

A Project Report on

Implementing AI Based Comprehensive Web Framework for Tourism

Submitted in partial fulfillment of the requirements for the award
of the degree of

Bachelor of Engineering

in

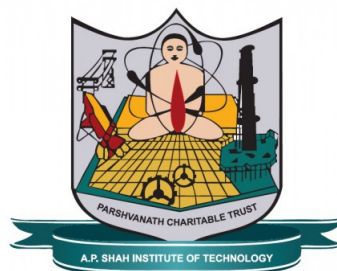
Information Technology

by

Nada Rajguru(19204005)
Jaynam Shah(18104047)
Harsh Shah(18104072)

Under the Guidance of

Prof. Anagha Aher
Prof. Nahid Shaikh



Department of Information Technology
NBA Accredited

A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI

Academic Year 2021-2022

Approval Sheet

This Project Report entitled “***Implementing AI Based Comprehensive Web Framework for Tourism*** ” Submitted by “***Nada Rajguru***”(19204005), “***Jaynam Shah***”(18104047), “***Harsh Shah***”(18104072) is approved for the partial fulfillment of the requirement for the award of the degree of ***Bachelor of Engineering*** in ***Information Technology*** from ***University of Mumbai***.

Prof. Nahid Shaikh
Co.Guide

Prof. Anagha Aher
Guide

Prof. Kiran Deshpande
Head Department of Information Technology

Place:A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project entitled “*Implementing AI Based Comprehensive Web Framework for Tourism* ” submitted by “*Nada Rajguru*”(19204005), “*Jaynam Shah*”(18104047), “*Harsh Shah*”(18104072) for the partial fulfillment of the requirement for award of a degree *Bachelor of Engineering* in *Information Technology*, to the University of Mumbai, is a bonafide work carried out during academic year 2021-2022.

Prof. Nahid Shaikh
Co.Guide

Prof. Anagha Aher
Guide

Prof. Kiran Deshpande
Head Department of Information Technology

Dr. Uttam D.Kolekar
Principal

External Examiner(s)

1.

2.

Place: A.P. Shah Institute of Technology, Thane

Date:

Acknowledgement

We have great pleasure in presenting the report on **Implementing AI Based Comprehensive Web Framework for Tourism**. We take this opportunity to express our sincere thanks towards our guide **Prof. Anagha Aher and Prof. Nahid Shaikh** Department of IT, APSIT Thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Prof. Kiran B. Deshpande** Head of Department, IT, APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We thank **Prof. Vishal S. Badgujar** BE project co-ordinator, Department of IT, APSIT for being encouraging throughout the course and for guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Student Name1: Nada Rajguru
Student ID1: 19204005

Student Name2: Jaynam Shah
Student ID2: 18104047

Student Name3: Harsh Shah
Student ID3: 18104072

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Nada Rajguru 19204005)

(Jaynam Shah 18104047)

(Harsh Shah 18104072)

Date:

Abstract

In current societies, the movement of all business activities is for effective participation among the market contenders, in addition not just withstanding their physical presence but also challenging their virtual presence. The Smart Tour recommender is a web-based framework for facilitating tourists through tour planning. Unlike other similar web-based systems our system is unique as it streamlines all the processes required for travel planning making it easy and convenient to use. It provides a number of features like providing information about tourist attractions, recommending tours based on the user's interests, searching hotels or restaurants based on the user's budget, booking accommodation, and providing users with a personalized itinerary. It focuses on making e-tourism easier and convenient as more and more people use such travel websites to plan their trips.

Contents

1	Introduction	1
1.1	Technology Stack	1
1.1.1	Flask Dashboard.	1
1.1.2	JavaScript Object Notation (JSON).	1
1.1.3	TensorFlow using Python	2
1.1.4	Pandas, Numpy	2
1.1.5	Natural Language Toolkit	2
1.1.6	Google Positioning System (GPS).	2
1.1.7	MongoDB.	2
1.1.8	Google Maps API.	3
1.2	Collaborative Recommendation Filtering	3
1.2.1	Memory-based Collaborative Filtering.	4
1.2.2	User-based collaborative Filtering	4
1.2.3	Item-based collaborative Filtering	4
1.2.4	Model-based collaborative Filtering.	4
1.3	Problem Definition	5
1.4	Obstacles	6
2	Literature Review	7
2.1	Objectives	8
3	Project Design	9
3.1	Proposed System	9
3.2	UML Diagrams	10
3.2.1	Use Case Diagram	10
3.2.2	Activity Diagrams	12
3.2.3	Sequence Diagram	13
3.2.4	Class Diagram	14
4	Project Implementation	16
4.1	Application Implementation	16
4.1.1	Hotel Booking	16
4.1.2	Flight Booking	17
4.1.3	Model for hotel recommendation	18
4.1.4	AI chatbot	21

5	Testing	24
5.1	Functional Testing	25
5.1.1	Unit Testing	25
5.1.2	Integration Testing	26
5.2	Non Functional Testing	26
5.2.1	Compatibility Testing	26
5.3	Test Cases	27
6	Result	28
6.1	Login Process	28
6.2	Signup Process	29
6.3	AI Based Chatbot	30
6.4	Hotel Booking	31
6.5	Flight Booking	32
6.6	Application Performance	32
7	Conclusions and Future Scope	34
7.1	Conclusions	34
7.2	Future Scope	34
	Bibliography	34
	Appendices	37
	Appendix-A	37
	Appendix-B	37
	Appendix-C	37
	Appendix-D	38
	Publication	39

List of Figures

1.1	Collaborative filtering	4
1.2	Collaborative filtering technique (Isinkaye, Folajimi, and Ojokoh, 2015) . . .	5
3.1	Proposed System Architecture	10
3.2	Use Case Diagram	11
3.3	Activity Diagram for User	12
3.4	Activity Diagram for Tour Details	13
3.5	Sequence Diagram for Tour Details	14
3.6	Class Diagram for Tour Website	15
4.1	Hotel booking implementation	16
4.2	Flight booking	17
4.3	Model for hotel recommendation	18
4.4	Model for hotel recommendation	19
4.5	Model for hotel recommendation	20
4.6	Intents for Chatbot	21
4.7	Training of Chatbot	22
4.8	Model for Chatbot	23
5.1	Test Cases	27
6.1	Login Screen UI	28
6.2	Register Screen UI	29
6.3	Chatbot UI	30
6.4	Hotel Search	31
6.5	Hotel Search Output	31
6.6	Flight Booking	32
6.7	Application Performance	33

List of Abbreviations

AI:	Artificial intelligence
ML:	Machine Learning
UML:	Unified Modeling Language
JSON:	JavaScript Object Notation
CRUD:	Create, Read, Update, Delete
NLTK:	Natural Language Toolkit

Chapter 1

Introduction

The focus of this research is to create an AI-based tour recommender system. The recommendation framework is a web-based system that may deliver a customized rundown of vacation sites, eateries, and accommodations relying upon the traveler's inclinations. Conventional recommendation techniques like content-based filtering and synergistic sifting are known to be advantageous in the tourism industry. Moreover, in light of the information gained from user's interests and inclinations, recommendations are made utilizing content-based suggestions. Data dependent on comparable profiles is suggested when the client utilizes the service more using community-oriented recommendation. Furthermore, a design of a chat box is provided which gives a genuine and precise response for any question using Artificial Intelligence, which permits individuals to textually communicate with the objective of organizing visits and requesting fascinating spots worth visiting. The Flask development framework is utilized to construct the system, since it accentuates rapid development and simple, pragmatic design.

The system methodology is discussed comprehensively with the aim of explaining the functionalities of the system.

1.1 Technology Stack

1.1.1 Flask Dashboard.

We introduce, Flask Dashboard, a Python library which is an extension for Flask-based Python applications functionalities that help developers to monitor the performance and utilization. The dashboard and the web application are created in python using Flask to bind the application's web services and adding extra routes to the service for making interaction quite efficient.

1.1.2 JavaScript Object Notation (JSON).

JSON is an appropriate information trading language since it is used for serializing and transferring structured data over a network for all cutting-edge languages. It is used to evaluate the results of all models and choose the best-performing model for chatbot on our dataset. It is used to pre-process and alter data before it is passed to the model. We need

to pass the natural language representation that we need to perform. JSON is the language utilized in this application for making the information base.

1.1.3 TensorFlow using Python

TensorFlow is an open-sourced software library for jobs that require numerical processing. As compared to other machine learning libraries, TensorFlow has better compile time. It is compatible with CPUs, GPUs, and distributed processing. Python API is simpler to use than C++ API.

1.1.4 Pandas, Numpy

Pandas is open-source Python package that is used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays. Pandas makes it simple to do many of the time consuming, repetitive tasks associated with working with data. NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

1.1.5 Natural Language Toolkit

The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP). It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets as well as accompanied by a cook book and a book which explains the principles behind the underlying language processing tasks that NLTK supports.

1.1.6 Google Positioning System (GPS).

The GPS in our system keeps track of the tourist location in real-time. The web application keeps on updating location as the tourist changes location. Using GPS, the system is location-aware. The GPS receiver sends position and time data to any device by utilizing navigation satellites, 24 of which are operational and three of which serve as backups [21]. Trilateration is then used to estimate the user's precise position which then is fed to the system for a further recommendation of a nearby location to visit.

1.1.7 MongoDB.

MongoDB is a JSON-formatted open-source document database that stores organized and unstructured data. Documents, which are groupings of fields with a dynamic design, are

used to hold all individual records. MongoDB is a more versatile and user-friendly database than SQL. In modern programming language, the document data model is a powerful way to store and retrieve data.

1.1.8 Google Maps API.

By utilizing Google Maps API, Maps can be added to any web application. The API helps to guide the system depending on Google Maps information. Google Maps servers handle all the reactions to the plan signal and to use the Google Maps API the developer has to register the project on the Google Developer Console and get a Google API key which can be used further on the site. Without the key Google Maps services cannot be accessed.

1.2 Collaborative Recommendation Filtering

A new filtering technique, collaborative Filtering, was introduced from the main disadvantage of the content-based filtering technique as they are highly dependent on the metadata or descriptions of the items or contents, which was completely independent of the domain and overcame the drawbacks of content-based Filtering. The collaborative filtering technique was the most popular filtering technique in which the recommendations are made upon the basis of past interactivity of the user or the basis of the interactions of the users of similar taste. In collaborative Filtering, a database, also called the user-item matrix, was set up to store the preferred contents. Then the users with similar tastes or interests are found by finding the similarities of their profile based on the likeliness and past preferences (Hdioud, Frikh, and Ouhbi, 2012). Likewise, the users with similar tastes in items or contents are gathered together to build a neighborhood cluster. Thus, a user can get recommendations based on the interests of another user in the same neighborhood.

The recommendations produced through the collaborative filtering techniques can be divided into two: predictions and recommendations. While rating predictions are the numerical values expressed as for a user and item, recommendations are the N list of items that the user can be interested in based on numerical values of the prediction ratings (Isinkaye, Folajimi, and Ojokoh, 2015). According to Nilashi et al., the collaborative filtering technique can be divided into memory-based and model-based collaborative Filtering.

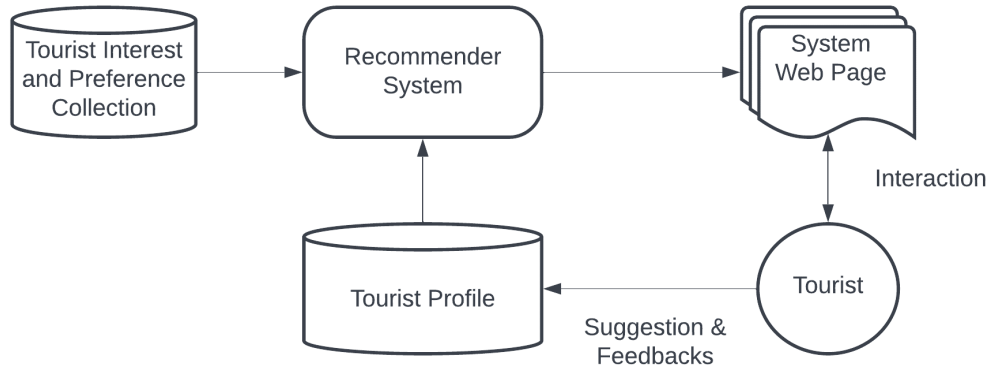


Figure 1.1: Collaborative filtering

1.2.1 Memory-based Collaborative Filtering.

In memory-based collaborative Filtering, ratings on a product or an item are crucial in producing recommendations. The neighbor with similar taste is found out after analyzing the ratings kept on memory. It is further categorized into user-based and item-based Collaborative Filtering.

1.2.2 User-based collaborative Filtering

Ratings provided by the users on the same item is analyzed primarily. Then these ratings are compared to find the users with similar interests. Then the weighted average of an item by similarly tasted users is computed to produce recommendations (Véras et al., 2015).

1.2.3 Item-based collaborative Filtering

For producing recommendations in this type of Filtering, the primary focus is on the item rather than looking for similarity among users. The ratings provided by the users are analyzed from the user-item matrix, and items similar to these are recommended (Véras et al., 2015).

1.2.4 Model-based collaborative Filtering.

In the model-based collaborative filtering technique, a prediction model is used to produce recommendations by analyzing the user-item matrix with the help of techniques such as data mining and machine learning. The recommendations produced upon these types are similar to neighborhood model predictions. The algorithms such as clustering, association rule, the artificial neural network are used to find similar items from the user-item matrix to produce the list of N recommendations according to the priority (Hdioud, Frikh, and Ouhbi, 2012).

Advantages of collaborative Filtering.

- content-based Filtering, it does not require much information regarding the content or item in the domain (Isinkaye, Folajimi, and Ojokoh, 2015).

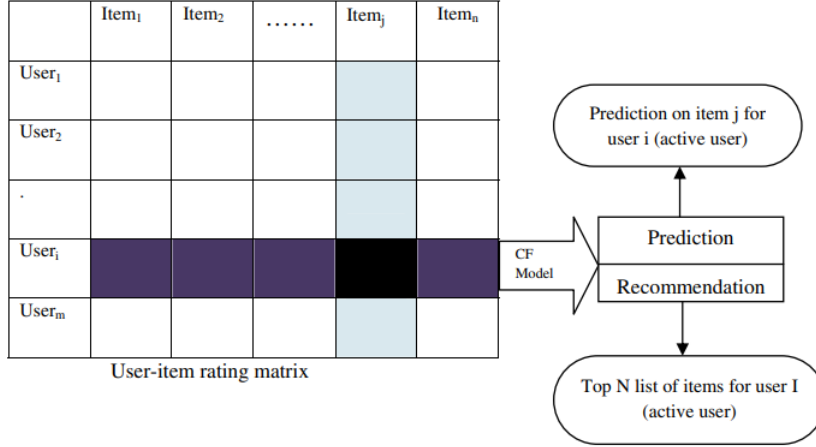


Figure 1.2: Collaborative filtering technique (Isinkaye, Folajimi, and Ojokoh, 2015)

- Collaborative filtering techniques can produce recommendations regarding the relevant item where the item does not need to be included in the user profile.
- Increased efficiency and precision for the recommendations as the users interact highly with the items (Nilashi et al., 2013).

Disadvantages of Collaborative Filtering.

- Cold start problem: When it comes to the case of a new user or a new item, the profile is left empty, and the recommender system will not have sufficient information to produce recommendations. This is the collaborative filtering technique (Isinkaye, Folajimi, and Ojokoh, 2015).
- Problems by Data sparsity: This can be defined as the generation of weak recommendations due to the development of a sparse user-item matrix. This problem arises because of the lack of information or ratings on the items provided by the user (Hdioud, Frikh, and Ouhbi, 2012).
- Scalability problems: As the number of data sets handled by the algorithm increases, the algorithms' computations also increase. Thus, the recommender system becomes unable to produce efficient recommendations.
- Tendency to be stereotyped: As the recommendations are produced upon the similarity on the user profiles, the chances of getting stereotyped recommendations are higher (Nilashi et al., 2013).

1.3 Problem Definition

There are many e-tourism websites that provide generalized tours to clients. Increasingly tourists are planning trips by themselves using the abundant information available on the internet, however they still expect and want trip plan advisory services. The vast number of tourists find it excessively challenging to do their own research by visiting various websites to get information to plan for the tour. The planning made by the users mostly can't be

accurate as the websites they refer to do not have a proper feedback system to analyze if data provided by the tourist websites or blogs is correct. Therefore, in order to avoid making mistakes users choose the mainstream mode of planning and booking a tour, which is done by physically approaching the travel agent or travel company and booking a tour. Usually booking with the tour agent or a company means a generalized tour plan and itinerary to fit all sorts of customers which might not be enjoyable or worth the user's interests. The travel companies have tie ups with accommodation and transport providers to cut costs, which might not be suitable for the user. Hence the user is not provided with choices which makes fun tours feel like field trips. Also, tourists have to visit multiple websites researching attraction sites, guides, restaurants, hotel reservations and in a nutshell planning the entire trip, which is not effective and very time consuming.

1.4 Obstacles

The origins of tour recommendations traced back to the Orienteering Problem and related variations, which differ in that they do not include customization for specific users. As a consequence, given the identical starting/ending POIs and time budget as inputs, the same tour route is recommended to all users [15]. Orienteering Problem. in which participants visit checkpoints with pre-determined scores that attempt to maximize their overall score within a certain time limit. Numerous research in recent years have used the OP and its many versions to predict tour suggestions. Similarly, several online applications [17, 18] have been built based on the OP. We begin by describing the original OP [19, 20] and how it has applied to the domain of tour recommendation.

Many tours recommendation concentrate on specific cities, each has its own set of POIs. A visitor visiting a specific city will have concerns of a specific time or distance budget, and have to choose starting and ending POIs. A tourist's budget often signifies the amount of time he or she is willing to spend on a tour or the distance that he or she is willing to travel.

Similarly, the starting and ending POIs indicate the tourist's preference to begin the trip near a specific spot (e.g., the tourist's hotel) and complete the tour at a different point (e.g., near a restaurant). Thus, given a set of POIs, a budget, a starting POI, and a destination, our major goal is to recommend a tour itinerary that maximizes a certain score while sticking to the budget, starting, and destination POI limitations.

Problem for Itinerary Mining . Based on the Orienteering Problem, [21] introduced the Itinerary Mining Problem (IMP), which tries to discover an itinerary that optimizes POI popularity while keeping touring time within a predetermined budget. They model POI Popularity based on the number of visits by unique visitors, transit times between POIs based on the median transit time by all tourists, and POI visit times based on visit time by all tourists. To solve the IMP, a recursive greedy algorithm [12] is implemented, which tries to estimate the middle node of the itinerary and the corresponding utility (popularity) gained and cost (time) spent, then recursively invokes itself on both portions of the itinerary.

Chapter 2

Literature Review

Recommender systems have typically been used in tourism applications to filter out irrelevant information and to provide personalized recommendations to the users. Barranco et al. has proposed a method to improve recommendations provided by recommendation systems by using context-aware filtering by providing the user's with smart and personalized point of interest's taking into account their current location[1].

Pazzani and Billsus suggests on using a profile of the user's interests which is used by most recommendation systems that are mainly based model of the user's preferences, and a history of the user's interactions with the recommendation system[3]. which is also suggested by M.D. Choudhary et al.[7].

Also to generate personalized tourist routes using the list of point of interest's generated by the recommendation functionality as well as other tourist-related data as proposed by Gavalas et al.[10].

In relation to the tourism industry and recommendation, the context can be defined as the characteristic information of an entity such as users or an object. The contextual information can be helpful to personalized recommendations when the available information about the item or person is not sufficient.

Context-aware recommender systems (CARS) generate more relevant recommendations by adapting them to the specific contextual situation of the user how context can be defined and used in recommender systems in order to create more intelligent and useful recommendations which is studied by Adomavicius and Tuzhilin. [2]; this method is implemented by Jafri et al from providing improved recommendations in their system[5].

Meteren and Someren also suggests using Content-Based Filtering for recommendation based on training data a user model is induced that enables the filtering system to classify unseen items into different classes to recommend[8].

The study and evaluation done by Joshua Stomberg proposes application of different models for clustering and classification or use of hybrid models for better recommendation performance and lower rating error[11].

Shafiee et al. describes how collaborative filtering (CF), a recommendation technique which provides its predictions and recommendations on the ratings or behavior of other users in the system can be used for such travel/tourist or data intensive systems to provide users with better results[6].

M. Ekstrand et al. propose the application of user-user based collaborative filtering, item-

item collaborative filtering, hybrid recommendation technique including demographic, content-based recommendations, preference-based filtering to travel and tourism service providers[9].

K. H. Lim et al. highlighted the key differences between tour itinerary recommendation and the related areas of Operations Research, next-location prediction, top-k location recommendation and travel package/region recommendation. They have developed a classification to describe general tourism-related research and discussed various evaluation methodologies and observed a trend for tour recommendations moving towards context-aware approaches as they can provide better tour recommendations[12].

2.1 Objectives

- To develop an extensive Web portal that provides tourists/clients a tour itinerary furthermore displays hotel recommendations.
- To develop a system that generates a personalized tour itinerary, formulated on varying aspects of the client's budget, interests and time constraints.
- To design a recommendation system that will suggest tours and hotels.
- To allow users to make changes in the generated tour itineraries by providing an option for customization.
- To implement an AI chat bot for customer support and solve basic queries.
- To provide a forum where customers can communicate with travel agents/support groups to solve complicated queries or doubts.

Chapter 3

Project Design

3.1 Proposed System

The proposed system is represented through a flowchart below introducing the steps involved to generate a personalized tour plan. Users can make use of the website's features by checking in to the system after browsing the site. Users who have registered/signed up for the website are the only ones who can log in. New users can sign up for an account on the website and create a username, password, and other credentials, which are then recorded in the system database. After registering, a user may log in to the website using the username and password they used to sign up. The login is successful when the login information match those maintained in the database, and the user continues to the next page. After successfully logging in, the process of locating the optimal attraction location for the user begins with the user providing tour information. This procedure works by taking into account the interests and preferences of the users. The category of tourist attraction site that the tourist chooses to visit, such as interior sites, outdoor activities, architectural heritage sites, environmental regions, and so on, is collected from users. The user must choose between two and three different categories of favorite interests. The Recommendation system uses AI/ML to provide recommendations based on the data (interests and preferences) provided by the user. The recommendation system's two main objectives are to provide users with generic travel plans and lodging options based on their preferences. The user is given a generic plan to which they may apply various filters to design the components of the tour that are most appropriate for them.

Filters by gross tour budget, area of stay, type of lodging, length of stay, hotel type, culinary preferences, and so forth. Following that, the customer may make a hotel reservation after deciding on a budget, length of stay, quality of stay, service, and food preference. It allows users to personalize their plans and is appropriate for people of all ages and interests. The consumers are given a wide variety of options that may be modified from one individual to the next. By selecting the cuisine that the user likes, the user may pick a restaurant with convenience and speed. Next, consumers may narrow down the best restaurants for them by picking a restaurant category, such as fast food, vegetarian, non-vegetarian, Jain, and so on. Users may also pick a financial range to spend at the restaurant, as well as the sort of restaurant they want to visit, such as Fine Dining, Casual Dining, Buffet, Café, Fast Casual, and so on. The processes for booking a hotel room are identical to those for booking a restaurant. To begin, customers can choose their desired budget for lodging. The user then chooses the sort of accommodation they want to stay in, such as Single, Double, Deluxe,

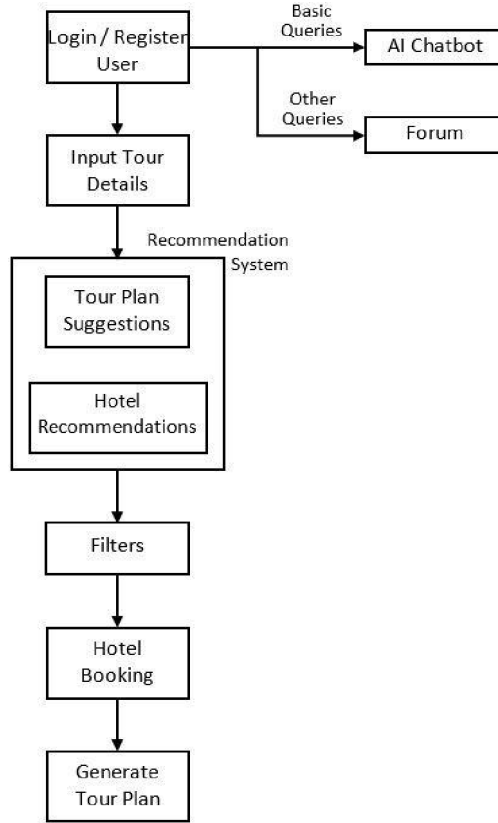


Figure 3.1: Proposed System Architecture

Twin, or Suite. The customer must then choose depending on availability, budget, luxury, cleanliness, pool, and laundry facilities, among other factors. Following the customer is given with a one-of-a-kind and tailored trip plan, complete with tourist information, nearby eateries, and an itinerary. The proposed system is represented through a flowchart below introducing the steps involved to generate a personalized tour plan.

3.2 UML Diagrams

3.2.1 Use Case Diagram

The purpose of a use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and State chart also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analysed to gather its functionalities, use cases are prepared and actors are identified. When the initial task is complete, use case diagrams are modelled to present the outside view. In brief, the purposes of use case diagrams can be said to be as follows

- Used to gather the requirements of a system.
- Used to get an outside view of a system.

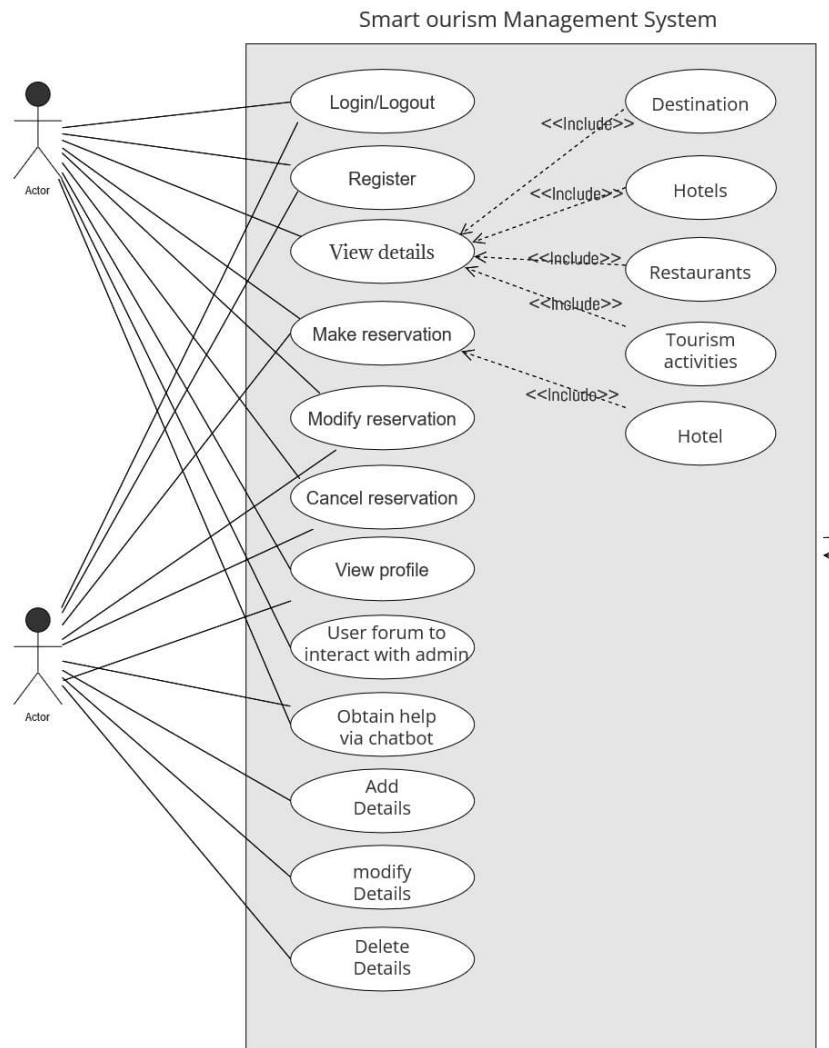


Figure 3.2: Use Case Diagram

- Identify the external and internal factors influencing the system.
 - Show the interaction among the requirements are actors.
- Use case diagrams can be used for –
- Requirement analysis and high-level design.
 - Model the context of a system.
 - Reverse engineering.
 - Forward engineering.

This Use Case Diagram is a graphic depiction of the interactions among the site users, admin with the Tourism Management System. It represents the methodology used in system analysis to identify, clarify, and organize system requirements of the System.

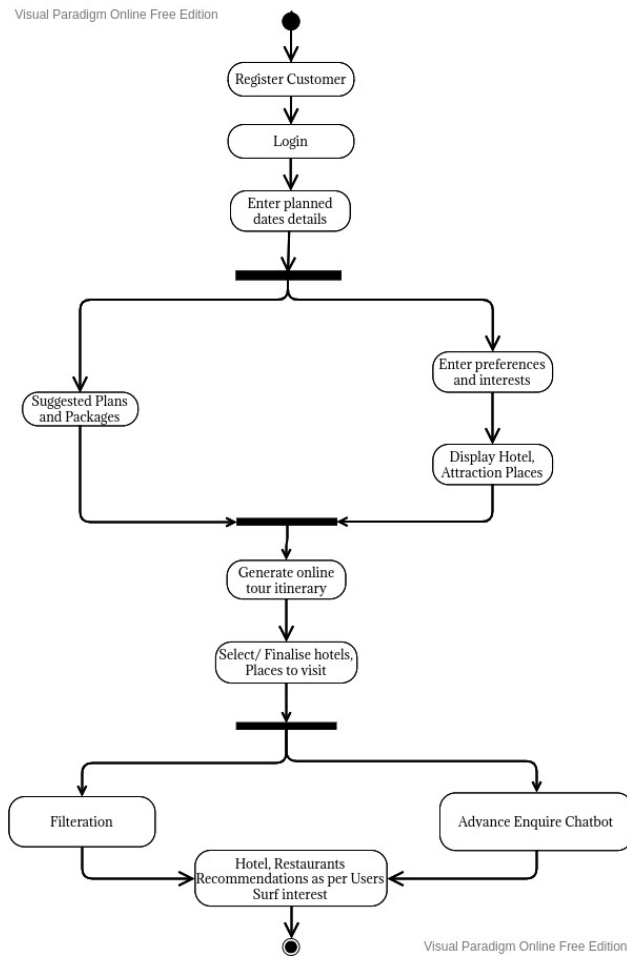


Figure 3.3: Activity Diagram for User

3.2.2 Activity Diagrams

This is an Activity UML diagram that depicts the activity flows of a Traveler (Customer), a Package, and a System. The following are the primary activity aspects depicted in the Activity diagram:

- Users (Travelers) may login, register, view existing packages, add interests and preferences, update and personalize tours, and so on.
- It depicts the traveler's activity flow from the time he or she first uses the system until the end of the journey.
- Throughout the journey, users will be able to personalize their tour itinerary (filtration).
- Users can use the advance inquiry chatbot to ask inquiries.

The activity features involved in this Activity diagram are detailed flow that shows the activity flow of generated tour itinerary from the display for tour details, updating of tours, adding tours.

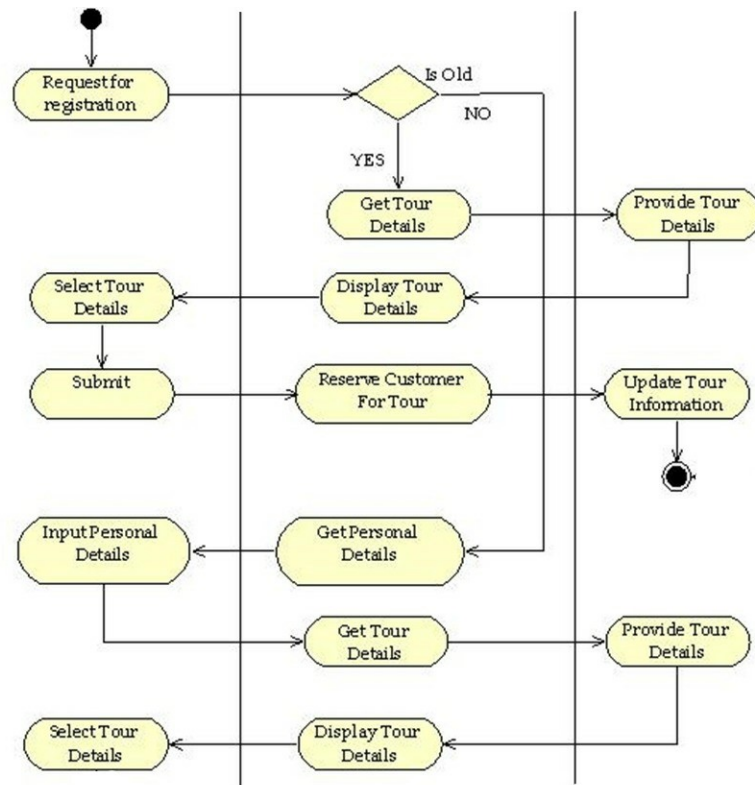


Figure 3.4: Activity Diagram for Tour Details

3.2.3 Sequence Diagram

A sequence diagram is the most commonly used interaction diagram. Interaction diagram – An interaction diagram is used to show the interactive behavior of a system. Since visualizing the interactions in a system can be a cumbersome task, we use different types of interaction diagrams to capture various features and aspects of interaction in a system. Sequence Diagrams – A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. User will be able to login into their account using their credentials. After login they need to enter some necessary details and book hotel according to their preferences and book flight tickets. User will be able to see the most recommendable places to visit. The diagram below help demonstrate the flow of tourism interact over of the course of the sequence. The main Purpose of a Sequence Diagram are

- To model high-level interaction among active objects within a system.
- To model interaction among objects inside a collaboration realizing a use case.
- It either models generic interactions or some certain instances of interaction.

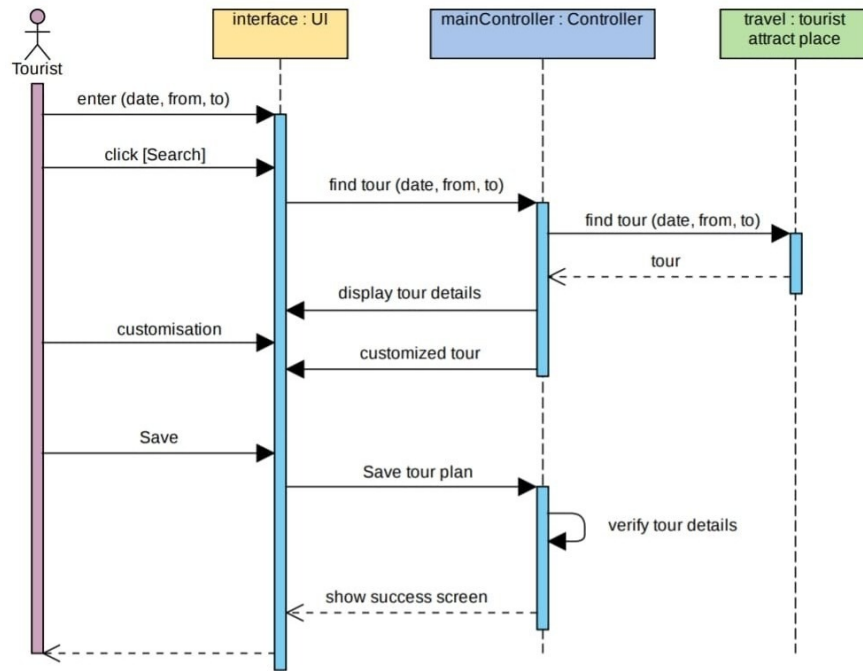


Figure 3.5: Sequence Diagram for Tour Details

3.2.4 Class Diagram

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of object-oriented modelling. It is used for general conceptual modelling of the structure of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

In the diagram, classes are represented with boxes that contain three compartments:

- The top compartment contains the name of the class. It is printed in bold and centred, and the first letter is capitalized.
- The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.
- The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase

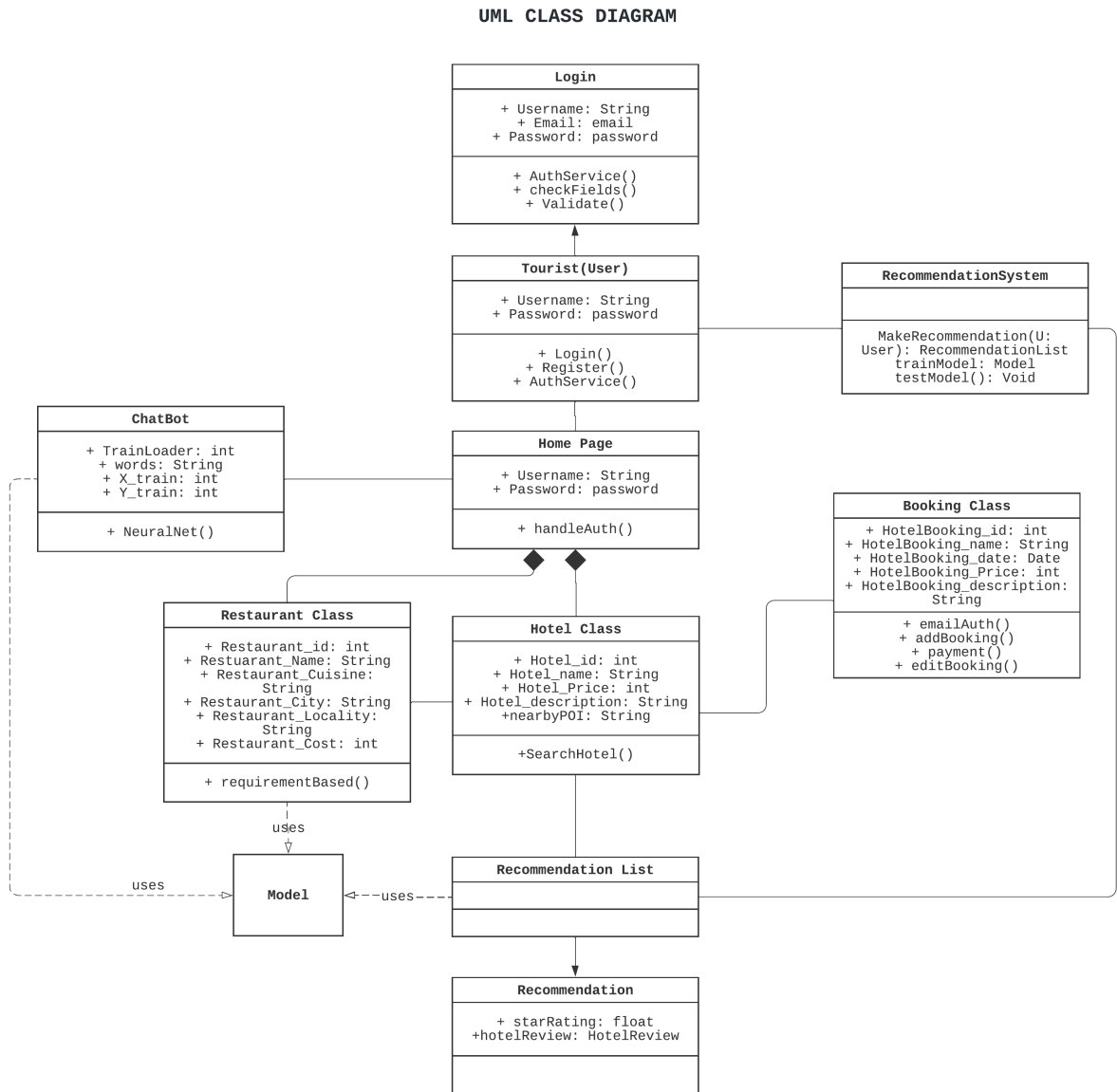


Figure 3.6: Class Diagram for Tour Website

Chapter 4

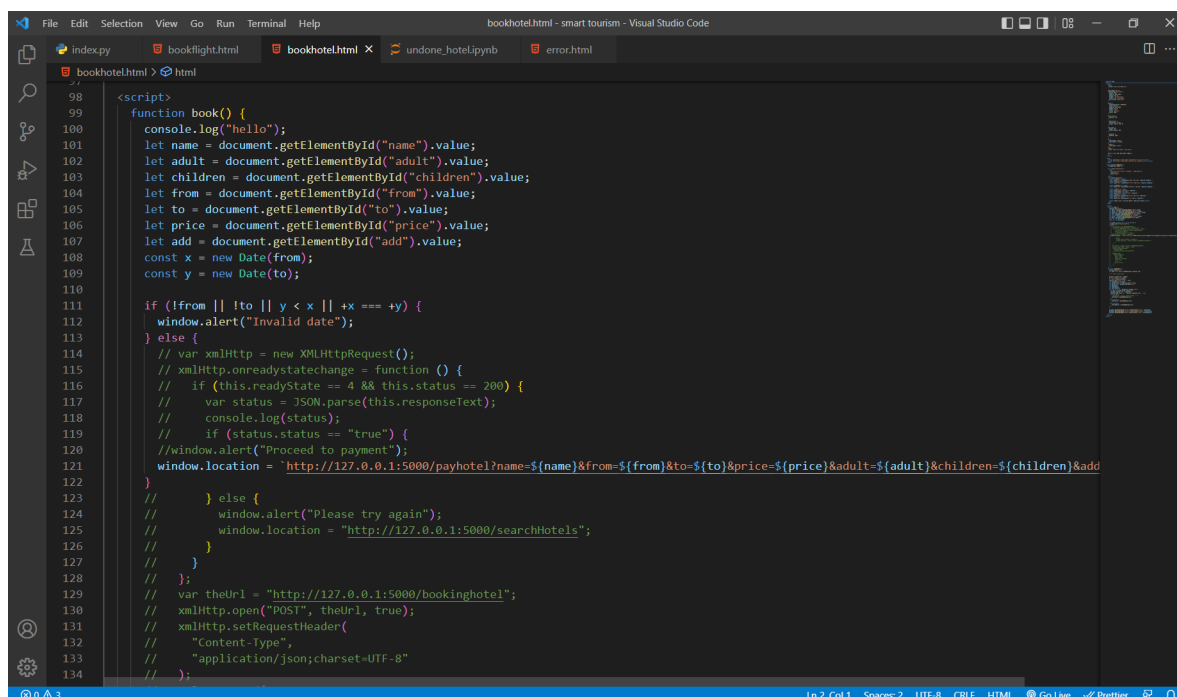
Project Implementation

This chapter contains the snapshots of the code written to implement different modules of the system. It starts with a screenshot of code block which handles booking an hotel and then booking flight and then a model for the hotel recommendation and another screenshot for AI Chatbot.

4.1 Application Implementation

4.1.1 Hotel Booking

The function book() is to provide hotel booking; it takes in the different parameters (customer information) from the user that is required, validates it and confirms the booking.

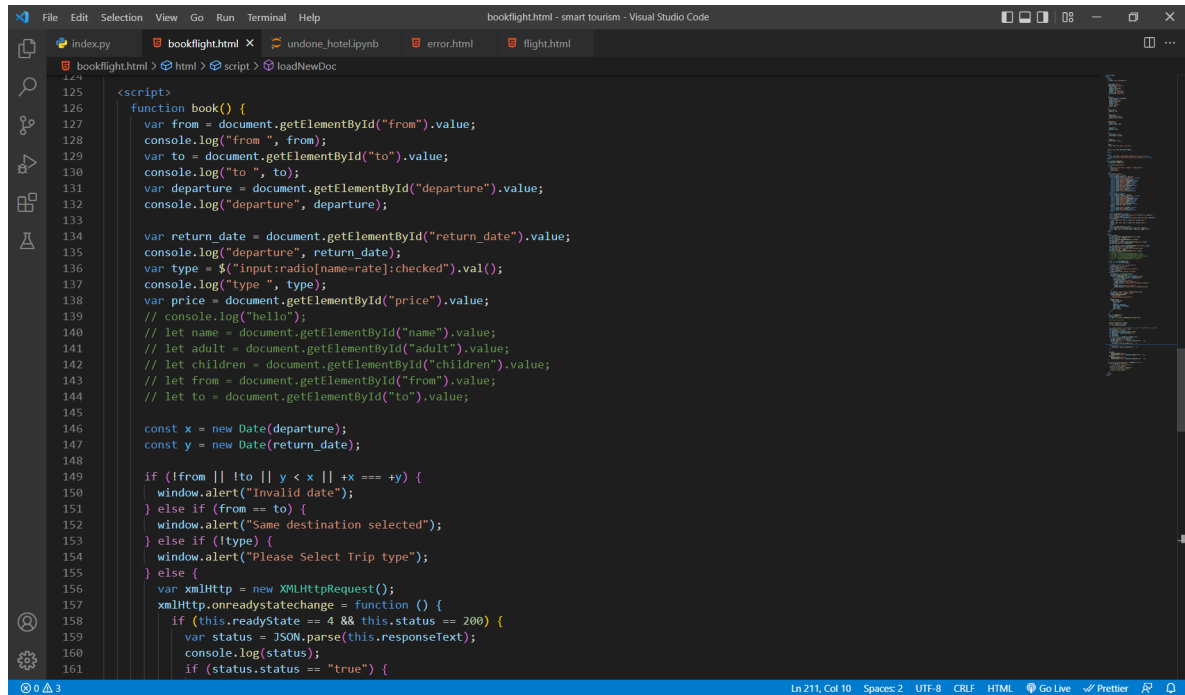


```
98 <script>
99 function book() {
100   console.log("hello");
101   let name = document.getElementById("name").value;
102   let adult = document.getElementById("adult").value;
103   let children = document.getElementById("children").value;
104   let from = document.getElementById("from").value;
105   let to = document.getElementById("to").value;
106   let price = document.getElementById("price").value;
107   let add = document.getElementById("add").value;
108   const x = new Date(from);
109   const y = new Date(to);
110
111   if (!from || !to || y < x || +x === +y) {
112     window.alert("Invalid date");
113   } else {
114     // var xmlhttp = new XMLHttpRequest();
115     // xmlhttp.onreadystatechange = function () {
116     //   if (this.readyState == 4 && this.status == 200) {
117     //     var status = JSON.parse(this.responseText);
118     //     console.log(status);
119     //     if (status.status == "true") {
120     //       window.alert("Proceed to payment");
121     //       window.location = "http://127.0.0.1:5000/payhotel?name=${name}&from=${from}&to=${to}&price=${price}&adult=${adult}&children=${children}&add";
122     //     } else {
123     //       window.alert("Please try again");
124     //       window.location = "http://127.0.0.1:5000/searchHotels";
125     //     }
126     //   }
127     // };
128     // var theUrl = "http://127.0.0.1:5000/bookinghotel";
129     // xmlhttp.open("POST", theUrl, true);
130     // xmlhttp.setRequestHeader(
131     //   "Content-Type",
132     //   "application/json; charset=UTF-8"
133     // );
134   }
```

Figure 4.1: Hotel booking implementation

4.1.2 Flight Booking

The book function for flights takes in the required parameters from the user such as flight departure location, destination, departure date and/or return date, and validates the dates and then displays flights for the user; then the user can proceed with completing the booking for selected flight.



```
125
126
127 <script>
128   function book() {
129     var from = document.getElementById("from").value;
130     console.log("from ", from);
131     var to = document.getElementById("to").value;
132     console.log("to ", to);
133     var departure = document.getElementById("departure").value;
134     console.log("departure", departure);
135
136     var return_date = document.getElementById("return_date").value;
137     console.log("departure", return_date);
138     var type = $("input:radio[name=rate]:checked").val();
139     console.log("type ", type);
140     var price = document.getElementById("price").value;
141     // console.log("hello");
142     // let name = document.getElementById("name").value;
143     // let adult = document.getElementById("adult").value;
144     // let children = document.getElementById("children").value;
145     // let from = document.getElementById("from").value;
146     // let to = document.getElementById("to").value;
147
148     const x = new Date(departure);
149     const y = new Date(return_date);
150
151     if (!from || !to || y < x || +x === +y) {
152       window.alert("Invalid date");
153     } else if (from == to) {
154       window.alert("Same destination selected");
155     } else if (!type) {
156       window.alert("Please Select Trip type");
157     } else {
158       var xmlhttp = new XMLHttpRequest();
159       xmlhttp.onreadystatechange = function () {
160         if (this.readyState == 4 && this.status == 200) {
161           var status = JSON.parse(this.responseText);
162           console.log(status);
163           if (status.status == "true") {
```

Figure 4.2: Flight booking

4.1.3 Model for hotel recommendation

This function integrates all the methods in one single module. It is the most critical part of the code. The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP). .

```
predict.py > getAdditionalHotels
1 from nltk.corpus import wordnet
2 from nltk.stem.wordnet import WordNetLemmatizer
3 from nltk.stem.snowball import SnowballStemmer
4 from nltk.tokenize import word_tokenize
5 import numpy as np
6 import pandas as pd
7 import nltk
8 from nltk.corpus import stopwords
9 from nltk.stem import WordNetLemmatizer
10 import json
11
12 df = pd.read_csv('book.csv')
13
14 hotel = df.drop(columns=['additional_info', 'crawl_date', 'hotel_brand', 'hotel_category',
15                        'hotel_description', 'image_count', 'latitude', 'locality', 'longitude', 'pageurl', 'property_id',
16                        'province', 'qts', 'query_time_stamp', 'review_count_by_category',
17                        'room_area', 'room_facilities',
18                        'similar_hotel', 'site_review_count',
19                        'site_stay_review_rating', 'sitename', 'state', 'uniq_id'])
20
21 #hotel = hotel_data1.dropna()
22
23
24 def citybased(city):
25     hotel['city'] = hotel['city'].str.lower()
26     citybase = hotel[hotel['city'] == city.lower()]
27     citybase = citybase.sort_values(by='hotel_star_rating', ascending=False)
28     if(citybase.empty == 0):
29         hname = citybase[['property_name', 'hotel_star_rating',
30                        'address', 'hotel_facilities', 'property_type']]
31         print(hname.head())
32         return hname.head()
33     else:
34         print('No Hotels Available')
35
36
37 def getAdditionalHotels(names):
38     hotel['property_name'] = hotel['property_name'].str.lower()
```

Figure 4.3: Model for hotel recommendation

The data required such as city, amenities and ratings is taken as input from the search hotels html file and same data will be used as the input for our function for hotel search. Using the NLP in the function we are finding and matching the similar words here from the data set. After finding the similar words from the data set the cosine similarities function is used to find the distance and according to the distance from the cosine similarities we recommend the respective hotels. Hence we are able to find the hotels here based on our requirements.

```
def getAdditionalHotels(names):
    hotel['property_name'] = hotel['property_name'].str.lower()
    data = []
    for i in names:
        hotelbase = hotel[hotel['property_name'] == i.lower()]
        if(hotelbase['city'].any()):
            # print("---", hotelbase['city'].item())
            data.append(json.loads(
                citybased(hotelbase['city'].item()).to_json()))

    # print(data)
    return data

# getAdditionalHotels(['Taj hotel', 'Taj Plaza', 'Ashiyana'])

def requirementbased(city, rating, features):
    hotel['city'] = hotel['city'].str.lower()
    hotel['hotel_facilities'] = hotel['hotel_facilities'].str.lower()
    features = features.lower()
    features_tokens = word_tokenize(features)
    sw = stopwords.words('english')
    lemm = WordNetLemmatizer()
    f1_set = {w for w in features_tokens if not w in sw}
    f_set = set()
    for se in f1_set:
        f_set.add(lemm.lemmatize(se))
    reqbased = hotel[hotel['city'] == city.lower()]
    reqbased = reqbased[reqbased['hotel_star_rating'] == rating]
    reqbased = reqbased.set_index(np.arange(reqbased.shape[0]))
    l1 = []
    l2 = []
    cos = []
    # print(reqbased['roomamenities'])
    for i in range(reqbased.shape[0]):
```

Figure 4.4: Model for hotel recommendation

Similarly we have used the NLTK for searching the favourable local restaurant. Here also we have taken the inputs such as city name and locality in the search restaurant html file. These similarly taking these inputs in the function for search restaurant we have as output also searched the favourable local restaurants

```
reqbased = hotel[hotel['city'] == city.lower()]
reqbased = reqbased[reqbased['hotel_star_rating'] == rating]
reqbased = reqbased.set_index(np.arange(reqbased.shape[0]))
l1 = []
l2 = []
cos = []
# print(reqbased['roomamenities'])
for i in range(reqbased.shape[0]):
    temp_tokens = word_tokenize(reqbased['hotel_facilities'][i])
    temp1_set = {w for w in temp_tokens if not w in sw}
    temp_set = set()
    for se in temp1_set:
        temp_set.add(lemm.lemmatize(se))
    rvector = temp_set.intersection(f_set)
    # print(rvector)
    cos.append(len(rvector))
reqbased['similarity'] = cos
reqbased = reqbased.sort_values(by='similarity', ascending=False)
test = reqbased[['property_name', 'hotel_star_rating', 'address',
                 'hotel_facilities', 'property_type', 'point_of_interest', 'rate']].head(2)
return list(test.T.to_dict().values())
```

Figure 4.5: Model for hotel recommendation

4.1.4 AI chatbot

Intents

Now to understand what kind of data we will need to provide our chatbot with. Since this is a simple chatbot we don't need to download any massive datasets. We will just use data that we write ourselves. To follow along properly you will need to create a .JSON file that contains the same format as the one seen below. I've called my file "intents.json". What we are doing with the JSON file is creating a bunch of messages that the user is likely to type in and mapping them to a group of appropriate responses. The tag on each dictionary in the file indicates the group that each message belongs too. With this data we will train a neural network to take a sentence of words and classify it as one of the tags in our file. Then we can simply take a response from those groups and display that to the user. The more tags, responses, and patterns you provide to the chatbot the better and more complex it will be.

```
109 "tag": "options",
110 "patterns": [
111     "how you could help me?",
112     "what you can do?",
113     "what help you provide?",
114     "how you can be helpful?",
115     "what support is offered"
116 ],
117 "responses": [
118     "I can give travel tips, including weather reports and time zone information!",
119     "I have tips on travel, weather, and time zones."
120 ],
121 "context": [
122     ""
123 ],
124 },
125 {
126     "tag": "bookings",
127     "patterns": [
128         "can you book us a ticket?",
129         "can you make reservation for hotels",
130         "book me a cab",
131         "book me a table at restaurants"
132 ],
133     "responses": [
134         "Yeah I will do that for you",
135         "Sure thing why not?",
136         "let me check for the availability"
137 ],
138     "context_set": [
139         ""
140 ],
141 },
142 {
143     "tag": "travel tips",
144     "patterns": [
145         "how to get a travel tip",
146         "Open travel tips module",
147         "Give me a travel tip",
148         "I want a tip for travel",
149         "help me with travel"
150 ],
151     "responses": [
152         "Here's a tip: always pack light!",
153         "Don't forget to bring a towel! No, seriously!",
154         "Always pack extra socks!",
155         "Here's a tip: keep your passport and credit cards safe and on your person!",
156         "Here's a tip: Solo travel is amazing, but do your research first and be extra cautious!",
157         "Keep your phone charged, and pack a charging pack. You never know when you'll need to find a rideshare, check your map, or call an emergency number.",
158         "Here's a tip: remember to make extra copies of your passport and other documents.",
159         "Try to learn at least a few basic phrases in the local language. It's useful and respectful",
160         "Lunch time is a great time to visit historical sites without the huge crowd.",
161         "Here's a tip: when traveling to a new city, be on the lookout for free walking tours!",
162         "Remember: don't beat up your feet! Bring sensible, comfortable, broken-in shoes.",
163         "Don't forget to take pictures!",
164         "Everyone should wear sunscreen every day, especially while spending a lot of time outside or visiting sunny locales."
165 ],
166     "context": [
167         ""
168 ],
169 },
170 {
171     "tag": "metro",
172     "patterns": [
173         "nearest metro station",
174         "what is the nearest metro station",
175         "which metro station is near",
176     ]
177 }
```

Figure 4.6: Intents for Chatbot

Model

The Dataset retrieves our dataset's features and labels one sample at a time. While training a model, we typically want to pass samples in "minibatches", reshuffle the data at every epoch to reduce model overfitting, and use Python's multiprocessing to speed up data retrieval. DataLoader is an iterable that abstracts this complexity for us in an easy API.

We are trying to map a user utterance (which is just a sequence of tokens) to one of the N intents that we specify. The data we start with should just have an utterance and an intent ground truth label as the columns. The order of this process is as follows:

```

learning_rate = 0.001
input_size = len(X_train[0])
hidden_size = 8
output_size = len(tags)
print(input_size, output_size)

class ChatDataset(Dataset):
    def __init__(self):
        self.n_samples = len(X_train)
        self.x_data = X_train
        self.y_data = y_train

    # support indexing such that dataset[i] can be used to get i-th sample
    def __getitem__(self, index):
        return self.x_data[index], self.y_data[index]

    # we can call len(dataset) to return the size
    def __len__(self):
        return self.n_samples

dataset = ChatDataset()
train_loader = DataLoader(dataset=dataset,
                           batch_size=batch_size,
                           shuffle=True,
                           num_workers=0)

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

model = NeuralNet(input_size, hidden_size, output_size).to(device)

# Loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

# Train the model
for epoch in range(num_epochs):
    for (words, labels) in train_loader:
        words = words.to(device)
        labels = labels.to(dtype=torch.long).to(device)

        # Forward pass
        outputs = model(words)
        # if y would be one-hot, we must apply
        # labels = torch.max(labels, 1)[1]
        loss = criterion(outputs, labels)

        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    if (epoch+1) % 100 == 0:
        print(f'Epoch {(epoch+1)/(num_epochs)}, Loss: {loss.item():.4f}')

print(f'Final loss: {loss.item():.4f}')

data = {
    "model_state": model.state_dict(),
    "input_size": input_size,
    "hidden_size": hidden_size,
    "output_size": output_size,
    "all_words": all_words,
    "tags": tags
}

```

Figure 4.7: Training of Chatbot

- NLTK Word Tokenizer
- Loop through each sentence in our intents patterns and perform Stemming
- Train Test Split
- Initialize the model architecture
- Initialize the model call-backs using Data Loaders (techniques to address overfitting)
- Train the model using Linear Regression and Rectified Linear Unit (ReLU)
- Fit the model and save it
- Load the model and save the output


```
model.py 2 X
model.py > ...
1  import torch
2  import torch.nn as nn
3
4
5  class NeuralNet(nn.Module):
6      def __init__(self, input_size, hidden_size, num_classes):
7          super(NeuralNet, self).__init__()
8          self.l1 = nn.Linear(input_size, hidden_size)
9          self.l2 = nn.Linear(hidden_size, hidden_size)
10         self.l3 = nn.Linear(hidden_size, num_classes)
11         self.relu = nn.ReLU()
12
13     def forward(self, x):
14         out = self.l1(x)
15         out = self.relu(out)
16         out = self.l2(out)
17         out = self.relu(out)
18         out = self.l3(out)
19         # no activation and no softmax at the end
20         return out
21
```

Figure 4.8: Model for Chatbot

Chapter 5

Testing

Software testing is an activity to check whether the actual results match the expected results and to ensure that the software system is defect free. It involves execution of a software component or system component to evaluate one or more properties of interest. Software testing also helps to identify errors, gaps or missing requirements in contrast to the actual requirements. It can be either done manually or using automated tools. Some prefer saying Software testing as a white box and black box testing.

- Black-Box Testing:

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

- White-Box Testing:

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called glass testing or open-box testing. In order to perform white-box testing on an application, a tester needs to know the internal workings of the code. The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

- Grey-Box Testing:

Grey-box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In grey-box testing, the tester has access to design documents and the database. Having this knowledge, a tester can prepare better test data and test scenarios while making a test plan.

Levels of software testing are:

- Functional Testing:

Functional testing is a quality assurance (QA) process and a type of black-box testing that bases its test cases on the specifications of the software component under test.

- Unit Testing:

This type of testing is performed by developers before the setup is handed over to the testing team to formally execute the test cases.

- Integration Testing:

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly.

- System Testing:

System testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards.

- Regression Testing:

Whenever a change in a software application is made, it is quite possible that other areas within the application have been affected by this change.

- Acceptance Testing:

This is arguably the most important type of testing, as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client's requirement.

5.1 Functional Testing

5.1.1 Unit Testing

Unit testing is the first level of testing and is often performed by the developers themselves. It is the process of ensuring individual components of a piece of software at the code level are functional and work as they were designed to. Developers in a test-driven environment will typically write and run the tests prior to the software or feature being passed over to the test team. Unit testing can be conducted manually, but automating the process will speed up delivery cycles and expand test coverage. Unit testing will also make debugging easier because finding issues earlier means they take less time to fix than if they were discovered later in the testing process. After each unit is thoroughly tested, it is integrated with other units to create modules or components that are designed to perform specific tasks or activities. These are then tested as group through integration testing to ensure whole segments of an application behave as expected (i.e., the interactions between units are seamless). These tests are often framed by user scenarios, such as logging into an application or opening files. Integrated tests can be conducted by either developers or independent testers and are usually comprised of a combination of automated functional and manual tests.

Unit tests help to fix bugs early in the development cycle. It helps us to understand the testing code base and enables to make changes quickly. Unit tests also helped with code re-use and to Migrate both our code and our tests to our new project. Followed by the unit testing performed by individual developers Integration Testing was suited as the second best for integration of small modules made by each individual developer. The goal of integration testing was to ensure that individual modules are working as expected after combining them with other modules. It is a good to perform frequent integration testing so that it ensures that after combining modules the integration works perfectly.

5.1.2 Integration Testing

After each unit is thoroughly tested, it is integrated with other units to create modules or components that are designed to perform specific tasks or activities. These are then tested as group through integration testing to ensure whole segments of an application behave as expected (i.e, the interactions between units are seamless). These tests are often framed by user scenarios, such as logging into an application or opening files. Integrated tests can be conducted by either developers or independent testers and are usually comprised of a combination of automated functional and manual tests.

As we have discussed unit testing the next step is integration testing. All the units which we have tested and debugged are now ready to integrate into a whole single module. The integration part is crucial as we need to know which unit must interact without error, calling them in a different class accessing the instance of that class all these can be cleared with help of the sequence diagram which was represented in Project Design. So accordingly, modules are integrated and checked whether they behave as for the objectives.

5.2 Non Functional Testing

5.2.1 Compatibility Testing

Compatibility testing is used to gauge how an application or piece of software will work in different environments. It is used to check that your product is compatible with multiple operating systems, platforms, browsers, or resolution configurations. The goal is to ensure that your software's functionality is consistently supported across any environment you expect your end-users to be using.

5.3 Test Cases

A test case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. The process of developing test cases can also help find problems in the requirements.

Test Case No	Test Case Name	Action	Expected Output	Actual Output	Status
TC1	Register	Enter new Email id and password for validation	Add user into database	Adding user into the database	Pass
TC2	Login	Enter User ID & Password	If both are valid then Logged in else not	Logged in successfully	Pass
TC3	Booking Hotel	Entering details to book hotel	Hotel booked	Hotel Booking Successful	Pass
TC4	Booking Flight	Entering details to book flight	Flight Booked	Flight Booking Successful	Pass
TC5	Booking Confirmation	Booking Details	Mail with Booking Details	Mail with Booking Details Successfully	Pass
TC6	Logout	Click on button	Logged out from application	Successfully logged out	Pass

Figure 5.1: Test Cases

Chapter 6

Result

6.1 Login Process

Users can use the website's features by signing in to the system after browsing through it. Users who have registered or signed up for the website can log in. Once a user has been registered, they can log in to the website using the input username and password used during sign-up. The login and password data entered by the user are taken in the system and compared to the data saved on the database. Once the login details match the details saved on the database the login is successful and the user moves to the next page

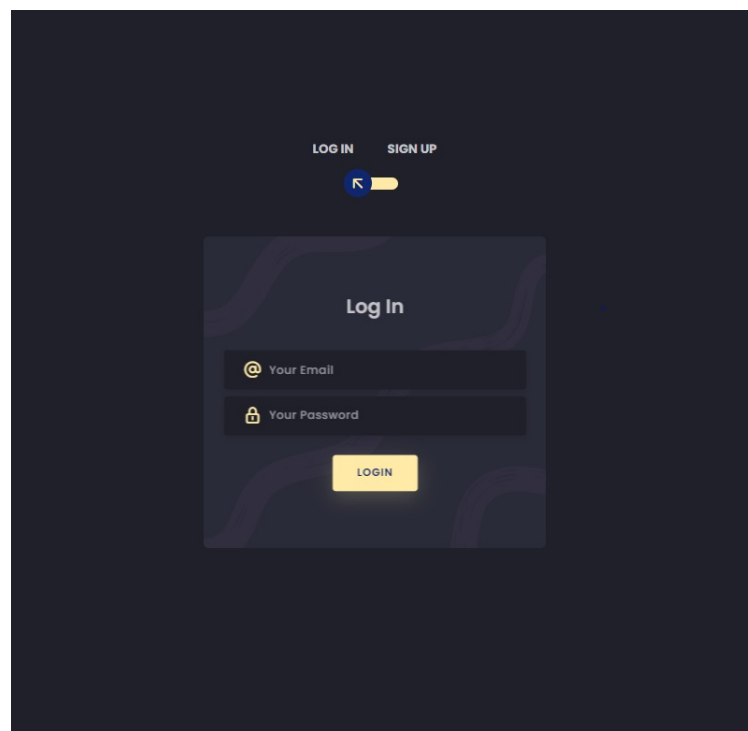


Figure 6.1: Login Screen UI

6.2 Signup Process

On the off chance that a client is new to the website, the client would need to initially join and register themselves on the website. In this cycle, the client is approached to enter essential data. The client is additionally approached to give a username and a password for signing into the website later on. The information taken from the client is shipped off the server data set where it is put away for future references. The password will be put away in the data set while carrying out a hash secret key for the client's information security. After completion of the registration process, the message will display "Sign up successful". .

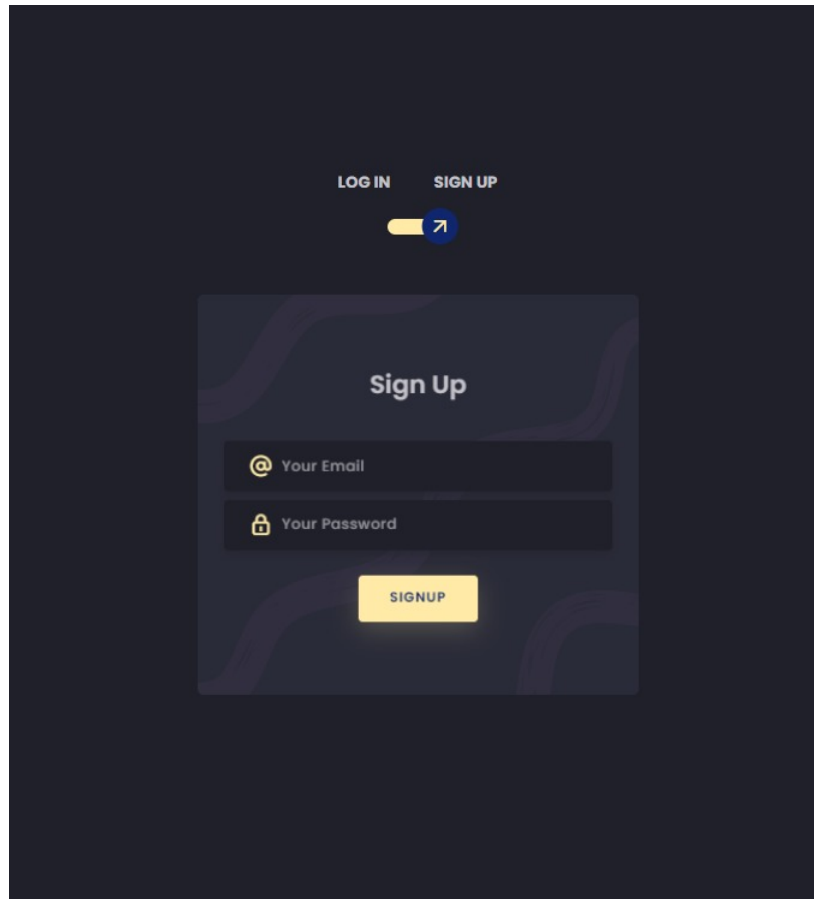


Figure 6.2: Register Screen UI

6.3 AI Based Chatbot

A chatbot is an artificial intelligence (AI) program that can mimic a discussion with a user in genuine human language through informing mediums like websites, mobile applications, or messaging apps. This project incorporates the presence of an AI chatbot using TensorFlow which is a library of python. Thus, while operating the website the users can get a personalized automated support assistant to guide them through in and out of the system. The AI bot is trained using intents and entities to identify and understand the user and to provide suitably trained replies. However, the AI chatbot can only be used for basic user queries. Furthermore, other complex user queries can be posted on the forum. The forum is a community of travel agents and designated forum administrators that assist users by resolving their questions.

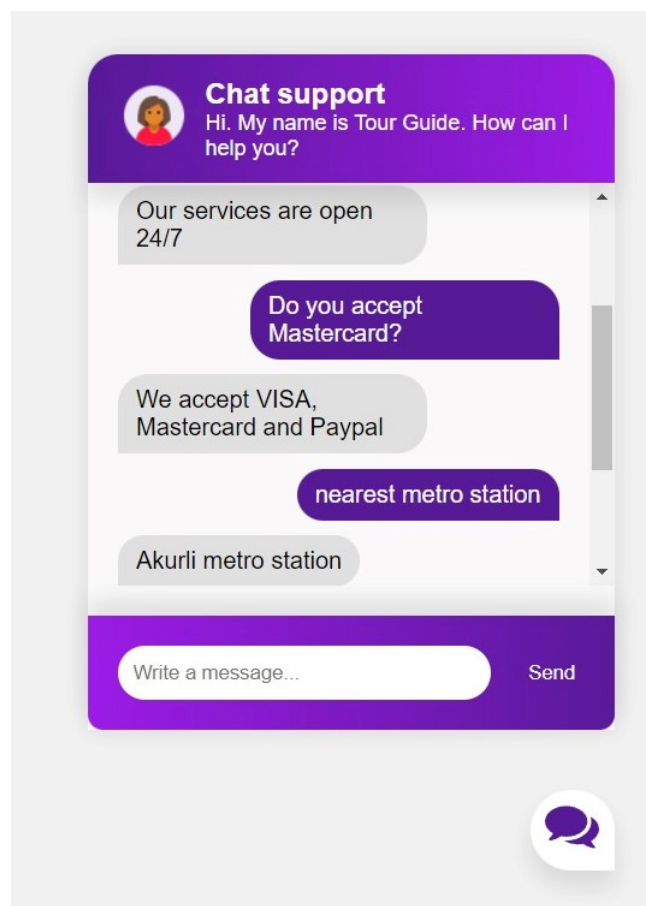
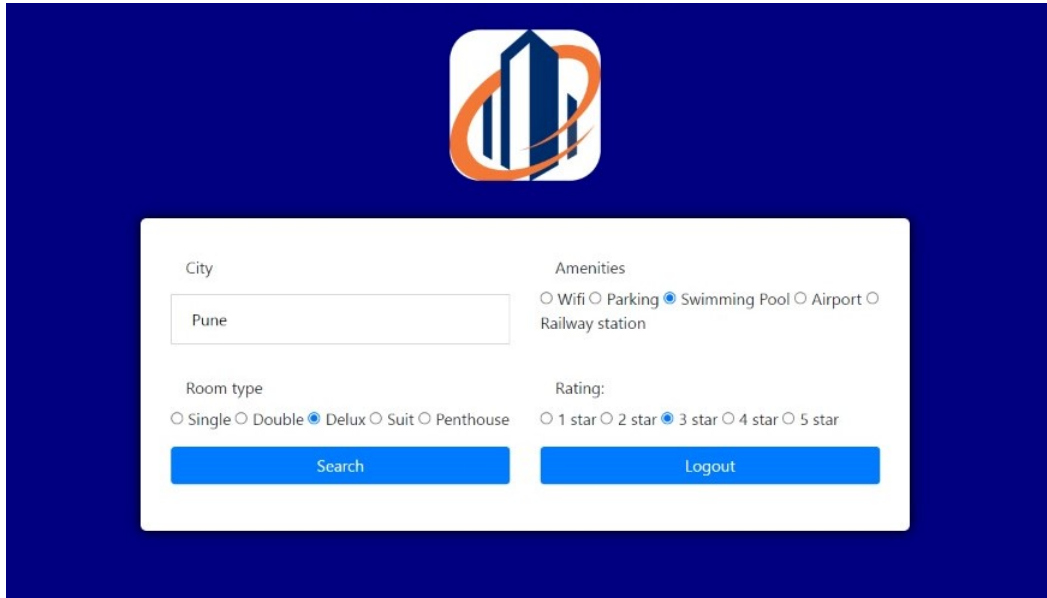


Figure 6.3: Chatbot UI

6.4 Hotel Booking

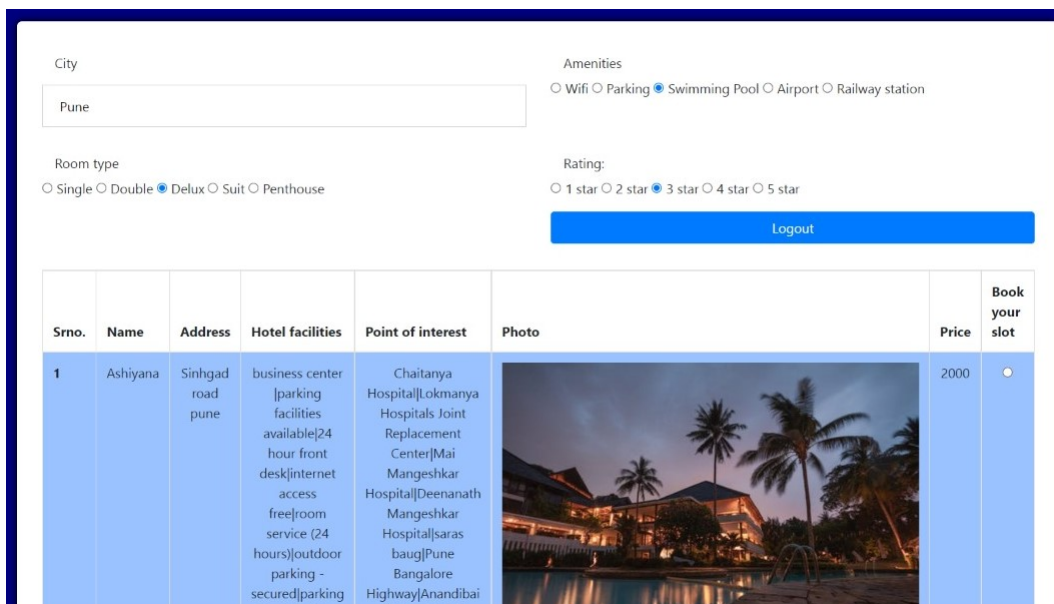
The user searches for the location he wishes to visit, selects the amenities and price range for which hotels are displayed along with nearby attraction places of those hotels, this saves user time and provides more convenience and ease of use to the user; then the user selects the hotel of his/her choice and proceeds with the hotel booking.



The image shows a hotel search form on a dark blue background. At the top center is a logo featuring a stylized building with orange and blue elements. The form itself is a white rectangle with the following fields and options:

- City:** A text input field containing "Pune".
- Amenities:** Radio buttons for ☐ Wifi, ☐ Parking, ☒ Swimming Pool, ☐ Airport, and ☐ Railway station.
- Room type:** Radio buttons for ☐ Single, ☐ Double, ☒ Delux, ☐ Suit, and ☐ Penthouse.
- Rating:** Radio buttons for ☐ 1 star, ☐ 2 star, ☒ 3 star, ☐ 4 star, and ☐ 5 star.
- Buttons:** A blue "Search" button and a blue "Logout" button.

Figure 6.4: Hotel Search



The image shows the output of the hotel search form. It includes the same search filters as Figure 6.4, followed by a table of search results. The table has columns for Serno, Name, Address, Hotel facilities, Point of interest, Photo, Price, and Book your slot. The first row shows a hotel named "Ashiyana" with a price of 2000 and a booking slot available.


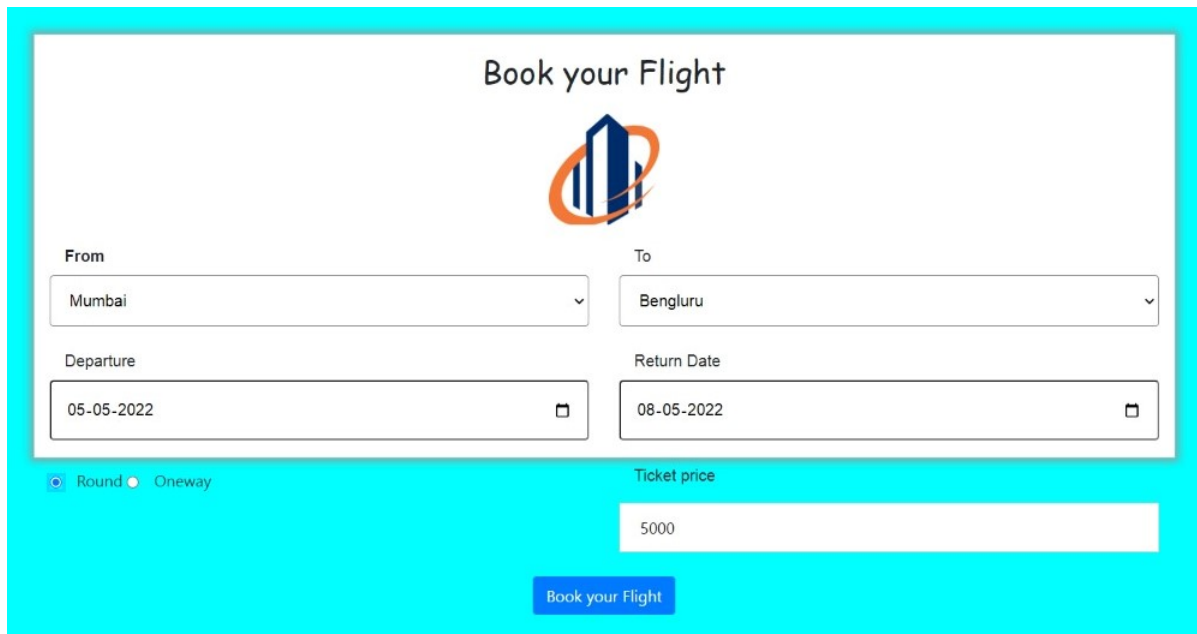
Serno.	Name	Address	Hotel facilities	Point of interest	Photo	Price	Book your slot
1	Ashiyana	Sinhgad road pune	business center [parking facilities available]24 hour front desk]internet access free]room service (24 hours)]outdoor parking - secured]parking	Chaitanya Hospital]]Lokmanya Hospitals Joint Replacement Center]]Mai Mangeshkar Hospital]]Deenanath Mangeshkar Hospital]]saras baug]]Pune Bangalore Highway]]Anandibai		2000	<input type="radio"/>

Figure 6.5: Hotel Search Output

6.5 Flight Booking

The user searches for a flight by inputting the location from where the user will depart, the destination, departure date, return date, whether the trip is round trip, or one way; this displays the user with a list of flights from which the user can select a flight of their choice and complete the booking.



The screenshot shows a web form for booking a flight. At the top, it says "Book your Flight" with a logo of a stylized building. Below the title, there are four input fields: "From" (Mumbai), "To" (Bengluru), "Departure" (05-05-2022), and "Return Date" (08-05-2022). Each field has a dropdown arrow. Below these fields, there are two radio buttons: "Round" (selected) and "Oneway". To the right of the radio buttons, there is a "Ticket price" label and a text box showing "5000". At the bottom center, there is a blue button labeled "Book your Flight".

Figure 6.6: Flight Booking

6.6 Application Performance

The existing system used Contextual Filtering based on Area of Interest for developing recommendations. And we have used context based filtering along with user profile filtering (hybrid recommendation) which provides more personalized recommendations considering various different parameters. These more personalized recommendations provide better performance to the system as it improves user experience. The figure below indicates user experience/satisfaction based on personalized recommendations between our system .



Figure 6.7: Application Performance

Chapter 7

Conclusions and Future Scope

7.1 Conclusions

In this project we have created an e-tourism website that allows users to plan their trip entirely by themselves from researching places, to know about restaurants and hotels suited to the user's budget and preference, booking hotels, generating unique personalized tour plans etc. The Smart Tourism System provides a streamlined approach through the entire tour planning process. It helps users by avoiding waste of the user's time to go through various tourist platforms to research and plan the tour, streamlines the process of booking hotels and restaurants based on users' preferences, prepare the entire travel plan at the ease of the user at the comfort of his home. The users can plan and book the tour virtually than to physically visit the travel agent or tourist companies. The users can safely book tours by avoiding some travel agents that can scam tourists. This project helps users connect better with the tourism community and support when users have any tour related queries. This project thereby improves the efficiency, simplifies the process and consumes less time to plan a tour.

7.2 Future Scope

As AI/ML is employed for analysis and solution generation, the framework will become quicker and more efficient. Once used and managed domestically, the system may be used for an international tour all over the world. It will allow customers to reserve trains, automobiles, planes, and other kinds of transportation. A system can collaborate with hotels, restaurants, and other eateries to promote and highlight them on the site. If a recommendation system can be developed based solely on considering the visual content of the video, it would become the most accurate recommender system. A more realistic step towards achieving this can be to develop a model that could make recommendations according to the visual content of the video shorts instead of the whole video. This could also ensure that the recommended list varies according to the latest trends in video content.

Bibliography

- [1] Barranco M.J., Noguera J.M., Castro J., Martínez L. (2012). A Context-Aware Mobile Recommender System Based on Location and Trajectory. In: Casillas J., Martínez-López F., Corchado Rodríguez J. (eds) Management Intelligent Systems. Advances in Intelligent Systems and Computing, vol 171. Springer, Berlin, Heidelberg.
- [2] Adomavicius G., Tuzhilin A. (2011) Context-Aware Recommender Systems. In: Ricci F., Rokach L., Shapira B., Kantor P. Recommender Systems Handbook. Springer, Boston, MA.
- [3] Pazzani M.J., Billsus D. (2007) Content-Based Recommendation Systems. In: Brusilovsky P., Kobsa A., Nejdl W. (eds) The Adaptive Web. Lecture Notes in Computer Science, vol. 4321. Springer, Berlin, Heidelberg.
- [4] Aw Yoke Cheng, Ab Hamid, N. R. (2011). Behaviour and preferences in browsing the travel and tourism websites. 2011 IEEE Colloquium on Humanities, Science and Engineering.
- [5] Jafri, R., Alkhunji, A. S., Alhader, G. K., Alrabeiah, H. R., Alhammad, N. A., Alzahrani, S. K. (2013). Smart Travel Planner: A mashup of travel-related web services. 2013 International Conference on Current Trends in Information Technology (CTIT).
- [6] Shafiee, M. M., Rahimzadeh, S., Haghighizade, R. (2016). The effect of implementing SEO techniques and websites design methods on e-tourism development: A study of travel agencies e-tourism websites. 2016 10th International Conference on e-Commerce in Developing Countries: With Focus on e-Tourism (ECDCC). doi:10.1109/ecdc.2016.7492963.
- [7] M. D. Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In Proc. of HT'10, pages 35–44, 2010.
- [8] Meteren, R. and Someren, M., 2000. Using Content-Based Filtering for Recommendation.
- [9] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. 2011. Collaborative Filtering Recommender Systems. Found. Trends Hum.-Comput. Interact. 4, 2 (February 2011), 81–173.
- [10] Gavalas, D., Konstantopoulos, C., Mastakas, K. et al. A survey on algorithmic approaches for solving tourist trip design problems. J Heuristics 20, 291–328 (2014).

- [11] Stomberg, Joshua C., "A COMPARATIVE STUDY AND EVALUATION OF COLLABORATIVE RECOMMENDATION SYSTEMS", Master's Thesis, Michigan Technological University, 2014.
- [12] Lim, Kwan Hui Chan, Jeffrey Karunasekera, Shanika Leckie, Christopher. (2019). Tour recommendation and trip planning using location-based social media: a survey. Knowledge and Information Systems. 60. 10.1007/s10115-018-1297-4.

Appendices

Appendix-A: Python Download and Installation

1. Install python from <https://www.python.org>
2. Go to environment variable set path for python.

Appendix-B: Flask Download and Installation

1. Download Flask after python is installed using pip
pip install flask

Appendix-C: MongoDB Download and Installation

1. Download the MongoDB .msi file from <https://www.mongodb.com/try/download/community?tck=docs-server>
2. Open Windows Explorer/File Explorer.
3. Change the directory path to where you downloaded the MongoDB .msi file. By default, this is
4. Double-click the .msi file.
5. The Windows Installer guides you through the installation process.
6. If you choose the Custom installation option, you may specify an installation directory.
7. MongoDB does not have any other system dependencies.
You can install and run MongoDB from any folder you choose.

Appendix-D: Packages for App implementation

Once python is installed; following packages can be installed using pip (package install manager) with commands

- `pip install pandas`
- `pip install Sklearn`
- `pip install Numpy`
- `pip install nltk`

Publication

Paper entitled “**Implementing AI Based Comprehensive Web Framework for Tourism**” is presented at “**Springer ICTIS 2022**” by “**Nada Rajguru, Jaynam Shah and Harsh Shah**”.