



DAVANGERE UNIVERSITY  
DON BOSCO DEGREE COLLEGE  
CHITRADURGA-577501



PROJECT REPORT ON

## **TRAVEL MANAGEMENT SYSTEM**

A Project submitted for the partial fulfilment of the requirements for qualifying

**BACHELOR IN COMPUTER APPLICATION**

SUBMITTED BY

**DON BOSCO K JOSE**

REG NO: U13D021S0032

UNDER THE GUIDENCE OF

**Miss SOWMYA M S** B.E., M.Tech(CS),

**ASSISTANT PROFESSOR**

BACHELOR IN COMPUTER APPLICATION  
DON BOSCO DEGREE COLLEGE  
CHITRADURGA-577501

**2023-2024**

## **CERTIFICATE**

This is to certify that the project entitled "**TRAVEL MANAGEMENT SYSTEM**" carried out by **DON BOSCO K JOSE (U13DO21S0032)** student of **DON BOSCO DEGREE COLLEGE, CHITRADURGA**. In partial fulfilment of the requirements for the award of **BACHELOR IN COMPUTER APPLICATION (BCA)** of the **DAVANGERE UNIVERSITY, CHITRADURGA** during the year 2023-2024. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

PRINCIPAL SIGNATURE

DON BOSCO DEGREE COLLEGE

CHITRADURGA-577501

## **CERTIFICATE**

This is to certify that the project entitled " **TRAVEL MANAGEMENT SYSTEM**" carried out by **DON BOSCO K JOSE (U13DO21S0032)** student of **DON BOSCO DEGREE COLLEGE, CHITRADURGA**. In partial fulfilment of the requirements for the award of **BACHELOR IN COMPUTER APPLICATION (BCA)** of the **DAVANGERE UNIVERSITY, CHITRADURGA** during the year 2023-2024. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

---

HOD Signature

**Mr. R. Poobalan., M.Sc.,M.Ed**

---

Guide Signature

**Miss. Sowmya M S., B.E., M.Tech(CS)**

## **CERTIFICATE**

This is to certify that the project entitled " **TRAVEL MANAGEMENT SYSTEM**" carried out by **DON BOSCO K JOSE (U13DO21S0032)** student of **DON BOSCO DEGREE COLLEGE, CHITRADURGA**. In partial fulfilment of the requirements for the award of **BACHELOR IN COMPUTER APPLICATION (BCA)** of the **DAVANGERE UNIVERSITY, CHITRADURGA** during the year 2023-2024. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

External Examiner

Date :

\_\_\_\_\_

Internal Examiner

Date:

\_\_\_\_\_

## **DECLARATION**

I hereby declare that the dissertation work entitled "**TRAVEL MANAGEMENT SYSTEM**" which is being submitted by me in the partial fulfilment for the award of the degree of Bachelor Of Computer Applications from **DAVANGERE UNIVERSITY**, Tholahunase Davangere is an authentic record of my own independent work carried out by me during the academic year 2023-2024, under the valuable guidance of Lecturer **Miss. Sowmya M S., B.E., M.Tech(CS)**, Department of Computer Application of **DON BOSCO DEGREE COLLEGE CHITRADURGA**. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree and due acknowledgement has been made in the text and all other material used.

**DON BOSCO K JOSE**

Date:

Place: CHITRADURGA

# ACKNOWLEDGEMENT

The completion of this brings a sense of satisfaction, but it is never completed without thanking the persons who responsible for the successful completion.

First and foremost I wish to express my thanks to the almighty God who is the real one to help me through the preparation of this project.

I extend my sincere gratitude to our former-Principal Dr. Fr. JOMON KOKKANDATHIL, The Principal Fr. BENNY C and Vice Principal Fr. CHRISTY JOSEPH for their motivation and encouragement.

I thank our beloved class In-charge Miss. SOWMYA M.S., M.Tech., for being the source of inspiration.

I would like to extend my thanks to Mr. R POOBALAN., M.Sc., M.Ed., Head of the Department of Computer Application for his motivation and encouragement throughout the project work.

I express my thanks to Miss. SOWMYA M S., B.E., M.Tech(Cs)., Assistant professor and my Internal guide for his valuable guidance for my project in a much determined way within a stipulated time.

I also wish to express our sincere feelings of gratitude to our DON BOSCO DEGREE COLLEGE, CHITRADURGA for providing support during this project.

I thank my family members and friends for their immersive encouragement and all of family and friends for their help rendered to me in many ways to complete the project.

I am thankful to all those who have directly helped us in successful completion of our project.

# ABSTRACT

The web-based travel agency management system built with Flask and MySQL represents a significant leap forward in the efficiency of travel service operations. Developed to streamline tasks such as managing travel packages, guide assignments, customer registrations, and package opt-ins, its primary goal is to offer a comprehensive and intuitive platform for both travel agencies and customers. The MySQL database design plays a pivotal role, meticulously structured to manage data for agencies, customers, packages, guides, assignments, and opt-ins while ensuring data integrity through foreign key constraints. This foundational structure supports seamless system operation, enabling efficient data retrieval and management.

Flask, chosen for its lightweight and flexible nature, empowered the development of dynamic routes and endpoints for key functionalities like user authentication, package management, and customer interactions. Utilizing session management via global variables such as `authenticated_agency_id` and `authenticated_customer_id` ensures secure, personalized user interactions. Despite successful core feature implementation, future enhancements are envisioned, including real-time updates using WebSocket technology for instant notifications on package availability and booking confirmations. Integrating secure payment gateways directly into the platform and incorporating analytics for operational insights and revenue trends are also planned to further optimize service delivery and customer experience.

In conclusion, this travel agency management system showcases technology's transformative impact on the travel industry. Leveraging Flask and MySQL, it provides a robust, scalable solution that enhances management capabilities for travel agencies while improving overall user experience. Continual enhancement promises to solidify its position as a leading solution in the travel market, driving industry growth and innovation through data-driven decision-making and optimized service offerings.

# TABLE OF CONTENT

S.NO	CONTENTS	PAGE NO
<b>1</b>	<b>INTRODUCTION</b>	<b>1-8</b>
1.1	INTRODUCTION	2
1.2	INTRODUCTION TO PYTHON PROGRAMMING	2
1.3	HISTORY OF PYTHON	2-3
1.4	APPLICATION OF PYTHON	4-3
1.5	INTRODUCTION TO DATABASE MANAGEMENT SYSTEM	4-5
1.6	HISTORY OF DBMS	5
1.7	APPLICATIONS OF DBMS	6
1.8	THEORY AND CONCEPTS	6
1.9	REQUIREMENT SPECIFICATION	7-8
<b>2</b>	<b>SYSTEM STUDY</b>	<b>9-17</b>
2.1	FEATURES OF THE SYSTEM	10-11
2.2	DRAWBACKS OF THE SYSTEM	11-12
2.3	SOFTWARE DESCRIPTION	12-15
2.4	FUNCTION AND REQUIREMENT	15-16
2.5	NON-FUNCTIONAL REQUIREMENT	16-17



# TABLE OF CONTENT

S.NO	CONTENTS	PAGE NO
<b>3</b>	<b>SYSTEM DESIGN</b>	<b>18-32</b>
3.1	ER DIAGRAM	19
3.2	MAPPING FROM ER DIAGRAM TO SCHEMA DIAGRAM	19-22
3.3	NORMALIZATION	22-24
3.4	ASSUMPTIONS	24
3.5	SCHEMA DIAGRAM	24-25
3.6	INPUT DESIGN OF THE SYSTEM	25-30
3.7	OUTPUT DESIGN OF THE SYSTEM	30-32
<b>4</b>	<b>METHODOLOGY</b>	<b>33-37</b>
4.1	STAGE OF THE ITERATIVE DEVELOPMENT PROCESS	34-35
4.2	ADVANTAGES OF ITERATIVE DEVELOPMENT	36
4.3	DISADVANTAGES OF ITERATIVE DEVELOPMENT	37
<b>5</b>	<b>TESTING</b>	<b>38-42</b>
5.1	INTRODUCTION TO TESTING	39
5.2	TYPES OF TESTING	39-40
5.3	TESTING PERFORMED	40-42

# TABLE OF CONTENT

S.NO	CONTENTS	PAGE NO
<b>6</b>	<b>IMPLEMENTATION</b>	<b>43-49</b>
6.1	TABLES USED IN DATABASE	44-47
6.2	DATABASE CONNECTION	47
6.3	INSERT OPERATION	47-48
6.4	RETRIEVE OPERATION	48
6.5	UPDATE OPERATION	48
6.6	DELETE OPERATION	49
<b>7</b>	<b>SNAPSHOTS</b>	<b>50-56</b>
7.1	HOME PAGE	51
7.2	CUSTOMER LOGIN	52
7.3	CUSTOMER SIGNUP PAGE	52
7.4	AGENCY LOGIN PAGE	53
7.5	AGENCY SIGNUP PAGE	53
7.6	AGENCY VIEW PAGE	54
7.7	CUSTOMER VIEW PAGE	54
7.8	DISPLAY AGENCY LIST	55
7.9	DISPLAY PACKAGE LIST	55
7.10	ASSIGNING GUIDE	56
<b>8</b>	<b>CONCLUSION</b>	<b>57</b>
8.1	CONCLUSION	58
8.2	FUTURE WORKS	58
<b>9</b>	<b>BIBLIOGRAPHY</b>	<b>59</b>

# LIST OF FIGURES

S.NO	CONTENTS	PAGE NO
3.1	ER DIAGRAM FOR AUTOMATED TIMETABLE GENERATION SYSTEM	19
3.2	MAPPING OF REGULAR ENTITIES	20
3.3	MAPPING BINARY 1:N	21
3.4	MAPPING OF BINARY M:N RELATIONSHIPS	21
3.5	PACKAGE NORMALIZATION	22
3.6	AGENCY NORMALIZATION	22
3.7	GUIDE NORMALIZATION	23
3.8	CUSTOMER NORMALIZATION	23
3.9	OPT NORMALIZATION	23
3.10	ASSIGN NORMALIZATION	24
3.11	SCHEMA DIAGRAM FOR TRAVEL MANAGEMENT SYSTEM	25
4.1	ITERATIVE METHODOLOGY	34
7.1	HOME PAGE	51
7.2	CUSTOMER LOGIN	52
7.3	CUSTOMER SIGNUP PAGE	52
7.4	AGENCY LOGIN PAGE	53
7.5	AGENCY SIGNUP PAGE	53
7.6	AGENCY VIEW PAGE	54
7.7	CUSTOMER VIEW PAGE	54
7.8	DISPLAY AGENCY	55
7.9	DISPLAY PACKAGES	55
7.10	ASSIGNING GUIDE	56

# CHAPTER 1

# INTRODUCTION

## 1.1 INTRODUCTION

In an era marked by unprecedented connectivity and mobility, the travel industry stands at the forefront of global interaction, offering individuals the opportunity to explore diverse destinations and cultures. However, navigating the plethora of travel options and packages presented by numerous agencies can often be overwhelming for customers seeking their ideal travel experience. In response to this challenge, the development of a comprehensive Travel Management System emerges as a pivotal solution, bridging the gap between customers and travel agencies while facilitating seamless view of a variety of options for the customers. In essence, the Travel Management System not only serves as a solution to the challenges posed by the abundance of travel options but also embodies the transformative potential of technology in revolutionizing the travel industry. By fostering connectivity, simplifying processes, and enhancing customer satisfaction, the System emerges as a cornerstone of modern travel management, driving growth, innovation, and excellence in the ever-expanding global tourism landscape. The travel industry continues to expand, driven by factors such as globalization, increasing disposable income, and a growing desire for experiential travel. As more people seek personalized and convenient travel experiences, platforms like ours will facilitate connections between customers and agencies which is very valuable.

## 1.2 INTRODUCTION TO PYTHON PROGRAMMING

Python is a high-level, interpreted programming language that has gained immense popularity due to its simplicity, readability, and versatility. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its use of significant indentation. This makes Python particularly attractive for beginners, as its syntax is clear and straightforward. Python is dynamically typed, meaning that variable types are determined at runtime, allowing for greater flexibility in coding. It boasts an extensive standard library that supports a wide array of programming tasks, from web development and data analysis to artificial intelligence and scientific computing. Python is also cross-platform, enabling code to run on various operating systems such as Windows, macOS, and Linux. The language has a large and active community, contributing to a rich ecosystem of libraries and frameworks like Django for web development, Pandas for data manipulation, and TensorFlow for machine learning. These features make Python a powerful tool for both simple scripting and complex application development, fostering a supportive environment for developers of all levels.

## 1.3 HISTORY OF PYTHON

Python, created by Guido van Rossum, was first released in 1991. Its development began in the late 1980s at Centrum Wiskunde & Informatica (CWI) in the Netherlands, where Van Rossum was inspired by the ABC language, which emphasized simplicity and readability. Python was designed to be easy to understand and use, with a focus on code readability and a syntax that allowed programmers to express concepts in fewer lines of code compared to other languages like C++ or Java.

The first official version, Python 1.0, was released in January 1994. This version already included many of Python's core features such as exception handling, functions, and the core data types: lists, dictionaries, and strings. Over the next few years, Python evolved with significant contributions from its growing community.

In October 2000, Python 2.0 was released, introducing list comprehensions, garbage collection, and the first version of the `print` statement as a function. Python 2.x versions continued to improve the language, but certain design flaws and the need for more modern features led to the development of Python 3.

Python 3.0, released in December 2008, was a major overhaul aimed at removing redundant features and fixing inconsistencies in the language design. This new version was not backward-compatible with Python 2.x, which initially caused some resistance in the community. However, Python 3 introduced several new features, such as a more consistent syntax, improvements to the standard library, and enhancements in performance.

Over the years, Python 3.x has become the standard, with Python 2 reaching its end of life in January 2020. The transition from Python 2 to Python 3 was supported by the community through tools like `2to3`, which helped automate code conversion.

Today, Python continues to grow in popularity due to its simplicity and versatility. It is widely used in web development, scientific computing, data analysis, artificial intelligence, and more. The language's active community, extensive libraries, and frameworks contribute to its ongoing evolution and success, ensuring that Python remains a top choice for developers around the world.

## 1.4 APPLICATION OF PYTHON

Python is renowned for its versatility and wide range of applications across various domains. Here are some of the key areas where Python is extensively used:

- **Web Development:** Python provides robust frameworks like Django, Flask, and Pyramid that streamline web development by offering powerful tools and libraries for building scalable, secure, and maintainable web applications. Django, for instance, is known for its "batteries-included" philosophy, offering built-in features like authentication, database management, and an admin interface.
- **Data Science and Analytics:** Python has become the go-to language for data scientists due to its simplicity and the extensive range of libraries available for data analysis, visualization, and machine learning. Libraries such as Pandas, NumPy, and Matplotlib facilitate data manipulation and visualization, while Scikit-learn and TensorFlow are widely used for machine learning and deep learning applications.
- **Artificial Intelligence and Machine Learning:** Python's simplicity and the powerful libraries it offers make it ideal for AI and ML projects. TensorFlow, Keras, and PyTorch are popular frameworks that enable the development of complex neural networks and deep learning models. These libraries provide pre-built models and tools for tasks like natural language processing, image recognition, and predictive analytics.

- **Automation and Scripting:** Python's ease of use and readability make it a perfect choice for writing scripts to automate repetitive tasks. It is commonly used for automating system administration tasks, data scraping from websites, and automating software testing processes. Libraries like Selenium and BeautifulSoup enhance its capabilities in web scraping and browser automation.
- **Game Development:** Python is used in game development for prototyping and developing simple games. Libraries like Pygame provide functionality for game development, including handling graphics, sound, and user input.
- **Education:** Python's simplicity and readability make it an excellent language for teaching programming and computer science concepts. It is often the first language taught in introductory programming courses around the world.
- **Software Development:** Python is used for developing a variety of software applications, from desktop applications to complex enterprise systems. Its integration capabilities with other languages and tools make it a versatile choice for software development projects.

## 1.5 INTRODUCTION TO DATABASE MANAGEMENT SYSTEM

Databases and database technology have a major impact on the growing use of computers. It is fair to say that databases play a critical role in almost all areas where computers are used, including business, electronic commerce, engineering, medicine, genetics, law, education, and library science. The word database is so commonly used that User must begin by defining what a database is. Our initial definition is quite general. A database is a collection of related data.1 By data, User mean known facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers, and addresses of the people you know. You may have recorded this data in an indexed address book or you may have stored it on a hard drive, using a personal computer and software such as Microsoft Access or Excel. This collection of related data with an implicit meaning is a database. The preceding definition of database is quite general; for example, User may consider the collection of words that make up this page of text to be related data and hence to constitute a database. However, the common use of the term database is usually more restricted.

A database has the following implicit properties:

- A database represents some aspect of the real world, sometimes called the mini world or the universe of discourse. Changes to the inworld are reflected in the database.
- A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database
- A database is designed, built, and populated with data for a specific purpose. It has an intended. group of users and some preconceived applications in which these users are interested.

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users

and applications. Defining a database involves specifying the data types, structures, and constraints of the data to be stored in the database. The database definition or descriptive information is also stored by the DBMS in the form of a database catalogue or dictionary; it is called meta-data. Constructing the database is the process of storing the data on some storage medium that is controlled by the DBMS. Manipulating a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the inworld, and generating reports from the data. Sharing a database allows multi CWC users and programs to access the database simultaneously.

## **1.6 History of DBMS**

In 1959, the TX-2 computer was developed at MIT's Lincoln Laboratory. The TX-2 integrated a number of new man-machine interfaces. A light pen could be used to draw sketches on the computer using Ivan Sutherland's revolutionary Sketchpad software. Using a light pen, Sketchpad allowed one to draw simple shapes on the computer screen, save them and even recall them later. The light pen itself had a small photoelectric cell in its tip. This cell emitted an electronic pulse whenever it was placed in front of a computer screen and the screen's electron gun fired directly at it. By simply timing the electronic pulse with the current location of the electron gun, it was easy to pinpoint exactly where the pen was on the screen at any given moment. Once that was determined, the computer could then draw a cursor at that location. Also, in 1961 another student at MIT, Steve Russell, created the first video game, E. E. Zajac, a scientist at Bell Telephone Laboratory (BTL), created a film called "Simulation of a two-gravity attitude control system" in 1963. During 1970s, the first major advance in 3D computer graphics was created at UU by these early pioneers, the hidden-surface algorithm. In order to draw a representation of a 3D object on the screen, the computer must determine which surfaces are "behind" the object from the viewer's perspective, and thus should be "hidden" when the computer creates (or renders) the image. In the 1980s, artists and graphic designers began to see the personal computer, particularly the Commodore Amiga and Macintosh, as a serious design tool, one that could save time and draw more accurately than other methods. In the late 1980s, SGI computers were used to create some of the first fully computer-generated short films at Pixar. The Macintosh remains a highly popular tool for computer graphics among graphic design studios and businesses. Modern computers, dating from the 1980s often use graphical user interfaces (GUI) to present data and information with symbols, icons and pictures, rather than text. Graphics are one of the five key elements of multimedia technology. 3D graphics became more popular in the 1990s in gaming, multimedia and animation. In 1996, Quake, one of the first fully 3D games, was released. In 1995, Toy Story, the first full-length computer-generated animation film, was released in cinemas worldwide. Since then, computer graphics have only become more detailed and realistic, due to more powerful graphics hardware and 3D modeling software.



## 1.7 APPLICATIONS OF DBMS

Applications where we use Database Management Systems are:

**Telecom:** There is a database to keep track of the information regarding calls made, network usage, customer details etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond.

**Industry:** Where it is a manufacturing unit, warehouse or distribution Centre, each one needs a database to keep the records of ins and outs. For example, distribution Centre should keep a track of the product units that supplied into the Centre as well as the products that got delivered out from the distribution Centre on each day; this is where DBMS comes into picture.

**Banking System:** For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems.

**Education sector:** Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees detail etc. There is a hell lot amount of inter-related data that needs to be stored and retrieved in an efficient manner.

**Online shopping:** You must be aware of the online shopping websites such as Amazon, Flipkart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.

## 1.8 Theory and Concepts

**Inheritance:** In object-oriented programming, inheritance is when an object or class is based on another object or class, using the same implementation (inheriting from an object or class) or specifying a new implementation to maintain the same behavior (realizing an interface). Such an inherited class is called a subclass of its parent class or super class.

**Encapsulation:** In object-oriented programming, encapsulation is a mechanism of binding the data, and the functions together in a class and use them by creating an object of that class.

**Data Abstraction:** Data abstraction refers to, providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the implementation details. Data abstraction is a programming (and design) technique that relies on the separation of interface and implementation.

## 1.9 REQUIREMENT SPECIFICATION

### Hardware Requirements

#### 1. Processor

- **Minimum:** Intel Core i3-1005G1 CPU @ 1.20GHz or equivalent.
- **Recommended:** Intel Core i5 or higher for better performance.

#### 2. RAM

- **Minimum:** 8 GB.
- **Recommended:** 16 GB or more, especially if running additional services or for a higher number of concurrent users.

#### 3. Hard Disk

- **Minimum:** 500 GB HDD.
- **Recommended:** 1 TB or more, SSD for faster data access and improved performance.

#### 4. Input Devices

- **Minimum:** Standard keyboard and mouse.

#### 5. Output Devices

- **Minimum:** Monitor with at least 1080p resolution.
- **Recommended:** Monitor with higher resolution for better user experience.

### Software Requirements

#### 1. Operating System

- **Minimum:** Windows 10, macOS 10.13 High Sierra, or a popular Linux distribution such as Ubuntu 18.04.
- **Recommended:** Latest stable version of the operating system for better security and support.

#### 2. Web Server

- **Minimum:** Flask development server (for small-scale or testing purposes).
- **Recommended:** A production-ready web server like Nginx or Apache for better performance and security.

#### 3. Database

- **Minimum:** MySQL 8.0.36.
- **Recommended:** Latest stable version of MySQL or an equivalent database management system.

#### 4. Programming Language

- **Python Version:** Python 3.11.

#### 5. Integrated Development Environment (IDE)

- **Minimum:** Visual Studio Code.
- **Recommended:** Any other IDE or text editor the developer is comfortable with, such as PyCharm or Sublime Text.

## 6. Browser

- **Minimum:** Modern web browsers like Google Chrome, Mozilla Firefox, Microsoft Edge.
- **Recommended:** Latest versions of these browsers for optimal performance and security.

## 7. Dependencies and Libraries

- **Flask:** For web application development.
- **mysql-connector-python:** For MySQL database connectivity.
- **Jinja2:** For HTML templating.
- **Werkzeug:** For handling HTTP requests.

## 8. Additional Software

- **Git:** For version control.
- **Docker:** For containerization and easier deployment (optional but recommended).

# **CHAPTER 2**

# **SYSTEM STUDY**

## 2.1 FEATURES OF THE SYSTEM

### User Authentication and Authorization

- **Agency Sign-Up and Login:** Allows travel agencies to register and log into the system using unique identifiers.
- **Customer Sign-Up and Login:** Enables customers to create accounts and access the system by logging in.

### Agency Management

- **Add Travel Packages:** Agencies can create and add new travel packages, specifying details like package ID, type, amount, and duration.
- **View Travel Packages:** Agencies can view all the packages they have added, making it easy to manage and update offerings.
- **Assign Guides to Packages:** Agencies can assign guides to specific travel packages and manage these assignments.

### Package Management

- **Display All Packages:** Both customers and agencies can view the list of all available travel packages.
- **Opt for Packages:** Customers can select and opt for specific travel packages they are interested in.
- **View Opted Packages:** Customers can view the list of packages they have opted for, making it easy to track their bookings.
- **Delete Opted Packages:** Customers have the option to cancel their opted packages if needed.

### Guide Management

- **Add Guides:** Agencies can add new guides to the system, including details like guide ID, name, address, and phone number.
- **View Guides:** Agencies can view the list of all guides available, facilitating the assignment process.

### Assignment Management

- **View Assignments:** Agencies can view all current guide assignments, helping them manage who is assigned to which package.
- **Update Assignments:** Allows updating the assignment dates for guides to accommodate changes in scheduling.
- **Delete Assignments:** Agencies can remove guides from packages as necessary.

### Customer Interaction

- **View All Packages:** Customers can browse all available travel packages offered by various agencies.
- **Opt for Packages:** Customers can opt for packages directly through the system, streamlining the booking process.
- **Manage Opted Packages:** Customers can view and manage their opted packages, including the ability to cancel bookings if required.

### Database Integration

- **MySQL Database:** The system uses MySQL for storing and managing data related to agencies, customers, packages, guides, and assignments.

### User Interface

- **HTML Templating:** The system uses HTML templates to render dynamic web pages, providing a user-friendly interface for both agencies and customers.
- **Flask Framework:** Built using the Flask web framework, the system handles HTTP requests and responses efficiently, ensuring smooth operation.

### Security

- **Authentication Mechanisms:** Ensures that only registered users can access their respective functionalities, maintaining the integrity and security of the system.

### Ease of Use

- **Intuitive Navigation:** The system is designed to be user-friendly, with intuitive navigation and straightforward forms for data input.
- **Responsive Design:** Ensures compatibility across different devices, including desktops, tablets, and smartphones.

## 2.2 DRAWBACKS OF THE SYSTEM

### Limited User Role Customization

- **Fixed Roles:** The system currently supports fixed roles for agencies and customers without much flexibility in role customization or the creation of additional user roles.
- **Lack of Fine-Grained Permissions:** There is no support for fine-grained permissions, which could restrict more complex user management scenarios.

### Scalability Issues

- **Single Server Configuration:** As designed, the system may not scale well with a single server configuration. High traffic and a growing number of users might require significant reengineering to support load balancing and distributed database systems.

- **Database Performance:** The use of a single MySQL database without optimization for high-volume operations might lead to performance bottlenecks.

### **Limited Functionality**

- **No Payment Integration:** The system lacks integrated payment processing, which means customers cannot pay for packages directly through the platform.
- **No Real-Time Updates:** The system does not support real-time updates or notifications, which could enhance user experience and provide timely information on bookings and changes.

### **Dependency on Internet Connection**

- **Offline Functionality:** The system requires a constant internet connection to function, which could be a limitation in areas with unreliable internet access

## **2.3 SOFTWARE DESCRIPTION**

### **FRONT END**

#### **1. HTML (HyperText Markup Language)**

**Purpose:** HTML is the fundamental building block of web pages. It defines the structure and layout of a webpage by using a variety of tags and elements.

#### **Key Features:**

- **Tags and Elements:** HTML uses tags like <div>, <p>, <a>, <table>, and many others to structure the content. Each tag serves a specific purpose.
- **Forms:** HTML forms are used to collect user inputs, such as text fields, radio buttons, checkboxes, and submit buttons.
- **Hyperlinks:** Using the <a> tag, HTML enables navigation between different pages of the web application.
- **Multimedia:** HTML supports embedding images, videos, and audio with tags like <img>, <video>, and <audio>.

#### **Role in the System:**

- **Page Layout:** Defines the structure of various pages such as the home page, login/signup pages, package details, and user dashboards.
- **Forms:** Used for agency and customer registration, login forms, package addition forms, etc.
- **Tables:** Displaying lists of agencies, customers, packages, and bookings.

## 2. CSS (Cascading Style Sheets)

**Purpose:** CSS is used to style HTML elements. It defines how HTML elements are displayed, allowing for the separation of content from presentation.

**Key Features:**

- **Selectors:** CSS selectors are used to select the HTML elements you want to style.
- **Box Model:** CSS defines the spacing, border, and padding of HTML elements.
- **Flexbox and Grid:** Modern layout systems for creating responsive and flexible designs.
- **Media Queries:** Used to apply different styles for different devices and screen sizes, making web pages responsive.

**Role in the System:**

- **Visual Design:** Ensures a consistent look and feel across the web application. It styles headers, footers, navigation bars, forms, buttons, and tables.
- **Responsive Design:** Ensures that the web application is usable on different devices, including desktops, tablets, and smartphones.
- **User Experience:** Improves the user experience by providing visual feedback on user interactions, such as hover effects and form validations.

## 3. JAVASCRIPT

**Purpose:** JavaScript is a programming language that enables interactive web pages. It is executed on the client-side (in the user's browser).

**Key Features:**

- **DOM Manipulation:** JavaScript can read and alter the Document Object Model (DOM), allowing dynamic content updates.
- **Event Handling:** JavaScript can respond to user actions like clicks, form submissions, and mouse movements.
- **AJAX:** Asynchronous JavaScript and XML (AJAX) allows for asynchronous data loading without reloading the entire page.
- **Libraries and Frameworks:** Tools like jQuery, React, or Angular provide additional functionalities and ease of use.

**Role in the System:**

- **Form Validation:** Ensures user inputs are valid before submission to the server, reducing server load and improving user experience.
- **Dynamic Content:** Updates parts of the web page dynamically, such as updating package details or user profiles without a full page reload.



- **Interactivity:** Enhances user interaction with elements like dropdowns, modals, and real-time notifications.

## **BACKEND**

### **1. PYTHON**

**Purpose:** Python is a versatile, high-level programming language used for a wide range of applications, including web development.

**Key Features:**

- **Readability:** Python's syntax is clear and easy to read, making it accessible for beginners and efficient for experienced developers.
- **Extensive Libraries:** Python has a rich ecosystem of libraries and frameworks for web development, data analysis, machine learning, etc.
- **Interpreted Language:** Python code is executed line by line, which aids in debugging and testing.

**Role in the System:**

- **Business Logic:** Handles the core functionality of the application, such as user authentication, data processing, and application workflows.
- **Integration:** Python integrates seamlessly with various databases, third-party APIs, and other services.

### **2. FLASK**

**Purpose:** Flask is a micro web framework for Python that is lightweight and easy to use, suitable for building small to medium-sized web applications.

**Key Features:**

- **Routing:** Maps URLs to functions in the Python code, defining how web requests are handled.
- **Templates:** Uses Jinja2 for rendering HTML templates with dynamic content from the server.
- **Session Management:** Handles user sessions, allowing for login and authentication mechanisms.
- **Extensions:** Easily extendable with plugins for features like form handling, database integration, and more.

**Role in the System:**

- **URL Routing:** Defines routes for various functionalities, such as /agencysignup, /customerlogin, and /addpackages.
- **Template Rendering:** Generates HTML pages dynamically with data from the backend.
- **Session Handling:** Manages user sessions to keep track of logged-in users and their activities.
- **Form Handling:** Processes data submitted through forms and interacts with the database.

### 3. MySQL

**Purpose:** MySQL is a relational database management system used to store and retrieve structured data using SQL.

**Key Features:**

- **Relational Database:** Organizes data into tables with rows and columns, supporting relationships between different tables.
- **SQL:** Structured Query Language for querying and managing data.
- **Transactions:** Supports ACID (Atomicity, Consistency, Isolation, Durability) properties for reliable transaction processing.
- **Scalability:** Can handle large datasets and concurrent users efficiently.

**Role in the System:**

- **Data Storage:** Stores data related to agencies, customers, travel packages, bookings, and assignments.
- **Data Retrieval:** Executes queries to retrieve data for display on web pages, such as listing available packages or showing customer details.
- **Data Integrity:** Ensures data consistency and integrity through constraints and transactions.
- **Security:** Manages access control and user permissions to protect sensitive data.

## 2.4 FUNCTIONAL REQUIREMENT

Functional requirements describe the specific behavior or functions of the system. For the System, these include:

### 1. User Authentication

- **Agency Signup/Login:** Agencies must be able to sign up and log in to manage their travel packages.

- **Customer Signup/Login:** Customers must be able to sign up and log in to browse and book travel packages.

## 2. Agency Management

- **Add/Edit/Delete Packages:** Agencies must be able to create, update, and delete travel packages.
- **Assign Guides:** Agencies must be able to assign guides to specific travel packages.
- **View Packages:** Agencies must be able to view a list of their travel packages.
- **View Assignments:** Agencies must be able to view guide assignments for their packages.

## 3. Customer Management

- **Browse Packages:** Customers must be able to browse available travel packages.
- **Book Packages:** Customers must be able to book travel packages.
- **View Booked Packages:** Customers must be able to view their booked packages.
- **Cancel Bookings:** Customers must be able to cancel their bookings.

## 4. Data Management

- **Database Operations:** The system must support CRUD (Create, Read, Update, Delete) operations for all data entities, including agencies, customers, packages, and assignments.
- **Search and Filter:** The system must provide search and filtering capabilities for packages and other entities.

## 5. User Interaction

- **Forms Handling:** The system must handle various forms for data input (e.g., registration forms, booking forms).
- **Responsive UI:** The system must provide a responsive user interface that works on various devices and screen sizes.

## 2.5 NON-FUNCTIONAL REQUIREMENT

Non-functional requirements describe the quality attributes, performance criteria, and constraints of the system. For the System, these include:

### 1. Performance

- **Response Time:** The system should have a fast response time, with page load times under 2 seconds under normal conditions.
- **Scalability:** The system should be able to handle an increasing number of users and transactions without performance degradation.

## 2. Usability

- **User-Friendly Interface:** The system should have an intuitive and easy-to-use interface for both agencies and customers.
- **Accessibility:** The system should be accessible to users with disabilities, adhering to accessibility standards (e.g., WCAG).

## 3. Security

- **Data Protection:** The system must protect user data through encryption and secure data storage.
- **Authentication and Authorization:** The system must enforce proper authentication and authorization mechanisms to prevent unauthorized access.
- **Input Validation:** The system must validate all user inputs to prevent security vulnerabilities like SQL injection and cross-site scripting (XSS).

## 4. Reliability

- **Availability:** The system should have high availability, with minimal downtime.
- **Backup and Recovery:** The system should have mechanisms for data backup and recovery to prevent data loss.

## 5. Maintainability

- **Code Quality:** The system should be developed with clean, modular, and well-documented code to facilitate maintenance and future enhancements.
- **Error Handling:** The system should provide meaningful error messages and handle errors gracefully.

## 6. Compatibility

- **Cross-Browser Compatibility:** The system should work across major web browsers (e.g., Chrome, Firefox, Safari, Edge).
- **Cross-Device Compatibility:** The system should be compatible with various devices, including desktops, tablets, and smartphones.

## 7. Portability

- **Deployment:** The system should be easy to deploy on different environments, such as local servers, cloud services, and web hosting platforms.

# CHAPTER 3

# SYSTEM DESIGN

### 3.1 ER DIAGRAM

Travel Management System has two views, one as customer and another as agency. Customer can view packages inserted by the agencies. Customer can also opt packages from the view packages option. The Agency offers Packages which can be viewed by the customers. Agency can add packages, delete packages, update packages. Agency also assigns a guide after the customer selects a package.

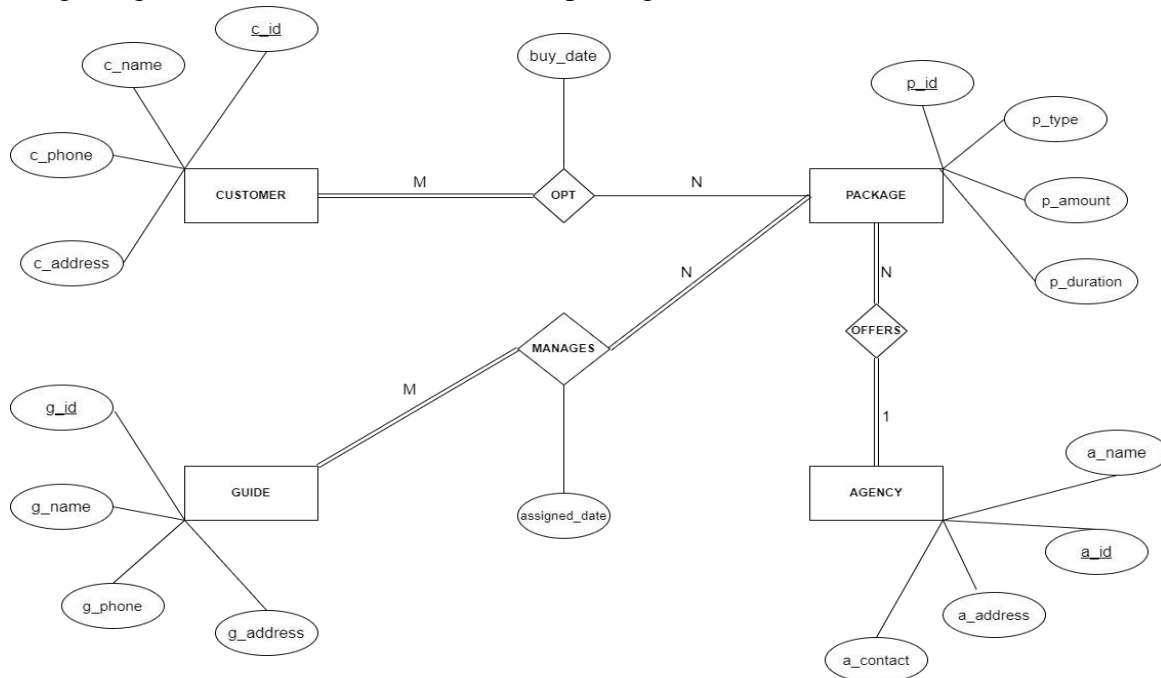


Figure 3.1: ER Diagram for Automated Timetable Generation System

The Agencies Have to insert tour packages, which will be viewed by the customers. The customers then can opt any of the package. After opting the package, the Agency will receive the customers information. When customer opts a package, Agency will then Assign a Guide for the package.

### 3.2 MAPPING FROM ER DIAGRAM TO SCHEMA DIAGRAM

#### STEP-1: Mapping of Regular Entities

Regular entities in an ER diagram possess a primary key attribute and can exist independently, typically depicted with a single rectangle. They have attributes that uniquely identify each instance of the entity.

## Package

<u>p_id</u>	p_type	p_amount	p_duration	<u>a_id</u>
-------------	--------	----------	------------	-------------

## Agency

<u>a_id</u>	a_name	a_address	a_contact
-------------	--------	-----------	-----------

## Guide

<u>g_id</u>	g_name	g_address	g_phone
-------------	--------	-----------	---------

## Customer

<u>c_id</u>	c_name	c_phone	c_address
-------------	--------	---------	-----------

## Opt

<u>p_id</u>	<u>c_id</u>	buy_date
-------------	-------------	----------

## Assign

<u>p_id</u>	<u>g_id</u>	assigned_date
-------------	-------------	---------------

Figure 3.2: Mapping of Regular Entities

**STEP-2: Mapping of Weak Entities**

Weak entities in an ER diagram lack a primary key and rely on a related identifying entity for their existence, typically depicted with a double rectangle. They often have a partial key and a total participation constraint with the identifying entity. The above ER diagram does not contain any weak entities

**STEP-3: Mapping of Binary 1:1 Relationships**

In an ER diagram, a 1:1 relationship signifies a one-to-one association between two entities, where each instance of one entity corresponds to exactly one instance of the other entity. It is depicted with a straight line connecting the two entities, indicating mutual uniqueness and dependency. The above ER diagram has no binary 1:1 relationships.

### STEP-4: Mapping of Binary 1:N Relationships

A binary 1:N (one-to-many) relationship in an ER diagram denotes that each instance of one entity can be associated with multiple instances of another entity, while each instance of the latter entity is related to only one instance of the former. The relationship between Agency and Package is a binary 1:N Relationship.

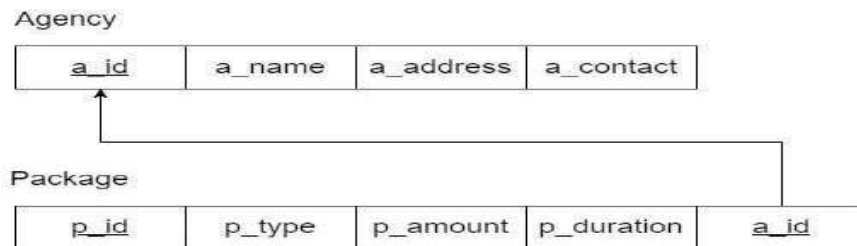


Figure 3.3: Mapping of Binary 1:N Relationships

### STEP-5: Mapping of Binary M:N Relationships

A binary M:N (many-to-many) relationship in an ER diagram signifies that each instance of one entity can be associated with multiple instances of another entity, and vice versa. It is depicted by a diamond shape connecting the entities, representing the junction table necessary to resolve the M:N relationship into two 1:N relationships. The availability of a Teacher at the TimeSlot is a binary M:N relationship.

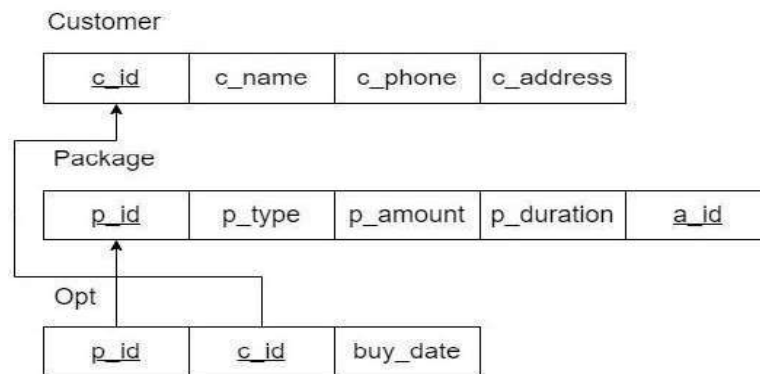


Figure 3.4: Mapping of Binary M:N Relationships

### STEP-6: Mapping of Multivalued attributes

Multivalued attributes in an ER diagram represent attributes that can hold multiple values for a single entity instance. They are depicted with a double ellipse and typically require a separate entity to resolve them into separate instances. The above ER diagram has no multivalued attributes



## STEP-7: Mapping of N-ary relationships

N-ary relationships in an ER diagram denote associations between multiple entities, where "n" represents the number of entities involved. They illustrate complex relationships that involve more than two entities and are depicted using diamonds with connecting lines to each participating entity. The teaching relation between the Teacher, Subject and Class is a ternary relationship.

### 3.3 NORMALIZATION

#### PACKAGE

Pid → {Ptype, Pamount, Pduration}

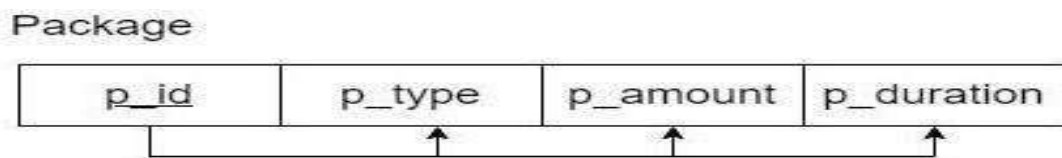


Figure 3.5 Package normalization

**1NF:** The Functional Dependency is in 1NF since all the attributes are atomic. **2NF:** The Functional Dependency is in 2NF since all the attributes solely depend on the whole primary key, eliminating any partial dependencies.

**3NF:** The Functional Dependency is in 3NF since there does not exist any transitive dependencies.

#### AGENCY

Aid → {Aname, Address, Acontact}

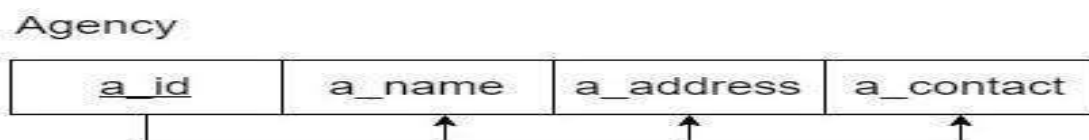


Figure 3.6 Agency normalization

**1NF:** The Functional Dependency is in 1NF since all the attributes are atomic. **2NF:** The Functional Dependency is in 2NF since all the attributes solely depend on the whole primary key, eliminating any partial dependencies.

**3NF:** The Functional Dependency is in 3NF since there does not exist any transitive Dependencies

**GUIDE**

Gid → {gname, gaddress, gphone}

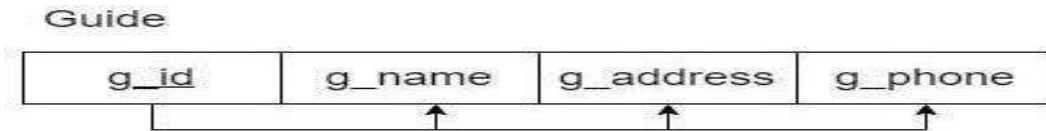


Figure 3.7 Guide normalization

**1NF:** The Functional Dependency is in 1NF since all the attributes are atomic.

**2NF:** The Functional Dependency is in 2NF since all the attributes solely depend on the whole primary key, eliminating any partial dependencies.

**3NF:** The Functional Dependency is in 3NF since there does not exist any transitive dependencies.

**CUSTOMER**

cid → {cname, cphone, caddress}

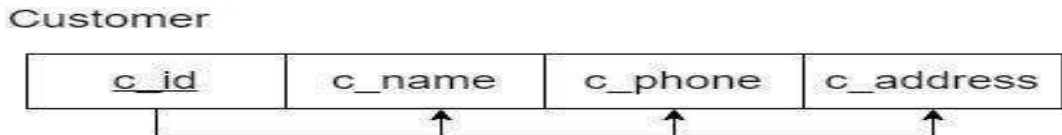


Figure 3.8 Customer normalization

**1NF:** The Functional Dependency is in 1NF since all the attributes are atomic.

**2NF:** The Functional Dependency is in 2NF since all the attributes solely depend on the whole primary key, eliminating any partial dependencies.

**3NF:** The Functional Dependency is in 3NF since there does not exist any transitive dependencies.

**OPT**

pid,cid → {buydate}

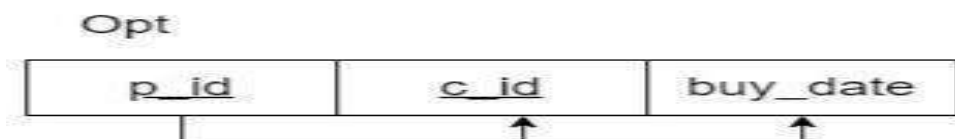


Figure 3.9 Opt normalization

**1NF:** The Functional Dependency is in 1NF since all the attributes are atomic. **2NF:** The Functional Dependency is in 2NF since all the attributes solely depend on the whole primary key, eliminating any partial dependencies.

**3NF:** The Functional Dependency is in 3NF since there does not exist any transitive dependencies.

## ASSIGN

$pid, cid \rightarrow \{assigneddate\}$

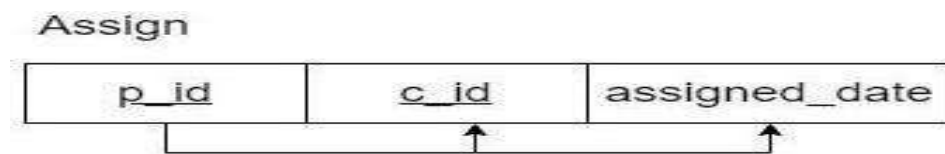


Figure 3.10 Assign normalization

**1NF:** The Functional Dependency is in 1NF since all the attributes are atomic. **2NF:** The Functional Dependency is in 2NF since all the attributes solely depend on the whole primary key, eliminating any partial dependencies.

**3NF:** The Functional Dependency is in 3NF since there does not exist any transitive dependencies.

## 3.4 ASSUMPTIONS

- One Customer can Opt N packages
- One package can be opted by N customers
- One Guide can manage N Packages
- One Package can be managed by N guides
- One Agency can Offer N packages

## 3.5 SCHEMA DIAGRAM

A schema diagram is a visual depiction of a database's structure, including entities, properties, relationships, and constraints. Unlike data flow diagrams (DFDs), which depict the flow of data across a system, schema diagrams provide a static snapshot of the database architecture. They serve as database development blueprints, visualising the components and linkages of the database schema to facilitate stakeholder communication. While DFDs focus on data flow, schema diagrams provide a complete overview of the database structure, allowing for a better understanding of its design and functionality.

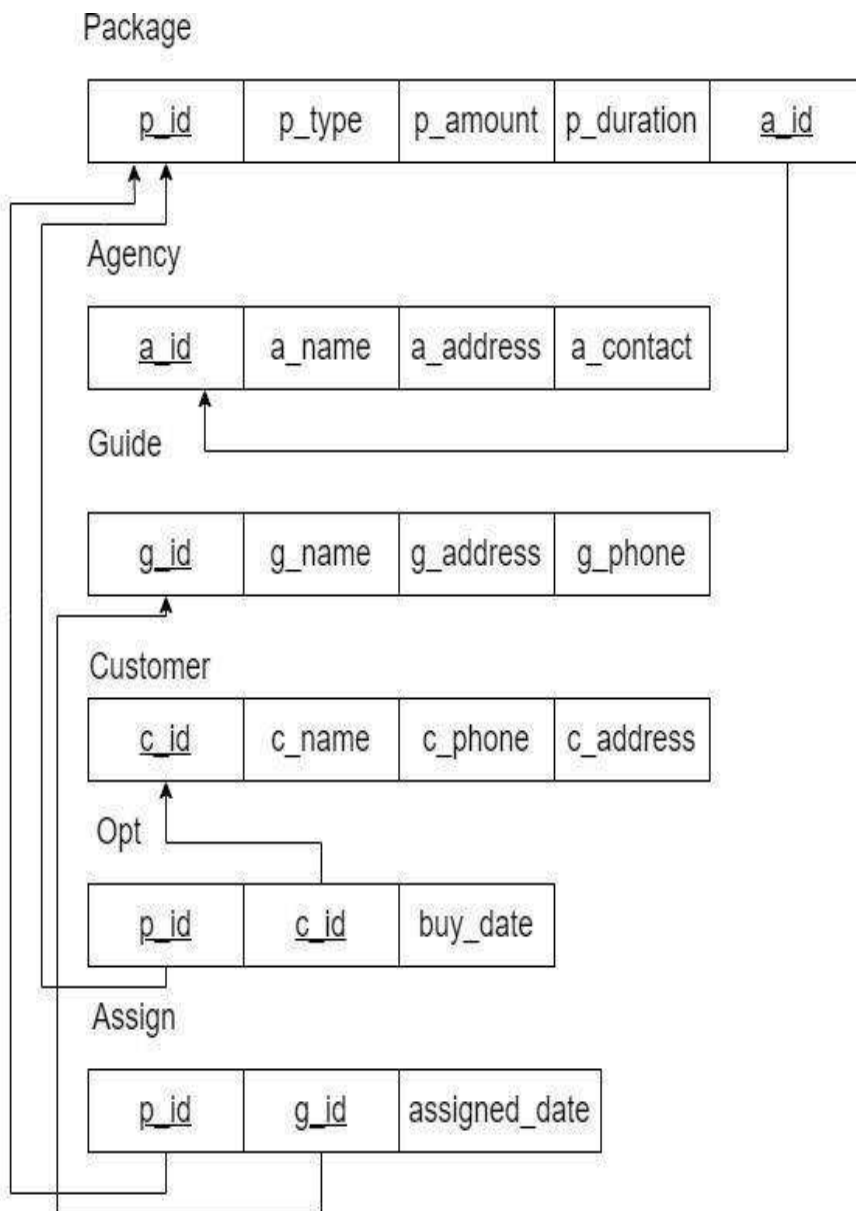


Figure 3.11: Schema Diagram for Travel Management System

### 3.6 INPUT DESIGN OF THE SYSTEM

Input design is a critical part of system design that focuses on how data is entered into the system. Effective input design improves user interaction and data accuracy. For the Travel Agency Management System, various input forms are designed to collect information from users, including agencies and customers. Below is an outline of the key input forms and their design considerations.

## 1. AGENCY SIGNUP FORM

**Purpose:** To allow travel agencies to register on the platform.

**Fields:**

- **Agency ID (a\_id):** A unique identifier for the agency.
  - ❖ **Input Type:** Text
  - ❖ **Validation:** Required, alphanumeric, unique
- **Agency Name (a\_name):** The name of the travel agency.
  - ❖ **Input Type:** Text
  - ❖ **Validation:** Required, alphabetic
- **Agency Address (a\_address):** The address of the travel agency.
  - ❖ **Input Type:** Text
  - ❖ **Validation:** Required
- **Agency Contact (a\_contact):** Contact number of the travel agency.
  - ❖ **Input Type:** Text
  - ❖ **Validation:** Required, numeric, valid phone number format

**Design Considerations:**

- ❖ **Clear Labels:** Ensure each field is clearly labeled.
- ❖ **Placeholder Text:** Provide examples in the placeholder text to guide the user.
- ❖ **Validation Messages:** Display error messages for invalid inputs.
- ❖ **Submit Button:** Clearly labeled "Sign Up".

**Example HTML:**

```
<form method="POST" action="/agencysignup">
  <label for="a_id">Agency ID:</label>
  <input type="text" id="a_id" name="a_id" required>
  <label for="a_name">Agency Name:</label>
  <input type="text" id="a_name" name="a_name" required>
  <label for="a_address">Agency Address:</label>
  <input type="text" id="a_address" name="a_address" required>
  <label for="a_contact">Agency Contact:</label>
  <input type="text" id="a_contact" name="a_contact" required>
  <button type="submit">Sign Up</button>
</form>
```

## 2. CUSTOMER SIGNUP FORM

**Purpose:** To allow customers to register on the platform.

**Fields:**

- **Customer ID (c\_id):** A unique identifier for the customer.

- ❖ **Input Type:** Text
- ❖ **Validation:** Required, alphanumeric, unique
- **Customer Name (c\_name):** The name of the customer.
  - ❖ **Input Type:** Text
  - ❖ **Validation:** Required, alphabetic
- **Customer Address (c\_address):** The address of the customer.
  - ❖ **Input Type:** Text
  - ❖ **Validation:** Required
- **Customer Phone (c\_phone):** Contact number of the customer.
  - ❖ **Input Type:** Text
  - ❖ **Validation:** Required, numeric, valid phone number format

#### Design Considerations:

- **Clear Labels:** Ensure each field is clearly labeled.
- **Placeholder Text:** Provide examples in the placeholder text to guide the user.
- **Validation Messages:** Display error messages for invalid inputs.
- **Submit Button:** Clearly labeled "Sign Up".

#### Example HTML:

```
<form method="POST" action="/customersignup">
  <label for="c_id">Customer ID:</label>
  <input type="text" id="c_id" name="c_id" required>
  <label for="c_name">Customer Name:</label>
  <input type="text" id="c_name" name="c_name" required>
  <label for="c_address">Customer Address:</label>
  <input type="text" id="c_address" name="c_address" required>
  <label for="c_phone">Customer Phone:</label>
  <input type="text" id="c_phone" name="c_phone" required>
  <button type="submit">Sign Up</button>
</form>
```

### 3. AGENCY LOGIN FORM

**Purpose:** To allow travel agencies to log in to the platform.

#### Fields:

- **Agency ID (a\_id):** The unique identifier for the agency.
  - ❖ **Input Type:** Text
  - ❖ **Validation:** Required, alphanumeric
- **Agency Name (a\_name):** The name of the travel agency.
  - ❖ **Input Type:** Text
  - ❖ **Validation:** Required, alphabetic

**Design Considerations:**

- **Clear Labels:** Ensure each field is clearly labeled.
- **Placeholder Text:** Provide examples in the placeholder text to guide the user.
- **Validation Messages:** Display error messages for invalid inputs.
- **Submit Button:** Clearly labeled "Log In".

**Example HTML:**

## 4. CUSTOMER LOGIN FORM

**Purpose:** To allow customers to log in to the platform.

**Fields:**

- **Customer ID (c\_id):** The unique identifier for the customer.
  - ❖ **Input Type:** Text
  - ❖ **Validation:** Required, alphanumeric
- **Customer Name (c\_name):** The name of the customer.
  - ❖ **Input Type:** Text
  - ❖ **Validation:** Required, alphabetic

**Design Considerations:**

- **Clear Labels:** Ensure each field is clearly labeled.
- **Placeholder Text:** Provide examples in the placeholder text to guide the user.
- **Validation Messages:** Display error messages for invalid inputs.
- **Submit Button:** Clearly labeled "Log In".

**Example HTML:**

```
<form method="POST" action="/customerlogin">
  <label for="c_id">Customer ID:</label>
  <input type="text" id="c_id" name="c_id" required>
  <label for="c_name">Customer Name:</label>
  <input type="text" id="c_name" name="c_name" required>
  <button type="submit">Log In</button>
</form>
```

## 5. ADD TRAVEL PACKAGE FORM

**Purpose:** To allow agencies to add new travel packages.

**Fields:**

- **Package ID (p\_id):** A unique identifier for the package.
  - ❖ **Input Type:** Text
  - ❖ **Validation:** Required, alphanumeric, unique
- **Package Type (p\_type):** The type or name of the package.
  - ❖ **Input Type:** Text
  - ❖ **Validation:** Required, alphabetic
- **Package Amount (p\_amount):** The cost of the package.
  - ❖ **Input Type:** Number
  - ❖ **Validation:** Required, numeric
- **Package Duration (p\_duration):** The duration of the package in days.
  - ❖ **Input Type:** Number
  - ❖ **Validation:** Required, numeric

### Design Considerations:

- **Clear Labels:** Ensure each field is clearly labeled.
- **Placeholder Text:** Provide examples in the placeholder text to guide the user.
- **Validation Messages:** Display error messages for invalid inputs.
- **Submit Button:** Clearly labeled "Add Package".

### Example HTML:

```
<form method="POST" action="/addpackages">
  <label for="p_id">Package ID:</label>
  <input type="text" id="p_id" name="p_id" required>
  <label for="p_type">Package Type:</label>
  <input type="text" id="p_type" name="p_type" required>
  <label for="p_amount">Package Amount:</label>
  <input type="number" id="p_amount" name="p_amount" required>
  <label for="p_duration">Package Duration (days):</label>
  <input type="number" id="p_duration" name="p_duration" required>
  <button type="submit">Add Package</button>
</form>
```

## 6. GENERAL INPUT DESIGN

**1. Consistency:** Use a consistent design for all forms, ensuring that users can easily navigate and use the system.

**2. Feedback:** Provide immediate feedback for user actions, such as form submissions and validation errors.



**3. Accessibility:** Ensure forms are accessible to users with disabilities by following web accessibility guidelines.

**4. Security:** Implement measures to prevent common security issues, such as SQL injection and cross-site scripting (XSS), through proper input validation and sanitization.

**5. Responsiveness:** Design forms to be responsive, ensuring they work well on various screen sizes and devices.

## 3.7 OUTPUT DESIGN

Output design focuses on how the information generated by the system is presented to the users. Effective output design ensures that the system communicates information clearly, concisely, and in a user-friendly manner. Here's a detailed description of the output design for the Travel Agency Management System:

### 1. WEB PAGES AND INTERFACES

#### 1. Home Page

- **Description:** The home page is the landing page of the application. It provides an overview of the services offered by the travel agency and includes navigation links to other sections of the website.
- **Content:** Welcome message, brief introduction, navigation menu, login, and sign-up options.

#### 2. Agency Signup and Login Pages

- **Description:** These pages allow travel agencies to sign up for an account or log in to the system.
- **Content:** Forms for entering agency details (ID, name, address, contact), login credentials (ID and name), and submission buttons.

#### 3. Customer Signup and Login Pages

- **Description:** These pages allow customers to sign up for an account or log in to the system.
- **Content:** Forms for entering customer details (ID, name, address, phone), login credentials (ID and name), and submission buttons.

#### 4. Agency Dashboard

- **Description:** The dashboard for authenticated agencies, displaying an overview of their activities and options to manage packages and guides.
- **Content:** Links to add/view packages, add/view guides, assign guides to packages, and view assignments.

#### 5. Customer Dashboard

- **Description:** The dashboard for authenticated customers, displaying available packages and their opted packages.
- **Content:** List of available packages, option to opt for packages, and view of packages they have opted for.

#### 6. Package Management Pages

- **Description:** Pages for agencies to add new packages and view their existing packages.
- **Content:** Forms for entering package details (ID, type, amount, duration), tables displaying package details, and options to edit or delete packages.

#### 7. Guide Management Pages

- **Description:** Pages for agencies to add new guides and view their existing guides.
- **Content:** Forms for entering guide details (ID, name, address, phone), tables displaying guide details, and options to edit or delete guides.

#### 8. Assignment Pages

- **Description:** Pages for assigning guides to packages and viewing assignments.
- **Content:** Forms for selecting a guide and assigning a package, tables displaying assignment details with guides' names and assignment dates, and options to update or delete assignments.

#### 9. Opted Packages Page

- **Description:** Pages for customers to view the packages they have opted for and delete their opted packages if necessary.
- **Content:** Tables displaying opted package details (ID, type, amount, duration), and options to delete opted packages.

## 2. REPORTS AND DATA DISPLAYS

### 1. Agency Reports

- **Description:** Reports generated for agencies to view their package details, assignments, and overall performance.
- **Content:** Tables and charts summarizing package sales, guide assignments, and performance metrics.

### 2. Customer Reports

- **Description:** Reports generated for customers to view their opted packages and history of interactions.
- **Content:** Tables and charts summarizing opted packages and booking history.

## 3. Notifications and Alerts

### 1. Confirmation Messages

- **Description:** Messages confirming successful operations, such as signing up, logging in, adding packages, and assigning guides.
- **Content:** Pop-up alerts or inline messages indicating success.

### 2. Error Messages

- **Description:** Messages indicating errors or issues with user inputs or operations.
- **Content:** Pop-up alerts or inline messages specifying the error and suggesting corrective actions.

## 3. IMPLEMENTATION

- **HTML/CSS:** Used for structuring and styling the web pages, ensuring that the output is presented in a clear and visually appealing manner.
- **JavaScript:** Enhances the user experience by providing dynamic updates and interactive elements, such as form validation and real-time data display.
- **Flask Templates:** Used for rendering dynamic content on the server side, generating HTML pages based on the data retrieved from the database.
- **Bootstrap:** Ensures that the design is responsive and consistent across different devices and screen sizes.

# CHAPTER 4

# METHODOLOGY

## ITERATIVE METHODOLOGY

In this Model, you can start with some of the software specifications and develop the first version of the software. After the first version if there is a need to change the software, then a new version of the software is created with a new iteration. Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.

The Iterative Model allows the accessing earlier phases, in which the variations made respectively. The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process.

### 4.1 STAGE OF THE ITERATIVE DEVELOPMENT PROCESS

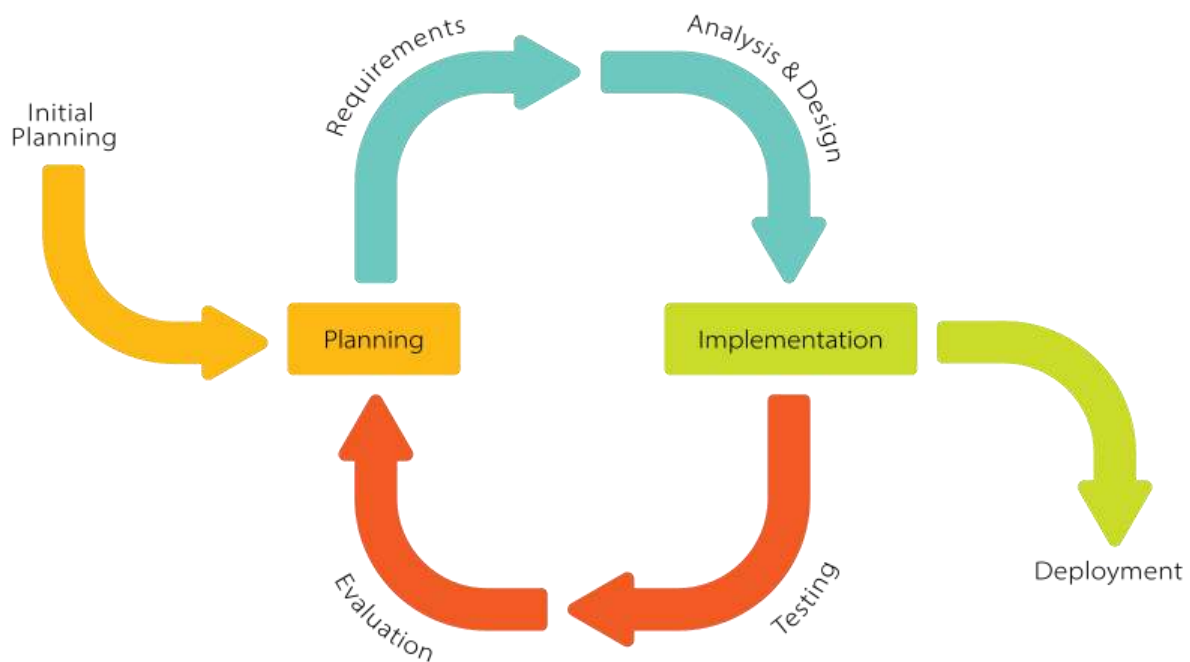


Figure 4.1 Iterative methodology

#### PLANNING:

In the planning phase for your travel agency management system, the focus is on defining clear objectives for each iteration. For example, in the first iteration, you might prioritize implementing basic functionalities like agency and customer management. This includes setting up forms for agency signup and login, customer signup and login, and storing their details securely in the database. Success criteria could be ensuring that agencies and customers can register, log in securely, and their data is correctly stored.

#### DEVELOPMENT:

During the development phase, developers implement the planned features iteratively. For instance, they would create Flask routes and views for agency signup and login, customer signup and login, and integrate these with MySQL database operations. They ensure that user inputs are validated to prevent errors or security vulnerabilities. Collaboration with testers ensures that each feature meets the defined requirements and integrates smoothly with existing functionalities like database operations and session management.

### **TESTING:**

Testing in your system involves various levels to ensure quality and functionality. Unit testing verifies that each function or method works correctly in isolation, such as checking if agency and customer registration functions handle input validation and database interactions properly. Integration testing ensures that components work together seamlessly, like verifying that agency login integrates securely with session management. System testing validates end-to-end scenarios, such as testing the entire signup and login flow for agencies and customers. User acceptance testing involves stakeholders and end-users to confirm that the system meets usability and business requirements.

### **REVIEW:**

In the review phase, stakeholders and end-users provide feedback on the implemented features, such as the ease of use of signup and login processes and any issues encountered during testing. Based on this feedback, adjustments may be made to improve user experience or address any functional gaps. For example, stakeholders might suggest adding additional security measures or enhancing error handling in the login process. Documentation is updated to reflect any changes and decisions made during the review process.

### **DEPLOYMENT:**

Deployment involves releasing the completed iteration of the system to a staging or production environment. Automated deployment scripts ensure consistency and minimize deployment errors. Post-deployment monitoring ensures that the newly released features function as expected and that any issues are promptly addressed. For example, after deploying agency and customer management features, monitoring ensures that registration and login functionalities perform reliably under varying user loads.

### **ITERATIVE REFINEMENT:**

Throughout the iterative refinement phase, the system evolves based on feedback and lessons learned from each iteration. Technical debt, such as optimizing database queries for faster performance or enhancing user interface elements for better usability, is addressed. Each iteration builds upon the previous one, adding new features like package management, guide assignment, and customer booking functionalities based on stakeholder priorities and user feedback. Continuous integration ensures that new features integrate smoothly with existing functionalities, maintaining a cohesive and reliable travel agency management system.

## 4.2 ADVANTAGES OF ITERATIVE DEVELOPMENT:

### 1. FLEXIBILITY AND ADAPTABILITY:

- **Advantage:** Iterative development allows for flexibility in accommodating changes and evolving requirements. Each iteration focuses on delivering a specific set of features, enabling adjustments based on feedback and new insights without disrupting the entire project timeline.
- **Benefit for Travel Agency System:** In a dynamic industry like travel, where customer preferences and market trends can change rapidly, flexibility is crucial. Iterative development allows you to respond quickly to changes, integrate new features, and adapt to emerging business needs or regulatory requirements.

### 2. EARLY AND CONTINUOUS DELIVERY OF VALUE:

- **Advantage:** Iterative development emphasizes delivering functional increments of the system early and continuously. Stakeholders receive tangible results sooner, providing opportunities for early feedback and validation.
- **Benefit for Travel Agency System:** You can prioritize core functionalities such as agency and customer management, package handling, and booking features in initial iterations. Early delivery of these features allows stakeholders to start using and benefiting from the system sooner, enhancing customer satisfaction and operational efficiency.

### 3. RISK REDUCTION THROUGH ITERATIVE TESTING:

- **Advantage:** Iterative development includes frequent testing throughout each iteration. This approach helps identify and address issues early in the development cycle, reducing overall project risks.
- **Benefit for Travel Agency System:** Testing at various stages—unit, integration, system, and user acceptance—ensures that functionalities like agency and customer registration, package management, and booking processes are robust and reliable. Early detection and resolution of issues enhance system quality and reliability.

### 4. ENHANCED STAKEHOLDER COLLABORATION:

- **Advantage:** Iterative development fosters continuous collaboration and communication with stakeholders. Regular feedback loops ensure alignment between development progress and stakeholder expectations.
- **Benefit for Travel Agency System:** Stakeholders, including travel agents, customers, and management, can provide feedback on features like user interfaces, booking workflows, and reporting functionalities. This collaboration ensures that the system meets user needs effectively and aligns with business goals.

### 5. CONTINUOUS IMPROVEMENT AND ADAPTATION:

- **Advantage:** The iterative refinement phase allows for continuous improvement of the system based on feedback, lessons learned, and changing requirements. This iterative cycle supports ongoing innovation and adaptation.
- **Benefit for Travel Agency System:** As the travel industry evolves, your system can evolve with it. Iterative development enables you to add new features, enhance existing functionalities, and incorporate industry best practices to stay competitive and meet evolving customer expectations.

### 4.3 DISADVANTAGES OF ITERATIVE DEVELOPMENT:

#### 1. Increased Complexity and Management Overhead:

- **Disadvantage:** Managing multiple iterations simultaneously can increase project complexity and require effective coordination among development teams, testers, and stakeholders.
- **Challenge for Travel Agency System:** Coordinating development efforts for features like package management, guide assignments, and booking processes across iterations requires robust project management and communication strategies to maintain clarity and alignment.

#### 2. Potential for Scope Creep:

- **Disadvantage:** Iterative development may lead to scope creep if new features or requirements are continuously added without proper prioritization or evaluation of their impact on project timelines and resources.
- **Challenge for Travel Agency System:** In a fast-paced industry like travel, there may be a temptation to incorporate all customer requests or market trends immediately. Effective prioritization and stakeholder alignment are essential to manage scope and maintain project focus.

#### 3. Dependency on Continuous Feedback:

- **Disadvantage:** Iterative development relies heavily on continuous feedback from stakeholders and end-users. Delays in feedback or unclear requirements can affect the pace of development and lead to inefficiencies.
- **Challenge for Travel Agency System:** Ensuring timely and constructive feedback from travel agents, customers, and management is crucial for iterative refinement. Clear communication channels and proactive engagement are necessary to mitigate potential delays or misunderstandings.

#### 4. Potential for Integration Challenges:

- **Disadvantage:** Integrating new iterations with existing system components and third-party services can pose technical challenges, especially if there are dependencies or compatibility issues.
- **Challenge for Travel Agency System:** Integrating functionalities such as payment gateways, CRM systems, or external APIs for flight and accommodation bookings requires meticulous planning and testing to ensure seamless operation and data integrity.

#### 5. Resource and Budget Management:

- **Disadvantage:** Iterative development may require ongoing resource allocation and budget management to sustain continuous development cycles, testing efforts, and stakeholder engagement.
- **Challenge for Travel Agency System:** Ensuring adequate resources—such as skilled developers, testing infrastructure, and stakeholder time commitment—is essential for maintaining momentum and achieving iterative development goals within budgetary constraints.



# CHAPTER 5

# TESTING

## **5.1 TESTING**

Testing is a crucial phase in the software development lifecycle aimed at ensuring the quality, functionality, and reliability of a system. It involves systematically evaluating the software to identify any defects, errors, or missing requirements. By rigorously checking each component, as well as the system as a whole, testing helps verify that the software performs as expected under various conditions. This process not only enhances the software's performance and security but also ensures it meets user needs and business goals, ultimately leading to a more robust and user-friendly product.

## **5.2 TYPES OF TESTING**

### **1. UNIT TESTING**

Unit testing is a software testing technique where individual units or components of a software are tested in isolation. The primary goal of unit testing is to validate that each unit of the software performs as expected. A "unit" refers to the smallest testable part of an application, such as a function, method, or object. Unit tests are typically automated and written by developers during the development phase to ensure that each unit functions correctly. This form of testing helps in identifying and fixing bugs early in the development process, promoting code quality and reliability, and facilitating easier maintenance and refactoring of code.

### **2. INTEGRATION TESTING**

Integration testing is a software testing phase where individual units or components are combined and tested as a group to identify issues in their interactions. The main goal is to ensure that these integrated parts work together seamlessly and that data flows correctly between them. This phase can be approached incrementally, where components are integrated and tested one by one, or through a big bang approach, where all components are integrated simultaneously and tested as a whole. By detecting defects in the interactions early, integration testing helps ensure that the system as a whole functions correct

### **3. SYSTEM TESTING**

System testing is a comprehensive phase in the software testing lifecycle where a fully integrated software system is tested to verify that it meets specified requirements. This level of testing evaluates the system's overall functionality, performance, security, and compliance with regulatory standards. By examining the system as a whole, including all integrated components and their interactions, system testing ensures that the software behaves as expected in real-world scenarios and is ready for deployment.

## 4. USER ACCEPTANCE TEST(UAT)

User Acceptance Testing (UAT) is the final phase in the software testing process where the actual users test the system to ensure it meets their requirements and works as expected in real-world scenarios. Conducted in a production-like environment, UAT focuses on validating the end-to-end business flow and ensuring that the system is ready for deployment. The primary objective of UAT is to identify any issues or discrepancies from the user's perspective, ensuring that the software is user-friendly, functional, and fulfills the business needs before it goes live.

### 5.3 TESTING PERFORMED

#### 1. UNIT TESTING

**Objective:** Validate individual components or functions to ensure they work as intended.

**Tests Conducted:**

- **User Input Validation:** Test the validation logic for agency and customer signup forms.
- **Database Operations:** Test CRUD operations (Create, Read, Update, Delete) for agency, customer, and package data.
- **Authentication Logic:** Verify the login functionality for both agencies and customers.

**Test Results:**

- **User Input Validation:**
  - ❖ Test case: Empty fields, invalid email format, and incorrect phone number format.
  - ❖ Result: Passed. All invalid inputs are correctly flagged, and appropriate error messages are displayed.
- **Database Operations:**
  - ❖ Test case: Insert, update, and delete records in the agency table.
  - ❖ Result: Passed. Records are correctly inserted, updated, and deleted without any data integrity issues.
- **Authentication Logic:**
  - ❖ Test case: Valid and invalid login credentials.
  - ❖ Result: Passed. Valid credentials allow access, while invalid credentials show appropriate error messages.

## 2. INTEGRATION TESTING

**Objective:** Ensure that different modules or components work together seamlessly.

**Tests Conducted:**

- **User Registration and Login Flow:** Test the end-to-end flow from registration to login for agencies and customers.
- **Package Management:** Verify the integration of package creation and display functionality.
- **Guide Assignment:** Ensure that guide assignment integrates correctly with package management.

**Test Results:**

- **User Registration and Login Flow:**
  - ❖ Test case: Complete registration and login process for a new agency and customer.
  - ❖ Result: Passed. New users can register and log in successfully, with their data correctly stored and retrieved from the database.
  
- **Package Management:**
  - ❖ Test case: Add a new package and display it in the list of available packages.
  - ❖ Result: Passed. The new package appears correctly in the list after being added.
- **Guide Assignment:**
  - ❖ Test case: Assign a guide to a package and verify the assignment.
  - ❖ Result: Passed. Guides are correctly assigned to packages, and the assignments are accurately reflected in the database.

## 3. SYSTEM TESTING

**Objective:** Validate the entire system's functionality and performance as a whole.

**Tests Conducted:**

- **Booking and Package Opting:** Test the process of customers opting for packages and viewing their opted packages.
- **Data Integrity:** Ensure data consistency across different functionalities and user roles.
- **Security Testing:** Verify that sensitive data is handled securely, and user sessions are managed correctly.

**Test Results:**

- **Booking and Package Opting:**
  - ❖ Test case: Customer opts for a package and views their opted packages.
  - ❖ Result: Passed. The package opting process works smoothly, and customers can view their opted packages without issues.
- **Data Integrity:**
  - ❖ Test case: Cross-check data consistency between agency, customer, package, and assignment tables.
  - ❖ Result: Passed. Data remains consistent and accurate across different functionalities and user interactions.
- **Security Testing:**
  - ❖ Test case: Attempt unauthorized access to different parts of the system.
  - ❖ Result: Passed. Unauthorized access attempts are correctly blocked, and user sessions are managed securely.

**4. USER ACCEPTANCE TESTING (UAT)**

**Objective:** Validate the system's functionality and usability from the end-users' perspective.

**Tests Conducted:**

- **Usability Testing:** Assess the ease of use and user experience of the application interfaces.
- **Functional Validation:** Confirm that the system meets all business requirements and user expectations.
- **Feedback Collection:** Gather feedback from actual users (travel agents, customers) to identify areas for improvement.

**Test Results:**

- **Usability Testing:**
  - ❖ Test case: Users navigate through different sections (signup, login, package viewing, booking) and perform tasks.
  - ❖ Result: Passed. Users find the interface intuitive and easy to navigate. Minor suggestions for UI enhancements are noted.
- **Functional Validation:**
  - ❖ Test case: Users perform key functions such as signing up, logging in, adding packages, assigning guides, and booking packages.
  - ❖ Result: Passed. All functions work as expected, meeting the business requirements.

# CHAPTER 6

# IMPLEMENTATION

## IMPLEMENTATION

The implementation step involves turning design specifications into a working system. The project is being developed repeatedly, with Python as the primary programming language and MySQL as the backend database management system.

The frontend user interface is built in HTML and CSS, with tools like Python Flask and Bootstrap. These libraries enable the building of an intuitive and user-friendly experience. This interface allows agencies to input package details and guide details.

The backend logic is also written in Python, and it handles responsibilities like database connectivity, data validation, and scheduling algorithm execution. MySQL acts as the database management system, keeping data on Packages, Customer details, Agency details, Guide details, and their relationships.

By following best practices in software development and properly using Python and MySQL, the project implementation intends to create a stable, scalable, and user-friendly Travel Management System. This system meets the needs of customers to easily find the perfect tour package.

## 6.1 TABLES USED

### AGENCY

The "agency" table comprises fields for agency identification, name, address, and contact information, with the agency ID serving as the primary key for unique identification.

```
create table agency( a_id
    varchar(20), a_name
    varchar(20),
    a_address varchar(20),
    a_contact int, primary
    key(a_id));
```

Field	Type	Null	Key	Default	Extra
a_id	varchar(20)	NO	PRI	NULL	
a_name	varchar(20)	YES		NULL	
a_address	varchar(20)	YES		NULL	
a_contact	int	YES		NULL	

## CUSTOMER

The "customer" table stores customer data including ID, name, address, and phone number, with the customer ID serving as the primary key for unique identification.

```
create table customer( c_id
    varchar(20),    c_name
    varchar(20),    c_address
    varchar(20), c_phone int,
    primary key(c_id)
);
```

Field	Type	Null	Key	Default	Extra
c_id	varchar(20)	NO	PRI	NULL	
c_name	varchar(20)	YES		NULL	
c_address	varchar(20)	YES		NULL	
c_phone	int	YES		NULL	

## GUIDE

The "guide" table maintains information about guides, including ID, name, address, and phone number, with the guide ID as the primary key for unique identification.

```
create table guide( g_id
    varchar(20),    g_name
    varchar(20),    g_address
    varchar(20), g_phone int,
    primary key(g_id)
);
```

Field	Type	Null	Key	Default	Extra
g_id	varchar(20)	NO	PRI	NULL	
g_name	varchar(20)	YES		NULL	
g_address	varchar(20)	YES		NULL	
g_phone	int	YES		NULL	



## PACKAGE

The "package" table records details about travel packages, such as ID, type, amount, and duration, linked to agencies through a foreign key relationship on agency ID with cascade deletion for referential integrity.

```
create table package( p_id varchar(20), p_type varchar(20), p_amount int,
    p_duration varchar(20), a_id varchar(20), primary key(p_id), foreign
    key(a_id) references agency(a_id) on delete cascade
);
```

Field	Type	Null	Key	Default	Extra
p_id	varchar(20)	NO	PRI	NULL	
p_type	varchar(20)	YES		NULL	
p_amount	int	YES		NULL	
p_duration	varchar(20)	YES		NULL	
a_id	varchar(20)	YES	MUL	NULL	

## OPT

The "opt" table tracks the selection of packages by customers, capturing package and customer IDs along with the purchase date, ensuring data integrity through foreign key constraints with cascade deletion for both package and customer references.

```
create table opt( p_id
    varchar(20), c_id
    varchar(20), buy_date date,
    primary key(p_id,c_id),
    foreign key(p_id)
    references package(p_id)
    on delete cascade, foreign
    key(c_id) references
    customer(c_id) on delete
    cascade
);
```

Field	Type	Null	Key	Default	Extra
p_id	varchar(20)	NO	PRI	NULL	
c_id	varchar(20)	NO	PRI	NULL	
buy_date	date	YES		NULL	

## ASSIGN

The "assign" table manages the assignment of guides to packages, recording package and guide IDs along with assignment dates, maintaining data integrity through foreign key constraints with cascade deletion for both package and guide references.

```
create table assign( p_id varchar(20), g_id varchar(20), assign_date date,
    primary key(p_id,g_id), foreign key(p_id) references package(p_id) on
    delete cascade, foreign key(g_id) references guide(g_id) on delete
    cascade
);
```

Field	Type	Null	Key	Default	Extra
p_id	varchar(20)	NO	PRI	NULL	
g_id	varchar(20)	NO	PRI	NULL	
assign_date	date	YES		NULL	

## 6.2 DATABASE CONNECTION

```
from flask import Flask, render_template, redirect, url_for, request import
mysql.connector from datetime import datetime app = Flask(__name__)
authenticated_agency_id = None authenticated_customer_id=None
```

```
# MySQL configuration db_config
= {
    'host': 'localhost',
    'user': 'root',
    'password': 'donbosco',
    'database': 'traveldb',
    'autocommit': True
}
```

```
def sql_connection():
    conn = mysql.connector.connect(**db_config) c =
    conn.cursor(buffered=True) return conn, c
```

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

## 6.3 INSERT OPERATION

it establishes a database connection and inserts package details obtained from the request form into the package table, associating them with the authenticated agency ID. After successful insertion, it redirects the user to the agency view route; otherwise, it renders a template for adding packages.

```

@app.route('/addpackages', methods=['GET', 'POST'])
def add_package():
    if request.method == 'POST':
        conn, c = sql_connection()
        p_id = request.form['p_id']
        p_type = request.form['p_type']
        p_amount = request.form['p_amount']
        p_duration = request.form['p_duration']
        a_id = authenticated_agency_id
        c.execute("INSERT INTO package (p_id, p_type, p_amount, p_duration, a_id) VALUES (%s, %s, %s, %s, %s)",
                [(p_id, p_type, p_amount, p_duration, a_id)])
        c.close()
        conn.close()
        return redirect(url_for('agency_view'))
    return render_template('/package/addpackages.html')

```

## 6.4 RETRIEVE OPERATION

The below code retrieves all packages from the database using an SQL query and subsequently renders a template `cdisplaypackages.html` to display these packages for customers. This functionality enhances the Travel Management System project by enabling customers to view available travel packages conveniently

```

@app.route('/cdisplaypackage')
def display_cpackage():

    conn, c = sql_connection()
    c.execute("SELECT * FROM package ")
    packages = c.fetchall()
    c.close()
    conn.close()
    return render_template('package/cdisplaypackages.html', packages=packages)

```

## 6.5 UPDATE OPERATION

The below code updates assignment dates for specific package and guide IDs in the database, enhancing the system's efficiency by streamlining assignment modifications.

```

@app.route('/update_assignment/<p_id>/<g_id>', methods=['POST'])
def update_assignment(p_id, g_id):
    if request.method == 'POST':
        new_assign_date = request.form['assign_date']
        conn, c = sql_connection()
        c.execute("UPDATE assign SET assign_date = %s WHERE p_id = %s AND g_id = %s",
                (new_assign_date, p_id, g_id))
        conn.close()
        return redirect(url_for('view_assignments'))

```

## 6.6 DELETE OPERATION

The below code deletes a selected package option linked to the authenticated customer ID. Upon deletion, users are redirected to the opted packages page, ensuring smooth system navigation.

```
@app.route('/delete_opt/<package_id>', methods=['POST'])
def delete_opt(package_id):
    global authenticated_customer_id
    if request.method == 'POST':
        conn, c = sql_connection()
        c.execute("DELETE FROM opt WHERE p_id = %s AND c_id = %s", (package_id, authenticated_customer_id))
        conn.close()
        # Redirect to opted_packages after deletion
        return redirect(url_for('opted_packages'))
    # Redirect to opted_packages if not a POST request
    return redirect(url_for('opted_packages'))
```

# CHAPTER 7

# SNAPSHOTS

## SNAPSHOTS

This section shows snapshots of the system's interface, providing a visual representation of its working. The images provide a glimpse at the system's design, navigation, and important features, highlighting its usability and performance. The images below help readers understand the system's capabilities and potential benefits.

### 7.1 HOME PAGE

The home page serves as the system's centre, providing users with a navigational starting point as well as a summary of key information or features. It provides a userfriendly interface that allows them to quickly access important functionality and manage the system.

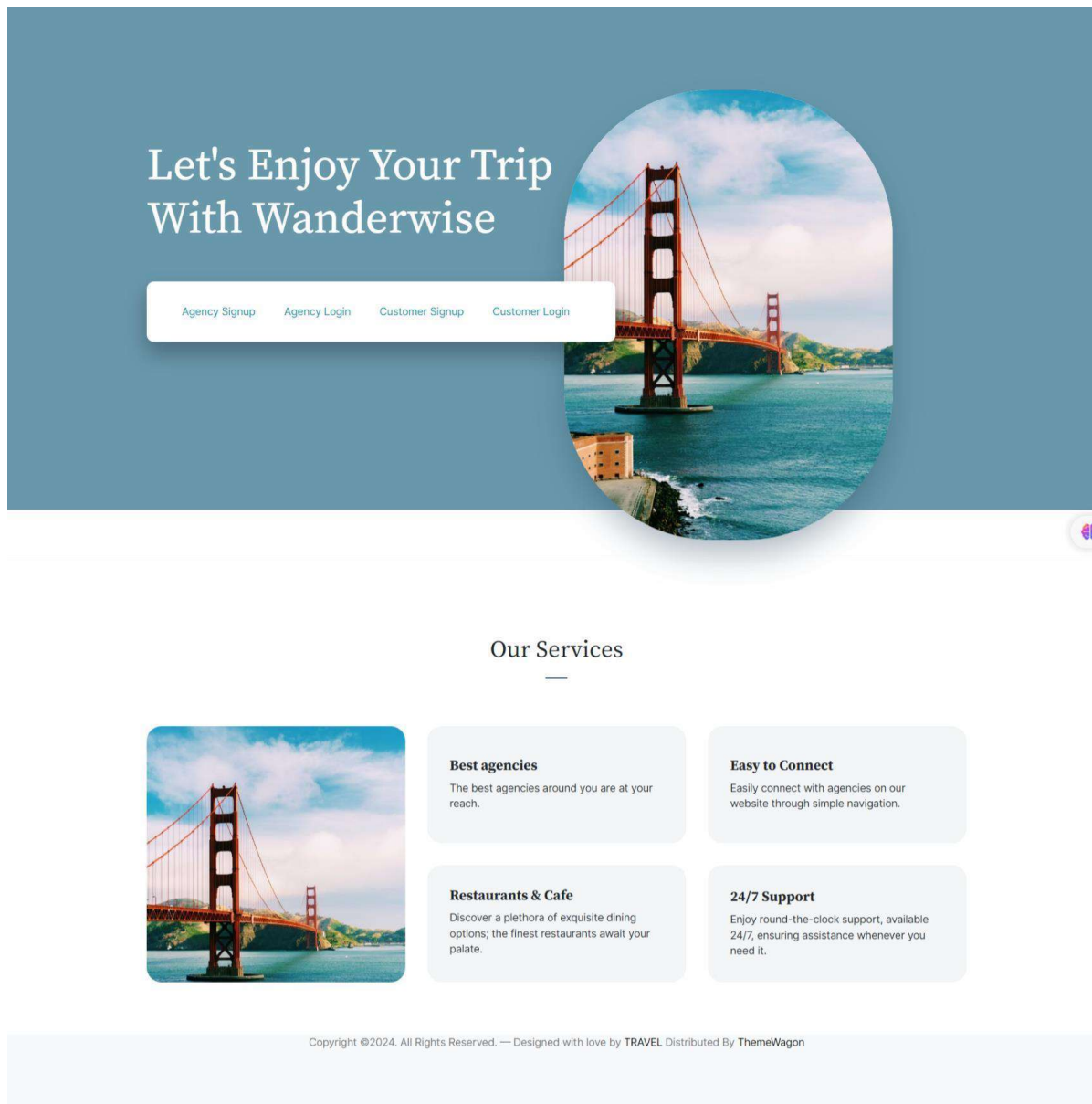
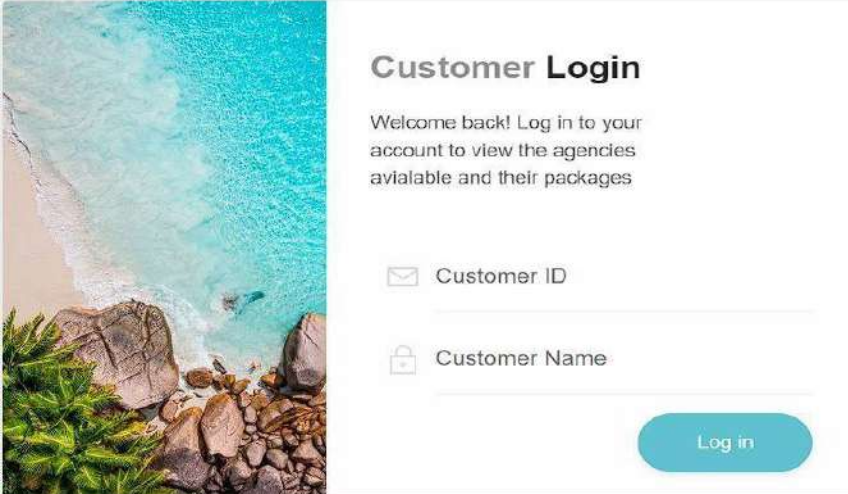


Figure 7.1 Home page

## 7.2 CUSTOMER LOGIN

This page serves as the entry point for customers to access their accounts and view available agencies and packages. Let's delve deeper into the various aspects of this project page:



**Customer Login**

Welcome back! Log in to your account to view the agencies available and their packages

Customer ID

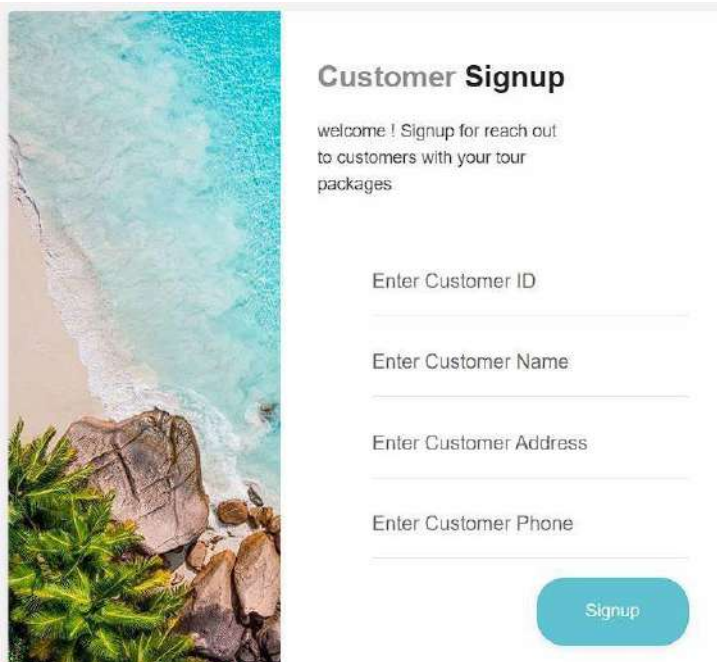
Customer Name

Log in

Figure 7.2 customer login

## 7.3 CUSTOMER SIGNUP PAGE

The signup page for users interested in tour packages. It presents a visually engaging experience with a split-screen layout, allowing users to input their customer details. While the page excels in visual presentation and form functionality, it lacks backend integration for form submission and requires attention to accessibility considerations for a comprehensive signup experience.



**Customer Signup**

welcome ! Signup for reach out to customers with your tour packages

Enter Customer ID

Enter Customer Name

Enter Customer Address

Enter Customer Phone

Signup

Figure 7.3 Customer 7.3



## 7.4 AGENCY LOGIN PAGE

The login page designed for agencies to access and update tour packages. With a visually appealing layout, it offers a split-screen design, featuring an image on one side and the login form on the other. Users are prompted to enter their agency ID and name to access their accounts.

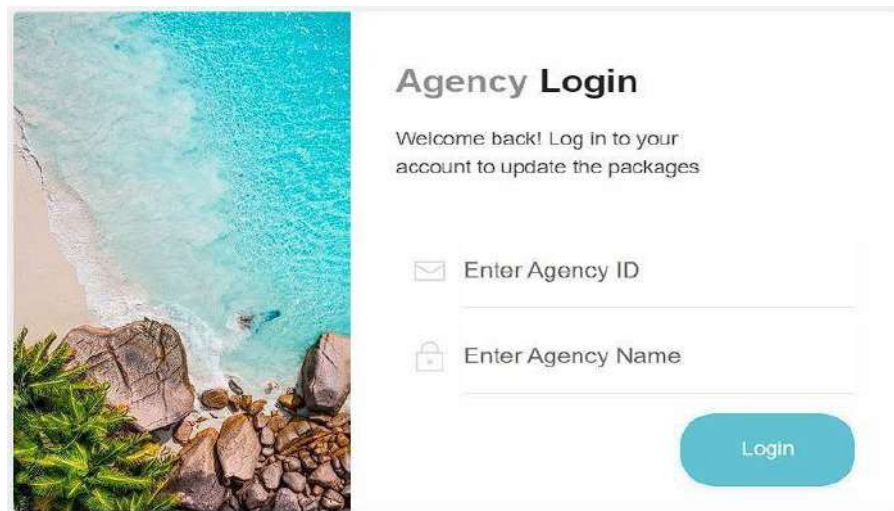


Figure 7.4 Agency Login Page

## 7.5 AGENCY SIGNUP PAGE

This signup form tailored for agencies interested in creating accounts to manage tour packages. With a sleek design, it offers a split-screen layout featuring an image on one side and the signup form on the other. Agencies are prompted to provide essential details to register their accounts.

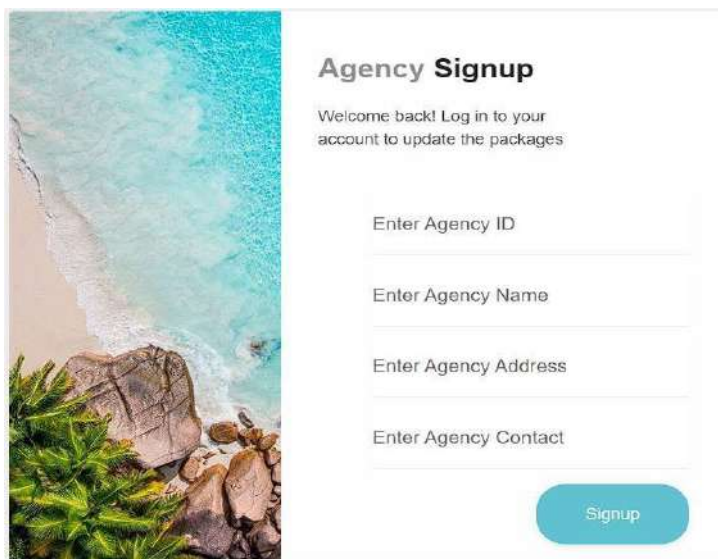


Figure 7.5 Agency Signup page



## 7.6 AGENCY VIEW PAGE

This page is designed as the "Agency View" for WanderWise, providing an interface for travel agencies to manage their offerings. The page includes a navigation bar for easy access to various functionalities such as displaying agencies, adding packages, viewing assigned tasks, and adding guides. The main content area highlights the purpose of the page, allowing agencies to add and manage packages efficiently.



Figure 7.6 Agency View page

## 7.7 CUSTOMER VIEW PAGE

This page is designed as the "Customer View" for WanderWise, providing an interface for customers to explore travel agencies and available packages. The page features a navigation bar that allows users to display agencies, view available packages, see opted packages, and log out. The main content area highlights the purpose of the page, enabling customers to view and select travel packages conveniently.



Figure 7.7 Customer View Page

## 7.8 DISPLAY AGENCY LIST

This "Agency View" page displays a list of travel agencies with a clean, user-friendly design. It features a navigation bar, a table with agency details, and a footer with a copyright notice. The design is responsive, using a professional color scheme and hover effects for a better user experience.

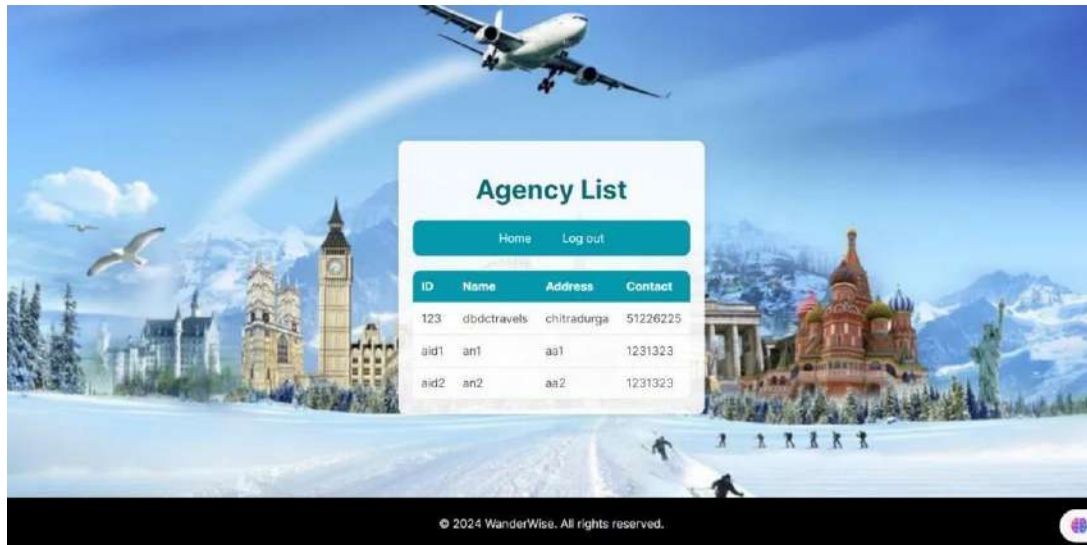


Figure 7.8 Display Agency

## 7.9 DISPLAY PACKAGES LIST

This page, styled with a modern and professional look, showcases various travel packages available for viewing and management. The layout ensures clarity and ease of navigation, with a central focus on presenting package details in a structured table format. Integrated navigation links provide seamless access to other sections, maintaining a cohesive user experience. Designed with a clean aesthetic and responsive layout, it offers a user-friendly interface for efficient package management and assignment tasks.

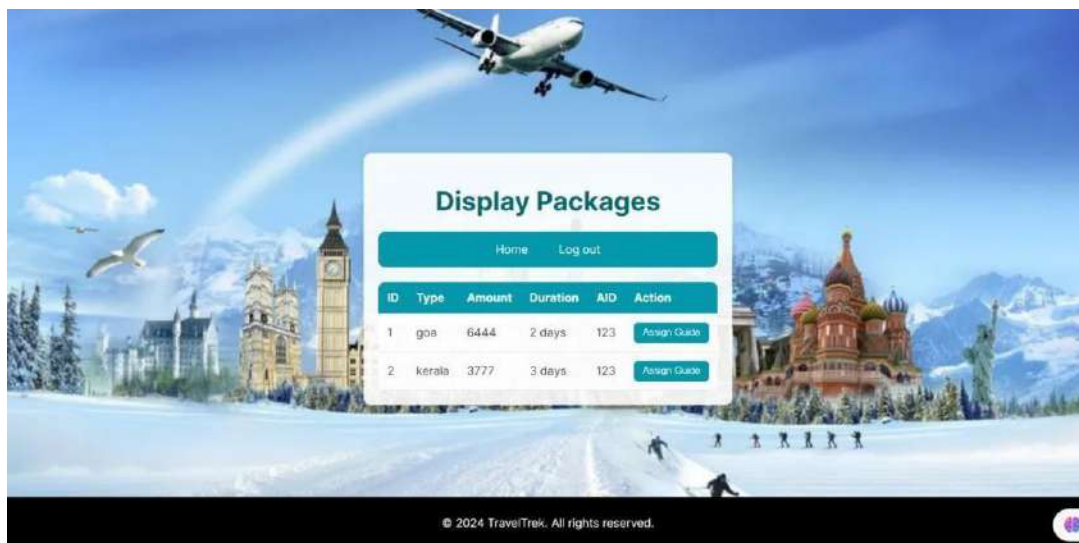


Figure 7.9 Display Packages list

## 7.10 ASSIGNING GUIDE

This page provides a straightforward interface to assign guides to travel packages. Designed with simplicity and clarity in mind, it features a form where administrators can input the guide's ID and the date of assignment. The form's responsive layout ensures ease of use on various devices, maintaining a clean aesthetic against a backdrop of engaging visuals.

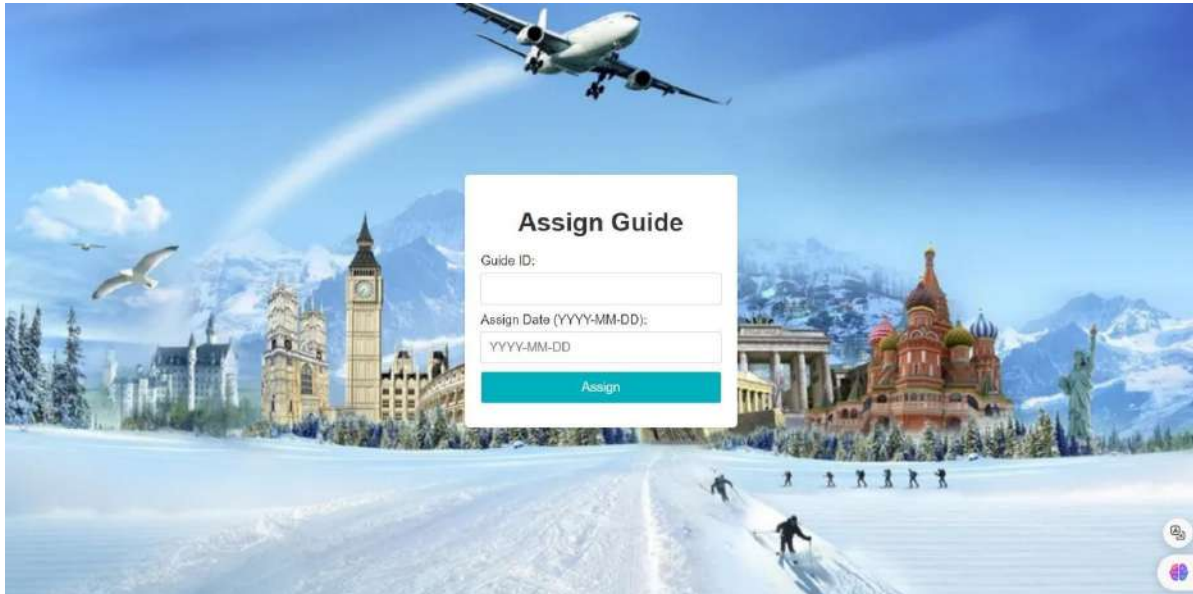


Figure 7.10 Assigning Guide

# CHAPTER 8

# CONCLUSION

## 8.1 CONCLUSION

In conclusion, the development of this web-based travel agency management system using Flask and MySQL marks a significant step towards modernizing travel service management. This project aimed to provide agencies and customers with a seamless platform for managing travel packages, guide assignments, customer registrations, and package opt-ins. Key features such as robust database design, user authentication mechanisms, and intuitive web interfaces were integral to achieving these goals.

Throughout the project, meticulous attention was given to database design using MySQL. The schema was structured to efficiently store and manage crucial data entities such as agencies, customers, packages, guides, assignments, and package opt-ins. Utilizing foreign key constraints ensured data integrity, maintaining relational consistency across tables. This foundational design facilitated streamlined operations and enabled secure data management, crucial for a system handling sensitive customer and business information.

The implementation of Flask as the web framework provided a flexible and scalable architecture for the travel agency management system. Flask routes were meticulously crafted to handle various functionalities, including agency and customer authentication, package management, guide assignments, and opt-in processes. Global variables like `authenticated_agency_id` and `authenticated_customer_id` were employed to manage user sessions effectively, ensuring secure and personalized interactions.

## 8.2 FUTURE WORKS

Looking ahead, several areas for future enhancement and development have been identified to further elevate the travel agency management system. Implementing advanced search and filtering functionalities will empower customers to find travel packages based on specific criteria like destination, price range, or duration, enhancing their overall booking experience.

Real-time updates using WebSocket technology represent another pivotal area for improvement. Enabling real-time notifications for agencies and customers regarding package availability, assignment updates, or booking confirmations can significantly enhance communication and user engagement. Furthermore, integrating secure payment gateways to facilitate online transactions for package opt-ins will streamline the booking process, offering customers a seamless and efficient booking experience directly from the platform.

Analytics and reporting capabilities are also essential for future iterations of the system. By implementing robust analytics tools, agencies can gain valuable insights into package popularity, customer preferences, revenue trends, and operational efficiencies. These insights will empower agencies to make data-driven decisions, optimize service offerings, and tailor marketing strategies to meet customer demands effectively.

## BIBLIOGRAPHY

### BOOKS

1. **Efficient MySQL Performance: Best Practices and Techniques**  
By Daniel Nichter
2. **Think python: how to think like a computer scientist**  
by Allen B. Downey

### WEB SITE

1. <https://app.diagrams.net/>
2. <https://www.geeksforgeeks.org/python-programming-language-tutorial/>
3. [https://www.w3schools.com/mysql/mysql\\_intro.asp](https://www.w3schools.com/mysql/mysql_intro.asp)
4. [https://www.youtube.com/watch?v=Z1RJmh\\_OqeA&t=41s&pp=ygUMZmxhc2sgcHI0aG9u](https://www.youtube.com/watch?v=Z1RJmh_OqeA&t=41s&pp=ygUMZmxhc2sgcHI0aG9u)
5. <https://www.youtube.com/watch?v=6M3LzGmIAso&pp=ygUMZmxhc2sgcHI0aG9u>
6. <https://chatgpt.com/>