

---

## Block Diagram - Design Document for iMP3

---

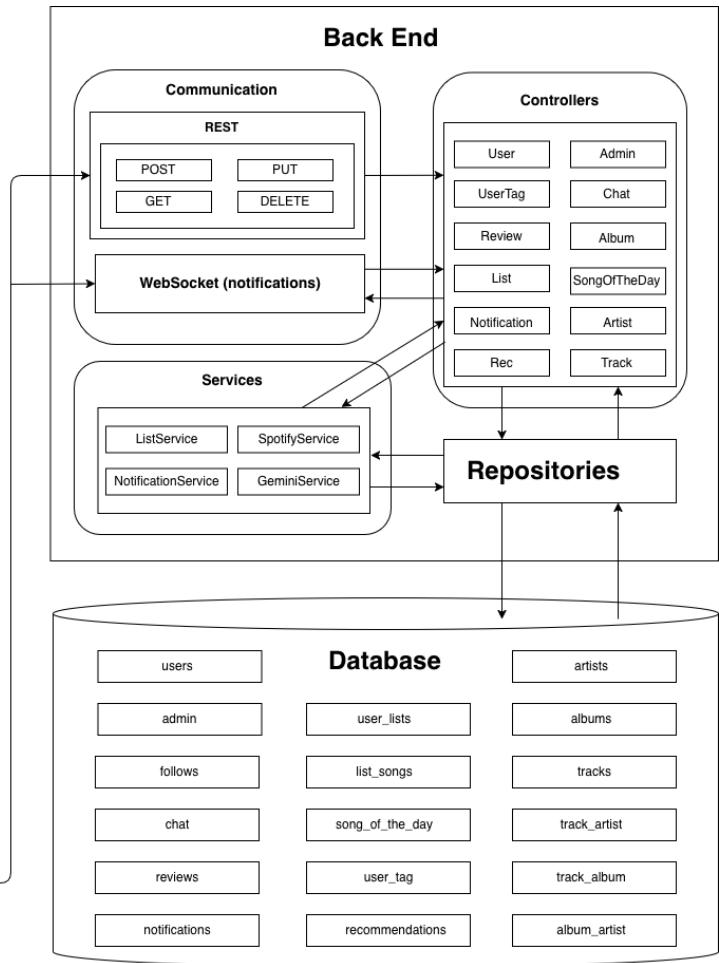
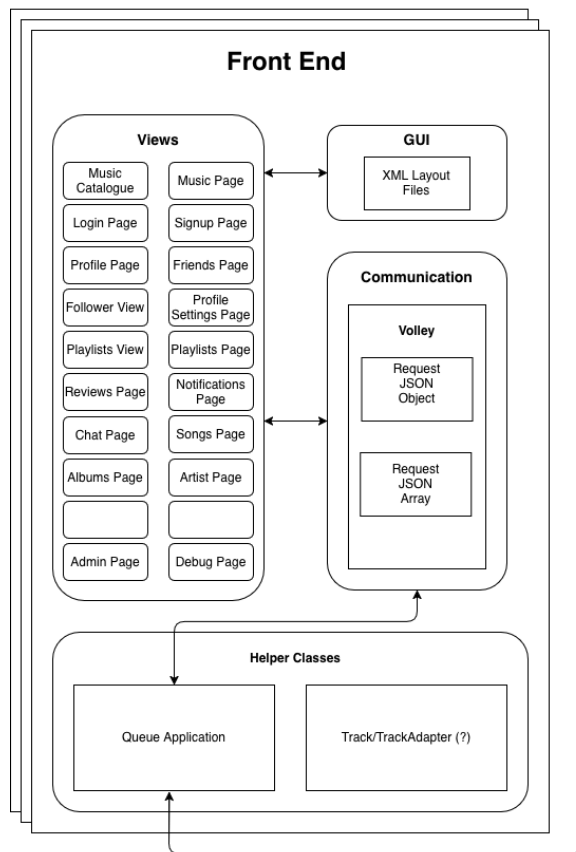
Group 2\_jubair\_2

Sara Theriault - contribution: list of tables & fields generated by MySQL Workshop, Block Diagram

Penny Mayer - contribution: Block Diagram, Backend Writeup

Cayden Olsen - contribution: Frontend Writeup

Graysen Schwaller - contribution: Block Diagram



## **Frontend:**

### **RecyclerViews:**

- Multiple pages, such as notifications and chat use recycler views to create changeable lists
  - Uses models and adapters to create new elements
  - Notifications and Chat both use websockets to live update the user with information. This comes into play with the recycler views.
  - Upon “handleIncomingWebsocketMessage” the model for a chat or notification is called
  - This creates a ‘chain’ of events that lead to the adapter being called, filling the ui elements with the incoming message, updating the list, and notifying a data set change to update the UI in live time

### **DebugPage:**

- The debug page generates these elements:
  - Edit text so a developer can manually change the userId variable on the frontend side to change account login (NOTE: this does not create an HTTP session ID)
  - Multiple buttons that allow the developer to easily access the main pages
- The debug page is not technically a main feature as it is mostly for the frontend devs to easily maneuver, change, and test the app without running through the main login flow multiple times

### **QueueApplication:**

- The helper class QueueApplication is not an interactive page
- The class is used to create, manage, and maintain the request queue
- Upon login, a HTTP session ID is created to pass to the backend to keep a collection of who is the logged in user
- All requests call the request queue to make sure the cookie doesn’t get lost and the backend requests function

## **Backend:**

The backend is written in SpringBoot, and handles the storage/retrieval of information in the database, accessed via mapped urls and JSON requests. Http POST, GET, PUT, and DELETE requests can fully manipulate all information stored in the database, provided the sender has an appropriate permission level

### **Database:**

The database is a collection of permanent information stored in MySQL tables. This includes user account information, user-generated content, and a catalog of music. All tables are keyed by a primary UID number, with additional information in descriptive columns as necessary.

#### **User Information**

Username, emails, passwords, personal information, photos, and a short biography.

#### **User Generated Content**

Users can review music in the catalog, on a 5-star scale and with a written description, and post those reviews publicly for other users to read and react to. Music ratings are aggregated into an average score, displayed alongside the relevant music in the catalog.

Users can also make and share playlists of music, which requires a one-to-many relationship between each playlist object and the music catalog. Users can follow each other, and receive notifications when new reviews or playlists are posted by those they follow. Mutual followers can also chat with each other in real time.

#### **Music Catalog**

The music catalog’s primary tables are Artists, Albums, and Tracks. Each contains information like name, art, dates released/active, genre, and so on. However, while albums are composed of tracks and artists have collections of albums, artists can also release single tracks with no album, and collaborations between artists can lead to tracks and albums with multiple artists. So there are relational tables between Artists-Albums, Artists-Tracks, and Albums-Tracks, each with a many-to-many relation.

