CDOE GLA University

GUIDELINES FOR MCA MINI PROJECT REPORT

SESSION-2025 (3rd Semester)

PROJECT REPORT

Submitted to

Prof. Sowmyashree

Submitted by

Shashank Kumar Trivedi

MCA Final Year Student

CDOE, GLA University

Moodify

A Face Recognition based Music Player

https://github.com/strivedi4u/moodify

https://moodify4u.azurewebsites.net/

https://hub.docker.com/repository/docker/strivedi4u/moodify/

CERTIFICATE

The report of the project titled "**Moodify :- A Face Recognition Based Music Player**," submitted by Shashank Kumar Trivedi MCA of 2025, has been prepared under my supervision. This report fulfills the partial requirements for the MCA program at GLA University.

I also certify that this Project Report is entirely made by his genius efforts and reference from acknowledge of textual matter.

He prossesses a good moral character and I wish him for a bright future and success in life.

Prof. Sowmyashree

Head of Department (H.O.D.) (MCA)

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Prof. Sowmyashree, Department of MCA in GLA University, whose role as project guide was invaluable for the project. We are extremely thankful for the keen interest he took in advising us, for the books and reference materials provided for the moral support extended to us.

Date:- ………………..

Students Name

Shashank Kumar Trivedi

MCA Final Year

CERTIFICATE OF ACCEPTANCE

The report of the Project titled **Moodify :- A Face Recognition Based Music Player** submitted by Shashank Kumar Trivedi & Shraddha Jain of MCA Final Year of 2025 is hereby recommended to be accepted for the partial fulfilment of the requirements for MCA in GLA University.

Examiner Name	**Signature**
1.)……………………………	……………………………

ANTI PLAGIARISM DECLARATION

1. I know that plagiarism means taking and using the ideas, writings, works or inventions of another as if they were one's own. I know that plagiarism not only includes verbatim copying, but also the extensive use of another person's ideas without proper acknowledgement (which includes the proper use of quotation marks). I know that plagiarism covers this sort of use material found in textual sources and from the Internet.

2. I acknowledge and understand that plagiarism is wrong.

3. I understand that my project work must be accurately referenced. I have followed the rules and conventions concerning referencing, citation and the use of quotations as set out in the Department Guide.

4. This assignment is my own work, or my group's own unique group assignment. I acknowledge that copying someone else's assignment, or part of it, is wrong, and that submitting identical work to others constitutes a form of plagiarism.

5. I have not allowed, nor will I in the future allow, anyone to copy my work with the intention of passing it off as their own work.

Student Name	**Signature**
Shashank Kumar Trivedi	

TABLE OF CONTENTS

1.) AIM & OBJECTIVES

1. Emotion-Driven Music Playback: Moodify uses advanced facial recognition technology to detect emotions and play music that matches the user's current mood.
2. Seamless Integration: Allows easy integration with popular music streaming platforms like Spotify or YouTube for an expansive song selection.
3. Real-Time Emotion Analysis: Captures and analyzes facial expressions in real-time to adjust the playlist dynamically based on mood changes.
4. Customized Playlists: Generates personalized playlists for different emotional states such as happiness, sadness, excitement, or relaxation.
5. User-Friendly Interface: Features an intuitive and visually appealing interface, making it accessible and enjoyable for users of all technical levels.
6. Offline and Online Mode: Offers both online and offline modes for music playback, ensuring uninterrupted listening experiences.
7. Interactive Song Suggestions: Provides users with interactive song suggestions and options to skip, repeat, or explore similar tracks.
8. High Accuracy Face Detection: Incorporates cutting-edge face recognition algorithms to ensure precise emotion detection across various lighting conditions and environments.
9. Multi-Device Compatibility: Works seamlessly across devices like smartphones, tablets, and desktops, allowing users to enjoy Moodify anywhere.
10. Enhanced Privacy Features: Ensures user data privacy by processing facial recognition data locally without storing sensitive information on external servers.

2.) ABSTRACT

The **Moodify** project focuses on developing a cutting-edge web application that combines facial recognition technology with music playback capabilities. This intelligent music player analyzes users' emotions in real-time using advanced face detection algorithms and plays songs tailored to their mood. Moodify leverages technologies such as Node.js, Express.js, React.js, and TensorFlow.js for seamless functionality, while tools like Git, GitHub, and Postman facilitate version control, collaboration, and API testing.

The application integrates AI-driven emotion detection and music streaming services to provide a personalized and engaging user experience. By employing real-time emotion analysis, Moodify dynamically adjusts playlists to align with the user's emotional state. The project highlights the fusion of artificial intelligence, machine learning, and modern web technologies to create a sophisticated music player that entertains, soothes, and adapts to the user's needs.

3.) INTRODUCTION

Welcome to Moodify, an innovative AI-powered music player developed as part of a final year project. Moodify combines the power of facial recognition technology with intelligent music recommendation systems to deliver a unique and personalized listening experience. This project aims to revolutionize how users interact with music by analyzing their emotions in real-time and curating playlists that match their mood.

Moodify provides a seamless and user-friendly interface, eliminating the need for complex logins or administrative access. The application features an intuitive navigation bar with essential links, including "Home" and "How It Works." The "Home" page serves as the central hub where users can engage with Moodify, start the emotion detection process, and enjoy curated music. The "How It Works" section offers a comprehensive guide to help users understand Moodify’s functionality, from emotion analysis to personalized music playback.

With Moodify, users can experience music tailored to their emotions, making it a perfect companion for any moment. The application uses advanced face detection algorithms to identify emotions such as happiness, sadness, anger, or surprise and automatically selects songs that resonate with the detected mood. Here’s a glimpse of what Moodify offers:

- Real-time Emotion Detection: Analyze your facial expressions in real time using advanced AI algorithms.
- Mood-Based Music Recommendations: Experience a playlist perfectly curated for your emotional state.
- Interactive Features: Play, pause, or skip tracks effortlessly through a responsive interface.
- Dynamic Adaptation: Switch moods, and Moodify adapts to reflect your emotional transition.
- Seamless Music Playback: Enjoy a wide range of songs without interruptions or delays.
- No Login Required: Dive straight into the experience without any barriers or registration.

Moodify is designed to entertain, uplift, and soothe users by combining modern technology with the universal appeal of music. Its advanced algorithms and natural language processing ensure accurate emotion recognition and seamless integration with music libraries.

Join us on this exciting journey with Moodify and redefine your music-listening experience. Let Moodify be your personal DJ, capable of understanding your emotions and creating a soundscape that aligns with your feelings. Together, we can explore the intersection of AI, music, and human emotions, transforming how we connect with technology and art.

Discover Moodify today—where emotions meet music, and technology meets creativity.

4.) SOFTWARE REQUIREMENT SPECIFICATION (SRS)

4.1 Introduction

a) Purpose

The purpose of this document is to define the requirements for the project "Moodify," an AI-powered music player. This document outlines the features and functionalities of the application, as well as the hardware and software configurations required for its operation. The focus is to specify the required technologies and system capabilities without mentioning the specific technologies used in the project's implementation..

b) Scope of Document Project

The purpose of this document is to outline the functionalities of the "Moodify" project, an AI-powered music player designed to enhance user experience by combining facial emotion recognition with dynamic music playback. The project focuses on leveraging innovative technology to provide personalized and seamless interaction.

1. User Functionality:

i. Emotion Detection:

Users can interact with Moodify through a user-friendly graphical interface.

The application detects the user's facial expressions in real-time to identify their emotional state.

ii. Mood-Based Music Playback:

Moodify dynamically generates and plays music tailored to the detected emotion.

Users can enjoy a personalized playlist without needing to manually select songs.

iii. Interactive Music Controls:

Users can control music playback with options to play, pause, skip, or stop tracks easily.

The application adjusts the music playlist dynamically as the user's mood changes.

iv. Simplified Access:

No login or registration is required, ensuring easy and hassle-free access for all users.

v. How It Works Section:

A dedicated guide is available to explain the steps and functionalities of Moodify, making it accessible for first-time users.

vi. Seamless Integration:

Users can operate Moodify on any device with a camera and internet connection, ensuring broad accessibility.

Moodify aims to transform the music-listening experience by intuitively responding to the user's emotional state and delivering a unique, engaging interaction.

2. Technology Stack:

i. Core Technologies:

- The project is developed using a combination of HTML, CSS, JavaScript, Bootstrap, Node.js, Express.js, Face-API.js, Git, GitHub, and VSCode.

ii. Front-End Development:

- HTML, CSS, and JavaScript are used to build the user-friendly graphical interface.

- Bootstrap enhances the design with responsive and modern UI components.

iii. Server-Side Development:

- Node.js and Express.js handle server-side logic, managing user requests and responses efficiently.

iv. Face Recognition Integration:

- Face-API.js is employed for real-time facial emotion detection, enabling mood-based functionalities.

v. Deployment and Hosting:

- The application is hosted on a reliable cloud platform for easy accessibility, ensuring optimal performance and availability.

vi. Version Control and Collaboration:

- Git and GitHub are used for tracking changes, version control, and team collaboration during development.

vii. Development Environment:

- VSCode serves as the primary integrated development environment (IDE) for coding, debugging, and testing.

Moodify's technology stack ensures a robust, scalable, and user-centric application that seamlessly integrates emotion detection with personalized music playback.

3. Assistance and Information:

i. Emotion-Based Music Personalization:

- Moodify analyzes users' facial expressions to detect their current mood and provides intelligent assistance in curating personalized music playlists based on their emotions.

ii. Real-Time Feedback:

- The application identifies emotional states like happiness, sadness, anger, or surprise and responds by suggesting music tracks that align with the user's detected mood.

iii. Dynamic Content Delivery:

- Moodify retrieves song suggestions and recommendations from external APIs and pre-defined music databases, ensuring an up-to-date and diverse selection of music for every mood.

4. Customizability and Scalability:

i. Flexibility for Personalization:

- Moodify allows customization of features, such as adding new music genres, integrating additional APIs, or enhancing the emotion-detection algorithm for greater accuracy and user satisfaction.

ii. Expandable Capabilities:

- The platform is designed to accommodate the addition of advanced features, such as multi-user profiles, improved emotion analysis, or integration with third-party streaming services.

iii. Scalable Architecture:

- Moodify’s architecture supports scalability, enabling it to handle a growing number of users, expanding music libraries, and incorporating additional AI-driven functionalities for an enhanced user experience.

The Moodify project aspires to revolutionize music personalization by combining advanced emotion recognition with intelligent music recommendations. With its customizable and scalable framework, the project is well-positioned to evolve alongside user needs, ensuring a dynamic and immersive musical journey.

Definitions, Acronyms and Abbreviations

1. NLP - Natural Language Processing
2. API - Application Programming Interface
3. UI - User Interface
4. UX - User Experience
5. SDLC - Software Development Life Cycle
6. JSON - JavaScript Object Notation
7. HTTPS - Hypertext Transfer Protocol Secure

4.2 Overall Descriptions

a) User Classes and Characteristics

Moodify caters to diverse user groups by offering a seamless and engaging music experience tailored to their emotions. The primary user classes include general users looking for a unique and personalized music experience based on their current mood, music enthusiasts who enjoy discovering new music, and students and professionals who rely on music to focus, relax, or boost productivity while working or studying. Fitness enthusiasts seeking energetic playlists to match their exercise routines or calming tunes for cool-down sessions are also a key audience. Additionally, mental wellness advocates can use the platform to enhance mental health and manage stress, anxiety, or emotions effectively.

Moodify provides an intuitive, user-friendly web interface that enables effortless interaction with the platform. The application requires a device with a web browser and a stable internet connection, ensuring accessibility for a wide range of users. Its versatile design and functionality deliver a customized and impactful musical journey for every individual.

b) Operating Environment

Moodify operates as a web-based application that can be accessed through any modern web browser. It is compatible with major operating systems, including Windows, macOS, and Linux. The platform is designed to work seamlessly across devices, providing users with an immersive experience regardless of their operating system. Moodify relies on technologies widely supported by contemporary web browsers, ensuring smooth functionality and a consistent experience across different devices and platforms. Assumptions and Dependencies The assumptions for Brainic Expert include:

1. The application should be accessible via modern web browsers such as Chrome, Firefox, Safari, and Edge.
2. Users will need a stable internet connection for uninterrupted access to the application, especially for features like music playback and face recognition.
3. The hardware configuration of users' devices should meet the minimum requirements for web browsing, streaming, and real-time face detection.

The dependencies for Moodify include:

1. Availability of external APIs for music playback, face recognition, and emotion detection.
2. Proper functioning of web browsers and technologies supporting video streaming, canvas rendering, and real-time processing.

c) Requirement

Software Configuration:
The Moodify application is developed using HTML, CSS, Bootstrap, ReactJS for the front end, JavaScript, and jQuery for client-side validation. Ajax is used for server-side validation, with Java & Node.js handling business logic. MongoDB serves as the database storage solution. The application utilizes TensorFlow for face recognition, Razorpay for payment integration, JUnit for Java testing, Postman for API testing, and SpringBoot as the Java framework. GitHub is used for repository creation, Azure for deployment, IntelliJ IDEA for backend development, and VSCode for frontend coding. Apache Tomcat acts as the web server.

Operating System: Mac OS, Linux, Windows 10, Windows 11
Front end: HTML, CSS, Bootstrap
Client-side Validation: JavaScript, jQuery
Server-side Validation: Ajax
Business Logic: JAVA, Node.js
Database: MongoDB
Web Server: Apache Tomcat
Testing Tools: JUnit, Postman
Payment Integration: Razorpay
Face Recognition: TensorFlow
IDE: IntelliJ IDEA, VSCode
Framework: ReactJS, SpringBoot
Deployment: Azure

Hardware Configuration:

Processor: Core i3, 1.5MHz

Hard Disk: 150 GB

RAM: 8GB

Resolution: 480 X 800

d) Data Requirement

The Moodify system takes input from the user through face recognition and emotion detection, analyzing facial expressions to determine the user’s mood. Based on this analysis, it generates personalized music recommendations and plays the appropriate songs. The system also integrates with external APIs to provide additional music-related data, such as song details and artist information. Moodify acts as an intelligent, interactive music player that adapts to the user's emotional state, offering a seamless and personalized music experience.

4.3 External User Interface

a) GUI

The Moodify project does not include a login functionality, focusing solely on personalized music recommendations based on mood analysis. The GUI will be designed to provide a seamless and engaging experience for users based on their facial expressions, with the system responding to their emotional state through music selection.

The Moodify project's GUI will include the following features:

- ❖ Customizable Interface: The GUI will be customizable to enhance user experience, providing a personalized layout that adapts to the user's mood and preferences.
- ❖ Modular Design: Different modules, such as music selection, mood analysis, and recommendations, will be integrated into a consistent and easy-to-use interface.
- ❖ Simple and Standard Design: The design will be clean, intuitive, and user-friendly, ensuring ease of navigation. All pages and interactions will follow a standard template, simplifying the overall experience.
- ❖ Mood Detection Interface: The GUI will display real-time mood analysis results based on facial recognition, helping users understand their emotional state and offering tailored music choices.
- ❖ Interactivity: Users will engage with the Moodify system by simply interacting with the interface. No text-based input will be required, as the system automatically adapts to the user's mood through facial expression detection.

- Audio Feedback: The system will provide spoken responses where necessary, enhancing the user experience by combining visual and auditory feedback.
- Personalized Music Recommendations: The GUI will feature real-time music recommendations that adapt to the user's mood, providing personalized playlists based on detected emotions.

The focus of the GUI will be on a smooth and immersive experience, where users can enjoy music tailored to their emotional state. The interface will be simple, engaging, and responsive, with mood detection and music playback as core functionalities.

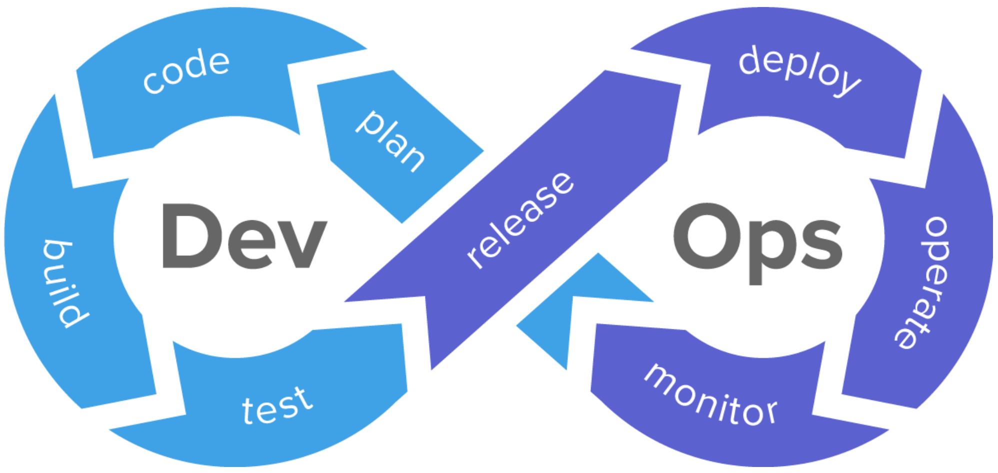
4.4 System Features

The Moodify project will provide a platform for users to receive personalized music recommendations based on their mood. It will not require traditional user login credentials but may offer optional user profiles for better personalization. The system will integrate mood detection via facial recognition and suggest music based on users' emotional states. Users will be able to enjoy music in real-time, with the system offering personalized song choices, mood analysis, and additional features like music metadata display.

5).SYSTEM PLANNING & SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

We have chosen “**DevOps** Life Cycle Model” for developing this application, because an DevOps life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.

Here is the diagram of the DevOps life cycle model which depicts its working flow.



Few advantages for choosing the SDLC are –

- In DevOps model we are building and improving the product step by step. Hence, we can track the defects at early stages. This avoids the downward flow of the defects.
- Testing and debugging in smaller iteration is easy.

- In DevOps model we can get the reliable user feedback. When presenting sketches and blueprints of the product to user for their feedback, we are effectively asking them to imagine how the product will work.
- Progress can be measured.
- In DevOps model less time is spent on documentation and more time is given for development.
- Risk are identified and resolved during iteration; and each iteration is an easily managed milestone. It supports changing requirements.

6.)CONSTRANTS OF USE

- **No User Accounts Required**: Branic Expert does not mandate user accounts or formal login credentials. The system operates without the need for users to remember login IDs and passwords, making it easy and hassle-free for users to access information.
- **Internet Connection**: Users must have access to a personal computer or cell phone with an active internet connection to interact with the Branic Expert system. The system operates as an online service, providing information through conversational queries and speech synthesis.

  The Branic Expert project ensures a user-friendly experience by eliminating the need for user accounts and login credentials. Users can seamlessly access the system through their devices and an internet connection, making information generation quick and efficient without any additional complexities.

7.)TECHNOLOGY USED

7.1 Front-end Design:

7.1.1 HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

7.1.2 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

7.1.3 Bootstrap

Bootstrap is a free and open-source front-end library for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only.

7.1.4 JavaScript

JavaScript often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm. Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it.

7.1.5 React JS

React's primary role in an application is to handle the view layer of that application just like the V in a model-view-controller (MVC) pattern by providing the best and most efficient rendering execution. Rather than dealing with the whole user interface as a single unit, React.js encourages developers to separate these complex UIs into individual reusable components that form the building blocks of the whole UI. In doing so, the ReactJS framework combines the speed and efficiency of JavaScript with a more efficient method of manipulating the DOM to render web pages faster and create highly dynamic and responsive web applications.

7.1.6 NPM

npm, which stands for Node Package Manager, is a widely used package manager for Node.js and JavaScript projects. It simplifies the process of installing, managing, and sharing various libraries, frameworks, and tools within a project. With npm, developers can easily access a vast repository of open-source packages, making it an indispensable tool for modern web development. By providing a simple and efficient way to handle dependencies and streamline project workflows, npm significantly contributes to the growth and scalability of JavaScript-based applications.

7.2 Back-end Design:

7.2.1 Node JS

Node.js is an open source, cross-platform runtime environment and library that is used for running web applications outside the client's browser. It is used for server-side programming, and primarily deployed for non-blocking, event-driven servers, such as traditional web sites and back-end API services, but was originally designed with real-time, push-based architectures in mind. Every browser has its own version of a JS engine, and node.js is built on Google Chrome's V8 JavaScript engine.

7.2.2 Express JS

Express.js is a popular and minimalist web application framework for Node.js, designed to simplify the process of building robust and scalable web applications and APIs. It provides a set of powerful features and middleware that enable developers to create server-side applications with ease. Express.js follows the "middleware" architecture, allowing developers to customize and extend functionalities by adding various middleware components. Its simplicity, flexibility, and extensive community support make it a preferred choice for developers when building web applications and APIs in the Node.js ecosystem.

7.2.3 REST- API

A REST API (Representational State Transfer Application Programming Interface) is a software architectural style used for designing and implementing web services. It follows the principles of simplicity, scalability, and statelessness. In a REST API, resources are identified by unique URLs, and operations (such as retrieving, creating, updating, or deleting data) are performed using standard HTTP methods like GET, POST, PUT, and DELETE. REST APIs enable seamless communication between different applications and systems over the internet, making it a widely adopted approach for building modern, flexible, and interoperable web services.

7.3: Testing Tools

7.3.1 ESLint

ESLint is a powerful static code analysis tool for JavaScript that helps developers maintain consistent and error-free code. It enforces coding standards, detects potential bugs, and highlights stylistic issues in the codebase. By integrating ESLint into the development workflow, teams can improve code quality, enhance readability, and prevent common programming mistakes. It provides customizable rules that can be tailored to match the specific coding conventions of a project. ESLint has become an essential tool for modern JavaScript development, enabling developers to write cleaner and more maintainable code.

7.3.2 Postman

Postman is an API(application programming interface) development tool which helps to build, test and modify APIs. Almost any functionality that could be needed by any developer is encapsulated in this tool.It has the ability to make various types of HTTP requests(GET, POST, PUT, PATCH), saving environments for later use, converting the API to code for various languages(like JavaScript).

7.4: Intregated Development Kit (IDE)

7.4.1 Visual Studio Code

Visual Studio Code is a free, lightweight but powerful source code editor that runs on your desktop and on the web and is available for Windows, macOS, Linux, and Raspberry Pi OS. It comes with built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other programming languages (such as C++, C#, Java, Python, PHP, and Go), runtimes (such as .NET and Unity), environments (such as Docker and Kubernetes), and clouds (such as Amazon Web Services, Microsoft Azure, and Google Cloud Platform).

7.5: Version Control Tool

7.5.1 Git

Git is a distributed version control system: tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

7.5.2 GitHub

GitHub, Inc. is an Internet hosting service for software development and version control using Git. It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project.

7.6: Deploy

7.6.1 Microsoft Azure

Microsoft Azure, often referred to as Azure is a cloud computing platform operated by Microsoft for application management via around the world-distributed data centers. Microsoft Azure has multiple capabilities such as software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) and supports many different programming languages, tools, and frameworks, including both Microsoft-specific and third-party software and systems.

7.6.1 FileZilla

FileZilla is a free and open-source, cross-platform FTP application, consisting of FileZilla Client and FileZilla Server. Clients are available for Windows, Linux, and macOS. Both server and client support FTP and FTPS, while the client can in addition connect to SFTP servers.

8.)FEASIBILITY STUDY

A feasibility study is an analysis of how successfully a project can be completed, accounting for factors that affect it such as economic, technological and operational. Project managers use feasibility studies to determine potential positive and negative outcomes of a project before investing a considerable amount of time and money into it.

During the stage of our feasibility study, we had to undergo the following steps as described under:

- Identify the origin of data at different levels of the system.
- Identify the expectation of end user from the finished product/system.
- Analyze the drawback(s) of the existing system.

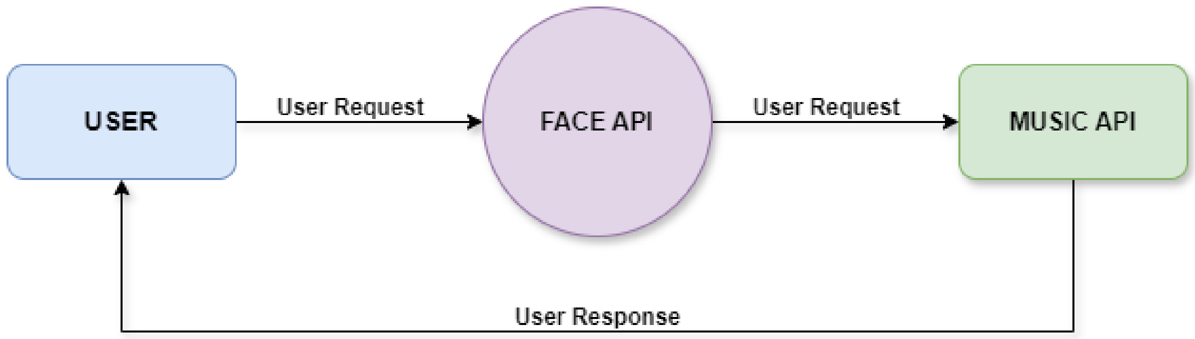
- ➢ **Technical feasibility study:** It lays out details on how a good or service will be delivered, which includes transportation, business location, technology needed, materials and labour.

- ➢ **Financial feasibility study:** It is a projection of the amount of funding or startup capital needed, what sources of capital can and will be used and what kind of return can be expected on the investment.

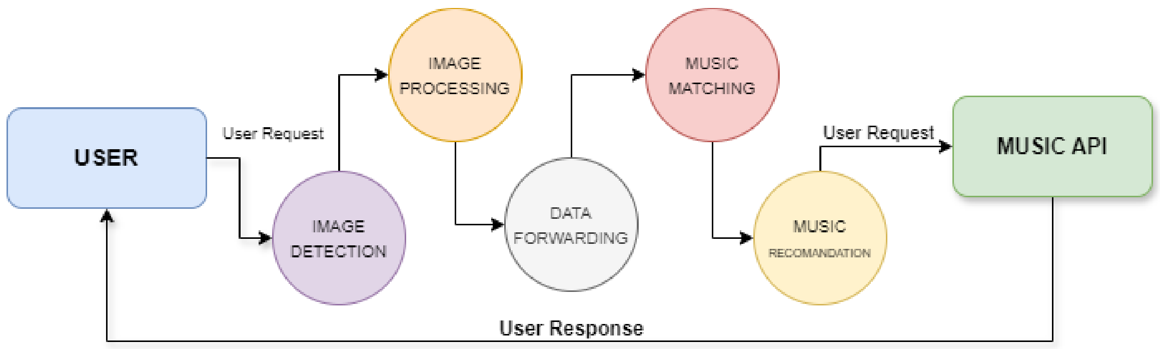
- ➢ **Organizational feasibility study:** It is a definition of the corporate and legal structure of the business; this may include information about the founders, their professional background and the skills they possess necessary to get the company off the ground and keep it operational.

9.)DATA FLOW DIAGRAM(DFD)

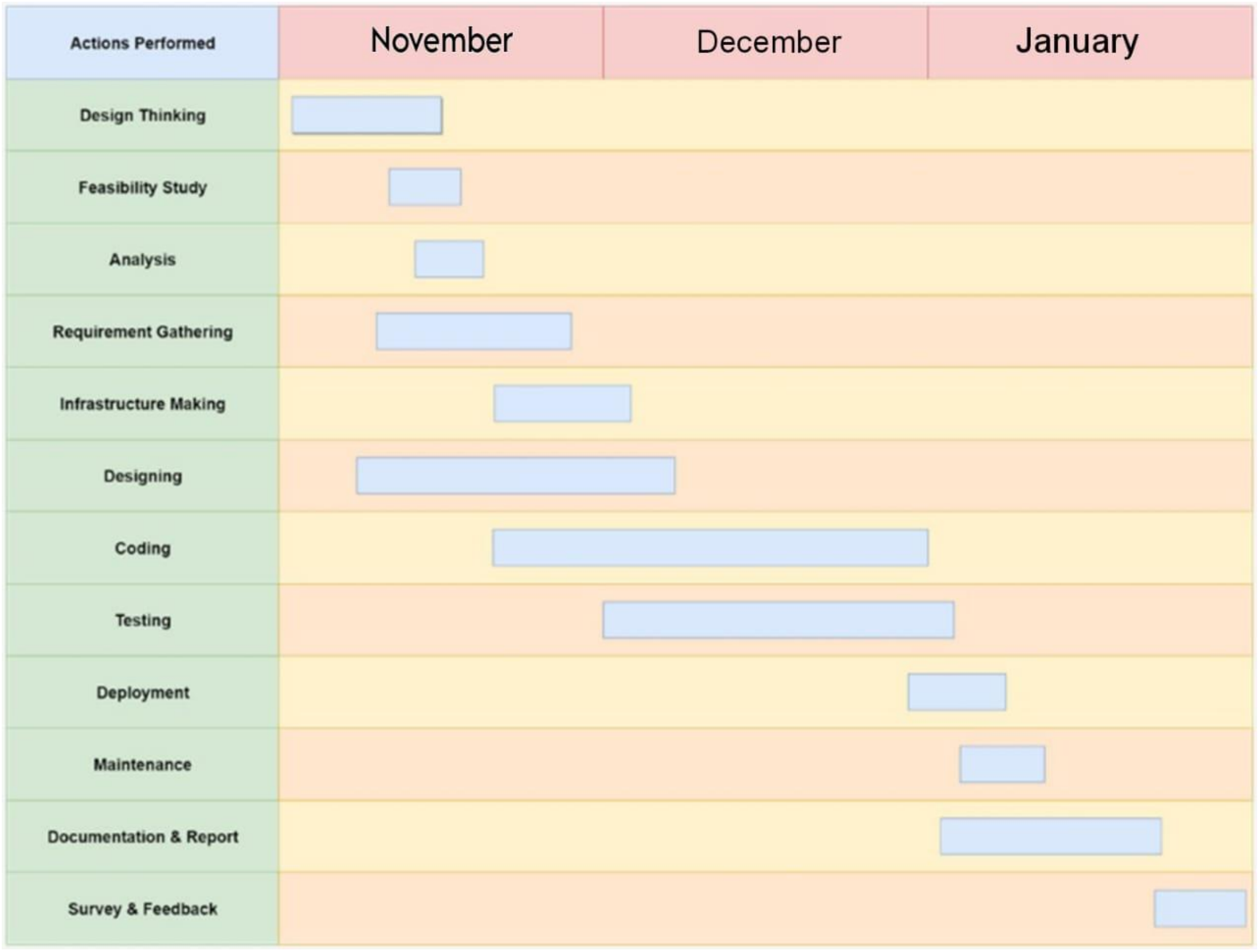
- level 0



- level 1



10.) GANTT CHART



11.)UNIFIED MODELING LANGUAGE (UML) DIAGRAM

13.1 Use Case Diagram

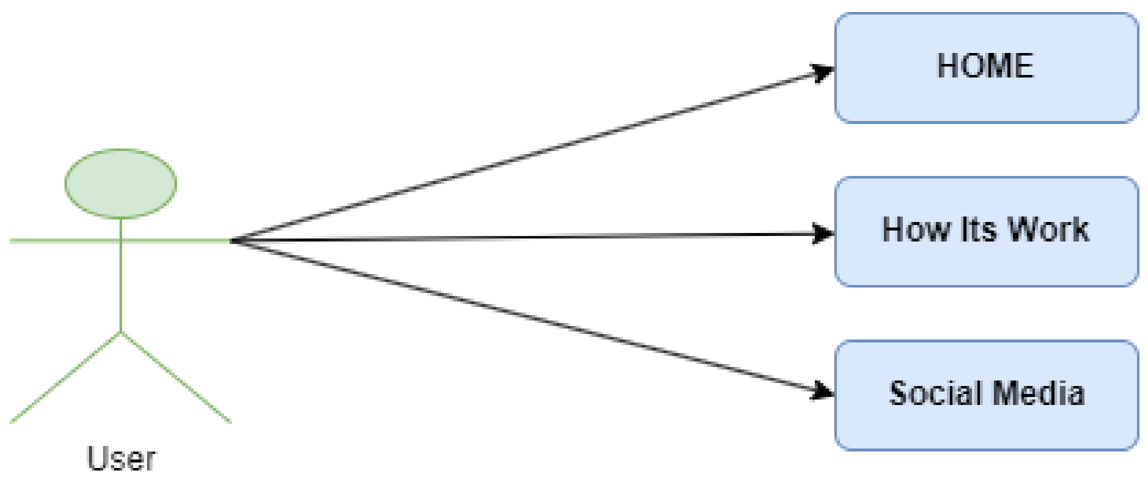
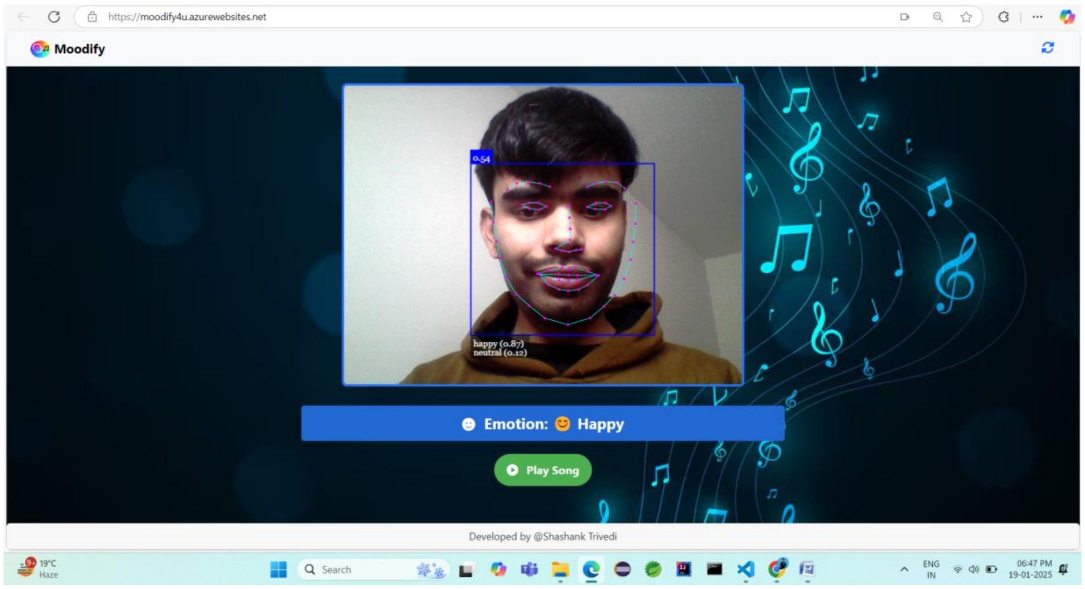


Fig: USE CASE DIAGRAM OF APPLICATION PAGE

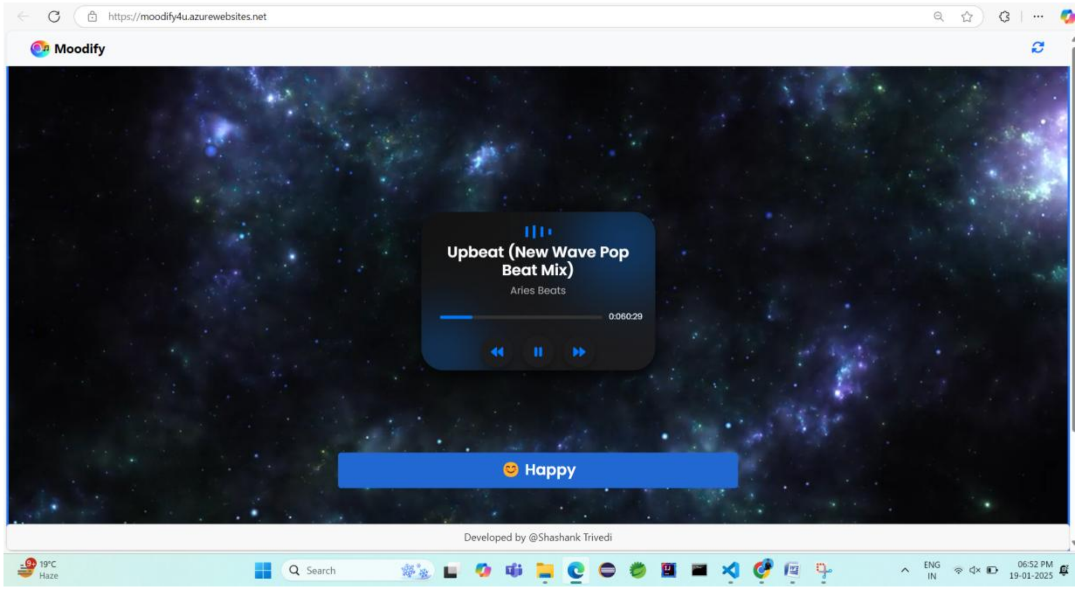
12.)SCREENSHOTS OF APPLICATION

14.1 Home Page



https://moodify4u.azurewebsites.net
Moodify
happy (0.87)
neutral (0.12)
Emotion: 😊 Happy
Play Song
Developed by @Shashank Trivedi

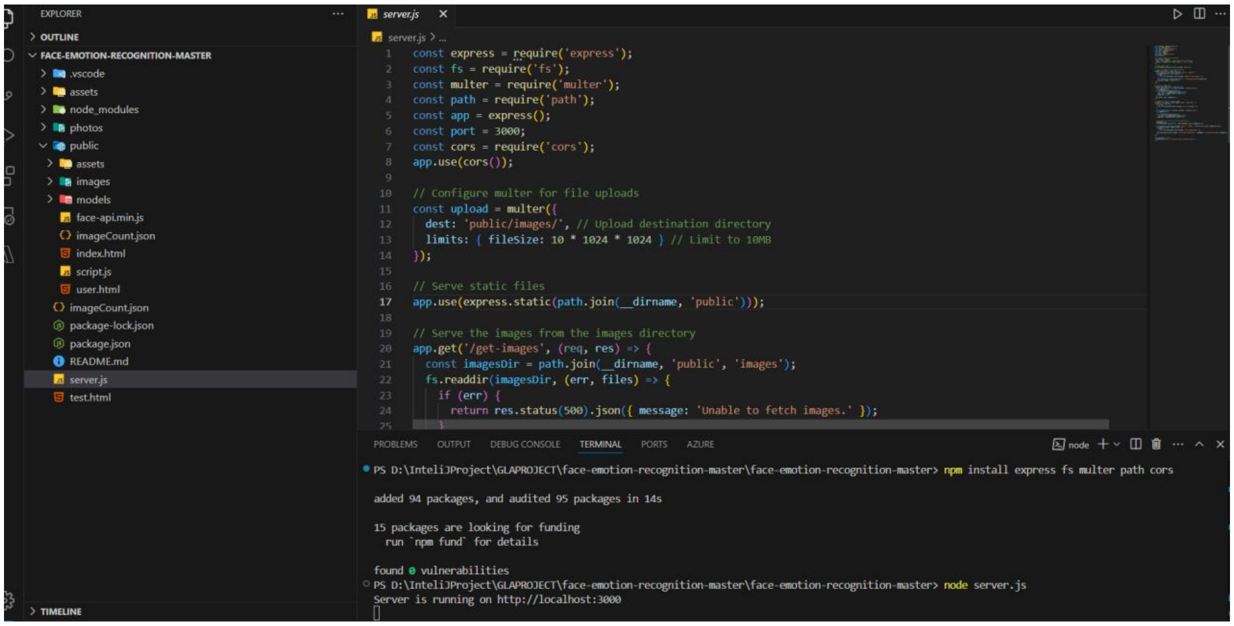
14.2 Music page



https://moodify4u.azurewebsites.net
Moodify
Upbeat (New Wave Pop Beat Mix)
Aries Beats
0:06 0:29
😊 Happy
Developed by @Shashank Trivedi

13.)SCREENSHOTS OF TOOLS

15.1 Backend



```javascript
const express = require('express');
const fs = require('fs');
const multer = require('multer');
const path = require('path');
const app = express();
const port = 3000;
const cors = require('cors');
app.use(cors());

// Configure multer for file uploads
const upload = multer({
  dest: 'public/images/', // Upload destination directory
  limits: { fileSize: 10 * 1024 * 1024 } // Limit to 10MB
});

// Serve static files
app.use(express.static(path.join(__dirname, 'public')));

// Serve the images from the images directory
app.get('/get-images', (req, res) => {
  const imagesDir = path.join(__dirname, 'public', 'images');
  fs.readdir(imagesDir, (err, files) => {
    if (err) {
      return res.status(500).json({ message: 'Unable to fetch images.' });
    }
```

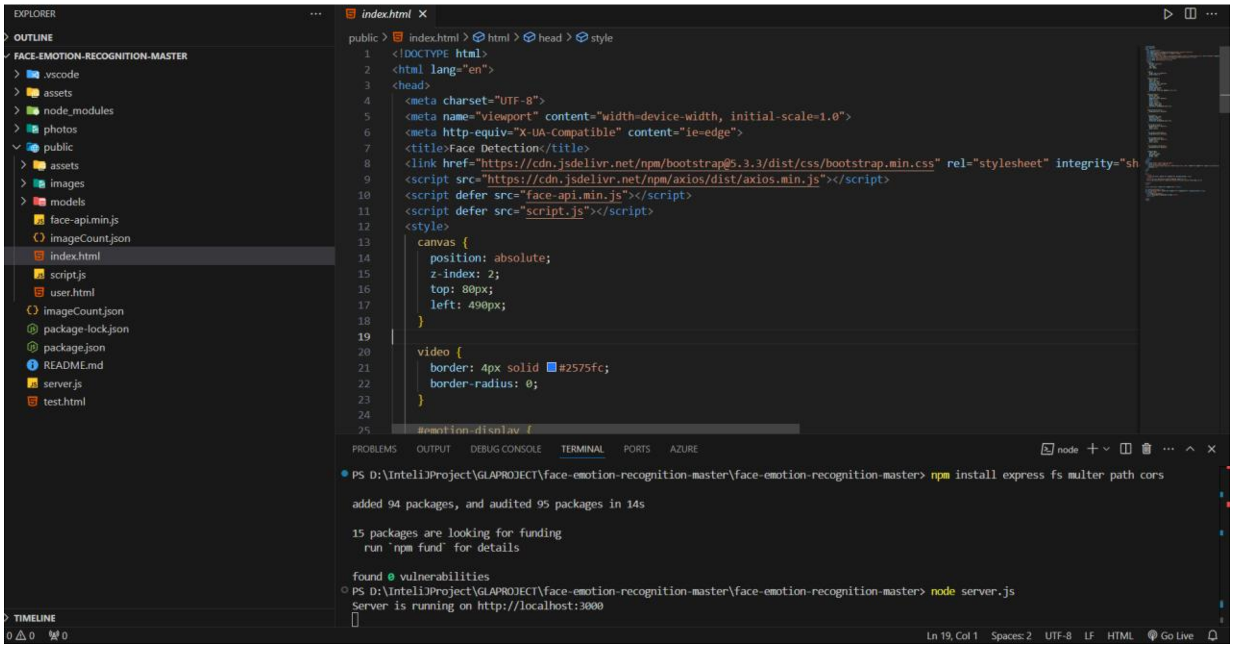
```
PS D:\InteliJProject\GLAPROJECT\face-emotion-recognition-master\face-emotion-recognition-master> npm install express fs multer path cors

added 94 packages, and audited 95 packages in 14s

15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS D:\InteliJProject\GLAPROJECT\face-emotion-recognition-master\face-emotion-recognition-master> node server.js
Server is running on http://localhost:3000
```

15.2 Frontend



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Face Detection</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sh
  <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
  <script defer src="face-api.min.js"></script>
  <script defer src="script.js"></script>
  <style>
    canvas {
      position: absolute;
      z-index: 2;
      top: 80px;
      left: 490px;
    }

    video {
      border: 4px solid #2575fc;
      border-radius: 0;
    }

    #emotion-display {
```

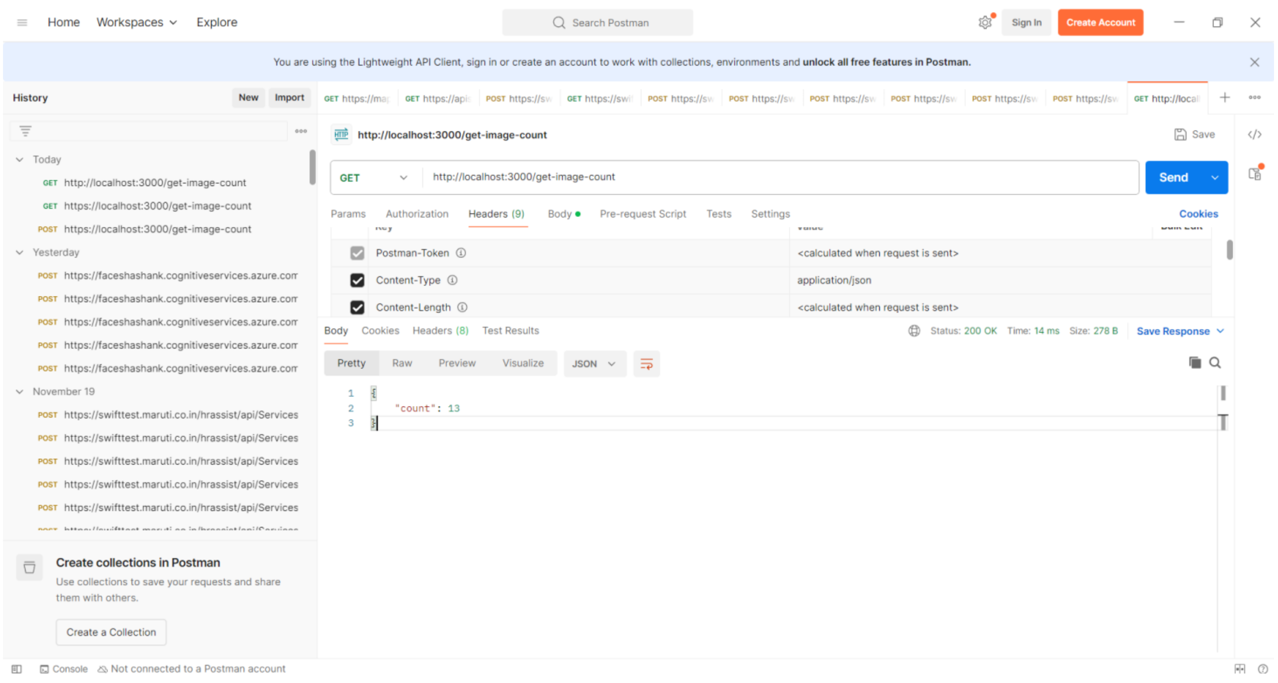
```
PS D:\InteliJProject\GLAPROJECT\face-emotion-recognition-master\face-emotion-recognition-master> npm install express fs multer path cors

added 94 packages, and audited 95 packages in 14s

15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS D:\InteliJProject\GLAPROJECT\face-emotion-recognition-master\face-emotion-recognition-master> node server.js
Server is running on http://localhost:3000
```

15.3 Testing on Postman



You are using the Lightweight API Client, sign in or create an account to work with collections, environments and **unlock all free features in Postman.**

History

Today
- GET http://localhost:3000/get-image-count
- GET https://localhost:3000/get-image-count
- POST https://localhost:3000/get-image-count

Yesterday
- POST https://faceshashank.cognitiveservices.azure.com
- POST https://faceshashank.cognitiveservices.azure.com
- POST https://faceshashank.cognitiveservices.azure.com
- POST https://faceshashank.cognitiveservices.azure.com
- POST https://faceshashank.cognitiveservices.azure.com

November 19
- POST https://swifttest.maruti.co.in/hrassist/api/Services
- POST https://swifttest.maruti.co.in/hrassist/api/Services
- POST https://swifttest.maruti.co.in/hrassist/api/Services
- POST https://swifttest.maruti.co.in/hrassist/api/Services
- POST https://swifttest.maruti.co.in/hrassist/api/Services

Create collections in Postman

Use collections to save your requests and share them with others.

Create a Collection

http://localhost:3000/get-image-count

GET http://localhost:3000/get-image-count

Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings

	Key	Value
✓	Postman-Token	<calculated when request is sent>
✓	Content-Type	application/json
✓	Content-Length	<calculated when request is sent>

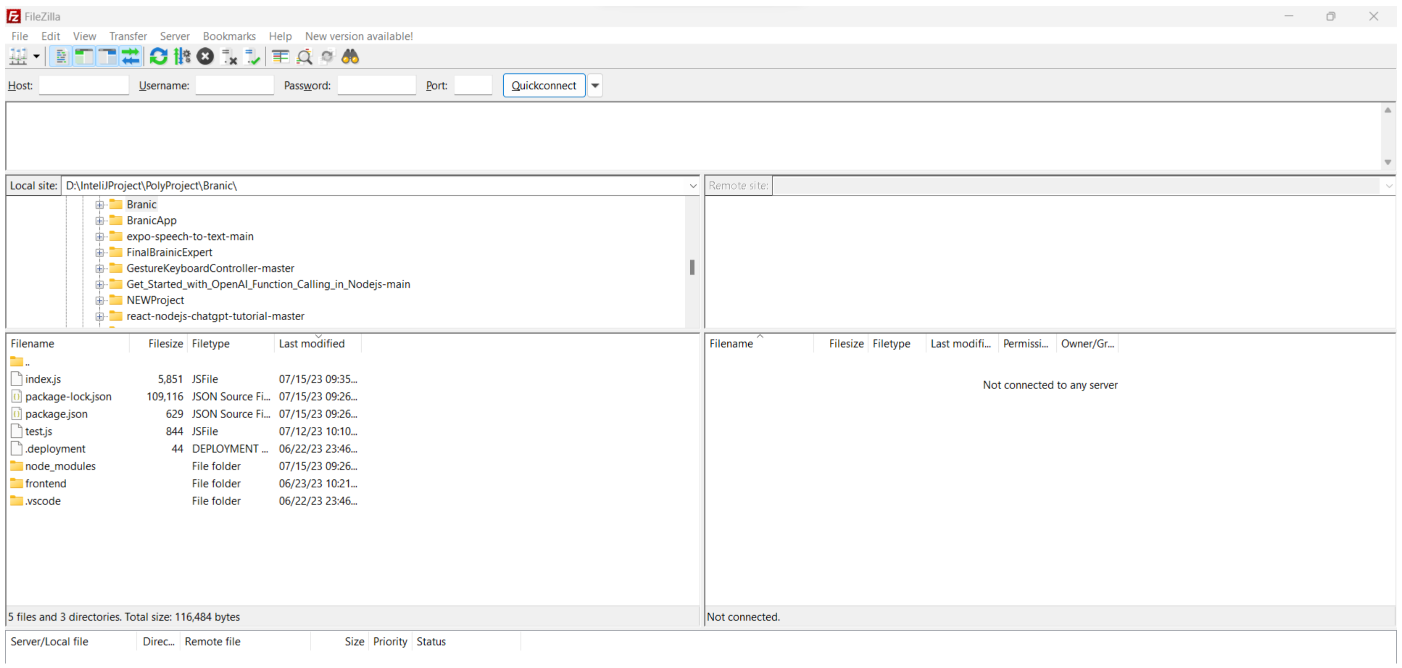
Body | Cookies | Headers (8) | Test Results — Status: 200 OK Time: 14 ms Size: 278 B

Pretty | Raw | Preview | Visualize | JSON

```
{
    "count": 13
}
```

Console | Not connected to a Postman account

15.4 Deployment from FileZila


FileZilla
File Edit View Transfer Server Bookmarks Help New version available!
Host: Username: Password: Port: Quickconnect

Local site: D:\InteliJProject\PolyProject\Branic\

- Branic
- BranicApp
- expo-speech-to-text-main
- FinalBrainicExpert
- GestureKeyboardController-master
- Get_Started_with_OpenAI_Function_Calling_in_Nodejs-main
- NEWProject
- react-nodejs-chatgpt-tutorial-master

Remote site:

Filename	Filesize	Filetype	Last modified
..			
index.js	5,851	JSFile	07/15/23 09:35...
package-lock.json	109,116	JSON Source Fi...	07/15/23 09:26...
package.json	629	JSON Source Fi...	07/15/23 09:26...
test.js	844	JSFile	07/12/23 10:10...
.deployment	44	DEPLOYMENT ...	06/22/23 23:46...
node_modules		File folder	07/15/23 09:26...
frontend		File folder	06/23/23 10:21...
.vscode		File folder	06/22/23 23:46...

5 files and 3 directories. Total size: 116,484 bytes

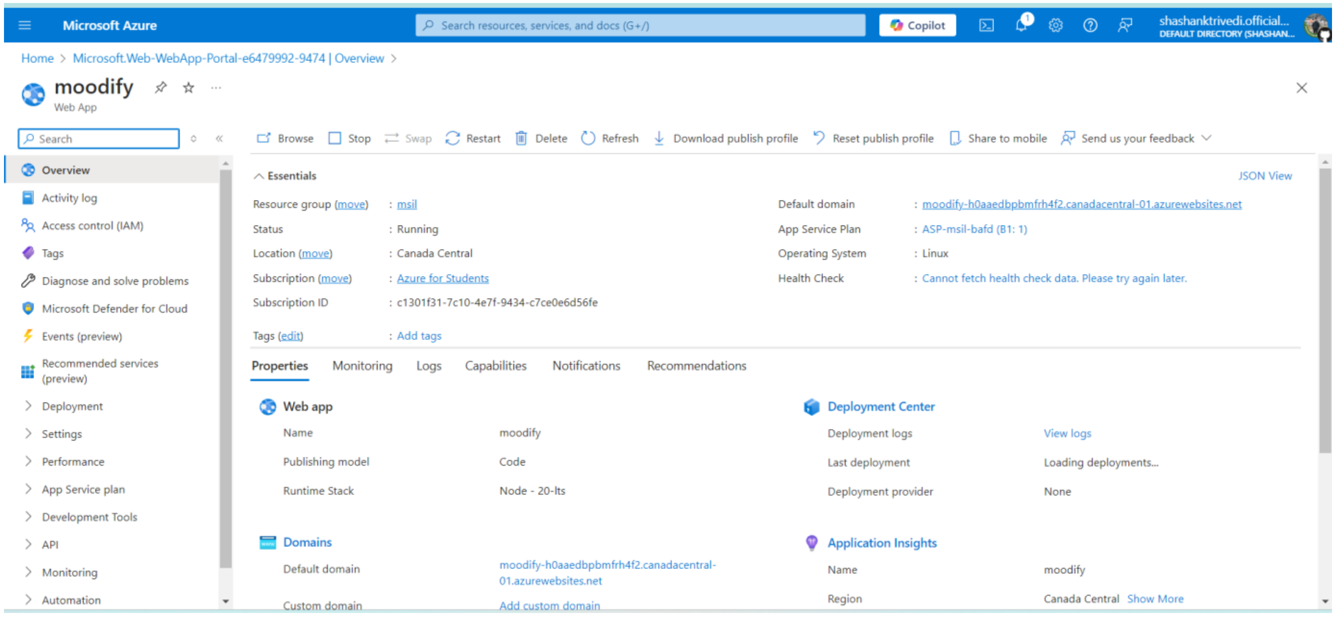
Filename	Filesize	Filetype	Last modifi...	Permissi...	Owner/Gr...

Not connected to any server

Not connected.

Server/Local file	Direc...	Remote file	Size	Priority	Status

15.5 Deploy on Azure



Resource group (move) : msil
Status : Running
Location (move) : Canada Central
Subscription (move) : Azure for Students
Subscription ID : c1301f31-7c10-4e7f-9434-c7ce0e6d56fe
Tags (edit) : Add tags

Default domain : moodify-h0aaedbpbmfrh4f2.canadacentral-01.azurewebsites.net
App Service Plan : ASP-msil-bafd (B1: 1)
Operating System : Linux
Health Check : Cannot fetch health check data. Please try again later.

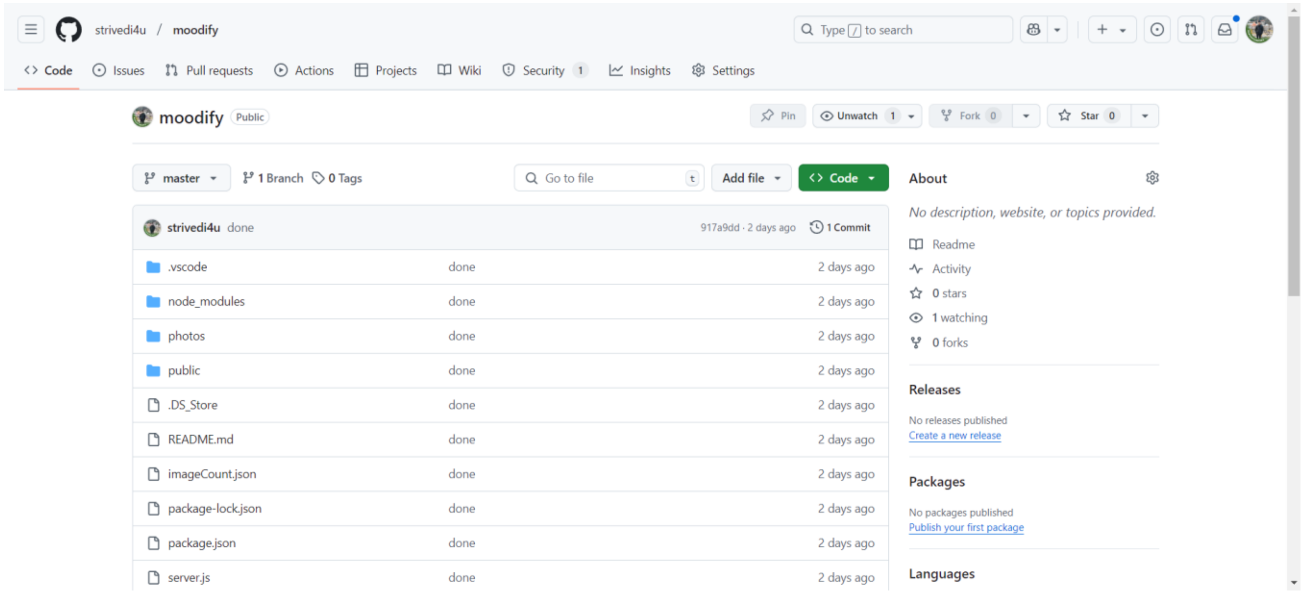
Web app
Name moodify
Publishing model Code
Runtime Stack Node - 20-lts

Deployment Center
Deployment logs View logs
Last deployment Loading deployments...
Deployment provider None

Domains
Default domain moodify-h0aaedbpbmfrh4f2.canadacentral-01.azurewebsites.net
Custom domain Add custom domain

Application Insights
Name moodify
Region Canada Central Show More

15.6 Backup on GitHub



strivedi4u done 917a9dd · 2 days ago 1 Commit

.vscode	done	2 days ago
node_modules	done	2 days ago
photos	done	2 days ago
public	done	2 days ago
.DS_Store	done	2 days ago
README.md	done	2 days ago
imageCount.json	done	2 days ago
package-lock.json	done	2 days ago
package.json	done	2 days ago
server.js	done	2 days ago

About
No description, website, or topics provided.
Readme
Activity
0 stars
1 watching
0 forks

Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

Languages

14.)SOURCE CODE

1 Index.htm in Frontend

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Face Detection</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
  <script defer src="face-api.min.js"></script>
  <script defer src="script.js"></script>
  <style>
    canvas {
      position: absolute;
      z-index: 2;
      top: 80px;
      left: 490px;
    }

    video {
      border: 4px solid #2575fc;
      border-radius: 0;
    }

    #emotion-display {
      width: 45%;
      color: #ffffff;
      font-size: 24px;
      background-color: #2168d2;
      font-weight: bold;
      padding: 10px;
      border-radius: 5px;
      margin-top: 20px;
      box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    }

    #emotion-button {
      bottom: 80px;
      font-size: 18px;
      background-color: #2575fc;
      border: none;
      color: white;
      cursor: pointer;
      border-radius: 5px;
      padding: 10px 20px;
      transition: background-color 0.3s;
    }

    #emotion-button:hover {
      background-color: #2168d2;
    }

    .btn-custom {
      padding: 10px 20px;
      font-size: 24px;
      border-radius: 5px;
      transition: background-color 0.3s;
    }

    .btn-primary-custom {
      background-color: #2575fc;
      border: none;
      color: white;
    }

    .btn-primary-custom:hover {
      background-color: #2168d2;
    }

    .btn-secondary-custom {
      background-color: #d61c1c;
      border: none;
      color: white;
    }

    .btn-secondary-custom:hover {
      background-color: #b01616;
    }

    #song-iframe {
      width: 100%;
      max-width: 720px;
      height: 400px;
      border: none;
    }
  </style>
</head>
<body>
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">
    <img src="/assets/images/logo.png" style="margin-left: 15px;" width="35" height="30"
class="d-inline-block align-top" alt="">
    <b>Moodify </b>
  </a>
</nav>
<br>

<div>
  <center>
    <video id="video" width="720" height="540" autoplay muted></video>
  </center>
  <center><br><div id="emotion-display">Emotion: None</div><br>
  <button id="emotion-button" class="btn-custom btn-primary-custom">Play Song</button>

</center>
</div>

<canvas id="canvas" width="720" height="560"></canvas>

<div id="song-message"></div>
<iframe id="song-iframe" width="560" height="315" frameborder="0" allowfullscreen></iframe>
<div class="card">
  <div class="card-header">
   <center>Developed by @Shashank Trivedi</center>
  </div></div>

</body>
</html>


```

2 server.js in Backend

```
const express = require('express');
const fs = require('fs');
const multer = require('multer');
const path = require('path');
const app = express();
const port = 3000;
const cors = require('cors');
app.use(cors());

// Configure multer for file uploads
const upload = multer({
  dest: 'public/images/', // Upload destination directory
  limits: { fileSize: 10 * 1024 * 1024 } // Limit to 10MB
});

// Serve static files
app.use(express.static(path.join(__dirname, 'public')));

// Serve the images from the images directory
app.get('/get-images', (req, res) => {
  const imagesDir = path.join(__dirname, 'public', 'images');
  fs.readdir(imagesDir, (err, files) => {
    if (err) {
      return res.status(500).json({ message: 'Unable to fetch images.' });
    }
    const imageFiles = files.filter(file => /\.(jpg|jpeg|png|gif)$/.test(file));
    res.json({ images: imageFiles });
  });
});

// Serve the image count
app.get('/get-image-count', (req, res) => {
  const imageCountFile = path.join(__dirname, 'imageCount.json');
  let imageCount = 0;
  if (fs.existsSync(imageCountFile)) {
    const data = fs.readFileSync(imageCountFile);
    const count = JSON.parse(data).count || 0;
    imageCount = count;
  }
  res.json({ count: imageCount });
});

// Endpoint to save the captured image
app.post('/save-image', upload.single('image'), (req, res) => {
  const file = req.file;
  if (!file) {
    return res.status(400).json({ message: 'No file uploaded' });
  }

  const imageCountFile = path.join(__dirname, 'imageCount.json');
  let imageCount = 0;

  if (fs.existsSync(imageCountFile)) {
    const data = fs.readFileSync(imageCountFile);
    imageCount = JSON.parse(data).count || 0;
  }

  // Increment the count and save the updated value
  imageCount += 1;
  fs.writeFileSync(imageCountFile, JSON.stringify({ count: imageCount }));

  const filePath = path.join(__dirname, 'public', 'images', `face-detection-image-${imageCount}.png`);
  fs.rename(file.path, filePath, (err) => {
    if (err) {
      return res.status(500).json({ message: 'Error saving image.' });
    }
    res.status(200).json({ message: 'Image saved successfully.', imageName: `face-detection-image-${imageCount}.png` });
  });
});

app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});

```

15.)FUTURE SCOPE OF WORK

The future scope of the Moodify project is broad and full of potential, with plans for continuous expansion and enhancement to meet the evolving needs of its users. As the system evolves, it can find applications in various industries, particularly in sectors that require personalized music recommendations based on mood or facial expressions.

Future prospects for **Moodify** include:

- **Diverse Organizational Implementation**: The system can be deployed in various organizations, such as entertainment platforms, fitness centers, or even healthcare settings, providing personalized music recommendations that improve user engagement and overall satisfaction.

- **Integration of Advanced AI and Smart Devices**: Moodify can integrate advanced AI systems like voice assistants, IoT devices, or even smart speakers, enhancing user experience by enabling voice commands or facial recognition features to control music playback.

- **Mobile Integration**: Moodify can be expanded into mobile applications, making it accessible to users on the go. Mobile integration would allow the system to deliver personalized music recommendations based on real-time mood assessments and context, such as location or activity.

- **Secure and Scalable Backend**: With user data being an integral part of personalized music experiences, Moodify will employ secure backend technologies to ensure data privacy and security. Scalable cloud infrastructure will accommodate increasing user loads and data processing needs, ensuring seamless performance.

- **Data Analytics and Personalization**: Data analytics can be applied to better understand user preferences and improve recommendations. Analyzing user behavior, mood patterns, and music choices could lead to smarter, more accurate music suggestions tailored to individuals.

- **Integration with Wearable Devices**: By incorporating wearable technology (e.g., fitness trackers or smartwatches), Moodify can receive real-time data to adjust music recommendations based on physical activity, heart rate, or even sleep patterns.

In conclusion, the future scope of **Moodify** is vast, offering opportunities for growth and adaptation. As the project progresses, it can continue to provide innovative and personalized music experiences while integrating with emerging technologies and ensuring a seamless user experience.

16.)CONCLUSION

The Moodify project is a groundbreaking initiative designed to provide users with personalized music recommendations based on their mood. The system aims to offer an interactive and engaging experience, using facial recognition technology to detect users' emotional states and tailor music suggestions accordingly. Throughout the development process, a strong focus has been placed on user experience, ensuring that the application is both intuitive and enjoyable to use.

The primary objective of the software planning for Moodify is to establish a framework that supports efficient mood detection and music recommendation. The system has been designed to provide accurate music suggestions in real time, ensuring that users always have the perfect soundtrack to match their emotional state. As the project progresses, it will be continuously improved to adapt to changing user preferences and to enhance the overall performance.

The Moodify project represents a significant step toward creating a more personalized and emotionally aware music experience. By focusing on user satisfaction and leveraging advanced technologies like facial recognition, the project aims to offer a truly unique way for users to interact with music. As the system evolves, it will continue to provide relevant and tailored music experiences, helping users find the perfect soundtrack for every mood.

17.)BIBLIOGRAPHY

- ✓ **TensorFlow Documentation**: Official documentation for TensorFlow - https://www.tensorflow.org/learn
- ✓ **Kaggle**: Platform for data science and machine learning - https://www.kaggle.com
- ✓ **React Documentation**: Official documentation for React - https://reactjs.org/docs/
- ✓ **Express.js Documentation**: Official documentation for Express.js - https://expressjs.com/en/4x/api.html
- ✓ **npm Documentation**: Official documentation for npm (Node Package Manager) - https://docs.npmjs.com/
- ✓ **Node.js Documentation**: Official documentation for Node.js - https://nodejs.org/en/docs/
- ✓ **OpenAI Documentation**: Official documentation for OpenAI's APIs and services - https://platform.openai.com/docs/introduction
- ✓ **Wikipedia**: www.wikipedia.org

- ✓ The **Moodify** project uses TensorFlow and Kaggle as the primary AI tools to power its face recognition and music recommendation system, integrating them into a dynamic and interactive web platform. These resources have provided the foundation for integrating advanced AI functionalities into the application.