MOODSPHERE

AN EMOTION BASED MUSIC RECOMMENDER

AGENDA

- Team Member Roles and Responsibilities
- Improvements made from Professor Feedback
- Project Description
- Team-Work Agreement
- Personas
- MVP
- Technologies
- Algorithms
- Diagrams
- Sprint 3 Recap
- Product Backlog
- Sprint Summary (Sprint 2 + Sprint 3)
- Sprint 4 Backlog
- Metrics
- Retrospective
- Project Demo - Sprint 4 (current sprint)
- GitHub link
- Live Application Demo

TEAM

Bhavik Chopra

Data Scientist

Dhyey Dave

Lead Developer

Krushil Sheladiya

UI/UX Developer

Mahesh Nakka

Backend developer/QA

TEAM

Nisarg Bhuva

Scrum Master/QA

Shane Parmar

Backend Developer/Product Owner

Urmil Trivedi

Architect/Developer

Vijay Devkate

ML Engineer/ UI Designer

IMPROVEMENTS FROM FEEDBACK, CHANGES AND ADDITIONS

- **Added Sprint Summary(Sprint 2 + sprint 3)** *(Slide 43-51)*
- **Added Assignee in Product Backlog** *(Slide 37-42)*
- **Updated Sprint 4 Test Cases** *(Slide 55-57)*
- **Updated Stories Sprint 4 Completed and Not Completed** *(Slide 58-62)*
- **Added Sprint 4 Backlog** *(Slide 52-54)*
- **Updated Metrics** *(Slide 63-80)*
- **Updated Application Screenshots** *(Slide 85-97)*
- **Updated APIs** *(Slide 98-105)*

♫ This symbol represents the **Improvements, Changes or Additions** made.




PROJECT DESCRIPTION

Project Name:	MoodSphere
Team:	DevDynasty
Project Description:	Using facial expression detection techniques and machine learning algorithms in real time to understand the user's mood. **for** the music enthusiasts seeking a personalized and emotionally resonant listening experience, **who** wants to listen to their music based on their current emotional state **the** MoodSphere **is an** application that takes out the hassle of going through all the options in traditional music applications to hear the right music for their mood, that employs facial recognition and CNNs to analyze your real-time emotional expressions captured through the camera. **unlike** traditional music recommendation systems solely relying on user preferences, MoodSphere distinguishes itself by incorporating facial emotion analysis. **our** application dynamically adjusts music recommendations as your emotional expressions evolve allowing users to experience enhanced recommendations within the application.




PROBLEM STATEMENT

Conventional music streaming services frequently rely on general recommendation algorithms that ignore emotional relevance, which leaves users' emotional demands unmet. The Emotion-Based Music Recommender using Facial Recognition and Convolutional Neural Networks (CNN) project aims to combine state-of-the-art computer vision techniques in order to address this. The system uses CNNs and facial recognition technologies to accurately assess users' facial expressions for emotional indicators. The goal of this integration of sophisticated machine learning algorithms with visual data is to provide consumers with individualized music recommendations that are in line with their current emotional states, improving their listening experience as a whole.

TEAM-WORK AGREEMENT

CS691 - Teamwork Agreement (Team Dev Dynasty)

We at Dev Dynasty follow the seamless and triumphant completion in the project, where we are committed to the following principles and expectations. Team Collaboration, Proactiveness are classified by Involvement, Awareness, Task Allocation, and Time management. As, we members of Team Dev Dynasty take initiative based on individual skills, prioritise the capstone project success to ensure the completion.

INVOLVEMENT

- As the involvement is necessary we came to a point to meet In-person twice in 10 days which will make the team more stronger and collaborative.
- We shall agree that during the meet time or discussion about any crucial part all are requested to put their opinions and comments on what will be the best for the team to succeed in the outcome.
- The three moto's i.e. Trustworthiness, Truthfulness and Openness based on this value every individual shall or can have diverse perspectives, provide equal opportunity and a new ideology which can be developed towards great success of the project,instead of blaming people when issues occur.
- If the task assigned to the teammate gets undone or its tough to get complete then he must report and communicate to teammates via whatsapp.

AWARENESS

- As performing tasks if a teammate posts something which he's unaware the other if are available can react to it as quickly as possible.
- All should be aware that the discussion took place during the meeting hours and it's no one's responsibility to take care of each other; some exceptions might be taken in a state of medical emergency or sickness.
- We will communicate on every second day to keep updated about the task distributed and its individual responsibilities to openly ask for help if needed rather wasting time which he can't persist off. This can make a smooth process for the project to be on the right track.
- All the deliverables or the task based on theories will only be performed via Google doc which can be given access before submitting the final copy.
- Following the task distributions, planning for the upcoming sprint, next meeting times all will be followed on a single platform i.e. Jira.

Task allocation:

- Meeting via online or In-person for a daily standup which will eventually help teammates to get updated on every task.
- The project work should be distributed according to the individual knowledge and skill which can provide actual results and can help in problem solving.
- Based on the roles assigned if the teammates failed to perform correctly and failed to meet the deadline the scrum master has the right of decision making and make sure the task gets fulfilled or the teammate completes it in the next sprint.
- As respecting the privacy of every teammate, timing contact unless it's necessary regarding the project work.

Time Management

- The scrum master will make sure that the meeting links have been reached to everyone and everyone is readily available on same time for better coordination.
- We shall coordinate on each other's schedules to maintain consistency on working projects followed by a track on it after every discussion.

Team Member's	Email IDs
Urmil Trivedi	ut24256n@pace.edu
Bhavik Chopra	bc04992n@pace.edu
Dhyey Dave	dd28633n@pace.edu
Krushil Sheladiya	ks84830n@pace.edu
Shane Parmar	sp91003n@pace.edu
Vijay Devkate	vd19129n@pace.edu
Mahesh Nakka	mn01776n@pace.edu
Nisarg Bhuva	nb95325n@pace.edu

PERSONAS

PERSONA 1

MOODSPHERE

Elena

Technical Project Manager

Background:
Elena is a 32-year-old project manager in a tech company, often under high stress and pressure of completing project to deliver best outcomes.

Emotional Profile:
Given her high-stress role, Elena values an app that can detect subtle shifts in her mood through her interactions — be it the pace at which she scrolls or the time of day — and adjust the music without her needing to manually search for tracks. This anticipates her needs and saves time, allowing her to remain focused on her work or relaxation.

Preferred Genres:
Classical for relaxation, soft rock for mood uplift, ambient for focus.

Interests:
Yoga, meditation, and wellness podcasts.

PERSONA 2

MOODSPHERE

Max:

The Energetic Fitness Enthusiast

Background:
Max is a 27-year-old fitness trainer reliant on energetic music during workouts to stay energized and motivated.

Emotional Profile:
Seeks dynamic, high-tempo music for workouts and calming tunes for cooldowns. The app would suggest serene and tranquil music, aiding in the transition from high activity to relaxation. The app could potentially measure the duration of the workout to time the cooldown music perfectly.

Preferred Genres:
Upbeat EDM for workouts, soft instrumentals for cooldowns.

Interests:
High-intensity interval training, marathons, and sports nutrition.

PERSONA 3

Lia:

Remote Worker

Background:
Lia is a 20-year-old freelance writer who works from home on content on client's demand

Emotional Profile:
Her choice of music transcends mere background ambience, acting as a vital catalyst for her focus and creativity. It's this profound relationship with music that not only entertains but also significantly nurtures her artistic expression and emotional health. Each melody plays a specific role in her day, meticulously chosen to align with her feelings, thereby transforming her workspace into a sanctuary for inspiration and emotional balance.

Preferred Genres:
Lo-fi hip hop, smooth jazz, and classical.

Interests:
Creative writing, reading, and interior design.

PERSONA 4

MOODSPHERE

Sam:

The Commuter

Background:
Sam is a 35-year-old commuter who has long commutes and uses music to mentally prepare or unwind

Emotional Profile:
Sam needs energizing music in the morning and relaxing music in the evening. Sam also seeks music that can serve as a personal soundtrack to the rhythm of the city during his commutes, providing a sense of connection and vitality. In moments of stress or fatigue, he looks for tracks that can offer solace and a mental escape, helping to rejuvenate his spirit.

Preferred Genres:
Pop and indie for energy, acoustic for relaxation.

Interests:
Travel podcasts, audiobooks, and city walking tours.

PERSONA 5

MOODSPHERE

Zara:

An Aspiring Chef

Background:
Zara is a 22-year-old culinary student who enjoys cooking with music which makes her mood calm and focused.

Emotional Profile:
Zara seeks music that matches the energy of her cooking sessions
Zara enjoys music that keeps her kitchen lively and her mind sharp while she cooks. She also likes songs that go well with the different kinds of food she makes, making her cooking feel more special.

Preferred Genres:
Upbeat Latin for cooking, classical for baking.

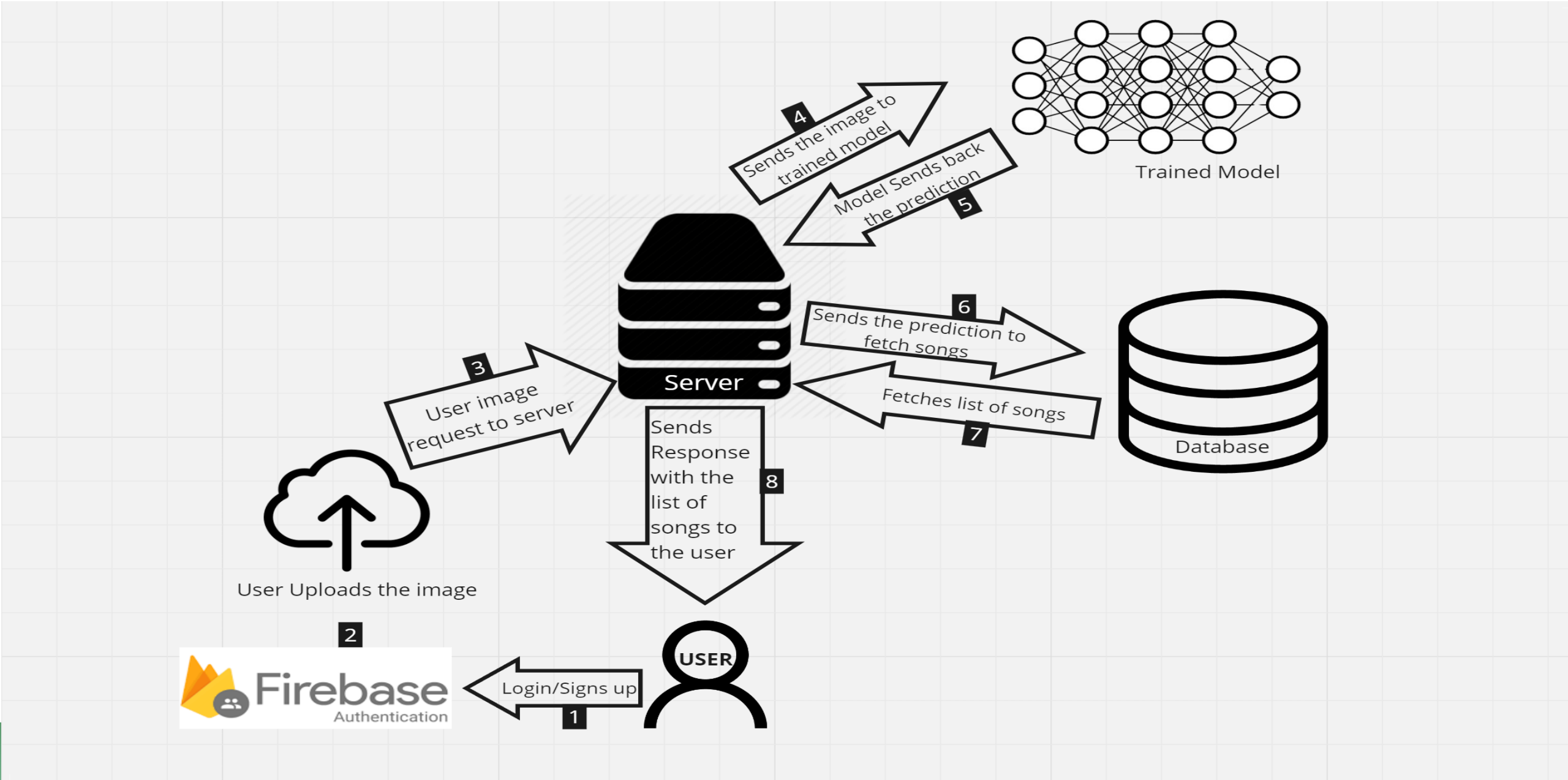
Interests:
World cuisines, food blogging, and cooking shows.

MINIMAL VIABLE PRODUCT(MVP)

MINIMUM VIABLE PRODUCT

- Web application with user emotion prediction and song recommendation functionality.
- Allowing user to create an account in the application.
- Interactive UI to display user uploaded image.
- CNN model to predict user's emotion on image input.
- API to accept user image in request and send the model's prediction and the list of songs from static database in response.
- Server to handle the above API along with two different endpoints to send songs based on genre and artist.
- Model and backend connection to predict emotions.

MVP DIAGRAM


Trained Model
4
Sends the image to trained model
Model Sends back the prediction
5
Server
6
Sends the prediction to fetch songs
Fetches list of songs
7
Database
3
User image request to server
Sends Response with the list of songs to the user
8
User Uploads the image
2
Firebase
Authentication
Login/Signs up
1
USER

MVP Experiment Canvas

Project name: MoodSphere

Made by: Dev Dynasty

Start date/time: 01/23/24

End date/time:05/07/24

1. Problem

- Users often struggle to find music that aligns with their current emotional state.
- Difficulty in understanding users' emotions through digital platforms.

2. Existing Alternatives

- Manual selection of songs based on mood or genre.
- Basic recommendation algorithms that suggest music based on listening history rather than current emotional state.

3. Solution

- A web application that predicts the user's emotion based on an uploaded image and recommends songs accordingly.
- Features include account creation, an interactive UI for displaying the uploaded image, and a CNN model for emotion prediction.
- The application will also offer an API to process the image, predict the emotion, and send back a list of song recommendations from a static database.

4. Unique Value Proposition

- Real-time emotion recognition from user-uploaded images to tailor music recommendations, enhancing the user experience by aligning music with their current emotional state.

5. Key Metrics

- User engagement rates (time spent on the application, number of songs listened to, etc.).
- Accuracy of emotion prediction and user satisfaction with song recommendations.

6. Channels

- Social media marketing targeting music enthusiasts and tech-savvy individuals.
- Collaborations with music forums and communities.

7. User Perspective Workflow

1. ***Account Setup***: Users sign up or log in, entering basic information.
2. ***Image Upload***: Users upload an image reflective of their current mood.
3. ***Waiting for Prediction***: The system processes the image, and users wait briefly.
4. ***Receiving Recommendations***: Based on the emotion detected, users are presented with a list of song recommendations.
5. ***Engaging with Music***: User will be allowed to click on song's external music links.

8. System Perspective Workflow

1. ***Manage Accounts:*** Securely handle user registrations and logins.
2. ***Process Image:*** Store and validate the uploaded image before processing.
3. ***Predict Emotion:*** Use a CNN model to identify the user's emotion from the image.
4. ***Retrieve Songs:*** Select songs matching the predicted emotion from a categorized database.
5. ***Serve Content:*** Offer additional endpoints for music exploration and ensure a responsive UI for user interaction.

9. Learnings and Insights

9a. Technical Learning

- ***AI and Machine Learning:*** Gain expertise in CNN models for emotion recognition, including data handling and model optimization.
- ***Web Development:*** Learn full-stack development, focusing on responsive UI design, API integration, and user data management.
- ***Deployment:*** Experience in deploying AI models in a web environment, emphasizing scalability and performance.

9b. Personal and Professional Growth

1. ***Project Management:*** Navigating the complexities of bringing an AI-driven product from concept to market enhances skills in project management, team collaboration, and agile development methodologies.
2. ***Adaptability and Problem-Solving:*** The challenges encountered during development and post-launch adjustments will foster adaptability and innovative problem-solving skills.
3. ***Networking and Collaboration:*** Engaging with users, investors, and other stakeholders offers opportunities for networking and may lead to collaborations or partnerships.

9c. Design and UX

UX/UI Design for AI Applications: Understanding how users interact with AI features and what design elements enhance their experience. This includes simplifying complex processes (like emotion prediction) into user-friendly interfaces.

TECHNOLOGIES

TECHNOLOGIES

TensorFlow

React

Firebase

Python

HTML5

CSS3

Flask

Jira

TECHNOLOGIES DESCRIPTION

TensorFlow will be the foundation for building the emotion detection engine of our music recommender.
Main applications of using TensorFlow will be Model Development and Real-Time Inference

React serves as the cornerstone for constructing the user interface (UI) of your music recommender app. Main applications of using React will be UI Design and Data Visualization

It is a suite of cloud-based services from Google that simplifies building mobile and web applications by providing features like: Authentication, Database Cloud Functions Hosting Analytics

Python supports OOP principles, allowing you to structure your code using classes and objects. This promotes code reusability, modularity, and maintainability, especially when dealing with complex data structures like user profiles and music recommendations.

TECHNOLOGIES DESCRIPTION

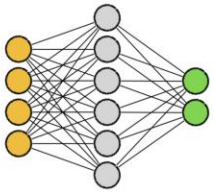
HTML5 Defines the structure and content of web pages. Creates the basic building blocks of web pages using elements like headings, paragraphs, lists, images, and forms.

CSS3 controls the presentation of web pages, applying visual styles like fonts, colors, layout, and animations.
It defines styles that are applied to HTML elements, controlling their appearance.

High-level web framework built in Python, simplifying the development process for building web applications.
Can be employed to create the backend of your application, handling data processing, routing, and interacting with Jira

Project management software specifically designed for agile teams, facilitating task tracking, issue reporting, and collaboration.

ALGORITHMS AND EXPLANATION



Convolutional neural networks (CNNs) are a form of deep neural network that are frequently used in computer vision tasks like object detection, picture segmentation, and image recognition. When the term "CNN algorithm" is used, it usually refers to CNNs.

ResNet-50
PRETRAINED MODEL

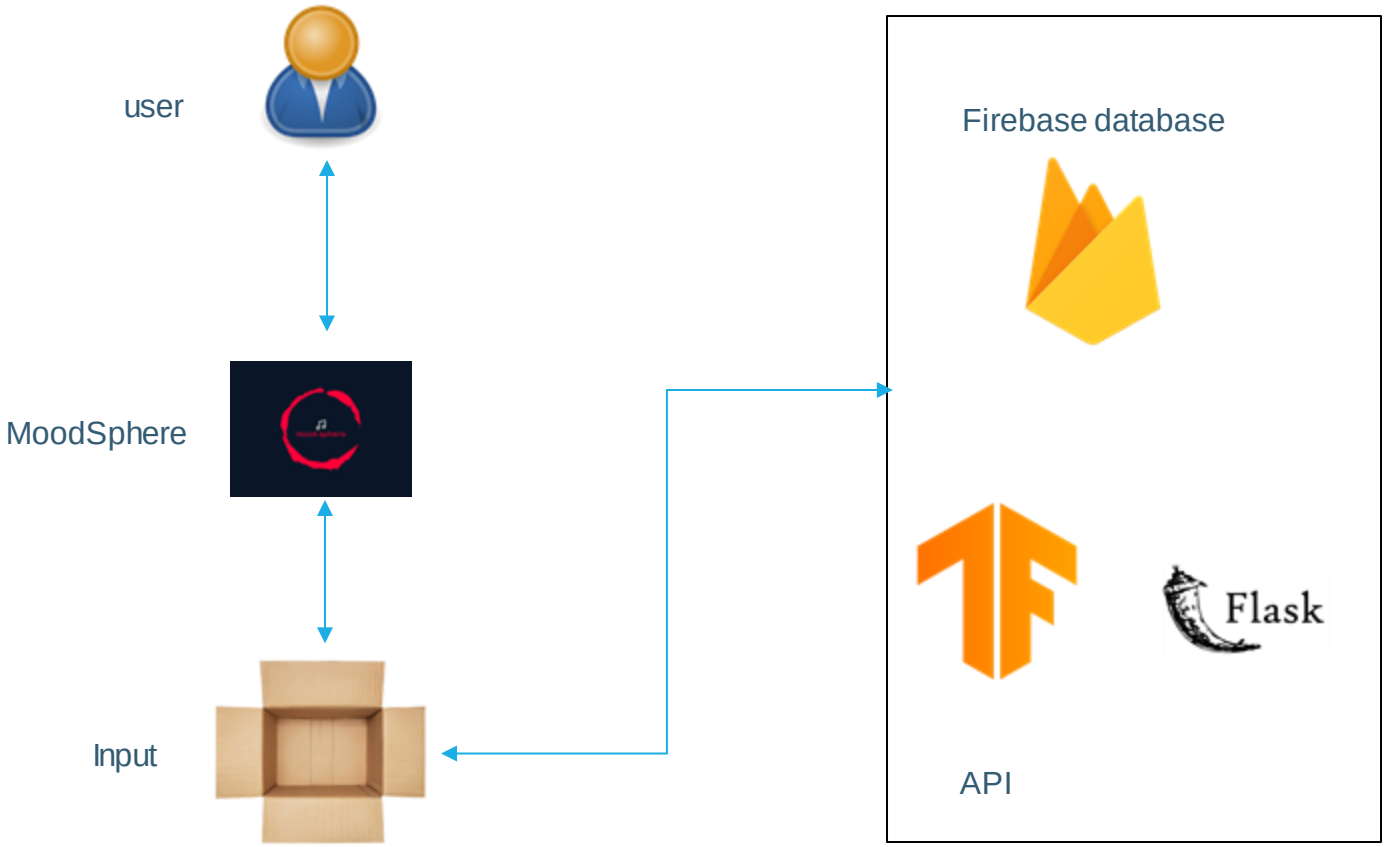
ResNet-50 is a deep convolutional neural network with 50 layers, renowned for its use of residual connections that aid in training very deep networks. Because of its outstanding performance in tasks like object identification, picture segmentation, and image classification, it has been frequently used in computer vision applications.

K

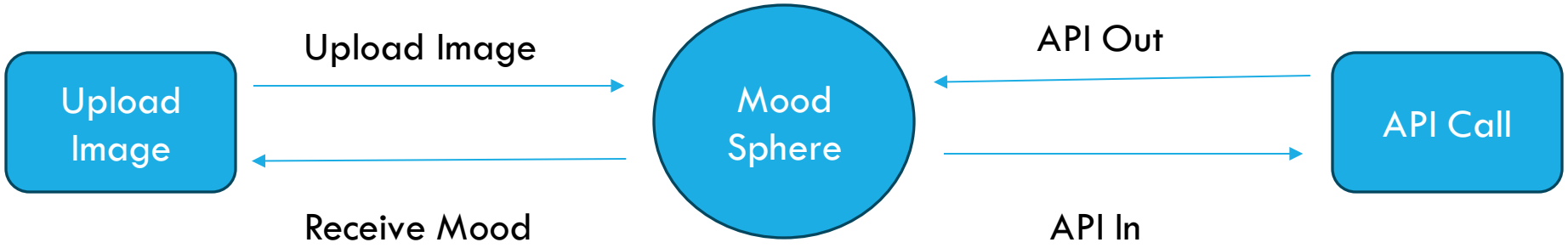
Keras is a popular Python framework that makes neural network construction and training (simple and straightforward). It provides an elevated interface to deep learning frameworks like as TensorFlow and Theano, facilitating swift model deployment and development.

DIAGRAMS

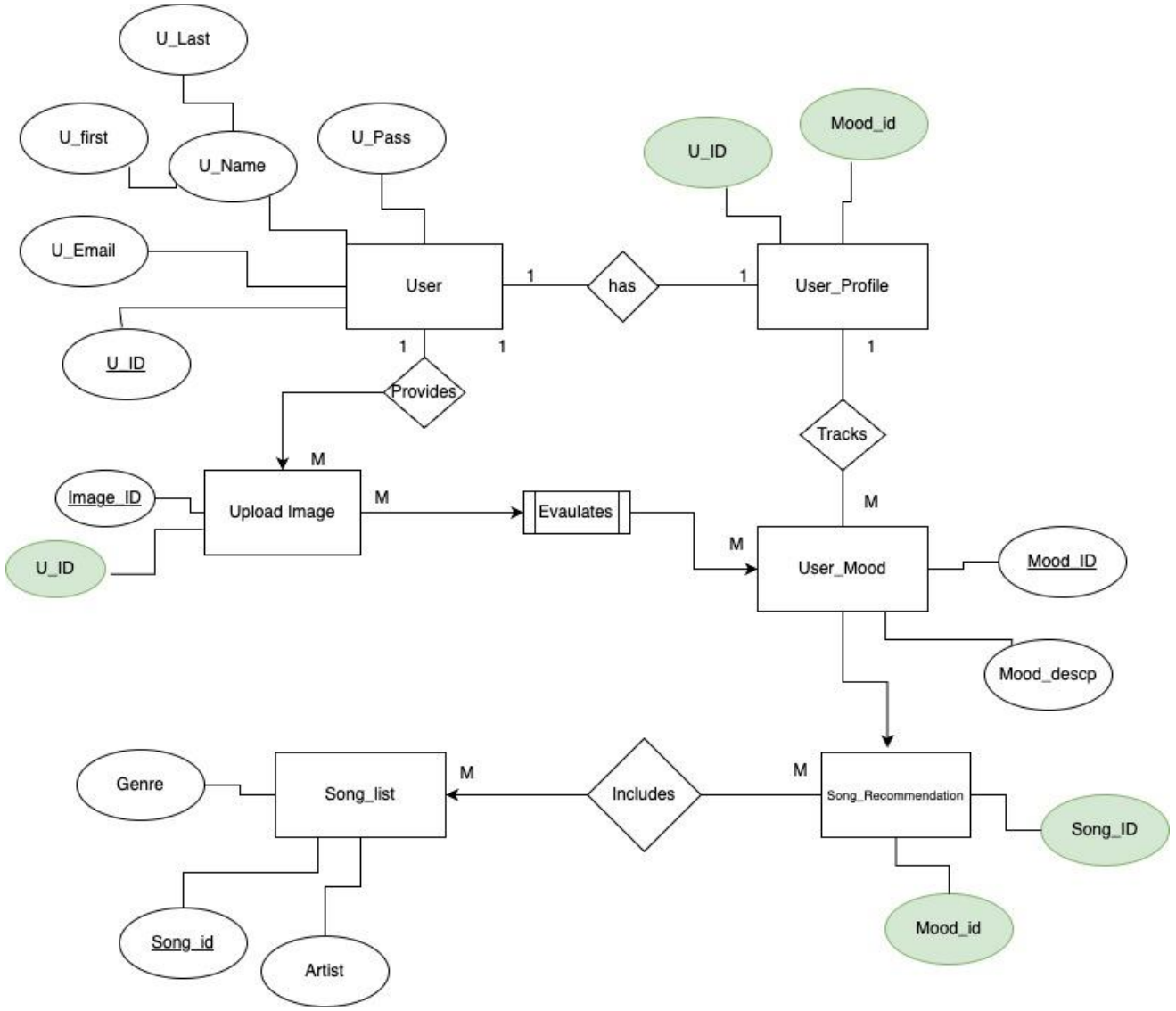
Architecture Diagram



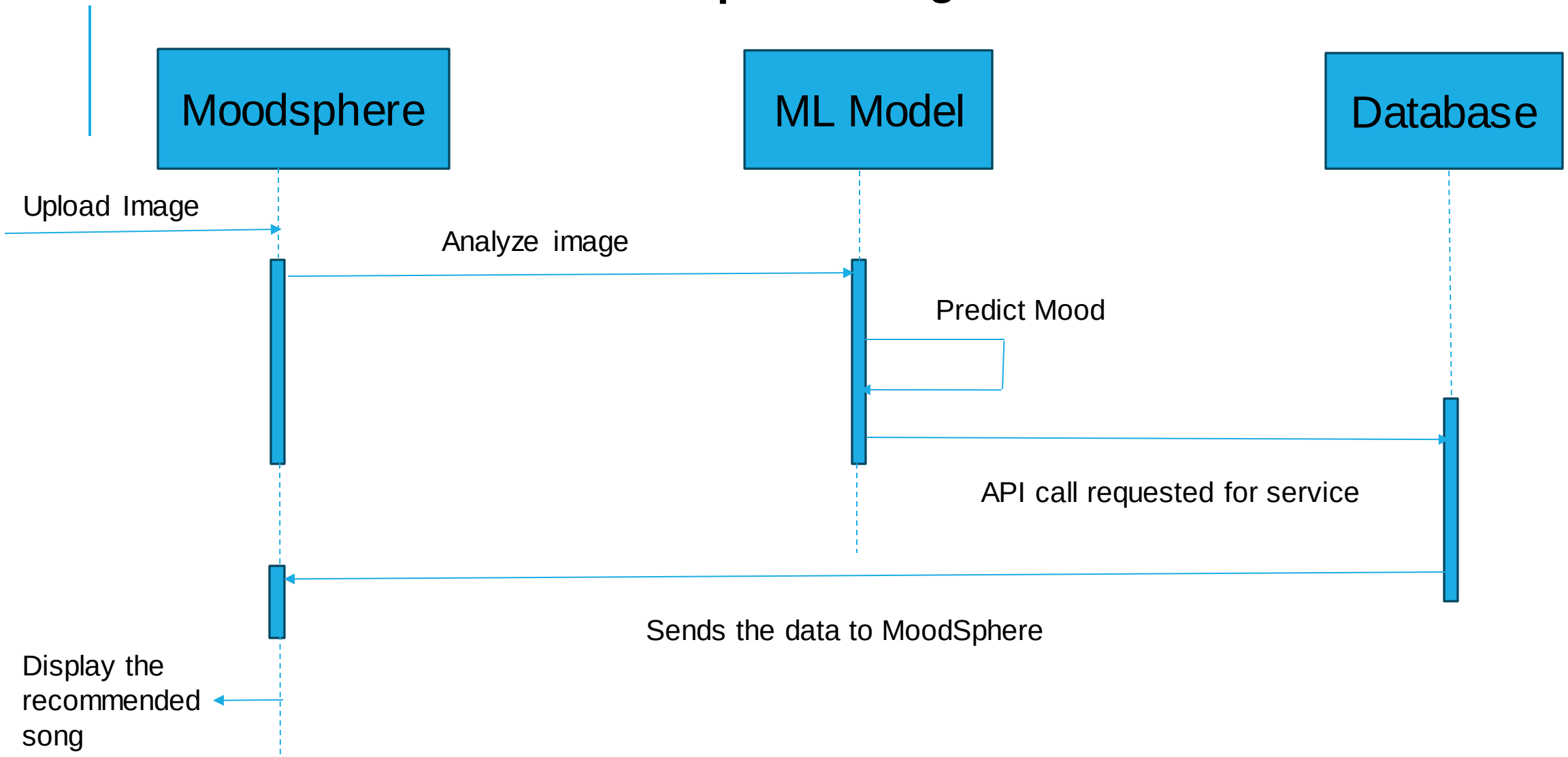
Context Diagram



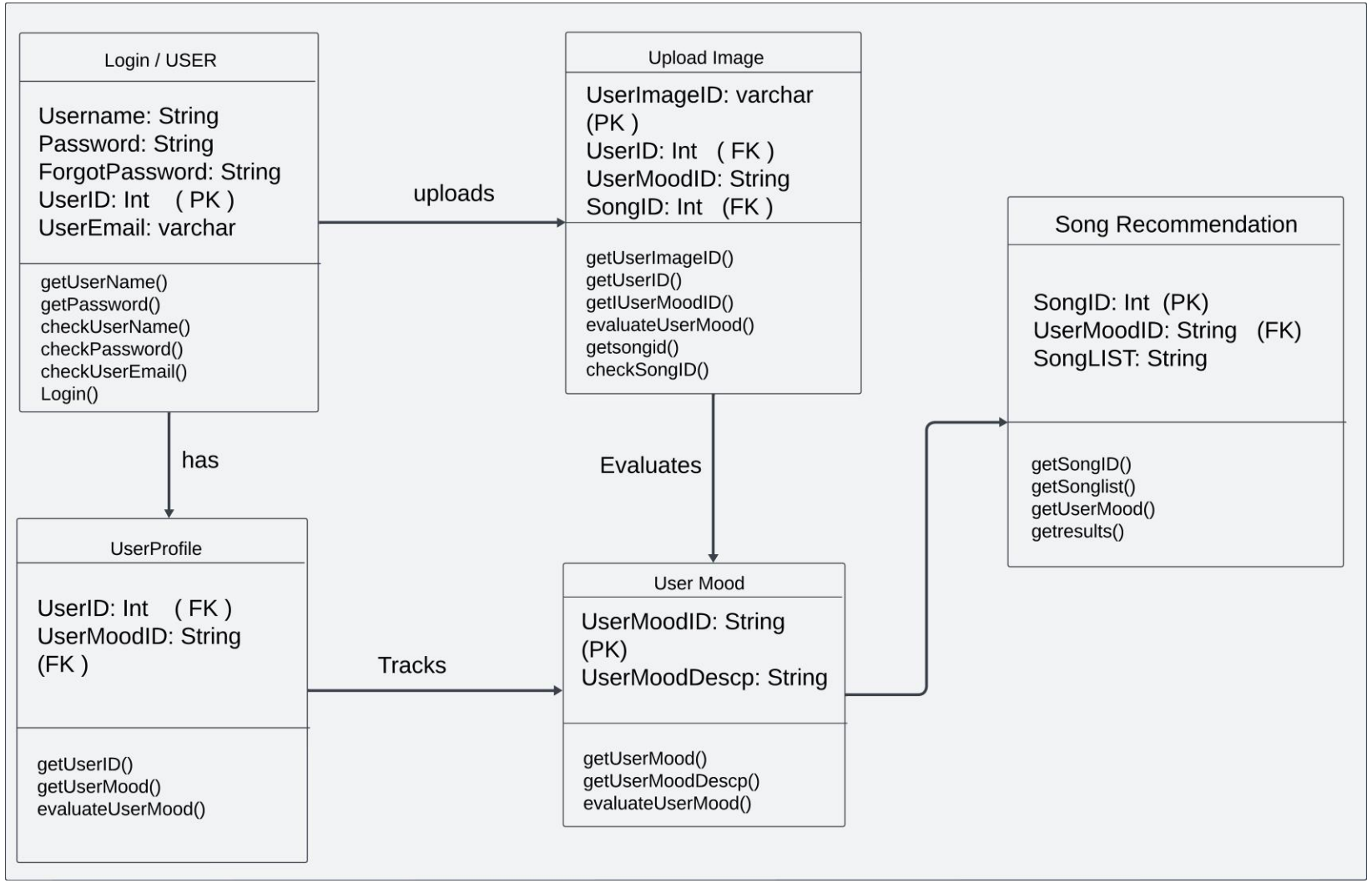
ENTITY RELATIONSHIP DIAGRAM



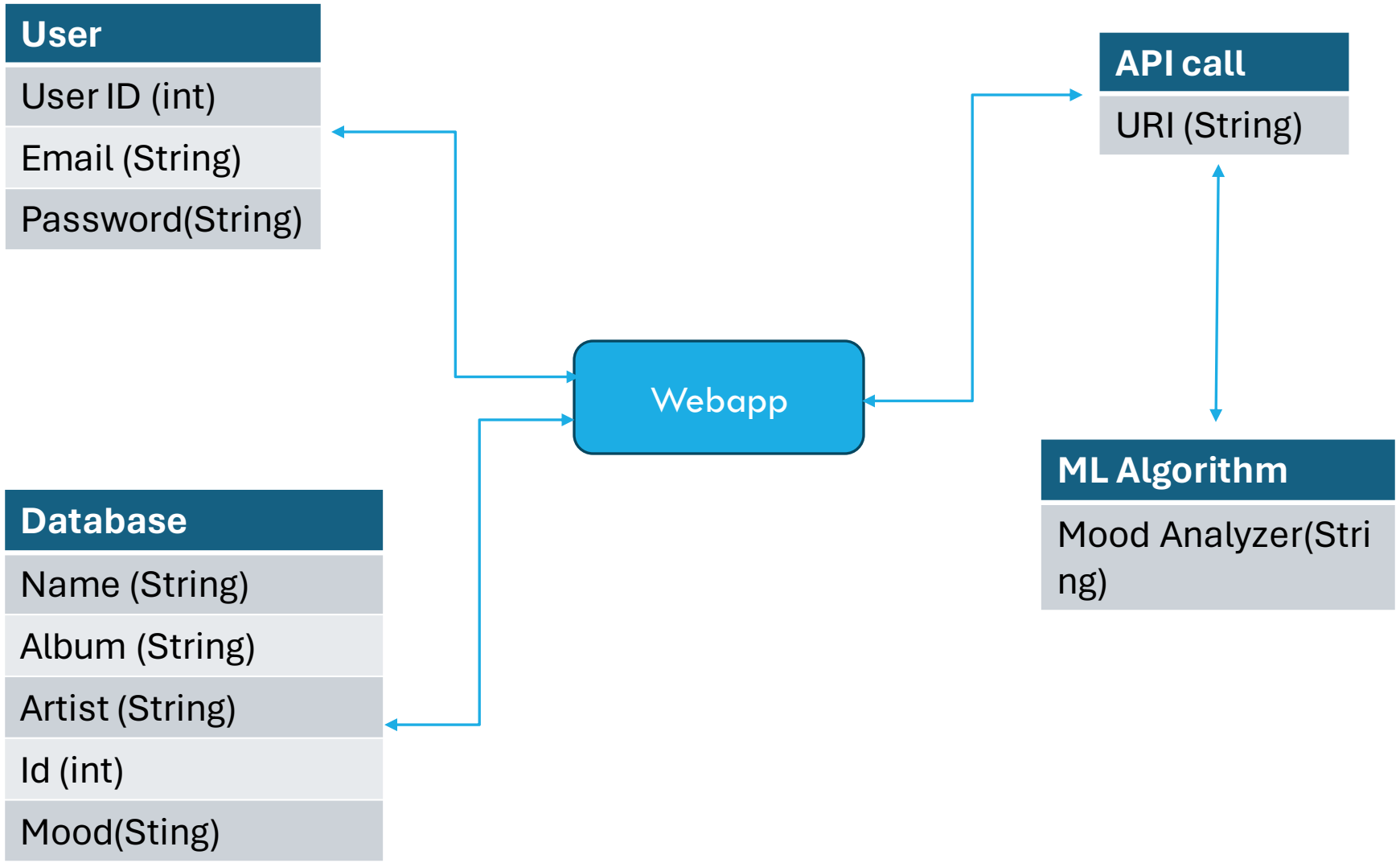
Sequence Diagram



STATE DIAGRAM



Class Diagram



SPRINT 3 RECAP

SPRINT 3 RECAP

Task ID	Story Points	User Story	Status	Sprint
MS_1	3	As a Music Enthusiast, I want to be able to upload a picture representing my current mood From the Already Saved Images in the gallery	In Progress	Sprint - 3
		Criteria, The user can upload an image file.Supported image formats include JPG, PNG, and GIF.		
MS_2	5	As a Music Enthusiast, I want the system to analyze the uploaded picture to determine my current mood More Precisely.	Done	Sprint - 3
		Criteria, The system should use a CNN model to extract features from the image.The model should classify the mood into predefined categories (e.g., happy, sad, angry) More Precisely		
MS_3	2	As a Music Enthusiast, I want to see the predicted mood based on the uploaded picture.	Done	Sprint - 3
		Criteria, The predicted mood should be displayed to the user.The mood prediction should be visually represented (e.g., emoticons).		
MS_6	3	As a Tech savvy, I want to be able to navigate the application easily and intuitively.	Done	Sprint - 3
		Criteria, The application should have clear navigation menus and buttons.Users should be able to easily switch between different sections of the application.		
MS_22	3	As a Music Enthusiast, I want the application to display the mood category along with each recommended song.	Done	Sprint - 3
		Criteria, Each recommended song entry should include the corresponding mood category.Mood categories should be clearly labeled and easy to distinguish.		
MS_23	5	As a song explorer, I want the application to provide recommendations based on my current mood, even if I'm not logged in or don't have an account.	In Progress	Sprint - 3
		Criteria, Users should be able to access mood-based recommendations without requiring authentication or account creation.		

SPRINT 3 RECAP

Task ID	Story Points	User Story	Status	Sprint
MS_7	5	As a tech savvy, I want the application to be responsive and work well on different devices, including desktops, tablets, and smartphones.	Done	Sprint - 3
		Criteria, The application should adapt to different screen sizes and resolutions.All features should be accessible and functional on various devices.		
MS_8	5	As a song explorer, I want to be able to search for specific songs or artists.	In Progress	Sprint - 3
		Criteria, Users can search for songs or artists using a search bar.Search results should be relevant and displayed in a clear manner.		
MS_12	5	As a tech savvy, I want the application to provide an option to sort recommended songs based on popularity or other criteria.	Done	Sprint - 3
		As a user, I want the application to provide an option to sort recommended songs based on popularity or other criteria.		
MS_16	3	As a song explorer, I want to be able to access external links associated with each recommended song.	Done	Sprint - 3
		Criteria, The list of songs should include external links in one column, allowing users to access additional information or listen to the song.		

SPRINT 3 RECAP

Sprint 3 Vs Sprint 4

Home Page
Created Home Page With Main Four Navigation Buttons.

MVP
User Can Upload/Capture Image, Predict Mood and Get Song recommendations

Find Music By Artists
Provided Songs Categorized By Artists

Profile Page
Enabled Users to View/Edit Their Personal Information

MVP Extension
User Can Retrieve Saved Images And use to predict mood and get song recommendations.

Find Music By Genre
Provided Songs Categorized By Genre

Playlist Creation
Users can Easily Create Multiple Playlists And add Songs

Feedback And Loading Indicator
Visual Cues And Feedback Mechanism For Users Was Added.

PRODUCT BACKLOG

PRODUCT BACKLOG

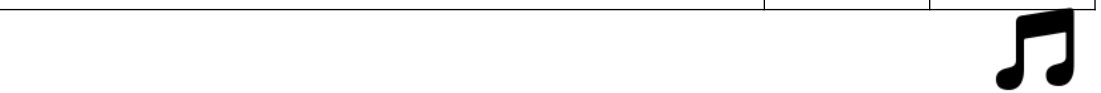
Story ID	Story Points	User Story	Assignee	Sprint
MS_4	3	As a Music Enthusiast, I want to receive personalized music recommendations based on my current mood.	Mahesh	**Sprint - 2**
		Criteria, The system should retrieve a list of songs associated with the predicted mood from a CSV file. Songs should be relevant and appropriate for the predicted mood.		
MS_9	5	As a Music Enthusiast, I want to be able to explore different moods and receive recommendations tailored to each mood.	Dhyey	**Sprint - 2**
		Criteria, Users should have the option to select different moods and receive corresponding song recommendations.Each mood selection should provide a unique set of recommended songs.		
MS_13	5	As a Music Enthusiast, I want to be able to view and edit my profile information.	Urmil	**Sprint - 2**
		Criteria, Users can view and update their profile information (e.g., name, email, profile picture).Profile editing should be straightforward and secure.		
MS_14	5	As a Traveller, I want to be able to sign up for an account to access personalized features and settings.	Dhyey	**Sprint - 2**
		Criteria, Users can create a new account by providing necessary information (e.g., email, password).Account creation should include email verification for security purposes.		
MS_18	5	As a Music Enthusiast, I want the application to provide a diverse range of song recommendations for each mood, including different genres and styles based on the popularity of the song.	Mahesh	**Sprint - 2**
		Criteria, Recommended songs should cover a variety of genres and musical preferences. Users should have options to explore different music styles within each mood category with the recommendation based on popularity.		

PRODUCT BACKLOG

Story ID	Story Points	User Story	Assignee	Sprint
MS_19	5	As a Wellness Seeker, I want the application to be secure and protect my privacy when uploading pictures or interacting with the site.	Urmil	Sprint - 2
		Criteria, The application should use HTTPS to encrypt data transmission. User data should be stored		
MS_26	3	As a Wellness Seeker, I want to view my current mood status without any song recommendations.	Krushil	Sprint - 2
		Criteria, Users can navigate to a separate page or section to view their current mood status. The mood status should be displayed visually (e.g., emoticons) along with a text description		
MS_1	3	As a music Enthusiast, I want to be able to upload a picture representing my current mood From the Already Saved Images in the gallery	Shane	Sprint - 3
		Criteria, The user can upload an image file. Supported image formats include JPG, PNG, and GIF.		
MS_2	5	As a Music enthusiast, I want the system to analyze the uploaded picture to determine my current mood More Precisely.	Mahesh	Sprint - 3
		Criteria, The system should use a CNN model to extract features from the image. The model should classify the mood into predefined categories (e.g., happy, sad, angry) More Precisely		
MS_3	2	As a Music enthusiast, I want to see the predicted mood based on the uploaded picture.	Urmil	Sprint - 3
		Criteria, The predicted mood should be displayed to the user. The mood prediction should be visually represented (e.g., emoticons).		

PRODUCT BACKLOG

Story ID	Story Points	User Story	Assignee	Sprint
MS_6	3	As a tech savvy, I want to be able to navigate the application easily and intuitively.	Vijay	Sprint - 3
		Criteria, The application should have clear navigation menus and buttons. Users should be able to easily switch between different sections of the application.		
MS_7	5	As a tech savvy, I want the application to be responsive and work well on different devices, including desktops, tablets, and smartphones.	Bhavik	Sprint - 3
		Criteria, The application should adapt to different screen sizes and resolutions. All features should be accessible and functional on various devices.		
MS_8	5	As a song explorer, I want to be able to search for specific songs or artists.	Dhyey	Sprint - 3
		Criteria, Users can search for songs or artists using a search bar. Search results should be relevant and displayed in a clear manner.		
MS_12	5	As a tech savvy, I want the application to provide an option to sort recommended songs based on popularity or other criteria.	Mahesh	Sprint - 3
		Criteria, users want the application to provide an option to sort recommended songs based on popularity or other criteria.		
MS_16	3	As a song explorer, I want to be able to access external links associated with each recommended song.	Nisarg	Sprint - 3
		Criteria, The list of songs should include external links in one column, allowing users to access additional information or listen to the song.		



PRODUCT BACKLOG

Story ID	Story Points	User Story	Assignee	Sprint
MS_22	3	As a Music Enthusiast, I want the application to display the mood category along with each recommended song.	Shane	Sprint - 3
		Criteria, Each recommended song entry should include the corresponding mood category. Mood categories should be clearly labeled and easy to distinguish.		
MS_23	5	As a Song explorer, I want the application to provide recommendations based on my current mood, even if I'm not logged in or don't have an account.	Urmil	Sprint - 3
		Criteria, Users should be able to access mood-based recommendations without requiring authentication or account creation.		
MS_5	5	As a Music Enthusiast, I want to be able to save songs I like to a playlist for future reference.	Dhyey	Sprint - 4
		Criteria, Users should have the option to add songs to a personal playlist. The playlist should persist across sessions for registered users.		
MS_10	3	As a Music Enthusiast, I want to be able to create and customize multiple playlists for different moods or occasions.	Mahesh	Sprint - 4
		Criteria, Users can create new playlists and give them custom names. Users can add or remove songs from playlists.		
MS_11	5	As a Song explorer, I want the application to provide additional information about recommended songs	Krushil	Sprint - 4
		Criteria, Users can access additional information about recommended songs by clicking on them. Information should be presented in a clear and concise manner.		

PRODUCT BACKLOG

Story ID	Story Points	User Story	Assignee	Sprint
MS_15	5	As a Tech savvy, I want to be able to provide feedback on recommended songs or report any issues encountered while using the application.	Vijay	Sprint - 4
		Criteria, Users should have a feedback mechanism to submit comments or report problems		
MS_17	3	As a Song Explorer, I want to be able to clear my mood selection and start over if I'm not satisfied with the recommended songs.	Dhyey	Sprint - 4
		Criteria, Users should have the option to clear their mood selection and return to the initial state. Clearing the mood selection should reset the recommended songs.		
MS_20	3	As a tech savvy, I want the application to provide clear instructions and guidance on how to use its features effectively.	Nisarg	Sprint - 4
		Criteria, The application should include help documentation or tooltips to explain its functionality.		
MS_21	3	As a tech savvy, I want the application to display loading indicators or progress bars to indicate when mood analysis and song retrieval are in progress.	Shane	Sprint - 4
		Criteria, Users should see visual cues indicating that the application is processing their request.Loading indicators should be displayed prominently and disappear once processing is complete.		
MS_24	3	As a song Explorer, I want the application to store uploaded pictures for future reference.	Mahesh	Sprint - 4
		Criteria, Uploaded pictures should be saved securely in the user's account.Users should be able to access their stored pictures from their profile or settings.		

PRODUCT BACKLOG

Story ID	Story Points	User Story	Assignee	Sprint
MS_25	3	As a Song explorer, I want the ability to delete uploaded pictures if needed.	Dhyey	**Sprint - 4**
		Criteria, Users can delete uploaded pictures from their account. Deletion should be confirmed with a confirmation prompt to prevent accidental deletion.		

SPRINT SUMMARY

SPRINT-2 USER STORIES & BURNDOWN CHART

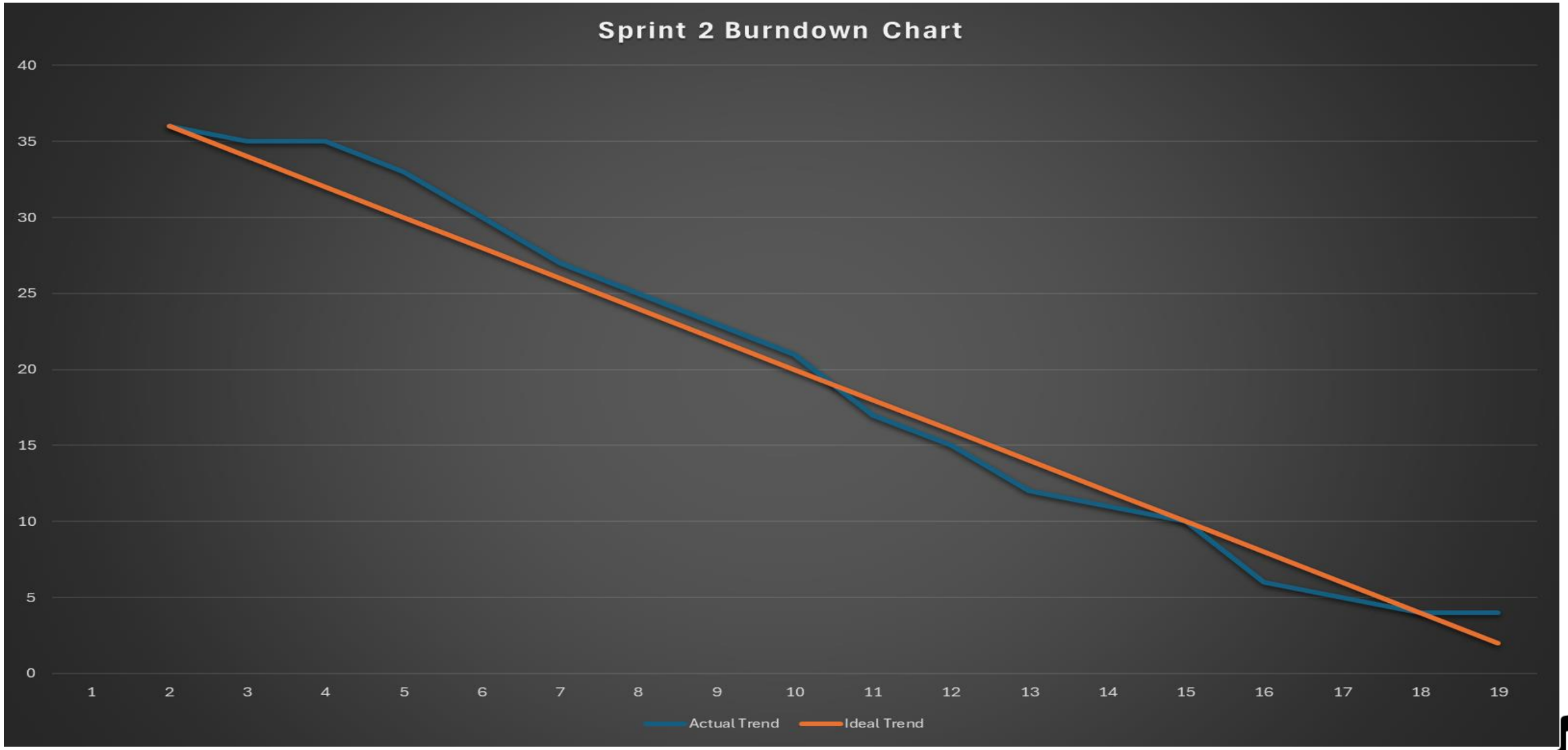
SPRINT 2 USER STORIES

Story ID	Story Points	User Story	Status	Sprint
MS_4	3	As a Music Enthusiast, I want to receive personalized music recommendations based on my current mood.	Done	**Sprint - 2**
		Criteria, The system should retrieve a list of songs associated with the predicted mood from a CSV file. Songs should be relevant and appropriate for the predicted mood.		
MS_9	5	As a Music Enthusiast, I want to be able to explore different moods and receive recommendations tailored to each mood.	Done	**Sprint - 2**
		Criteria, Users should have the option to select different moods and receive corresponding song recommendations. Each mood selection should provide a unique set of recommended songs.		
MS_13	5	As a Music Enthusiast, I want to be able to view and edit my profile information.	Done	**Sprint - 2**
		Criteria, Users can view and update their profile information (e.g., name, email, profile picture).Profile editing should be straightforward and secure.		
MS_14	5	As a Traveller, I want to be able to sign up for an account to access personalized features and settings.	Done	**Sprint - 2**
		Criteria, Users can create a new account by providing necessary information (e.g., email, password). Account creation should include email verification for security purposes.		
MS_18	5	As a Music Enthusiast, I want the application to provide a diverse range of song recommendations for each mood, including different genres and styles based on the popularity of the song.	Done	**Sprint - 2**
		Criteria, Recommended songs should cover a variety of genres and musical preferences.Users should have options to explore different music styles within each mood category with the recommendation based on popularity.		

SPRINT 2 USER STORIES

Story ID	Story Points	User Story	Status	Sprint
MS_19	5	As a wellness seeker, I want the application to be secure and protect my privacy when uploading pictures or interacting with the site.	Done	**Sprint - 2**
		Criteria, The application should use HTTPS to encrypt data transmission. User data should be stored securely and not shared with third parties without consent.		
MS_26	3	As a user, I want to view my current mood status without any song recommendations.	In Progress	**Sprint - 2**
		Criteria, Users can navigate to a separate page or section to view their current mood status.The mood status should be displayed visually (e.g., emoticons) along with a text description		

Sprint 2 Burndown Chart



SPRINT-3 USER STORIES & BURNDOWN CHART

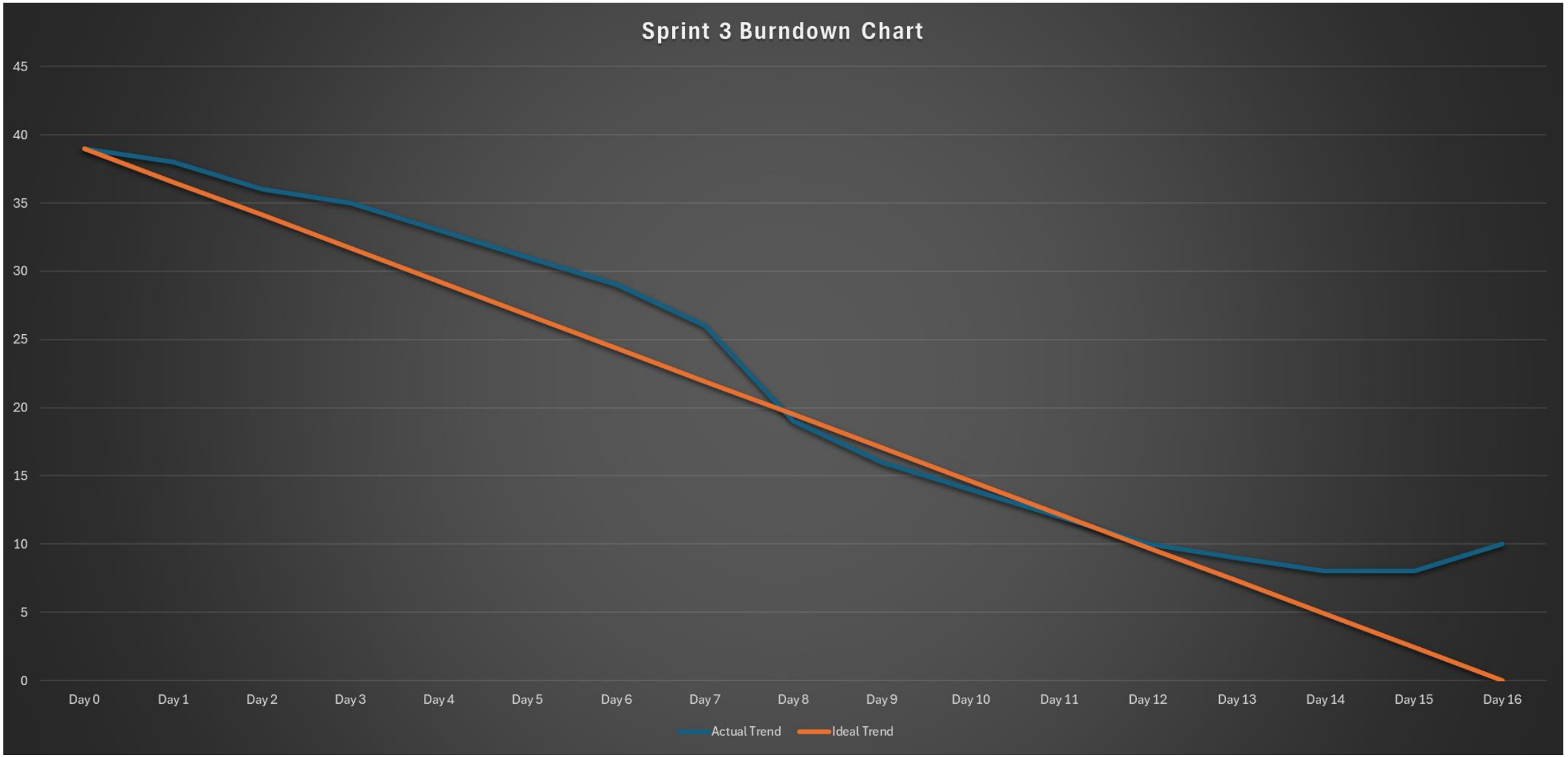
SPRINT 3 USER STORIES

Story ID	Story Points	User Story	Status	Sprint
MS_1	3	As a Music Enthusiast, I want to be able to upload a picture representing my current mood From the Already Saved Images in the gallery	In Progress	Sprint - 3
		Criteria, The user can upload an image file. Supported image formats include JPG, PNG, and GIF.		
MS_2	5	As a Music Enthusiast, I want the system to analyze the uploaded picture to determine my current mood More Precisely.	Done	Sprint - 3
		Criteria, The system should use a CNN model to extract features from the image. The model should classify the mood into predefined categories (e.g., happy, sad, angry) More Precisely		
MS_3	2	As a Music Enthusiast, I want to see the predicted mood based on the uploaded picture.	Done	Sprint - 3
		Criteria, The predicted mood should be displayed to the user.The mood prediction should be visually represented (e.g., emoticons).		
MS_6	3	As a tech savvy, I want to be able to navigate the application easily and intuitively.	Done	Sprint - 3
		Criteria, The application should have clear navigation menus and buttons.Users should be able to easily switch between different sections of the application.		
MS_22	3	As a Music Enthusiast, I want the application to display the mood category along with each recommended song.	Done	Sprint - 3
		Criteria, Each recommended song entry should include the corresponding mood category.Mood categories should be clearly labeled and easy to distinguish.		

SPRINT 3 USER STORIES

Story ID	Story Points	User Story	Status	Sprint
MS_23	5	As a Song Explorer, I want the application to provide recommendations based on my current mood, even if I'm not logged in or don't have an account.	In Progress	Sprint - 3
		Criteria, Users should be able to access mood-based recommendations without requiring authentication or account creation.		
MS_7	5	As a tech savvy, I want the application to be responsive and work well on different devices, including desktops, tablets, and smartphones.	Done	Sprint - 3
		Criteria, The application should adapt to different screen sizes and resolutions.All features should be accessible and functional on various devices.		
MS_8	5	As a song explorer, I want to be able to search for specific songs or artists.	In Progress	Sprint - 3
		Criteria, Users can search for songs or artists using a search bar.Search results should be relevant and displayed in a clear manner.		
MS_12	5	As a tech savvy, I want the application to provide an option to sort recommended songs based on popularity or other criteria.	Done	Sprint - 3
		As a user, I want the application to provide an option to sort recommended songs based on popularity or other criteria.		
MS_16	3	As a song explorer, I want to be able to access external links associated with each recommended song.	Done	Sprint - 3
		Criteria, The list of songs should include external links in one column, allowing users to access additional information or listen to the song.		

Sprint 3 Burndown Chart



SPRINT-4 BACKLOG

SPRINT 4 BACKLOG

Story ID	Story Points	User Story	Status	Sprint
MS_5	5	As a Music Enthusiast, I should have an option to add songs to a personal playlist	Done	**Sprint - 4**
		Criteria, Users should have the option to add songs to a personal playlist. The playlist should persist across sessions for registered users.		
MS_10	3	As a Music Enthusiast, I want to be able to create and customize multiple playlists for different moods or occasions.	Done	**Sprint - 4**
		Criteria, Users can create new playlists and give them custom names. Users can add or remove songs from playlists.		
MS_11	5	As a Song explorer, I want the application to provide additional information about recommended songs	Done	**Sprint - 4**
		Criteria, Users can access additional information about recommended songs by clicking on them. Information should be presented in a clear and concise manner.		
MS_21	3	As a Tech savvy, I want the application to display loading indicators or progress bars to indicate when mood analysis and song retrieval are in progress.	Done	**Sprint - 4**
		Criteria, Users should see visual cues indicating that the application is processing their request. Loading indicators should be displayed prominently and disappear once processing is complete.		
MS_24	3	As a Tech savvy, I want the application to store uploaded pictures for future reference.	Done	**Sprint - 4**
		Criteria, Uploaded pictures should be saved securely in the user's account. Users should be able to access their stored pictures from their profile or settings.		

SPRINT 4 BACKLOG

Story ID	Story Points	User Story	Status	Sprint
MS_15	5	As a tech savvy, I want to be able to provide feedback on recommended songs or report any issues encountered while using the application.	Done	**Sprint - 4**
		Criteria, Users should have a feedback mechanism to submit comments or report problems or problems.		
MS_17	3	As a song explorer, I want to be able to clear my mood selection and start over if I'm not satisfied with the recommended songs.	Done	**Sprint - 4**
		Criteria, Users should have the option to clear their mood selection and return to the initial state. Clearing the mood selection should reset the recommended songs.		
MS_20	3	As a tech savvy, I want the application to provide clear instructions and guidance on how to use its features effectively.	Done	**Sprint - 4**
		Criteria, The application should include help documentation or tooltips to explain its functionality.		
MS_25	3	As a song explorer, I want the ability to delete uploaded pictures if needed.	In Progress	**Sprint - 4**
		Criteria, Users can delete uploaded pictures from their account. Deletion should be confirmed with a confirmation prompt to prevent accidental deletion.		

Total Story Points: 33

TESTCASES FOR SPRINT-4 BACKLOG

SPRINT 4 TEST CASES

Test ID	Story ID	Test Scenario	Test Case	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
TC_MS21_01	**MS_21**	Verify display of loading indicators during mood analysis and song retrieval.	Displaying Loading Indicators	1.Trigger the mood analysis or song retrieval process by uploading an image. 2.Observe the application interface for loading indicators or progress bars. 3.Monitor the progress until the analysis or retrieval is complete.	JPG/PNG File	The application should visually indicate progress with loading indicators during data processing.	Can see the Loading indicator until it fetches the songs.	pass
TC_MS8_01	**MS_8**	Verify the functionality to search for specific songs or artists.	Searching for Songs and Artists	1.Access the search feature within the application. 2. Enter a song title or artist name in the search bar, Submit the search query to check results	Song titles, artist names.	The application should return accurate search results based on the entered query if it is in the Database.	Should show the respective songs based on the search and the existence of the data.	pass
TC_MS5_01	**MS_5**	Verify the ability to save liked songs to a playlist.	Saving Liked Songs in the recommendation.	1.Select a song from the recommended list. 2.Locate the option(+) to add the liked song to a playlist. 3.Choose a playlist to add the song or create one and confirm that the song has been successfully saved to the playlist.	Liked songs, playlist creation.	The selected song should be saved and accessible within the designated playlist.	Can see the Song in the respective playlist	pass
TC_MS10_01	**MS_10**	Verify the creation and customization of multiple playlists.	Creating Custom Playlists	1.Navigate to the playlist section and Check for the delete option on the existing playlist. 2.Select a song from the recommended list and create multiple playlists.	Playlist names	The application should allow the user to create and delete multiple playlists based on different moods or occasions.	Can be able to delete and create multiple playlists.	pass
TC_MS15_01	**MS_15**	Verify the ability to provide feedback or report issues on application	Providing Feedback or reporting an issue	1.Locate the report option on the home page. 2.Click on the Report and write your issue and click on the submit button.	Song recommendation feedback, issue reporting.	The application should accept users input and send an email to the development mail address.	Take the submitted and send an email to the applications email ID.	pass

SPRINT 4 TEST CASES

Test ID	Story ID	Test Scenario	Test Case	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
TC_MS23_01	**MS_23**	Verify recommendations based on user's current mood without logging in.	Recommendations for Anonymous Users	1. Open the application without logging into any user account. 2. Locate the mood Status feature on the homepage. 3. Click on the upload button to upload your image and check the recommendations	Navigating without entering logging information to upload the picture	The application should successfully recommend songs based on the selected mood, even for anonymous users.	Uploaded Image is successfully passed to the CNN model and getting the recommendation.	pass
TC_MS17_01	**MS_17**	Verify the ability to clear mood selection(Remove Button) and start over.	Clearing Mood Selection	1.Navigate to the mood Status feature within the application. 2. Upload your image and get the recommendations. 3. Click on the Remove button to clear the Recommendations to start over.	Navigate to the Mood Status Feature to find the Remove button	The selected mood should be cleared, allowing the user to choose a new mood for song recommendations.	Recommendations are successfully cleared.	pass
TC_MS24_01	**MS_24**	Verify uploaded pictures are stored for future reference.	Storing Uploaded Pictures	1.Navigate to the mood Status feature within the application. 2. Upload your image and get the recommendations. 3.Check for the pop-up Image uploaded to FireStore	JPG/PNG File	The uploaded pictures should be Stored in DB and accessible after logging out and logging back into the application.	Can Access the image from DB	pass
TC_MS12_01	**MS_12**	Verify sorting of recommended songs based on popularity.	Sorted Recommended Songs	Check if the Recommended Songs are Displayed in Ascending order with our DataSet.	List of Recommended Songs	Songs being displayed in ascending order	Navigation buttons are working	pass
TC_MS11_01	**MS_11**	Verify access to additional information about recommended songs.	Accessing Song Information	1.Choose a recommended song from the list. 2.Look for option(i) to access additional information (e.g., artist details, album info).	Recommended songs.	The application should provide more information about the selected song upon clicking the option(i).	Can clearly be able to see addition information from initial results.	pass
TC_MS20_01	**MS_20**	Verify clarity of instructions and guidance within the application.	Checking Instructions	Navigate to home and click on the option User Guide.	User interaction with the application.	The application should offer User instructions and guide on using its features effectively.	Can clearly be able to navigate and see the results	pass

COMPLETED STORIES IN SPRINT-4

Sprint 4 completed Stories

Story ID	Story Points	User Story	Status	Sprint
MS_1	3	As a Music Enthusiast, I want to be able to upload a picture representing my current mood From the Already Saved Images in the gallery	Done	Sprint - 4
		Criteria, The user can upload an image file. Supported image formats include JPG, PNG, and GIF.		
MS_8	5	As a Song Explorer, I want to be able to search for specific songs or artists.	Done	Sprint - 4
		Criteria, Users can search for songs or artists using a search bar. Search results should be relevant and displayed in a clear manner.		
MS_23	5	As a Song Explorer, I want the application to provide recommendations based on my current mood, even if I'm not logged in or don't have an account.	Done	Sprint - 4
		Criteria, Users should be able to access mood-based recommendations without requiring authentication or account creation.		
MS_5	5	As a Music Enthusiast, I want to be able to save songs I like to a playlist for future listening.	Done	Sprint - 4
		Criteria, Users should have the option to add songs to a personal playlist. The playlist should persist across sessions for registered users.		
MS_10	3	As a Music Enthusiast, I want to be able to create and customize multiple playlists for different moods or occasions.	Done	Sprint - 4
		Criteria, Users can create new playlists and give them custom names. Users can add or remove songs from playlists.		
MS_11	5	As a Song Explorer, I want the application to provide additional information about recommended songs	Done	Sprint - 4
		Criteria, Users can access additional information about recommended songs by clicking on them. Information should be presented in a clear and concise manner.		

Sprint 4 completed Stories

Story ID	Story Points	User Story	Status	Sprint
MS_15	5	As a Tech savvy, I want to be able to provide feedback on recommended songs or report any issues encountered while using the application.	Done	**Sprint - 4**
		Criteria, Users should have a feedback mechanism to submit comments or report problems.		
MS_17	3	As a Song Explorer, I want to be able to clear my mood selection and start over if I'm not satisfied with the recommended songs.	Done	**Sprint - 4**
		Criteria, Users should have the option to clear their mood selection and return to the initial state. Clearing the mood selection should reset the recommended songs.		
MS_20	3	As a Tech savvy, I want the application to provide clear instructions and guidance on how to use its features effectively.	Done	**Sprint - 4**
		Criteria, The application should include help documentation or tooltips to explain its functionality.		
MS_21	3	As a Song Explorer, I want the application to display loading indicators or progress bars to indicate when mood analysis and song retrieval are in progress.	Done	**Sprint - 4**
		Criteria, Users should see visual cues indicating that the application is processing their request. Loading indicators should be displayed prominently and disappear once processing is complete.		
MS_24	3	As a Song Explorer, I want the application to store uploaded pictures for future reference.	Done	**Sprint - 4**
		Criteria, Uploaded pictures should be saved securely in the user's account. Users should be able to access their stored pictures from their profile or settings.		

INCOMPLETED STORIES IN SPRINT-4

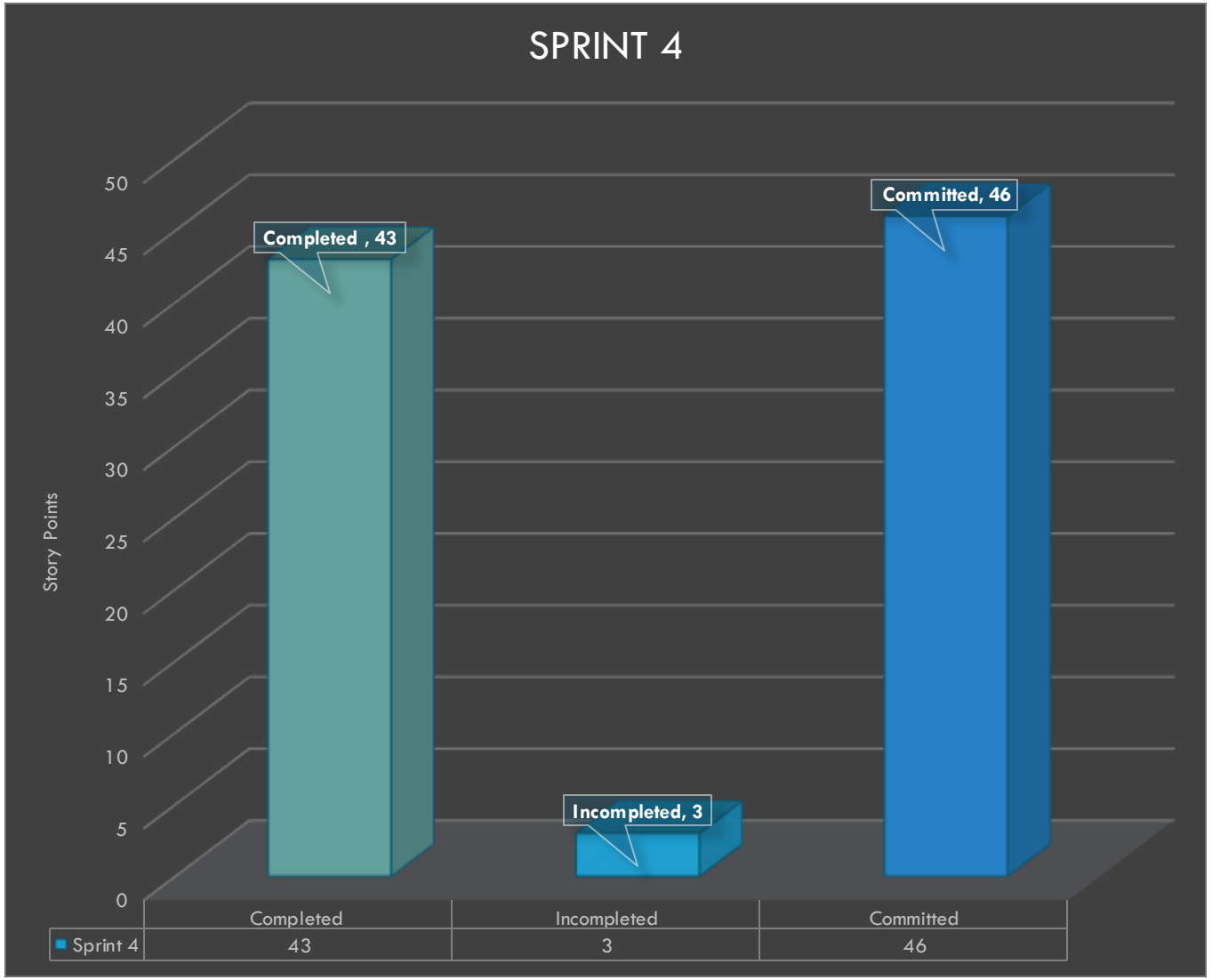
SPRINT 4 INCOMPLETED STORIES

Story ID	Story Points	User Story	Status	Sprint
MS_25	3	As a Song Explorer, I want the ability to delete uploaded pictures if needed.	In Progress	**Sprint - 4**
		Criteria, Users can delete uploaded pictures from their account. Deletion should be confirmed with a confirmation prompt to prevent accidental deletion.		

METRICS

SPRINT -4 TEAM VELOCITY

TEAM VELOCITY FOR SPRINT- 4 IS (NUMBER OF COMPLETED STORY POINTS) = 43

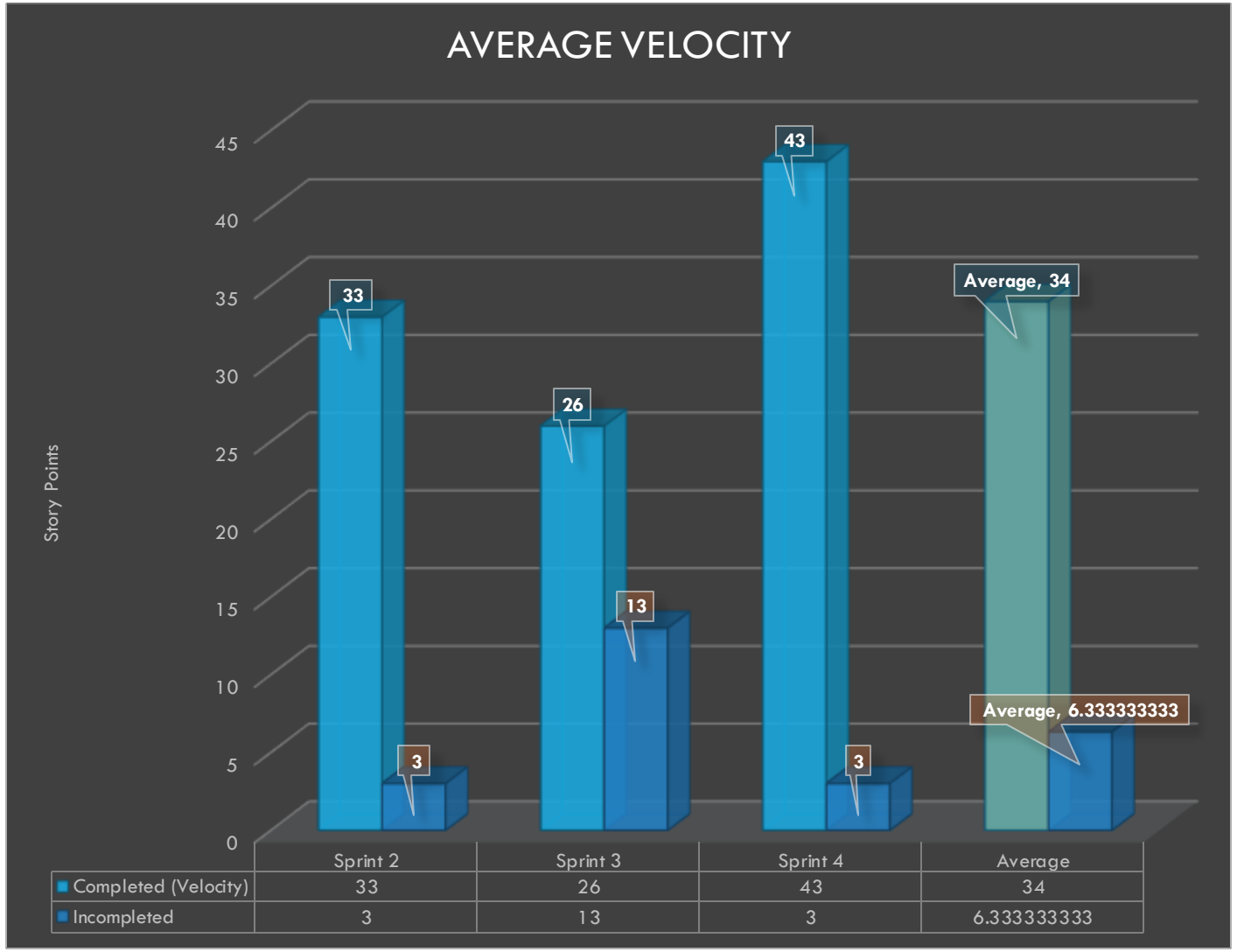


TEAM'S HISTORICAL(AVG) VELOCITY

AVERAGE VELOCITY =

$$\frac{\text{SUM OF VELOCITIES FOR COMPLETED SPRINTS}}{\text{NUMBER OF COMPLETED SPRINTS}}$$

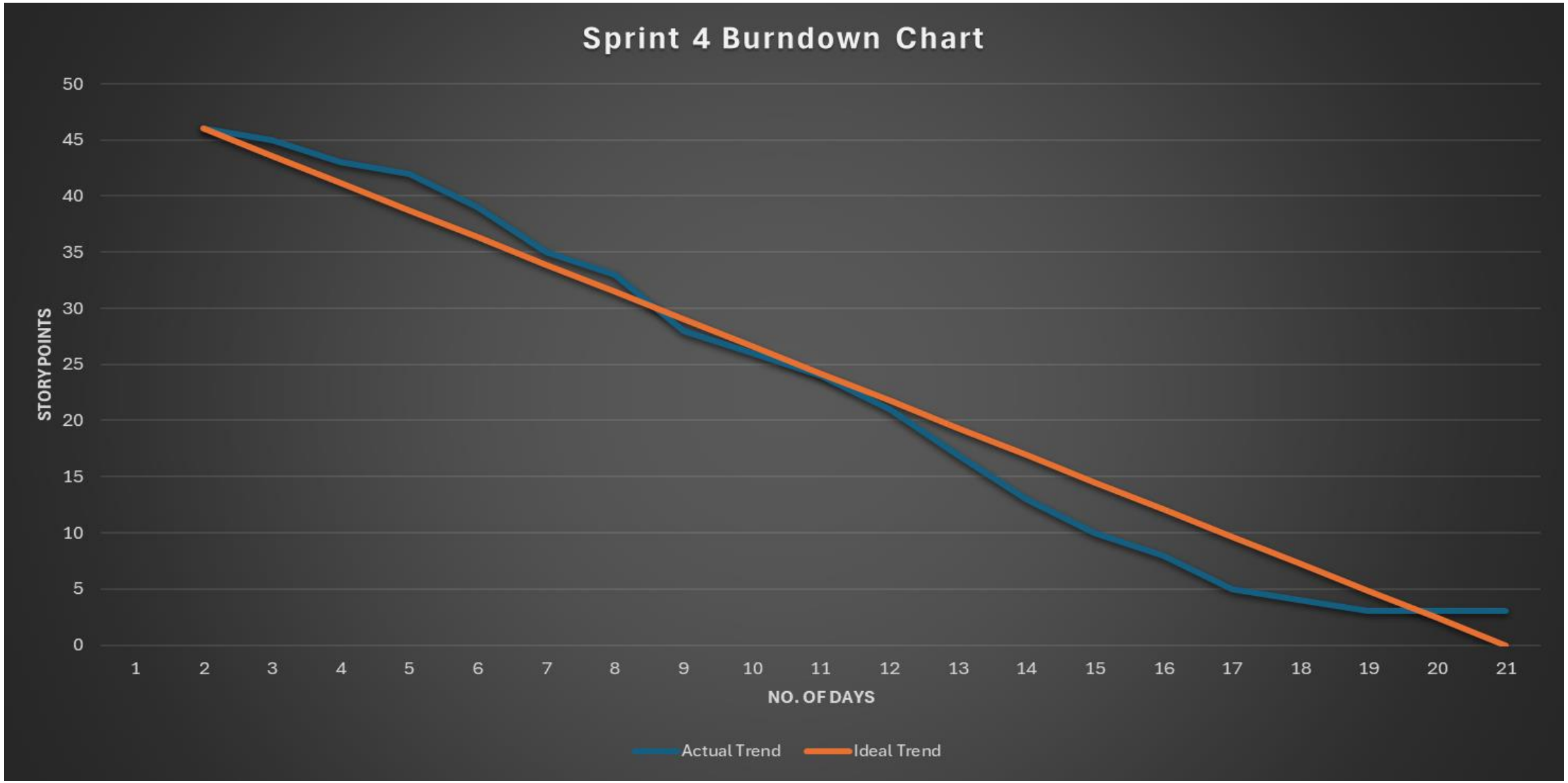
HISTORICAL VELOCITY OF MOODSPHERE IS 34



BURNDOWN CHARTS

SPRINT-4 BURNDOWN CHART

Sprint 4 Burndown Chart

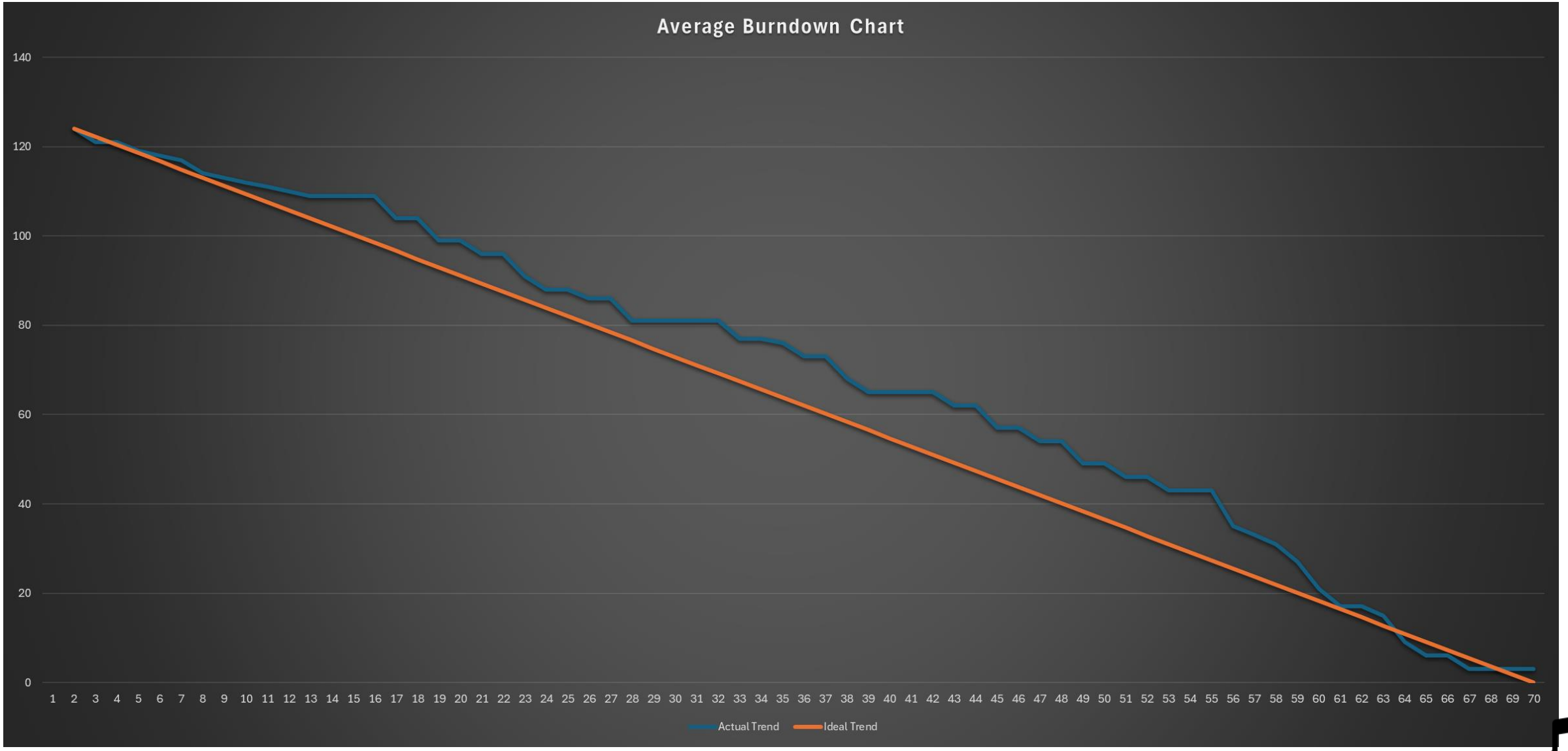


AVERAGE BURNDOWN CHART

Average Burndown Rate = Total Variance/ Total Number of Intervals
Where:

- Total Variance = Σ (Ideal Burndown - Actual Burndown) for all intervals
- Total Number of Intervals = Number of intervals (e.g., days, weeks) during the project duration. In our Case it is 3days.

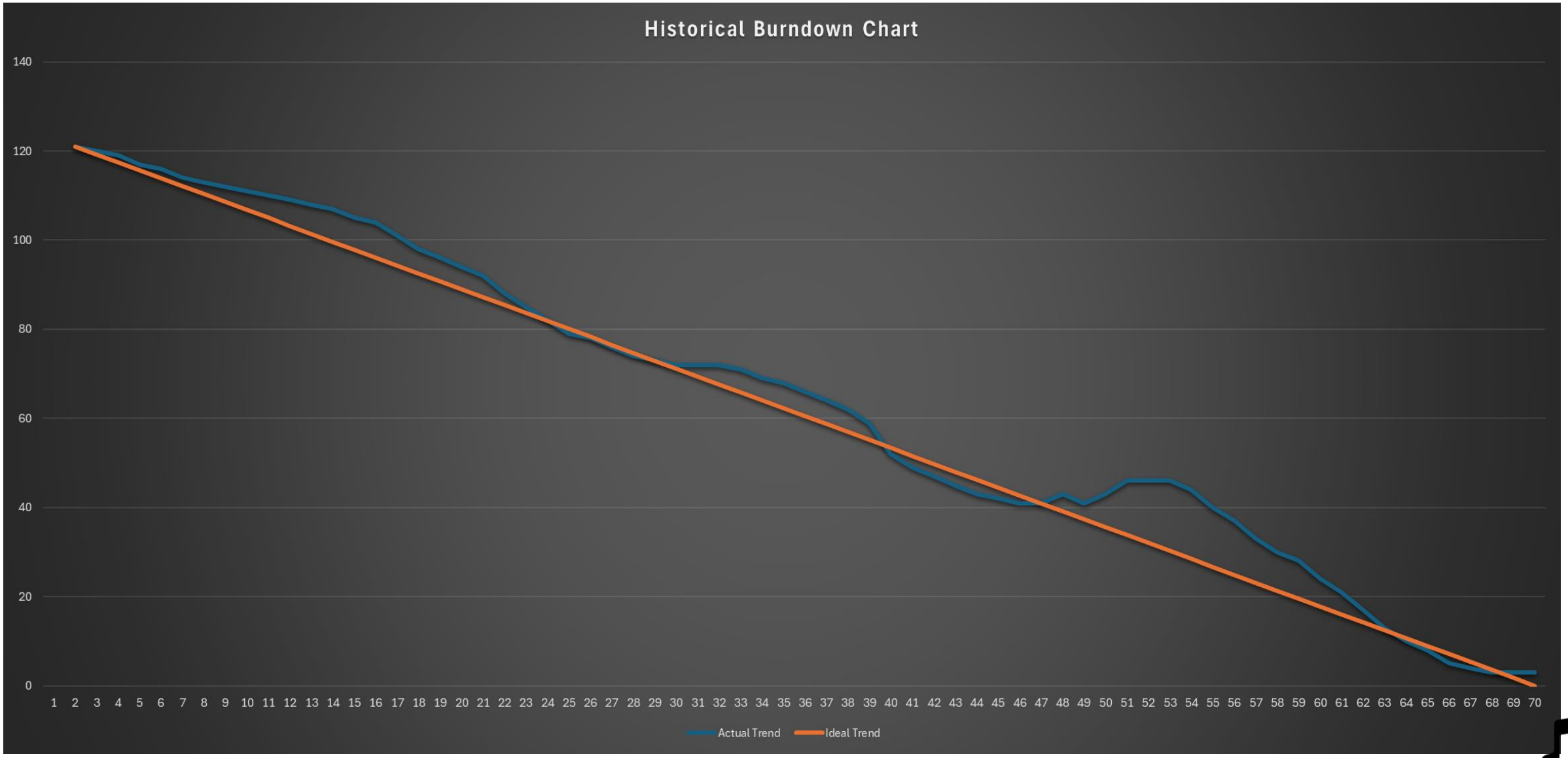
Average Burndown Chart



HISTORICAL BURNDOWN CHART

Historical burndown chart = Sum(Sprint 1 + Sprint 2 + Sprint 3 + Sprint 4)

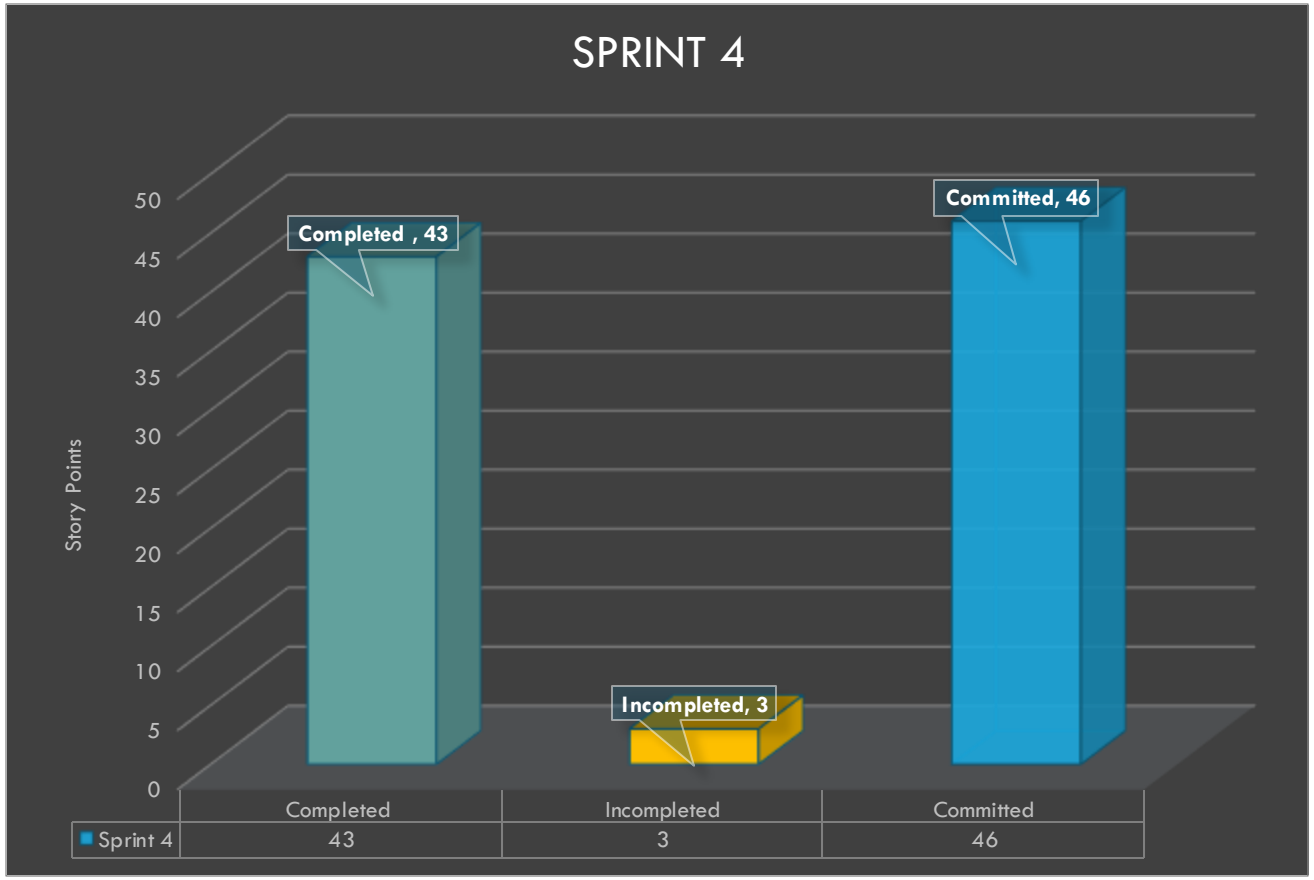
Historical Burndown Chart



SPRINT- 4 COMPLETED/COMMITTED RATIO

COMPLETED/COMMITTED RATIO FOR SPRINT – 4 IS

43/46 = 93.47%

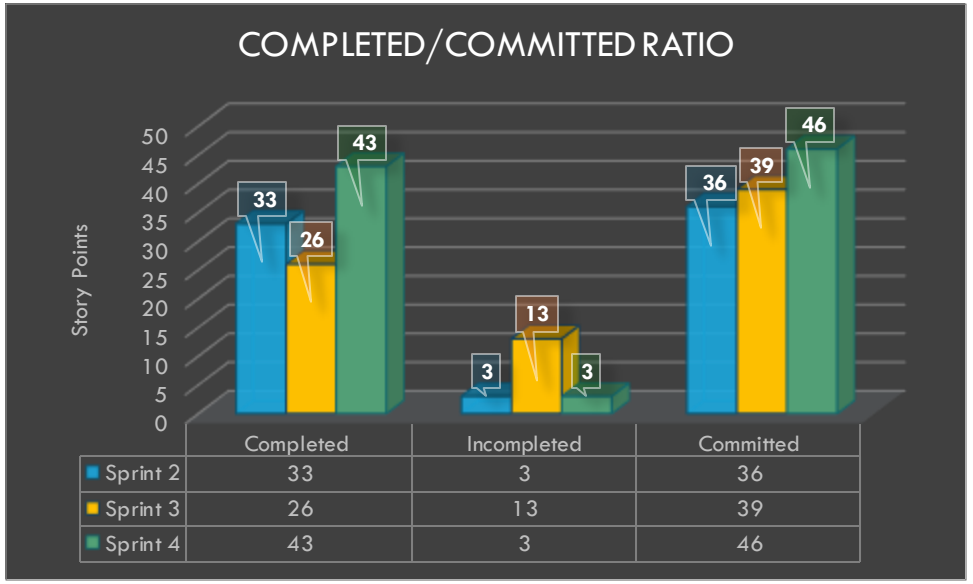


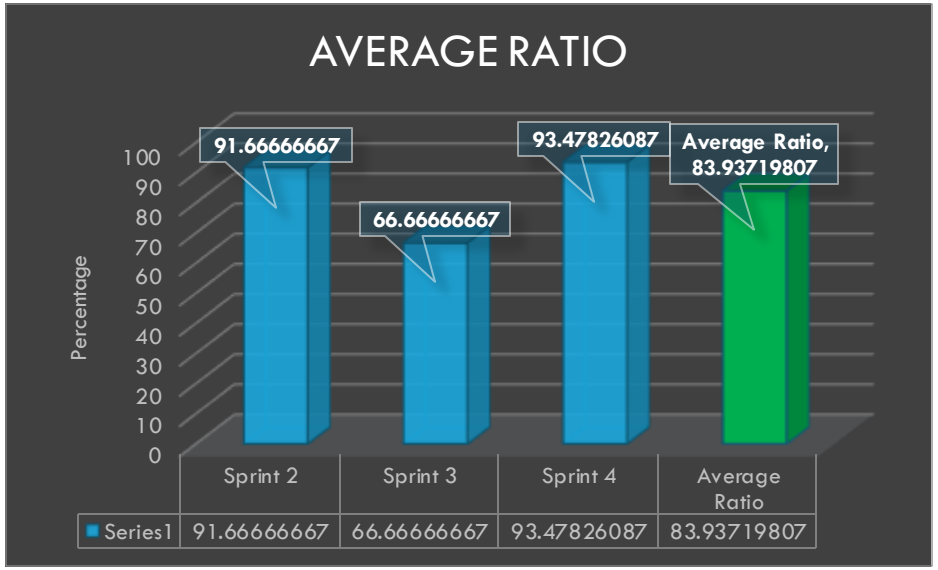
AVERAGE COMPLETED/COMMITTED RATIO

AVERAGE RATIO =

AVG(SPRINT-2 +SPRINT-3 + SPRINT 4 RATIO)

AVERAGE RATIO IS - 83.93%



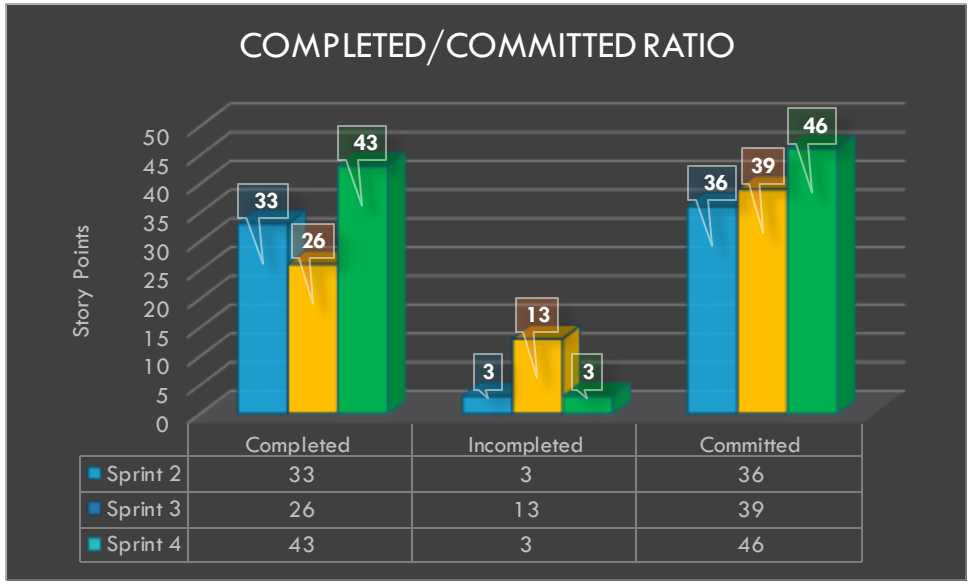


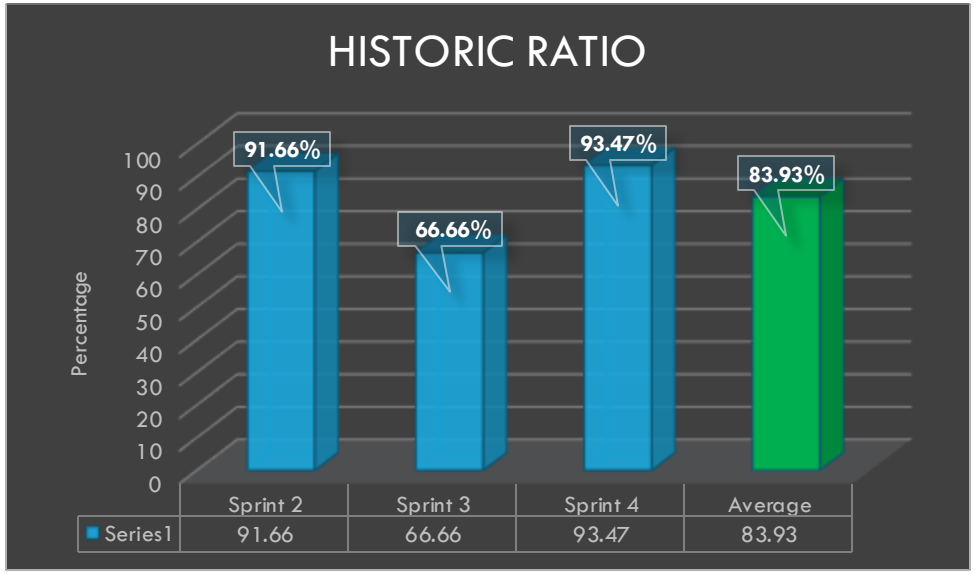
HISTORIC COMPLETED/COMMITTED RATIO

HISTORIC RATIO =

$$\frac{\text{TOTAL COMPLETED WORK}}{\text{TOTAL COMMITTED WORK}}$$

HISTORIC RATIO IS – 83.93%





RETROSPECTIVE

Retrospective Sprint 4

What went well

collaborated well on last minute errors

+ 10

Individuals Took initiative in certain tasks, knowing they have sound knowledge of particular Tech stack.

+ 6

Everyone collaborated on Github!

Learned complex things about GIT and applied them

+ 6

Excellent communication between the teammates which lead to the timely completion of the tasks.

+ 5

Taking daily updates on progress and blockers in daily stand-ups.

+ 6

Everyone was Consistent with our progress and keeping up to it

Everybody helped each other in the development ideas

+ 5

Everyone took up their task and

Everybody took initiatives for documenting tasks

Everyone took up their task and worked fairly on it

+ 13

What can be improved

More Active Participation by each team member in team meetings

+ 6

Too many meetings

Team communication and individual Project understanding

+ 4

There were some problems faced with merge conflicts when pushing code on GitHub, fixing that took a day. That could've been avoided.

+ 5

Document your work regularly so that makes easy for somebody else to know how it is done even after project is done

+ 6

Taking up tasks even if you are not up to date with the tech stack or are not aware of the workflow

+ 6

Every one should be considered as part in the team as rather than letting individual figuring out the assign and distribution of work.

+ 5

Documenting every commit in github in describe session to ease the work of other teammates , who dont have to go through whole code

+ 8

Action Items

Reduced Meetings but regularly updating in communication channels and daily standups

+ 13

Before pushing your work on GitHub, let everyone on the group know. Keep everyone in loop.

+ 15

Adding descriptions in the branch pushes will help the team to understand about changes

+ 6

Retrospective Sprint-4

What went Well?

- The team collaborated effectively, even when addressing last-minute errors. Certain individuals demonstrated initiative by taking charge of tasks they were knowledgeable about in specific tech stacks
- Collaboration on GitHub was consistent across the board. Complex aspects of Git were learned and successfully applied.
- Daily stand-ups were utilized to provide updates on progress and address any blockers. Consistency in progress was maintained by all team members.
- Development ideas were collectively brainstormed, with everyone pitching in to help. Each team member took ownership of their tasks and worked diligently on them.

What can we do better

- Active participation by each team member in team meetings is essential for effective collaboration. Improving team communication and ensuring everyone understands individual project requirements is crucial.
- Taking on tasks, even if one isn't fully familiar with the tech stack or workflow, can be beneficial for learning and team flexibility.
- Assigning and distributing work should involve everyone in the team, rather than leaving it to individuals.

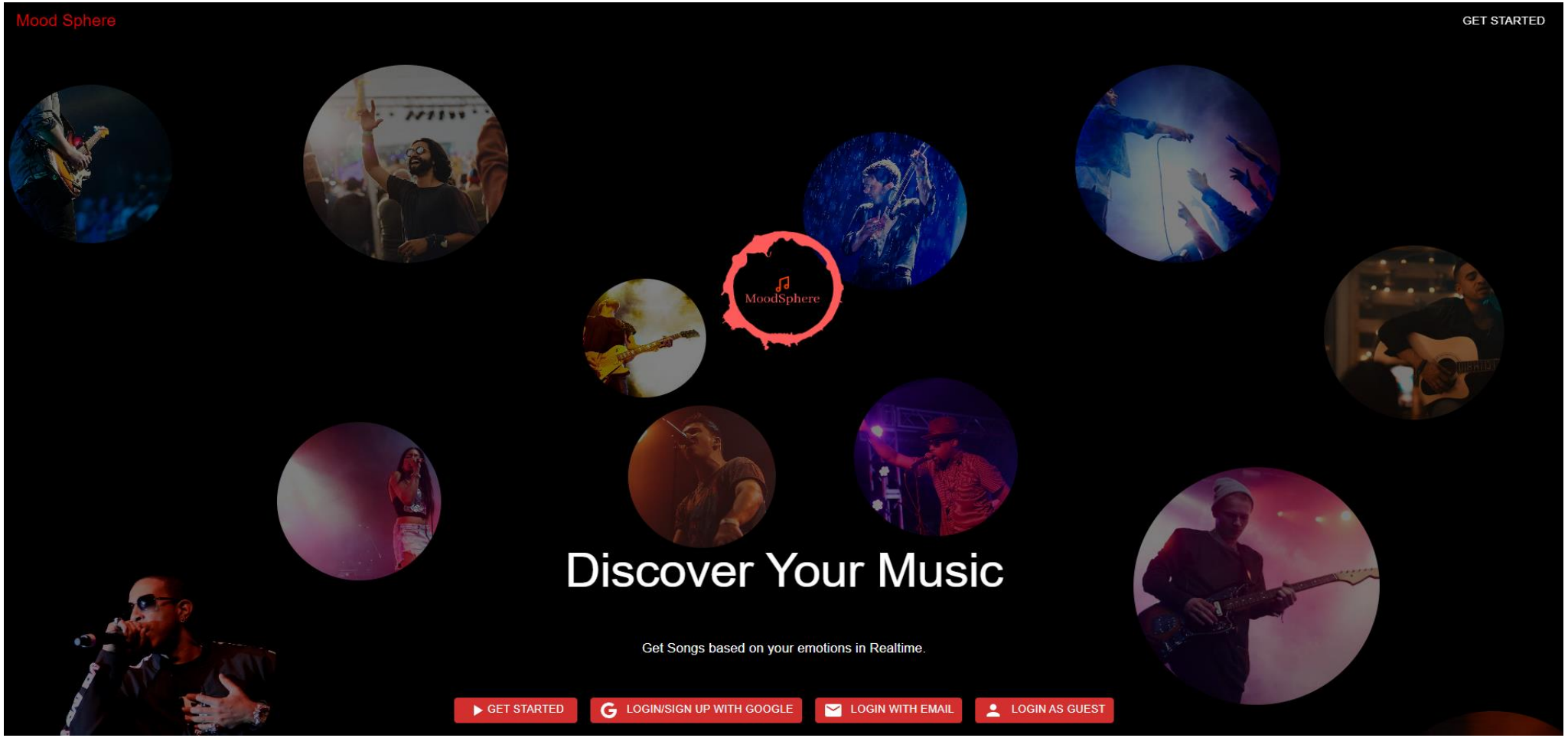
Actions items

- We will be regularly updating through communication channels and daily standups.
- Before pushing your work on GitHub, make sure to let everyone in the group know. Keeping everyone in the loop is crucial. Additionally, adding descriptions in the branch pushes will help the team understand the changes being made.

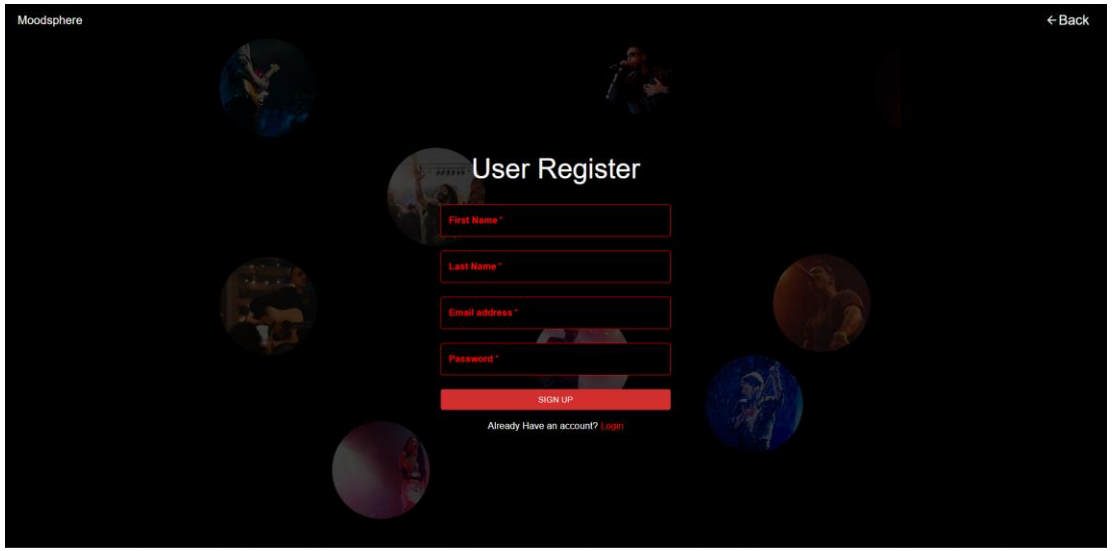
PROJECT DEMO

APP SCREENSHOTS

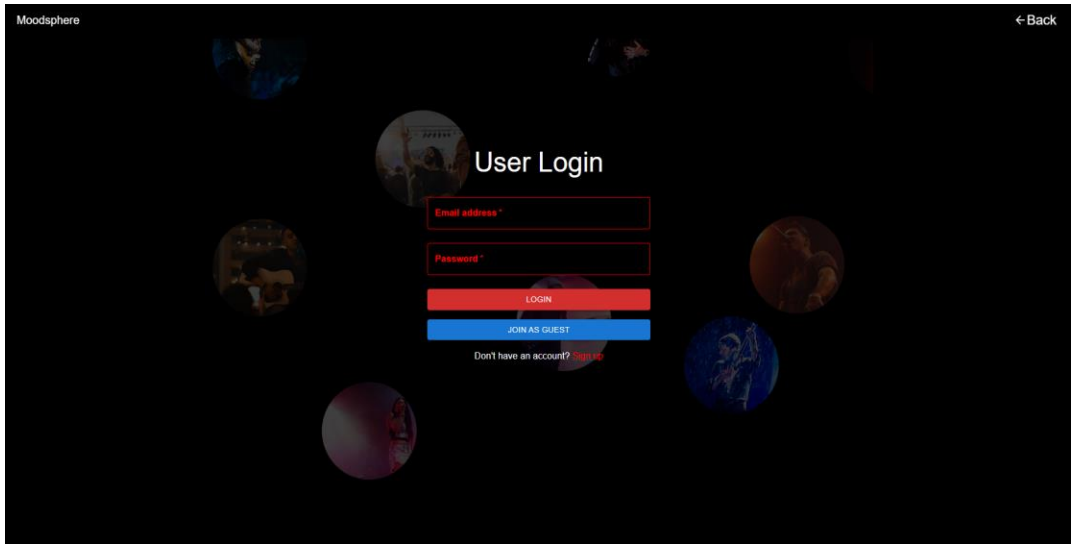
Landing Page



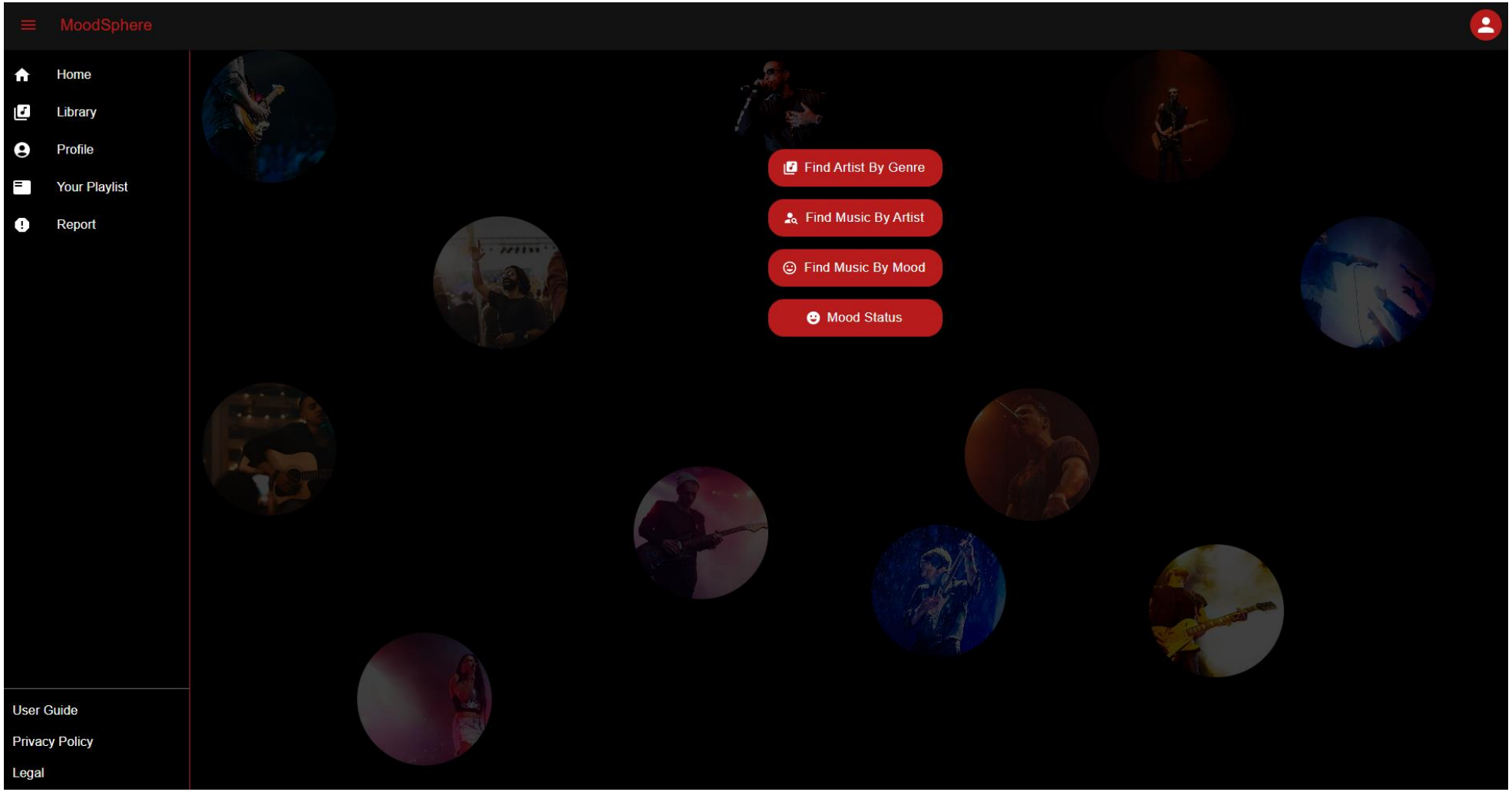
Signup



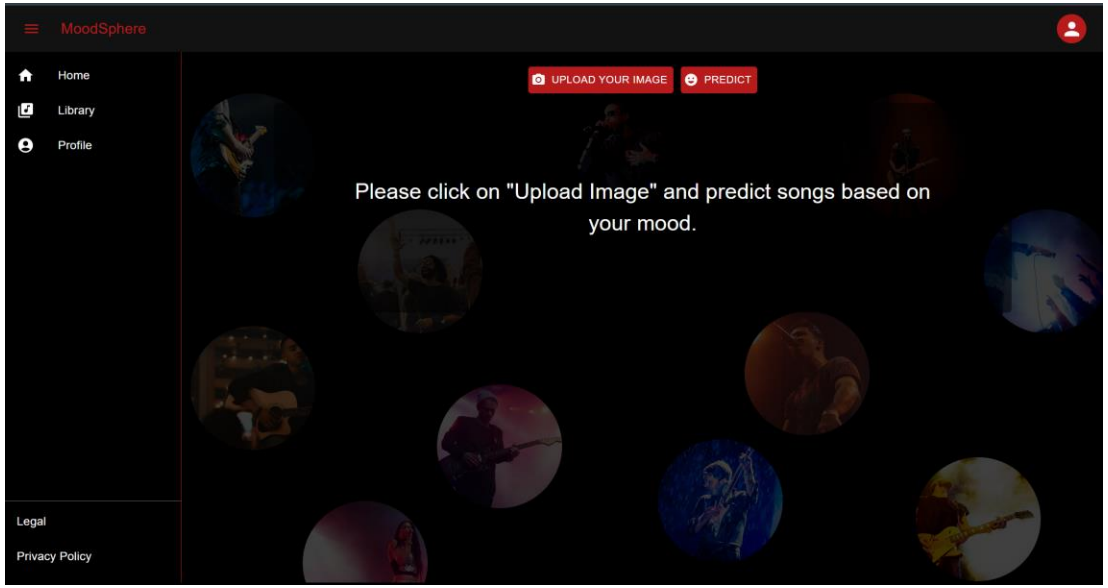
Login

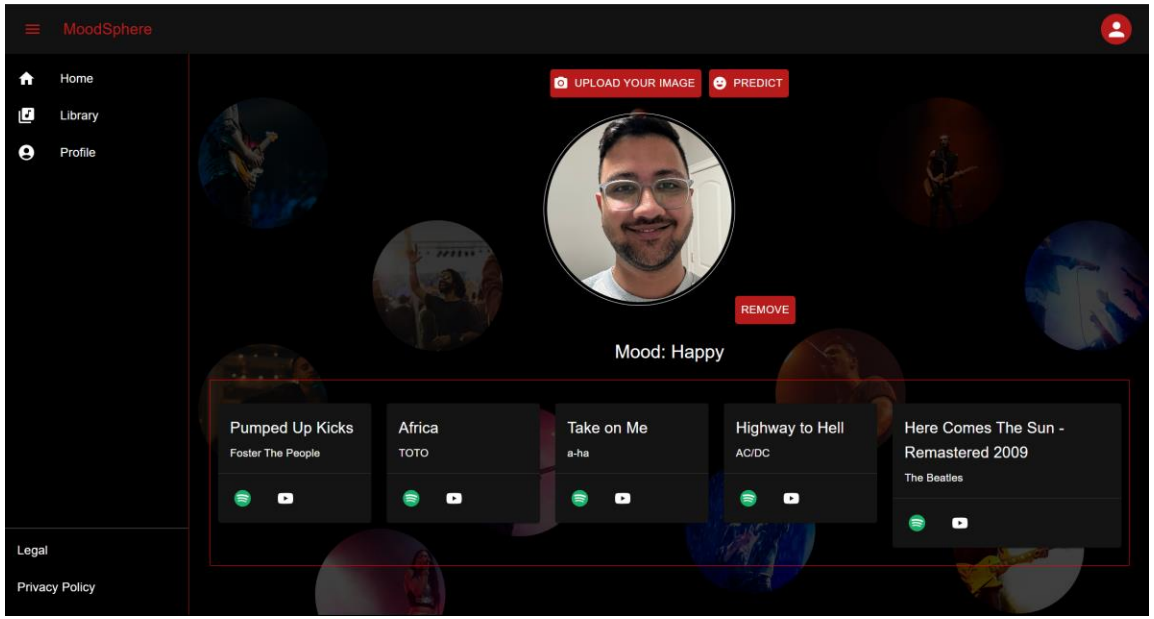


Home Page

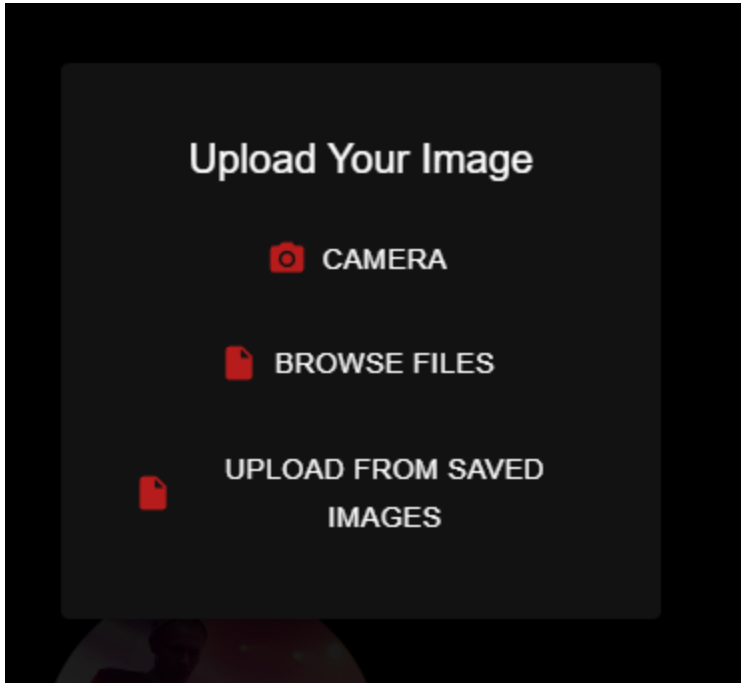

MoodSphere
Home
Library
Profile
Your Playlist
Report
User Guide
Privacy Policy
Legal
Find Artist By Genre
Find Music By Artist
Find Music By Mood
Mood Status

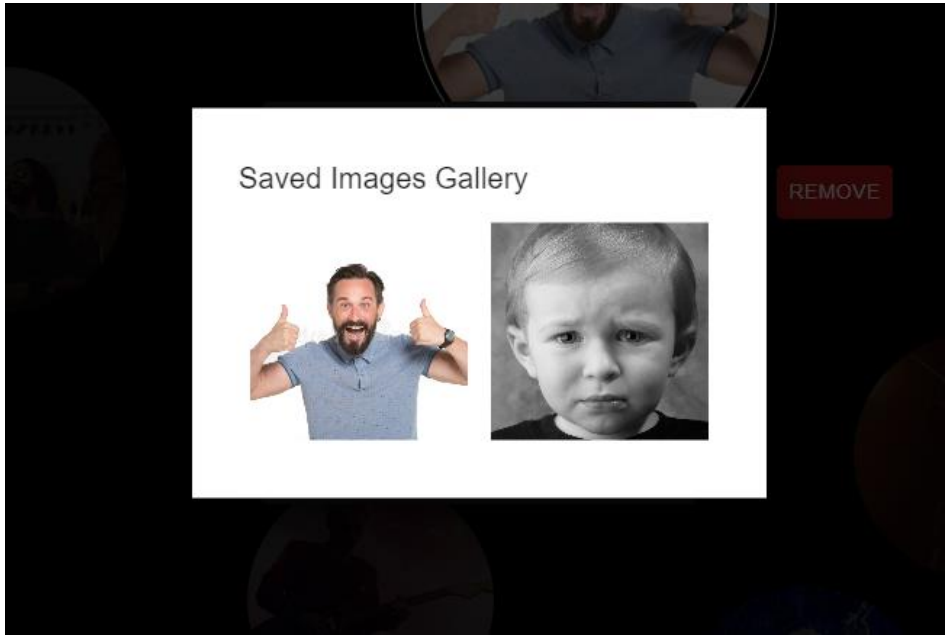
Mood Prediction and Song Recommendation



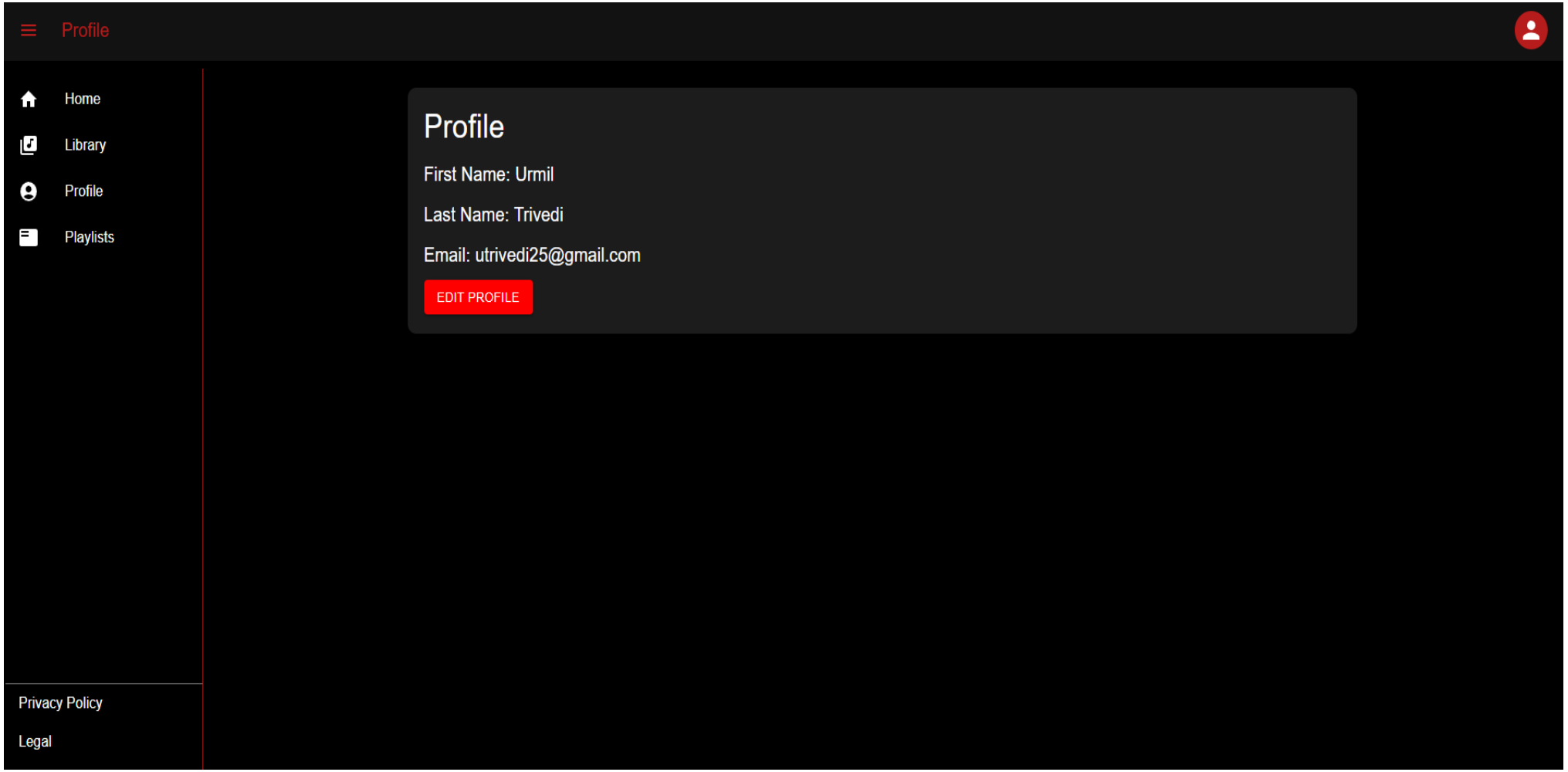


Saved Images Gallery

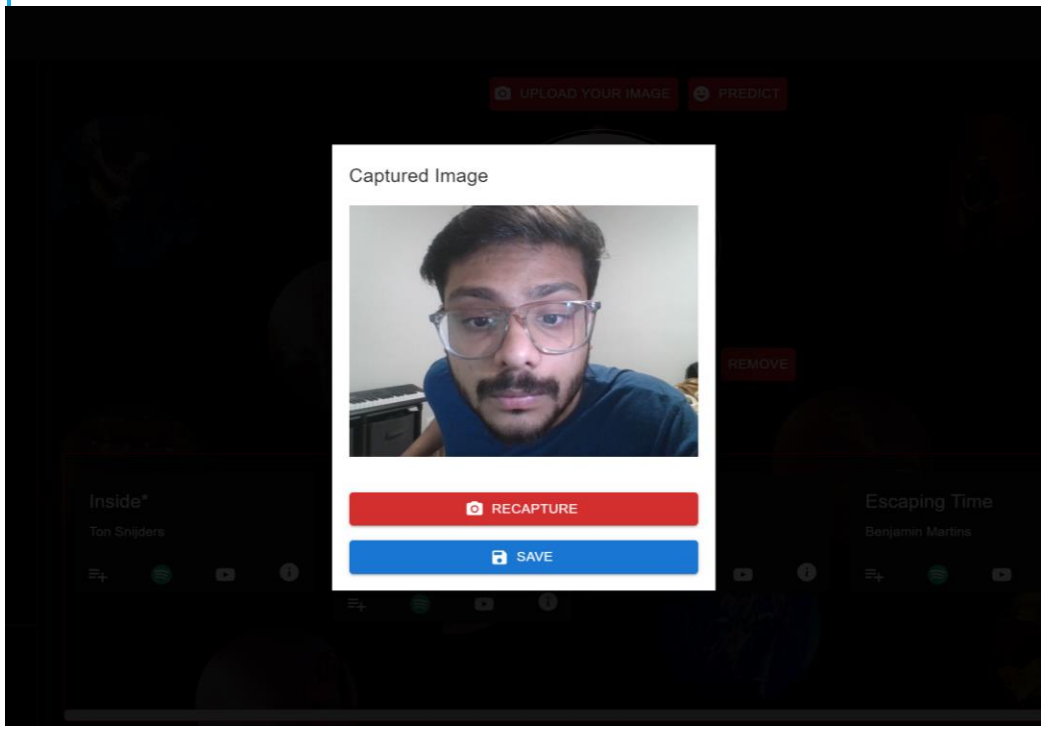


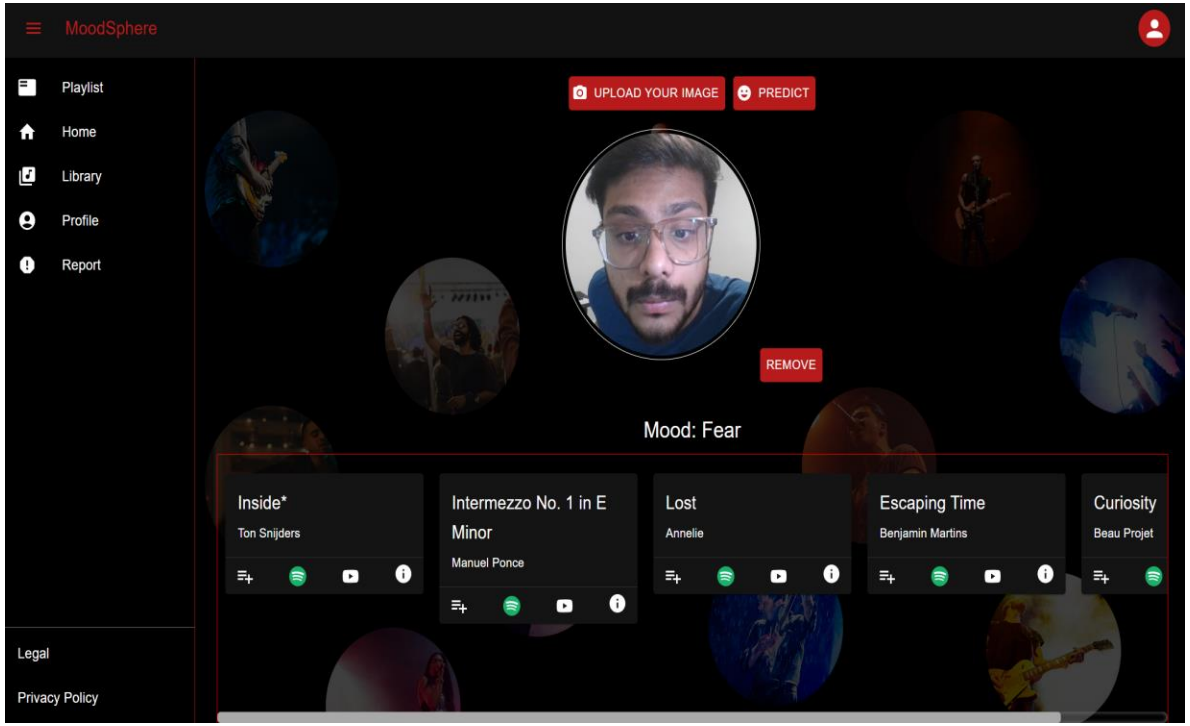


Profile Page

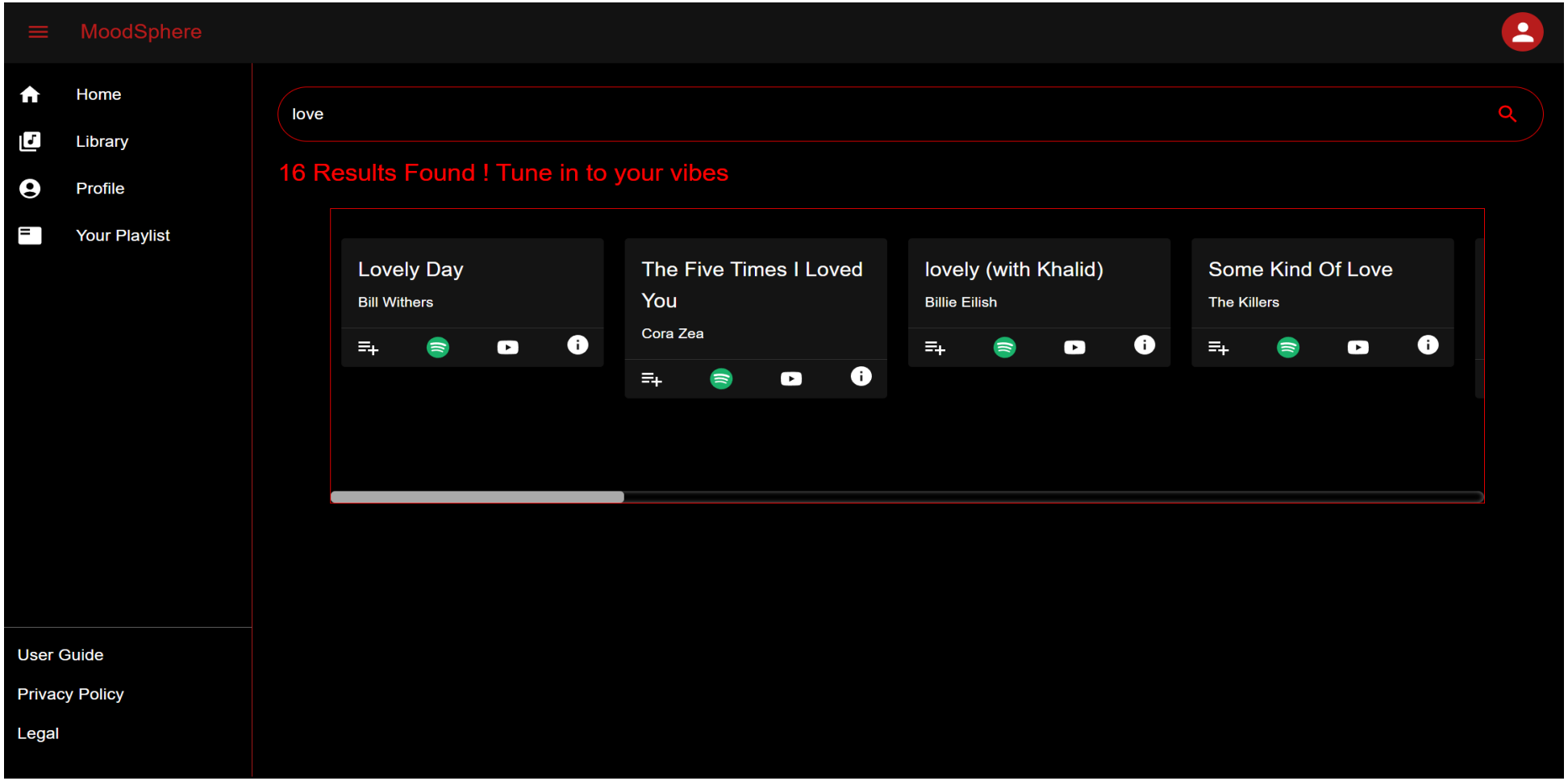


User Clicked picture and Recommendation

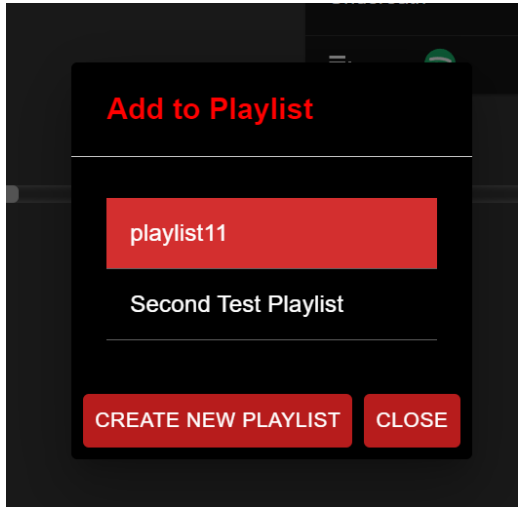




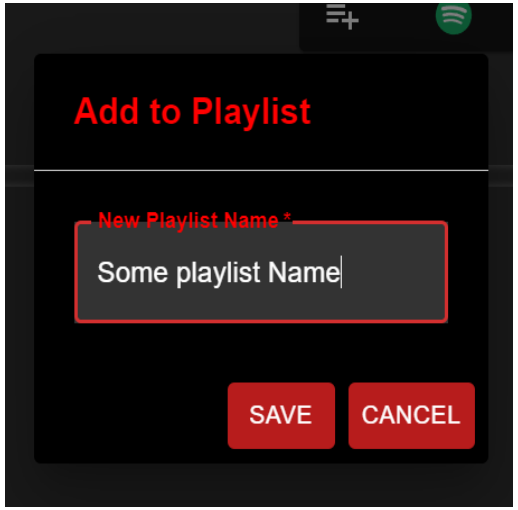
Song and Artist Search Library



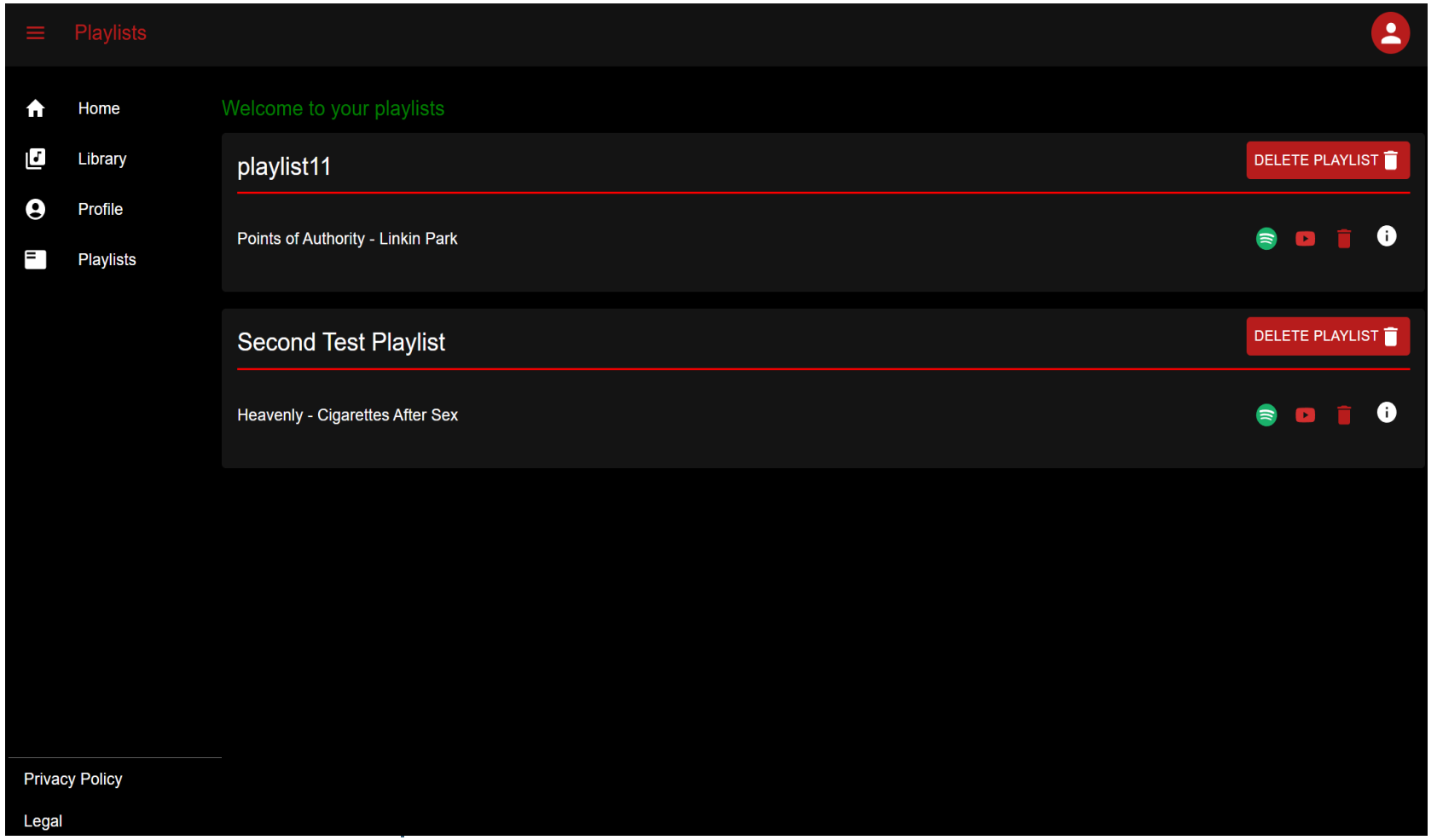
Playlist Feature



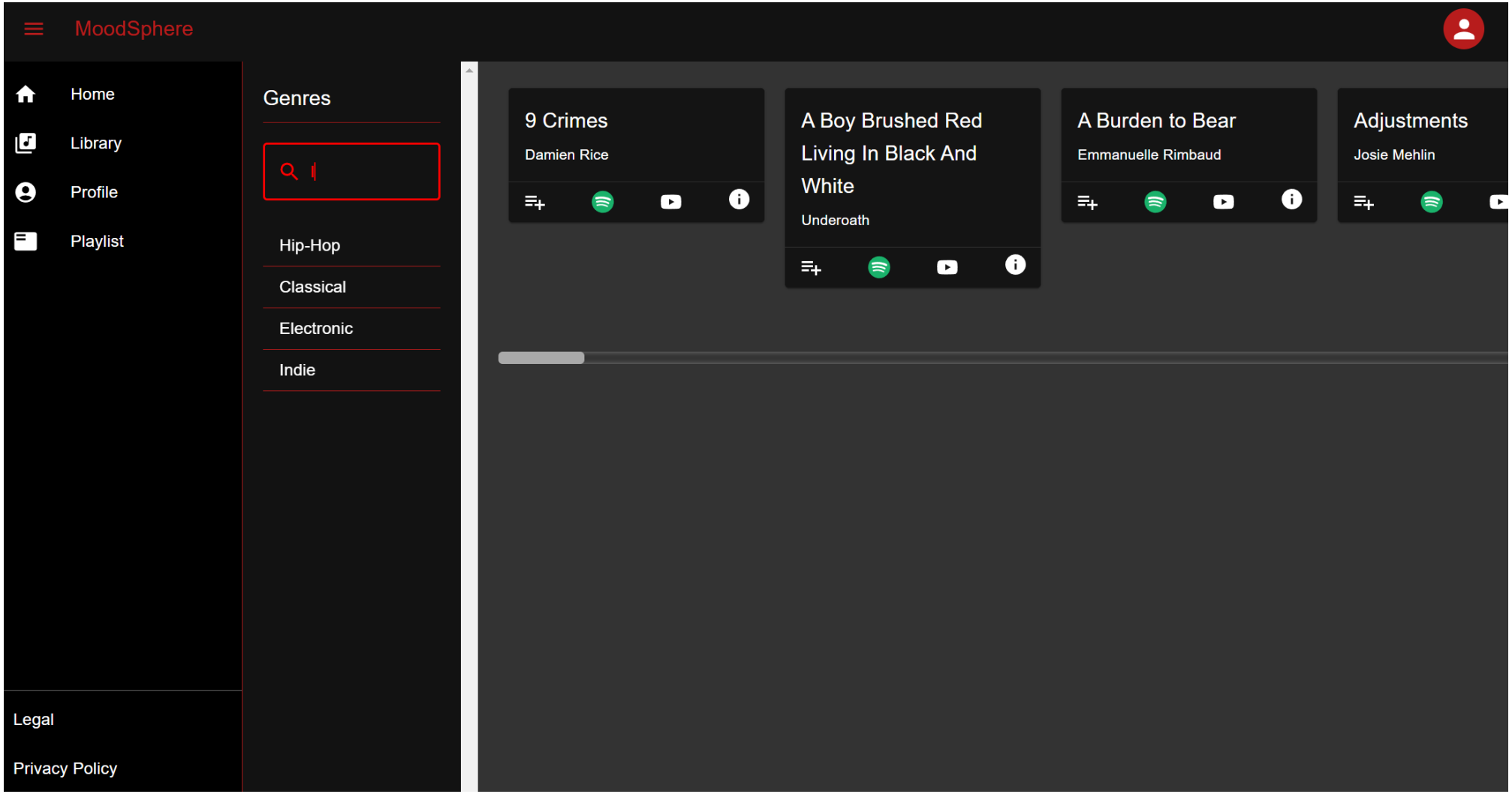
Add to Existing



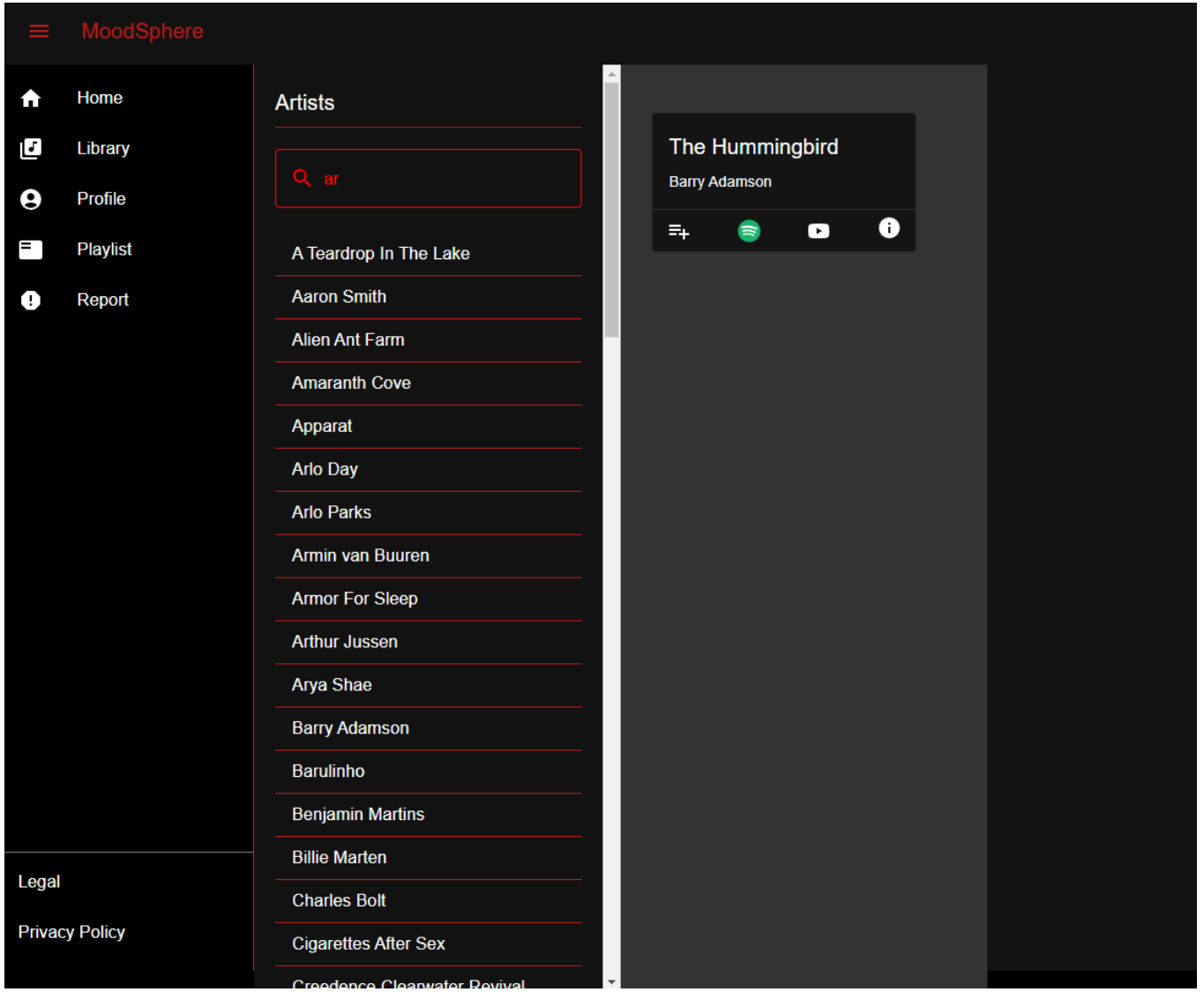
Create new



Search Songs by Genre



Find music by artist



User Guide, Privacy policy and Legal Modals

User Guide

Sign Up

Select "Sign up with Google" or "Sign up with Email" from the available sign-up options. Signup using Google Select "Sign up with Google" from the menu. You'll be asked to log into your Google account if you haven't already. Review the permissions requested by Mood Sphere. To allow access to your Google account details, click "Allow". You'll be taken to the Mood Sphere dashboard after your account has been created. Signup Using Email The "Sign up with Email" button should be clicked. Type your email address into the designated space. Make sure your account password is secure. For the registration process to be completed, click the "Sign Up" button. You will be taken to the Mood Sphere dashboard after registration.

Find Music by Genre

Choose this option to listen to music in the genres that you enjoy. Choose a Genre: Choose from a wide range of genres, including jazz, pop, rock, hip hop, and more. Find New Music: Look into the songs and musicians that fall within your favorite genres. Enjoy Personalized Recommendations: Get recommendations based on your preferred genres, which will assist you in finding new songs and musicians that share your interest in music.

Find Music by Artist

CLOSE

Privacy Policy

Moodsphere is committed to protecting your privacy and ensuring a secure experience for all our users. This Privacy Policy outlines our practices concerning the collection, use, and sharing of your personal information.

Information Collection

We collect information necessary for providing our services, including:

- Email addresses for account registration and communication purposes.
- Usage data to enhance our services and user experience.

Image Processing

Images uploaded for mood prediction are processed securely and deleted immediately after analysis.

Use of Information

Your data enables us to personalize the app experience and recommend music tailored to your preferences. It also helps us in improving our services.

Data Sharing

We do not share your personal information with third parties, except as

CLOSE

This legal notice contains the terms and conditions governing your use of the Moodsphere app and its services. By accessing or using Moodsphere, you agree to be bound by these terms.

Intellectual Property Rights

All content within Moodsphere, including texts, graphics, logos, icons, images, audio clips, digital downloads, and software, is the property of Moodsphere or its content suppliers and protected by international copyright and trademark laws.

User Conduct

Users are expected to use Moodsphere services responsibly and ethically. Any form of misuse, including the upload of offensive or copyrighted material, may result in termination of your access to the app.

Limitation of Liability

Moodsphere and its affiliates will not be liable for any damages arising from the use of this app beyond the scope permitted by applicable law.

Amendments

We reserve the right to amend these terms at any time. Your continued use of Moodsphere following any changes indicates your acceptance of the new terms.

CLOSE

API SLIDES

API Endpoint: /songs-by-artist

- This API endpoint "/songs-by-artist" is designed to recommend songs based on a provided artist name.
- The API endpoint accepts POST requests and expects to receive a JSON payload containing an artist name. It processes this input to find and return a list of songs associated with the specified artist.
- Upon receiving the request, it attempts to parse the JSON to extract the artist's name.
- If no artist name is provided, it responds with an error and a 400-status code. If an artist name is given, it then uses the get_songs_by_artist function to retrieve the corresponding songs

```python
def get_songs_by_artist(artist_name):
    songs_by_artist = Music_Player[Music_Player['artist'] == artist_name]
    if songs_by_artist.empty:
        return []
    return songs_by_artist[['name', 'artist']].values.tolist()

def get_songs_by_genre(artist_name):
    songs_by_artist = Music_Player[Music_Player['artist'] == artist_name]
    if songs_by_artist.empty:
        return []
    return songs_by_artist[['name', 'artist']].values.tolist()

#API for recommending songs based on artists
@app.route('/songs-by-artist', methods=['POST'])
def artist():
    try:
        data = request.get_json()
        artist_name = data.get('artist_name')

        if not artist_name:
            return jsonify({'error': 'Artist name not provided'}), 400

        songs = get_songs_by_artist(artist_name)
        if not songs:
            return jsonify({'message': 'No songs found for the provided artist'}),

        return jsonify({'songs': songs}), 200

    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

API Endpoint: /songs-by-genre

This endpoint is for handling POST requests to the URL path '/songs-by-genre'. It processes incoming JSON data to extract a 'genre_name' and uses it to retrieve songs matching that genre via the get_songs_by_genre function. The function ensures robust error handling by validating the presence of 'genre_name', checking the availability of songs, and catching exceptions. If 'genre_name' is not provided or no songs are found, it returns appropriate error messages and HTTP status codes (400 for bad input and 404 for no data found). In case of success, it returns a list of songs with a 200 status code. Any exceptions during the process result in a 500 status code, signaling an internal server error.

```python
#API for retrieving songs by genre ~Shane
@app.route('/songs-by-genre', methods=['POST'])
def genre():
    try:
        data = request.get_json()
        genre_name = data.get('genre_name')

        if not genre_name:
            return jsonify({'error': 'Genre name not provided'}), 400

        songs = get_songs_by_genre(genre_name)
        if not songs:
            return jsonify({'message': 'No songs found for the provided genre'}), 404

        return jsonify({'songs': songs}), 200

    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

API Endpoint: /predict

- The function recommend_songs takes a predicted class of mood from an image and maps it to a corresponding mood category. For example, 'Disgust' is mapped to 'Sad', and combinations like ['Happy', 'Sad'] or ['Fear', 'Angry'] are mapped to 'Happy' or 'Calm' respectively. Any other mood prediction results in the 'Energetic' category.
- Once the mood is determined, the system filters the songs from the Music_Player dataset that match the mood category. Then it sorts these songs by their popularity in descending order and returns the top 5 as a dictionary in the 'records' format.

```python
# Function to recommend songs based on predicted class
def recommend_songs(pred_class):
    if pred_class == 'Disgust':
        mood = 'Sad'
    elif pred_class in ['Happy', 'Sad']:
        mood = 'Happy'
    elif pred_class in ['Fear', 'Angry']:
        mood = 'Calm'
    else:
        mood = 'Energetic'

    # Filter music based on predicted mood and sort by popularity
    recommended_songs = Music_Player[Music_Player['mood'] == mood]
    recommended_songs = recommended_songs.sort_values(by="popularity", ascending=False).head(5).to_dict('records')

    return recommended_songs


# Define route for prediction
@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Get the uploaded image file from the request
        file = request.files.get('image')
        if not file:
            return jsonify({'error': 'No file uploaded'}), 400
        print(request.headers)
        # Read the image file as an array
        img = Image.open(io.BytesIO(file.read()))
        img = img.resize((150, 150))  # Resize image to match model's expected sizing
        img_array = np.asarray(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array = img_array / 255.0  # Normalize pixel values
```

API Endpoint: /predict

- Before making a prediction, the code asserts that the shape of the pre-processed image array matches the expected input shape for the model, which is (1, 150, 150, 3). If the shape does not match, an assertion error with the message "Unexpected image shape" will be raised.
- The response is constructed as a JSON object with two fields: 'prediction', containing the name of the predicted mood class, and 'recommended_songs', containing the list of songs recommended by the system. This response is then returned to the client. If an exception occurs during the process, an error response with a 400-status code and the exception message is returned.

```
    # Make sure image shape matches the expected input shape
    assert img_array.shape == (1, 150, 150, 3), "Unexpected image shape"

    # Perform prediction using your model
    prediction = model.predict(img_array)
    predicted_class_index = np.argmax(prediction)
    predicted_class = class_names[predicted_class_index]

    prediction = model.predict(img_array)
    predicted_class_index = np.argmax(prediction)
    predicted_class = class_names[predicted_class_index]

    recommended_songs = recommend_songs(predicted_class)

    response = {
        'prediction': predicted_class,
        'recommended_songs': recommended_songs
    }

    return jsonify(response)

except Exception as e:
    return jsonify({'error': str(e)}), 400
```

API Endpoint: /search

The /search API endpoint processes search queries for songs or artists, ensuring the 'query' string is neither empty nor just spaces. Valid queries trigger a JSON response with a 'matches' array containing relevant search results.

Upon receiving a valid request, the server responds with a JSON object containing a field named 'matches', which includes an array of search results pertinent to the user's input. In cases where the input is invalid, the server encounters a network issue, or the response structure is not as expected, the endpoint logs an error and resets the search results to an empty array.

Client-side error or status messages reflect the success or failure of the search request. The endpoint can return various status codes: 200 OK for successful retrieval of search results, 400 Bad Request for improper request formatting or missing information, and 500 Internal Server Error when server issues prevent request fulfillment. This setup ensures that the endpoint effectively enhances user interaction by promptly providing relevant music and artist search results.

```python
@app.route('/search', methods=['POST'])
def search():
    query = request.json.get('query', '').lower().strip()  # Normalize query to lowercase
    if not query:
        return jsonify({'error': 'No search query provided'}), 400

    # Search for matching songs or artists
    results = search_songs_and_artists(query)
    return jsonify({'matches': results})

def search_songs_and_artists(query):
    # Perform a single scan to fetch all entries and filter manually for demo purposes
    # This is not efficient for large datasets
    songs_ref = db.collection('tracks')
    try:
        docs = songs_ref.stream()
        results = []
        for doc in docs:
            if len(results) >= 30:
                break  # Stop processing once we have 30 items
            song_data = doc.to_dict()
            # Check both the artist and name fields for the query
            if query in song_data.get('name', '').lower() or query in song_data.get('artist', '').lower():
                results.append(song_data)
        return results
    except Exception as e:
        print(f"Failed to fetch songs: {str(e)}")
        return []
```

API Endpoint: /get-playlist

Method: GET
Description: Fetches all playlists associated with the logged-in user.
Parameters:
user_id: The unique identifier of the user.
Response: Returns a JSON object containing a success status and an array of playlist data if successful. On failure, it provides an error message.
Error Handling: The component logs an error and sets the playlist array to empty if the fetch fails.

```python
@app.route('/get-playlist', methods=['GET'])
def get_playlist():
    user_id = request.args.get('user_id')

    if not user_id:
        return jsonify({'error': 'User ID is required'}), 400

    try:
        # Fetch all playlists for the given user ID
        playlists_col = db.collection('playlists').document(user_id).collection('user_playlists')
        playlists = playlists_col.stream()

        playlists_data = [doc.to_dict() for doc in playlists]
        if playlists_data:
            return jsonify({'success': True, 'data': playlists_data}), 200
        else:
            return jsonify({'success': False, 'message': 'No playlists found for the user'}), 404

    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

API Endpoint: /delete-playlist

Method: DELETE
Description: Deletes a specific playlist.
Parameters:
user_id: The user's unique identifier.
playlist_id: The unique identifier of the playlist to be deleted.
Response: Returns a success message upon successful deletion.
Error Handling: Logs an error message if the deletion fails and refreshes the playlist data to maintain consistency.

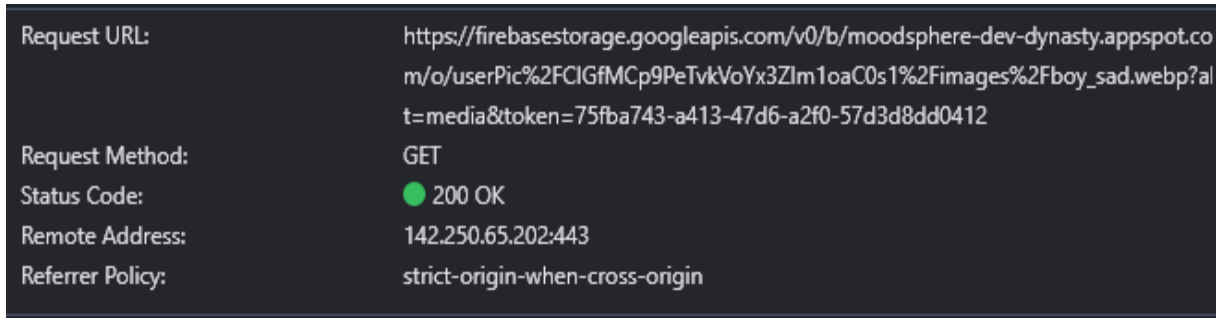
```python
@app.route('/delete-playlist', methods=['DELETE'])
def delete_playlist():
    try:
        user_id = request.args.get('user_id')
        playlist_id = request.args.get('playlist_id')

        playlist_ref = db.collection('playlists').document(user_id).collection('user_playlists').document(playlist_id)
        playlist_ref.delete()
        return jsonify({'success': True, 'message': 'Playlist deleted successfully'}), 200

    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

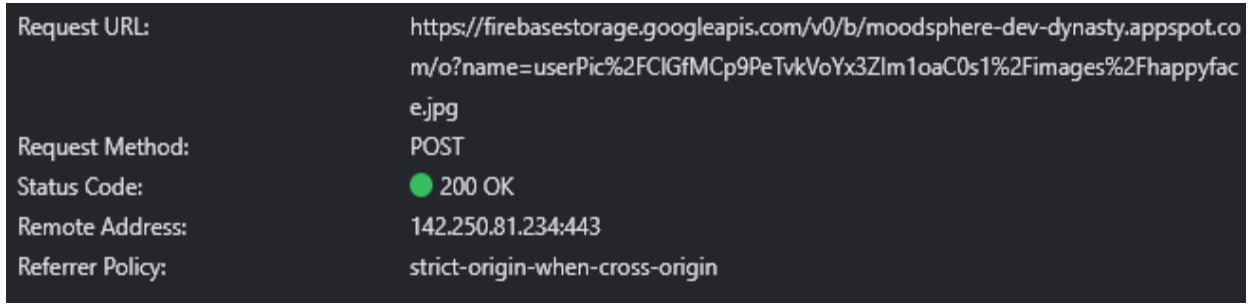
API Endpoint: userPic/${user.uid}/images

Method: GET
Description: Fetches all the user's saved images from Firestore, retrieves their download URLs from Firebase Storage, and updates the gallery UI
Parameters:
user_id: The unique identifier of the user, used to access the specific Firestore collection of the user's images
Response: Returns a JSON object containing a success status and an array of image data (including file names and download URLs) if successful. On failure, it provides an error message.
Error Handling: The function logs an error and updates the gallery UI with an empty array if the fetch fails.



API Endpoint: userPic/${user.uid}/images/${file.name}

Method: POST
Description: Uploads an image file to Firebase Storage, stores its metadata in Firestore, and updates the UI with the image preview and relevant notifications.
Parameters:
event: Contains the file selected by the user through a file input form.
Response: Returns a JSON object containing a success status and updates related UI components if successful. On failure, it provides an error message.
Error handling: Logs an error to the console and displays an error notification via toast if the upload process encounters any issues.



GITHUB LINK

[HTTPS://GITHUB.COM/HTMW/2024S-DEV-DYNASTY/WIKI](https://github.com/htmw/2024s-dev-dynasty/wiki)

LIVE APPLICATION DEMO LINK

HTTPS://WWW.YOUTUBE.COM/WATCH?V=0OUBHMGBNQG&FEATURE=YOUTU.BE

THANK YOU

DevDynasty