**Birla Institute of Technology and Science Pilani, Dubai**

**Campus**

**Dubai International Academic City**

**Title: FlyHippo – Travel and Tourism Management**

****

**Semester 1 2025-26**

**OBJECT-ORIENTED PROGRAMMING ASSIGNMENT REPORT SUBMISSION**

**CS F213**

**Submission Date - 30-11-2025**

**Submitted by:**

**Vansh Sethi - 2024A7PS0191U**

**Atharva Karanjkar - 2024A7PS0230U**

**Abstract-**  
The Travel Management System is a desktop-based Java application designed to simplify the process of booking tour packages, hotels, and managing customer travel information. The system solves the problem of manual and unorganized travel planning by providing a digital platform where users can sign up, log in, view destinations, book customized packages, check hotel details, and store personal travel records securely. It integrates a graphical user interface using Java Swing and connects to a MySQL database for persistent data storage.

Additional features include automated email notifications, QR-based payment windows, and comprehensive customer management modules. After the payment is done, a email is auto sent to the customer having the trip details and cost breakdown. Overall, the project enhances efficiency, reduces human error, and provides a user-friendly solution for small travel agencies or individual users.

**Introduction-**

**Background and Motivation of the Project**

Travel planning traditionally involves multiple manual steps such as searching destinations, comparing hotel prices, selecting packages, and managing customer details. These tasks are often time-consuming, repetitive, and prone to errors when done manually. With the increasing need for fast and reliable travel services, there is a strong demand for computerized systems that can automate these operations. The motivation behind developing the **Travel Management System** is to provide a centralized platform where users can easily browse destinations, book hotels or tour packages, and store personal travel information in an organized manner. This system also benefits travel agencies by reducing manual workload and improving accuracy.

**Objectives and Scope**

The primary objective of this project is to develop a user-friendly Java application capable of handling common travel-related tasks efficiently. The system allows users to register, log in, view destinations, book hotels and packages, update personal details, and make payments through an integrated QR interface and receive a email after booking is confirmed.

The scope includes GUI design using Java Swing, secure data operations via MySQL, email-based notifications, and proper management of all customer and booking data. The system is intended for small to medium-scale travel agencies and individual users who require a simple yet effective travel planning tool.

**System Analysis and Design-**

1. **Requirement Analysis**

**Functional Requirements**

The Travel Management System provides the following core functionalities:

1. User Authentication
   1. User signup with basic details
   2. Secure login using username and password
   3. Forgot Password feature with email verification
2. Customer Management
   1. Add new customer details
   2. Update existing customer information
   3. View customer profiles
   4. Delete customer records
3. Travel Services
   1. View available destinations
   2. Book tour packages
   3. Book hotels with pricing details
   4. View previously booked packages or hotels
4. Payment Processing
   1. QR code–based payment window
   2. Payment confirmation interface
5. System Notifications
   1. Automatic email confirmations using JavaMail API
6. Database Operations
   1. Store and retrieve all information from MySQL
   2. Maintain customer, package, and hotel records

**Non-Functional Requirements**

1. **Usability**  
   The application’s GUI uses Java Swing components designed to be intuitive and easy to navigate for any user.
2. **Performance**  
   The system retrieves and processes records efficiently using well-structured SQL queries in the backend.
3. **Scalability**  
   More destinations, packages, or services can be added without modifying the core structure.
4. **Security**  
   Passwords are validated securely. Database connections are handled using prepared statements to prevent misuse.
5. **Reliability**  
   Ensures consistent operation with proper exception handling across all modules.
6. **Use Case Diagram / Flowchart**

*(Since diagrams cannot be drawn here directly, I will describe them clearly so you can draw them in your report.)*

**Use Case Diagram Components**

**Actors:**

* User / Customer
* System (Travel Management Software)
* Admin (implicit, for database operations)

**Main Use Cases:**

* Sign Up
* Login
* Recover Password
* View Destinations
* Book Package
* Book Hotel
* View Bookings
* Edit Personal Details
* Make Payment
* Logout

You can draw a simple UML diagram where the **User** is connected to all these use cases.

1. **Class Diagram**
2. **Login**
   * Handles user authentication
   * Opens Dashboard after validation
3. **Signup**
   * Registers new user information
4. **Dashboard**
   * Central menu to open all other modules
5. **AddCustomer / UpdateCustomer / ViewCustomer / DeleteDetails**
   * Manage all customer-related data
6. **BookPackage / BookHotel**
   * Handle booking operations
   * Calculate costs
   * Store bookings in database
7. **CheckPackage / CheckHotels**
   * Display available package and hotel options
8. **Payment / Paytm / QRWindow**
   * Payment interface using QR code
9. **EmailSender**
   * Sends email notifications (JavaMail API)
10. **Conn**
    * Database connection class for all SQL operations

**Implementation-**

1. Explanation of Java Code Modules

Login.java

* Handles user authentication.
* Accepts username and password from the user.
* Validates credentials using a SQL query.
* On success → opens the Dashboard.
* Includes exception handling for wrong passwords or connection issues.

Concepts Used: Swing GUI (JFrame, JTextField, JButton), JDBC, Event Handling.

Signup.java

* Allows new users to register by entering personal details.
* Stores data in MySQL using SQL INSERT.
* Provides a user-friendly form interface.

Concepts Used: Form-based GUI, SQL insertion, Input validation.

Dashboard.java

* Central menu after login.
* Provides navigation to all major features: Add Customer, View Customer, Book Package, Book Hotel, Check Prices, Payments, etc.

Concepts Used: Swing layouts, ActionListener, Modular window launching.

AddCustomer.java

* Form to collect customer details like ID, number, username, gender, country, etc.
* Stores the data in the database.

Concepts Used: Form GUI, SQL insertion, Handling dropdowns & radio buttons.

UpdateCustomer.java & DeleteDetails.java & ViewCustomer.java

* Manage already-saved customer details.
* Update uses SQL UPDATE.
* Delete uses SQL DELETE.
* View uses SQL SELECT and displays in labels or tables.

Concepts Used: Database CRUD operations, JTable, PreparedStatements.

BookPackage.java

* Lets users choose from predefined travel packages.
* Calculates total cost based on selection.
* Saves the booking to the database.

Concepts Used: Event handling, Conditional logic, Cost calculation.

BookHotel.java

* Similar to BookPackage.
* Shows hotels, calculates price based on AC/non-AC, food options, number of days.

Concepts Used: ComboBoxes, Calculations, SQL storage.

CheckPackage.java & CheckHotels.java

* Display available travel packages or hotels.
* Uses images and labels for attractive UI.

Concepts Used: Image handling (ImageIcon), Swing labels.

Payment.java & Paytm.java & QRWindow.java

* Payment section of the project.
* Displays a QR code for completing payments.
* Includes a “Done” button to confirm.

Concepts Used: Image scaling, Frames, Navigation.

EmailSender.java

* Sends email confirmations (e.g., Forgot Password).
* Uses the Jakarta Mail API.
* Connects to Gmail SMTP for sending mails.

Concepts Used: SMTP, Authenticator, MimeMessage, Exception Handling.

Conn.java

* Central database connection class.
* Uses JDBC with driver com.mysql.cj.jdbc.Driver.
* Provides a Connection and Statement object for all SQL operations.

Concepts Used: JDBC, Exception handling, Reusability.

Splash.java & Loading.java

* Startup UI screens.
* Provide a loading animation or introduction page.
* Used for better UX.

Concepts Used: Thread.sleep(), Swing splash screens, Threads.

Destinations.java

* Slideshow of different travel destinations.
* Automatically cycles images using a thread.
* Allows manual left-right navigation.

Concepts Used: Multithreading, Image sliders, Event handling.

2. Main Java Concepts Used

✔ Inheritance

Extending JFrame in almost every class to build GUI windows.

✔ Interfaces

ActionListener is implemented to handle button clicks.

✔ Exception Handling

Used extensively around database queries and email sending.

✔ GUI Components

Swing elements like: JLabel, JButton, JTextField, JComboBox, JTable, JRadioButton, ImageIcon, etc.

✔ JDBC (Java Database Connectivity)

Used for performing SQL queries on MySQL.

✔ Multithreading

Used in Loading.java and Destinations.java for delays and image rotations.

✔ Email API Integration

Using Jakarta Mail for sending emails.

3. Screenshots



A screenshot of a computer

AI-generated content may be incorrect.

A computer screen shot of a login page

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A long wooden walkway leading to a beach

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

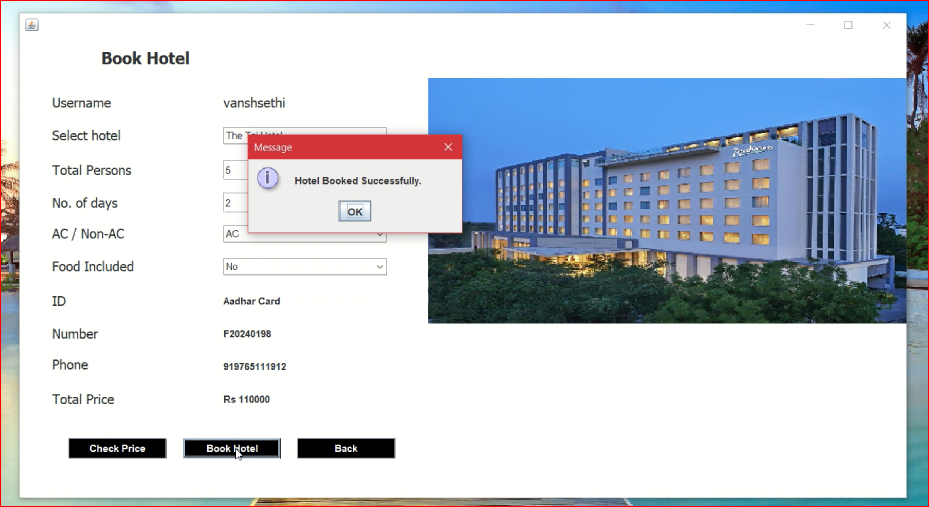
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a hotel

AI-generated content may be incorrect.

A pool next to a beach

AI-generated content may be incorrect.A screenshot of a hotel

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A long shot of a beach

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

**Conclusion-**

The Travel Management System successfully provides an efficient, user-friendly, and integrated solution for managing travel-related operations. Through its modular design and intuitive GUI, the system simplifies tasks such as customer registration, hotel booking, package booking, payment processing, and data management. By leveraging Java Swing for the interface, JDBC for database connectivity, and JavaMail API for email notifications, the project demonstrates how different Java technologies can work together to build a complete real-world application.  
Overall, the system helps reduce manual effort, minimizes errors, increases reliability, and improves the overall travel planning experience for both users and administrators.

**Future Work-**

Although the current system covers the essential functionalities, there are several enhancements that can be implemented to make it more robust and scalable:

1. Online Payment Gateway Integration

Instead of only QR-based payments, integrate UPI, credit/debit cards, or PayPal APIs.

1. Mobile App Version

Build an Android/iOS version using Java/Kotlin or Flutter for better accessibility.

1. Advanced Security Features

Password hashing, two-factor authentication (2FA), encrypted database storage.

1. Dynamic Package and Hotel Management

Admin dashboard to add/remove packages, hotels, and destinations without modifying the code.

1. Recommendation System

Suggest destinations based on user preferences using machine learning.

1. Invoice Generation

Automatically generate PDF invoices after booking.

1. Real-Time APIs

Use travel APIs to fetch live hotel availability, pricing, maps, etc.

1. Cloud Deployment

Host the system online and make the database cloud-based for multi-user access.

**References-**

1. Herbert Schildt, *Java: The Complete Reference*, McGraw-Hill Education – Used for understanding core Java concepts, Swing, exception handling, and OOP principles.
2. Oracle Java Documentation, *Java SE Technical Documentation* – Official reference for Swing components, event handling, and multithreading.  
   <https://docs.oracle.com/javase/>
3. MySQL Documentation, Oracle Corporation – Used for learning SQL queries, JDBC connectivity, and database operations.  
   <https://dev.mysql.com/doc/>
4. Jakarta Mail API Documentation, Eclipse Foundation – Used for implementing email-sending functionality such as Forgot Password.  
   https://eclipse-ee4j.github.io/mail/
5. GeeksforGeeks, Swing and JDBC Tutorials – Helped in understanding GUI design patterns and database connectivity examples.  
   <https://www.geeksforgeeks.org/>
6. Stack Overflow – Used for resolving coding errors, debugging, and optimizing SQL queries.  
   <https://stackoverflow.com/>
7. Various online Java forums and resources used for image handling, scaling, layout management, and troubleshooting issues during development.