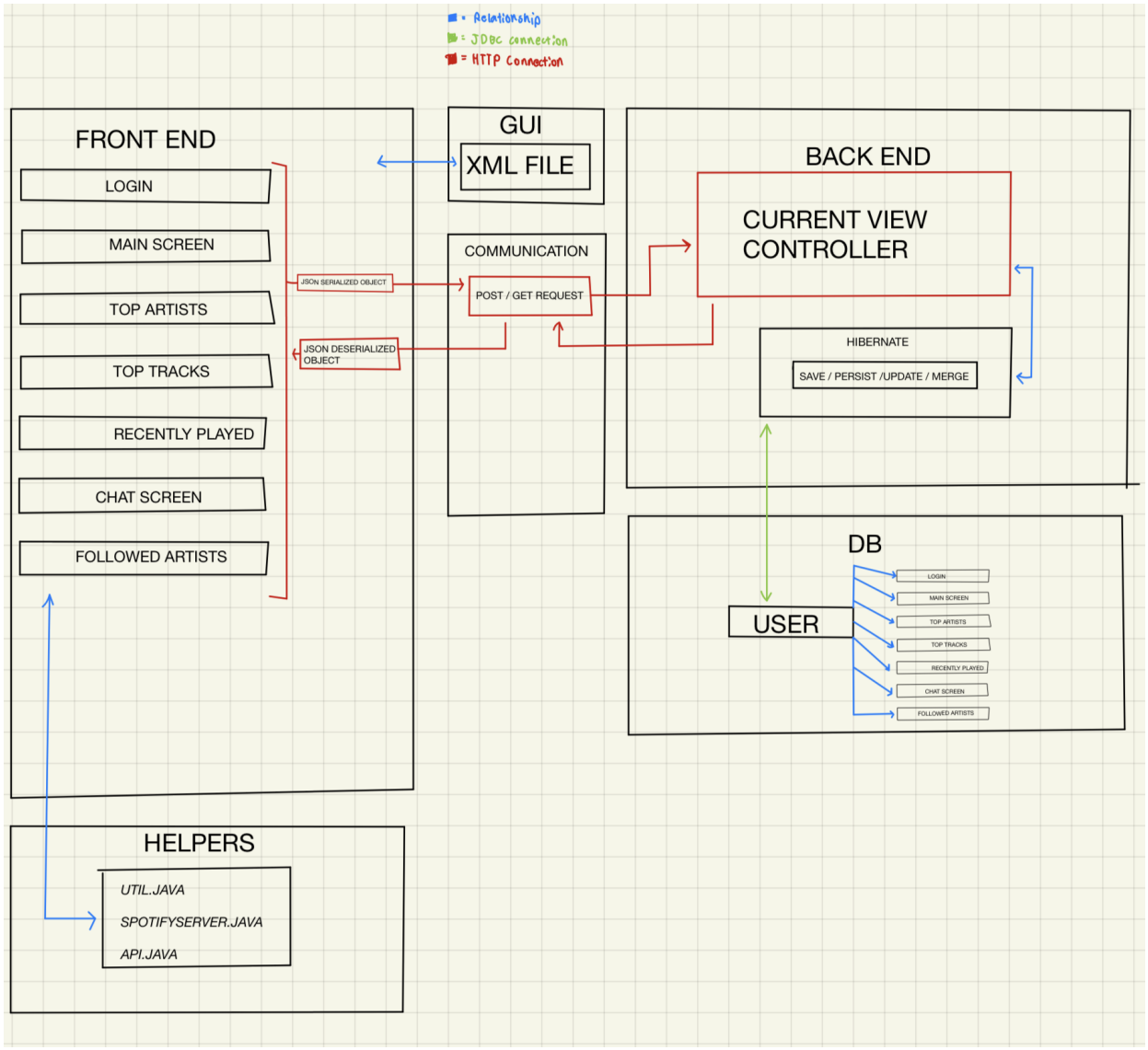
Design Document for QuickSC

Group 324

Ty Beresford: 25%
Sajan Patel: 25%
Tiffanie Fix: 25%
Jonathan Madden: 25%


= Relationship
= JDBC connection
= HTTP Connection
FRONT END
LOGIN
MAIN SCREEN
TOP ARTISTS
TOP TRACKS
RECENTLY PLAYED
CHAT SCREEN
FOLLOWED ARTISTS
GUI
XML FILE
COMMUNICATION
JSON SERIALIZED OBJECT
POST / GET REQUEST
JSON DESERIALIZED OBJECT
BACK END
CURRENT VIEW CONTROLLER
HIBERNATE
SAVE / PERSIST /UPDATE / MERGE
DB
USER
LOGIN
MAIN SCREEN
TOP ARTISTS
TOP TRACKS
RECENTLY PLAYED
CHAT SCREEN
FOLLOWED ARTISTS
HELPERS
UTIL.JAVA
SPOTIFYSERVER.JAVA
API.JAVA

Front End

The Front End is the first layer of the application, and it's where the user interacts with the app. It includes several features such as Login, Main Screen, Top Artists, Top Tracks, Recently Played, Chat Screen, and Followed Artists. Each feature represents a distinct function of the application, offering the user a range of options for interacting with the music service.

GUI

The Graphical User Interface (GUI) interacts directly with the front end, providing the visual elements that users interact with. It is likely defined by an XML file, which is standard practice in Android development for designing layouts in a declarative way. This allows for a separation of the presentation of the app from the logic that controls the user's interaction with it.

Communication

Communication between the front end and the back end is handled through POST and GET requests, likely involving HTTP protocols. This is an essential part of the architecture as it allows the app to send and receive data from the server. The JSON deserialized object notation suggests that the data being exchanged between the front and back ends is in JSON format, which is a lightweight data interchange format.

Back End

The Back End is represented by the Current View Controller, which is responsible for managing the views that the user sees at any given time. It receives data from the communication layer, processes it, and then updates the view accordingly. The use of the term "controller" implies a Model-View-Controller (MVC) architecture is being used, where the controller manages the flow of data between the model (data) and the view (UI).

Hibernate

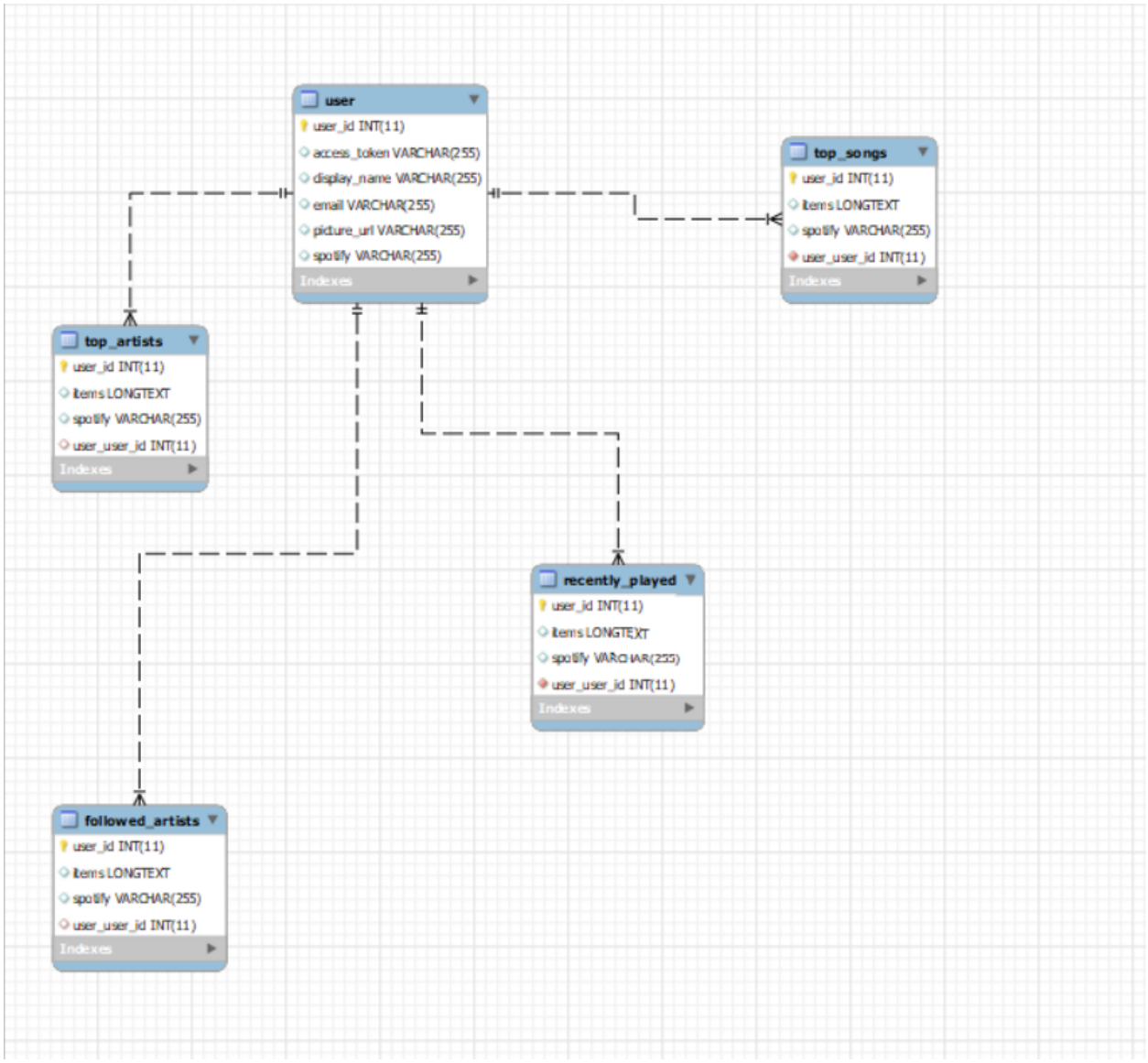
Hibernate is an Object-Relational Mapping (ORM) tool for Java, and it's being used here to manage database operations such as save, persist, update, and merge. It simplifies interactions with the database by allowing developers to work with objects rather than SQL queries, thereby abstracting the complexity of database manipulations.

DB

The database is shown with a structure that reflects the user's data: Login, Non-Token, Top Artists, Top Tracks, Recently Played, Chat Screen, and Followed Artists. This structure likely corresponds to tables within the database, which Hibernate interacts with. The database is central to the app, as it stores all the user-related data, preferences, and interaction history.

Helpers

Helpers, such as UTIL.java, SpotifyServer.java, and API.java, are utility classes that likely provide common functionality across the app. They could handle tasks such as formatting data, managing API calls, and other services that support the main operations of the app.


user
user_id INT(11)
access_token VARCHAR(255)
display_name VARCHAR(255)
email VARCHAR(255)
picture_url VARCHAR(255)
spotify VARCHAR(255)
Indexes
top_songs
user_id INT(11)
items LONGTEXT
spotify VARCHAR(255)
user_user_id INT(11)
Indexes
top_artists
user_id INT(11)
items LONGTEXT
spotify VARCHAR(255)
user_user_id INT(11)
Indexes
recently_played
user_id INT(11)
items LONGTEXT
spotify VARCHAR(255)
user_user_id INT(11)
Indexes
followed_artists
user_id INT(11)
items LONGTEXT
spotify VARCHAR(255)
user_user_id INT(11)
Indexes