Travel Booking System Documentation

Project Overview

The Travel Booking System is a web application designed to allow users to search, book, and manage travel options such as flights, trains, and buses. It provides essential functionalities like user registration, authentication, ticket booking, and payment processing. The system is built using Java Servlets, JSP, and a MySQL database, ensuring a robust and scalable solution.

Features

Core Features

1. **User Management:**
   - User registration and profile management.
   - Login and authentication.
2. **Travel Booking:**
   - Search for travel options based on date, destination, and budget.
   - Book tickets and generate e-tickets.
3. **Payment Processing:**
   - Simulated payment integration for seamless transactions.
4. **Confirmation and Notifications:**
   - Booking confirmation with details.
   - Optional email notifications for booking confirmation.

Advanced Features

- Integration with third-party APIs for real-time travel data.
- Multi-language support.
- Mobile responsiveness.

Technologies Used

Backend

- Java Servlets for handling requests and responses.
- JSP (Java Server Pages) for dynamic content generation.
- MySQL as the database for storing user and booking details.

Frontend

- HTML, CSS, and Bootstrap for user interface.
- JavaScript for client-side validation and interactivity.

Tools

- Apache Maven for project management and build automation.
- Tomcat Server for deployment.
- JUnit for unit testing.

Directory Structure

```
TravelBookingSystem/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   ├── com/
│   │   │   │   ├── travelbooking/
│   │   │   │   │   ├── controllers/
│   │   │   │   │   │   ├── BookingServlet.java
│   │   │   │   │   │   ├── UserServlet.java
│   │   │   │   │   ├── dao/
│   │   │   │   │   │   ├── BookingDAO.java
│   │   │   │   │   │   ├── UserDAO.java
│   │   │   │   │   ├── models/
│   │   │   │   │   │   ├── Booking.java
│   │   │   │   │   │   ├── User.java
│   │   │   │   │   ├── services/
│   │   │   │   │   │   ├── BookingService.java
│   │   │   │   │   │   ├── UserService.java
│   │   │   │   │   ├── utils/
│   │   │   │   │   │   ├── DBConnection.java
│   ├── main/
│   │   ├── resources/
│   │   │   ├── application.properties
│   │   │   ├── database.sql
│   │   ├── webapp/
│   │   │   ├── WEB-INF/
│   │   │   │   ├── web.xml
│   │   │   ├── pages/
│   │   │   │   ├── index.jsp
│   │   │   │   ├── login.jsp
│   │   │   │   ├── register.jsp
│   │   │   │   ├── booking.jsp
│   │   │   │   ├── confirmation.jsp
├── pom.xml
├── README.md
```

Database Schema

Tables

1. **Users:**
   - id (INT, PRIMARY KEY, AUTO_INCREMENT)

    - name (VARCHAR)
    - email (VARCHAR, UNIQUE)
    - password (VARCHAR)
2. **Bookings:**
    - id (INT, PRIMARY KEY, AUTO_INCREMENT)
    - user_id (INT, FOREIGN KEY REFERENCES Users(id))
    - destination (VARCHAR)
    - date (DATE)
    - amount (DECIMAL)

UML Class Diagram for a Travel Booking System

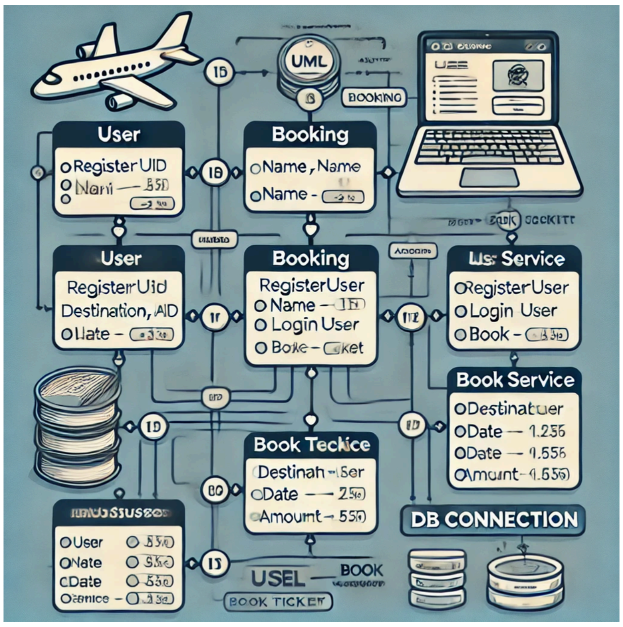
This diagram illustrates the key classes and their relationships within a simplified travel booking system.

Classes:

- **Customer:**
    - Attributes: CustomerID, Name, Contact, Address, Email, Password
    - Methods: Register, Login, ViewProfile, UpdateProfile
- **Flight:**
    - Attributes: FlightID, Origin, Destination, DepartureTime, ArrivalTime, Price, Airline
    - Methods: SearchFlights, BookFlight, CancelFlight
- **Hotel:**
    - Attributes: HotelID, Name, Location, StarRating, RoomTypes, Prices
    - Methods: SearchHotels, BookHotel, CancelHotel
- **Package:**
    - Attributes: PackageID, Name, Description, Price, Includes (flights, hotels, activities)
    - Methods: SearchPackages, BookPackage, CancelPackage
- **Booking:**
    - Attributes: BookingID, CustomerID, TravelType (flight, hotel, package), Date, Status, TotalPrice
    - Methods: MakeBooking, CancelBooking, ViewBooking
- **Payment:**
    - Attributes: PaymentID, BookingID, Amount, PaymentMethod, Status
    - Methods: MakePayment

Relationships:

- **Customer** has many **Bookings**
- **Booking** belongs to a **Customer**
- **Booking** can be for a **Flight**, **Hotel**, or **Package**
- **Flight**, **Hotel**, and **Package** can have many **Bookings**



Setup and Execution

Prerequisites

- Java JDK 11 or higher.
- Apache Maven.
- MySQL Server.
- Apache Tomcat Server.

Steps to Run

1. Clone the repository:
2. git clone https://github.com/your-repository/travel-booking-system.git
3. Import the project into your IDE (e.g., IntelliJ IDEA, Eclipse).
4. Configure the database:
   - Create a database named travel_booking.
   - Run the SQL script from src/main/resources/database.sql to create the necessary tables.
5. Update database credentials in DBConnection.java.
6. Build the project using Maven:
7. mvn clean install
8. Deploy the application to the Tomcat server.
9. Access the application at:
10. http://localhost:8080/TravelBookingSystem

Testing

Unit Tests

- JUnit is used for testing the DAO and service layers.

- Sample test cases are included in the src/test/java directory.

Manual Testing

- Validate user registration and login.
- Test booking creation with valid and invalid inputs.
- Verify database updates after each booking.

Future Enhancements

- Implement OAuth for social media login.
- Add support for dynamic pricing based on demand.
- Integrate real-time notifications using WebSocket.

Contributors

- Priyanka Patel
- Nistha Sharma
- Suhani Verma
- Vidisha Maurya

License

This project is licensed under the MIT License. See the LICENSE file for details.