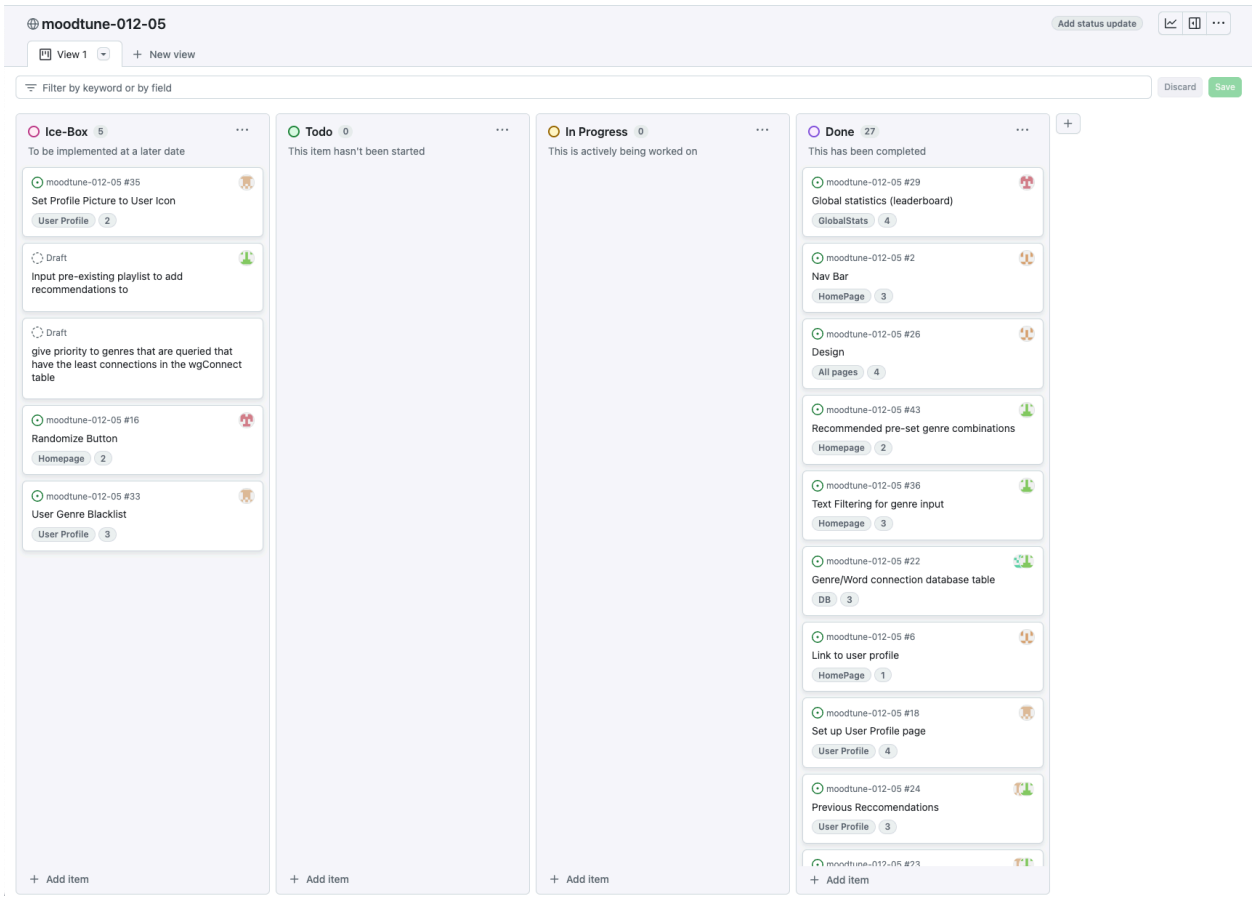
Title: Moodtune

Names: Robby Toomey, Tahn Jandai, Prachi Soni, Conner Neuhart, Margaret Muhich

Project Description:
An app that is able to generate songs from a database of music based on how the user is feeling or what setting/scenario they are currently in. The user could make an account, log in, and it opens to a page that asks them a series of questions about their current mood/situation, etc. Based on their responses, a recommended song/playlist/music genre can be generated for the player to listen to based on the information collected from the user. Additionally, they could create their own playlists and add songs of their choice, and simply just explore new genres/songs/albums if they want to.

Project Tracker: https://github.com/users/cneuhart/projects/1/views/1



Video:
https://drive.google.com/file/d/15wvZIzVkPumTtBe-hot8NdCvNfTaxKSh/view?usp=sharing (google drive link, also available in github repo)

VCS: https://github.com/cneuhart/moodtune-012-05

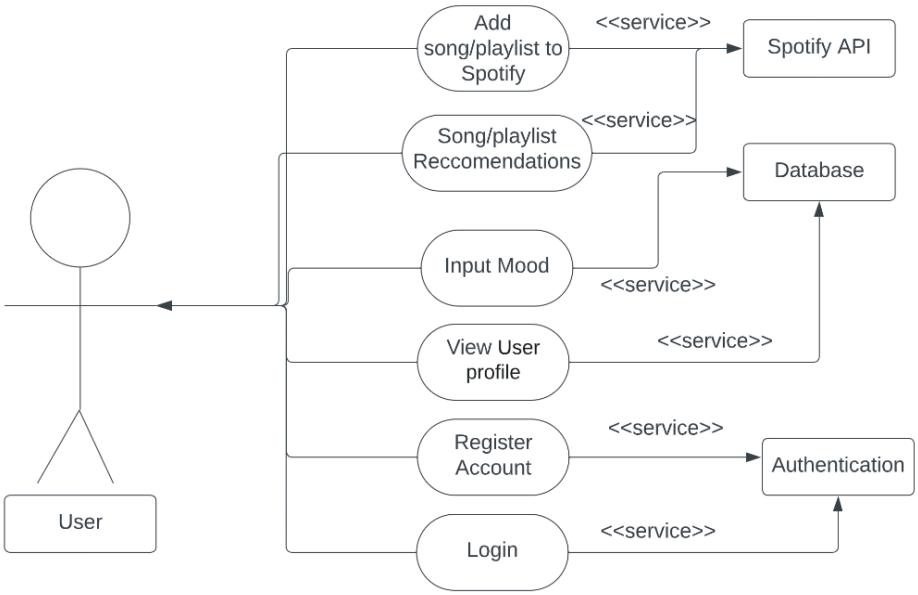
Contributions:

- Robby Toomey:
    - Resources
        - Images
            - Found or created all images in the project
        - .CSS
            - Created and implemented everything in style.css, a .css file applying to all pages
    - Views
        - Design
            - Design, styling, and implementation of html and .CSS code for login, register, recommendations, and user profile pages
        - Navbar
            - All navbar design and functionality

- Tahn Jandai:
    - Login functionalities
        - Login and register functions
    - Logout functionalities
        - Destroying seasons
    - API Route redirection/Middleware
        - Routes to different pages
    - Global Statistics
        - Using collected user data to show the most recommended tracks and artists
    - Footer
        - Basic footer
    - Unit testing
        - Login testing
- Prachi Soni:
    - User Profile
        - Username display
        - Profile Picture display and replacement
    - Previous Recommendations
        - Database to store filtered recommendations from recommendations table after mood input
        - API Route to grab Recommendations
        - Display for Previous Recommendations on User Profile
    - Drop Down Menu for mood input
- Conner Neuhart:
    - Technologies/Feature List:
        - OAuth Spotify API connection

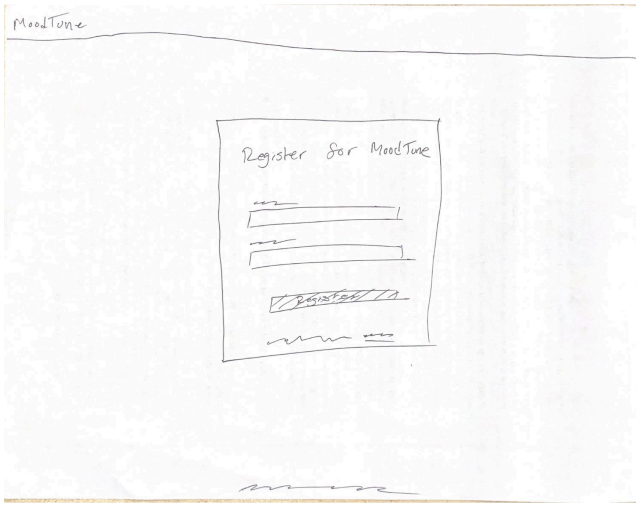
        - Moodtune user registration
        - Retrieve Spotify recommendation based on submitted genre(s)
        - Store all generated recommendations in DB
        - Personal spotify statistics page + API queries
        - Add Spotify playlist from recommendations
            - API requests for create empty playlist, populate playlist
        - Add playlist from previous recommendations
        - Matching user input to spotify genres
            - Genre/Word table implementation
        - Text filtering/sanitization for recommendation input
        - homepage/recommendations page redesign
            - Includes proper tables and scrollbars for homepage and stats pages
        - Recommend pre-set genre combinations (“fusion”)
- Margaret Muhich:
    - Framework
        - main.hbs
        - message.hbs
        - Created other base files for main branch
    - Recommendations
        - Recommendation display
        - handlebars
    - Word connections database
        - Created tables
            - genres
                - Copied & numbered genres
            - moods
                - Numbered & added moods
            - wgConnect
                - Added & numbered connections
    - App testing

Case Diagram:

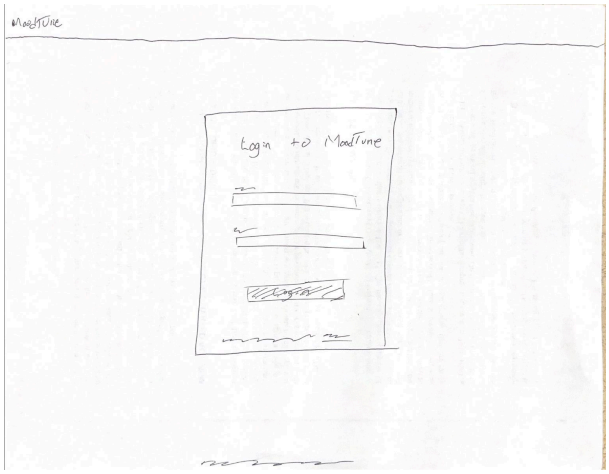


Wireframes:

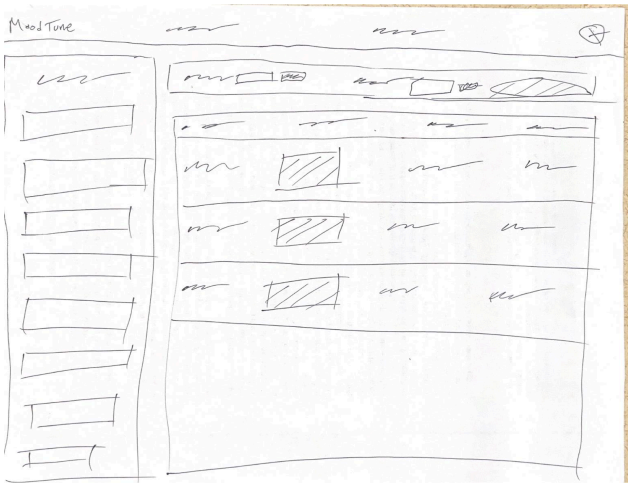
Register page



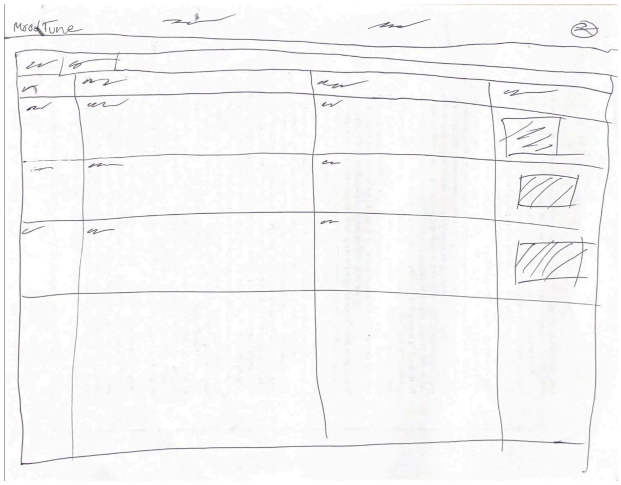
Login page



Recommendations page

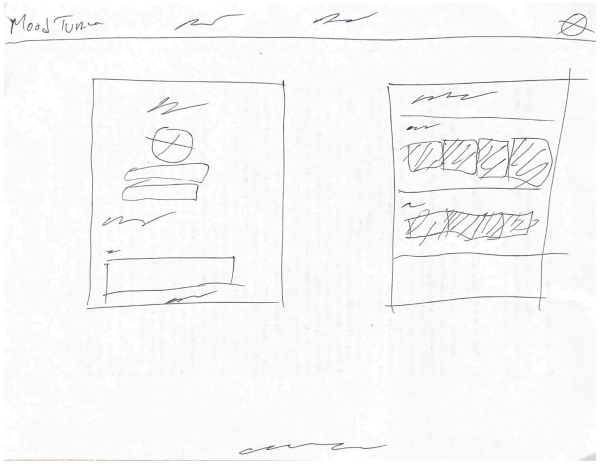


Global stats page



Statistics page

User profile page



Test Results:

- **Mood input and generate recommendations**
    - Task
        - Navigate to the recommendations text input field, enter the desired mood and click submit. The desired result is the correct recommendation being rendered.
    - Observations

        - The user successfully found the input field and could input a mood that they wanted. There was some confusion with the “Choose a Mood” drop down menu as that also appeared to be a mood input but in the end chose the right option. We added a placeholder saying "What are you looking for?" to the recommendations input field to make it more user friendly.
- **Save a recommendations to a spotify playlist**
    - Task
        - Once a recommendation is generated, navigate to the add to the “add to spotify playlist” button and click it. The desired result is the playlist being added to the users spotify account.
    - Observations
        - The user successfully generated a recommendation and navigated to the “add to spotify playlist” button and clicked it. Feedback was that they liked the appearance of the “add to spotify playlist” button as it was easy to find and has the Spotify logo on it. The confirmation message after was also noted. This feature worked well and did not need any changes.
- **Save previous recommendations to a spotify playlist**
    - Task
        - Every recommendation is stored in the profile page. The task is to navigate there and add a previously recommended song to a spotify playlist.
    - Observations
        - The user was initially a little confused where to locate the previous recommendation. Though with the limited options of pages the user figured out that it was located in the account page. The rest was very straightforward and the user had no problem finding and a desired playlist and adding it to spotify. When designing the app we expected this feature to be hidden away as we wanted users to focus on new generations while still letting them see their history.
- **Look at personal statistics**
    - Task
        - Navigate to the statistics page and look at their top artists and tracks with varying time periods.
    - Observations
        - The user quickly found the statistics page as it is clearly indicated on the nav bar. Once there the user found navigating the page easy but there were some issues. The first was when the time range was

changed the page would refresh and the dropdown menu would default to the “short” time even though results for medium or long were showing. Another annoyance was when changing the time period in the tracks tab, once the page refreshed with the new results it would change back to the artists tab. This was frustrating and confusing for the user. We decided to address these issues by adding functions that would remember what the previous selection was and render the page accordingly.

- What are the users doing?
- What is the user's reasoning for their actions?
- Is their behavior consistent with the use case?
- If there is a deviation from the expected actions, what is the reason for that?
- Did you use that to make changes to your application? If so, what changes did you make?

Deployment:

- Clone repository
- Create .env file in the /ProjectSourceCode/ folder
  - must input own session_secret, Spotify API client_id, and Spotify API client_secret if running locally*
    - # database credentials
    - POSTGRES_USER="postgres"
    - POSTGRES_PASSWORD="pwd"
    - POSTGRES_DB="users_db"
    -
    - # Node vars
    - SESSION_SECRET=””
    - client_id =“”
    - client_secret =””
    - host="db"
  - *TA will need to message/email us for client_id and client_secret (will also add your spotify account email to manual access list for spotify API)
- cd to /ProjectSourceCode/ folder
- “npm init”
- “docker compose up”

**Rather than having the TA create their own spotify application to access the API (inputting their own client_id and client_secret), the TA needs to send us a spotify account email so we can add it to the developer accounts on our Spotify API dashboard (developer access to Spotify API*

*requires accounts must be added manually to a list, while deployed applications that are accepted after Spotify reviews the application, which takes a certain number of weeks, can be used by any spotify account)**