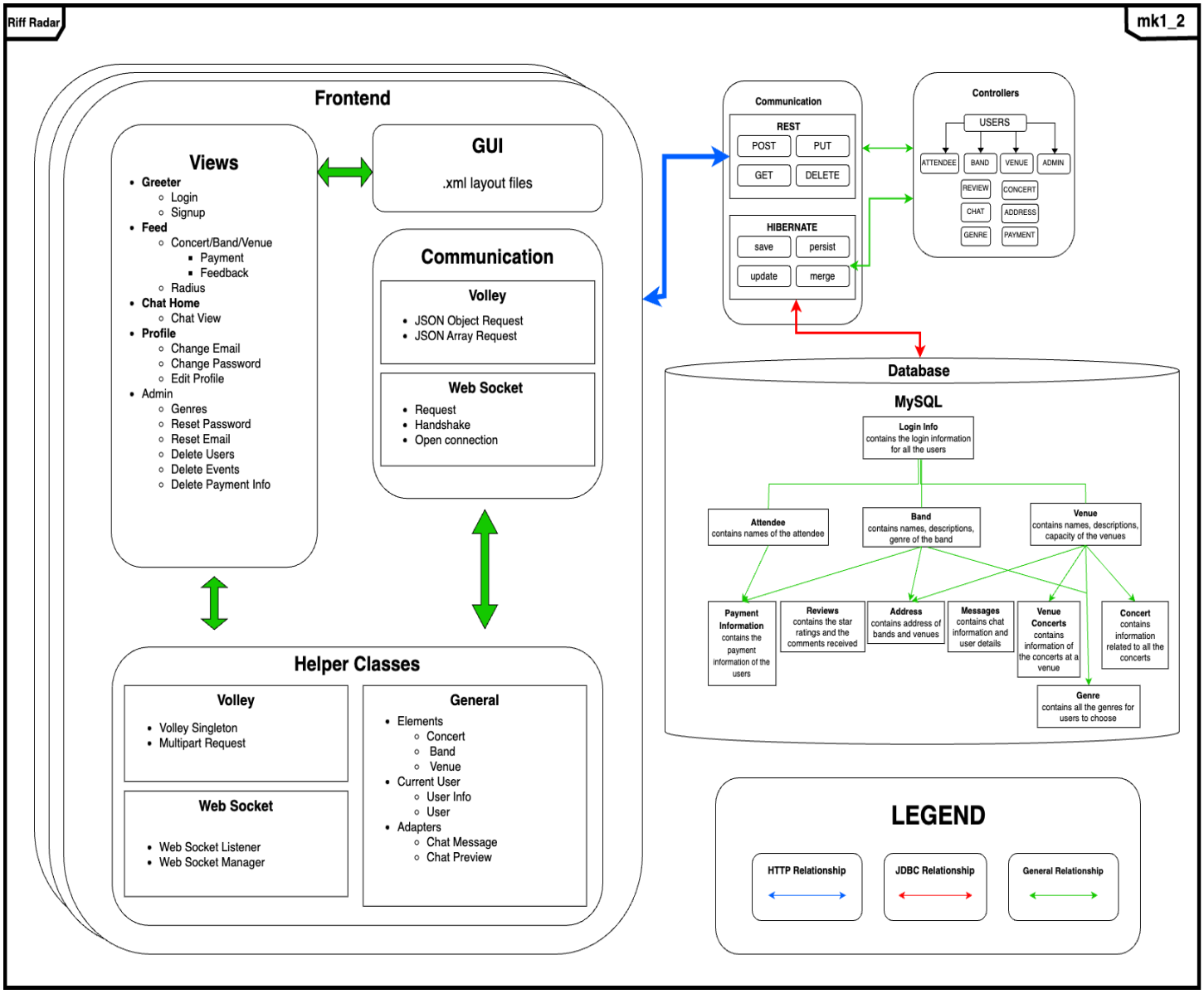
Design Document for Riff Radar

MK1_2

Kaden Berger: 25 %
Sullivan Hart: 25 %
Pranava Sai Maganti: 25 %
Peter Yehl: 25 %

Block Diagram



Backend

Communication

The backend updates the database using the mapping's URLs. The types of URLs are:

- **Get**: No update to the database. Instead, return the requested data.
- **Post**: Add an element to the database.
- **Put**: Update already existing information in the database.
- **Delete**: Delete a specified element in the database.

The backend also hosts two different web sockets. When opened, the frontend and backend can communicate. When a new message is sent, it is saved to the database.

Controllers

The backend updates the database using the mapping's URLs. The types of URLs are:

- **Address**: Uses the mappings above to create addresses that can be used to store location data for different user types.
- **Concerts**: Similar to address but is specifically used for creating an event that has a one-to-one relationship with a venue user.
- **Genre**: Standalone class with a singular variable, Name. This controller is utilized by the admin user to add and delete genres in a list that can be used by users, Venues and Bands.
- **Payment**: Standalone class with variables like cardNumber, expirationDate, and other personal information that can be used for a user to pay for a concert.
- **Reviews**:Websocket controller with a connected table that can be used to leave reviews with a rating and feedback on different concerts, similar to a chat, but prioritizes anonymity and doesn't display any non-entered user information.
- **Users**
    - Attendee: The most common user type, contains only basic information like name
    - Band: Similar to attendee but has a few more parameters like genre, description, and an image.
    - Venue: Also similar to attendee but has the most parameters out of all the user types, has parameters such has capacity and description, also has multiple relationships with other classes like address and concerts.
- **Login Info**: Used for every user type to enable login functionality. Linked with a one-to-one relationship with every user type as well. Contains an identifying id as well as the email, password, and userType of a user.

Frontend

Greeter (All)

- Greeter generates a page with the following elements:
    - Button: **Login**
    - Button: **Signup**
    - Button: **Guest**
- Based on the button pressed, the user is taken to the appropriate screen.

Login (All)

- Login page, has only two fields:
        - Email
        - Password
- The user will have to enter their details then the data received from the user will be verified with the details in the loginInfo table to make sure they match up to an existing user.
- If the user enters the correct details, the user will be taken to the respective feed pages, depending on their user category.

Signup (All)

- The signup page has different fields for different users that show up based on the user category selected.
- The common fields are:
        - Name
        - Email
        - Password
        - Confirm Password
- If the User Type selected is Attendee, no new fields appear on the screen.
- If the User Type selected is Band, the address, description, and genre fields appear on the screen.
- If the User Type selected is Venue, the address, description, and capacity fields appear on the screen.
- The email and password are to follow the regular expression format, otherwise there will be an error message appearing on the screen near the respective field.

Feed (Guest, Attendee, Band, Venue)

- Feed generates a page with the following elements:
    - Top Bar
        - Button: **Profile**
            - Goes to the profile page
        - Button: **Logo**
            - Resets to the first page of the feed
        - Button: **Map**
            - Goes to the radius activity

        - Cards
            - ImageView: **Element's image**
            - TextView: **Element's name**
            - TextView: **Price** (Guest, Attendee)
        - Bottom Bar
            - Button: **Back Arrow** (if sufficient elements)
                - Previous 10 elements
            - Button: **Next Arrow** (if sufficient elements)
                - Next 10 elements
- On create or resume, the feed activity gets the appropriate array of JSON objects (concert for guest/attendee, venue for band, band for venue), and fills in cards with the info of each object (displays up to 10 at a time).
- If any part of a card is pressed, the user is brought to that element's page

Concert (Guest, Attendee)

- Concert generates a page with the following elements:
    - Button: **Back Button**
    - ImageView: **Concert Image**
    - TextView: **Name**
    - TextView: **Genre**
    - TextView: **Bands**
    - TextView: **Ages**
    - TextView: **Description**
    - MapView: **Map & Address**
- Concert page creates a Volley request for the specified concert element, fills in fields based on the response.

Band (Venue)

- Band generates a page with the following elements:
    - Button: **Back Button**
    - ImageView: **Band Image**
    - TextView: **Name**
    - TextView: **Genre**
    - TextView: **Description**
    - MapView: **Map & Address**
- Band page creates a Volley request for the specified band element, fills in fields based on the response.

Venue (Band)

- Venue generates a page with the following elements:
    - Button: **Back Button**
    - ImageView: **Venue Image**
    - TextView: **Name**

    - TextView: **Genre**
    - TextView: **Description**
    - MapView: **Map & Address**
- Venue page creates a Volley request for the specified venue element, fills in fields based on the response.

Radius (All)

- Radius generates a page with the following elements:
    - Button: **Back Button**
    - EditText: **Enter Address**
    - Button: **Search Button**
    - MapView: **Map**
    - Slider: **Radius Select**
    - TextView: **Selected Radius**
- Map view displays the current map, changes on slider change, enter key in enter address, or search button pressed. The map does a get request to the Google Maps API to get the correct map view.

Chat Home (Band, Venue)

- Chat Home generates a page with the following elements:
    - Button: **Back Button**
    - Recycler: **Chat Previews**
- The Chat Previews recycler is populated with all the users available to chat with. This information is received via a Volley request for an array of users to chat with. When a chat preview is clicked, the chat with the appropriate user is opened.

Chat (Band, Venue)

- Chat generates a page with the following elements:
    - Button: **Back Button**
    - Button: **Delete Chats**
    - Recycler: **Chats**
    - EditText: **Enter Message**
    - Button: **Send Message**
- When the chat is opened, a web socket connection is opened. The chat history is loaded into the recycler from the backend.
- When the delete button is clicked, a Volley request deletes that user from the current user's list of users to chat with.
- When a new message is received, a new item is added to the recycler depending on if it is sent, received, or system.
- When the send button is clicked or the enter key is pressed in the enter message, the text from the enter message is sent to the server.

Profile (Attendee, Band, Venue)

- Profile generates a page with the following elements:

    - Top Bar
        - Button: **Back Button**
        - Button: **Dropdown Menu**
    - Dropdown Menu
        - Button: **Change Password**
        - Button: **Change Email**
        - Button: **Edit Profile**
        - Button: **Logout**
        - Button: **Delete Account**
    - ImageView: **Image**
    - TextView: **Title**
    - TextView: **Genre** (Band, Venue)
    - TextView: **Street Address** (Venue)
    - TextView: **City** (Band, Venue)
    - TextView: **State** (Band, Venue)
    - TextView: **Description** (Band, Venue)
- When the page is created or resumed, the fields will fill using a Volley JSON object request. The dropdown menu will default to hidden.
- When the dropdown button is clicked, a menu will appear with the dropdown menu.
- When any of the buttons (besides delete account) are clicked, they will bring the user to the appropriate page.
- When the delete account button is clicked, the account will be deleted.

Reset Password (Attendee, Band, Venue, Admin)

- Currently, the reset password button is located in the User Profile, where the user can navigate a click on the button. Upon clicking the button, the user will be redirected to the reset password screen, where there are three fields:
    - Old Password
    - New Password
    - Confirm New Password
        - Note: All the passwords need to pass the specified conditions.
            - Minimum 8 characters long
            - At Least one special character
            - At Least one number
            - At Least one alphabet
            - The old and the new passwords cannot be the same
- We plan to implement a security feature, for the sequence of reset password, where in when the reset password button is clicked, the user will be navigated to a screen, where they will have to enter their email address linked with the account, which if exists in the database, the user will receive an OTP on their email, which will be validated with the server when the

user inputs the OTP. Only upon the validation of the security code, the user will be able to reset their password.

- This is a feature available for the users as well as the admin. In cases, where the user cannot reset their password, then the admin will be able to reset the password for the user.

Change Email (Attendee, Band, Venue)

- There are cases, where the users lose access to their email or want to change their email linked with the app, to accommodate this need, we have a feature that the user could use to change their email.
- There are again three fields on this page:
    - Old Email
    - New Email
    - Confirm New Email
        - The user needs to adhere to the regular expression format while signing up, logging in or even while changing their email.
        - Example of a Regular Expression Email: **alpha@bravo.charlie**
        - Example of a Non-Regular Expression Email: **alpha**
- This is also a feature that is accessible by both the users and the admin. In cases where the user cannot change their email, then the admin will be able to update the email for the user.

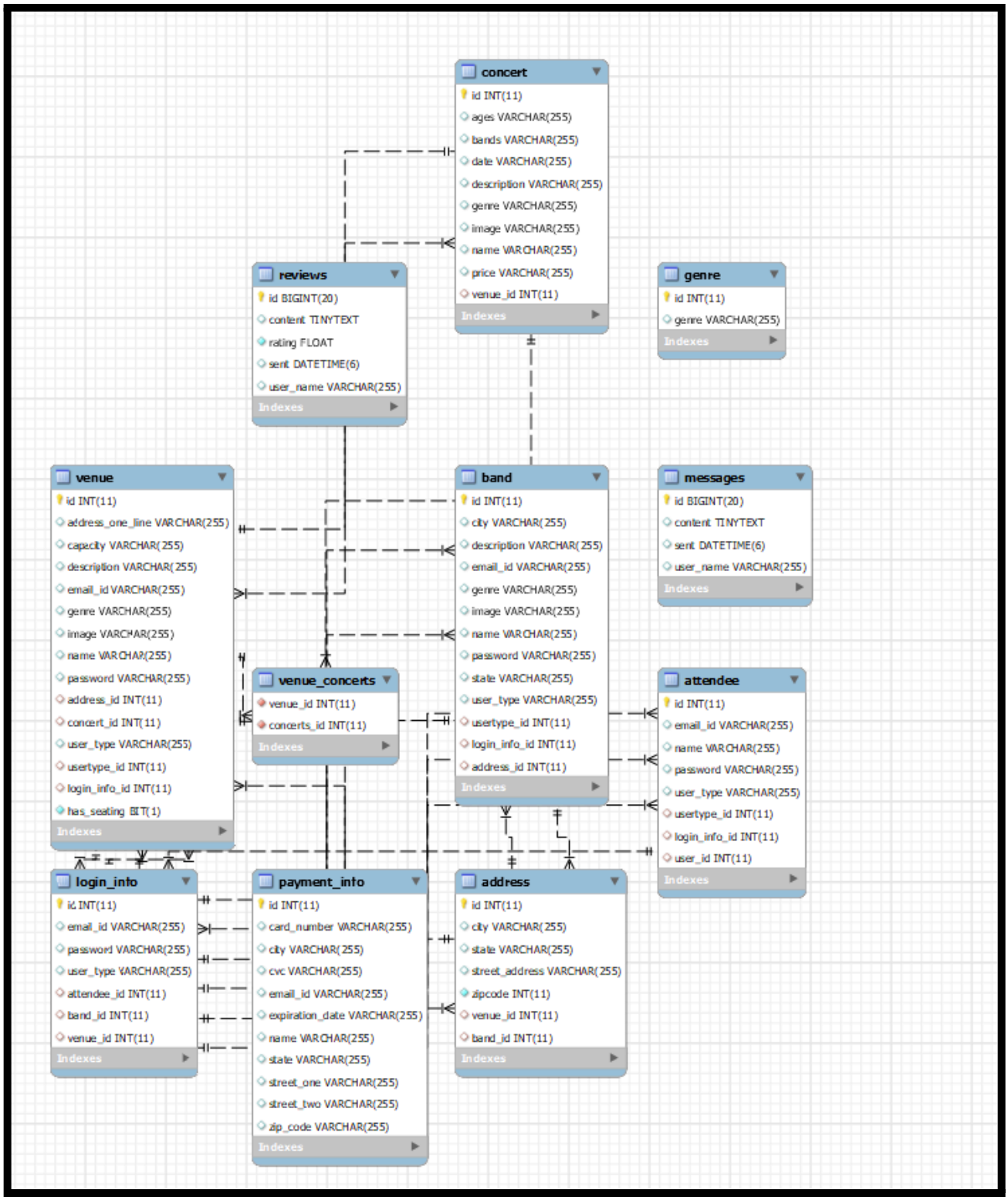
Edit Profile (Attendee, Band, Venue)

- Edit Profile generates a page with the following elements:
    - Button: **Back Button**
    - ImageView: **Image**
    - EditText: **Title**
    - EditText: **Genre** (Band, Venue)
    - EditText: **Street Address** (Venue)
    - EditText: **City** (Band, Venue)
    - EditText: **State** (Band, Venue)
    - EditText: **Description** (Band, Venue)
    - Button: **Update**
- When created, the EditText's hints are set from the data in a Volley JSON get request for the profile being edited.
- When the update button is clicked, a Volley JSON put request is sent. This put request contains the previously requested JSON object with any fields that the user filled out overwritten with the updated information. When the user returns to the profile page, the element with the updated information will be there.

Admin (Admin)

- The Admin button is a page, that contains multiple features, included within it. This is an all-in-one page, starting from deleting users, deleting events to deleting user payment information, from changing email to resetting password, updating concert information like changing the genre, adding or updating the bands.

- *Delete Users* (Admin)
  - The Admin has access to the database, so, the admin will be able to delete any user and ban their account from usage. There are two fields on this screen:
    - User Category
    - User ID
  - Using the information entered by the admin, the user account will be deleted.
  - If the user wants to use their account again, then they have to sign up again.
- *Delete Events* (Admin)
  - There are situations when the venue or the bands have to cancel their concert or event. Similar to the User Deleting, the Delete Events, that is only available to the Admin, where the Admin enters the concert ID, and deletes it by providing a valid reason.
- *Delete Payment Info* (Admin)
  - Payment Information - Credit Card Number, Expiration Date, Address are sensitive information, which will be stored in the database, upon completion of the payment. But, due to privacy and security reasons, if the user does not want their information to be saved, then the admin will be able to delete their respective payment information.
  - This page has one field, the payment id - this is the id that is generated automatically, but the database when a new payment information is received.
- *Request Form* (Attendee, Band, Venue)
  - Note: This is currently not developed.
  - This can be accessed through the User Profile page.
  - We plan to develop a request form where the user will need to enter their details and request help. Help can be requested for features, that only the admin can access like:
    - Delete User Account
    - Delete Events
    - Delete Payment Information
    - Updating the Event Information
  - This field will have potentially have the three fields (*subject to change):
    - Email: To match with the appropriate user
    - Password: To verify the user
    - Dropdown: To select the assistance they need from the admin


concert
id INT(11)
ages VARCHAR(255)
bands VARCHAR(255)
date VARCHAR(255)
description VARCHAR(255)
genre VARCHAR(255)
image VARCHAR(255)
name VARCHAR(255)
price VARCHAR(255)
venue_id INT(11)
Indexes
reviews
id BIGINT(20)
content TINYTEXT
rating FLOAT
sent DATETIME(6)
user_name VARCHAR(255)
Indexes
genre
id INT(11)
genre VARCHAR(255)
Indexes
venue
id INT(11)
address_one_line VARCHAR(255)
capacity VARCHAR(255)
description VARCHAR(255)
email_id VARCHAR(255)
genre VARCHAR(255)
image VARCHAR(255)
name VARCHAR(255)
password VARCHAR(255)
address_id INT(11)
concert_id INT(11)
user_type VARCHAR(255)
usertype_id INT(11)
login_info_id INT(11)
has_seating BIT(1)
Indexes
band
id INT(11)
city VARCHAR(255)
description VARCHAR(255)
email_id VARCHAR(255)
genre VARCHAR(255)
image VARCHAR(255)
name VARCHAR(255)
password VARCHAR(255)
state VARCHAR(255)
user_type VARCHAR(255)
usertype_id INT(11)
login_info_id INT(11)
address_id INT(11)
Indexes
messages
id BIGINT(20)
content TINYTEXT
sent DATETIME(6)
user_name VARCHAR(255)
Indexes
venue_concerts
venue_id INT(11)
concerts_id INT(11)
Indexes
attendee
id INT(11)
email_id VARCHAR(255)
name VARCHAR(255)
password VARCHAR(255)
user_type VARCHAR(255)
usertype_id INT(11)
login_info_id INT(11)
user_id INT(11)
Indexes
login_info
id INT(11)
email_id VARCHAR(255)
password VARCHAR(255)
user_type VARCHAR(255)
attendee_id INT(11)
band_id INT(11)
venue_id INT(11)
Indexes
payment_info
id INT(11)
card_number VARCHAR(255)
city VARCHAR(255)
cvc VARCHAR(255)
email_id VARCHAR(255)
expiration_date VARCHAR(255)
name VARCHAR(255)
state VARCHAR(255)
street_one VARCHAR(255)
street_two VARCHAR(255)
zip_code VARCHAR(255)
Indexes
address
id INT(11)
city VARCHAR(255)
state VARCHAR(255)
street_address VARCHAR(255)
zipcode INT(11)
venue_id INT(11)
band_id INT(11)
Indexes