

**EMOTION TUNE PROJECT REPORT**

**Were Vincent**

**19/January/2025**

**Introduction**

The EmotionTune project aims to develop a music streaming application that caters to users' emotional preferences. It leverages a relational database to store and manage user information, music details, emotions, recommendations, playback history, and user preferences. This report delves into the project's database design, user manual, SQL queries, and potential distribution of work among team members.

**Database Scheme**

***Tables***

1. **Users**

**Columns**

1. UserID (INTEGER PRIMARY KEY AUTOINCREMENT): Unique identifier for each user.
2. Username (TEXT NOT NULL UNIQUE): User's chosen username for login.
3. Email (TEXT NOT NULL UNIQUE): User's email address for registration and communication.
4. Password (TEXT NOT NULL): Encrypted user password for secure authentication.
5. **Music**

* **Columns**

1. MusicID (INTEGER PRIMARY KEY AUTOINCREMENT): Unique identifier for each song.
2. Title (TEXT NOT NULL): Title of the song.
3. Artist (TEXT NOT NULL): Artist or band who created the song.
4. Genre (TEXT): Genre classification of the song (e.g., Pop, Rock, Jazz).
5. ReleaseDate (DATE): Date the song was officially released.
6. Album (TEXT): Album the song belongs to (if applicable).
7. LastPlayed (TIMESTAMP): Timestamp of the most recent playback for this song (updated by trigger).
8. **Emotions**

* **Columns**

1. EmotionID (INTEGER PRIMARY KEY AUTOINCREMENT): Unique identifier for each emotion.
2. EmotionName (TEXT NOT NULL UNIQUE): Name of the emotion (e.g., Happy, Sad, Energetic).
3. **Recommendations**

* **Columns**

1. RecommendationID (INTEGER PRIMARY KEY AUTOINCREMENT): Unique identifier for each recommendation.
2. UserID (INTEGER): Foreign key referencing the Users table.
3. MusicID (INTEGER): Foreign key referencing the Music table.
4. EmotionID (INTEGER): Foreign key referencing the Emotions table.
5. FOREIGN KEY (UserID) REFERENCES Users(UserID): Ensures recommendations are linked to valid users.
6. FOREIGN KEY (MusicID) REFERENCES Music(MusicID): Ensures recommendations point to existing songs.
7. FOREIGN KEY (EmotionID) REFERENCES Emotions(EmotionID): Ensures recommendations are associated with valid emotions.
8. **PlaybackHistory**

* **Columns**

1. PlaybackID (INTEGER PRIMARY KEY AUTOINCREMENT): Unique identifier for each playback event.
2. UserID (INTEGER): Foreign key referencing the Users table.
3. MusicID (INTEGER): Foreign key referencing the Music table.
4. EmotionID (INTEGER): Foreign key referencing the Emotions table.
5. PlayTime (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP): Timestamp of the playback event.
6. FOREIGN KEY (UserID) REFERENCES Users(UserID): Ensures playback history entries are linked to valid users.
7. FOREIGN KEY (MusicID) REFERENCES Music(MusicID): Ensures playback history entries reference existing songs.
8. FOREIGN KEY (EmotionID) REFERENCES Emotions(EmotionID): Ensures playback history entries are associated with valid emotions.
9. **UserPreferences**

* **Columns**

1. PreferenceID (INTEGER PRIMARY KEY AUTOINCREMENT): Unique identifier for each user preference.
2. UserID (INTEGER): Foreign key referencing the Users table.
3. PreferredEmotionID (INTEGER): Foreign key referencing the Emotions table.
4. FOREIGN KEY (UserID) REFERENCES Users(UserID): Ensures user preferences are linked to valid users.
5. FOREIGN KEY (PreferredEmotionID) REFERENCES Emotions(EmotionID): Ensures user preferences align with valid emotions.
6. **Indexes**
7. ***Users.Email -*** Enables efficient user lookups by email for login and other operations.
8. ***Music.Title -*** Allows for faster searches and filtering of songs by title.

**Triggers**

* ***UpdateLastPlayed (AFTER INSERT ON PlaybackHistory) -*** Automatically updates the LastPlayed field in the Music table whenever a new playback entry is added, ensuring the most recent play time is always reflected.

**Relationships**

* One user can have many playback history entries (one-to-many).
* One song can be in many playback history entries and have many recommendations (one-to-many).
* One emotion can be associated with many recommendations and playback history entries (one-to-many).
* One user can have one set of preferences (one-to-one).

**User Manual**

**Getting Started**

1. Download and install the EmotionTune application.
2. Create an account by providing a username, email address, and secure password.
3. Customize Preferences

* Access the "User Preferences" page.
* Use the radio buttons to rate your preference level (1-5) for different emotions (Happy, Energetic, Calm, Sad, Nervous).
* Select your favorite genres from the provided list (Pop, Jazz, Hip Hop, Rap, Country, Rock).
* Click the "Save Preferences" button to save your selections.

**4. Explore Music**

* Browse through the music library, exploring songs by artists, genres, or emotions.
* Use the search bar to find specific songs or artists.
* View song details, including album art, artist information, and release date.

**5. Play Music**

* Select a song to play.
* Use the playback controls (play, pause, skip, etc.) to manage playback.
* Adjust volume and playback speed (if available).

**6. View Playback History**

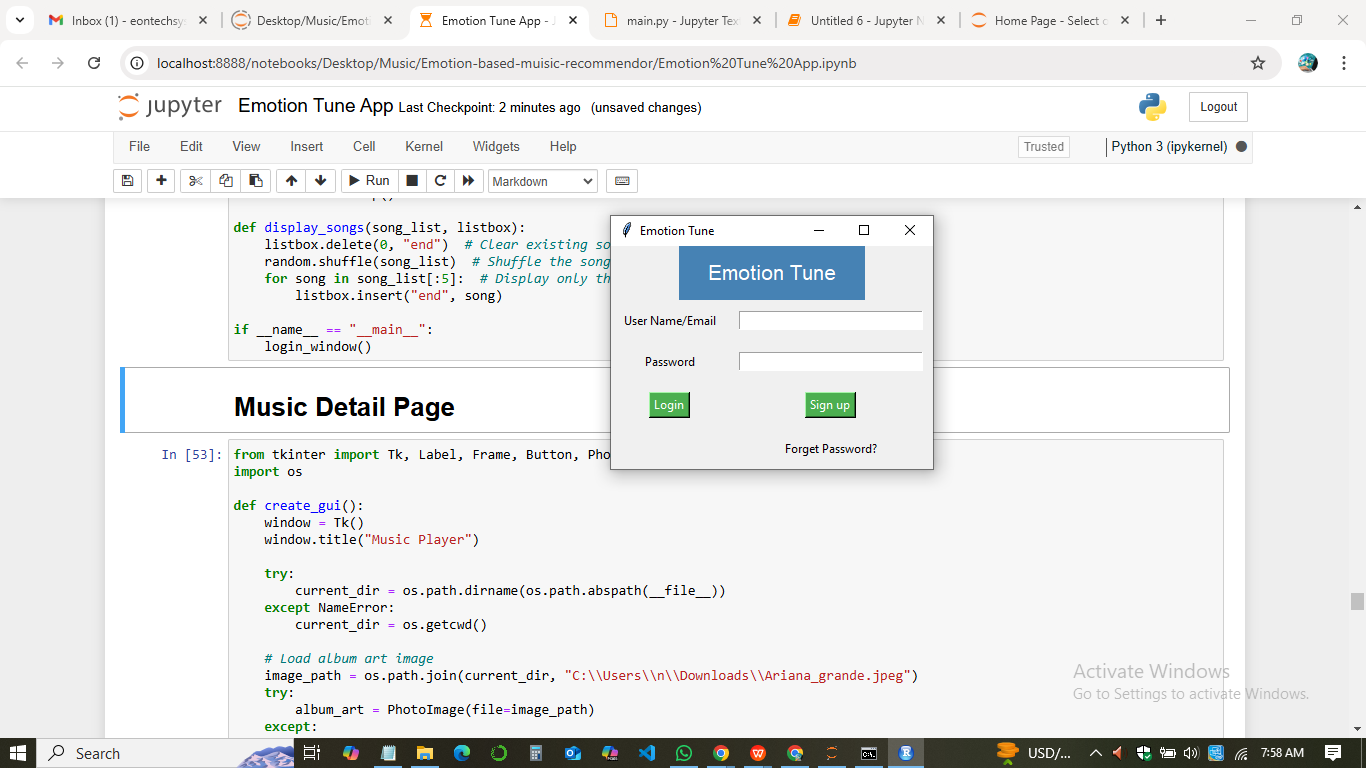
* Access the "Playback History" page.
* View a list of recently played songs with song title, artist, and play time.
* Filter the history by date, genre, or emotion.
* Use the "Replay" button to play a previously listened song.

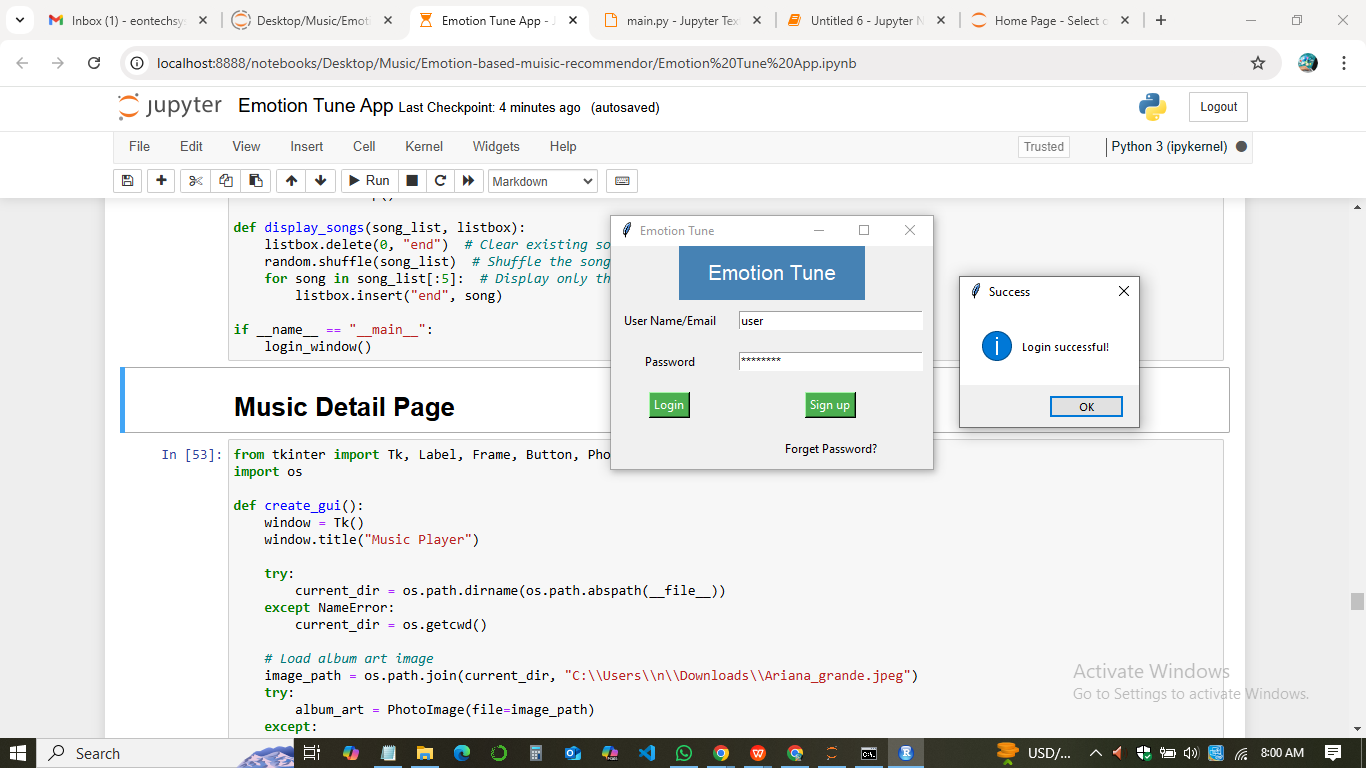
**7. Manage Account**

* Update account information (username, email, password).

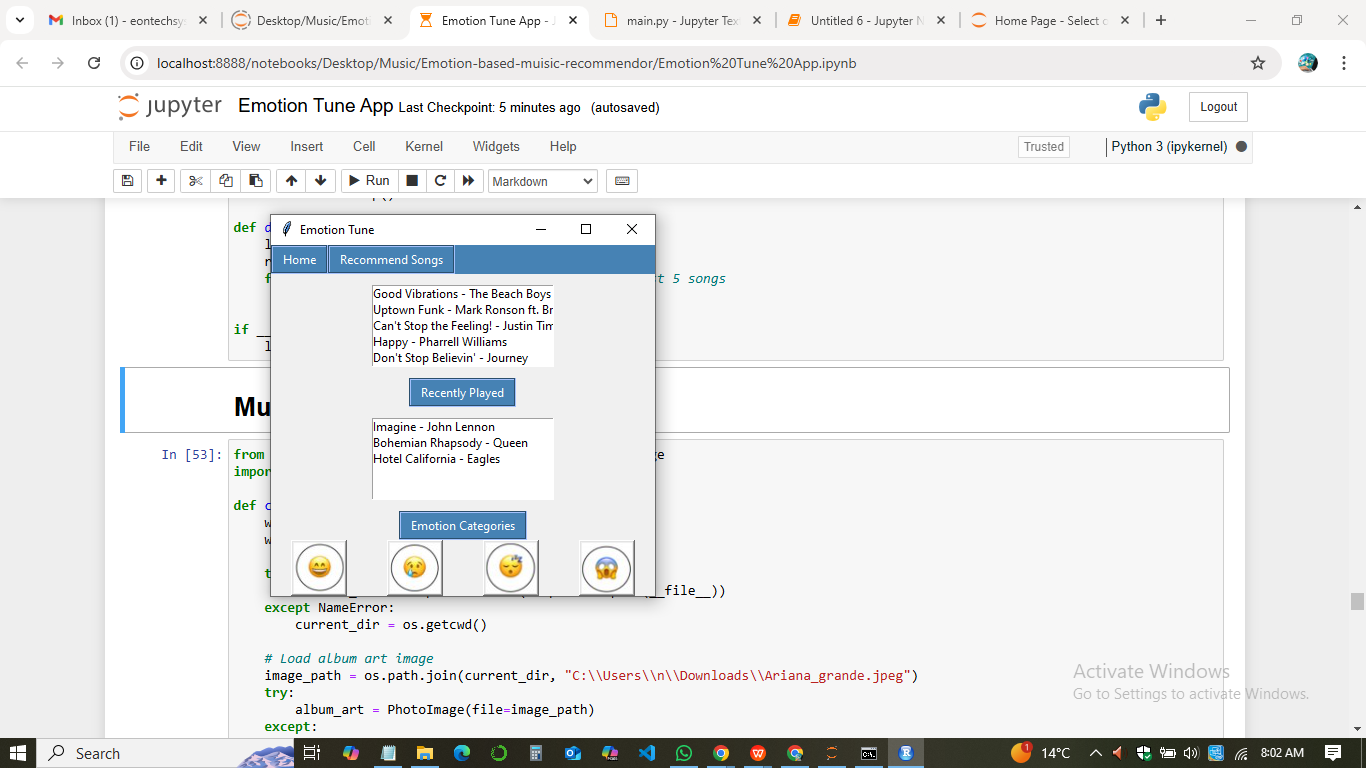
**Screenshots**

* **Login Screen**

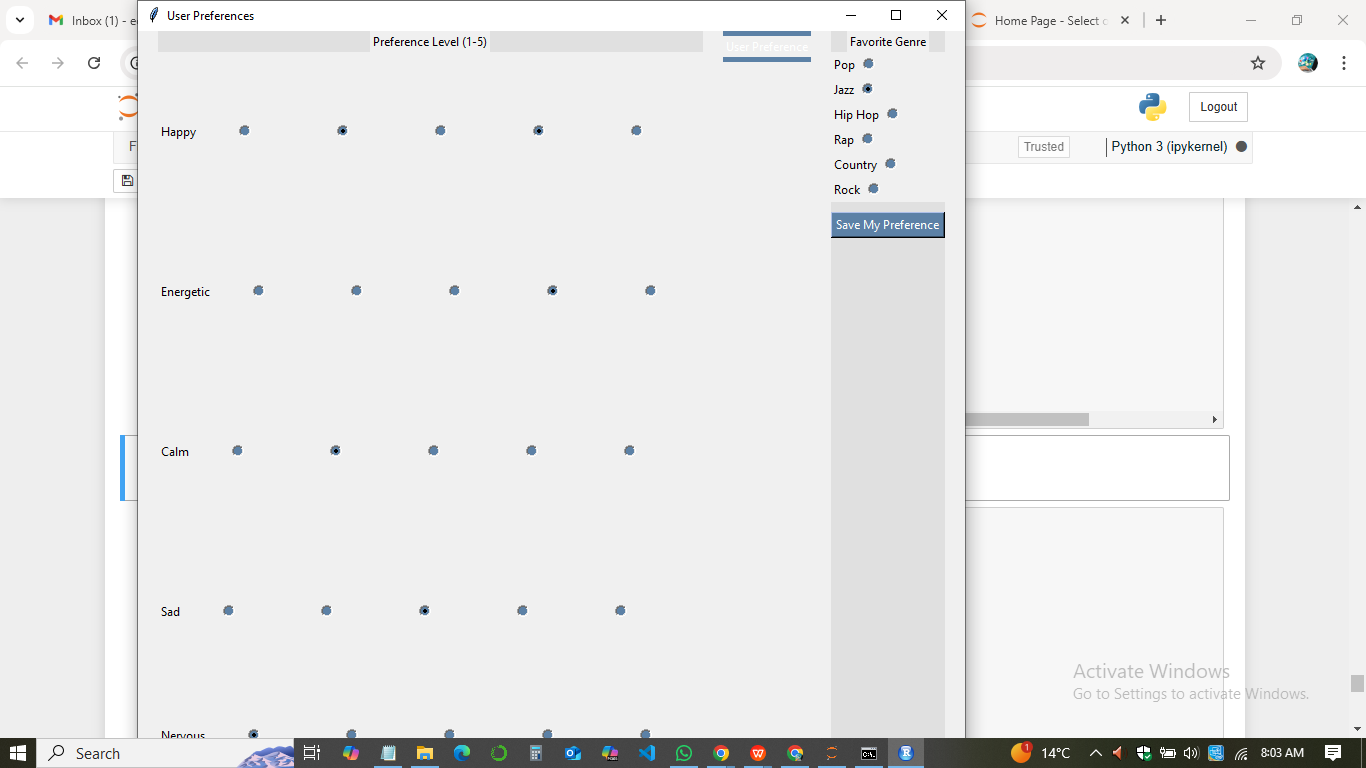




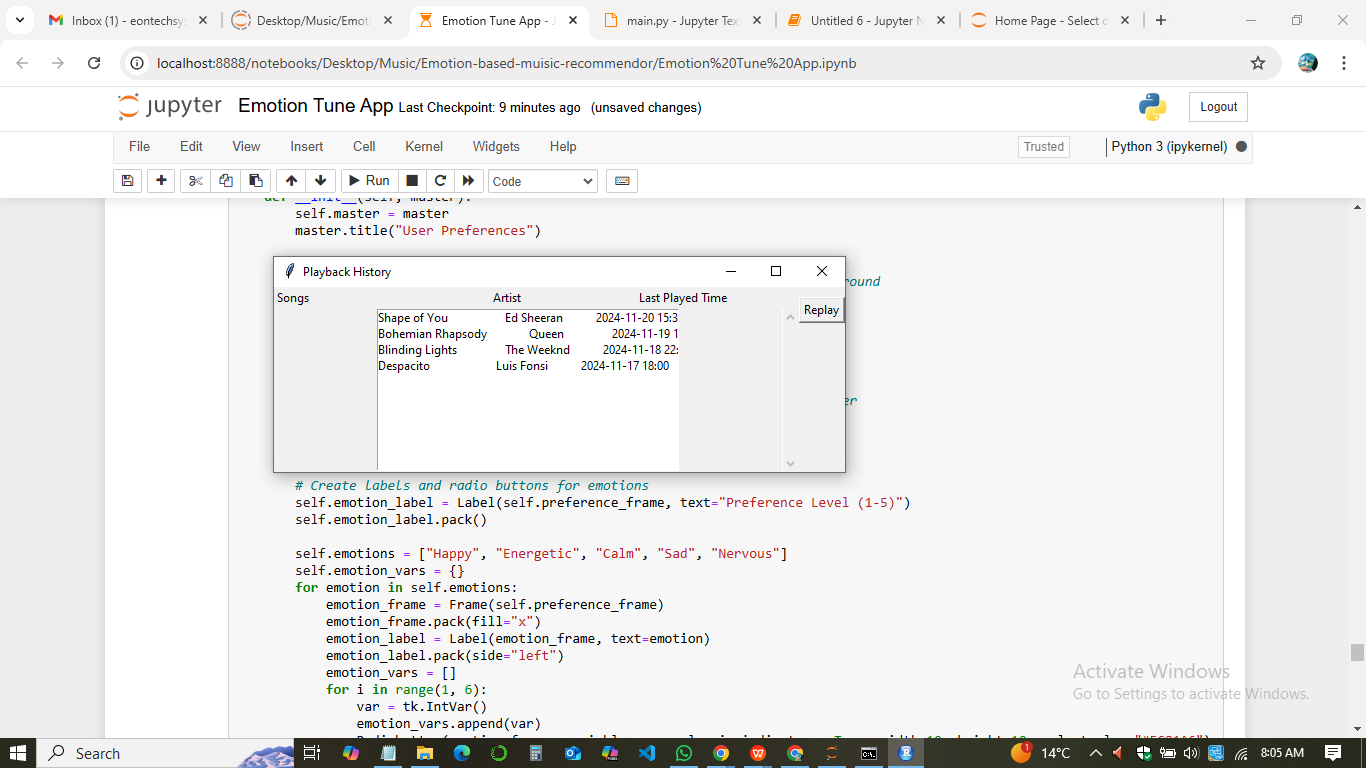
* **Home Screen (main music library view)**



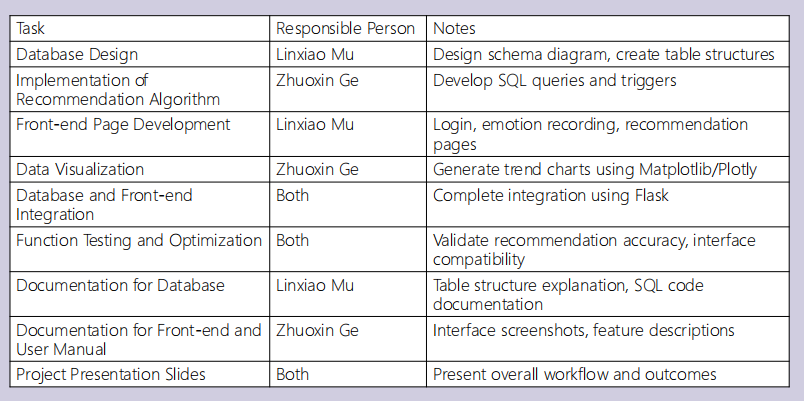
* **User Preferences Page**



* **Playback History Page**



**Team Member Contributions**



**References**

Van Rossum, G., & Drake, F. L. (2009). *Introduction to python 3: python documentation manual part 1*. CreateSpace.

<https://dl.acm.org/doi/abs/10.5555/1592885>

Venkatesan, V., & Hariharan, G. An Implementation Approach Towards The Automation Of Document Formatting Using Python. *International Journal of Aquatic Science*, *12*(02), 3946-3959.

<https://www.journal-aquaticscience.com/article_135903_3cb3777731dafc0976d56144e5e48a19.pdf>

Kreibich, J. (2010). *Using SQLite*. " O'Reilly Media, Inc.".

<https://books.google.co.ke/books?hl=en&lr=&id=HFIM47wp0X0C&oi=fnd&pg=PR7&dq=%EF%82%B7SQLite3+documentation&ots=Fj-BfYtm6R&sig=-aSiCWwEt3YKdt8hZrXSCEG2TsQ&redir_esc=y#v=onepage&q=%EF%82%B7SQLite3%20documentation&f=false>

Ronacher, A. (2021). *Flask documentation*.

<https://readthedocs.org/projects/flask-russian-docs/downloads/pdf/0.9/>

Dirix, M., Muller, A., & Aranega, V. (2013). Genmymodel: an online uml case tool. In *ECOOP*. <https://hal.science/hal-01251417/>