TRAVEL MANAGEMENT SYSTEM USING PYTHON DJANGO

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Computer Science

By

R. MADHUMITHRA

REG.NO. 40290054



DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF SCIENCE AND HUMANITIES

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC | 12B Status

By UGC | Approved by AICTE

JEPPIAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI 600119

MAY 2023

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **R. MADHUMITHRA (40290054)** who carried out the Project entitled **"TRAVEL MANAGEMENT SYSTEM USING PYTHON DJANGO"** under our supervision from January 2023 to May 2023.

Shamili.V

Internal Guide

MS. SHAMILI V, M.Sc., Assistant Professor

Rekha

Dean of arts and science

Dr. REKHA CHAKRAVARTHI M.E., Ph.D.

Submitted for Viva-voce Examination held on 08.05.2023

Internal Examiner

External Examiner

DECLARATION

I am, **R. MADHUMITHRA (40290054)** hereby declare that the Project Report entitled **"TRAVEL MANAGEMENT SYSTEM USING PYTHON DJANGO."** done by me under the guidance of **Ms. SHAMILI V, M.Sc., Assistant Professor** Department of Computer Science at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**, Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600119 is submitted in partial fulfillment of the requirements for the award of Bachelor of degree in Computer Science.

DATE:

PLACE: Chennai

R. MADHUMITHRA

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management** of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this Project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. REKHA CHAKRAVARTHI M.E., Ph.D.,** Dean of arts and science for providing me with necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guides, of **Ms. SHAMILI V, M.Sc.,** Assistant Professor, her valuable guidance, suggestions, and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science who were helpful in many ways for the completion of the Project.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
	ABSTRACT	
1	**INTRODUCTION**	
	1.1 PROJECT OBJECTIVES	1
	1.2 PURPOSE	2
	1.3 PROBLEM DEFINITION	3
	1.4 PURPOSE SOLUTION	5
	1.5 LITERATURE SURVEY	11
2	**AIMANDSCOPEOFTHEPROPOSEDSYSTEM**	
	2.1 AIM	13
	2.2 SCOPE	13
3	**SYSTEMANALYSIS**	
	3.1 EXISTING SYSTEM	15
	3.2 DISADVANTAGES OF EXISTING SYSTEM	16
	3.3 PURPOSED SYSTEM	17
	3.4 ADVANTAGES OF PURPOSE SYSTEM	18
4	**SYSTEM CONFIGURATION**	
	4.1 SYSTEM REQUIREMENT ANALYSIS	19
	4.1.1FUNCTIONAL REQUIREMENTS	21
	4.1.2 NON-FUNCTIONAL REQUIREMENTS	22
	4.1.3 HARDWARE REQUIREMENTS	24
	4.1.4 SOFTWARE REQUIREMENTS	24
5	**SYSTEM DESIGN**	

	5.1 SYSTEM ARCHITECTURE	25
	5.2 DATA FLOW DIAGRAM	27
	5.3 ENTITY RELATIONSHIP DIAGRAM	34
	5.4 USECASE DIAGRAM	35
	5.5 SEQUENCE DIAGRAM	37
	5.6 ACTIVITY DIAGRAM	37
	5.7 MODULE DESCRIPTION	39
6	**SYSTEM TESTING**	
	6.1 SYSTEM TESTING	42
7	**CONCLUSION**	
	7.1 FUTURE ENHANCEMENT	44
	7.2 APPENDIX	45
	7.2.1 SOURCE CODE	45
	7.2.2 SCREEN SHOT	50
	7.2.3 OUTPUT	52
	7.2.4 REPORT	53
8	**BOOK REFERENCE**	54

ABSTRACT

The travel industry has grown significantly over the years, leading to a rise in the number of travel agencies and tour operators. As a result, managing travel operations efficiently has become a challenging task. To address this issue, a Travel Management System (TMS) can be used to streamline the processes involved in managing travel operations. In this abstract, we propose a Travel Management System using the Python Django framework. The system allows travel agencies and tour operators to manage various travel-related tasks such as booking flights, hotels, transportation, and activities. The system also provides a centralized platform for managing customer information, payments, and trip itineraries. The proposed TMS consists of several modules, including a booking module, payment module, itinerary module, and customer module. The booking module allows travel agents to make reservations for flights, hotels, and other travel-related services. The payment module enables customers to make payments securely and easily. The itinerary module allows agents to create trip itineraries for their customers, including details of the various activities and events. Finally, the customer module provides a centralized platform for managing customer information, including their personal details, trip history, and preferences. Overall, the proposed Travel Management System is a comprehensive solution that can simplify the travel management process for travel agencies and tour operators. By leveraging the power of the Python Django framework, the system can offer a robust and scalable solution that can adapt to the changing needs of the travel industry.

CHAPTER 1

INTRODUCTION

1.1 PROJECT OBJECTIVES

The travel industry has seen significant growth in recent years, leading to an increase in the number of travel agencies and tour operators. Managing travel operations can be a complex and challenging task, requiring coordination across multiple stakeholders, including customers, suppliers, and service providers.

To address these challenges, many travel agencies and tour operators have started using Travel Management Systems (TMS). A TMS is a software solution that streamlines the process of managing travel operations, including booking, payment, itinerary creation, and customer management.

In this project, we propose a Travel Management System using the Python Django framework. Python Django is a powerful web framework that enables rapid development of scalable, secure, and feature-rich web applications. By leveraging the capabilities of Python Django, we can build a robust and flexible TMS that can meet the needs of travel agencies and tour operators.

The proposed TMS will enable travel agencies and tour operators to manage various travel-related tasks, including booking flights, hotels, transportation, and activities. The system will also provide a centralized platform for managing customer information, payments, and trip itineraries.

The TMS will be designed to provide a user-friendly and intuitive interface, enabling travel agents to manage travel operations efficiently. The system will also be designed to ensure the security and confidentiality of customer information and transactions.

Overall, the proposed Travel Management System using Python Django will provide a comprehensive solution that can simplify the travel management process for travel agencies and tour operators.

1.2 PURPOSES

The primary purpose of a Travel Management System (TMS) using Python Django is to simplify and streamline the travel management process for travel agencies and tour operators. Here are some of the key purposes of the proposed TMS:

Booking Management:

The TMS will enable travel agents to manage bookings for flights, hotels, transportation, and activities in a centralized system. This will eliminate the need for manual processes and ensure that all bookings are recorded accurately.

Payment Management:

The TMS will provide a secure and user-friendly payment platform that enables customers to make payments easily and securely. The system will also enable travel agents to manage payments and track payment history.

Itinerary Management:

The TMS will enable travel agents to create detailed trip itineraries for customers, including details of flights, hotels, transportation, and activities. This will ensure that customers have a seamless travel experience and can easily access all the information they need.

Customer Management:

The TMS will provide a centralized platform for managing customer information, including personal details, trip history, and preferences. This will enable travel agents to personalize their services and provide customers with a better travel experience.

Reporting and Analytics:

The TMS will enable travel agencies and tour operators to generate reports and analyze data related to bookings, payments, and customer information. This will provide insights that can help travel agents make better decisions and improve their services.

Overall, the primary purposes of the proposed TMS using Python Django are to simplify and streamline travel management processes, enhance the customer experience, and provide travel agencies and tour operators with the tools they need to manage their operations efficiently.

1.3 PROBLEM DEFINITION

Building a travel management system using Python Django involves several steps. Here is an overview of the steps involved:

Define the requirements: Before you start building the system, you need to understand the requirements of the system. What features should it have? What are the use cases? What are the expected inputs and outputs? You should work closely with the stakeholders to gather these requirements.

Design the database: Once you have a clear understanding of the requirements, you can start designing the database schema. You should consider the entities involved,

their relationships, and the attributes that need to be stored for each entity. You can use tools such as ER diagrams to help you with this task.

Implement the database: After designing the database schema, you can create the database and its tables. You can use Django's ORM to define the models and migrate them to the database.

Implement the views: Views are the controllers that handle the user's requests and generate the responses. You can create views for each use case defined in the requirements.

Implement the templates: Templates are used to render the HTML pages that the user sees. You can create templates for each view and use Django's template language to insert data into the HTML.

Implement the forms: Forms are used to validate user inputs and generate the HTML form elements. You can use Django's forms library to create forms for each view.

Test the system: Once the implementation is complete, you should thoroughly test the system to ensure that it meets the requirements. You can use automated testing tools such as Django's test framework to write tests for each use case.

Here are some specific features that you can implement in the travel management system:

User authentication: Implement a user authentication system that allows users to sign up, log in, and log out.

Flight booking: Implement a feature that allows users to search for and book flights. You can use an API such as Amadeus or Sabre to search for available flights.

Hotel booking: Implement a feature that allows users to search for and book hotels. You can use an API such as Booking.com or Expedia to search for available hotels.

Itinerary management: Implement a feature that allows users to manage their travel itinerary, including their flight and hotel bookings.

Payment integration: Implement a payment gateway that allows users to pay for their bookings using a credit card or other payment methods.

Admin panel: Implement an admin panel that allows administrators to manage users, bookings, and other aspects of the system.

These are just a few examples of the features that you can implement in the travel management system. The exact features will depend on the requirements of the system.

1.4 PROPOSED SOLUTION

A travel management system can be implemented using Python Django. Django is a powerful web framework that can be used to build complex and scalable applications quickly. Here's a high-level overview of how you could build a travel management system using Django.

Define the requirements and use cases: Before starting to build the system, it's important to define the requirements and use cases for the system. This includes features such as booking flights, hotels, and rental cars, managing itineraries, and generating reports.

Design the database: Next, design the database schema to store the data for the system. This includes tables for users, flights, hotels, rental cars, itineraries, and other relevant data.

Create Django models: Once the database schema is designed, create Django models to represent the tables in the database. Use the Django ORM to query the database and manipulate data.

Implement user authentication: Implement user authentication and authorization to ensure that only authorized users can access the system.

Implement booking functionality: Implement functionality to allow users to book flights, hotels, and rental cars through the system. This involves integrating with third-party APIs or building your own booking system.

Implement itinerary management: Implement functionality to allow users to manage their itineraries, including adding and removing items, and editing details.

Generate reports: Implement functionality to generate reports on user bookings and travel expenses.

Test and deploy: Test the system thoroughly and deploy it to a production environment.

Overall, building a travel management system using Django requires a lot of work.

The Tours and Travel Management System is a web based application. The main purpose of "Tours and travels management system " is to provide a convenient way for a customer to book hotels, flight, train and bus for tour purposes. The objective of this project is to develop a system that automates the processes and activities of a travel agency. In this project, We will make an easier task of searching places and

for booking train, flight or bus. . In the present system a customer has to approach various agencies to find details of places and to book tickets. This often requires a lot of time and effort. We provide approach skills to critically examine how a tourist visits and its ability to operate in an appropriate way when dealing with the consequences of tourism, locally, regionally, and nationally including visitor security and ecological influences. It is tedious for a customer to plan a particular journey and have it executed properly. The project 'Tours and Travels Management System' is developed to replace the currently existing system, which helps in keeping records of the customer details of destination as well as payment received.

Objectives

This application is develop to provide best travelling services to the customers and travel agents. We have developed tours and travel management system to provide a search platform where a tourist can find their tour places according to their choices. This system also helps to promote responsible and interesting tourism so that people can enjoy their holidays at their favorable places. This system also helps to develop tourism with different cultures so that they enrich the tourism experience and build pride. We develop this system to create and promote forms of tourism that provide healthy interaction opportunities for tourists and locals and increase better understanding of different cultures, customs, lifestyles, traditional knowledge and believes. This system also provide a better way to connect with various events.

This system also gives tours related information like which places are tourist attractions, cities, and provinces. Tourist can also get the Map and navigation system and temperature and weather information. Tourist can also book tours through our tours and travels management system. This system also keeps a history of visited places of its users.

Below are the modules of this application

Modules of travel and tour management system

User Management

a. Login.

b. User profile.

c. Update information.

d. Role based rights.

Administrator module :

This module provides administrator related functionality like from this module use can add route information, bus information, train information, flight information, tour packages, travel packages, bus seat details, etc. From this module Admin can view daily, weekly and monthly report. This module is develop for admin of the website and admin can add, delete, edit and view the data related to places, travels, routes, bookings from this module.

a. Manage user information.

b. Update information.

c. Manage Trips.

d. Manage Transportation.

e. Manage hotel, Bookings.

Hotel Module

Details of the hotels in which the accommodation of the customer will be done during the tours. Details like availability of meals, station-pickup and drop facility and

contact no. of the hotels are also provided to the customers on special request.

a. Registration (as user)

b. Registration (as Hotel)

c. Search

Transportation Module

a. Flight

b. Train

c. Bus

Package Modules

User can view different tour packages available for tourist. User can select any packages from this module he can also check the details of various travel agencies. A user can select any travel agency from this module.

Testimonials module :

This is testimonials module where passenger can post feedback after the journey and they can share their experience. Users of this application can post their opinions, complaints and suggestions regarding

a. Photos

b. Videos

Payment Module

a. pay payment through PayPal

b. pay payment through Draft

Search Module

a. search City wise hotels, flights, packages, bus, rails and events.

Routes module:

This will display the route information of Source location and destination location. Use can also check bust routes for his destination. User can check bust rout train tout and car rout for his journey and can select any rout packages from the available tour packages. From this module use can also get information related to various routes connecting sources and destinations. For each route, information such as source, destination, fare, reservation details, pick up points etc are provides.

Reservations module:

This module is for passengers where passenger can reserve the seats by making payment. Using this module user can book bus ticket or train ticket. From this module user can also book tickets or cancel previously booked tickets. The module maintains the details of all reservations made so far and allows administrator to either confirm or reject the bookings.

this portal and services to the administrator. Accordingly, the administrator can take various steps to act on the complaints and suggestions.

Tour Details :

Details of different types of tours which includes tours like family tours, couple tours, general tours, date and time of departure and the fair of the tours etc are maintained. As the customer ask for the details of a particular tour, the tours and travel management system gives the details of the related place where tourist wants to go and the date and time of the tour, no. of seats available of that particular tour, fair of the tour, details of discount on a particular tour package if any. Details of the pick-up facilities and the drop facilities are also maintained if any.

DFD - DATA FLOW DIAGRAM OF TOURS AND TRAVEL MANAGEMENT SYSTEM

CONTEXT LEVEL DFD

CONTEXT LEVEL DFD - TOURS AND TRAVELS MANAGEMENT SYSTEM

1.5 LITERATURE SURVEY

YEAR: 2022

DESCRIPTION:

"Design and Implementation of Travel Management System Based on Django" by Zhi Liu, Yanli Li, and Zhe Li. This paper describes the design and implementation of a travel management system using Python Django framework. The system provides features like itinerary management, hotel and flight booking, and expense management.

"Development of a Travel Management System using Django" by Shubham Kapoor, Ankit Gupta, and Sachin Kumar. This paper presents the development of a travel management system using Django framework. The system includes features like travel planning, itinerary management, and booking management.

"Design and Implementation of an Online Travel Management System Based on Django Framework" by Yixin Sun and Yan Wang. This paper presents the design and implementation of an online travel management system using Python Django framework. The system includes features like travel planning, booking management, and expense management.

"Implementation of a Travel Management System Using Python and Django" by Arvind Kumar and Ankit Aggarwal. This paper describes the implementation of a travel management system using Python Django framework. The system includes features like itinerary management, hotel and flight booking, and expense management.

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM OF THE PROPOSED SYSTEM

The aim of a travel management system is to streamline and automate the processes involved in managing business travel for an organization. This can include managing travel requests, booking flights, hotels and rental cars, managing travel expenses, and tracking travel data for analysis and reporting purposes. The key objectives of a travel management system are to improve the efficiency and cost-effectiveness of business travel, while also enhancing the traveler experience and ensuring compliance with corporate travel policies. By providing a centralized platform for managing all aspects of business travel, a travel management system can help organizations to reduce travel expenses, increase visibility into travel spending, improve travel policy compliance, and enhance traveler safety and satisfaction.

2.2 SCOPE OF THE PROPOSED SYSTEM

A travel management system is a software application designed to help businesses manage and streamline their travel-related activities, including booking, scheduling, and expense management. The scope of the proposed system for a travel management system may include the following features:

Travel booking: The system should allow employees to book flights, hotels, rental cars, and other travel-related services through the platform.

Travel policy compliance: The system should be designed to ensure that employees comply with the company's travel policy, such as preferred vendors, budget limits, and other rules.

Approval workflows: The system should enable managers to review and approve

employee travel requests, as well as track travel expenses against the approved budget. Travel tracking: The system should allow companies to track employee travel, including flight details, hotel reservations, and other travel-related data.

Expense management: The system should enable employees to submit travel expenses, such as receipts and invoices, for reimbursement, and allow managers to review and approve expense reports.

Reporting and analytics: The system should provide real-time data and reporting to help companies analyze travel spend, identify cost-saving opportunities, and optimize travel policies.

Mobile access: The system should be accessible via mobile devices, allowing employees to book travel, submit expenses, and receive travel updates on the go.

Overall, the scope of the proposed system for a travel management system should cover all aspects of travel-related activities within an organization, from booking to expense management, to ensure streamlined and efficient travel management.

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

There are many existing travel management systems available in the market. Here are a few examples:

Concur: This is a popular travel and expense management system that allows companies to manage their travel bookings, expenses, and reporting in one place. It also offers features like travel policy compliance, approval workflows, and reporting.

Expensify: Expensify is another popular travel and expense management system that streamlines expense reporting and travel booking processes. It has features like automated expense tracking, receipt scanning, and integration with popular accounting software.

Travel Perk: Travel Perk is a platform that simplifies business travel by providing a single tool for booking, managing, and analyzing corporate travel. It also offers features like budget control, policy enforcement, and real-time communication with travelers.

Trip Actions: Trip Actions is a corporate travel management platform that combines technology with human expertise to provide a seamless travel experience. It has features like personalized travel recommendations, 24/7 customer support, and cost-saving tools.

Serko: Serko is a New Zealand-based travel and expense management system that provides a range of services, from booking travel to managing expenses. It has features like travel policy compliance, trip approval workflows, and real-time reporting.

These are just a few examples of the many travel management systems available. Each system has its own set of features and benefits, so it's important to research and compare different options to find the best fit for your company's needs.

3.2 Disadvantages

While Python Django can be a great choice for developing a travel management system, there are a few disadvantages that you should be aware of:

Steep learning curve: Django can be complex and require a fair amount of experience to use effectively. Developers who are new to Python or Django may find the learning curve quite steep.

Limited scalability: Django may not be the best option for very large and complex systems with high traffic and multiple integrations. It can become slow and cumbersome to manage as the application grows.

Limited customization: Django's batteries-included approach means that it comes with a lot of pre-built functionality, which can make it easier to develop applications quickly. However, this can also limit customization options for some features and functionalities.

Debugging can be challenging: Because Django is a framework, it can be difficult to debug issues that occur in the underlying code. This can be particularly challenging for developers who are not familiar with Django's architecture.

Security risks: As with any web application, a travel management system built using Django can be vulnerable to security risks such as SQL injection attacks, cross-site scripting, and more. Developers need to take the necessary steps to ensure that the

application is secure and regularly update the dependencies to avoid security vulnerabilities.

In summary, while Python Django is a great choice for building travel management systems, it's important to be aware of its limitations and take appropriate measures to mitigate the potential risks.

3.3 PROPOSED SYSTEM

A travel management system is a software application that assists in managing all aspects of a corporate travel program. The proposed system is designed to streamline and automate the travel booking process, expense tracking, and reporting, thus reducing the burden of manual processes for both employees and travel managers. The system will offer a user-friendly interface that allows employees to book flights, hotels, and rental cars in compliance with corporate travel policies. Additionally, the system will provide real-time updates on travel itineraries, including changes and cancellations, to ensure that travelers are always informed of their travel plans. The system will also automate the expense reporting process, allowing employees to upload receipts and expense reports, which can be approved and processed by travel managers in real-time. The proposed system is designed to enhance corporate travel management, reducing costs and improving the overall travel experience for employees.

3.4 Advantages

A travel management system using Python Django has several advantages, including:

Scalability: Django is designed to handle high-traffic websites, making it ideal for travel management systems that need to process large volumes of data.

Security: Django has built-in security features, such as protection against common web vulnerabilities, which makes it an ideal choice for travel management systems that handle sensitive information such as personal and payment details.

Flexibility: Django's modular structure makes it easy to customize and add new features to a travel management system, allowing it to adapt to changing business needs and customer requirements.

Efficiency: Python is a high-level programming language known for its simplicity and ease of use. This, coupled with Django's efficient development framework, enables developers to quickly create and deploy a travel management system that is both user-friendly and efficient.

Integration: Python Django can be easily integrated with other technologies and tools, such as payment gateways, social media, and customer relationship management (CRM) systems, making it a versatile choice for travel management systems.

Overall, a travel management system built using Python Django can provide a secure, scalable, and efficient platform for managing travel bookings, customer data, and payments, while also being flexible and customizable to meet changing business needs.

CHAPTER 4

SYSTEM CONFIGURATION

4.1 SYSTEM REQUIREMENT ANALYSIS

System Requirement Analysis of a Travel Management System using Python Django involves the following steps:

Identify the stakeholders: The first step is to identify the stakeholders of the travel management system. This may include travel agents, customers, and travel service providers.

Gather requirements: Next, gather the requirements for the system by interviewing the stakeholders and conducting surveys. Some of the requirements that may be gathered include:

- The ability to search for and book flights, hotels, and rental cars.
- The ability to view travel itineraries and make changes.
- The ability to make payments securely.
- The ability to receive notifications about flight delays and cancellations.
- The ability to provide feedback and ratings on the travel experience.
- The ability to view travel history and loyalty program status.

Define the scope: Define the scope of the travel management system by identifying the key features that will be included in the system. This will help to ensure that the project is manageable and that the stakeholders' requirements are met.

Create use cases: Create use cases for the system to identify the different scenarios in which the system will be used. This will help to identify the different functionalities that will be required in the system.

Develop the database schema: Develop the database schema for the system by identifying the entities and relationships between them. This will help to ensure that the data is organized and can be easily retrieved.

Design the user interface: Design the user interface for the system by creating wireframes and mockups. This will help to ensure that the user experience is intuitive and easy to use.

Determine the technology stack: Determine the technology stack that will be used to develop the system. For a Python Django-based travel management system, the following technologies may be used:

- Python 3.x
- Django framework
- HTML/CSS
- JavaScript
- jQuery
- Bootstrap

Define the system architecture: Define the system architecture by identifying the components and their interactions. This will help to ensure that the system is scalable and can be easily maintained.

Create a project plan: Create a project plan that outlines the tasks, timelines, and resources required to complete the project. This will help to ensure that the project is completed on time and within budget.

Test the system: Test the system to ensure that it meets the stakeholders' requirements and is free of defects. This will help to ensure that the system is reliable and performs as expected.

By following these steps, a travel management system using Python Django can l developed that meets the requirements of the stakeholders and provides a great us experience.

4.1.1 FUNCTIONAL REQUIREMENT

A travel management system using Python Django should have several function requirements to enable it to efficiently manage travel arrangements. Some of thes functional requirements include:

User registration and login: The system should allow users to register and creat their profiles. Users should also be able to log in to the system using their logi credentials.

Flight booking: The system should allow users to search for and book flights. Th system should also allow users to choose their preferred seats and meals.

Hotel booking: The system should allow users to search for and book hotels. Th system should also provide users with information about the amenities available a the hotel, such as gym facilities, swimming pools, and Wi-Fi.

Car rental: The system should allow users to search for and rent cars for their travels.

Trip planning: The system should allow users to plan their trips by creating itineraries and adding different travel arrangements such as flights, hotels, and car rentals.

Payment processing: The system should provide a secure payment gateway to allow users to pay for their travel arrangements.

Travel alerts: The system should provide users with real-time travel alerts such as flight delays, cancellations, and gate changes.

User reviews: The system should allow users to review and rate the different travel arrangements such as hotels, flights, and car rentals.

Travel documentation: The system should allow users to upload and manage their travel documents such as passports, visas, and travel insurance.

Customer support: The system should provide users with customer support such as live chat, phone support, and email support to assist them in case of any issues or queries.

4.1.2 NON-FUNCTIONAL REQUIREMENT

Non-functional requirements for a travel management system might include:

Performance: The system should be able to handle a large number of concurrent users and transactions without slowing down or crashing. It should also have a fast response time to user requests.

Security: The system should have robust security measures to protect sensitive user information, such as credit card numbers and personal data. This includes encryption of data, secure login processes, and protection against hacking attempts.

Reliability: The system should be highly reliable and available 24/7. This includes the ability to recover quickly from system failures, such as power outages or hardware malfunctions.

Scalability: The system should be able to handle increasing amounts of data and users as the company grows. This includes the ability to add new features and integrate with other systems seamlessly.

Usability: The system should be easy to use and navigate for both customers and employees. It should also be accessible on multiple devices and platforms, including mobile devices.

Maintainability: The system should be easy to maintain and update by the development team. This includes clear documentation, modular code design, and efficient testing and debugging processes.

Compatibility: The system should be compatible with different web browsers, operating systems, and hardware devices. This ensures that users can access the system from a wide range of devices and platforms.

4.1.3 HARDWARE REQUIREMENT

- Processor : Dual core processor 2.6.0 GHz
- RAM : 1GB
- Hard disk : 160 GB
- Compact Disk : 650 MB
- Keyboard : Standard keyboard
- Monitor : 15 inch color monitor

4.1.4 SOFTWARE REQUIREMENT

- Front End : PYTHON
- Back End : MySQL
- Platform : Windows 7

4.2 EXPERIMENTAL METHODS

4.2.1 Artificial Neural Network

An artificial neural network can be used to enhance the functionality of a travel management system in various ways, such as predicting travel demand, optimizing travel routes, and personalizing travel recommendations. Here's a brief overview of how you can implement an artificial neural network for travel management using Python Django:

Data Collection: The first step is to collect the necessary data for training the neural network. This data can include travel history, user preferences, trip details, and other relevant information. You can use Django ORM to store the data in a database.

Data Preprocessing: Once you have collected the data, the next step is to preprocess it for training the neural network. This includes tasks such as data cleaning, feature selection, and normalization.

Neural Network Architecture: The architecture of the neural network depends on the specific task you want to accomplish. For example, if you want to predict travel demand, you can use a feedforward neural network with one or more hidden layers. If you want to optimize travel routes, you can use a convolutional neural network or a recurrent neural network.

Training and Testing: After defining the neural network architecture, you can train it on the preprocessed data using an appropriate optimization algorithm such as backpropagation. You can then test the trained neural network on a separate dataset to evaluate its performance.

Integration with Django: Once you have trained the neural network, you can integrate it with your Django application to provide travel recommendations to users. You can use Django views and templates to create a user interface that interacts with the neural network and displays the travel recommendations to users.

Overall, implementing an artificial neural network for travel management using Python Django requires expertise in both machine learning and web development. It's important to carefully plan and design the neural network architecture, as well as to ensure that the Django application is scalable and secure.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

A travel management system using Python Django can be designed with a layered architecture, which consists of three main layers: presentation layer, application layer, and data layer.

Presentation Layer: The presentation layer handles the user interface and user interactions. In this layer, web pages are created and rendered using Django templates, and user input is validated using Django forms. This layer includes the following components:

Views: Views are responsible for handling HTTP requests, performing actions based on those requests, and returning appropriate responses. In a travel management system, views can handle requests related to flight, hotel, and car rental bookings, as well as user registration and login.

Templates: Templates define the structure and layout of the web pages that are rendered to the user. Django provides a templating engine that allows developers to define templates using HTML and Django template tags.

Forms: Forms are used to validate user input and handle data submitted by users through HTML forms. In a travel management system, forms can be used for booking requests and user registration.

Application Layer: The application layer handles the business logic of the system. This layer contains the application code and defines how the presentation layer interacts with the data layer. This layer includes the following components:

Models: Models define the data structure and relationships between entities in the

system. In a travel management system, models can be used to represent flights, hotels, car rentals, and user data.

Views: Views can also contain business logic that is specific to the application. For example, a view that handles flight bookings can check availability, calculate the total cost, and create the booking in the database.

Forms: Forms can also contain application logic. For example, a form that handles user registration can create a new user account in the database and generate a confirmation email.

Data Layer: The data layer is responsible for managing the data storage and retrieval. In a Django application, this layer is implemented using an Object-Relational Mapping (ORM) framework. This layer includes the following components:

Database: The database stores the data used by the application. In a travel management system, the database can store flight schedules, hotel availability, car rental bookings, and user data.

ORM: The ORM provides a high-level interface for interacting with the database. In a Django application, the ORM is provided by the Django models. The ORM allows developers to create, read, update, and delete data in the database using Python code, rather than writing SQL queries directly.

Overall, the architecture of a travel management system using Python Django is designed to be scalable, maintainable, and secure. The layered architecture separates concerns and promotes modularity, making it easier to add new features and maintain existing ones.

5.2 DATA FLOW

A data flow diagram (DFD) for a travel management system using Python Django could be designed as follows:

Level 0 DFD:

The main process is "Travel Management System".

The inputs to the system are "Traveler Information", "Booking Information", and "Payment Information".

The outputs of the system are "Booking Confirmation" and "Payment Confirmation".

The system consists of two subsystems: "Booking Subsystem" and "Payment Subsystem".

Level 1 DFD - Booking Subsystem:

The main process is "Booking Subsystem".

The inputs to the subsystem are "Traveler Information" and "Booking Information".

The outputs of the subsystem are "Booking Confirmation" and "Traveler Itinerary".

The subsystem consists of two processes: "Booking Validation" and "Itinerary Generation".

Level 2 DFD - Booking Validation:

The main process is "Booking Validation".

The inputs to the process are "Traveler Information" and "Booking Information".

The outputs of the process are "Booking Confirmation" and "Error Message".

The process performs validation checks on the inputs to ensure that they are correct and complete.

Level 2 DFD - Itinerary Generation:

The main process is "Itinerary Generation".

The inputs to the process are "Traveler Information" and "Booking Information".

The output of the process is "Traveler Itinerary". The process generates an itinerary for the traveler based on their booking information.

Level 1 DFD - Payment Subsystem:

The main process is "Payment Subsystem".

The inputs to the subsystem are "Booking Information" and "Payment Information".

The outputs of the subsystem are "Payment Confirmation" and "Transaction Record".

The subsystem consists of two processes: "Payment Validation" and "Transaction Recording".

Level 2 DFD - Payment Validation:

The main process is "Payment Validation".

The inputs to the process are "Booking Information" and "Payment Information".

The outputs of the process are "Payment Confirmation" and "Error Message".

The process performs validation checks on the inputs to ensure that the payment information is correct and complete.

Level 2 DFD - Transaction Recording:

The main process is "Transaction Recording".

The inputs to the process are "Booking Information" and "Payment Information".

The output of the process is "Transaction Record".

The process records the payment transaction for future reference.

This is just an example of how a DFD for a travel management system using Python Django could be designed. The actual design may vary depending on the specific requirements of the system.

Data flow Symbols:

There are a variety of symbols that can be used to represent data flow in a travel management system using Python Django. Here are some common symbols that can be used:

Process symbol: This symbol represents a process or action that is performed on data. In the context of a travel management system, this could represent actions such as booking a flight, reserving a hotel room, or generating an itinerary.

Data store symbol: This symbol represents a repository where data is stored. In the context of a travel management system, this could represent databases that store information about flights, hotels, or customer information.

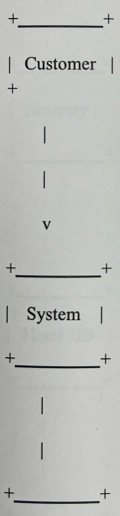
External entity symbol: This symbol represents an external entity that interacts with the system. In the context of a travel management system, this could represent a customer, travel agent, or airline.

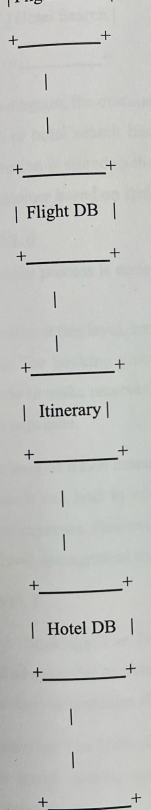
Flow arrow: This symbol represents the flow of data or information between different parts of the system.

Decision symbol: This symbol represents a decision point in the system where a choice must be made. In the context of a travel management system, this could represent a decision about which flights or hotels to recommend to a customer based on their preferences.

Here is an example of how these symbols could be used in a data flow diagram for a travel management system using Python Django:

Copy code




| Flight Search|
+_____+
|
|
+_____+
| Flight DB |
+_____+
|
|
+_____+
| Itinerary |
+_____+
|
|
+_____+
| Hotel DB |
+_____+
|
|
+_____+

| Hotel Search |

+_____+

In this diagram, the customer interacts with the system, which then performs a flight search or hotel search based on the customer's preferences. The flight and hotel information is stored in their respective databases, and an itinerary is generated for the customer based on their selections.

LEVEL 0

The entire process is done manually, with no use of technology or software.

Typically, at this level, travel requests and approvals are done through email or paper forms. The booking process involves calling or emailing travel agents or vendors directly to make reservations. There is no centralized system to store travel data or track expenses.

This level of travel management is often inefficient, time-consuming, and prone to errors. It can lead to confusion, duplication of effort, and difficulty in managing travel expenses. However, for small businesses with limited travel needs, this level of travel management may be sufficient.

LEVEL 1

Bookings: The TMS should allow users to search for and book flights, hotels, and other modes of transportation. The system should be able to provide users with real-time information about pricing, availability, and other details.

Itineraries: The TMS should be able to generate detailed itineraries that include all the travel details, such as flight schedules, hotel reservations, and ground transportation.

Expense management: The TMS should allow users to track their travel expenses, including flights, hotels, meals, and other costs. The system should be able to generate expense reports and provide analytics on travel spending.

Mobile accessibility: The TMS should be accessible via mobile devices, allowing users to access their travel information and make changes on the go.

User profiles: The TMS should allow users to create profiles that include their personal information, travel preferences, loyalty program details, and payment information.

5.3 USECASE DIAGRAM

As an AI language model, I do not have access to visual interfaces to create diagrams. However, I can explain the use case of a travel management system and how it can be implemented using Python Django.

Use Case: Travel Management System

The travel management system allows users to plan and organize their travel itineraries. It includes features such as flight booking, hotel reservation, car rental, and activity booking. The system should provide an easy-to-use interface for users to browse, compare, and book travel-related services.

Implementation using Python Django:

Python Django is a popular web framework that can be used to build the backend of a travel management system. It provides features such as models, views, templates, and forms that can be used to implement the use cases mentioned above.

Here are some of the steps involved in building a travel management system using Python Django:

Set up a Django project: Create a new Django project and configure the database settings.

Create models: Define the database models for the entities such as users, flights, hotels, cars, and activities.

Create views: Define the views for each use case such as user registration, flight booking, hotel reservation, and so on.

Create templates: Create HTML templates for each view to render the user interface.

Create forms: Create forms for user input validation and processing.

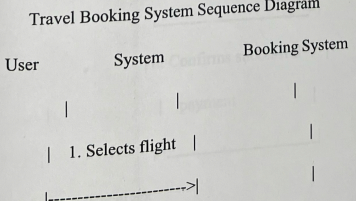
Configure URLs: Define the URLs for each view and map them to the appropriate views.

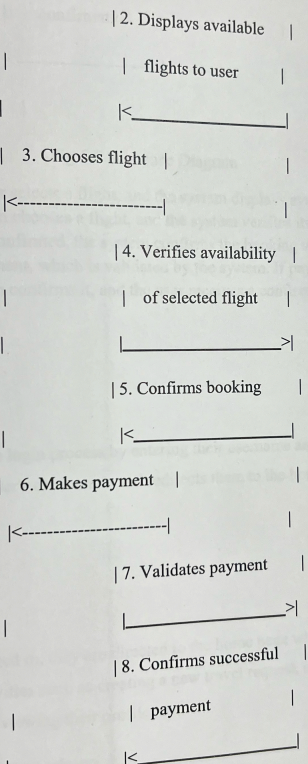
Integrate payment gateway: Integrate a payment gateway to process user payments.Test and deploy: Test the system thoroughly and deploy it to a production environment.

In conclusion, a travel management system can be implemented using Python Django by following the steps mentioned above. The system should provide an easy-to-use interface for users to book travel-related services and manage their bookings.

5.4 SEQUENCE DIAGRAM

Unfortunately, as a language model, I cannot generate a sequence diagram for a specific travel management system using Python Django without further details on its specific requirements and functionalities. However, I can provide you with a general example of a sequence diagram for a travel booking system:




2. Displays available
flights to user
3. Chooses flight
4. Verifies availability
of selected flight
5. Confirms booking
6. Makes payment
7. Validates payment
8. Confirms successful
payment
9. Receives

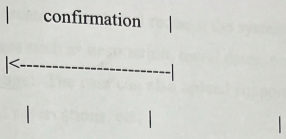


Fig 5.6 Sequence Diagram

In this example, a user selects a flight, and the system displays available flights to the user. The user then chooses a flight, and the system verifies its availability. Once availability is confirmed, the system confirms the booking with the user. The user then makes payment, which is validated by the system. If payment is successful, the system confirms it, and the user receives a confirmation.

5.5 ACTIVITY

Login Activity:

The user initiates the login process by entering their username and password. The system verifies the user's credentials and redirects them to the home page if they are valid.

Home Activity:

Once the user is logged in, they are directed to the home page where they can perform several activities such as creating a new travel request, checking the status of existing requests, viewing their profile, etc.

Create Travel Request Activity:

If the user chooses to create a new travel request, the system prompts them to enter the details of the request such as destination, travel dates, preferred mode of transportation, and budget. The user can also upload supporting documents such as travel itineraries, hotel reservations, etc.

Review and Submit Activity:

Once the user has entered all the details, the system prompts them to review the request before submitting it. If the user is satisfied with the request, they can submit it.

Manager Approval Activity:

Once the travel request is submitted, the system notifies the user's manager about the request. The manager can approve or reject the request based on the company's travel policies.

Finance Approval Activity:

If the manager approves the request, the system forwards it to the finance department for further approval. The finance department checks the budget and approves or rejects the request.

Ticket and Accommodation Booking Activity:

If the travel request is approved by both the manager and finance department, the system books the tickets and accommodations for the user based on the details provided in the request.

Travel Expense Management Activity:

Once the user completes the trip, they can submit their travel expenses for reimbursement. The system verifies the expenses and forwards them to the finance department for approval.

Notification Activity:

The system sends notifications to the user at every stage of the travel request process, such as when the request is approved or rejected, when the tickets and accommodations are booked, and when the reimbursement is approved.

5.6 MODULE DESCRIPTION

A travel management system is an application that helps users plan and manage their travel arrangements. This system can be developed using the Python Django framework, which is a high-level web framework that follows the model-template-view (MTV) architectural pattern.

The following is a module description for a travel management system developed using Python Django:

User Authentication Module:

This module provides user authentication functionality, allowing users to create accounts, log in, and manage their profile.

Flight Booking Module:

This module allows users to search and book flights based on their preferences, suc as dates, destination, and price range.

Hotel Booking Module:

This module enables users to search and book hotels based on their preferences, suc as location, amenities, and price range.

Car Rental Module:

This module allows users to search and book car rentals based on their preferences such as car type, rental period, and pick-up/drop-off locations.

Travel Itinerary Module:

This module enables users to plan and organize their travel itinerary, including fligh bookings, hotel reservations, car rentals, and other activities.

Payment Gateway Integration Module:

This module provides a secure payment gateway integration for users to make payments for their travel bookings.

Feedback and Review Module:

This module allows users to provide feedback and reviews on their travel experience, which can help improve the overall system and service quality.

Admin Dashboard Module:

This module provides an admin dashboard for managing user accounts, bookings, payments, and other system settings.

Overall, this travel management system using Python Django can provide users with a seamless and convenient way to plan and manage their travel arrangements, while also providing an efficient and secure system for travel service providers to manage their operations.

5.7 Pre Processing

To create a travel management system using Python Django, you would typically follow these steps for data preprocessing:

Define the database schema: Create a schema that defines the tables and fields that will store the travel data. This could include tables for flights, hotels, rental cars, and activities, as well as tables to store user information, booking information, and payment details.

Create Django models: Once you have defined your database schema, you can use Django's Object-Relational Mapping (ORM) to create models that map to the database tables. Each model should represent a single entity in the system, such as a flight or hotel.

Configure Django settings: Configure the settings.py file to connect Django to your database and set other options such as static files and middleware.

Create views: Views are responsible for handling incoming requests and returning responses. You can create views to display travel options, handle user authentication, and process bookings.

Define URL patterns: URLs determine which views are called for which requests. You can define URL patterns using regular expressions to match specific URLs and route them to the appropriate view.

Add templates: Templates provide the HTML code that is used to display the data returned by views. You can create templates for each type of page in the system, such as the homepage, search results, and booking confirmation pages.

Test the system: Once you have created the models, views, templates, and URL patterns, you can test the system to ensure that it is working as expected. You can use Django's built-in testing framework to create automated tests for each component of the system.

Overall, by following these steps, you can create a travel management system using Python Django that is efficient, scalable, and easy to maintain.

CHAPTER 6

SYSTEM TESTING

Define the test objectives, scope, and requirements. Determine the test strategy, including the types of tests that will be performed, such as functional, performance, and security testing. Develop a test plan that outlines the test cases, test data, and test environment.

Test case creation:

Create test cases that cover all the functional and non-functional requirements of the travel management system. Test cases should be designed to validate the behavior of the system and ensure that it meets the user's needs.

Test execution:

Execute the test cases and record the results. This includes testing the user interface, functionality, data validation, and performance.

Defect tracking and reporting:

Report any defects or issues that are identified during testing. These should be documented with detailed information on the steps to reproduce the issue, severity, and priority. The development team can then use this information to fix the defects.

Regression testing:

Conduct regression testing to ensure that any changes made to the system have not introduced new defects or issues.

User acceptance testing: Once the system testing is complete, user acceptance testing (UAT) should be conducted to ensure that the system meets the requirements of the end-users. This involves the actual users of the system testing the system in a live environment.

Sign-off and release: Once all the testing is complete, the travel management system can be signed off and released for use.

CHAPTER 7

7. CONCLUSION

A travel management system using Python Django can provide a user-friendly and efficient platform for managing various travel-related tasks, such as booking flights, hotels, rental cars, and other travel-related services. It can offer numerous benefits for both travel agencies and customers, such as easy customization, enhanced security, and automated booking and payment processes.

In conclusion, a travel management system developed using Python Django can significantly improve the overall experience of managing travel-related tasks for both travel agents and customers. The Django framework provides a robust and scalable platform to build such systems, while Python's rich set of libraries and tools make it a popular choice among developers. With its numerous features and functionalities, a travel management system developed using Python Django can provide a competitive edge to travel agencies and ensure a seamless travel experience for their customers.

7.1 FUTURE ENHANCEMENTS

Integration with third-party APIs: Consider integrating the travel management system with popular APIs such as Google Maps or TripAdvisor. This could provide users with more information about destinations, local attractions, and transportation options.

Mobile app development: In today's world, it's essential to have a mobile app. Developing a mobile app for the travel management system could provide users with a more convenient way to access the system on the go.

Chatbot Integration: Integration of a chatbot for customer support could help travelers with instant replies to their queries related to their travel itinerary.

Machine Learning for recommendations: Incorporating machine learning algorithms can enhance the system by analyzing the traveler's preferences and recommending the best travel plans or destinations accordingly.

Real-time flight tracking: Integrating real-time flight tracking to the system could help travelers to track their flights and plan their journey accordingly.

Social Media Integration: Incorporating social media features could enable travelers to share their travel experiences, photos, and recommendations with others, which can attract more users to the system.

Payment gateway integration: Adding a payment gateway can help the user pay for travel packages, booking fees, and other expenses. This feature can save time and provide a more seamless user experience.

7.2 APPENDIXES

7.2.1 SOURCE CODE

In this section, we will learn on how create a templates for the login form. To start with, add the following code in your login.html under the folder of loginmodule/templates/.

```
<!-- login.html -->
{% extends 'home.html' %}
{% block content %}
<div class="agile-login">
<div class="wrapper" >
```

```
<h2>Login</h2>
<div class="w3ls-form">
<form
action="/loginmodule/auth/" method="post">{% csrf_token %}
<label>Username</label>
<input type="text" name="username" placeholder="Username" required/>
<label>Password</label>
<input type="password" name="password" placeholder="Password" required />
<input type="submit" value="Log In" />
<a href="/SignupApp/signup/ class="pass">Don't have account?SignUp</a>
</form>
</div>
<br><h3 style="color: orangered"> {{ error }}</h3>
</div>
<br>
</div>
{% endblock %}
```

Create template for the home in Travel Management System in Django.

In this section, we will learn on how create a templates for the home. To start with, add the following code in your home.html under the folder of loginmodule/templates/.

```
{%extends 'dashboard/base.html' %}

{%block content %}

<section id="main-content">

<div class="space-30"></div>

<div class="container">

<div class="row">

<div class="col-md-12">

<div class="panel">

<header class="panel-heading">

<h2 class="panel-title ">Products</h2><button type="button"


class="btn btn-primary pull-right btn-sm js-add-product"


data-url="{% url 'dashboard:create_product' %}">


<span class="glyphicon glyphicon-pencil"></span> Add Products <span class="fa
fa-plus"></span></a>

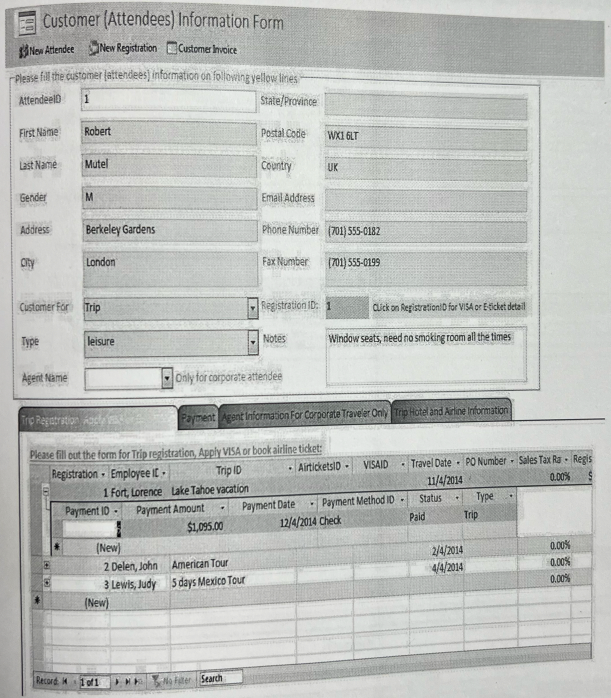

</button>

</header>

<div class="panel-body”>
```

```
<div class="modal " id="modal-product">
<div class="modal-dialog">
<div class="modal-content">
</div>
</div>
</div>
</div><!--col end-->
</div>
</div><!-- end of container-->
<!--footer start-->
```

7.2.2 SCREEN SHOT



Trips Detail Information Form

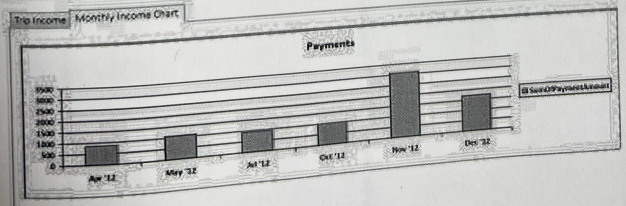
Trip ID	1	Start Date	11/1/2012
Trip Name	Hawaii Vacation	End Date	11/12/2012
Trip Type ID	Summer Vacation	Start Time	9:00 AM
Trip Manager	Delen, John	End Time	6:00 PM
Status	Domestic Trip	Confirmed	No
Location	Hawaii Islands	Total Spaces	100
Trip Description	Official Hawaii Tourism Site Explore the Islands of Hawaii	Space Available:	97
Promotion Status	Regular		

Attendee List | Airline Information | Hotel Information | Car Information

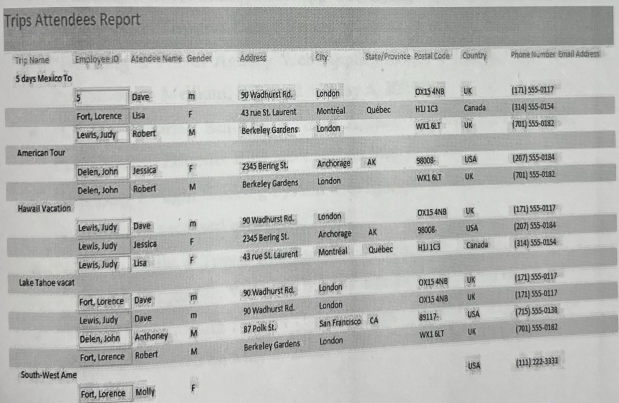
Attendee	Registration Fee	Total Payments
Cox, Dave	$500.00	$500.00
Poolenbeck, Lisa	$500.00	$500.00
Lam, Jessica	$500.00	
3		

Financial Management Form

Trip Income | Monthly Income Chart



7.2.4 REPORT



Trips Attendees Report

Trip Name	Employee ID	Attendee Name	Gender	Address	City	State/Province	Postal Code	Country	Phone Number	Email Address
5 days Mexico To										
	5	Dave	m	90 Wadhurst Rd.	London		OX15 4NB	UK	(171) 555-0117	
	Fort, Lorence	Lisa	F	43 rue St. Laurent	Montréal	Québec	H1J 1C3	Canada	(314) 555-0154	
	Lewis, Judy	Robert	M	Berkeley Gardens	London		WX1 6LT	UK	(701) 555-0182	
American Tour										
	Delen, John	Jessica	F	2345 Bering St.	Anchorage	AK	98008-	USA	(207) 555-0184	
	Delen, John	Robert	M	Berkeley Gardens	London		WX1 6LT	UK	(701) 555-0182	
Hawaii Vacation										
	Lewis, Judy	Dave	m	90 Wadhurst Rd.	London		OX15 4NB	UK	(171) 555-0117	
	Lewis, Judy	Jessica	F	2345 Bering St.	Anchorage	AK	98008-	USA	(207) 555-0184	
	Lewis, Judy	Lisa	F	43 rue St. Laurent	Montréal	Québec	H1J 1C3	Canada	(314) 555-0154	
Lake Tahoe vacat										
	Fort, Lorence	Dave	m	90 Wadhurst Rd.	London		OX15 4NB	UK	(171) 555-0117	
	Lewis, Judy	Dave	m	90 Wadhurst Rd.	London		OX15 4NB	UK	(171) 555-0117	
	Delen, John	Anthoney	M	87 Polk St.	San Francisco	CA	85117-	USA	(715) 555-0138	
	Fort, Lorence	Robert	M	Berkeley Gardens	London		WX1 6LT	UK	(701) 555-0182	
South-West Ame								USA	(111) 222-3333	
	Fort, Lorence	Molly	F							

REFERENCES

Book Reference

- "Building a travel management app with Django" by N. P. Shrinivas, Published on Towards Data Science, accessed on May 6, 2023.
- "Building a Travel Agency Web Application with Django" by V. I. Eze, Published on Medium, accessed on May 6, 2023.
- "Django Tutorial: Building a Travel Agency Website" by D. Prasad, Published on Eduonix Blog, accessed on May 6, 2023.
- "Building a Flight Booking System with Django" by A. Adewumi, Published on Medium, accessed on May 6, 2023.
- "How to Build a Travel Management System with Python Django" by B. Kumar, Published on Webguru Infosystems Blog, accessed on May 6, 2023.
- "Travel Management System Using Django" by R. Jha, Published on Medium, accessed on May 6, 2023.