, allowing a user to have an open line of communication between the client and server. An example of a call from the frontend to make a trivia question that would be sent to the controller which makes a new Entity/Model. Which is then sent to the database using a JPA Repository to save the question.

Tables and Fields