## ****Abstract****

This project, Travel Management System, addresses the need for an organized and user-friendly platform for booking travel packages and hotels. The system allows users to sign up, log in, book packages or hotels, view and update their bookings, and process payments online. Using **Object-Oriented Programming (OOP) principles** such as classes, objects, encapsulation, and inheritance, the system is structured to be modular, maintainable, and scalable.

The implementation is done using **Java** with the **Swing** library for GUI, **NetBeans IDE** for development, and **MySQL** for backend database management connected via **JDBC**. Users interact with the system through graphical interfaces like login forms, booking forms, and information panels. Key outcomes include efficient booking management, secured user authentication, and easy access to travel information. The project demonstrates practical application of OOP concepts and database integration in real-world scenarios.

## ****Table of Contents****

**Abstract** ...................................…..………………………………….............………..1

**Introduction** .........................……………………………………................…………3

**System Analysis & Design** .......…………………………………..............………….4  
3.1 Use Case Diagram ...................……………….…….…………..…...…………….5  
3.2 Class Diagram .....................…………………….…………….............…….…….6  
3.3 Other UML Diagrams .............………………………………...........………....….8

**Implementation** ......................................……………………………………………10  
4.1 Screenshots of Running Program .............……………………………………….11  
4.2 Important Code Snippets ....................………………..………………………….15

**Testing & Results** .................................…………………………………………….17

**Conclusion & Future Work** .....................……….…………………………………18

**Full Source Code (GitHub Link)** .................………………………………………18

## ****Introduction****

**Background & Motivation:**  
Travel planning is often cumbersome and time-consuming. Users face challenges in managing bookings, comparing packages, and ensuring secure payment. This project aims to streamline the process, providing a unified platform to handle all travel-related tasks efficiently.

**Objectives:**

* Enable user registration, login, and authentication.
* Provide hotel and package booking features.
* Allow users to view and update bookings.
* Implement online payment functionality.

**Scope:**  
The system is designed for individual users to manage their travel plans efficiently. It can be extended to multiple users and integrated with more payment gateways or travel services in future.

## ****System Analysis & Design****

**Problem Definition:**  
Manual travel booking systems are prone to errors, duplication, and inefficiency. Users need a centralized platform for booking, viewing, and managing trips.

**Requirements:**

**Functional Requirements:**

* User login/signup.
* Booking and updating travel packages and hotels.
* Payment integration.
* Viewing booking history.

**Non-Functional Requirements:**

* GUI must be responsive and user-friendly.
* Database must securely store user data.
* System must handle multiple user requests efficiently.

**Use Case Diagram:**

**User**

**|**

**---------------------------------**

**| | | |**

**Login Sign Up Forget Password View Packages**

**|**

**Book Package**

**|**

**Payment**

**| | | |**

**View Hotels Book Hotel Update Customer View Booked Details**

**Use Cases (main functionalities):**

**Login** – access the system.

**Sign Up** – create a new account.

**Forget Password** – recover account password.

**View Packages** – see available travel packages.

**Book Package** – book a travel package.

**View Hotels** – see available hotels.

**Book Hotel** – book a hotel.

**Payment** – make payment for bookings.

**Update Customer Details** – update profile information.

**View Booked Details** – see bookings made.

**Class Diagram:**

| **Class: Login** |
| --- |
| **Attributes:** username, password |
| **Methods:** validateUser(), showLoginForm() |

| **Class: SignUp** |
| --- |
| **Attributes:** username, password, name, email, phone |
| **Methods:** registerUser(), showSignUpForm() |

| **Class: Customer** |
| --- |
| **Attributes:** customerID, name, email, phone, address |
| **Methods:** updateDetails(), viewDetails() |

| **Class: Customer** |
| --- |
| **Attributes:** customerID, name, email, phone, address |
| **Methods:** updateDetails(), viewDetails() |

| **Class: Booking** |
| --- |
| **Attributes:** bookingID, customerID, packageID, hotelID, totalCost, date |
| **Methods:** createBooking(), updateBooking(), viewBooking() |

| **Class: Package** |
| --- |
| **Attributes:** packageID, packageName, duration, cost |
| **Methods:** viewPackageDetails() |

| **Class: Hotel** |
| --- |
| **Attributes:** hotelID, hotelName, location, costPerNight |
| **Methods:** viewHotelDetails() |

| **Class: Payment** |
| --- |
| **Attributes:** paymentID, bookingID, amount, paymentStatus |
| **Methods:** makePayment(), verifyPayment() |

| **Class: Conn** |
| --- |
| **Attributes:** connection |
| **Methods:** executeQuery(), executeUpdate() |

### ****Relationships****

Customer → Booking (1 to many) – A customer can have multiple bookings.

Booking → Package (1 to 1) – Each booking is for a specific package.

Booking → Hotel (1 to 1) – Each booking may include a hotel.

Booking → Payment (1 to 1) – Each booking has one payment record.

Login → Customer – Login interacts with Customer for authentication.

SignUp → Customer – SignUp interacts with Customer for registration.

Conn → All database-related classes – Used for executing queries and updates.

**Other UML diagrams**

## ****Sequence Diagram – Hotel/Package Booking****

**Scenario:** User books a package or hotel.

**Objects:** User, Login, Dashboard, Booking, Payment, Confirmation

**Flow:**

**User Login Dashboard Booking Payment Confirmation**

**| | | | | |**

**|--Login info->| | | | |**

**| |--validate()-->|**

**| |<--success------|**

**| | |--openDashboard()-->|**

**| | | |**

**| | |--selectBooking()-->|**

**| | | |**

**| | |--enterBookingDetails()-->|**

**| | | |**

**| | |--proceedPayment()-->|**

**| | | |**

**| | |--paymentProcessed()-->|**

**| | | |**

**|<------------------------confirmation details--------------------|**

## ****Activity Diagram – Login Process****

**Scenario:** User tries to log in.

**Flow:**

**[Start]**

**|**

**[Enter Username & Password]**

**|**

**[Validate Credentials]**

**|**

**+----> [Valid?] ----No----> [Show Error & Retry] --> [Enter Username & Password]**

**|**

**Yes**

**|**

**[Open Dashboard]**

**|**

**[End]**

## ****Implementation****

**Programming Language & IDE:**

**Java (JDK 1.8+)**

**NetBeans IDE**

**Libraries & Tools:**

**Swing** for GUI

**JDBC** for database connectivity

**MySQL** for backend storage

**Key Classes & Responsibilities:**

| **Class Name** | **Responsibility** |
| --- | --- |
| Login | User authentication. |
| SignUp | Registration of new users. |
| Loading | Display loading screen with progress bar. |
| Dashboard | Main navigation panel for booking and viewing options. |
| Payment / Paytm | Handle payment processing. |
| UpdateCustomer | Update user details. |
| ViewCustomer | Display customer information. |
| ViewPackage | Display booked package details. |
| ViewBookedHotel | Display booked hotel details. |

**OOP Concepts Applied:**

**Encapsulation:** Private attributes with public getters/setters.

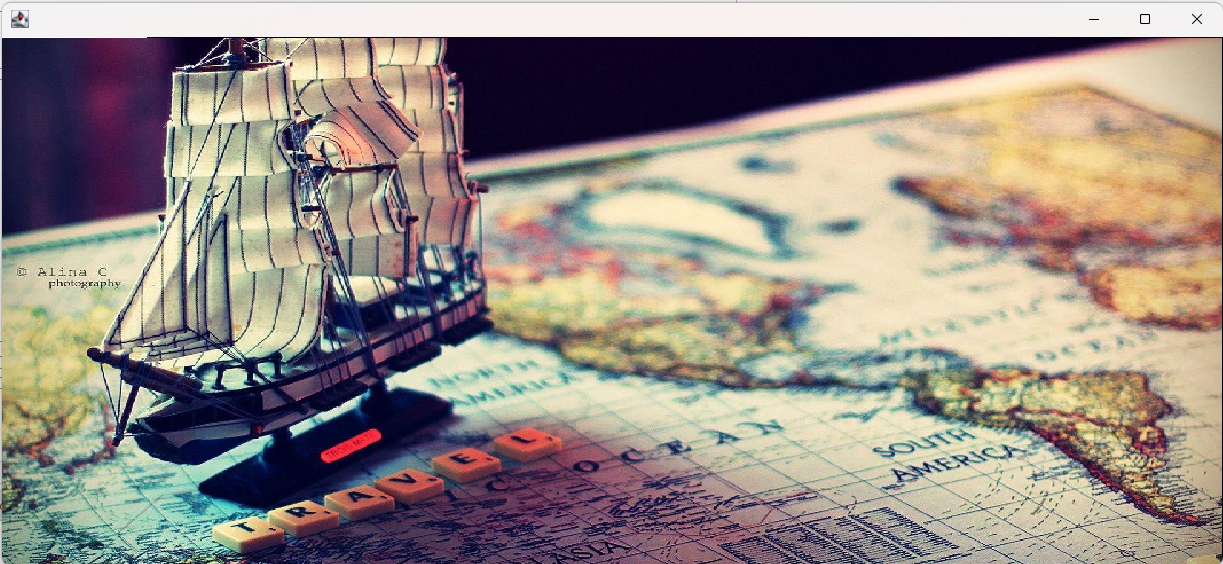
**Inheritance:** Common methods in parent classes (if implemented).

**Polymorphism:** Overloading for GUI actions (like button listeners).

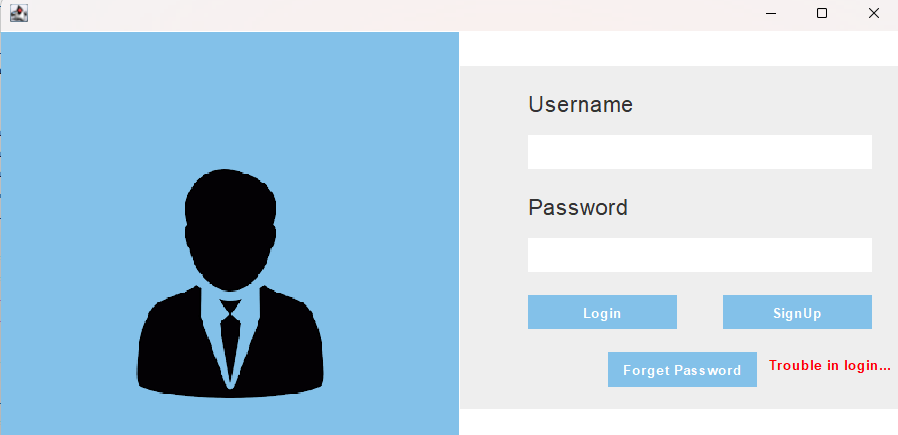
**Abstraction:** Abstracting database operations using Conn class.

**Screenshots of the running program:**

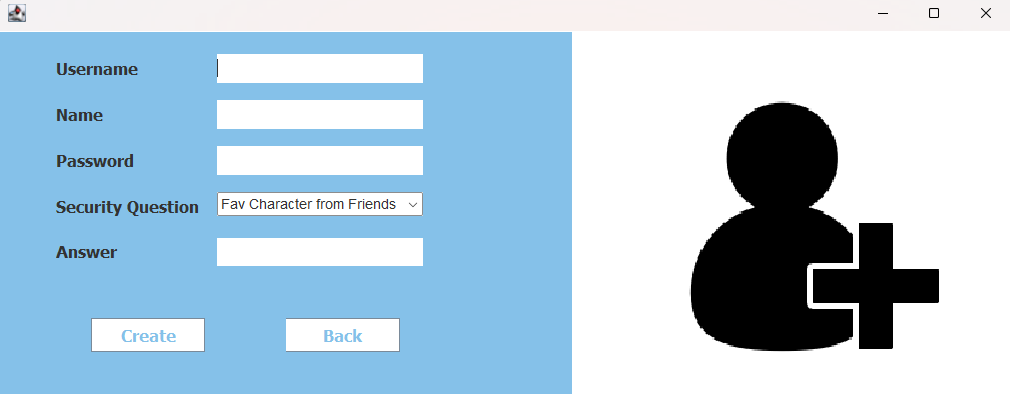
Splash / Loading Screen:



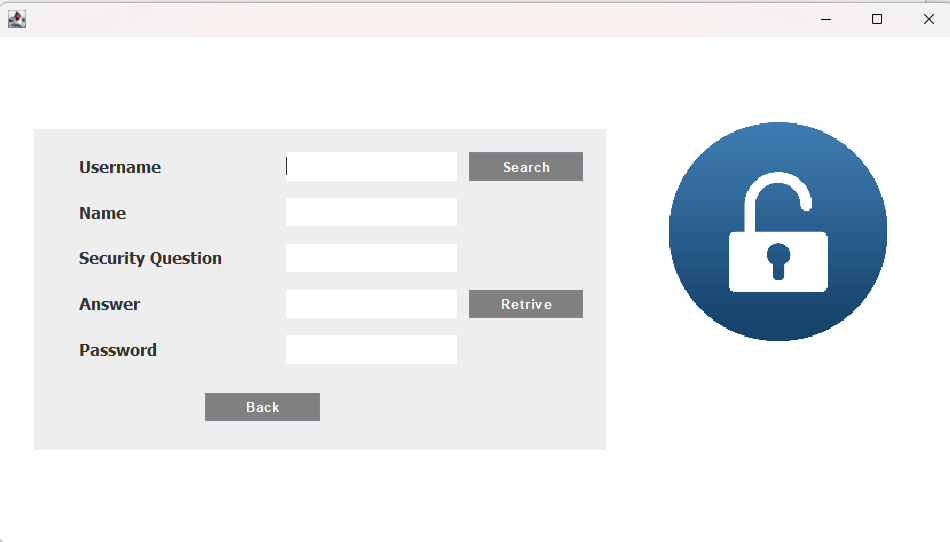
Login Window:



SignUp Window:



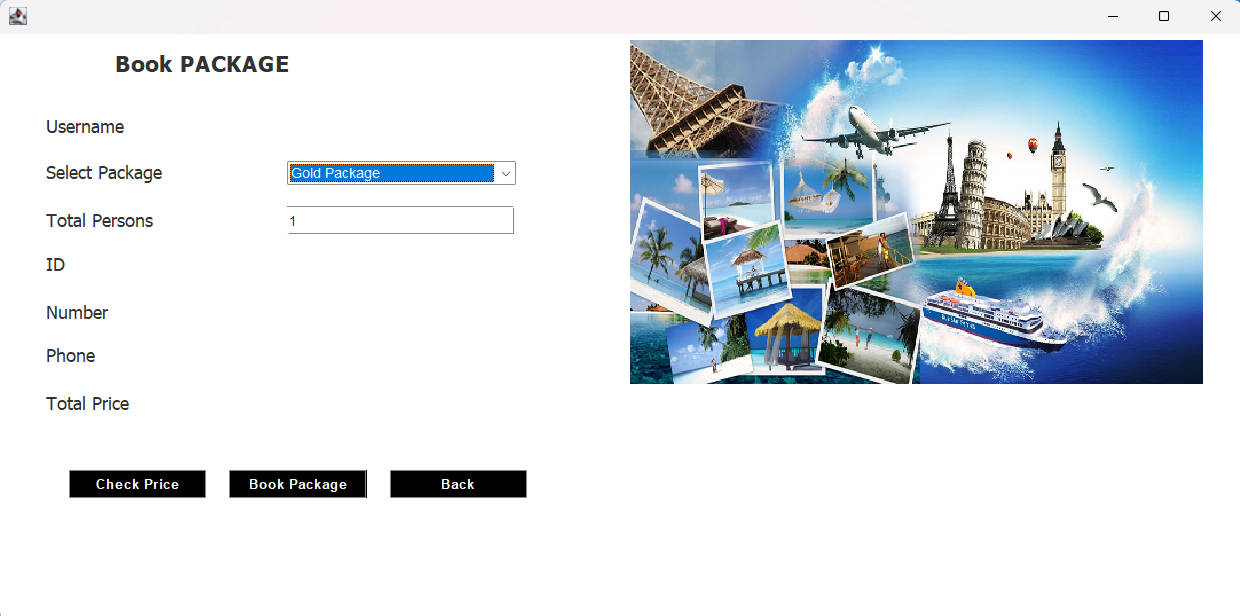
Forget Password Window:



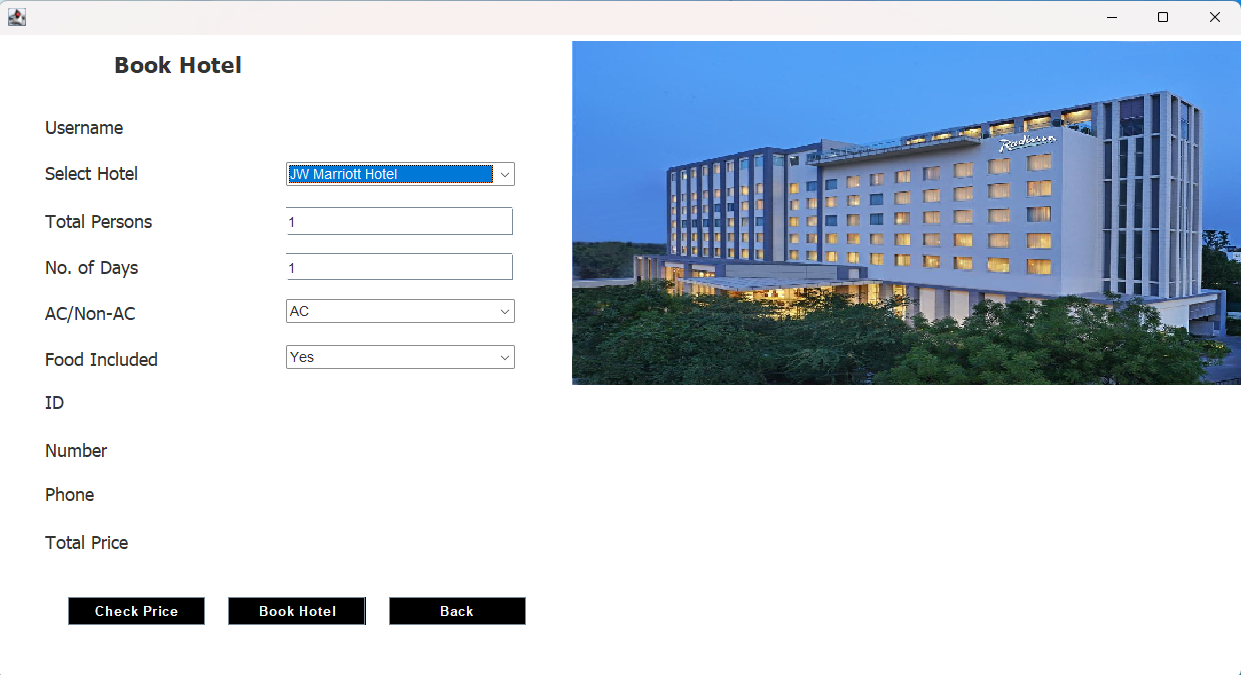
Dashboard / Main Menu:



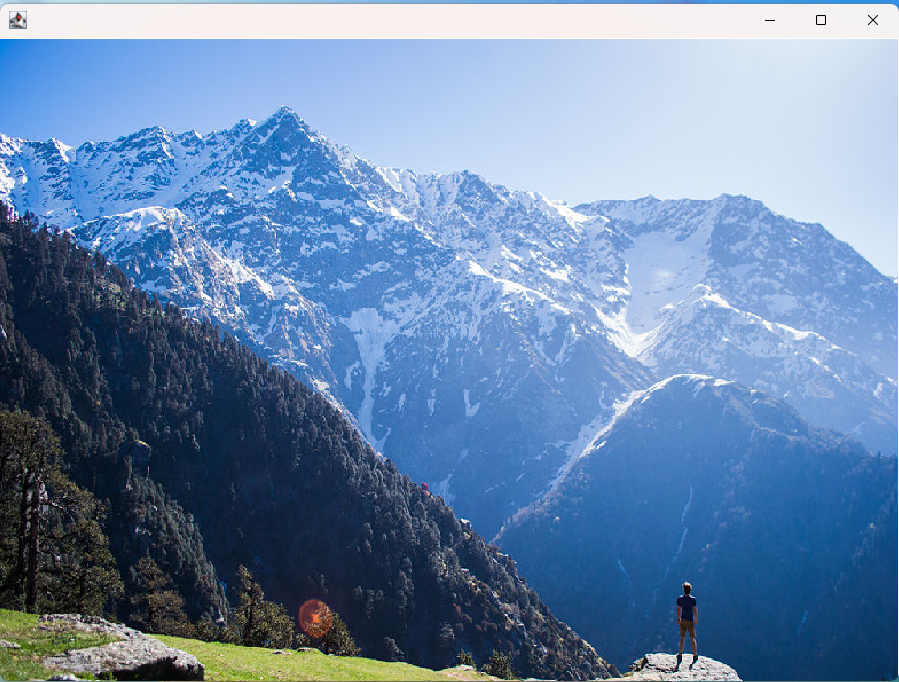
Book Package Window:



Book Hotel Window:



Destination Window( It slideshows with few pictures):



**Important code snippets**

Login Validation (Login.java):

String username = tfusername.getText();

String pass = tfpassword.getText();

String query = "select \* from account where username = '"+username+"' AND password= '"+pass+"'";

Conn c = new Conn();

ResultSet rs = c.s.executeQuery(query);

if(rs.next()){

setVisible(false);

new Loading(username);

}else{

JOptionPane.showMessageDialog(null,"Incorrect username or password");

}

**Why Important:** Validates user credentials and determines if the user can access the system.

SignUp / Registration (SignUp.java):

String username = tfusername.getText();

String name = tfname.getText();

String password = tfpassword.getText();

String question = security.getSelectedItem();

String answer = tfanswer.getText();

String query = "insert into account values('"+username+"','"+name+"','"+password+"','"+question+"','"+answer+"')" ;

try{

Conn c = new Conn();

c.s.executeUpdate(query);

JOptionPane.showMessageDialog(null,"Account Created Successfully");

setVisible(false);

new Login();

}

**Why Important:** Handles new user registration and stores user details in the database.

Payment Processing (Payment.java):

Conn c = new Conn();

c.s.executeUpdate("insert into bookpackage values('"+labelusername.getText()+"','"+cpackage.getSelectedItem()+"','"+tfpersons.getText()+"','"+labelid.getText()+"','"+labelnumber.getText()+"','"+labelphone.getText()+"','"+labelprice.getText()+"')");

JOptionPane.showMessageDialog(null, "Package Booked Successfully");

setVisible(false);

**Why Important:** Records payment information for the booking. Even if simulated, it shows payment functionality.

Update Customer Details(UpdateCustomer.java):

if(ae.getSource()==add){

String username = labelusername.getText();

String id = tfid.getText();

String number = tfnumber.getText();

String name = labelname.getText();

String gender= tfgender.getText();

String country = tfcountry.getText();

String address = tfaddress.getText();

String phone = tfphone.getText();

String email = tfemail.getText();

try{

Conn c = new Conn();

String query = "update customer set id='"+id+"', number='"+number+"', name='"+name+"', gender='"+gender+"', country='"+country+"', address='"+address+"', phone='"+phone+"', email='"+email+"' where username='"+username+"'";

c.s.executeUpdate(query);

JOptionPane.showMessageDialog(null, "Customer Details Updated Successfully");

setVisible(false);

}

catch(Exception e){

e.printStackTrace();

}

}else{

setVisible(false);

}

}

**Why Important:** Shows user can modify personal info after registration, demonstrating OOP and database update functionality.

## ****Testing & Results****

The Travel Management System was tested **manually** by interacting with the program to verify that all functionalities work as intended. Each feature was tested step by step, including user login, registration, package and hotel booking, and payment processing. Multiple users were able to use the system concurrently, and all interactions with the database were verified to ensure data was correctly stored and retrieved.

### ****Example Input / Output****

**Login:** Entering a valid username and password successfully logs in the user; invalid credentials display an error message.

**Booking Package / Hotel:** Selecting a package or hotel, entering the number of persons and days, and submitting produces a booking confirmation with the correct total cost.

**Payment:** Payment was processed via the real Paytm website, and successful transactions were reflected in the system.

**View Booked Details:** Users could retrieve and view their booked packages and hotel information correctly.

### ****Limitations****

* The GUI layout could be enhanced for a more modern and user-friendly appearance.
* Input validation can be further improved, for example, to prevent negative numbers or invalid characters.
* Advanced reporting features, such as generating booking summaries or invoices, are not implemented.

## ****Conclusion & Future Work****

**Achievements:**

* Successfully implemented a GUI-based travel management system.
* Applied OOP principles and database connectivity efficiently.
* Users can book packages/hotels, view and update details.

**Lessons Learned:**

* Integration of GUI, database and OOP concepts in a real project.
* Importance of exception handling and data validation.

**Future Work:**

* Multi-user and admin roles.
* Full payment gateway integration.
* Mobile application version for easier access.

## ****References****

1. Oracle Java Documentation, <https://docs.oracle.com/javase/>
2. MySQL Official Documentation, <https://dev.mysql.com/doc>
3. Java Swing Tutorials, <https://www.javatpoint.com/java-swing>
4. NetBeans IDE Documentation, <https://netbeans.apache.org>

**Appendices** :

Full source code : The complete source code for the Travel & Tourism Management System is available on GitHub. It includes all classes, database connection files and resources used in the project.

**GitHub Repository:**

[https://github.com/aurithri/Travel-Tourism-Management-System](https://github.com/aurithri/Travel-Tourism-Management-System" \t "_blank)

.