SMART TRAVEL RECOMMENDATION AND TOURISM SUPPORT MOBILE BASED SYSTEM

2023-308

B.Sc. (Hons) Degree in Information Technology
Specializing in Information Technology

Department of Information Technology

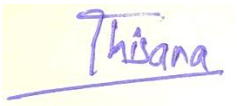
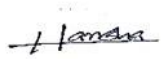
Sri Lanka Institute of Information Technology

November 2023

DECLARATION

We declare that this is our own work and this dissertation1 does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, we hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. We retain the right to use this content in whole or part in future works (such as articles or books).

Name	Student Registration No.	Signature
Athukorala Y.J.	IT20029968	
Shaminda W.G.T.	IT20051020	
Jayawardhana E.H.K	IT19192024	
Thennakoon S.U.	IT20147846	

The above candidate has carried out research for the bachelor's degree Dissertation under my supervision.

Supervisor Name	Signature	Date
Mrs. Gaya Thamali Dassanayaka		
Mr. Samantha Rajapaksha		

ACKNOWLEDGEMENT

I am incredibly appreciative of all the help and advice I have had while finishing my dissertation. Without the collaborative efforts and contributions of numerous people, this research project would not have been able to be completed successfully.

First and foremost, I want to thank my dissertation supervisor, Ms. Gaya Thamali Dassanayake, from the bottom of my heart. The foundation of this project has been their constant dedication to mentoring and their priceless guidance. The path and quality of this research have been greatly influenced by their knowledge, persistence, and commitment to my academic development. I consider myself extremely fortunate to have such a great supervisor.

I also want to express my gratitude to my co-supervisor, Mr. Samantha Rajapaksha, whose advice, and ideas added a new level of knowledge to this dissertation. Their valuable feedback and willingness to devote time to my research significantly improved the quality of the work.

My fellow students have been incredibly helpful to me during this academic journey. Their willingness to participate in academic discussions, to share information, and to offer helpful feedback has been invaluable. I am appreciative of the collaborative atmosphere we developed, which promoted idea sharing and aided in the creation of my dissertation.

Finally, I would want to express my gratitude to my family and friends for their constant support and inspiration along this trip. During trying times, their confidence in me and comprehension of the requirements of academic study served as a source of inspiration.

I am deeply grateful to everyone who contributed to the completion of my dissertation. Thank you for your continuous support, and I look forward to building on the foundation established during this research in my future academic endeavors.

Thank You

Athukorala Y.J - IT20029968

Group Leader

ABSTRACT

This abstract introduces a mobile-based system for smart travel recommendations and tourism support, featuring a Tourist Assistant, Location Detection System, Personalized Service Recommendations, and Emotion-Based Tourist Activity Suggestion.

Tourist Assistant: Developed as a virtual guide for travelers in Sri Lanka, it provides important functions such as emergency assistance, location information, and responses to frequent questions via a chat-bot interface. It provides real-time assistance for visitors to confidently navigate unfamiliar environments.

Location Detection and Discovery System: This system uses mobile phone cameras to record visual data, delivering detailed location information to tourists to help them explore landmarks and sites of interest and make informed decisions about their trip plans.

Personalized Location and Service Recommendations: Analyzes user preferences and historical behaviors to give customized ideas for points of interest, dining alternatives, lodgings, and more, improving the overall trip experience.

Emotion-Based Tourist Activity Suggestion System: A unique feature that analyzes the user's emotional state using facial analysis techniques. It offers activities that are emotionally connected with the user's feelings, resulting in a genuinely rewarding and emotionally satisfying vacation experience that fosters a closer connection with the travel place.

By seamlessly integrating cutting-edge technologies, this system represents a substantial shift in the travel and tourist business. It provides travelers with tailored and emotionally resonant itineraries, transforming ordinary travel encounters into unforgettable and highly fulfilling adventures.

Key features: Machine learning, Sentiment Analysis, NLP, mobile application, Mobile Application, Tourism, Sri Lankan Tourism, Python, Flask

Table of Content

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Description
GDP	Gross Domestic Product
RS	Recommender System
ED	Emotion Detection
ER	Emotion Recognition
WTO	World Tourism Organization
UR	User Recognition
AI	Artificial Intelligence

1. Introduction

Tourism is a powerful economic development, cultural interchange, and individual enrichment motivator. Emotions have a significant impact on travelers' experiences and contentment as they embark on their adventures. Recognizing and resolving these feelings can significantly improve their participation, resulting in a more profound and unforgettable trip. It has the potential to influence how global travelers discover Sri Lanka by incorporating activity suggestions based on emotional experiences into a tourism assistance application. This country is known for its breathtaking scenery, rich cultural heritage, and friendly people, making it an excellent canvas for such transforming travel experiences.

Because of rapid technological improvements, the travel sector has undergone tremendous transformation. Digital tools and software have transformed the way travelers plan and experience their journeys, assisting them in a variety of facets of travel discovery. Tourism support applications have arisen as comprehensive guides, providing information on attractions, accommodations, transportation, and vital services. However, despite their capability, these programs frequently fall short of providing a tailored experience, failing to respond to individual tourists' particular emotional requirements and aims.

Recognizing the enormous potential for improving the tourist experience, we propose developing an emotion-driven activity suggestion system within the Sri Lankan tourism support application. This technology would use algorithms to examine the emotions of tourists gathered through various means, such as facial recognition. Based on this research, the computer might then propose activities that are suited to their emotional states.

The purpose of this research is to assess the viability, efficiency, and user acceptance of an emotion-driven activity recommendation system created for visitors visiting Sri Lanka. The goal is to deliver personalized suggestions that fit with travelers' interests, preferences, and emotional states, thus improving their overall experience. Whether visitors are looking for pleasure, adventure, cultural involvement, or regeneration, the system will provide emotionally relevant recommendations, delivering a uniquely personalized experience for everyone.

Moreover, the purpose of this study is to look at the possible influence of emotion-driven activity recommendations on the Sri Lankan tourism industry. Expected effects include increased tourist satisfaction, longer stays, favorable word-of-mouth recommendations, and, eventually, a significant boost to the country's tourism economy. We hope to contribute to Sri Lanka's goal of providing unique, original experiences, developing sustainable tourism practices, and positioning itself as a major worldwide holiday destination through forging emotional ties with guests.

In conclusion, incorporating emotion-driven activity recommendations into the tourism assistance application has the potential to impact passengers' opinions of Sri Lanka. The goal of this research is to improve tourist satisfaction, create meaningful emotional relationships, and promote the tourism industry's long-term success. This project aims to provide travelers with a genuinely enriched and emotionally resonant experience of Sri Lanka's extraordinary wonders by leveraging technology and appreciating the vital role of emotions in generating unforgettable journeys.

1.1. Background and Literature survey

Sri Lanka is a highly regarded tourism destination around the world, drawing visitors with its diverse offers, which include a rich cultural legacy, attractive natural landscapes, and a diverse range of species. Despite its attractiveness, tourists usually face challenges while planning excursions to this enthralling country. These issues primarily revolve around the difficulties connected with obtaining reliable and accurate information about various parts of their journey, such as sites to visit, available activities, lodging options, and dining establishments.

While existing travel recommendation systems are widely used by passengers, they have drawbacks that restrict their efficacy. These algorithms frequently provide generic suggestions, frequently relying on subjective reviews and ratings offered by users. This reliance on subjective opinions creates inconsistency and unreliability into the recommendations provided by these systems. As a result, travelers are in a conundrum while attempting to arrange their trip itineraries, uncertain of the accuracy or usefulness of the advice provided to them.

The reliance on subjective reviews and ratings frequently fails to appropriately respond to individual passengers' various preferences and interests. As a result, the suggestions supplied by these systems may not correspond to each tourist's individual desires or expectations. This circumstance leaves passengers with questions regarding the quality and compatibility of the recommended destinations, activities, lodgings, or dining options for their specific tastes.

The limits of present travel suggestion systems create a void in the tourism experience, making it difficult for travelers to arrange a thorough and rewarding itinerary that corresponds to their interests and expectations. The absence of individualized and dependable assistance might cause travelers to become frustrated and dissatisfied, limiting their capacity to fully enjoy and experience Sri Lanka's offers.

Despite Sri Lanka's great popularity as a tourist destination, the obstacles caused by insufficient and subjective travel suggestions lead to uncertainty and difficulties for tourists in efficiently arranging their travels. The reliance on generic recommendations based on subjective reviews and ratings has limitations, resulting in inconsistency and dependability in the advice supplied, consequently affecting the whole travel experience of travelers visiting this enthralling country.

Existing approaches to travel recommendation systems have numerous shortcomings, according to research in the field. Traditional systems are frequently impersonal, failing to take into consideration individual tastes and various interests of travelers. These systems rely heavily on user evaluations and ratings, which may not precisely reflect the preferences of every visitor. [1] Personalized recommendation systems powered by data mining and machine learning algorithms have the potential to greatly improve the travel experience by delivering tailored suggestions based on individual tastes and previous data, according to studies. [2]

Furthermore, advances in data mining techniques and machine learning algorithms have transformed the tourism sector by allowing the study of massive information to get insights into tourists' behavior, tastes, and travel patterns. To deliver more accurate and tailored

recommendations, techniques such as collaborative filtering, content-based recommendation, and hybrid recommendation systems have been widely researched in the context of tourism. [3]

While some systems provide real-time information about tourist sites and weather, very few include elements that address tourists' emotional states or provide emergency assistance. The incorporation of emotional state analysis into recommendation systems is a relatively new topic of research, with the goal of tailoring suggestions based on the user's present mood or emotional state. This feature, along with emergency assistance and location identification, offers an unprecedented leap forward in travel suggestion systems, providing a more comprehensive and personalized travel experience. [4]

By integrating cutting-edge data mining and machine learning techniques, the proposed Smart Travel Recommendation and Sri Lankan Tourism Support Mobile-Based System aims to fill these gaps in existing systems. The goal is to create a system tailored to Sri Lanka that provides personalized recommendations, real-time information, emotional state analysis, and emergency assistance, ultimately improving the overall tourism experience.

1.2 Research Gap

The modern landscape of travel recommendation systems is characterized by major technological breakthroughs, leveraging data mining, machine learning, and user-generated content to deliver assistance and ideas to passengers all over the world. However, a significant research gap persists within this dynamic domain, particularly concerning personalized and reliable travel recommendations tailored explicitly for tourists visiting destinations renowned for their cultural heritage, natural beauty, and wildlife, as exemplified by the difficulties encountered by travelers planning trips to Sri Lanka.

The existing work on travel recommendation systems focuses mostly on generalizing recommendations generated from user reviews and ratings, frequently missing the complexities and various tastes of passengers. Although research into personalized recommendation systems has made significant advances in understanding individual

preferences, these techniques frequently lack the context-specific elements required for destinations with varied attractions such as Sri Lanka. Current systems fall short of appropriately accounting for Sri Lanka's rich cultural history, diverse landscapes, and diverse animals.

Furthermore, while there have been rare research attempts into emotional state analysis in recommendation systems, the use of such techniques explicitly in the context of travel suggestions has been limited. Understanding and incorporating travelers' emotional states as a factor impacting their preferences and choices for locations, activities, and lodgings in a culturally lively and diverse country like Sri Lanka is an unexplored topic in the existing body of study.

Furthermore, while location-based services and real-time information are present in some travel recommendation systems, they frequently lack the depth required to adapt to the intricacies of Sri Lanka's tourism landscape. The integration of location recognition, emergency assistance, and real-time updates on tourist sites, transportation, and weather conditions inside a personalized recommendation system aimed at Sri Lanka remains an unaddressed feature in current research efforts.

In the existing literature, there are few research that specifically focus on establishing a complete, context-aware, and personalized travel recommendation system for places such as Sri Lanka. The lack of approaches and systems that combine cultural, geographical, and emotional aspects of travelers' experiences while addressing the obstacles that tourists have when organizing journeys to such different sites represents a significant research gap.

As a result, there is an important research gap in the creation of a smart travel recommendation system that is specifically suited to handle the complexities and subtleties of Sri Lanka's tourism sector. This void necessitates novel approaches that integrate cultural understanding, emotional state analysis, location-based services, and real-time information within a personalized recommendation framework, catering to the diverse preferences and needs of tourists exploring Sri Lanka's rich offerings.

Refer below *Tables* to identify the Research gap of this research.

	Availability (24/7)	Develop as a Mobile-based application	Focus emergency support	Interactive chatbot interface
Research Paper [2]	Not Mentioned	✓	×	✓
Research Paper [3]	Not Mentioned	✓	×	×
Research Paper [4]	Not Mentioned	×	×	×
Proposed System	✓	✓	✓	✓

Table 1. 1 Research Gap of Chatbot

	Availability (24/7)	Develop as a Mobile based application	Focus on personalized locations and services recommendation state	Interactive user interface
Research Paper [1]	Not Mentioned	✓	×	✓
Research Paper [2]	Not Mentioned	Not Mentioned	×	×
Research Paper [3]	Not Mentioned	✓	×	✓
Proposed System	✓	✓	✓	✓

Table 1. 2 Research Gap of Location and Service Recommendation System

	Accuracy	Develop as a Mobile based application	Recognize Location/places provide information about them	Interactive user interface
Research Paper [1]	Not Mentioned	×	✓	×
Research Paper [2]	Not Mentioned	×	×	✓
Research Paper [3]	Not Mentioned	✓	×	×
Proposed System	✓	✓	✓	✓

Table 1. 3 Research Gap of Location Detection and Discovery

	Availability (24/7)	Develop as a Mobile based application	Focus on a user's emotional state	Interactive user interface
Research Paper [3]	Not Mentioned	×	×	×
Research Paper [4]	Not Mentioned	×	×	×
Research Paper [7]	Not Mentioned	×	✓	×
Proposed System	✓	✓	✓	✓

Table 1. 4 Research Gap of Emotion Detection and Activity Suggestion

1.3 Research Problem

The research problem at hand is the creation of an innovative and contextually tailored Smart Travel Recommendation System that is specifically designed to address the challenges faced by tourists planning trips to Sri Lanka, a destination known for its diverse cultural heritage, breathtaking natural landscapes, and abundant wildlife. The primary goal is to fill existing gaps in travel recommendation systems by developing a complete and tailored solution that improves Sri Lanka's overall tourism experience.

Key Components of Research problem

1.3.1. Inadequacies in Existing Travel Recommendation Systems

Current travel recommendation systems are incapable of providing tailored and trustworthy recommendations to passengers visiting Sri Lanka. They mostly rely on general recommendations obtained from subjective reviews and ratings, which results in discrepancies and inconsistency in the advice provided.

1.3.2. The Complexities of Sri Lanka's Tourism Landscape

The multiple nature of Sri Lanka's attractions, such as its rich cultural legacy, different natural landscapes, and diversified fauna, presents significant obstacles for tourists arranging their itineraries. Existing methods fail to capture and address the complexities of these issues sufficiently in their suggestions.

1.3.3. Need for Personalized advice

Tourists visiting Sri Lanka want advice that are tailored to their specific preferences and interests. The lack of individualized assistance limits tourists' capacity to discover and enjoy Sri Lanka's entire range of services, resulting in subpar travel experiences.

1.3.4. Innovative Feature Integration

There is a research need in the integration of cutting-edge technologies such as emotional state analysis, location recognition, and real-time information provision within a trip recommendation system built specifically for Sri Lanka. These features can improve the relevance, accuracy, and completeness of trip recommendations greatly.

1.3.5. Enhancement of Tourism Experience and Industry Support

Addressing the challenges through the development of an advanced Smart Travel Recommendation System has the potential not only to improve tourists' satisfaction and enjoyment, but also to support the growth and development of Sri Lanka's tourism industry.

1.4. Research Objectives

The comprehensive research objectives are focused at developing, improving, and validating a cutting-edge Smart Travel Recommendation System designed exclusively for travelers visiting Sri Lanka. Each research objective is explained in detail below.

Develop a Smart Travel Recommendation System focuses on the development of a sophisticated system that makes use of cutting-edge technology such as data mining, machine learning, and sophisticated algorithms. The system will be built to evaluate large amounts of data, including tourist preferences, historical travel data, and real-time location information. The goal is to create a comprehensive system capable of delivering personalized suggestions that cater to individual visitors' interests, preferences, and historical travel habits by leveraging various technologies. Patterns in preferences and historical data will be uncovered using data mining techniques, laying the groundwork for the recommendation engine.

Integrate Novel Features involves integrating new and contextually relevant features into the Smart Travel Recommendation System. The addition of emotional state analysis, in particular, will allow the system to assess visitors' emotional states during their vacation experiences. Incorporating location recognition capabilities will also allow the system to detect and deliver important information about the captured locations. Furthermore, the addition of real-time information about tourist attractions, transit alternatives, weather conditions, and other relevant factors would increase the system's relevance and usability for tourists visiting Sri Lanka. These innovative elements will enhance the system's ability to deliver comprehensive and dynamic recommendations that go beyond standard recommendations.

Evaluate Effectiveness and Usability requires conducting extensive user studies and real-world testing to analyze the efficacy, dependability, and usability of the produced Smart Travel Recommendation System. The purpose is to confirm the system's capacity to give tailored, accurate, and reliable travel recommendations using rigorous evaluation approaches such as user surveys, user experience assessments, and real-world testing scenarios. The evaluation method will collect user feedback, allowing for iterative changes based on user

insights and observations. This goal aims to ensure that the developed system is user-centric, dependable, and capable of providing comprehensive and tailored travel recommendations for tourists exploring Sri Lanka by validating the system's performance in real-world settings and measuring its effectiveness in meeting tourists' needs.

In summary, these study aims aim to create an original Smart Travel Recommendation System that is equipped with modern technology, novel features, and robust evaluation processes. This complete method aims to change the travel advice experience for tourists visiting Sri Lanka, thereby improving their satisfaction, convenience, and discovery of the country's numerous attractions.

1.5. Significance of the Study

The study's significance in developing a Smart Travel Recommendation System suited for travelers visiting Sri Lanka is multidimensional, with numerous essential features contributing to its significance:

1.5.1. Enhancing tourist experience

The implementation of a sophisticated travel suggestion system solves the difficulties that tourists have when arranging their vacations to Sri Lanka. The system improves the overall tourism experience by offering personalized and reliable recommendations based on tourists' interests and historical data. It allows visitors to explore more successfully the numerous cultural, natural, and wildlife features, resulting in increased satisfaction and enjoyment throughout their travels.

1.5.2. Addressing Existing System Limitations

The study addresses current travel recommendation system deficiencies by providing personalized suggestions derived from advanced data mining, machine learning algorithms, and innovative features such as emotional state analysis and location recognition. This approach intends to provide travelers with more precise and specialized information by addressing the constraints of generic and subjective recommendations, thereby filling a vacuum in trip planning services.

1.5.3. Promotion of the Tourism sector in Sri Lanka

A well-designed and functional Smart Travel Recommendation System can have a favorable impact on the tourism sector in Sri Lanka. The system has the potential to attract more visitors to the country by making travel more fun and convenient for tourists. Increased tourism can result in economic growth, job creation, and general development in tourism-related industries, all of which contribute considerably to the country's economy.

1.5.4. Technological Advancement and Innovation

The research entails incorporating cutting-edge technologies such as data mining, machine learning, and real-time information providing into the travel recommendation system. This emphasis on technical innovation not only allows the construction of a more sophisticated and intelligent system, but it also adds to the advancement of research and development in tourism technology and recommendation systems.

1.5.5. Validation and User-Centered Design

The evaluation component of the study verifies the system's effectiveness, usability, and reliability through user studies and real-world testing. Obtaining feedback from actual users enables incremental improvements, ensuring that the system is user-centric and meets the needs and preferences of travelers. This validation step is critical in ensuring that the established system achieves its intended goal of providing personalized, dependable, and comprehensive travel suggestions.

In summary, the study's significance arises from its ability to transform the tourism experience for travelers to Sri Lanka by solving existing constraints in travel recommendation systems. The study aims to improve travelers' experiences, contribute to the expansion of Sri Lanka's tourism industry, and stimulate technological innovation in the field of travel recommendation systems by developing an advanced and contextually relevant Smart Travel Recommendation System.

1.6. Scope and Limitations

An explanation of the study's scope and limitations for the development of a Smart Travel Recommendation System for visitors visiting Sri Lanka is include following areas:

1.6.1. Technological Scope

Utilization of cutting-edge technologies: The study employs data mining, machine learning, and advanced algorithms to examine tourist preferences and historical data in order to provide individualized recommendations.

Integration of novel features: This includes incorporating novel functionality within the recommendation system such as emotional state analysis, location identification, and real-time information providing.

1.6.2. Functional Scope

Comprehensive recommendation system: The study's goal is to develop a comprehensive system that suggests Sri Lanka-specific destinations, activities, lodgings, restaurants, and real-time information on tourist sites, transportation, and weather conditions.

User-centric approach: The system's development is centered on offering personalized, dependable, and contextually relevant recommendations tailored to individual travelers' interests and needs.

1.6.3. Evaluation Scope

User studies and real-world testing: The study includes detailed evaluations of the system's efficacy, usability, and reliability through user studies and real-world testing.

Validation of recommendations: The evaluation procedure confirms the system's ability to deliver accurate, tailored, and comprehensive travel recommendations to tourists visiting Sri Lanka.

1.6.4. Data Availability and Quality

Availability of comprehensive datasets: Access to vast and high-quality datasets comprising varied tourist tastes and historical data may be difficult to come by.

Data correctness and reliability: The accuracy and reliability of data sources utilized for analysis and suggestions may have an impact on the system's performance.

1.6.5. Constraints of Technology

Technical complexities: Implementation issues connected to integrating complicated technologies into the system, such as emotional state analysis and real-time information providing, may develop.

Computational resources: The need for extensive computational resources and infrastructure to handle and analyze massive datasets may impose constraints.

1.6.6. Generalization vs. Specificity

Balancing general advice with specificity: It may be difficult to strike a balance between providing basic recommendations that appeal to a wide spectrum of tourists and making precise, individualized suggestions.

1.6.7. User Acceptance and Adaptation

Adoption and adaptation of users: Ensuring that users adapt to and accept the system's recommendations may be dependent on user familiarity with technology, user interface design, and acceptance of algorithm-driven suggestions.

1.6.8. Contextual and Cultural Understanding

Cultural and contextual nuances: It is possible that the system's ability to grasp and effectively include varied cultural features and local nuances in recommendations is limited.

In summary, the scope of the project encompasses a wide variety of technological, functional, and evaluative factors with the goal of developing a comprehensive Smart Travel Recommendation System for travelers visiting Sri Lanka. However, various factors such as data availability, technological limits, user acceptance, and cultural understanding may have an impact on the system's creation and efficacy.

2. Methodology

2.1 Requirement Gathering and Analysis

Gathering and analyzing requirements in the context of constructing a Smart Travel Recommendation and Tourism Support System entails a systematic method of defining, recording, and comprehending the needs, preferences, and functions required for the system's effective creation.

2.6.1. Requirement Gathering

2.6.1.1. Stakeholders Engagement

Engagement with stakeholders, such as travelers, tourism industry professionals, and technology users, to better understand their expectations, preferences, and issues in trip planning.

2.1.1.2. User Interviews and Surveys

Interviews and surveys with potential users to acquire information about their interests, travel behaviors, and expectations from a travel recommendation system.

2.1.1.3. Technological and Functional Inputs

Collaboration with technical experts and professionals to define the technical needs and functionality required for integrating features such as emotion analysis, user recognition, and personalized suggestions.

2.1.1.4. Review of Existing System

Analyzing existing travel recommendation systems, mobile applications, and technology implementations to find beneficial aspects, flaws, and places for development.

2.1.2. Requirement Analysis

2.1.2.1. Requirement Prioritization

Sorting requirements based on their importance, feasibility, and impact on the system's functioning and user experience.

2.1.2.2. Defining Functional Specifications

Transforming the requirements gathered into system-specific functionalities and features such as emotion analysis algorithms, user recognition systems, and tailored recommendation engines.

2.1.2.3. Creating use cases and scenarios

Creating use cases and scenarios to demonstrate how users would engage with the system while considering various emotions, user identities, and personalized suggestion situations.

2.1.2.4. Validation and Refinement

Reviewing and validating requirements with stakeholders, ensuring alignment with project goals, and modifying requirements based on feedback and feasibility concerns.

2.1.2.5. Documentation

Detailing acquired requirements, such as functional specifications, use cases, and priority criteria, for usage as a reference during the system design and development phases.

The requirement collection and analysis phase serve as the foundation for the system design, development, and implementation stages that follow. It guarantees that the Smart Travel Recommendation and Tourism Support System addresses users' defined needs, incorporates novel features such as emotion analysis and user recognition, and delivers personalized suggestions tailored to travelers experiencing Sri Lanka.

2.2. Feasibility Studies

In the context of establishing a Smart Travel Recommendation and Tourism Support System, feasibility studies involve evaluating the project's viability, practicability, and possible success. Before devoting resources to the development process, these studies try to analyze numerous factors.

2.2.1. Technical Feasibility

2.2.1.1. Technological Infrastructure

Evaluating the availability and sufficiency of technological resources needed to implement features such as data mining, machine learning algorithms, emotion analysis, and facial recognition.

2.2.1.2. Software and Tools

Evaluating the availability and compatibility of system development software, tools, and frameworks.

2.2.2. Financial Feasibility

2.2.2.1. Budget analysis

Budget analysis is the process of doing a cost-benefit analysis to determine the financial resources needed for system development, such as software, hardware, personnel, and maintenance costs.

2.2.2.2. Return on Investment (ROI)

Calculating the potential returns and benefits of a system to justify the investment in its development.

2.2.3. Operational Feasibility

2.2.3.1. User Acceptance

The readiness of users, particularly tourists and tourism sector stakeholders, to embrace and use the system. Conducting surveys or interviews to measure the attitudes and expectations of potential users.

2.2.3.2. Ease of Implementation

Assessing the feasibility and convenience of incorporating advanced features such as emotion analysis and user recognition into the system.

2.2.4. Legal and Ethical Feasibility

2.2.4.1. Data Privacy and Security

Ensuring compliance with legal rules governing data privacy and security, particularly with regard to facial recognition data and personal information gathered from individuals.

2.2.4.2. Ethical Consideration

Evaluating the ethical implications of using technology such as emotion analysis and user recognition, as well as ensuring that the system respects user privacy and autonomy.

2.2.5. Time Feasibility

2.2.5.1. Project Timeline

Considering the projected timetable for creation, testing, and deployment of the project. Assuring that the project can be completed in a fair amount of time.

2.2.6. Risk Analysis

2.2.6.1. Identifying Potential Risks

Identifying and assessing potential risks and problems that may impede the system's development and implementation.

2.2.6.2. Mitigation Strategies

Creating ways to mitigate recognized risks and problems in order to reduce the project's impact.

Feasibility studies are critical in assessing whether the Smart Travel Recommendation and Tourism Support System is both practically and commercially viable. These studies assist project stakeholders in making educated decisions about moving forward with development by taking into account technical, financial, operational, legal, ethical, and time-related factors, as well as identifying potential risks and mitigation solutions.

2.3. System Diagrams

2.3.1. Overall System Diagram

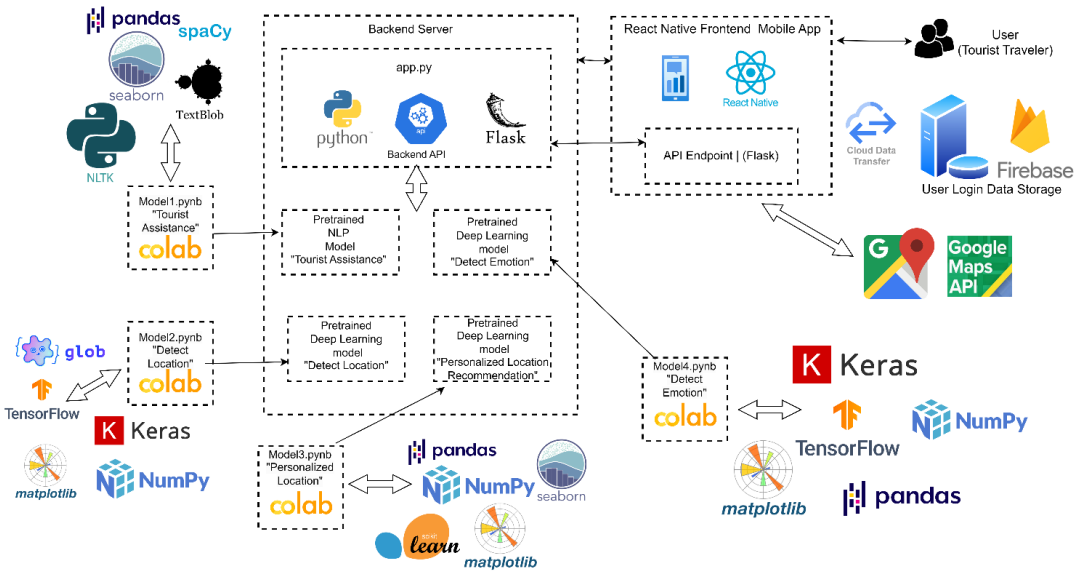


Figure 2. 1 Overall System Diagram

2.3.2. Individual Component Diagram

2.3.2.1. Tourist Assistant

The Tourist Assistance system in Sri Lanka is a mobile app-based solution that provides tourists with real-time assistance during their visit. It uses Natural Language Processing (NLP) and Artificial Intelligence (AI) to provide emergency assistance, geographical information, and answers to common questions. The technology, which includes a chatbot interface, a database, and an NLP engine, allows direct contact between tourists and the app, providing instant access to information. The database contains emergency contacts, locations, and FAQs, and the NLP engine interprets inquiries to provide accurate responses. The system, which emphasizes user-friendliness, efficiency, and accuracy, intends to improve tourists' experiences by eliminating misinformation risks and providing useful information.

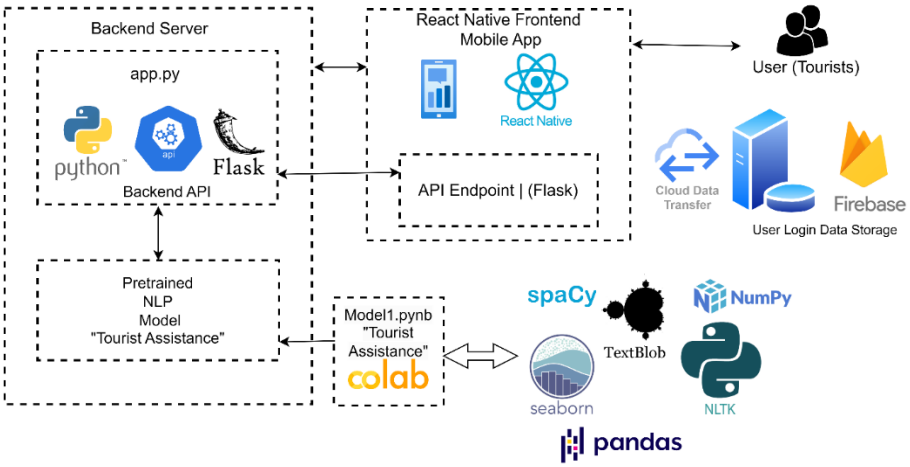


Figure 2. 2 Tourists Assistant System Diagram

2.3.2.2. Location Detection and Discovery

Through unique features, the Smart Travel Recommendation and Tourism Support Mobile-Based System in Sri Lanka improves travelers' travel experiences. It does not rely on GPS, but rather on image capture and analysis for position recognition, combining a massive database with historical and cultural data. The system uses the ResNet 50 deep learning model for efficient image analysis, provides real-time information presentation, a user-friendly interface, and connections to external services such as travel advisors and social networking platforms. It emphasizes individualized recommendations based on user choices, with the goal of providing a personalized and engaging travel experience. This system, which prioritizes accuracy, convenience of use, and personalization, intends to elevate exploration of Sri Lanka's cultural and historical monuments, thereby contributing to the growth and sustainability of the country's tourism sector.

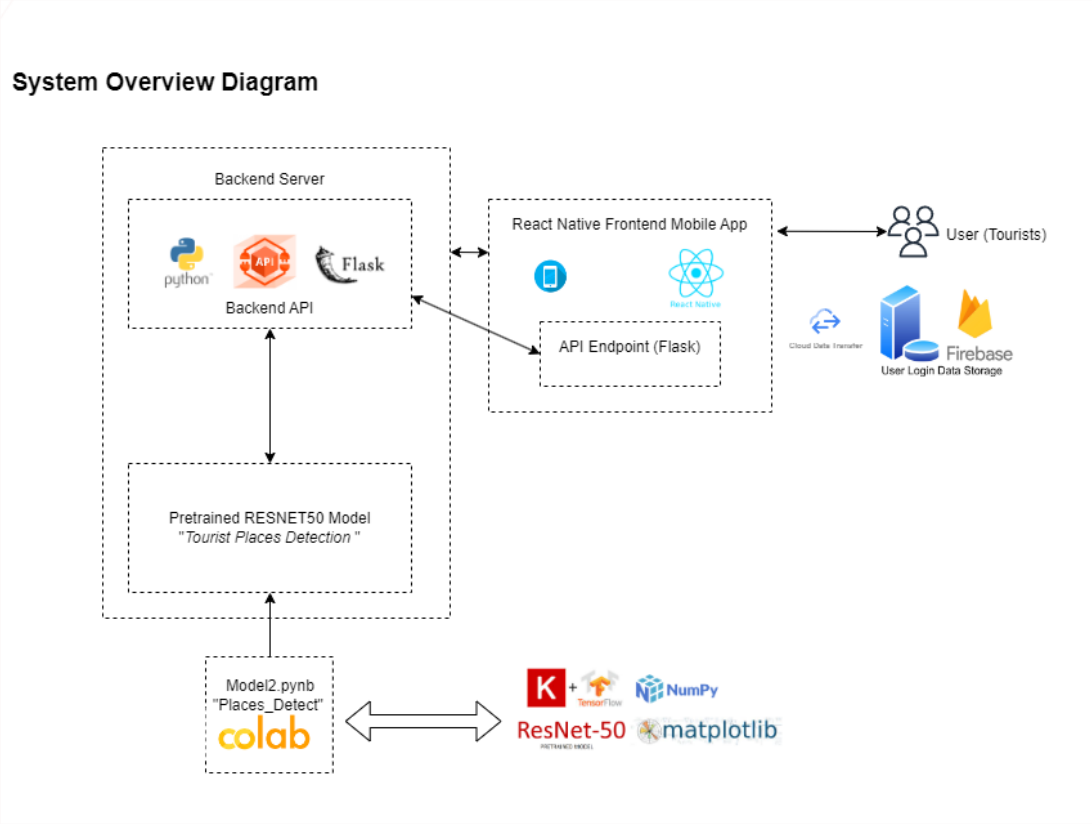


Figure 2. 3 Location Detection and Discovery System Diagram

2.3.2.3. Service Recommendation System

The creation of our cutting-edge machine learning-driven recommendation app for personalized location-based services is a big step forward in the revolutionization of the travel experience. This approach is intended to accommodate to everyone's specific travel tastes, providing proactive and specialized recommendations that cover many parts of their journey.

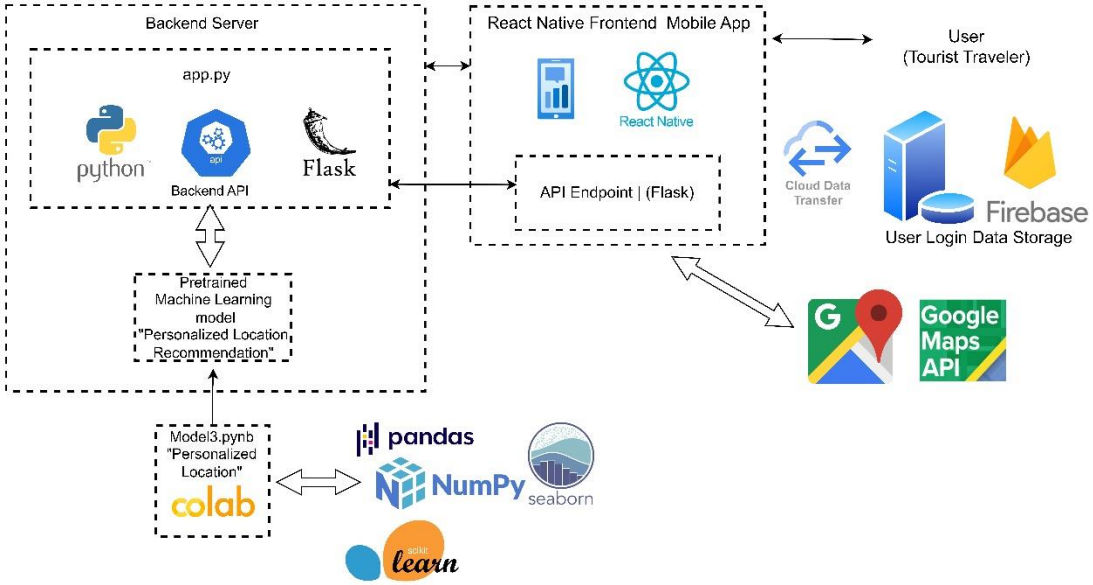


Figure 2. 4 Service Recommendation System Diagram

2.3.2.4. Emotion Based Activity Suggestion System

The mobile application, according to the system overview diagram, is the system's user-facing component, in which users utilize a camera to capture their faces and facial recognition and emotion analysis technology to collect emotion. This information is then sent into the AI engine, which incorporates facial recognition and emotion analysis algorithms. Based on the user's facial traits and emotions, the AI engine analyzes data and proposes unique trip possibilities. These suggestions are sent to the recommendation engine, which then suggests more travel alternatives depending on the user's interests. The user can select a trip mode as needed. Overall, this technology employs facial recognition and artificial intelligence to deliver personalized travel recommendations and tourist assistance, thereby improving the user's Sri Lanka vacation experience.

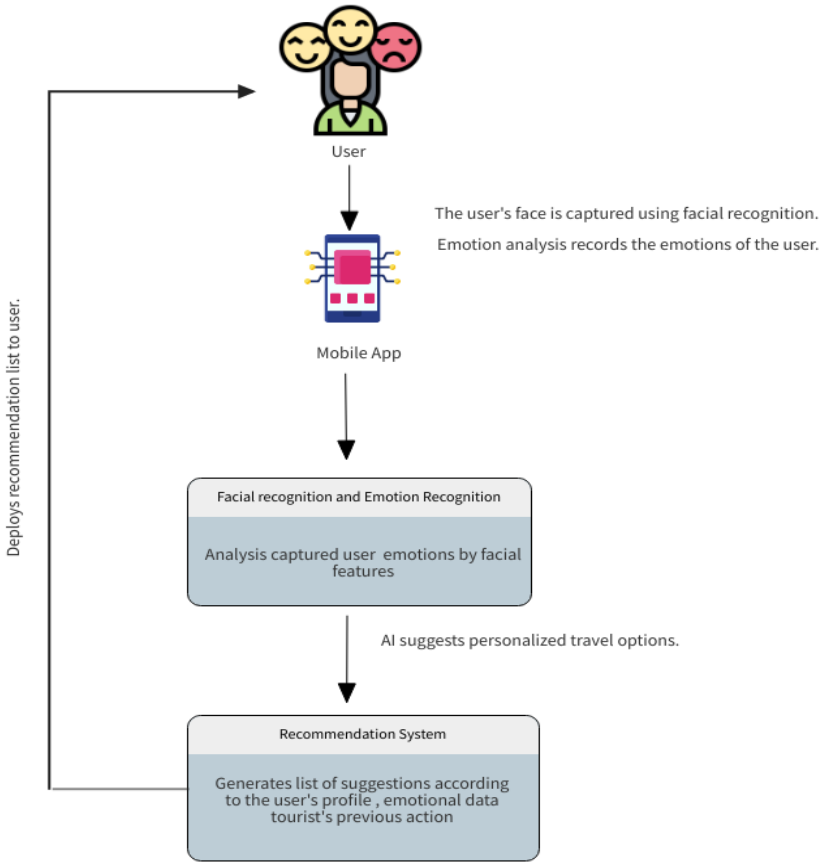


Figure 2. 5 Emotion based Activity Suggestion System Diagram

2.3.3. Individual Component Flow Charts

2.3.3.1. Tourists Assistant

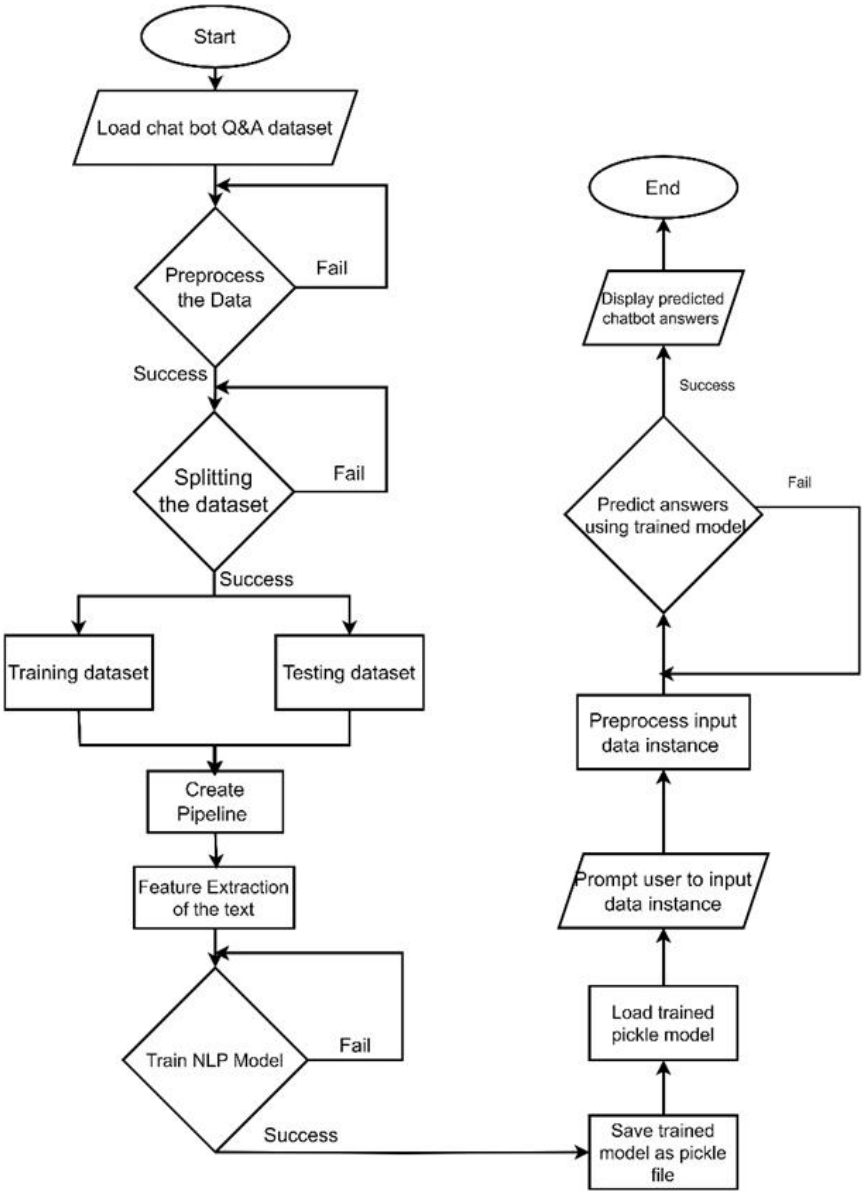


Figure 2. 6 Flow chart of Tourists Assistance

2.3.3.2. Location Detection and Discovery

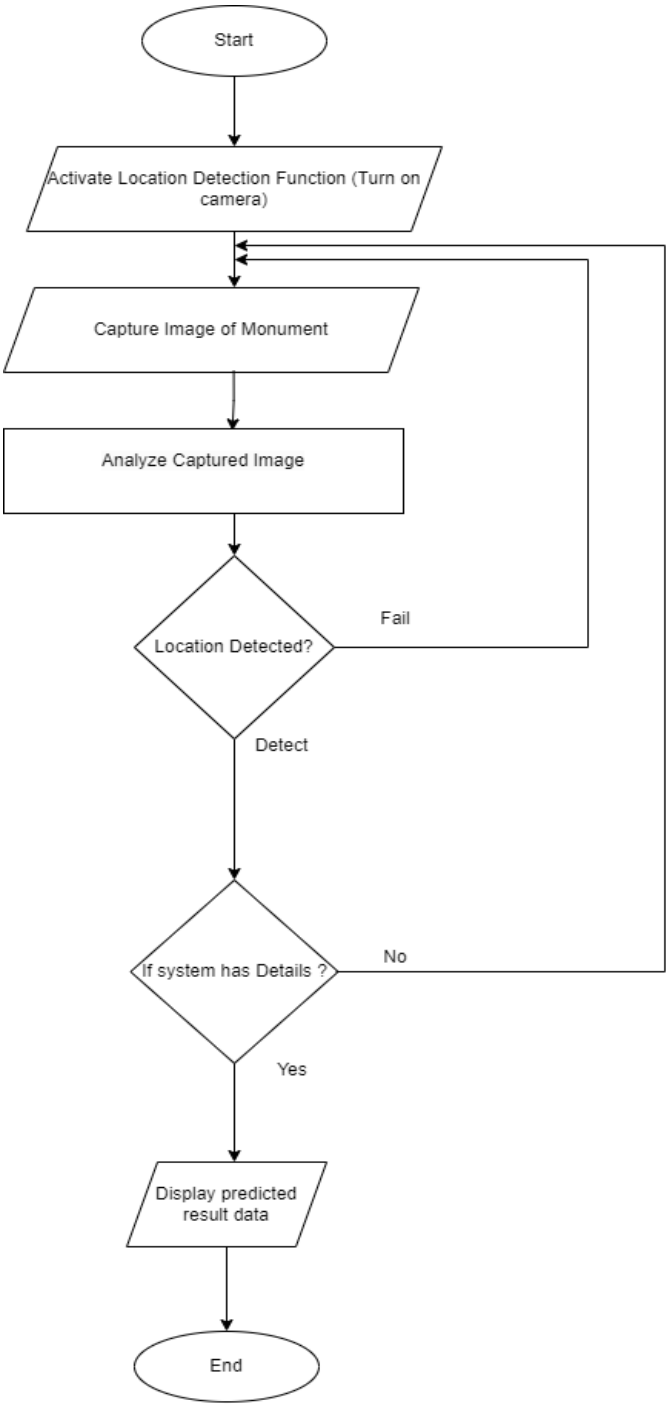


Figure 2. 7 Flow chart of Location Detection and Discovery

2.3.3.3. Service Recommendation System

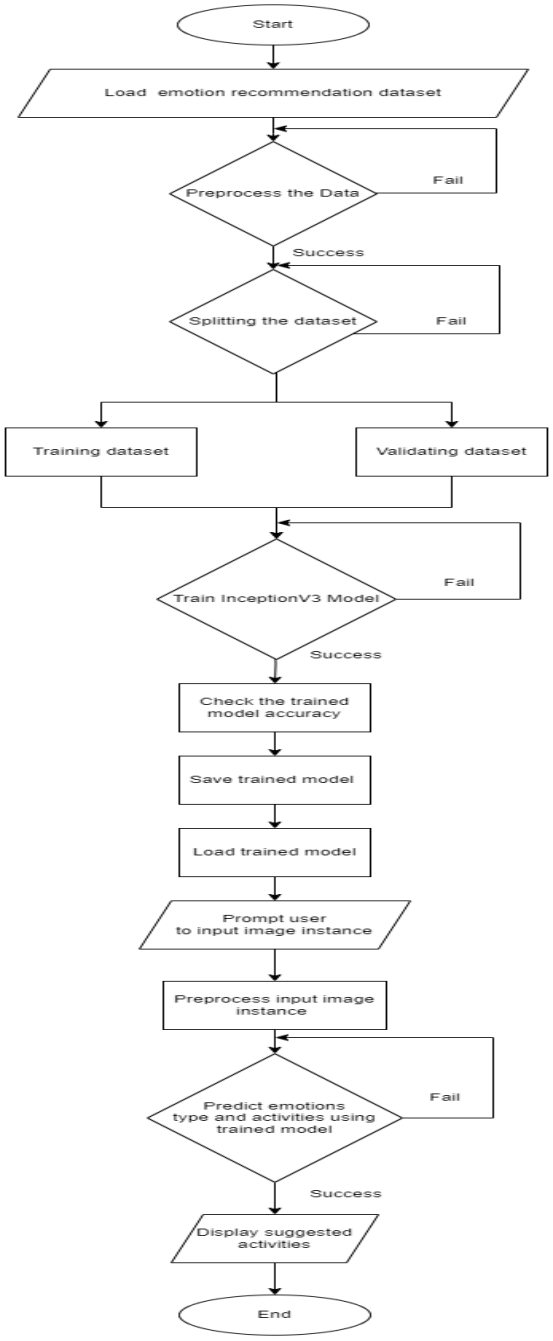


Figure 2. 8 Flow chart of Service Recommendation System

2.3.3.4. Emotion based Activity Suggestion System

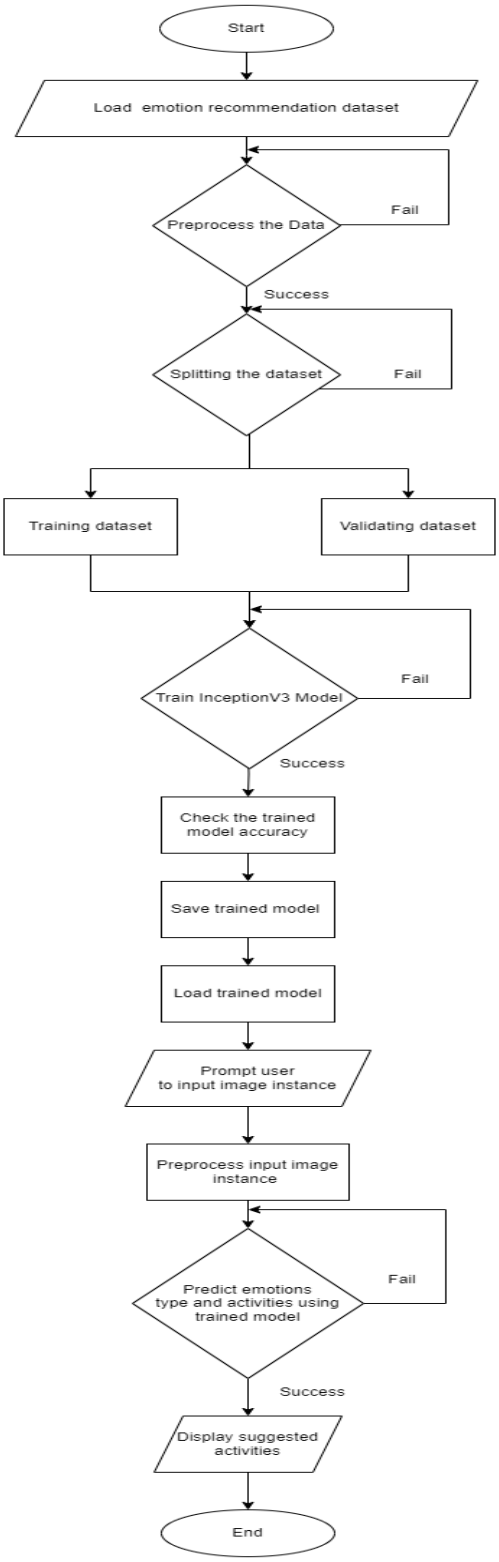


Figure 2. 9 Flow chart of Emotion based Activity Suggestion System

2.4. Understanding the Key Pillars of the Research Domain

Understanding the essential pillars of the study area in the context of establishing a Smart Travel Recommendation and Tourism Support System for travelers visiting Sri Lanka entails grasping the fundamental aspects that form the foundation of the project. These pillars are key factors that define and influence the domain's research. Here is how it works:

2.4.1. Tourism Industry Insights

2.4.1.1. Cultural and Natural Heritage

This component focuses understanding Sri Lanka's rich cultural heritage and natural settings. It includes historical sites, natural wonders, and unique attractions that make the country an enticing tourist destination. This insight emphasizes the richness and variety of experiences available, highlighting Sri Lanka's historical significance, natural beauty, and unique attractions that attract visitors.

2.4.1.2. Tourist Behavior and Preference

This entails researching the habits, tastes, and motives of tourists who visit Sri Lanka. It aims to discover what variables impact their decisions to visit Sri Lanka, as well as their experiences there. This investigation delves into the motivations for travelers' decisions, their interests, and what effects their overall travel experiences. Understanding these elements is critical for tourism stakeholders because it allows them to adjust services and offerings to better suit and attract tourists, hence improving their experiences and happiness.

2.4.2. Tourism Industry Insights

2.4.2.1. Data mining and Machine Learning

This entails using data mining techniques and machine learning algorithms to examine various sorts of tourist data. Preferences, historical data (past travel patterns, popular destinations, etc.), and real-time location data are all included. These techniques are utilized to extract significant insights from data, allowing personalized suggestions for tourists to be created. This technique tries to offer targeted ideas and improve overall travel experiences by studying travelers' interests and behaviors.

2.4.2.2. Advanced Features Integration

This element is concerned with introducing cutting-edge technical aspects into travel recommendation systems. Emotion analysis (understanding tourists' feelings through their interactions or expressions), facial recognition (for security or personalized interactions), and real-time information provision (delivering up-to-date information quickly) are among these aspects. The travel recommendation system intends to improve user experiences, give additional layers of protection, and provide more tailored and immediate help to visitors during their travels by including such advanced functions.

2.4.3. User Experience and Personalization

2.4.3.1. Enhancing User Experience

This element focuses on providing tourists with a smooth, intuitive, and delightful travel experience. It entails creating a user-friendly interface for travel applications or platforms. The goal is to streamline the travel planning process by making it easier for tourists to explore, obtain information, and use available services. A smooth interface helps greatly to a positive overall experience by allowing tourists to interact with the system or application without difficulty during their journey.

2.4.3.2. Personalized Recommendation

This refers to tailoring recommendations to travelers based on their personal interests, travel history, and emotional states. Personalized suggestions are generated for each user based on data analysis, machine learning, or human interaction. These recommendations are one-of-a-kind and tailored to the user's interests, previous travel experiences, and possibly current emotions or preferences. The goal is to provide recommendations that are more relevant and appealing to individual travelers, hence increasing their overall satisfaction with their travel experience.

2.4.4. Ethical and Privacy Consideration

2.4.4.1. Data Privacy and Security

This focuses on upholding ethical norms in the collecting, storage, and usage of user data in tourist technology. It underlines the need of protecting sensitive data, particularly data gathered through technology such as facial recognition and emotional analysis. It is critical

to preserve users' privacy to ensure that user data is acquired and maintained securely, and that its use follows to ethical rules and regulatory standards.

2.4.4.2. Ethical use of Technology

This component entails considering the ethical implications of various technologies used in the tourism business, such as emotion analysis and user recognition. Transparency in how these technologies is used is required, as is ensuring that consumers are informed about the data collected, its purpose, and the ability to opt-in or opt-out of such services. It is critical to respect user privacy and seek consent before using technology that may collect sensitive information. Using these technology breakthroughs in the tourism business while adhering to ethical norms ensures that consumers' rights and privacy are protected.

2.4.5. Industry Growth and Economic Impact

2.4.5.1. Contribution to Tourism Industry

The Smart Travel Recommendation System seeks to have a positive impact on the Sri Lankan tourism industry. It accomplishes this by attracting more tourists through tailored and engaging travel experiences. The system contributes to the development and expansion of Sri Lanka's tourism industry by providing specialized recommendations and improving travelers' overall experiences throughout their visit. The ability of this system to attract more visitors and provide them with unique and unforgettable experiences has the potential to improve tourism traffic and interest in the country.

2.4.5.2. Economic Implications

The Smart Travel Recommendation System's adoption and effectiveness can result in a variety of economic benefits for Sri Lanka. These include increased tourist revenue because of a greater influx of visitors enticed by personalized and optimized travel experiences. Furthermore, a better tourism sector can contribute to employment development in a variety of tourism-related industries, such as hotels, transportation, and local businesses catering to tourists. Overall, the system's ability to improve tourism experiences can benefit the country's economy, promoting growth and development.

2.4.6. Evaluation and Validation

2.4.6.1. Effectiveness Assessment

This examines the system's performance, usability, and dependability. Conducting user surveys, real-world testing, and developing feedback channels are examples of methodologies. User studies entail monitoring and getting feedback from people who use the system to determine how well it fits their needs. Real-world testing entails putting the system through its paces in real-world circumstances to evaluate its performance in real time. To acquire insights, feedback techniques like as surveys, reviews, or direct user feedback channels can be used. These evaluations aid in identifying the system's strengths, flaws, and areas for growth.

2.4.6.2. Continuous Improvement

Putting in place tools for continuous review and feedback guarantees that the system evolves and improves over time. The system may continuously adapt and increase its functionality by evaluating user interactions, incorporating user feedback, and remaining responsive to changing user needs and technology advancements. This iterative process allows the system to fix any faults that have been detected and to introduce new features or upgrades, resulting in a more refined and user-centric Smart Travel Recommendation System.

Understanding these essential research domain pillars provides a comprehensive viewpoint for designing the Smart Travel Recommendation and Tourism Support System. It combines tourism sector insights, technical improvements, user-centric approaches, ethical considerations, economic implications, and assessment methodologies to produce a comprehensive and successful solution for travelers visiting Sri Lanka.

2.5. Research Methods

The research methods employed in this project are mostly **quantitative**, with elements of **mixed methods** used to address specific aspects of the inquiry.

2.5.1. Quantitative method

Because the primary purpose is to create a deep learning model for facial emotion recognition and location detection, quantitative methods are generally employed. Quantitative techniques are particularly well adapted to tasks that necessitate numerical analysis and performance evaluation. Quantitative indicators such as accuracy, loss, and validation scores are crucial for assessing the model's performance in image categorization tasks.

2.5.2. Mixed method

Mixed methods are used to handle the qualitative components of the research, with a particular emphasis on the issues encountered during data collection and processing. This mixed methods strategy allows for the collection and interpretation of qualitative data related to research difficulties.

2.6. Data Collection and Data Analysis Methods

2.6.1. Data Collection

Data gathering for the development of a Smart Travel Recommendation and Tourism Support System for tourists visiting Sri Lanka entails acquiring multiple datasets spanning tourist preferences, historical travel data, real-time information, and inputs for emotion analysis and user recognition. It entails capturing tourists' preferences, habits, and interests via surveys, feedback forms, and interactions with the system to better understand their travel patterns and decisions. Historical data on previous travel itineraries, visited sites, activities, lodging, and user evaluations aid in extracting insights into visitor preferences and experiences. The system may give up-to-date information by obtaining real-time data on current tourist attractions, transit schedules, weather conditions, and activities taking place in Sri Lanka. Furthermore, capturing facial photos or video data for emotion analysis and

user recognition, as well as ensuring compliance with privacy standards and gaining user authorization, helps to improve personalized suggestions and user identification within the system.

Figure 2. 10 Meeting Tourist

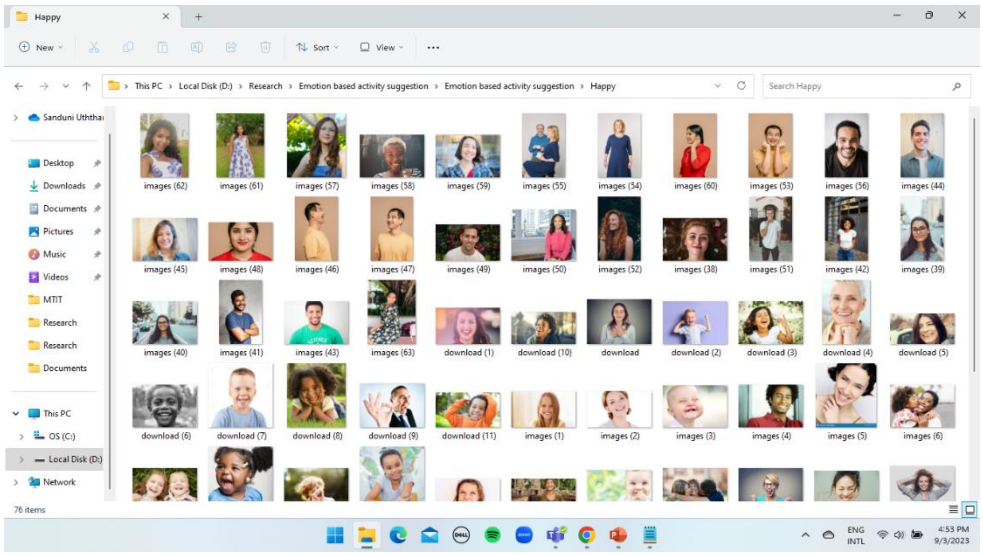


Figure 2. 11 Data Collection

2.6.2. Data Analysis

For data analysis, both quantitative and qualitative methodologies are used in the given study process:

- Convolutional Neural Network (CNN) Model Construction: This entails building a CNN model, which is a quantitative technique used to classify images. CNNs are deep learning models that are specifically built to evaluate visual input such as photos.
- Quantitative metrics for Training: During the CNN model's training phase, quantitative metrics such as loss functions and optimization methodologies are used. Loss functions quantify the difference between expected and actual values, assisting the model in optimizing its parameters.
- Quantitative Evaluation Criteria: A variety of quantitative evaluation criteria are employed to measure the performance of the CNN model. These metrics include test accuracy, validation loss (how effectively the model generalizes to new data), and precision-recall scores (how accurate the model is in identifying positive cases versus false positives).
- Qualitative Data Evaluation: Qualitative data is specific to difficulties found throughout the investigation. Methods such as content analysis and thematic analysis are used to examine this qualitative data. These strategies entail carefully evaluating and interpreting qualitative data to find common themes, patterns, or concerns that may not be easily measurable but are critical for understanding the study's context or nuances.

To discover common themes or patterns, qualitative data, specifically linked to issues encountered, is evaluated using content analysis or thematic analysis.

2.7. Commercialization Plan

A commercialization strategy for the Smart Travel Recommendation and Tourism Support System designed for travelers visiting Sri Lanka includes tactics for bringing the system to market, increasing user adoption, and generating income. The following is an outline of a commercialization strategy for this scenario:

2.7.1. Market Analysis

Conduct a detailed examination of the Sri Lankan tourism market, identifying important players, market trends, and consumer preferences. Recognize the market for unique travel suggestion systems as well as the competitive terrain.

2.7.2. Product Positioning and Unique Selling Proposition (USP)

Define the system's distinguishing features, such as emotion analysis, personalized suggestions, and real-time information delivery, stressing how they distinguish the system from competing solutions. Position the system as a complete and user-centric platform that improves the travel experience through personalized recommendations and assistance.

2.7.3. Go-to-Market Strategy

Create a strategy for introducing the system to the market. To acquire initial traction and publicity, agreements with tourism organizations, hotels, travel operators, or government bodies in Sri Lanka may be formed. Use digital marketing, social media, and targeted advertising campaigns to reach out to potential users, both those planning to visit Sri Lanka and those who are already there.

2.7.4. Pricing Model

Determine a price strategy based on the value proposition of the system, its features, and the intended user segment. Consider varied levels of access to system functionalities through multiple subscription tiers or freemium models. Examine competitive pricing while ensuring that the solution is affordable and valuable to users.

2.7.5. User Acquisition and Engagement

To encourage tourists to use the system, focus on user acquisition through promotional offers, referral schemes, or early adopter incentives. To retain consumers and encourage

regular usage, implement engagement tactics such as tailored notifications, user feedback mechanisms, and loyalty programs.

2.7.6. Partnerships and Collaborations

Collaborate with Sri Lankan tourism boards, hotels, restaurants, and other key stakeholders to improve the system's services and provide consumers with exclusive material or discounts. Investigate collaborations with technology vendors for continual system enhancement and the incorporation of new features.

2.7.7. Continuous Improvement and Support

Create a dedicated support team to respond to user inquiries, feedback, and technical difficulties as soon as possible. Plan for ongoing system changes and improvements in response to user input, technological breakthroughs, and shifting travel trends.

2.7.8. Revenue Streams

Subscription-based models, premium features, adverts, or agreements with businesses for sponsored content or recommendations within the system can all be used to generate cash.

2.7.9. Monitoring and Evaluation

Monitor key performance indicators (KPIs) such user engagement, retention rates, revenue streams, and user happiness on a regular basis to evaluate system performance and adapt strategies as needed.

This commercialization strategy seeks to effectively launch the Smart Travel Recommendation and Tourism Support System into the market, attract users, generate income, and ensure ongoing improvement to fulfill user demands and preferences in the context of tourism in Sri Lanka.

2.8. Consideration of the Aspect of the System

Examining essential factors that contribute to the system's functioning, usability, and effectiveness while considering various parts of the Smart Travel Recommendation and Tourism Support System for travelers visiting Sri Lanka. Here are some crucial points to consider:

2.8.1. User-Centric Designs

Assure that the system provides an intuitive user interface and a smooth user experience, making it simple for tourists to navigate, obtain information, and receive recommendations.

2.8.2. Personalization and Customization

Implement strong algorithms that assess travelers' interests, travel history, and real-time location data to provide personalized recommendations for Sri Lankan destinations, activities, accommodations, and dining alternatives.

2.8.3. Data Privacy and Security

Incorporate strong safeguards to protect user data, particularly when using facial recognition and emotion analysis, to ensure compliance with data privacy regulations and ethical issues. Obtain explicit user consent before collecting and processing data.

2.8.4. Real Time Information Provision

To improve travelers' travel experiences, integrate mechanisms that provide up-to-date information on tourist destinations, transportation timetables, weather conditions, and local activities.

2.8.5. Emotion Analysis and User Recognition

Create precise emotion analysis and user recognition algorithms to ensure precise identification of user emotions and reliable user authentication for personalized suggestions and user-specific recommendations.

2.8.6. Scalability and Performance

Create a system design that can handle increased user demand and data volumes while maintaining scalability and stable performance during peak tourist seasons.

2.8.7. Continuous Improvement and Adaptability

Plan for ongoing system updates, improvements, and adjustments based on user feedback, technological breakthroughs, and changing tourist preferences. Ensure tourist industry flexibility to changing trends.

2.8.8. Ethical Use of Technology

Examine the ethical implications and potential biases of technology such as emotion analysis and facial recognition. In system functionalities, prioritize user privacy, consent, and transparency.

2.8.9. Collaboration and Partnerships

Explore interaction with tourism boards, local companies, and technology providers to increase the system's services, gain access to exclusive information, and improve user experiences.

2.8.10. Support and Feedback Mechanism

Develop strong customer service channels for users to resolve questions, technical concerns, and comments as soon as possible. Implement ways for users to submit feedback on their system experiences.

Considering these factors into account guarantees that the Smart Travel Recommendation and Tourism Support System is user-friendly, safe, tailored, and constantly evolving to satisfy the different demands of travelers touring Sri Lanka while adhering to ethical and privacy concerns.

2.9. Project Requirement

2.9.1. Functional Requirements

2.9.1.1. Security and privacy

It is the system's responsibility to maintain users' personal information secure and private, such as facial recognition data and trip history. To put it another way, the system should have robust security measures in place to prevent unwanted access to sensitive data, and it should respect users' privacy by not sharing or using this information without their consent.

2.9.1.2. Facial Recognition

The system is designed to capture and evaluate user facial traits and emotions. As a result, it can understand how users are feeling and adjust its recommendations accordingly. For instance, if a person appears to be joyful, the algorithm may recommend positive content or activities. This personalized method attempts to improve user experience by offering suggestions based on the user's current emotions, so making interactions with the system more exciting and meaningful to individual users.

2.9.1.3. Emotion Analysis

The system must detect basic emotions (happy, sad, angry, surprise, fear, disgust, and neutrality) from facial expressions in real time. The accuracy of emotion recognition is critical for providing personalized activity suggestions based on users' emotional states.

2.9.1.4. Profiles and User Authentication

Users should be able to establish profiles, login securely, and obtain personalized suggestions based on their travel preferences and history.

2.9.1.5. Suggestions for Activities and Tips

Based on observed emotions and user profiles, the system should deliver contextually and culturally suitable activity suggestions.

Relevant suggestions increase user engagement and happiness, resulting in favorable behavioral changes.

2.9.1.6. User Engagement and Feedback

Users must interact with interactive interfaces to examine suggestions and provide feedback. User interaction and feedback systems are critical for improving system accuracy and satisfaction.

2.9.1.7. Recommendation Engine

To create tailored recommendations for attractions, activities, lodgings, and dining options in Sri Lanka, the system must examine tourist preferences, travel history, and current location.

2.9.2. Non- Functional Requirements

2.9.2.1. Performance

The system must be able to handle multiple concurrent user queries while also delivering quick response times for suggestions, real-time information, and user authentication.

2.9.2.2. User Experience and Usability

The system should have an intuitive user interface that allows users, regardless of technological expertise, to navigate and engage with it.

2.9.2.3. Scalability

Design the system to handle an increasing user base and data volume, while also assuring scalability to support future growth and user expectations.

2.9.2.4. Reliability

The system should be dependable and available around the clock, eliminating downtime and providing consistent performance to always fulfill the needs of tourists.

2.9.2.5. Accuracy of Recommendation

The recommendation algorithms in the system should deliver accurate and relevant suggestions while minimizing errors in location, activity, and accommodation recommendations.

3. Implementation And Testing

3.1. Implementation

The research project introduces the "Travel Buddy" suite, which includes web and mobile applications designed to improve the travel experiences of travelers visiting Sri Lanka. These applications seek to provide tailored travel advice and support services, transforming the way tourists plan their itineraries and access real-time information when visiting Sri Lanka.

3.1.1. Travel Buddy Mobile Application

- Purpose: "Travel Buddy," a mobile-based application, serves as a dynamic tool for customized trip recommendations and real-time tourist support services.

3.1.1.1. Functionality

- Personalized Recommendations: Travel Buddy provides customers with personalized recommendations for destinations, activities, lodging, and food alternatives based on their tastes, travel history, and current location.
- Real-time Information: The program enhances the whole trip experience by providing up-to-date information about tourist destinations, transportation options, weather conditions, and emergency assistance services.

3.1.1.2. Implementation Technology

The mobile application's frontend is built with React Native, a cross-platform framework noted for its efficiency and uniform user experiences across multiple mobile devices.

3.1.2. Travel Buddy Web Application

- Purpose: The web application is tailored to tourists and the tourism business in Sri Lanka, providing complete assistance and data-driven insights.

3.1.2.1. Functionality

- Tourist Monitoring: Allows for real-time monitoring of tourist activity, allowing stakeholders in the tourism industry to make data-driven decisions.
- Data Analysis: Supports the investigation of tourist habits and preferences, providing tourism authorities and enterprises with useful insights.

3.1.2.2. Implementation Technology

- Front-end Development: React.js is a powerful JavaScript toolkit known for building dynamic user interfaces and providing design flexibility.
- Backend Development: Using Node.js for the backend development of the web application, which provides a powerful server-side environment suited for handling real-time updates and allowing seamless data management.

In conclusion, Travel Buddy provides customized recommendations to tourists via its mobile app while also delivering real-time monitoring and data analysis to stakeholders via its web-based app. Travel Buddy's use of technologies such as React Native, React.js, and Node.js assures cross-platform compatibility, efficiency, and real-time data handling capabilities, making it an invaluable tool for improving travel experiences in Sri Lanka.

3.2. Model Implementation

3.2.1. Tourists Assistant Chatbot

- **Data Preparation**: The **train_test_split** method was used to divide the dataset into training and testing sets, with a test size of 25%.
- **Model Building:** The model was built by constructing a text classification model with a scikit-learn Pipeline, which includes the following components:

Pipeline

Why We Use Pipeline?

Pipeline from scikit-learn was used to organize and simplify the complicated procedures required in data preprocessing and model training. We were able to develop a seamless and reproducible procedure with the Pipeline while lowering the danger of data leakage during feature extraction and transformation.

Figure 3. 1 Pipeline

- Count Vectorization is the process of converting text input into numerical feature vectors.
- The TF-IDF transformation is used to weight the importance of terms in documents when applied to vectorized data.

- Multinomial Naive Bayes classifier was trained using the training data, which made up 75% of the dataset.

3.2.1.1. Model Training and Testing

Using the training data (75% of the dataset), train the text classification model. The trained model's performance was evaluated by making predictions on the test data (text_clf. predict(X_TEST)).

3.2.1.2. Model Serialization

The trained Multinomial Naive Bayes model was serialized and saved to a file named "chatbot.dat" after successful training and testing using the **pickle.dump** method. This serialized model would later be put into the backend of the chatbot.

3.2.1.3. Frontend and Backend Implementation

The final step was to integrate the chatbot into a mobile app-based solution. This was accomplished by:

- To obtain the trained chatbot model, data preparation and model training are carried out in Google Colab.

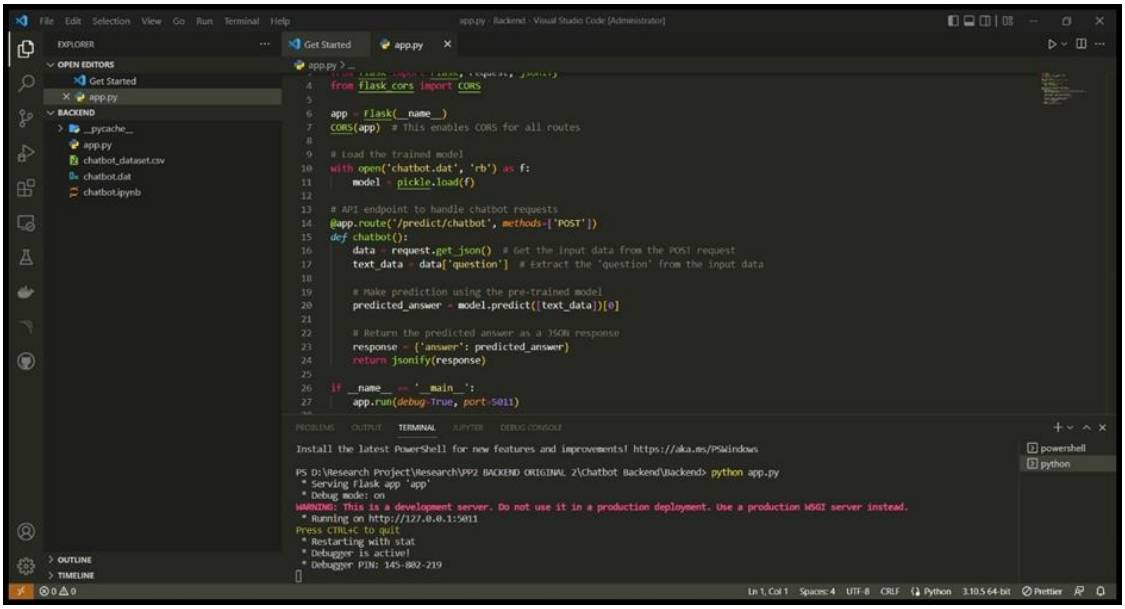


Figure 3. 2 Backend Implementation code (Flask API)

- Creating a Flask API to deploy the pretrained model file as the chatbot's backend. This API allows you to interact with the model in real time.

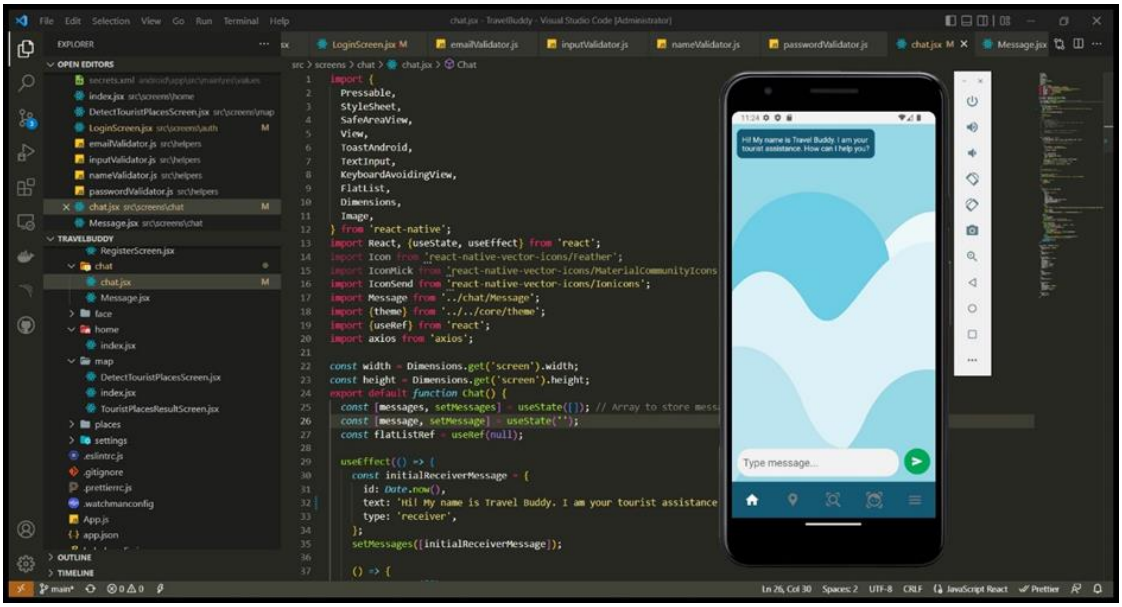


Figure 3. 3 Frontend Implementation code (React Native App) and Chatbot Interface

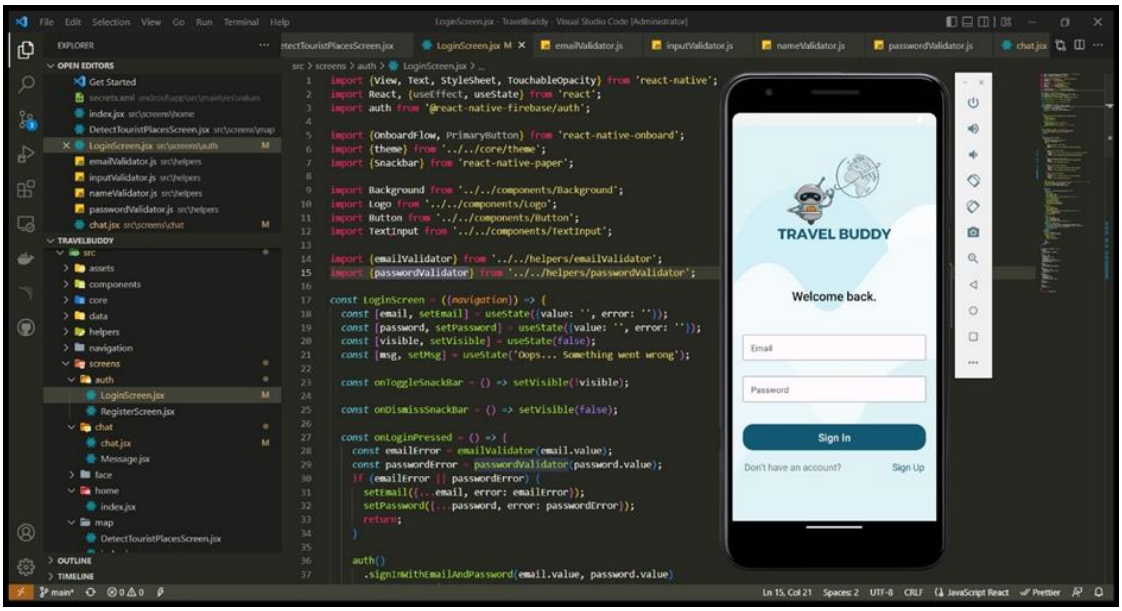


Figure 3. 4 Frontend Implementation code (React Native App) and Login Interface

- Creating a modern UI chatbot interface for the mobile app using React Native, delivering a user-friendly and intuitive experience for travelers.

To summarize, the technique used a systematic manner to successfully gather, preprocess, and use data to build and operate the Smart Tourist Assistance Chatbot. The Multinomial Naive Bayes classifier, together with meticulous data preprocessing, results in an effective and dependable solution for guiding travelers visiting Sri Lanka.

3.2.1.4. Testing & Implementation

Surveys distributed to travelers who have used the Tourist Assistance app will be used to assess the program's efficacy. The survey results will be examined statistically to determine the accuracy of the information offered to tourists as well as their overall experience with the app. The survey responses will be utilized to improve the app and expand its functionality to better suit the needs of tourists visiting Sri Lanka.

The research will be carried out in compliance with ethical norms, assuring the protection of human participants. All participants will provide informed consent, and their anonymity and confidentiality will be maintained throughout the research process.

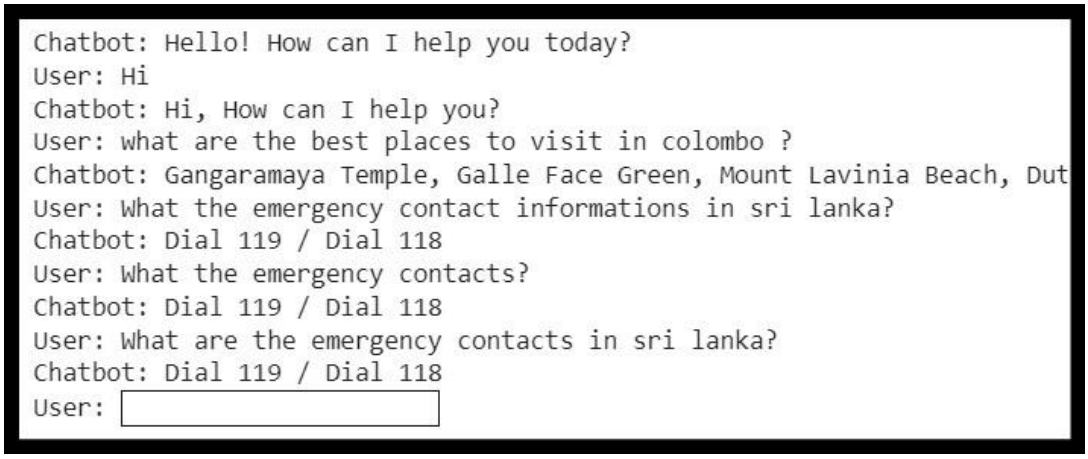


Figure 3. 5 . NLP Model Testing in Colab Example 1

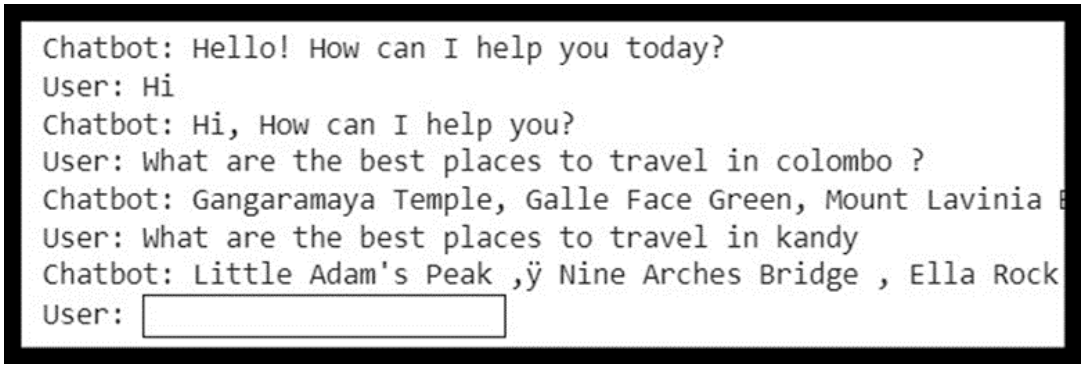


Figure 3. 6 . NLP Model Testing in Colab Example 2

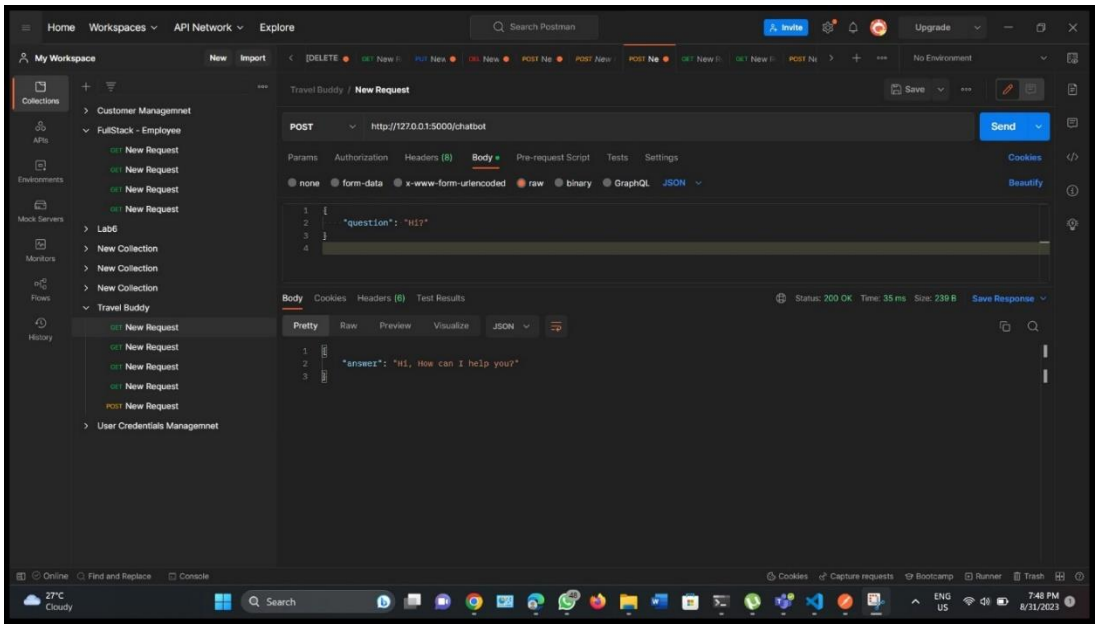


Figure 3. 7 Backend Flask API Testing and Results

3.2.2. *Location Detection and Discovery*

3.2.2.1. *Load Image path and load Google Drive*

This model was trained using Google Colab. First, we provide Google Drive permission and authorize access to the folder path.

Figure 3. 8 Load Image path and load Google Drive

3.2.2.2. *Identifying Classes*

This is useful for determining how many images and classes are in the path. This gives a good indication of how many images are along the route and how many will be trained.

Figure 3. 9 Identifying Classes

3.2.2.3. *Train Classes and images*

To train this model, we used the ResNet50 model. In this scenario, we train the data set 50 times to improve accuracy.

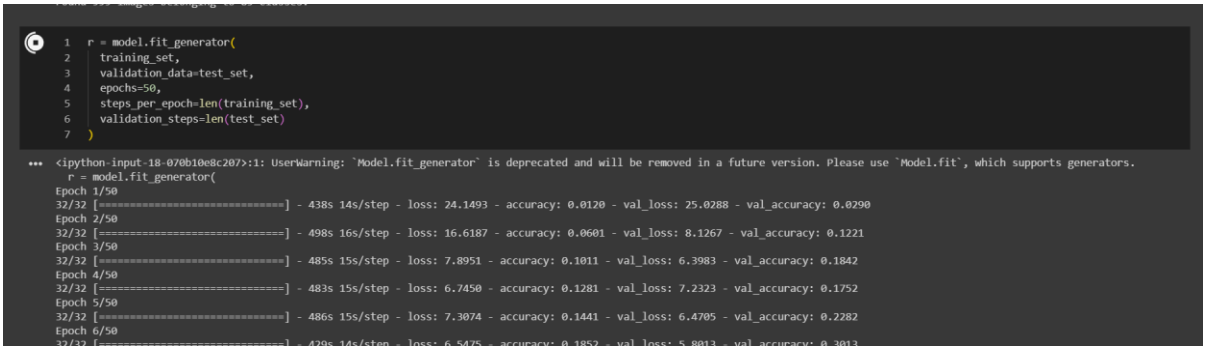


Figure 3. 10 Train Classes and images

3.2.2.4. Saving Load model to the google drive.

After we finish training the model, we should save it to Google Drive.

3.2.2.5. Post-processing for Predicted Output

The code snippet provided looks to do post-processing on the projected output. The function **np.argmax** is applied to the variable **y_pred** along a specified axis using the **numpy** library, most likely to acquire the indices of the array's maximum values. This post-processing stage is critical for extracting the model's final predictions or judgments, allowing the determination of the most likely class or label based on the model's predictions. The resulting **y_pred** variable may contain the predicted class's index or label, allowing for further analysis or utilization of the model's outputs in subsequent tasks.

Figure 3. 11 Post-processing for Predicted Output

3.2.2.6. Training and Validation Loss Plot & Training and Validation Accuracy Plot

During the model's training process, two plots were created to illustrate the training and validation metrics. The first plot shows how the training and validation losses fluctuate over time, providing a visual picture of the model's loss variation. The plot was constructed using

functions such as **plt.plot** and **plt.legend**, and it was presented using the **plt.show()** method. Furthermore, the **plt.savefig()** function was used to save the plot as an image file called **'LossVal_loss'**. The second plot depicts the changes in training and validation accuracy over epochs, offering insight into the model's accuracy evolution during training. **Plt.plot** and **plt.legend** methods were used for plotting, and the **plt.show()** function was used to display the plot, as in the first plot. **Plt.plot** and **plt.legend** methods were used for plotting, and the **plt.show()** function was used to display the plot, as in the first plot. The **plt.savefig()** function was also used to save this plot as an image file named **'AccVal_acc'**.

Figure 3. 12 Training and Validation Loss Plot & Training and Validation Accuracy Plot

3.2.2.7. Image Prediction Using Pre-Trained Models.

Using the function predict_image, image prediction has proven essential in a variety of applications. This function is critical because it reads photos from specified file directories and uses complex techniques such as image array conversion and normalization to prepare images for prediction by pre-trained machine learning models. It generates predictions based on processed images, extracting probability for distinct classes or labels, using the model's capabilities. Displaying the prediction results with the input image helps to see the selected class and provides a better grasp of the projected outcome. This streamlined and efficient approach demonstrates the efficiency of pre-trained models in effectively categorizing and processing photos for a wide range of practical applications.

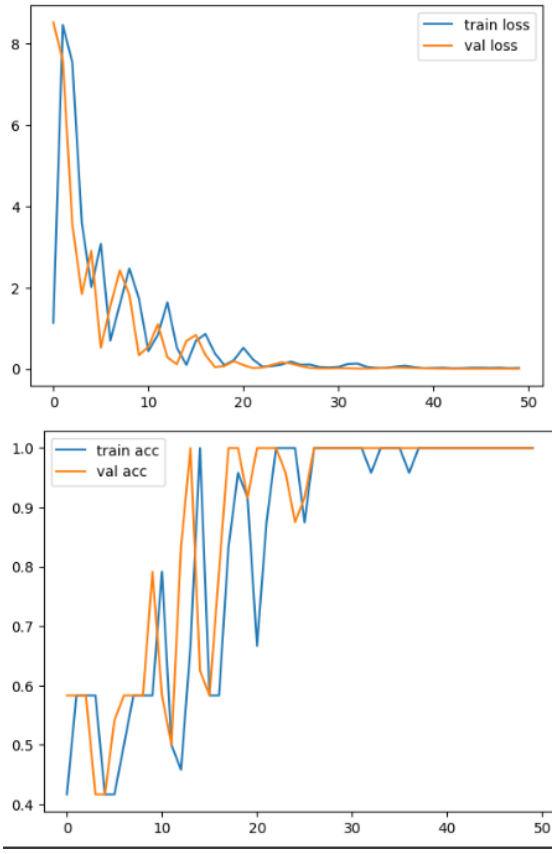


Figure 3. 13 Training and Validation Loss Plot & Training and Validation Accuracy Plot

```
classes
```

```
def predict_image(filename, model):
    img_ = image.load_img(filename, target_size=(224, 224))
    img_array = image.img_to_array(img_)
    img_processed = np.expand_dims(img_array, axis=0)
    img_processed /= 255.

    prediction = model.predict(img_processed)
    prob = prediction
    print(prob[0])

    index = np.argmax(prediction)

    plt.title("Prediction - {}".format(str(classes[index]).title()), size=18, color='red')
    plt.imshow(img_array)
```

```
predict_image('/content/drive/MyDrive/Smart_Travel/tourist_places/Dalada maligawa/22.jpg', model)
```

```
!pip install flask-ngrok
```

```
!pip install pyngrok
from pyngrok import ngrok
```

Figure 3. 14 Image Prediction Using Pre-Trained Models

3.2.2.8. Image Prediction Function for Pre-Trained Models

The predict_image function tries to predict the content of an image located at the supplied file path. Preprocessing processes performed by the function include loading the image, turning it to an array, and normalizing the array for compatibility with the machine learning model. The pre-trained model is then used to make predictions about the image content, with the projected class shown alongside the input image. To make accurate predictions, ensure that the model variable is properly declared and initialized before executing the predict_image method.

```
y_pred = np.argmax(y_pred, axis=1)
model=load_model('place_model.h5')
img=image.load_img('/content/drive/MyDrive/Smart_Travel/tourist_places/Dalada maligawa/10.jpg',target_size=(size,size))
```

Figure 3. 15 Image Prediction Function for Pre-Trained Models

3.2.2.9. Mobile Application

User can add images using their phone camera and images.

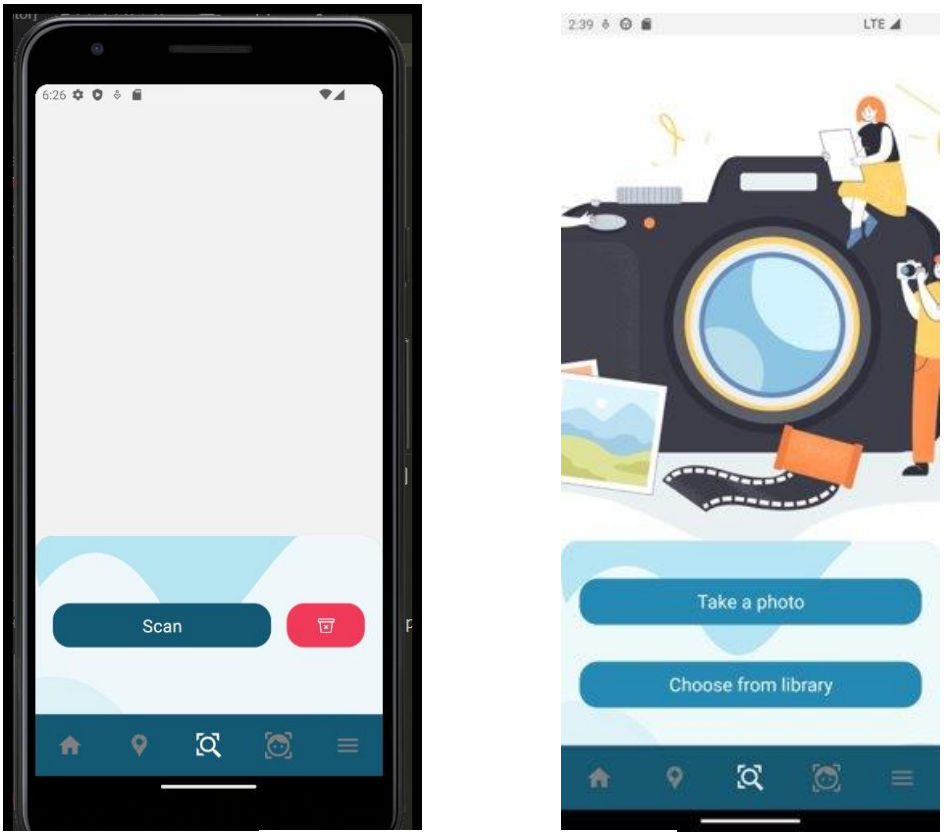


Figure 3. 16 Mobile Application

3.2.3. Location and Service Recommendation System

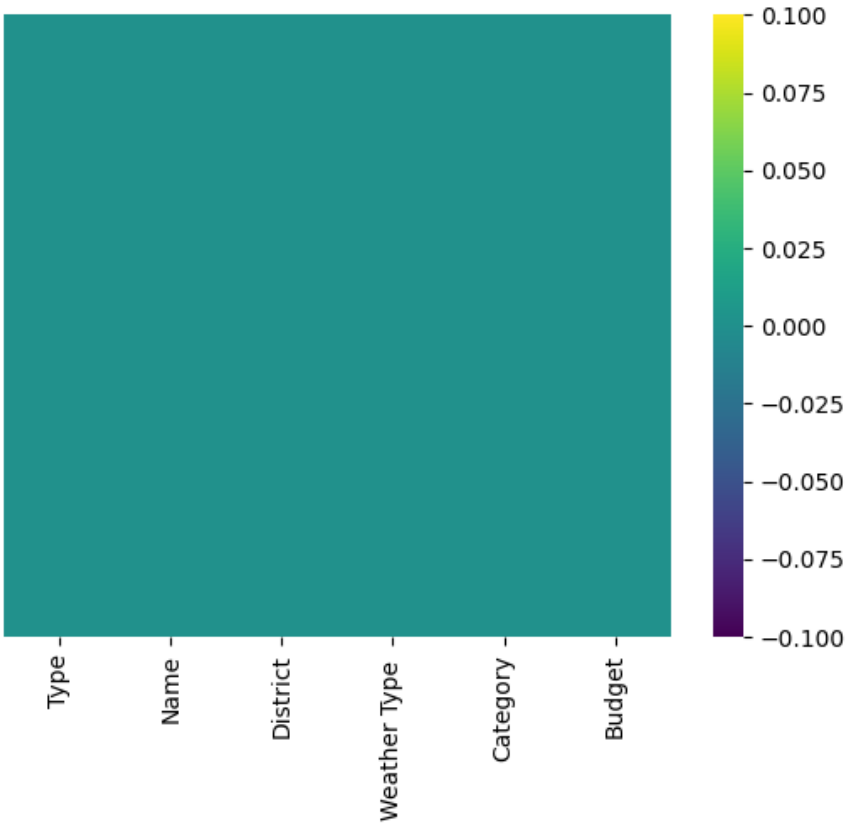
3.2.3.1. Data Quality Visualization

This heatmap aids in the identification of missing values in the dataset. Each heatmap cell represents a data point. The intensity of the color denotes the existence or absence of data in that cell. Darker spots imply that data is missing.

[yticklabels=False,] This option is configured to disable the display of y-axis labels. When there are many rows in the dataset, it is frequently used to avoid overcrowding and to focus on the missing data pattern.

cmap="viridis": The color scheme "viridis" is used to depict the intensity of missing values. Although other color maps are available, "viridis" has been chosen for this investigation.

```
sns.heatmap(data.isnull(),yticklabels=False, cmap="viridis")
```



3.2.3.2. District Name Standardization Function

The code defines a function that converts district names to numeric codes, which aids data pretreatment by standardizing district names for analysis, visualization, and modeling. When an input district name does not match any of the established mappings, it ensures robust preprocessing by returning an empty string.

```
def district_data(value):
    res=""
    if value=="Colombo":
        res=1
    elif value=="Gampaha":
        res=2
    elif value=="Mannar":
        res=3
    elif value=="Kilinochchi":
        res=4
    elif value=="Mulative":
        res=5
    elif value=="Jaffna":
        res=6
    elif value=="Vavuniya":
        res=7
    elif value=="Batticaloa":
        res=8
    elif value=="Trincomalee":
        res=9
    elif value=="Monaragala":
        res=10
    elif value=="Badulla":
        res=11
    elif value=="Hambanthota":
        res=12
    elif value=="Matara":
        res=13
    elif value=="Galle":
        res=14
    elif value=="Matale":
        res=15
    elif value=="Nuwara Eliya":
        res=16
    elif value=="Kandy":
        res=17
    elif value=="Polonnaruwa":
```

3.2.3.3. Convert Data into Numerical Values

The excerpt employs **'data['Budget'].value_counts()'** to count distinct categories in the 'Budget' column, which is the first step toward comprehending the distribution. It uses **'.str.replace()'** to turn category 'Budget' values into numerical codes, making machine learning methods and statistical models that require numerical input more accessible.

```
data['Budget'].value_counts()

data['Budget']=data['Budget'].str.replace('High','1')
data['Budget']=data['Budget'].str.replace('Low','0')
data['Budget']=data['Budget'].str.replace('Free','2')
data['Budget']=data['Budget'].str.replace('Medium','3')

def budget_data(value):
    res=""
    if value=="High":
        res=1
    elif value=="Low":
        res=2
    elif value=="Free":
        res=3
    elif value=="Medium":
        res=4


    if(res==""):
        print(value)

    return res
```

Another code section converts categorical values in the 'Type' column to numerical counterparts, which improves interoperability with statistical analysis and machine learning. This transformation maintains consistency across 'Type' categories, reducing errors caused by categorical data consumption.

```
data['Type'].value_counts()

def Type_data(value):
    res=""
    if value=="Tourist Shops":
        res=1
    elif value=="Travel Agents":
        res=2
    elif value=="Water Sports Centers":
        res=3
    elif value=="Spa & Wellness Centers":
        res=4
    elif value=="Restaurants":
        res=5

    if(res==""):
        print(value)

    return res

data['Type'] = data['Type'].apply(Type_data)
```

It converts categorical 'Weather Type' column values to numerical codes, making them more suitable for various analysis procedures. It ensures consistency and minimizes discrepancies in data analysis and machine learning by assigning numeric representations to each category.

```
def Weather_data(value):
    res=""
    if value=="Southwest Monsoon":
        res=1
    elif value=="Cooler Climate":
        res=2
    elif value=="Northeast Monsoon":
        res=3
    if value=="Tropical monsoon Season":
        res=4
    elif value=="Tropical savanna Season":
        res=5
    elif value=="Tropical nature Season":
        res=6
    elif value=="Dry-monsoonal Season":
        res=7
    elif value=="Tropical rainforest Season":
        res=8

    if(res==""):
        print(value)

    return res

data['Weather Type '] = data['Weather Type '].apply(Weather_data)
```

The code converts categorical values in the 'Category' column into numerical codes, which is necessary for data analysis, modeling, and comparison across categories. Converting 'Category' to numeric representation facilitates analysis, modeling, and meaningful comparisons across categories.

```
data['Category'].value_counts()

def Category_data(value):
    res=""
    if value=="Itinerary Planning":
        res=1
    elif value=="Urban Tourism Zones":
        res=2
    elif value=="Accommodation Booking":
        res=3
    elif value=="Facials":
        res=4
    elif value=="Villa Hotel Tourism Zones":
        res=5
    elif value=="Ayurvedic Shops":
        res=6
    elif value=="Handicraft Shops":
        res=7
    elif value=="Massages":
        res=8
    elif value=="Wildlife Tourism Zones":
        res=9
    elif value=="Visa Assistance":
        res=10
    elif value=="Batik Shops":
        res=11
    elif value=="Gem and Jewelry Shops":
        res=12
    elif value=="Tour Packages":
        res=13
    elif value=="Body Treatments":
        res=14
    elif value=="Beach Tourism Zones":
```

The code uses Label Encoding to convert categorical values in the 'Name' column into numerical representations, which are required for data analysis and machine learning. It creates a dictionary to map original 'Name' values to encoded number values, and then uses LabelEncoder to assign distinct numerical codes to each 'Name' value in the dataset.

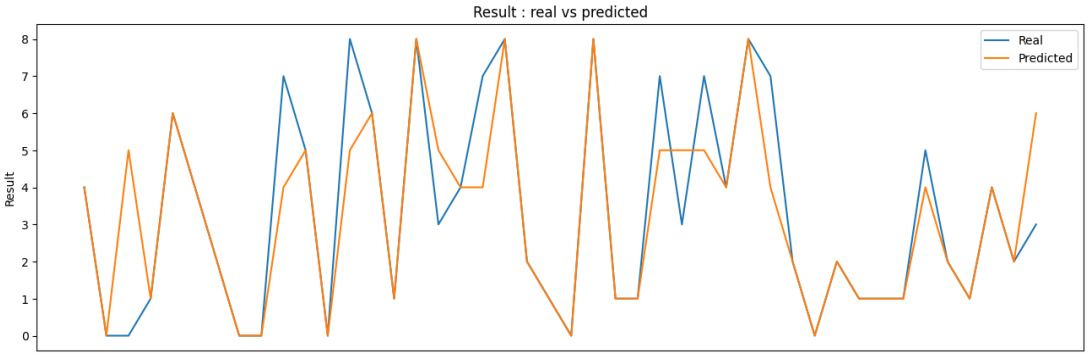
3.2.3.4. Model Training Evaluation and Persistence

It defines a function,'model_score(model),' which is used to evaluate the performance of a machine learning model by fitting it to training data and evaluating its accuracy on testing

data. It specifically initializes a Random Forest Regressor and uses this function to evaluate its correctness. Furthermore, the pickle library is used to save the trained Random Forest Regressor model as "Personalized_Locations.dat," assuring its preservation for future usage or deployment.

```
def model_score(model):
    model.fit(X_train, y_train)
    acc = model.score(X_test, y_test)
    print(str(model)+ ' --> ' +str(acc))

from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
model_score(rf)

import pickle

pickle.dump(rf, open("Personalized_Locations.dat", "wb"))

pickle.dump(lr, open("Personalized_Locations1.dat", "wb"))
```

```
RandomForestRegressor() --> 0.7253965635977052
```



3.2.4. Emotion based Activity Suggestion System

Deep learning has been used in the code to build a facial expression recognition algorithm. The main phases of the algorithm are outlined below:

3.2.4.1. Data preprocessing

To handle image loading and processing operations, the **ImageDataGenerator** method is used. It covers image preparation techniques such as rescaling, data augmentation, and the partitioning of data into training and validation sets.

3.2.4.1.1. Loading images and Rescaling

The **ImageDataGenerator** class is used to load photos from a specified directory **(data_dir)** at first. It also includes the operation **'rescale=1. /255'** which normalizes the image pixel values to a range between 0 and 1. This standardization step is critical for neural network computations since it simplifies calculations and improves the model's learning process.

```
# Load the dataset using ImageDataGenerator
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.2)
```

Figure 3. 17 Loading images and Rescaling

3.2.4.2. Data argumentation techniques

Data augmentation is modifying existing photographs to diversify the training dataset, which improves the model's capacity to generalize to new, previously unseen data. Among the code's enhancement strategies are:

- **'rotation_range=20'** rotates photos randomly by up to 20 degrees, aiding the model in learning facial expressions from various perspectives.

- **'width_shift_range=0.2' and 'height_shift_range=0.2'** horizontally and vertically shift images randomly by up to 20% of their width or height, simulating varied facial emotion placements.
- **'shear_range=0.2'** uses shearing transformations to tilt images in a specific direction, assisting the model in recognizing slanted or skewed expressions.
- **'zoom_range=0.2'** randomly zooms into photos by up to 20%, enabling the model to learn from different facial expression scales.
- **'horizontal_flip=True'** randomly flips photos horizontally, demonstrating that facial emotions aren't solely dependent on left-right orientation.
- **'fill_mode='nearest''** ensures consistency by filling newly formed pixels during augmentation with the closest pixel values from the original image.

3.2.4.3. Validation Set Splitting

The dataset is divided between **training** and **validation** sets using the **validation_split=0.2** parameter in the **ImageDataGenerator**. This means that 80% of the data is used to train the model, with the remaining 20% used to validate it. The validation set is a distinct dataset that is used to validate the model's performance during training, hence preventing overfitting.

3.2.4.4. Model Architecture

The implementation's model architecture is built on the inceptionV3 pre-trained model, with extra custom layers applied to the top.

3.2.4.4.1. Base model: InceptionV3

The algorithm uses the InceptionV3 model, which has been pre-trained on the ImageNet dataset, as the base model. InceptionV3 is a deep convolutional neural network that is particularly good at image categorization. Its complex architecture is designed to detect precise patterns in images, and it employs Inception modules, which let the model to learn features at many spatial scales.

3.2.4.4.2. Custom layers on top of InceptionV3

After extracting features with InceptionV3 layers, unique layers are added to adapt the model for the individual facial expression recognition job.

```
# Create a more complex model
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=img_shape, pooling='avg')
```

```
for layer in base_model.layers:
    layer.trainable = False
```

```
x = base_model.output
x = BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001)(x)
x = Dropout(0.5)(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(len(data.class_indices), activation='softmax')(x)
```

```
model = Model(inputs=base_model.input, outputs=predictions)
model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

Figure 3. 18 Custom layers on top of InceptionV3

The neural network architecture is made up of several essential layers:

- The **'BatchNormalization'** layer normalizes activations from the previous layer in each batch, assuring consistent training, hastening model convergence, and lowering the danger of gradient vanishing or explosion.
- **'Dropout'** layers use a regularization strategy that involves randomly removing neurons during training, which helps to prevent overfitting by avoiding overreliance on specific neurons.
- **'Dense'** layers are highly interconnected, with each getting input from all neurons in the preceding layer. The last dense layer contains neurons that correspond to the dataset's classes (facial expressions), with softmax as the activation function, converting raw scores into possibilities that indicate the likelihood of each class.

3.2.4.4.3. Model compilation

Following the development of the model architecture, it is compiled with the appropriate training setups. **Adam** was chosen as the optimizer, with a learning rate of 0.001. Adam combines the advantages of **AdaGrad** and **RMSProp** to provide an effective learning rate optimization solution. The loss function chosen for dealing with multi-class classification problems is categorical cross-entropy, which determines the difference between predicted and expected class probabilities. The accuracy metric is used as an evaluation measure during model performance tracking during training.

3.2.4.4.4. Custom training loop

A custom training loop is built to train the model over a specified number of epochs. Iterative training with training and validation data is performed, and metrics are produced.

3.2.4.5. Model Training

When compared to the **fit** technique, a custom training loop provides more control and in-depth performance analysis during model training. An alternative explanation, on the opposite hand, shows how the model might be trained using the **fit** technique with stated parameters. The fit technique performs forward and backward passes, computes gradients, updates model weights, and assesses training and validation metrics using **'data'** for training batches and **'validation data'** for unseen data assessment. The number of iterations over the full training dataset is controlled by **'epochs',** while **'verbose'** offers progress updates, including loss and accuracy metrics every epoch. The variable '**history'** stores training details such as loss and accuracy data for each epoch.

```
# Train the model using the built-in fit method
num_epochs = 100  # Adjust the number of epochs as needed
history = model.fit(
    data,
    validation_data=val_data,
    epochs=num_epochs,
    verbose=1
)
```

Figure 3. 19 Model Training

3.2.4.5.1. Model Evaluation

Training and validation loss and accuracy are displayed for each period. Following training, the model is tested on the validation dataset to verify accuracy and loss on unknown data.

```
# Evaluate the model on the test data
test_loss, test_accuracy = model.evaluate(val_data, verbose=1)
print(f"Test Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
```

```
1/1 [==============================] - 1s 1s/step - loss: 3.1309 - accuracy: 0.2632
Test Loss: 3.1309
Test Accuracy: 0.2632
```

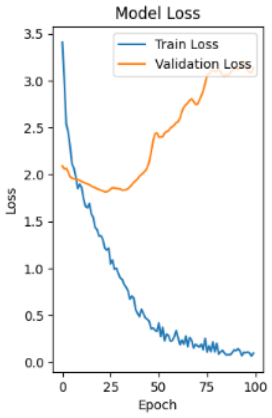
Figure 3. 20 Model Evaluation

3.2.4.6. Result Visualization

The training and validation loss and accuracy data for each epoch are visualized using Matplotlib. This visualization provides information about the model's learning process and assists in identifying potential flaws such as overfitting or underfitting.

```
# Plot training & validation loss values
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(loc='upper right')
```

```
<matplotlib.legend.Legend at 0x787e9fb7e6b0>
```




```
# Plot training & validation accuracy values
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(loc='lower right')
```

```
<matplotlib.legend.Legend at 0x787e9fbf7a90>
```

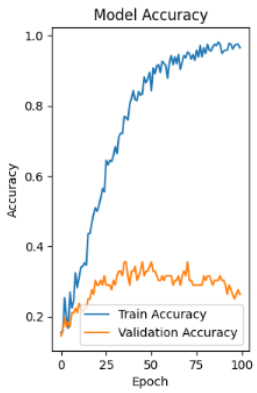


Figure 3. 21 Result Visualization

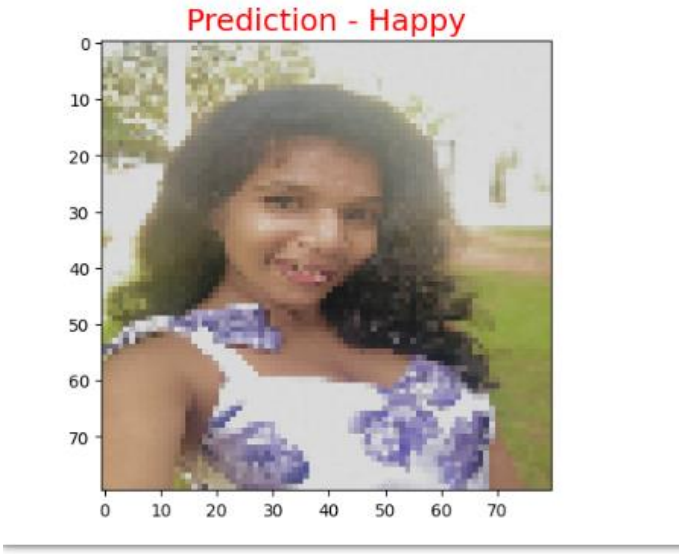


Figure 3. 22 Predicted Result

3.3. Testing

3.3.1. Test plan for Travel buddy Suit

The purpose of this project is to ensure the operation, dependability, and usability of the Travel Buddy suite, which includes the Travel Buddy mobile and web applications, to improve the tourism experience in Sri Lanka. Testing will cover both applications' functional, non-functional, and user-centric elements.

3.3.1.1. Testing Objectives

Test the accuracy and effectiveness of tailored recommendations with the Travel Buddy smartphone app. Examine the Travel Buddy web app's real-time information provision and user monitoring features. Compare system performance, security, and user experience across devices and platforms. Ensure that privacy legislation, data security, and ethical considerations are followed.

3.3.1.2. Test phase and activities

a. Unit Testing

Validation of individual components/modules for accuracy and correctness is the objective. Individual functions, classes, and methods within the applications should be tested.

b. Integration Testing

Ensure that diverse components interconnect and perform in a seamless manner. Activities include testing the interactions of components, APIs, and data flow between apps.

c. Functional Testing

Validation of the functionality and features of both programs is the goal. Activities include putting tailored recommendations, real-time information delivery, user monitoring, data analysis, and user interface aspects to the test.

d. Performance Testing

The intention is to evaluate system response times, resource utilization, and scalability. Conduct load testing, stress testing, and system performance analysis under various user loads.

e. Security and Compliance Testing

Validation of data security measures, privacy compliance, and ethical issues is the goal. Activities include testing for vulnerabilities, encrypting data, and ensuring compliance with privacy standards.

3.3.1.3. Test Environment

Android devices with varied screen sizes and operating system versions can use mobile apps. Web Application should be Compatible with multiple browsers (Chrome, Firefox, and Safari) and smartphones.

3.3.2. Test Strategy for Travel Buddy Suite

3.3.2.1. Approach

Use a combination of manual and automated testing techniques. Prioritize functional and user-centric testing, including evaluations of usability and user experience.

3.3.2.2. Test Tools

Selenium for online app automation and Appium for mobile app automation. Jira or a similar bug tracking tool can be used to track bugs and issues.

3.3.2.3. Testing Techniques

For input validations, use Equivalence Partitioning and Boundary Value Analysis. Exploratory testing is used to recreate real-world user scenarios. Compatibility testing across browsers and devices.

This test plan and strategy provide a thorough method to validating the Travel Buddy suite's functionality, performance, security, and user experience features, assuring its dependability and efficacy in boosting tourism experiences in Sri Lanka.

4. Result and Discussion

4.1. Tourists Assistant Chatbot

The research project met its aims by developing a functioning and effective Tourist Assistance Chatbot, which resulted in several remarkable outcomes. To begin, the chatbot dramatically improves the visitor experience by providing exact solutions to Sri Lanka travel-related queries in a timely manner. It includes a wide range of issues such as local attractions, food, transportation, and emergency assistance, all of which contribute to a more enjoyable journey.

Using the Multinomial Naive Bayes classifier, the built chatbot achieved an amazing accuracy rate of 0.910714, assuring consistent responses and improving the entire trip experience. Furthermore, the chatbot replaces guidebooks and online sources as a dependable and up-to-date information source, guaranteeing tourists receive accurate and timely information for well-informed decision-making.

Finally, the NLP model's fast deployment via a Flask API into an Android app assures easy accessibility, making it a convenient and user-friendly option for travelers seeking assistance on their mobile devices.

```
[ ]
    # Data preprocessing
    df = df.fillna("null_value")
    X = df.drop('Answers', axis=1)
    y = df['Answers']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)


    # Load the trained model
    with open('chatbot.dat', 'rb') as f:
        model = pickle.load(f)


    # Make predictions
    X_TEST = X_test['Question'].to_list()
    y_pred = model.predict(X_TEST)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy:", accuracy)

    Accuracy: 0.7586206896551724
```

Figure 4. 1 Trained NLP Model's Accuracy

4.1.1. Research Findings

a. Information Gathering

To assure the chatbot's efficacy, we took a multifaceted strategy, gathering insights from a variety of sources. Interviews with visitors, internet resources, and YouTube travel vloggers served as the foundation for our extensive data collection, allowing us to build a solid knowledge library for the chatbot.

b. Data Preprocessing

Extensive data preprocessing was carried out to structure questions and relevant replies utilizing the gathered data. Any missing values were addressed by designating a "null_value." To prepare the dataset for subsequent model training, we implemented tokenization, removed stop words, and used TF-IDF vectorization.

c. Model Selection

For our text classification project, we chose the Multinomial Naive Bayes classifier because of its lightweight nature and proficiency in managing text classification problems, particularly when dealing with multiple features. This model is adept at handling discrete information such as word counts within text documents, providing noteworthy speed and performance even when computational resources are limited.

d. Data Collection

We gathered extensive data about Sri Lanka, including geographical information, local cuisine insights, cultural nuances, tourist precautions, and emergency contact information. This information was gathered from a variety of sources, including YouTube travel vlogs, online resources, and direct interactions with travelers via interviews.

4.2. Location Detection and Discovery

The Location Detection and Prediction System makes use of a pre-trained ResNet50 model and Flask to allow users to upload photos of tourist destinations, after which the system predicts the location and gives pertinent information from a related text file. The picture data is augmented using the ImageDataGenerator for additional training data creation, while the model's layers stay frozen to keep the pre-trained weights. The training and validation data are generated using **flow_from_directory** with the model compiled using categorical cross-entropy loss and the Adam optimizer, and the training process is tracked using loss and accuracy visualizations. The trained model is saved in h5 format and loaded with **load_model**. The system's goal is to give tourists with a seamless interface via which they can simply identify and obtain information about various tourist locations.

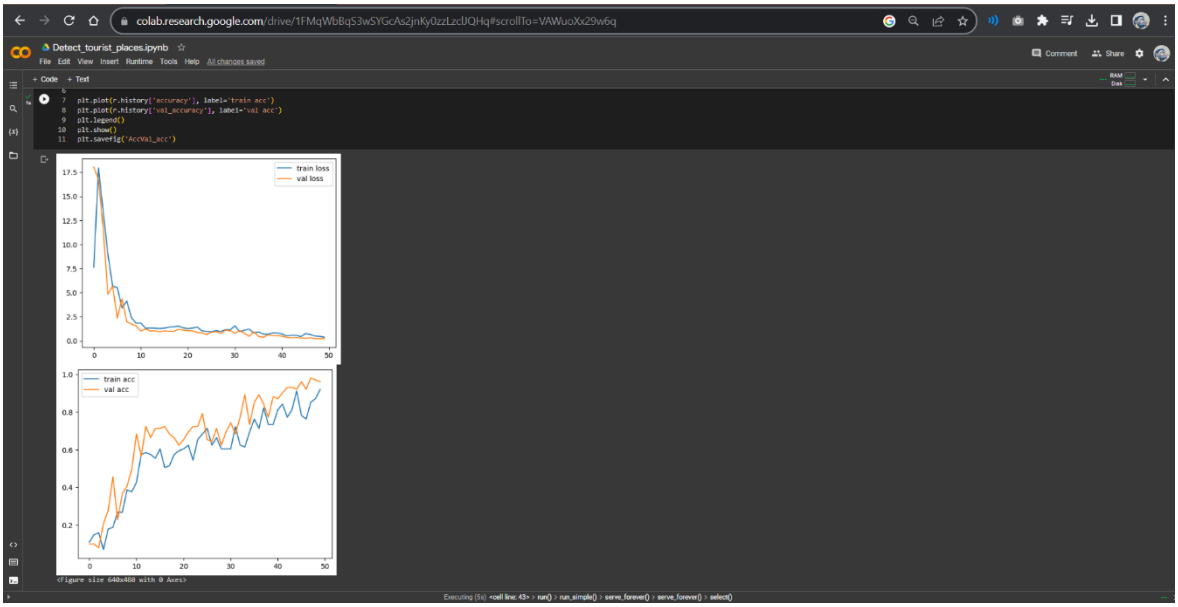


Figure 4. 2 Location Detection and prediction System: Data Preprocessing, Modeling, and Results Analysis

TensorFlow and Keras are used in the system for a variety of functions, including image preprocessing, model training, and prediction. To ensure a broad and robust training dataset, data preparation techniques like as rescaling, shearing, zooming, and flipping are used. The **fit_generator** function facilitates model training by allowing the system to optimize model parameters depending on training and validation datasets. The usage of the h5 file maintains the trained model's persistence, allowing for easy reusability and accessibility. Users may easily interact with the system via the Flask-based web application, which supports seamless

image uploads and real-time position predictions, as well as extensive information about the projected tourist destinations.

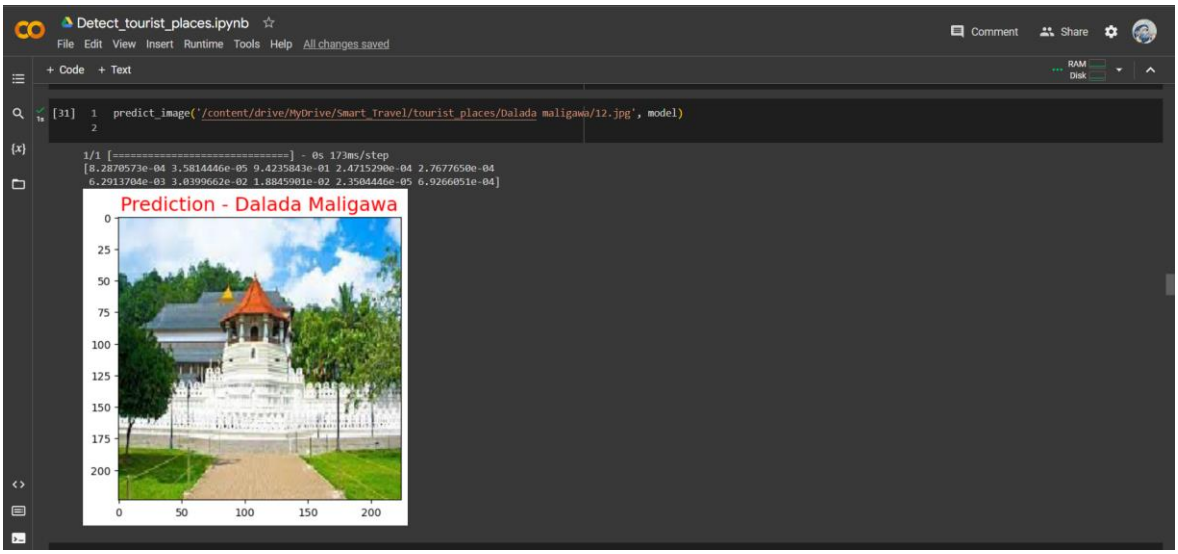


Figure 4. 3 Location Detection and prediction System: Data Preprocessing, Modeling, and Results Analysis

The "Flask-based Location Detection and Prediction System" is an advanced program that uses Flask's features to deliver real-time location predictions. It offers a simple web interface where users may upload images of tourist destinations to get fast location data estimates. Using a pre-trained machine learning model, the system processes submitted photographs, predicts locations consistently, and extracts complete information such as location name, builder, historical period, linked faith, and extensive descriptions. Flask's seamless integration enables fluid communication between the user interface, prediction model, and data storage, resulting in a more efficient user experience. Integration with ngrok allows for comprehensive testing, ensuring consistent performance even when accessed from outside sources.

Users can upload tourist site photographs for location prediction and detailed information retrieval from an Excel file using the Flask-based site Detection and Prediction System, which is implemented using the given code. CORS enables efficient front-end and back-end communication. The system uses a pre-trained model to create folders, pandas to load

location data from an Excel file, and has two key functions: predicting location and saving uploaded photographs.

The 'predict' route parses uploaded photos, predicts location, extracts relevant details from the Excel file, and produces a JSON response containing the expected location. An image-saving path also allows reading and downloading uploaded images, as well as checking file extensions (jpg, jpeg, and png) using the 'allowed_file' function.

Flask's debug mode, available via port 5012 in a local context, aids in issue detection during development. This solution enables accurate position identification and prediction, providing travelers with an interactive and user-friendly experience as they travel to various places.

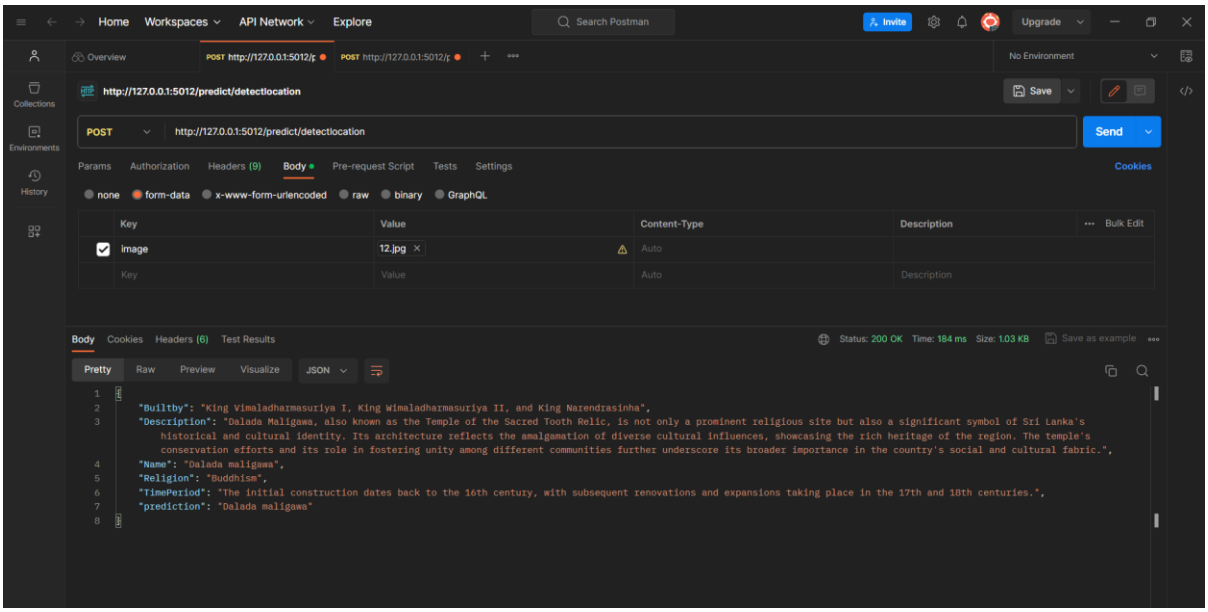


Figure 4. 4 Postman Output

The presented UI represents the development of a mobile application for location detection and retrieval, increasing the user experience with personalized tourism destination recommendations. The React Native-based application allows users to capture or select photographs of numerous tourist attractions and get full information about the identified locations. Users can start the scanning process after capturing or selecting an image, which transmits the image to the server for location recognition. Once the location has been located,

the application obtains pertinent information such as the location's name, the time it was created, the religion associated with it, and a descriptive overview.

Figure 4. 5 Select Image UI

The application's user interface (UI) is visually beautiful, providing users with an intuitive and smooth experience. A capture screen, which allows users to take or pick photographs, and a comprehensive information screen, which displays the detected location's name, historical information, and a brief description, are among the UI components. The application UI also includes interactive navigation and user involvement buttons. Furthermore, the UI includes loading indicators to indicate the processing and retrieval of location information.

The application incorporates critical features such as image uploading, data transmission to the server, and the presentation of returned data in an organized and user-friendly format.

The design encourages easy navigation and access to the application's primary functions, improving overall usability and accessibility for users looking for tailored tourism destination recommendations. The application, with its powerful capabilities and user-friendly layout, is a wonderful tool for anyone wishing to explore and get insights about numerous tourist spots.

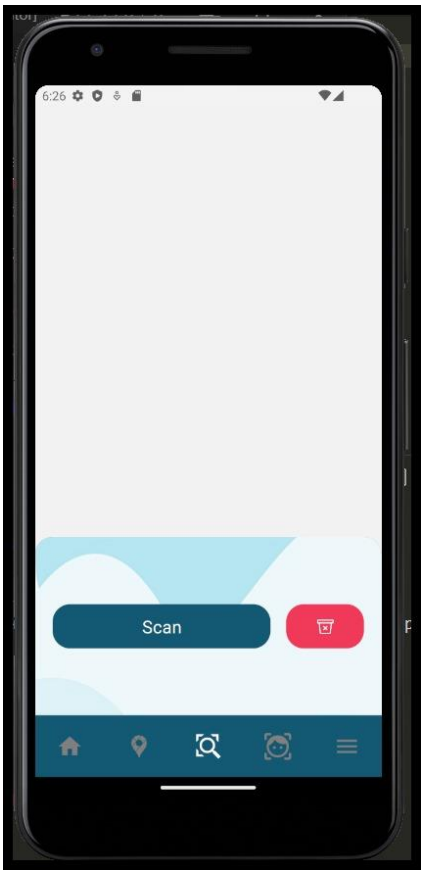


Figure 4. 6 Scan Image UI

- Image Capture/Selection: Users can capture an image using the device's camera by clicking the "Take a photo" button. They can also choose "Choose from library" to select an image from their device's photo gallery.
- Scanning: Users can begin the scanning process after taking or selecting a picture by touching the "Scan" button. This action causes the application to send the selected image to the server for location identification and information retrieval.

- Viewing Location Details: After correctly identifying the location, users can view the obtained information, which includes the place's name, historical background, the period it was built, related religion, and a descriptive overview of the location. On the detailed information screen, this information is presented in a well-organized and visually appealing manner.

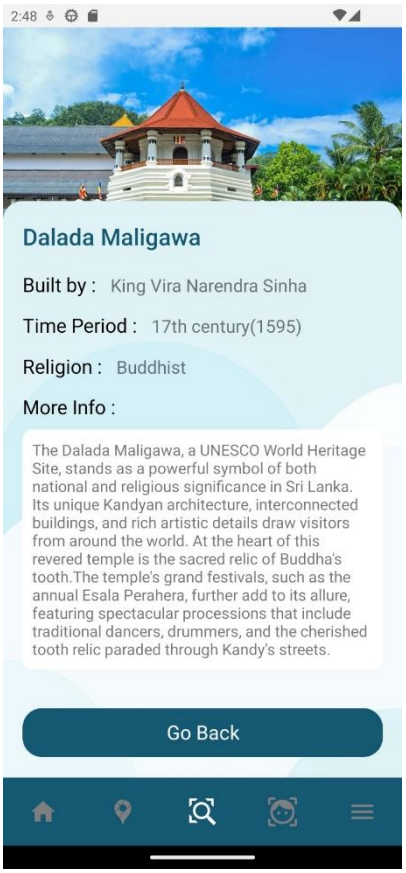


Figure 4. 7 Location Details & Detected image UI

The application allows users to easily browse between screens. By tapping the "Go Back" button, users can return to the image capture screen to take or select images of different tourist attractions for further research. The following user input highlights the steps users can take to engage with the React Native application for detecting places and getting location details, resulting in a user-friendly and engaging experience.

4.3. Location and Service Recommendation System

Using pandas, the script loads a dataset from the CSV file 'Personalized_Locations.csv'. It uses a heatmap from the Seaborn library to illustrate missing data. To understand the data distribution, the 'District' column is evaluated with its value counts. To translate category values into numerical representations, several functions are defined. This is a frequent machine learning stage in which category data is translated into a format that machine learning algorithms can understand. The 'Budget' column is translated from string labels ('High', 'Low', 'Free') to numerical values (1, 0, 2). Other categorical columns, such as 'Type,' 'Weather Type,' and 'Category,' are processed in the same way. LabelEncoder from scikit-learn is used to encode the 'Name' column.

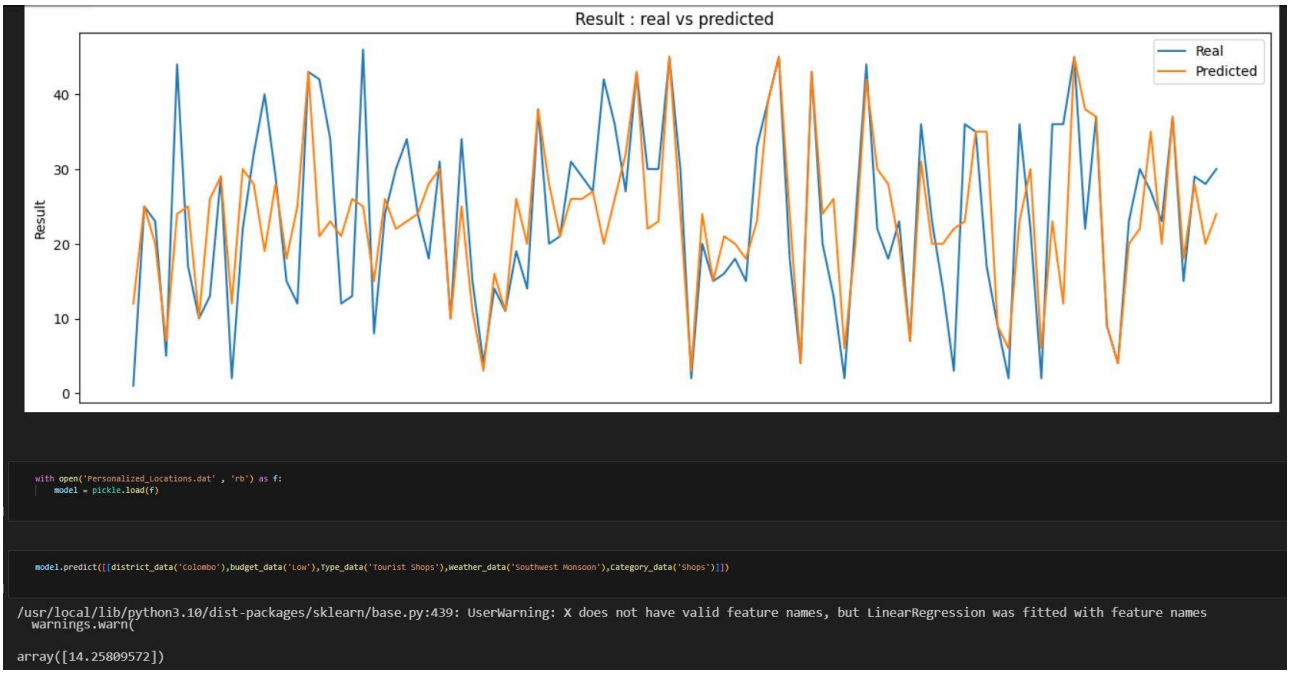


Figure 4. 8 Model Accuracy

For training, the code employs the Support Vector Regression (SVR) and Random Forest Regressor models. The train_test_split function from scikit-learn is used to divide the data into training and testing sets. The R-squared score is used to assess model accuracy. Using the pickle library, the trained Linear Regression model is saved to a file called

'Personalized_Locations.dat'. The saved model is loaded from the 'Personalized_Locations.dat' file by the script. Using sample input data, this model makes a prediction. The script appears to be a tourist suggestion system that predicts a tailored tourism location depending on input criteria. The code does not include the actual dataset or machine learning models used in this research.

In this section, we will look at how to use Flask to create a personalized location recommendation system as a web service. Users can utilize the web service to enter their location preferences and obtain personalized recommendations based on their choices. We give the web service code as well as an example of a user query and the system's answer.

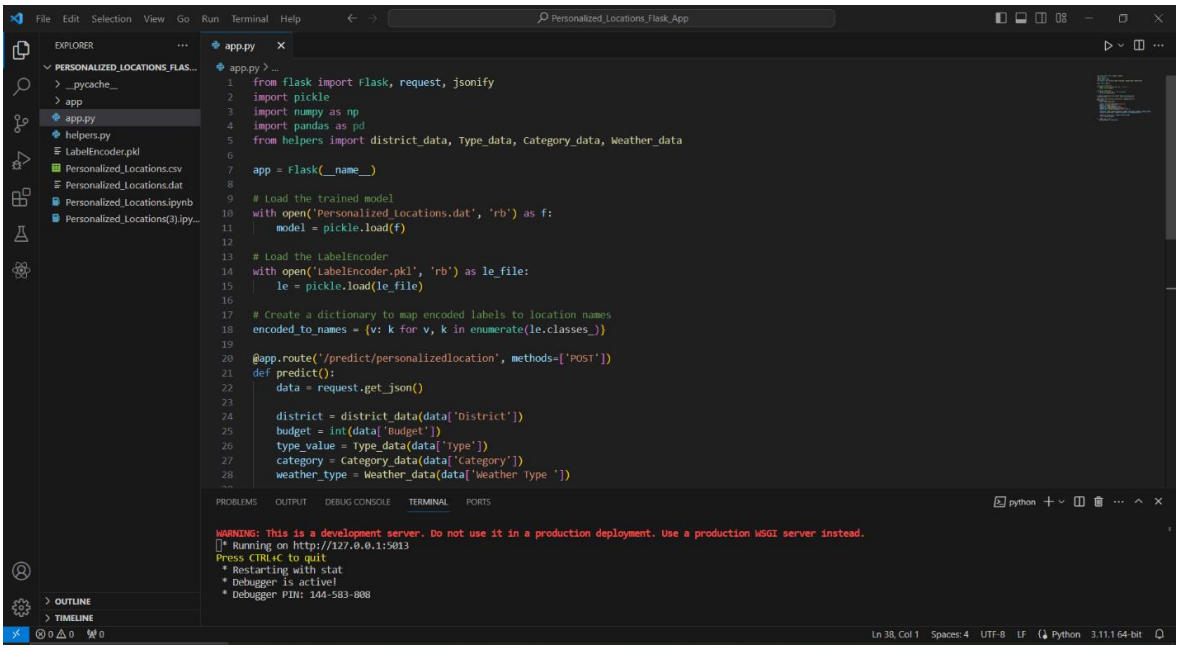


Figure 4. 9 Flask-based Personalized Location and Services Recommendation System Architecture

We successfully created a Flask web service that delivers tailored location recommendations to users. To produce predictions based on user input, the system employs a trained machine learning model and a LabelEncoder for categorical data.

District: The desired district for the location.

Budget: The user's budget preference (0 for low, 1 for high, 2 for free).

Type: The type of location preference (e.g., Tourist Shops, Travel Agents).

Category: The category of location (e.g., Shops, Restaurants).

Weather Type: The preferred weather type (e.g., Southwest Monsoon).

When the system receives user input, it processes it and encodes categorical values using helper functions such as 'district_data,' 'Type_data,' 'Category_data,' and 'Weather_data.' These functions transform user preferences into quantitative representations that the machine learning model can understand. Based on the altered input data, the machine learning model that was trained to predict user preferences generates a prediction. The model suggests the best location for the user based on their choices.

As an example, consider the following user question and system response. Users can submit their location choices as a JSON payload using the parameters listed below.

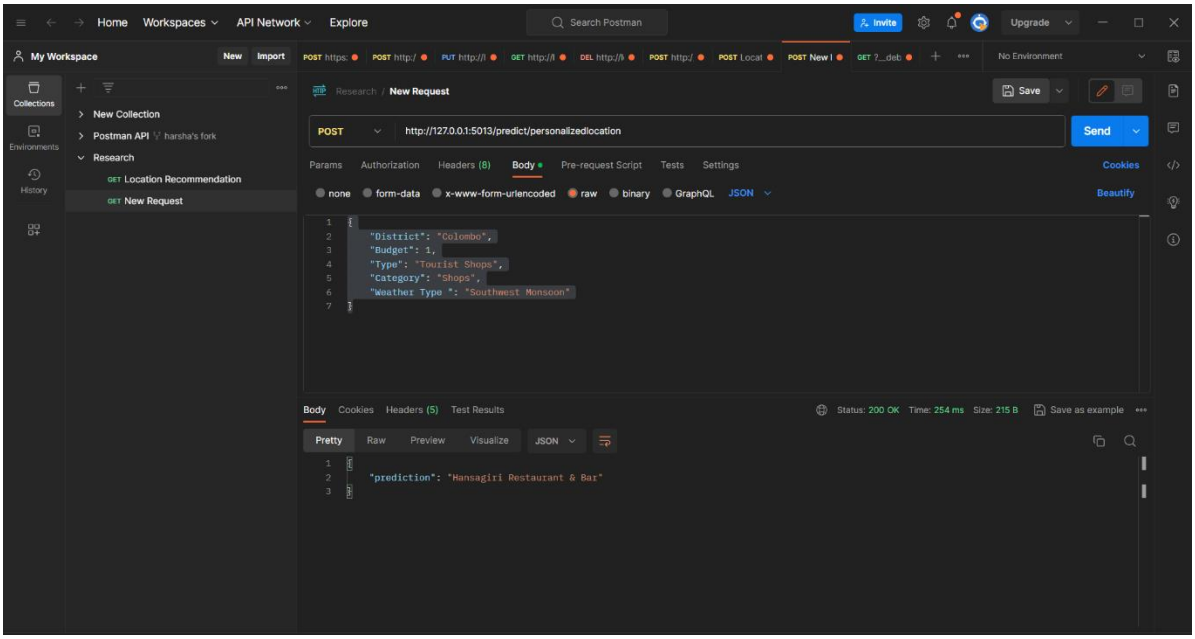


Figure 4. 10 Postmen Output

The React Native app communicates with a Flask-based backend to deliver personalized location suggestions depending on user-selected criteria. Its user-friendly interface includes dropdown menus for quick parameter selection. Location selections such as kind, district, weather, category, and budget are entered by users and submitted to the backend for specialized recommendations. React Native components, iconography, Axios for HTTP queries, and react-native-select-dropdown for dropdowns are used in the code. It makes advantage of React Native's useState hook to handle input field states, successfully capturing user selections.

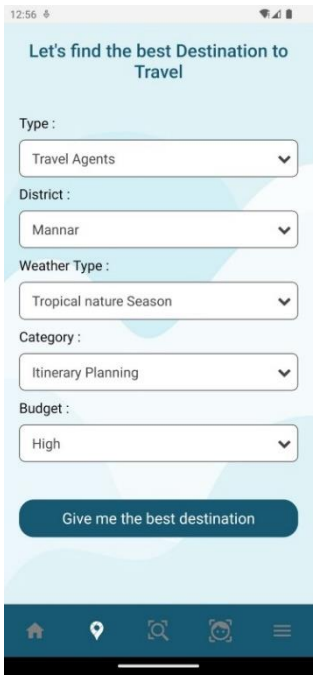


Figure 4. 11 Find the best Destination UI

The code creates dropdown menus for the kind, district, weather type, category, and budget. Types, Districts, WeatherTypes, Categories, and BudgetTypes are arrays that describe the possible options for each dropdown. When the backend responds, the app navigates to the **'TouristPlacesResult'** screen, passing the projected location, selected kind, and district as route parameters. The application is divided into multiple screens, one of which is the "**PredictionDetailsScreen**," which provides detailed information on recommended destinations, including building details. It strives for clarity in information presentation to improve users' understanding of the indicated location. The "**TouristPlacesResultScreen**" displays projected destinations, indicating the state of the processing before providing data such as name, kind, and district. Users can also view the location of the site. Finally, the "**DetectTouristPlacesScreen**" provides a user-friendly interface for preference input, allowing for simple parameter selection for a tailored experience. Overall, these panels offer a smooth and informative user journey, focusing on UI design, assistance, and error management for a better visitor experience.

4.4. Emotion based Activity Suggestion System

The code creates a deep learning model for face emotion identification, with training, validation, and testing phases to assess the model's performance. Here is a more in-depth explanation of the findings:

4.4.1. Training Result, Training Loss, and Accuracy

The model learns from labeled training data throughout the training phase, resulting in primary outcomes such as training loss and accuracy. The training loss measures the variance in the training dataset between anticipated and real emotions, with a lower training loss suggesting efficient emotion identification learning. The training accuracy, on the other hand, shows the percentage of correctly categorized emotions in the training data, with higher accuracy indicating superior facial expression identification by the model. These measures are updated after each training epoch, suggesting that a well-trained model typically exhibits decreased training loss and increasing training accuracy with time, indicating improved emotion categorization ability.

4.4.2. Validation Results, Validation Loss, and Accuracy

The validation phase is critical for evaluating the model's performance on unseen data, which is required for generalization beyond the training dataset. The validation loss and accuracy are two validation metrics:

Validation Loss quantifies the difference between predicted and observed emotions in the validation dataset. A consistent or decreasing validation loss indicates that the model is capable of generalizing to new data.

Validation Accuracy, similar training accuracy, measures the percentage of emotions correctly categorized in the validation dataset. Higher validation accuracy shows that the model is capable of generalization.

These validation results are used to discover potential overfitting issues, such as when the model performs well on training but fails on new data. Consistent and positive validation findings demonstrate the model's successful learning and ability to discriminate emotions across a variety of datasets.

4.4.3. Test Result, Test Loss, and Accuracy

The test step assesses the model's performance on a new, previously unseen test dataset, providing the following results:

The difference between the model's predictions and the actual emotions in the test dataset is represented by Test Loss. A lower test loss indicates improved performance on unique, previously unknown data.

The percentage of accurately classified emotions in the test dataset is indicated by Test Accuracy. Higher test accuracy indicates that the model is capable of recognizing emotions in real-world circumstances.

These test results are critical in determining the model's performance when confronted with completely new data. A high-test accuracy indicates that the model may transfer effectively to real-world settings, ensuring consistent performance beyond the training and validation datasets.

4.4.4. Results Interpretation

4.4.4.1. Accuracy and Loss Trends

Check the accuracy and loss figures for convergence. Convergence indicates that the model effectively learned the data without overfitting.

4.4.4.2. Generalization

Make certain that the model performs well not only on training data, but also on unknown validation and test data, indicating excellent generalization.

4.4.4.3. Fine-Tuning

If the model's performance is insufficient, consider fine-tuning the hyperparameters, changing the model architecture, or increasing the size of the training dataset.

The findings include examining the training, validation, and test metrics, displaying them to uncover trends, and interpreting the model's performance in reliably recognizing facial expressions, according to the previous information. These findings are crucial in determining the trained model's performance in emotion recognition tests. During this study, we also developed a mobile app-based emotion identification and activity suggestion system for travelers. The system employs the InceptionV3 model to precisely distinguish various

emotions (such as joyful, sad, neutral, furious, and so on) from tourists' facial expressions as recorded by the app on their mobile devices.

The major goal of the system is to enhance tourists' experiences by providing personalized activity recommendations or advice based on their observed emotions. To train the InceptionV3 model, a deep learning architecture, to distinguish emotions, a large dataset of face expressions was used. During evaluation, the model demonstrated a high rate of accuracy in determining the emotional states of the tourists. This high level of precision 35 can be attributed to the InceptionV3 architecture's capacity to record complex face patterns and features associated with a variety of emotions.

Figure 4. 12 Predicted Result after Training the Model

4.4.5. Activity Suggestion and Tips

Following successful emotion recognition, the system recommends appropriate activities or provides useful information to tourists. The recommendations are carefully chosen to match the identified emotion and the visitor's location. For example, if a tourist appears to be "happy," the algorithm recommends nearby sights, outdoor activities, or places with a positive mood. If, on the other hand, an emotion such as "sad" is recognized, the system may suggest serene areas, calming activities, or spots where the visitor can unwind and relax.

4.4.6. Personalization and User Experience

The capability of the system to adapt recommendations depending on individual emotions is one of its primary features. By making the trip more thrilling and memorable, personalization improves the overall guest experience. The technique reduces the amount of guessing required in suggesting activities by monitoring the tourist's emotional state, improving the possibility that the recommended experiences will be enjoyable for the visitor.

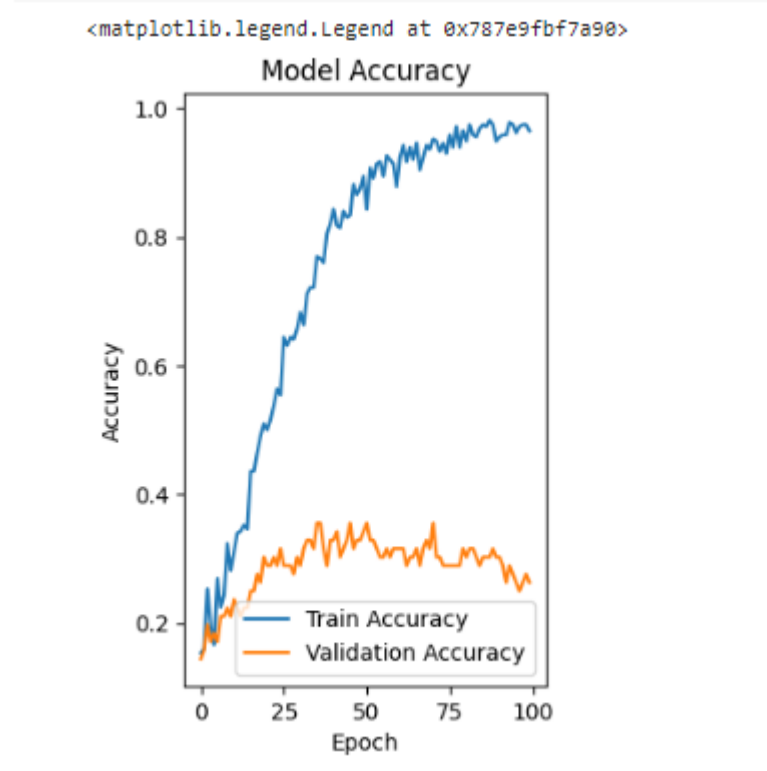


Figure 4. 13 Model Accuracy

Discussion

The creation of the Travel Buddy suite, which includes both mobile and web applications, will have a huge impact on the tourist business in Sri Lanka. This suite attempts to address significant difficulties faced by travelers in trip planning and accessing credible information by combining technology and tourism. Its emphasis on tailored recommendations, real-time updates, and user-centric features is an important step forward in harnessing technology to improve travel experiences.

Tourists visiting Sri Lanka frequently face difficulty in organizing their itineraries due to a lack of comprehensive, individualized recommendations and difficulties acquiring current, reliable information. The Travel Buddy suite tries to alleviate these issues by making personalized recommendations based on visitors' tastes, travel history, and current location. The ability of this system to provide up-to-date information on attractions, transportation, and emergency support services corresponds with tourists' increasing needs, simplifying their planning process, and enriching their experiences.

The usage of sophisticated technologies such as React Native, React.js, and Node.js demonstrates a dedication to innovation and user-centric design. The cross-platform compatibility of React Native, the dynamic interface of React.js, and the real-time data handling capabilities of Node.js all point to a shift toward more efficient and responsive software solutions. These technology decisions demonstrate a commitment to provide a consistent and unified experience across devices and platforms.

The user-centric strategy of the Travel Buddy suite is critical in transforming the tourism scene in Sri Lanka. It caters to the demands of tourists while empowering stakeholders in the tourism sector by providing tailored advice, real-time information, and monitoring capabilities. This user-centricity has the potential to improve the whole travel experience, thereby attracting more tourists and bolstering the growth of Sri Lanka's tourism sector.

However, despite its potential characteristics, the Travel Buddy suite's implementation presents ethical concerns and obstacles. Data privacy, user consent for facial recognition, and adherence to regulatory frameworks become critical. It is critical to address these concerns while preserving transparency and ethical standards to create and sustain user trust.

In the long run, the Travel Buddy suite provides a solid platform for future improvements in travel technology. Continuous improvement, engagement with local stakeholders, and responsiveness to user feedback will all play a role in the suite's evolution. The system's ability to adapt to changing user needs and technical improvements will determine its long-term viability and impact on Sri Lanka's tourism industry.

The following discussion delves deeper into the Travel Buddy suite's significance, challenges, implications, and potential impact, emphasizing its role in revolutionizing the tourism industry in Sri Lanka while emphasizing ethical considerations and the importance of continuous improvement for long-term impacts.

5. Conclusion

The Travel Buddy suite ushers in a new age in Sri Lanka's tourist landscape, promising transforming experiences for visitors. This revolutionary suite of mobile and web applications exemplifies technological innovation and user-centric design, with the goal of reshaping how travelers explore and engage with Sri Lanka's rich tapestry of experiences.

Travel Buddy appears as a trailblazing solution, solving long-standing issues confronting tourists arranging their Sri Lankan excursions. Its main features, such as personalized recommendations and real-time information delivery, are poised to reshape the itinerary planning process. Travel Buddy provides travelers with a tailored travel companion that enriches their exploration adventure by responding to individual interests and providing immediate, accurate insights.

Travel Buddy is, at its core, a synthesis of cutting-edge technology and user-centric design. The suite provides a harmonic balance of efficiency, cross-platform interoperability, and real-time responsiveness by using the prowess of React Native, React.js, and Node.js. This technological integration not only ensures flawless experiences, but also lays the groundwork for future advances in travel technology.

However, because the suite aspires for long-term good benefits, it recognizes the necessity of ethical issues and continuous evolution. Navigating ethical quandaries related to data protection and ensuring continuous improvements based on user feedback remain critical. Maintaining Travel Buddy's reputation and relevance will require upholding ethical standards and embracing an iterative approach.

Travel Buddy, in conclusion, emerges not just as a technological marvel, but also as a catalyst for changing Sri Lanka's travel experiences. Its promise of individualized advice, real-time support, and technological savvy represents a step toward providing travelers with an immersive, tailored exploring journey. Travel Buddy travels a new trajectory as it navigates ethical considerations and embraces continuous growth, guaranteeing amazing adventures and setting new milestones for the future of tourism in Sri Lanka.

6. References

[1] H. Alghamdi, . S. Zhu and A. E. Saddik, "E-Tourism: Mobile Dynamic Trip Planner," *2016 IEEE International Symposium on Multimedia,* pp. 185-188, 2016.

[2] C. Choi, M. Cho, J. Choi, M. Hwang, J. Park and P. Kim, "Travel Ontology for Intelligent Recommendation System," *2009 Third Asia International Conference on Modelling & Simulation,* pp. 637-642, 2009.

[3] Prasadini Ruwani Gunathilake , "Machine Learning-Based Smart Tourist Support System (Smart Guide)," *Springer,* 2022.

[4] S. M, "Industry 4.0 and lean management:," *a proposed integration model and research propositions,* vol. 6, no. 1, pp. 416-432, January 2018.

[5] S. Ziyadin, O. Litvishko, M. Dubrova, G. Smagulova and M. Suyunchaliyeva, "Diversification tourism in the conditions of the digitalization," vol. 10, no. 02, pp. 1055-1070, February 2019.

[6] X. Zheng, Y. Luo, Z. Xu, Q. Yu and L. Lu, "Tourism Destination Recommender System for the Cold Start Problem," *KSII Transactions on Internet and Information Systems,* vol. 10, no. 8, pp. 3192-3212, August 2016.

[7] E. T. S and V. C, "Understanding Personalization of Recommender System: A Domain," vol. 13, no. 15, pp. 12422-12428, 2018.

[8] WTO, "A Practical Guide to Tourism Destination Management," *World Tourism Organization,* 2007.

[9] B. Cvetkovic, R. Szeklicki, V. Janko, . P. Lutomski and M. Lustrek , "Real-time activity monitoring with a wristband and a smartphone.," *Information Fusion,* vol. 43, pp. 77-93, 2018.

[10] L. Santamaria-Granados, J. Mendoza-Moreno, A. ChantreAstaiza and M. Munoz-Organero, "Tourist Experiences Recommender System Based on Emotion Recognition with Wearable Data," *Sensors,* vol. 21, p. 7854, 2021.

[11] L. Santamaria-Granados, J. Mendoza-Moreno and G. Ramirez-Gonzalez, "Tourist Recommender Systems Based on Emotion Recognition," *A Scientometric Review,* vol. 13, no. 2, 2021.

[12] D. Buhalis and R. Law, "Progress in information technology and tourism management," *20 years on and 10 years after the Internet-The state of eTourism research,* 2008.

[13] S. L. T. D. Authority, "Sri Lanka Tourism Development Authority," Sri Lanka Tourism Development Authority, [Online]. Available: https://www.sltda.gov.lk/..

[14] K.K.D.N. Dilshan, C.A.J.P. Chandranath, U.M.D.M. Parussella, Samantha Thelijjagoda, H.M.C.J. Herath and Thilini Jayalath, "JESSY: An Intelligence Travel Assistant," 2021.

[15] Chen, W., Cheng, M. and Huang, Y., "Personalized Tourism Attraction Recommendation Using Natural Language Processing and Machine Learning," vol. 2019, 2019.

[16] Zhang, X. and Cao and J., "Chatbot technology in tourism services: A study of voice and text-based systems," 2018.

[17] Jiang, S., Li , H. and Wu, B., "Tourist chatbot: An AI-based tool for improving tourist," *Journal of Travel Research,* 2019.

[18] E. Sthapit , C. Basnet , Singh and R., "Developing a chatbot using natural language processing to assist tourists in Nepal," *Journal of Tourism and Hospitality Management,* 2020.

[19] M. Haddara , Hamed and S., "Enhancing the accuracy of tourist information retrieval by natural language processing," *Journal of Hospitality and Tourism Technology,* 2018.

[20] P. R. Gunathilake, "Machine Learning-Based Smart Tourist Support System (Smart Guide)," *Springer,* 2022.

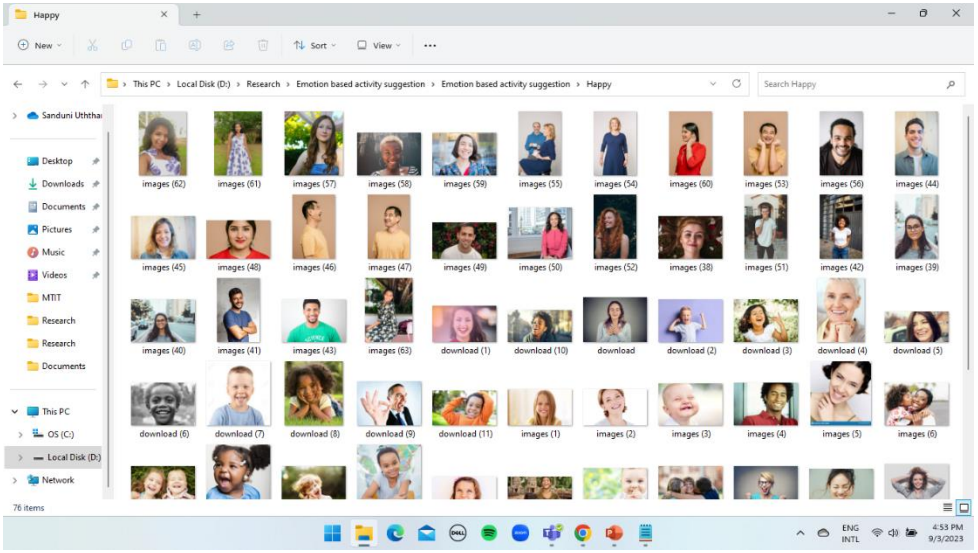
[21] A. Kontogianni and E. Alepis, "Smart tourism: State of the art and literature review for the last six years," *Array,* 2020.

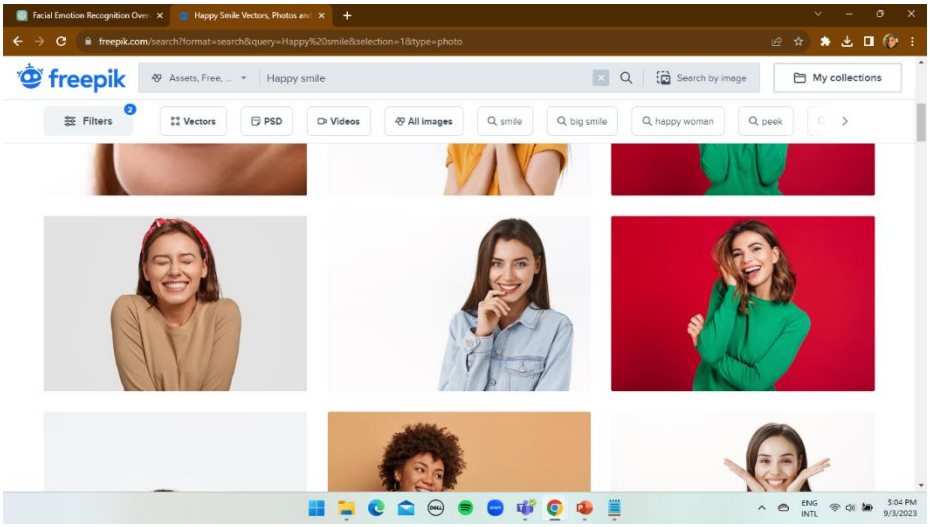
[22] U. Gretzel, M. Sigala, Z. Xiang and C. Koo, "Smart tourism: foundations and developments," *Crossmark,* pp. 180-188, 2015.

[23] H. Alghamdi , S. Zhu and A. E. Saddik, "E-Tourism: Mobile Dynamic Trip Planner," *2016 IEEE International Symposium on Multimedia,* pp. 185-188, 2016.

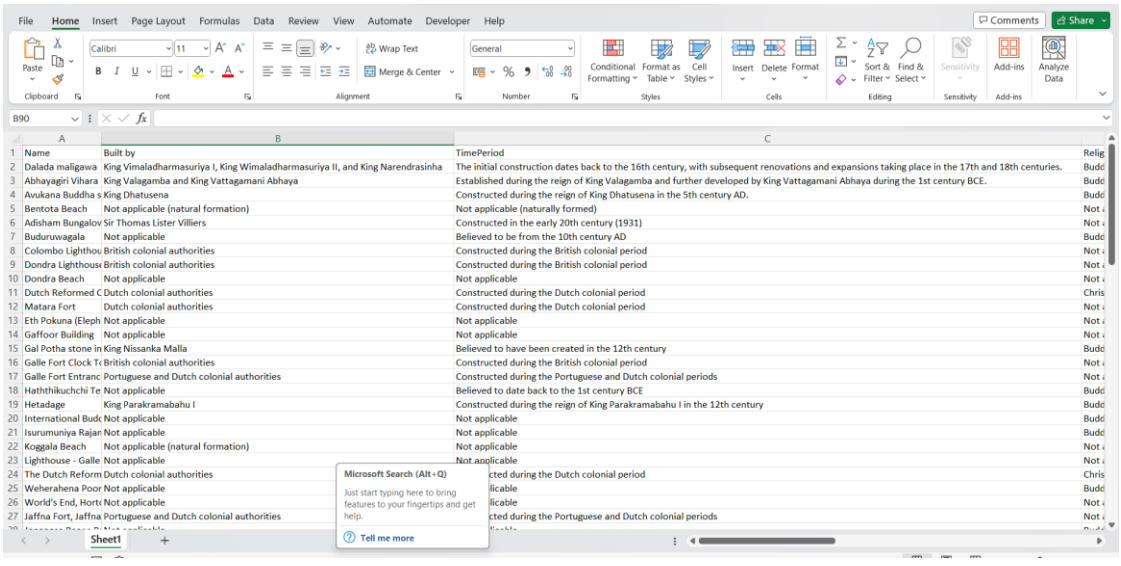
7. Appendices

Appendix A: Collected Dataset



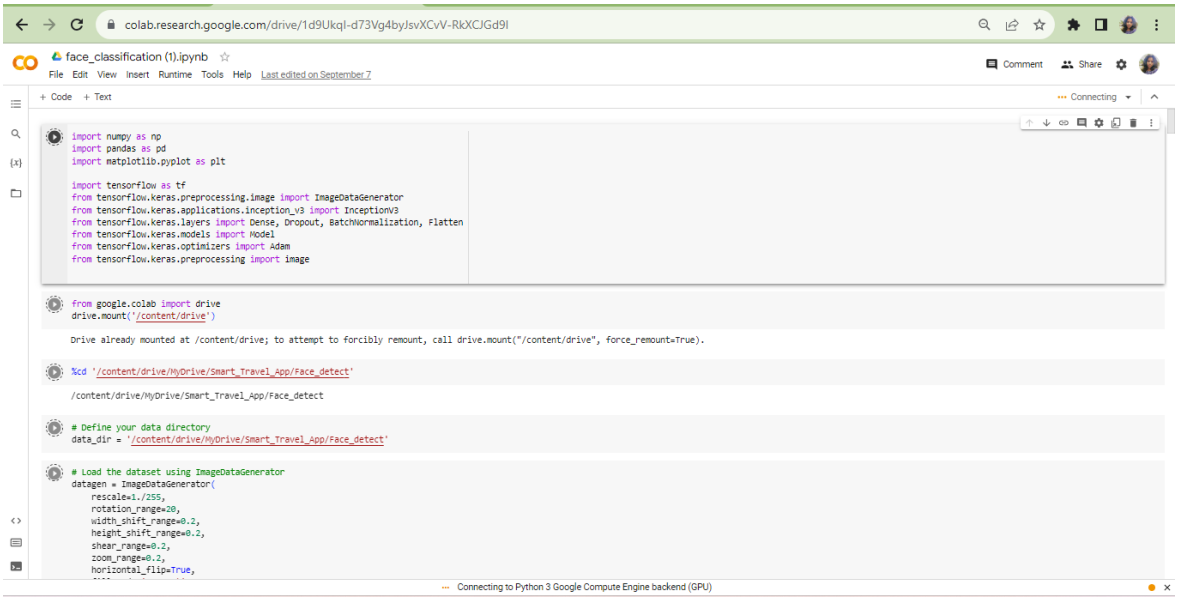

freepik
Happy smile
Search by image
My collections
Filters
Vectors
PSD
Videos
All images
smile
big smile
happy woman
peek


He was suddenly walking so fast toward me



	A	B	C
1	Name	Built by	TimePeriod
2	Dalada maligawa	King Vimaladharmasuriya I, King Wimaladharmasuriya II, and King Narendrasinha	The initial construction dates back to the 16th century, with subsequent renovations and expansions taking place in the 17th and 18th centuries.
3	Abhayagiri Vihara	King Valagamba and King Vattagamani Abhaya	Established during the reign of King Valagamba and further developed by King Vattagamani Abhaya during the 1st century BCE.
4	Avukana Buddha s	King Dhatusena	Constructed during the reign of King Dhatusena in the 5th century AD.
5	Bentota Beach	Not applicable (natural formation)	Not applicable (naturally formed)
6	Adisham Bungalov	Sir Thomas Lister Villiers	Constructed in the early 20th century (1931)
7	Buduruwagala	Not applicable	Believed to be from the 10th century AD
8	Colombo Lighthou	British colonial authorities	Constructed during the British colonial period
9	Dondra Lighthouse	British colonial authorities	Constructed during the British colonial period
10	Dondra Beach	Not applicable	Not applicable
11	Dutch Reformed C	Dutch colonial authorities	Constructed during the Dutch colonial period
12	Matara Fort	Dutch colonial authorities	Constructed during the Dutch colonial period
13	Eth Pokuna (Eleph	Not applicable	Not applicable
14	Gaffoor Building	Not applicable	Not applicable
15	Gal Potha stone in	King Nissanka Malla	Believed to have been created in the 12th century
16	Galle Fort Clock T	British colonial authorities	Constructed during the British colonial period
17	Galle Fort Entranc	Portuguese and Dutch colonial authorities	Constructed during the Portuguese and Dutch colonial periods
18	Haththikuchchi Te	Not applicable	Believed to date back to the 1st century BCE
19	Hetadage	King Parakramabahu I	Constructed during the reign of King Parakramabahu I in the 12th century
20	International Budd	Not applicable	Not applicable
21	Isurumuniya Rajan	Not applicable	Not applicable
22	Koggala Beach	Not applicable (natural formation)	Not applicable
23	Lighthouse - Galle	Not applicable	Not applicable
24	The Dutch Reform	Dutch colonial authorities	cted during the Dutch colonial period
25	Weherahena Poor	Not applicable	licable
26	World's End, Horto	Not applicable	licable
27	Jaffna Fort, Jaffna	Portuguese and Dutch colonial authorities	cted during the Portuguese and Dutch colonial periods

Appendix B: Backend Implementation (Model Implementation)



colab.research.google.com/drive/1d9Ukql-d73Vg4byJsvXCvV-RkXCJGd9I

face_classification (1).ipynb

File Edit View Insert Runtime Tools Help Last edited on September 7

Comment Share Connecting

+ Code + Text

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing import image
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

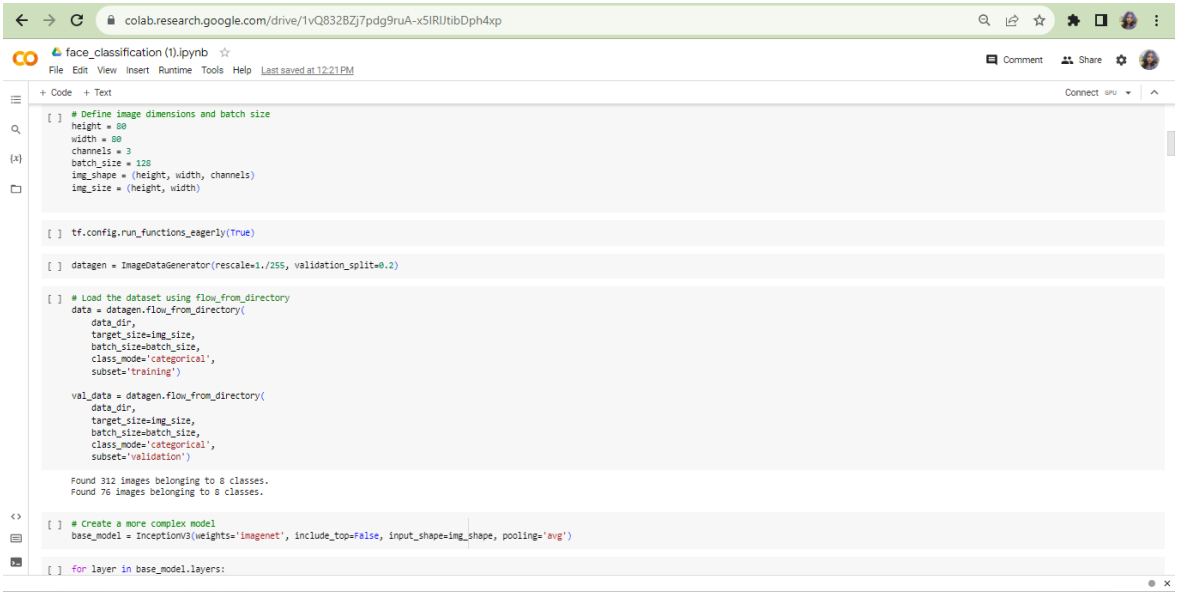
```
%cd '/content/drive/MyDrive/Smart_Travel_App/Face_detect'
```

```
/content/drive/MyDrive/Smart_Travel_App/Face_detect
```

```
# Define your data directory
data_dir = '/content/drive/MyDrive/Smart_Travel_App/Face_detect'
```

```
# Load the dataset using ImageDataGenerator
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
```

Connecting to Python 3 Google Compute Engine backend (GPU)



colab.research.google.com/drive/1vQ832BZj7pdg9ruA-x5IRlJtibDph4xp

face_classification (1).ipynb

File Edit View Insert Runtime Tools Help Last saved at 12:21 PM

Comment Share Connect GPU

+ Code + Text

```
# Define image dimensions and batch size
height = 80
width = 80
channels = 3
batch_size = 128
img_shape = (height, width, channels)
img_size = (height, width)
```

```
tf.config.run_functions_eagerly(True)
```

```
datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
```

```
# Load the dataset using flow_from_directory
data = datagen.flow_from_directory(
    data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training')

val_data = datagen.flow_from_directory(
    data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')
```

```
Found 312 images belonging to 8 classes.
Found 76 images belonging to 8 classes.
```

```
# Create a more complex model
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=img_shape, pooling='avg')
```

```
for layer in base_model.layers:
```

colab.research.google.com/drive/1vQ832BZj7pdg9ruA-x5IRlJtibDph4xp

face_classification (1).ipynb

File Edit View Insert Runtime Tools Help Last saved at 12:21 PM

Comment Share

+ Code + Text

Connect GPU

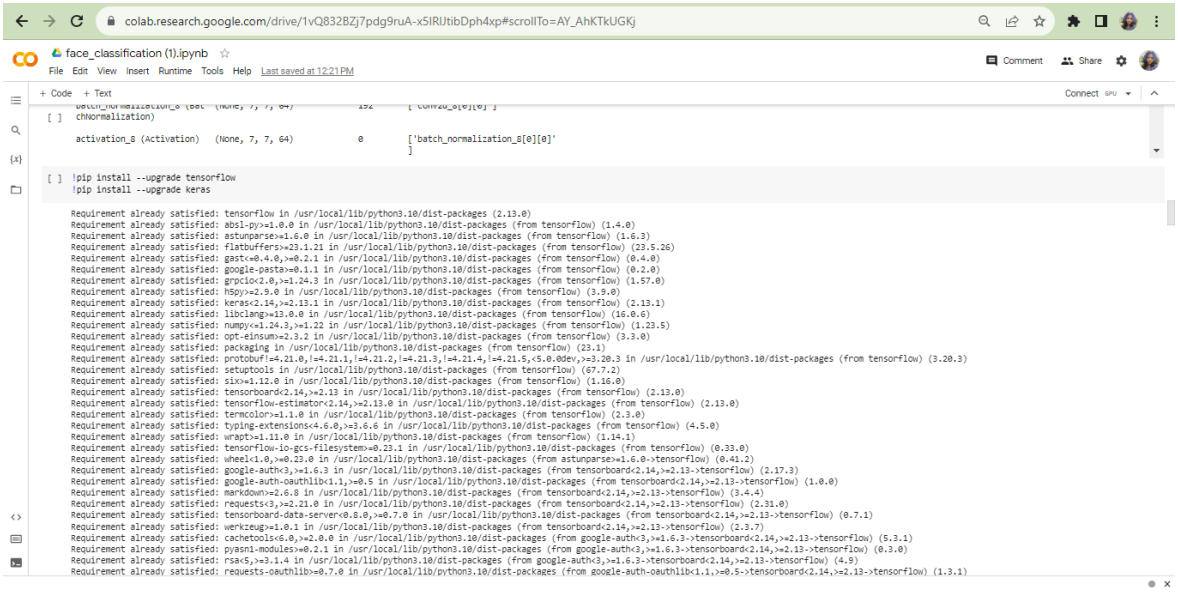
```
[ ] for layer in base_model.layers:
        layer.trainable = False
```

```
[ ] x = base_model.output
    x = BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001)(x)
    x = Dropout(0.5)(x)
    x = Dense(1024, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(256, activation='relu')(x)
    x = Dropout(0.5)(x)
    predictions = Dense(len(data.class_indices), activation='softmax')(x)
```

```
[ ] model = Model(inputs=base_model.input, outputs=predictions)
    model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
[ ] model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================
 input_1 (InputLayer)        [(None, 80, 80, 3)]       0         []

 conv2d (Conv2D)             (None, 39, 39, 32)        864       ['input_1[0][0]']

 batch_normalization (Batch  (None, 39, 39, 32)        96        ['conv2d[0][0]']
 Normalization)

 activation (Activation)     (None, 39, 39, 32)        0         ['batch_normalization[0][0]']

 conv2d_1 (Conv2D)           (None, 37, 37, 32)        9216      ['activation[0][0]']

 batch_normalization_1 (Bat  (None, 37, 37, 32)        96        ['conv2d_1[0][0]']
 chNormalization)
```



colab.research.google.com/drive/1vQ832BZj7pdg9ruA-x5IRlJtibDph4xp#scrollTo=AY_AhKTkUGKj

face_classification (1).ipynb

File Edit View Insert Runtime Tools Help Last saved at 12:21 PM

Comment Share

+ Code + Text

Connect GPU

```
[ ] chNormalization)

    activation_8 (Activation)   (None, 7, 7, 64)          0         ['batch_normalization_8[0][0]'
                                                                   ]
```

```
[ ] !pip install --upgrade tensorflow
    !pip install --upgrade keras
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.13.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.1.21 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.57.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: keras<2.14,>=2.13.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.13.1)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)
Requirement already satisfied: numpy<=1.24.3,>=1.22 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: tensorboard<2.14,>=2.13 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.13.0)
Requirement already satisfied: tensorflow-estimator<2.14,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.13.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.3.0)
Requirement already satisfied: typing-extensions<4.6.0,>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.5.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.33.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.41.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13->tensorflow) (2.17.3)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13->tensorflow) (1.0.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13->tensorflow) (3.4.4)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13->tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13->tensorflow) (0.7.1)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13->tensorflow) (2.3.7)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow) (5.3.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<1.1,>=0.5->tensorboard<2.14,>=2.13->tensorflow) (1.3.1)
```

```
[ ] # Evaluate the model on the test data
    test_loss, test_accuracy = model.evaluate(val_data, verbose=1)
    print(f"Test Loss: {test_loss:.4f}")
    print(f"Test Accuracy: {test_accuracy:.4f}")
```

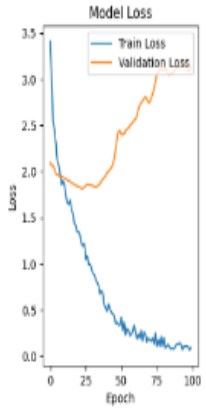
```
1/1 [==============================] - 1s 1s/step - loss: 3.1300 - accuracy: 0.2632
Test Loss: 3.1300
Test Accuracy: 0.2632
```

```
[ ] # Plot training history
    plt.figure(figsize=(12, 5))
```

```
<Figure size 1200x500 with 0 Axes>
<Figure size 1200x500 with 0 Axes>
```

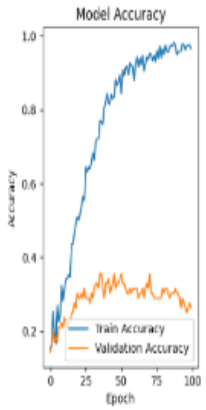
```
[ ] # Plot training & validation loss values
    plt.subplot(1, 2, 1)
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Model Loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(loc='upper right')
```

```
<matplotlib.legend.Legend at 0x787e9fb7e0d0>
```




```
[ ] # Plot training & validation accuracy values
    plt.subplot(1, 2, 2)
    plt.plot(history.history['accuracy'], label='Train Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.title('Model Accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(loc='lower right')
```

```
<matplotlib.legend.Legend at 0x787e9fbf7e50>
```

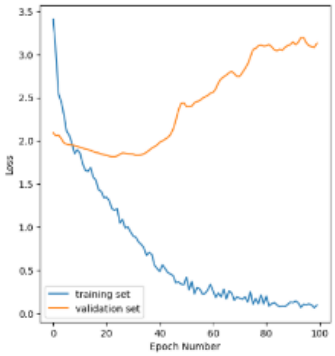


Epoch

```
[ ] plt.show()
```

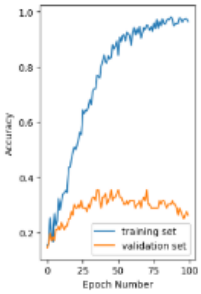
```
[ ] # Plot training history
    plt.figure(figsize=(12, 6))
    plt.subplot(1, 2, 1)
    plt.xlabel('tpoch Number')
    plt.ylabel('Loss')
    plt.plot(history.history['loss'], label='training set')
    plt.plot(history.history['val_loss'], label='validation set')
    plt.legend()
```

```
<matplotlib.legend.Legend at 0x787f6d123oc0>
```




```
[ ] plt.subplot(1, 2, 2)
    plt.xlabel('Epoch Number')
    plt.ylabel('Accuracy')
    plt.plot(history.history['accuracy'], label='training set')
    plt.plot(history.history['val_accuracy'], label='validation set')
    plt.legend()
```

```
<matplotlib.legend.Legend at 0xra7ecffebade>
```




```
# Save the trained model
model_name = 'face_model.h5'
model.save(model_name, save_format='h5')
```

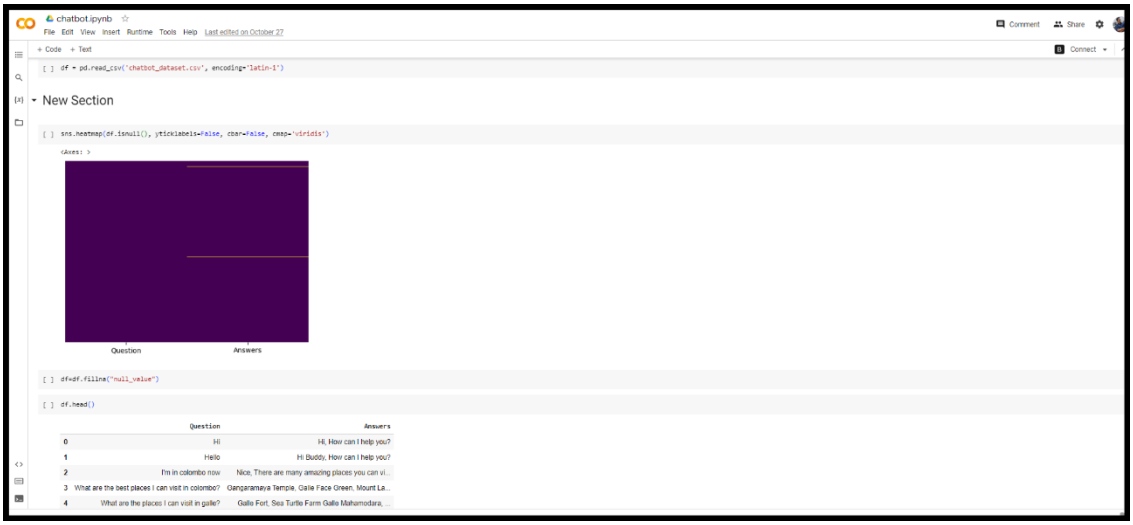
```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using
  saving_api.save_model(
```

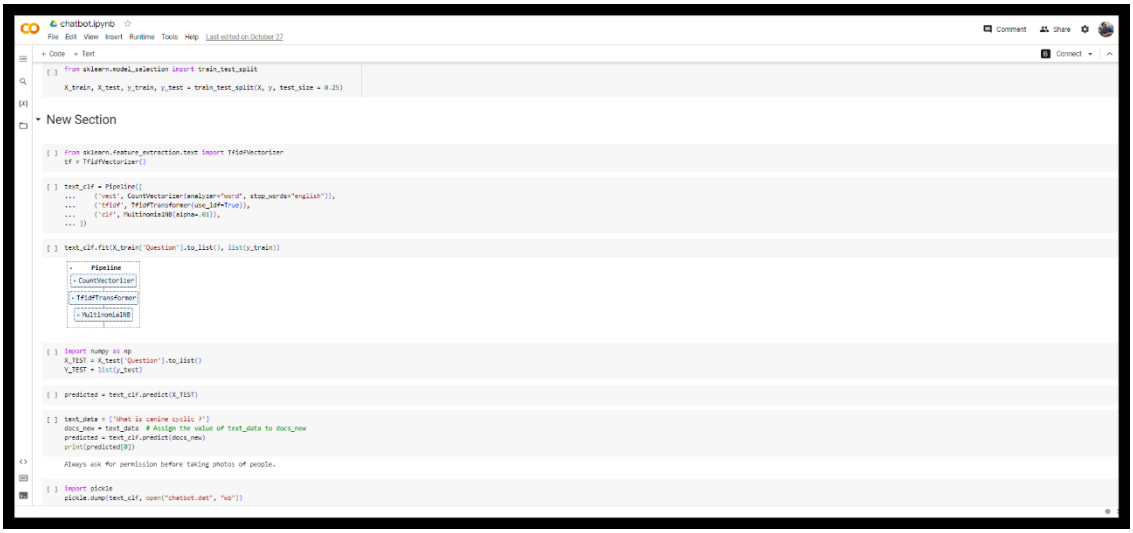
```
[ ] # Get class names
    class_map = data.class_indices
    classes = []
    for key in class_map.keys():
        classes.append(key)
```

```
[ ] # Load the model for prediction
    from tensorflow.keras.models import load_model
    model_name = 'face_model.h5'
    model = load_model(model_name)
```

```
[ ] def predict_image(filename, model, classes):
        img = tf.keras.preprocessing.image.load_img(filename, target_size=(80, 80))
        img_array = tf.keras.preprocessing.image.img_to_array(img)
        img_processed = np.expand_dims(img_array, axis=0)
        img_processed /= 255.

        prediction = model.predict(img_processed)
        class_index = np.argmax(prediction)
        class_name = classes[class_index]

        plt.title(f"Predicted Class: {class_name}", size=18, color='red')
        plt.imshow(img_array)
        plt.axis('off')
```


chatbot.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 27
Comment
Share
Connect
+ Code + Text
import pandas as pd
#numpy
#import numpy as np
import spacy
import seaborn as sns
import string
from tqdm import tqdm
from textblob import TextBlob
from nltk.corpus import stopwords
import nltk
from nltk.stem import WordNetLemmatizer
from nltk import word_tokenize
import re
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
#accuracy
#from sklearn.metrics import accuracy_score
from sklearn.preprocessing import FunctionTransformer
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.pipeline import FeatureUnion
from sklearn.feature_extraction import DictVectorizer
import swifter
[ ] !pip install swifter
Collecting swifter
Downloading swifter-1.4.0.tar.gz (1.2 MB)
1.2/1.2 MB 13.8 MB/s eta 0:00:00
Preparing metadata (setup.py) ... done
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from swifter) (1.5.3)
Requirement already satisfied: psutil>=5.6.6 in /usr/local/lib/python3.10/dist-packages (from swifter) (5.9.5)
Requirement already satisfied: dask[dataframe]>=2.10.0 in /usr/local/lib/python3.10/dist-packages (from swifter) (2023.8.1)
Requirement already satisfied: tqdm>=4.33.0 in /usr/local/lib/python3.10/dist-packages (from swifter) (4.66.1)
Requirement already satisfied: click>=8.0 in /usr/local/lib/python3.10/dist-packages (from dask[dataframe]>=2.10.0->swifter) (8.1.7)
Requirement already satisfied: cloudpickle>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from dask[dataframe]>=2.10.0->swifter) (2.2.1)
Requirement already satisfied: fsspec>=2021.09.0 in /usr/local/lib/python3.10/dist-packages (from dask[dataframe]>=2.10.0->swifter) (2023.6.0)


chatbot.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 27
Comment
Share
Connect
+ Code + Text
[ ] df = pd.read_csv('chatbot_dataset.csv', encoding='latin-1')
New Section
[ ] sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
<Axes: >
Question
Answers
[ ] df=df.fillna("null_value")
[ ] df.head()
Question
Answers
0 Hi
Hi, How can I help you?
1 Hello
Hi Buddy, How can I help you?
2 I'm in colombo now
Nice, There are many amazing places you can vi...
3 What are the best places I can visit in colombo?
Gangaramaya Temple, Galle Face Green, Mount La...
4 What are the places I can visit in galle?
Galle Fort, Sea Turtle Farm Galle Maharmodara, ...


chatbot.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 27
Comment
Share
Connect
+ Code + Text
[ ] from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
New Section
[ ] from sklearn.feature_extraction.text import TfidfVectorizer
tf = TfidfVectorizer()
[ ] text_clf = Pipeline([
... ('vect', CountVectorizer(analyzer="word", stop_words="english")),
... ('tfidf', TfidfTransformer(use_idf=True)),
... ('clf', MultinomialNB(alpha=.01)),
...])
[ ] text_clf.fit(X_train['Question'].to_list(), list(y_train))
Pipeline
CountVectorizer
TfidfTransformer
MultinomialNB
[ ] import numpy as np
X_TEST = X_test['Question'].to_list()
Y_TEST = list(y_test)
[ ] predicted = text_clf.predict(X_TEST)
[ ] text_data = ['What is canine cyclic ?']
docs_new = text_data # Assign the value of text_data to docs_new
predicted = text_clf.predict(docs_new)
print(predicted[0])
Always ask for permission before taking photos of people.
[ ] import pickle
pickle.dump(text_clf, open("chatbot.dat", "wb"))

```python
import numpy as np #numerical operations and data manipulation
import pandas as pd #handling missing data
import matplotlib.pyplot as plt #predected target value
import seaborn as sns #splitting, testing and training dataset
```

Python

```python
#read a CSV file and load its contents into a Pandas DataFrame
data = pd.read_csv('Personalized_Locations.csv', encoding='latin-1')
```

Python

```python
#Display the data in table
data
```

Python

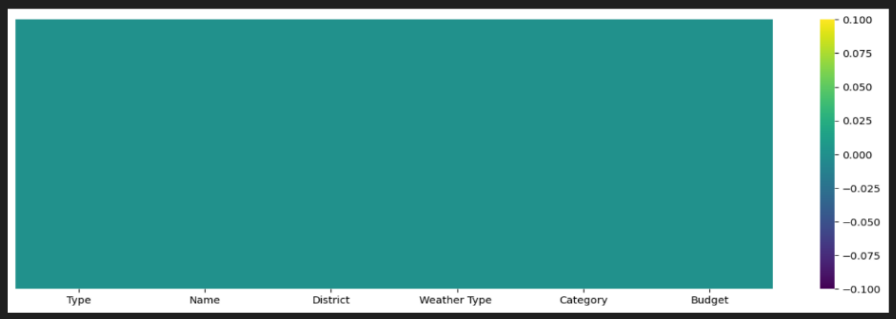
	Type	Name	District	Weather Type	Category	Budget
0	Restaurants	The Bavarian	Colombo	Tropical monsoon Season	Bar	Medium
1	Restaurants	The Bavarian	Colombo	Tropical monsoon Season	Bar	Medium
2	Restaurants	The Bavarian	Colombo	Tropical monsoon Season	Bar	Medium
3	Restaurants	The Bavarian	Colombo	Tropical monsoon Season	Bar	Medium
4	Restaurants	The Bavarian	Colombo	Tropical monsoon Season	Bar	High
...	...	...	...	...	...	...
1497	Spa & Wellness Centers	AZMAARA SPA	Kalutara	Tropical nature Season	Yoga and Pilates	High
1498	Spa & Wellness Centers	AZMAARA SPA	Kalutara	Tropical nature Season	Yoga and Pilates	High
1499	Spa & Wellness Centers	AZMAARA SPA	Kalutara	Tropical nature Season	Body Treatments	High
1500	Spa & Wellness Centers	AZMAARA SPA	Kalutara	Tropical nature Season	Body Treatments	Medium
1501	Spa & Wellness Centers	AZMAARA SPA	Kalutara	Tropical nature Season	Body Treatments	Medium

1502 rows × 6 columns

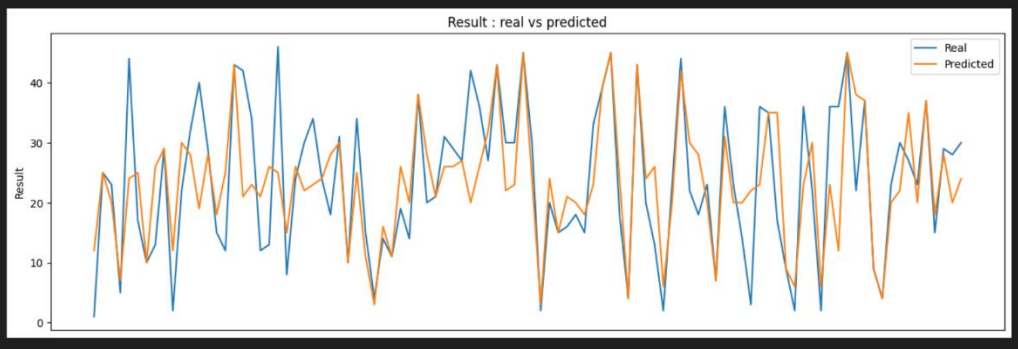
```python
#create a heatmap using the Seaborn library to visualize missing (null) values in the DataFrame
sns.heatmap(data.isnull(),yticklabels=False, cmap="viridis")
```

Python

```
<Axes: >
```




```
<class 'numpy.ndarray'>
<class 'pandas.core.series.Series'>
```

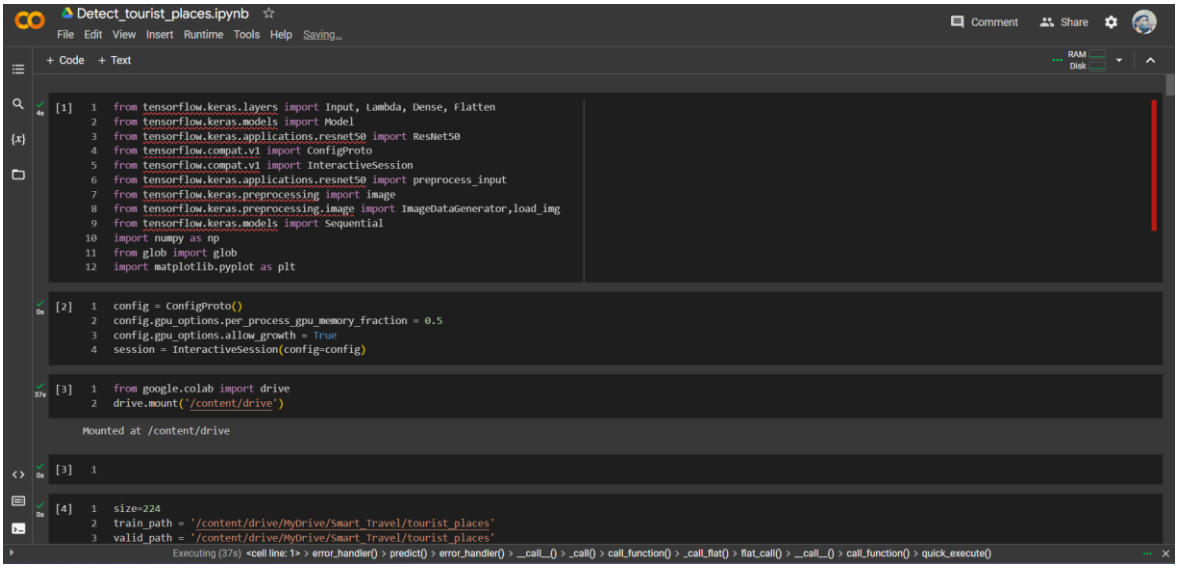


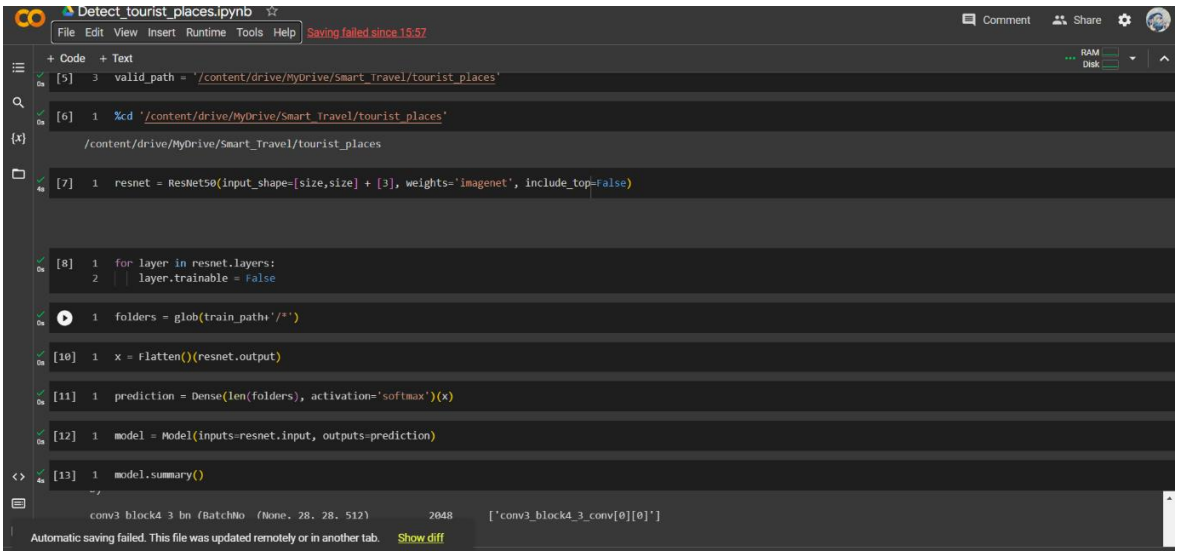

```python
with open('Personalized_Locations.dat' , 'rb') as f:
    model = pickle.load(f)
```

[940]

Python

```python
model.predict([[district_data['Colombo'], budget_data['Low'], Type_data['Tourist Shops'], Weather_data['Southwest Monsoon'], Category_data['Shops']]])
```


Detect_tourist_places.ipynb
File Edit View Insert Runtime Tools Help Saving...
Comment
Share
+ Code + Text
RAM
Disk
[1] from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
[2] config = ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.5
config.gpu_options.allow_growth = True
session = InteractiveSession(config=config)
[3] from google.colab import drive
drive.mount('/content/drive')
Mounted at /content/drive
[3] 1
[4] size=224
train_path = '/content/drive/MyDrive/Smart_Travel/tourist_places'
valid_path = '/content/drive/MyDrive/Smart_Travel/tourist_places'
Executing (37s) <cell line: 1> > error_handler() > predict() > error_handler() > __call__() > _call() > call_function() > _call_flat() > flat_call() > __call__() > call_function() > quick_execute()


Detect_tourist_places.ipynb
File Edit View Insert Runtime Tools Help Saving failed since 15:57
Comment
Share
+ Code + Text
RAM
Disk
[5] valid_path = '/content/drive/MyDrive/Smart_Travel/tourist_places'
[6] %cd '/content/drive/MyDrive/Smart_Travel/tourist_places'
/content/drive/MyDrive/Smart_Travel/tourist_places
[7] resnet = ResNet50(input_shape=[size,size] + [3], weights='imagenet', include_top=False)
[8] for layer in resnet.layers:
layer.trainable = False
folders = glob(train_path+'/*')
[10] x = Flatten()(resnet.output)
[11] prediction = Dense(len(folders), activation='softmax')(x)
[12] model = Model(inputs=resnet.input, outputs=prediction)
[13] model.summary()
conv3_block4_3_bn (BatchNo (None, 28, 28, 512) 2048 ['conv3_block4_3_conv[0][0]']
Automatic saving failed. This file was updated remotely or in another tab. Show diff

Detect_tourist_places.ipynb

File Edit View Insert Runtime Tools Help Saving...

Comment Share Connecting

+ Code + Text

```
# Define classes
class_map = training_set.class_indices
classes = list(class_map.keys())

# Define the folder containing image and text files
data_folder = '/content/drive/MyDrive/Smart_Travel_Test/tourist_places'

# Define an upload folder for storing the uploaded images
upload_folder = '/content/uploads'
os.makedirs(upload_folder, exist_ok=True)

@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Check if an image file is included in the request
        if 'image' not in request.files:
            return jsonify({'error': 'No image file provided'})

        image_file = request.files['image']

        # Check if the file has a valid extension
        if image_file and allowed_file(image_file.filename):
            filename = secure_filename(image_file.filename)
            image_path = os.path.join(upload_folder, filename)

            # Save the uploaded image to the upload folder
            image_file.save(image_path)

            img = Image.open(image_path)
            img = img.convert("RGB")
            img = img.resize((224, 224))  # Resize the image to match the model's input size
            img_array = np.asarray(img)
```

colab.research.google.com/drive/1FMqWbBqS3wSYGcAs2jnKy0zzLzcJQHq#scrollTo=VAWuoXx29w6q

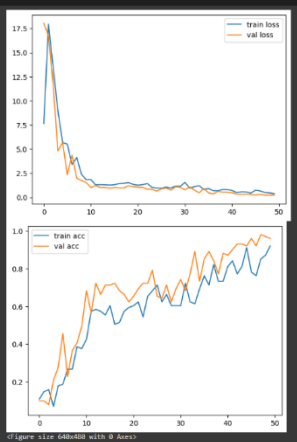
Detect_tourist_places.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

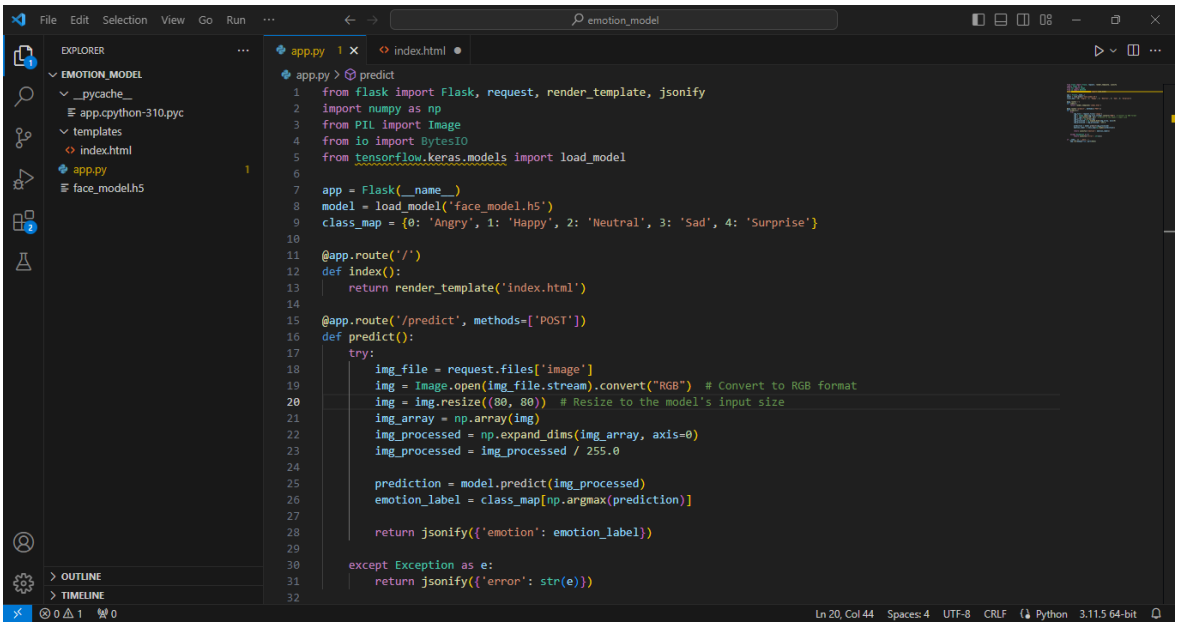
```
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```



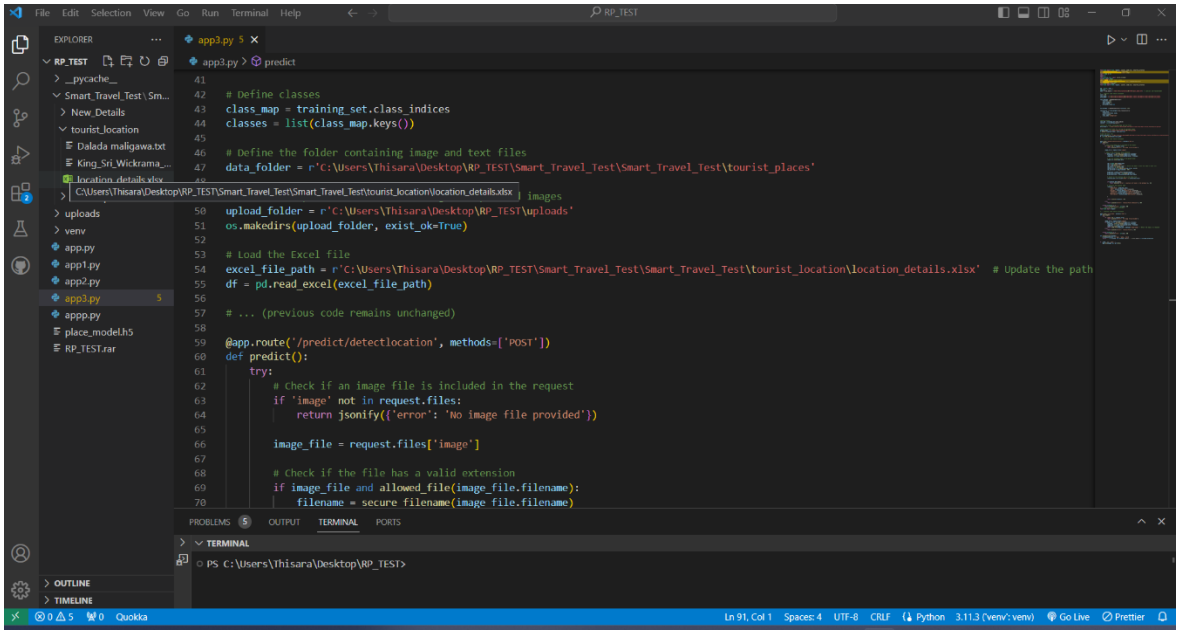
<Figure size 640x480 with 0 Axes>

Executing (5s) <cell line: 43> > run() > run_simple() > serve_forever() > serve_forever() > select()

Appendix C: Back-end Implementation using Flask

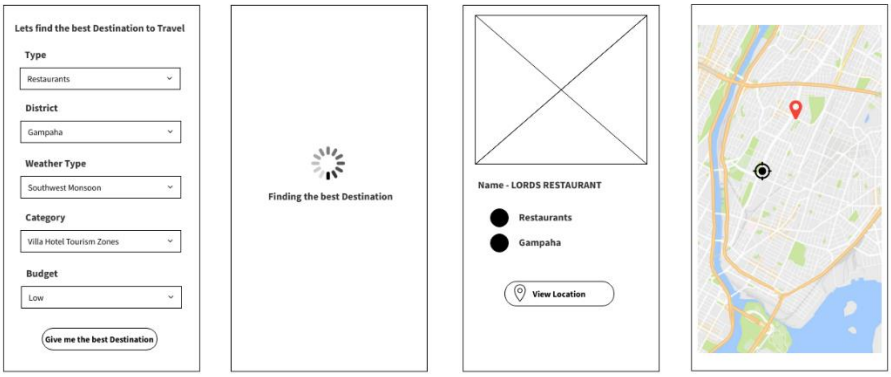


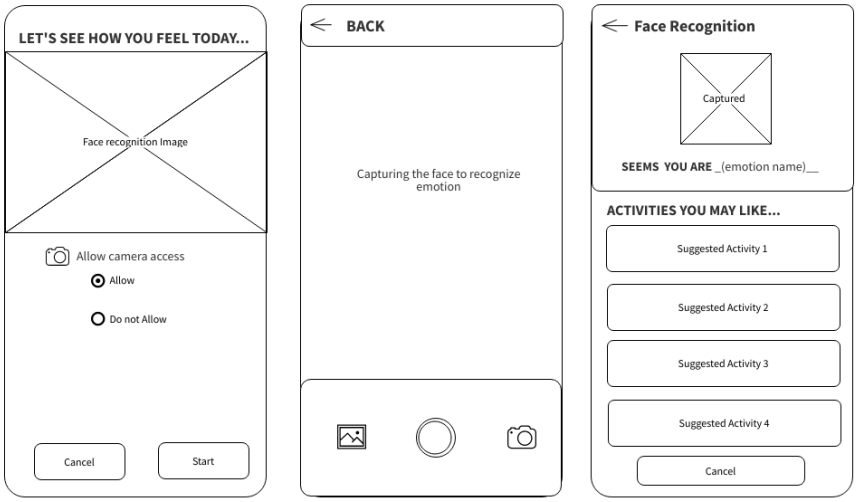
```python
from flask import Flask, request, render_template, jsonify
import numpy as np
from PIL import Image
from io import BytesIO
from tensorflow.keras.models import load_model

app = Flask(__name__)
model = load_model('face_model.h5')
class_map = {0: 'Angry', 1: 'Happy', 2: 'Neutral', 3: 'Sad', 4: 'Surprise'}

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    try:
        img_file = request.files['image']
        img = Image.open(img_file.stream).convert("RGB")  # Convert to RGB format
        img = img.resize((80, 80))  # Resize to the model's input size
        img_array = np.array(img)
        img_processed = np.expand_dims(img_array, axis=0)
        img_processed = img_processed / 255.0

        prediction = model.predict(img_processed)
        emotion_label = class_map[np.argmax(prediction)]

        return jsonify({'emotion': emotion_label})

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

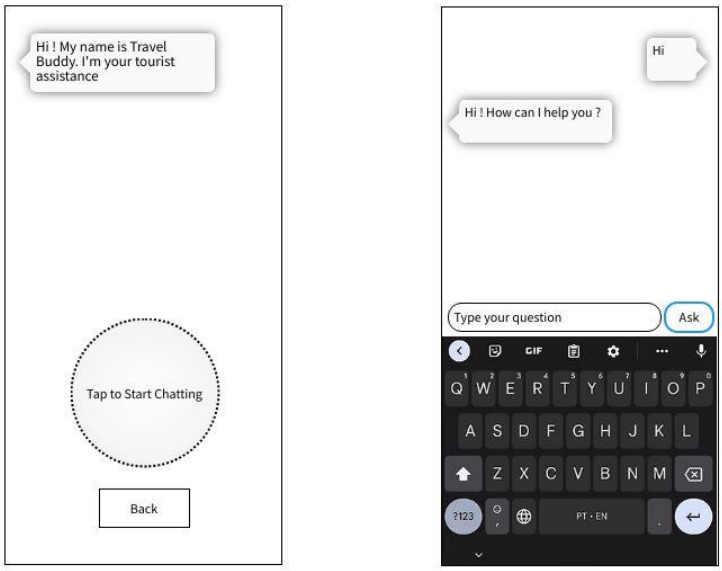


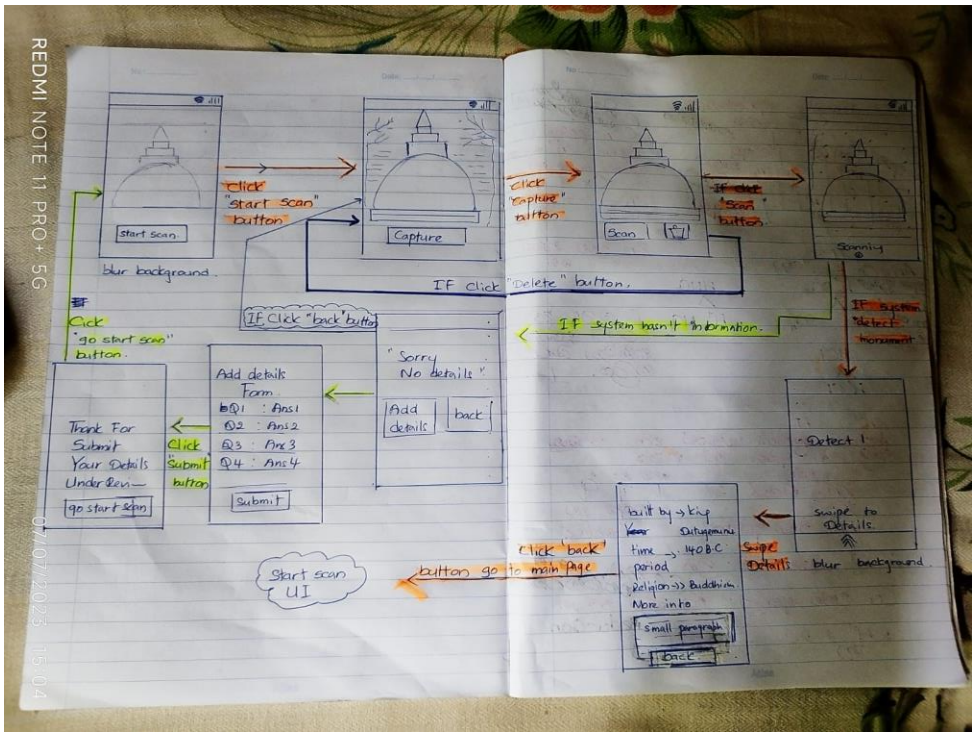
```python
# Define classes
class_map = training_set.class_indices
classes = list(class_map.keys())

# Define the folder containing image and text files
data_folder = r'C:\Users\Thisara\Desktop\RP_TEST\Smart_Travel_Test\Smart_Travel_Test\tourist_places'

upload_folder = r'C:\Users\Thisara\Desktop\RP_TEST\uploads'
os.makedirs(upload_folder, exist_ok=True)

# Load the Excel file
excel_file_path = r'C:\Users\Thisara\Desktop\RP_TEST\Smart_Travel_Test\Smart_Travel_Test\tourist_location\location_details.xlsx'  # Update the path
df = pd.read_excel(excel_file_path)

# ... (previous code remains unchanged)

@app.route('/predict/detectlocation', methods=['POST'])
def predict():
    try:
        # Check if an image file is included in the request
        if 'image' not in request.files:
            return jsonify({'error': 'No image file provided'})

        image_file = request.files['image']

        # Check if the file has a valid extension
        if image_file and allowed_file(image_file.filename):
            filename = secure_filename(image_file.filename)
```

Appendix D: UI Wireframe

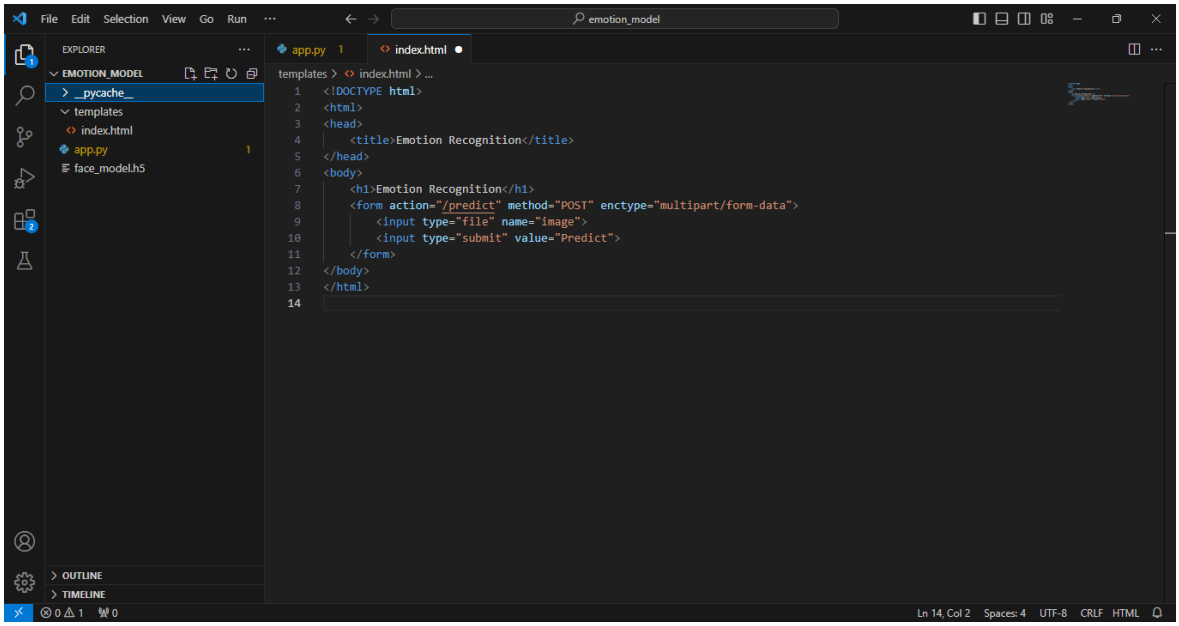


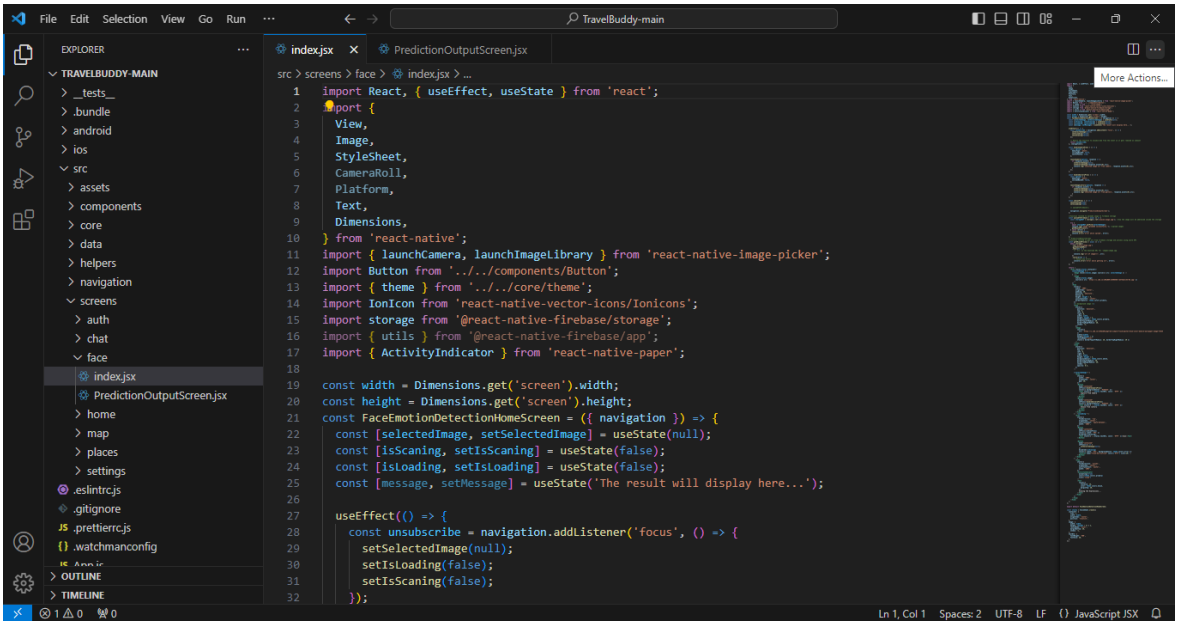


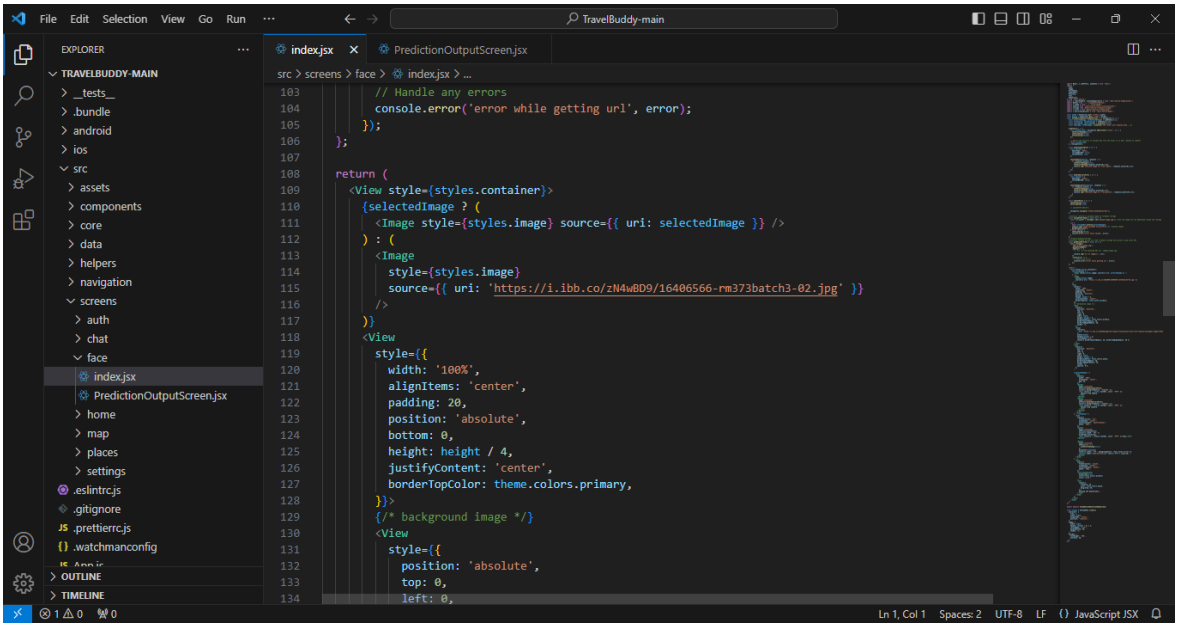


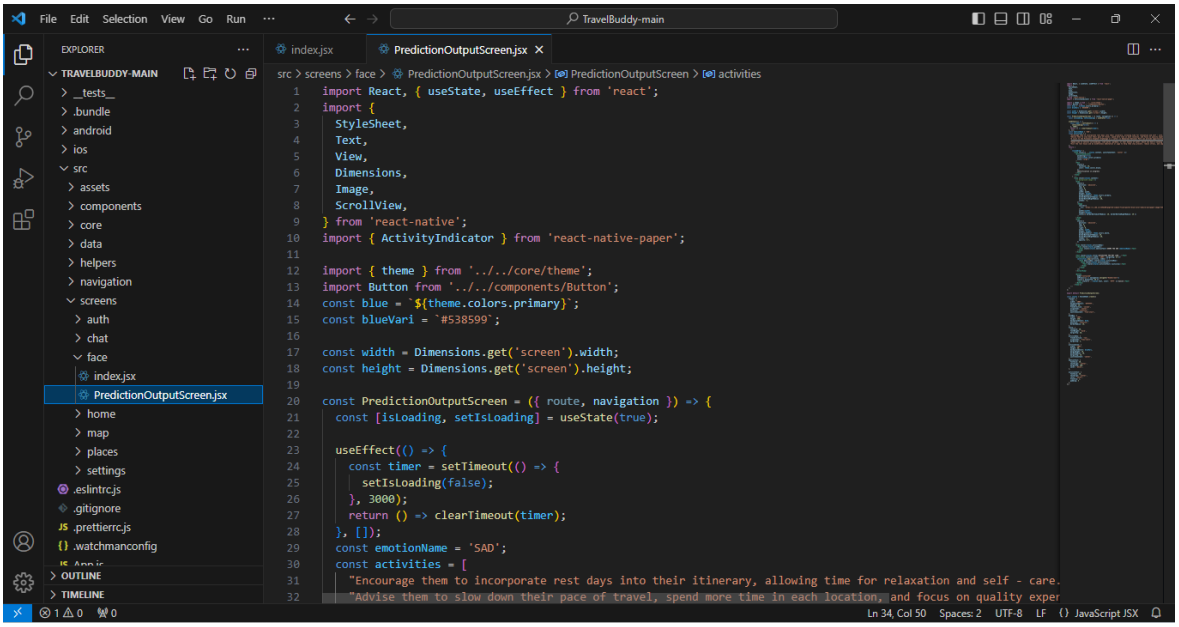


Appendix E: Frontend Implementation

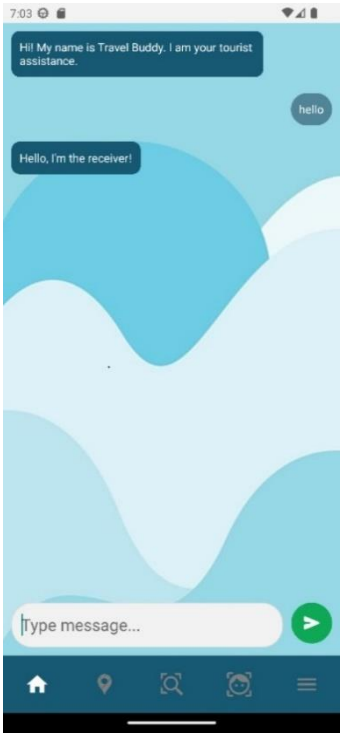



File Edit Selection View Go Run
TravelBuddy-main
EXPLORER
TRAVELBUDDY-MAIN
index.jsx
PredictionOutputScreen.jsx
src > screens > face > index.jsx > ...
More Actions...
1 import React, { useEffect, useState } from 'react';
2 import {
3 View,
4 Image,
5 StyleSheet,
6 CameraRoll,
7 Platform,
8 Text,
9 Dimensions,
10 } from 'react-native';
11 import { launchCamera, launchImageLibrary } from 'react-native-image-picker';
12 import Button from '../../components/Button';
13 import { theme } from '../../core/theme';
14 import IonIcon from 'react-native-vector-icons/Ionicons';
15 import storage from '@react-native-firebase/storage';
16 import { utils } from '@react-native-firebase/app';
17 import { ActivityIndicator } from 'react-native-paper';
18
19 const width = Dimensions.get('screen').width;
20 const height = Dimensions.get('screen').height;
21 const FaceEmotionDetectionHomeScreen = ({ navigation }) => {
22 const [selectedImage, setSelectedImage] = useState(null);
23 const [isScaning, setIsScaning] = useState(false);
24 const [isLoading, setIsLoading] = useState(false);
25 const [message, setMessage] = useState('The result will display here...');
26
27 useEffect(() => {
28 const unsubscribe = navigation.addListener('focus', () => {
29 setSelectedImage(null);
30 setIsLoading(false);
31 setIsScaning(false);
32 });
OUTLINE
TIMELINE
Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript JSX


File Edit Selection View Go Run
TravelBuddy-main
EXPLORER
TRAVELBUDDY-MAIN
index.jsx
PredictionOutputScreen.jsx
src > screens > face > index.jsx > ...
103 // Handle any errors
104 console.error('error while getting url', error);
105 });
106 };
107
108 return (
109 <View style={styles.container}>
110 {selectedImage ? (
111 <Image style={styles.image} source={{ uri: selectedImage }} />
112) : (
113 <Image
114 style={styles.image}
115 source={{ uri: 'https://i.ibb.co/zN4wBD9/16406566-rm373batch3-02.jpg' }}
116 />
117)}
118 <View
119 style={{
120 width: '100%',
121 alignItems: 'center',
122 padding: 20,
123 position: 'absolute',
124 bottom: 0,
125 height: height / 4,
126 justifyContent: 'center',
127 borderTopColor: theme.colors.primary,
128 }}>
129 {/* background image */}
130 <View
131 style={{
132 position: 'absolute',
133 top: 0,
134 left: 0,
OUTLINE
TIMELINE
Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript JSX



Appendix F: Result

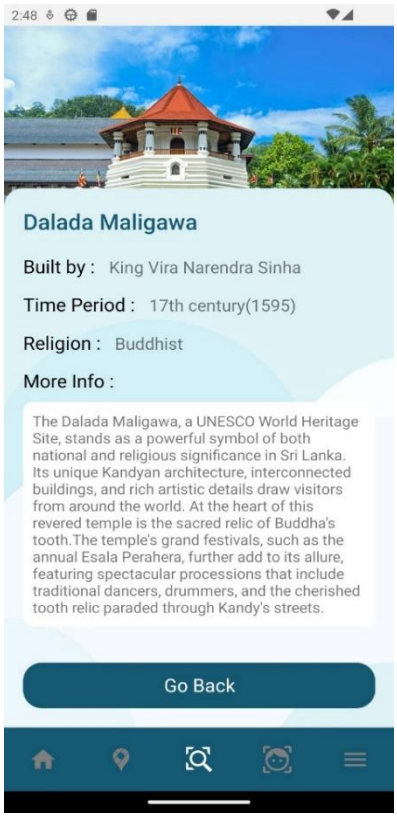


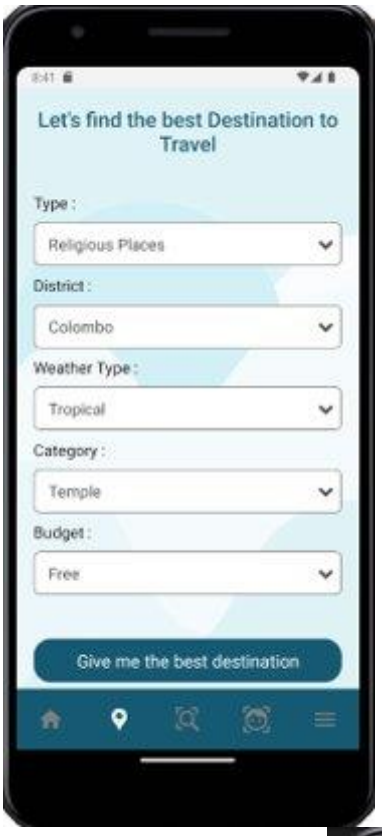

Capture From Camera
Choose from library


SEEMS YOU ARE HAPPY
ACTIVITIES YOU MAY LIKE...
Try Local Transport like traveling on local buses, trains or tuk-tuks to
If there's a local holiday or celebration during their visit, advise them to
Recommend eco-friendly activities like wildlife safaris, birdwatching, or
Cancel


2:39
LTE
Take a photo
Choose from library


9:32
Scan
Error uploading image: AxiosError: Request ...


2:48
Dalada Maligawa
Built by : King Vira Narendra Sinha
Time Period : 17th century(1595)
Religion : Buddhist
More Info :
The Dalada Maligawa, a UNESCO World Heritage Site, stands as a powerful symbol of both national and religious significance in Sri Lanka. Its unique Kandyan architecture, interconnected buildings, and rich artistic details draw visitors from around the world. At the heart of this revered temple is the sacred relic of Buddha's tooth.The temple's grand festivals, such as the annual Esala Perahera, further add to its allure, featuring spectacular processions that include traditional dancers, drummers, and the cherished tooth relic paraded through Kandy's streets.
Go Back


Let's find the best Destination to Travel
Type :
Religious Places
District :
Colombo
Weather Type :
Tropical
Category :
Temple
Budget :
Free
Give me the best destination


Name: Gangaramaya Temple
Religious Places
Colombo
View Location


Sri Lanka
Colombo