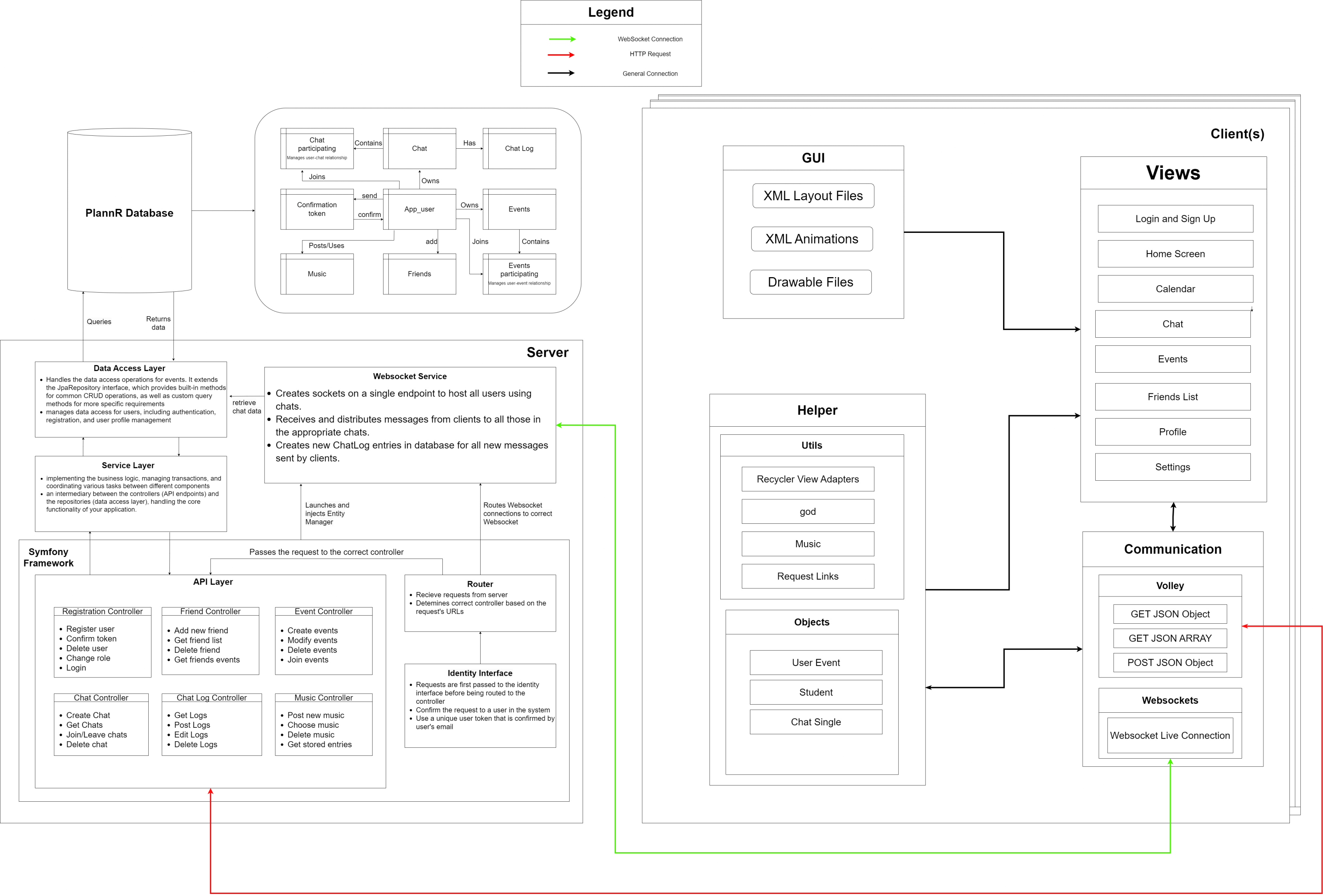
Design Document For MeetR

Group 2_TZ_9

Tasman Grinnell	50%
Phuong Nguyen	50%
Zachary Rapoza	50%
Spencer Thiele	50%



Data Access Layer

- Handles the data access operations for events. It extends the JpaRepository interface, which provides built-in methods for common CRUD operations, as well as custom query methods for more specific requirements
- manages data access for users, including authentication, registration, and user profile management

Websocket Service

- Creates sockets on a single endpoint to host all users using chats.
- Receives and distributes messages from clients to all those in the appropriate chats.
- Creates new ChatLog entries in database for all new messages sent by clients.

Service Layer

- implementing the business logic, managing transactions, and coordinating various tasks between different components
- an intermediary between the controllers (API endpoints) and the repositories (data access layer), handling the core functionality of your application.

API Layer

Registration Controller
- Register user
- Confirm token
- Delete user
- Change role
- Login

Friend Controller
- Add new friend
- Get friend list
- Delete friend
- Get friends events

Event Controller
- Create events
- Modify events
- Delete events
- Join events

Chat Controller
- Create Chat
- Get Chats
- Join/Leave chats
- Delete chat

Chat Log Controller
- Get Logs
- Post Logs
- Edit Logs
- Delete Logs

Music Controller
- Post new music
- Choose music
- Delete music
- Get stored entries

Router

- Recieve requests from server
- Detemines correct controller based on the request's URLs

Identity Interface

- Requests are first passed to the identity interface before being routed to the controller
- Confirm the request to a user in the system
- Use a unique user token that is confirmed by user's email

Separate Link for Diagram

Design Description

User:

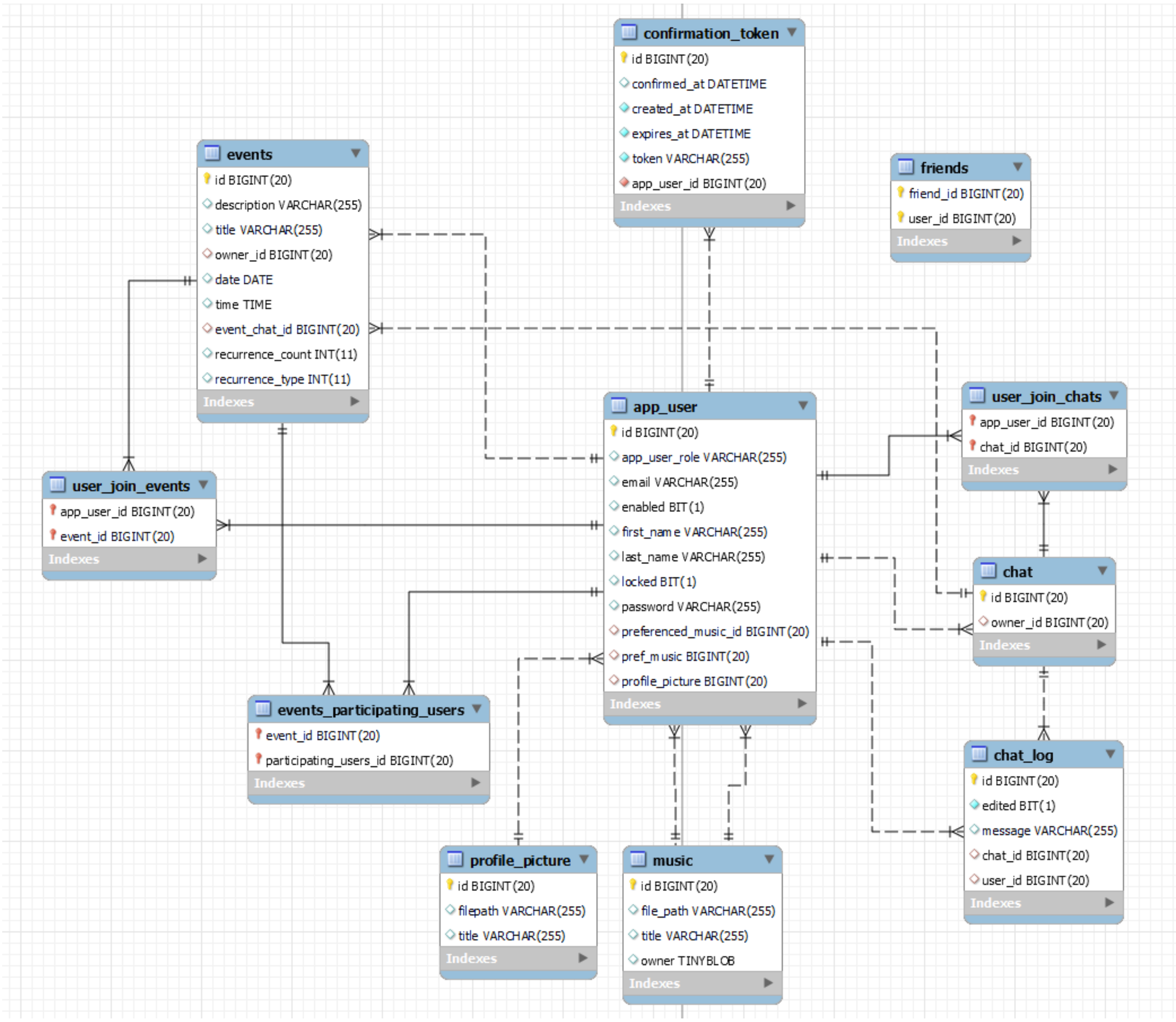
The user interacts with the views in our application. The primary way that users navigate between the main views is using the navigation bar. The views handle user input and display information to the user, occasionally getting or sending information to the server. This is done via the use of Volley requests and websockets. One way that Volley requests are used is Login, which takes the email and password that the user enters and sends it to the server using a POST request. The server then returns whether or not the user is registered. Websockets are used for client to client connections especially for the chat application that allows users to interact with each other and discuss plans in real time.

Server:

Managing requests and hosting our backend code, the server provides a method of communication between clients and the database, using Apache to host the server. The backbone of the communication consists of various REST controllers, which are able to handle HTTP requests that are made using Volley on client machines. These controllers use various logic to retrieve information from the database using Hibernate and return information to be used client-side. To host the communication with the database, we use CRUD/Jpa Repositories in the backend to allow our REST controllers to be interpreted to SQL queries using Hibernate. The queries facilitate various requests to be processed and stored, such as GET, POST, PUT, and DELETE requests that users will need to make throughout use of the app.

Database:

The database hosts entries for each entity used in the project, including various data fields for the chats, events, music, profile pictures, and app users. These data fields include information such as when and where events are, chat logs that are sent in each of the chats, music titles and filepaths that are hosting each individual song, and many more. Furthermore, the nature of our project contains many complex relationships between the various aspects, including Chats, Events, and Music. Using SQL tables to maintain and store the relationships is key, and having the intermediary tables connecting users joining and participating with chats and events is very important for opening and managing the events and chats during frontend-backend communication.



confirmation_token
id BIGINT(20)
confirmed_at DATETIME
created_at DATETIME
expires_at DATETIME
token VARCHAR(255)
app_user_id BIGINT(20)
Indexes

friends
friend_id BIGINT(20)
user_id BIGINT(20)
Indexes

events
id BIGINT(20)
description VARCHAR(255)
title VARCHAR(255)
owner_id BIGINT(20)
date DATE
time TIME
event_chat_id BIGINT(20)
recurrence_count INT(11)
recurrence_type INT(11)
Indexes

app_user
id BIGINT(20)
app_user_role VARCHAR(255)
email VARCHAR(255)
enabled BIT(1)
first_name VARCHAR(255)
last_name VARCHAR(255)
locked BIT(1)
password VARCHAR(255)
preferenced_music_id BIGINT(20)
pref_music BIGINT(20)
profile_picture BIGINT(20)
Indexes

user_join_chats
app_user_id BIGINT(20)
chat_id BIGINT(20)
Indexes

user_join_events
app_user_id BIGINT(20)
event_id BIGINT(20)
Indexes

chat
id BIGINT(20)
owner_id BIGINT(20)
Indexes

events_participating_users
event_id BIGINT(20)
participating_users_id BIGINT(20)
Indexes

chat_log
id BIGINT(20)
edited BIT(1)
message VARCHAR(255)
chat_id BIGINT(20)
user_id BIGINT(20)
Indexes

profile_picture
id BIGINT(20)
filepath VARCHAR(255)
title VARCHAR(255)
Indexes

music
id BIGINT(20)
file_path VARCHAR(255)
title VARCHAR(255)
owner TINYBLOB
Indexes