1. Team Number: 014-5

2. Team Name: Jamaica B

3. Team Members:
    - James Nguyen, nikuteh, jang5198@colorado.edu - back end
    - Cara Wang, wangcara, cawa5085@colorado.edu - front end
    - Benedikt Safigan, bsafig, besa8557@colorado.edu - database
    - Mason Chansamone, mchansamone1, mach3999@colorado.edu -
    - George Fisher, georgefisher1024, gefi6263@colorado.edu - front end

4. Application Name: Spotigang

5. Application Description:

Use Spotify's free API

Features
- View songs friends are listening to right now
- View your friend's list
- Profile
- Rate album/song
- Comment
- View friend's spotify wrapped

6. Audience: Spotify users who want to connect more with their friends' music

7. Vision Statement:

For music-lovers who want a better way to share reviews of music. Spotigang is a an app that makes it easy to post and view reviews of songs/albums/artists. Unlike Airbuds, our product can help users discover new songs, review detailed ratings from users who have listened to songs.

8. Version Control:

9. Development Methodology:

<u>Front end</u>

- Home page
    - Login page
    - Registration / sign up page
    - Feed/Home
- Profile pages
    - Friends list (if it's your own profile? Should you be able to see your friends' friends?)
        - Friends list has the current listening embedded?
    - Reviews of the user
    - If the user has a song currently playing, show that as well
- Review
    - Comment section to reply to a review [not yet]
    - Song
    - Star rating of song
    - Poster
    - Content
- Listening to page: see what your friends are listening to

<u>Back end</u>

- A server that allows the UI to communicate with the database
    - A database that stores user information
        - Passwords must be hashed and stored in the database
- Session Management - The user must be able to log in and out of the application and the session must be maintained

Application is built within Docker containers - you can find some updates to the docker-compose.yaml in the write-up below.

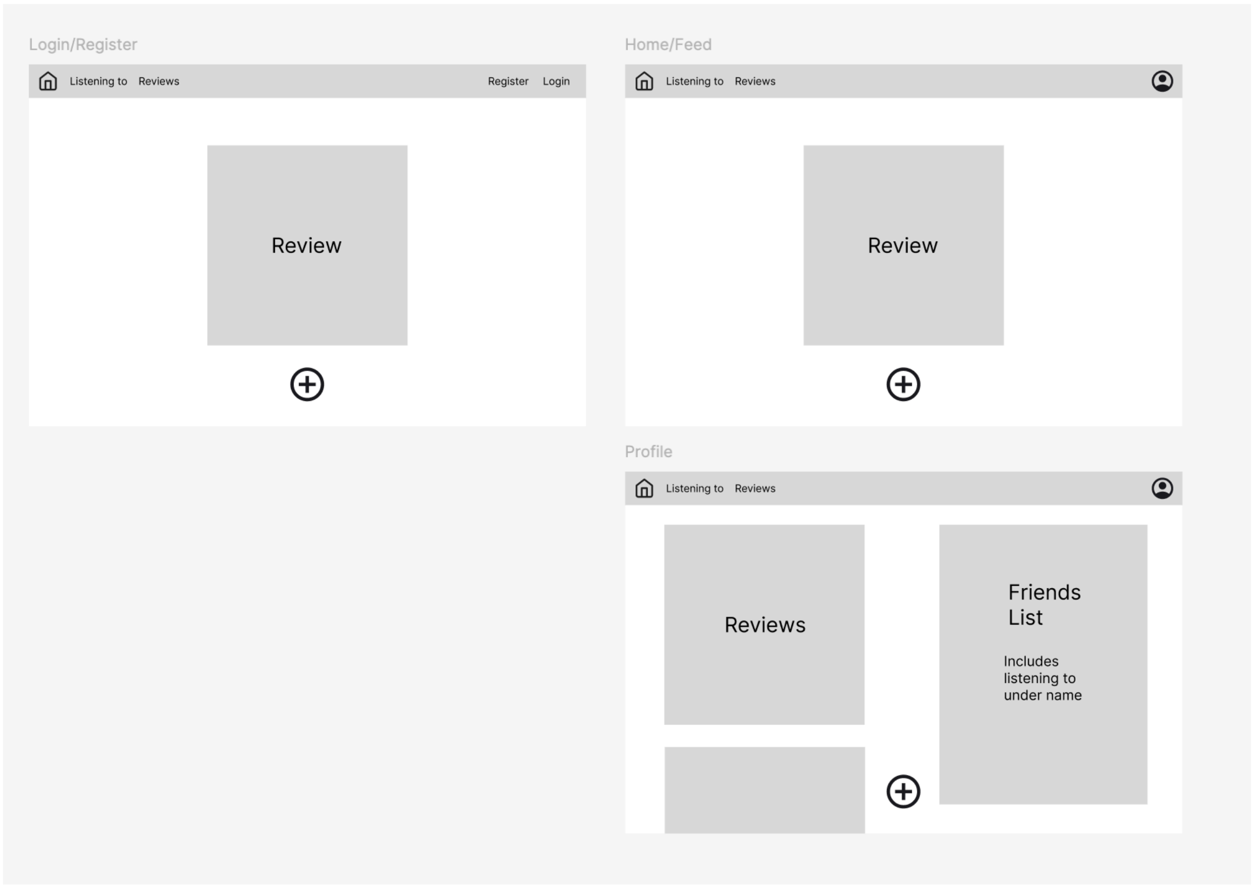
From a technical standpoint, this project will demonstrate the integration of a front-end GUI, a Node.js/Express back-end, and a PostgreSQL database within Docker containers. The architecture ensures scalability and portability while providing a strong foundation for future enhancements such as Spotify API integration, friend requests, or music recommendations. This app delivers both social connection and technical depth, making it an ideal full-stack software project.

10. Communication Plan: We will be communicating over Slack.

11. Meeting Plan:
    - Team Meeting: [day and time], meeting Online
    - Weekly Meeting with TA: Every Wednesday, 4:40 PM, meeting Online

12. Use Case Diagram:

13. Wireframes:
    - Home page
    - Listening to
    - Reviews
    - Profile
        - Friends list
        - Recently played
        - Top 5 artists
        - Top 5 songs
        - Currently listening to



14. Potential risks:

- Data breach — SEVERE — data is stored in raw text. To mitigate risks from a breach we could hash all data, so even if the data is obtained, it’s not readable without the hash keys

- Code Injection/SQL Injection, We need to ensure that any code a user may write can’t be injected into our application which may enable them to steal data or harm our database.
-