1. Project Overview

Project Title: Tourism Management System
Language Used: Java (JDK 17)
Database: MySQL
IDE: Eclipse (with Java Project extension)

The **Tourism Management System** is a Java-based desktop application designed to streamline the operations of travel agencies. It helps manage the wide variety of tasks typically associated with tourism businesses, such as managing customer profiles, tour packages, bookings, payments, and destinations.

This system reduces manual work and data entry errors by offering a centralized platform where admins can manage all services, and users can easily search, book, and pay for travel packages. The system not only boosts operational efficiency but also enhances the user experience through an intuitive and responsive interface.

The application features a role-based structure where admin and end users have different privileges. Admin users can control core functionalities like adding packages and destinations, viewing reports, managing bookings, and overseeing user activity. Regular users can register, login, browse packages, make bookings, and process payments.

This project follows a modular structure to ensure maintainability and scalability. Each module, like user management or bookings, is isolated and can be extended without impacting the rest of the system.

2. Technologies Used

The **Tourism Management System** is built using a modern set of technologies that ensure robustness, scalability, and performance. Each technology serves a specific role in the system's architecture. Here is a detailed explanation of the technologies involved:

Technology	Description
Java 17 (JDK)	Java is the core programming language used for developing the application. Version 17 is a Long-Term Support (LTS) release, offering new features and performance improvements. Java's platform independence and object-oriented features make it ideal for this project.
MySQL	MySQL is used as the backend relational database management system (RDBMS). It stores all persistent data, such as users, bookings, packages, and payment records. The schema is designed to normalize data and enforce relationships through foreign keys.
JDBC (Java Database Connectivity)	JDBC is the Java API used to connect and interact with the MySQL database. It allows the application to execute SQL queries, retrieve data, and perform CRUD operations directly from the Java code.
Eclipse IDE	Eclipse is the integrated development environment used to write, debug, and manage the Java code. It offers powerful features like auto-

Technology	Description
	completion, refactoring, and Maven integration, improving development productivity.
Swing / AWT (Abstract Window Toolkit)	Swing and AWT are used for building the desktop Graphical User Interface (GUI). Swing components like JFrame, JButton, JTable, and JTextField provide a rich UI experience, while AWT handles basic event-driven interactions.

These technologies are chosen for their compatibility, ease of use, and support within the Java ecosystem. The combination of a powerful programming language, efficient database system, and well-structured GUI ensures a high-quality and responsive tourism management system.

3. Modules

The **Tourism Management System** is organized into several independent and interrelated modules. Each module is responsible for handling a specific set of functionalities. The modular structure ensures clean separation of concerns, easy maintenance, and scalability.

3.1 User Management Module

This module handles all aspects of user registration, authentication, and profile management. It plays a vital role in providing access control and user-specific features.

Features:

- **User Registration**: Allows new users to register by providing their name, email, phone number, and password.
- **Login/Logout**: Secure login system with basic validation. Logout ensures session termination.
- **Profile Management**: Users can view and update their personal information like email or contact details.
- **User Roles**: Differentiates between normal users and administrators, offering role-based access to features.

3.2 Package Management Module

This module is primarily accessible to the admin and is used to manage tour packages. It ensures the travel agency can update its offerings as needed.

Features:

- **Add Package**: Admin can add new tour packages with details like name, description, price, duration, and destination.

- **Update/Delete Package**: Modify or remove packages based on availability or changes in itinerary.
- **View Packages**: Both users and admins can browse through available packages with complete details.

3.3 Booking System Module

This is the core module for customers, allowing them to book tours and manage their bookings.

Features:

- **Tour Booking**: Users can book packages of their choice. On submission, the booking is stored and linked with the user's ID.
- **Booking History**: Users can view their previous and current bookings with status indicators.
- **Cancel Bookings**: Bookings can be canceled before a certain deadline. The status of the booking is updated accordingly.

3.4 Destination Module

This module helps users explore destinations before booking and provides admins with control to manage destination information.

Features:

- **Destination Listing**: Displays a list of tourist destinations with images, descriptions, and location tags.
- **Filter/Search**: Users can search destinations by state, category, or attraction type.
- **Admin Controls**: Admin can add or remove destinations and update details as required.

3.5 Payment Module

This module records and tracks payment details related to each booking. Though it's currently offline/manual, the structure is ready for future integration with payment gateways.

Features:

- **Payment Entry**: Admin enters the payment details such as amount, method (e.g., cash, bank transfer), and date.
- **Status Tracking**: Tracks payment status (Success, Pending, Failed). This is important for generating revenue reports.
- **User View**: Users can view payment receipts/status linked to their bookings.

3.6 Admin Dashboard Module

This is the central hub for administrative users. It provides full access to the system’s core data and operations.

Features:

- **View All Users and Bookings**: Admins can access a list of all registered users and bookings.
- **Generate Reports**: Admins can generate summary reports on payments, bookings, and user activity.
- **Add/Remove Packages and Destinations**: Full CRUD functionality for core business data.

4. Folder Structure

After downloading and extracting the project ZIP file, you'll find a structured folder layout that adheres to clean coding principles and project organization standards in Java. Below is the overview of the project structure, followed by detailed descriptions of each major folder and file.

```
TourismMS/
 ├── src/
 |    ├── Main.java
 |    ├── db/              # DB connection setup and utilities
 |    ├── gui/             # Graphical User Interface (GUI) classes using Swing
 |    ├── model/           # Data models / POJOs (Plain Old Java Objects)
 |    └── services/        # Business logic and service classes
 ├── lib/                  # External libraries (if any, e.g., MySQL Connector JAR)
 ├── tourism.sql           # MySQL schema and seed data
 └── README.md             # Project instructions and metadata
```

src/

This is the main source code folder. All Java files and packages are kept here. It contains subfolders for better code management:

Main.java

This is the main entry point of the application. It initializes the GUI and manages the starting point of user interaction. All other modules are called from here.

db/

This package includes classes responsible for database connectivity and utility functions. It includes:

- DBConnection.java: A class that handles connection to the MySQL database using JDBC.
- Optional utility classes for closing connections or executing common queries.

gui/

This folder contains all user interface files developed using Java Swing (or AWT). Each form/window (e.g., login screen, booking form, admin dashboard) is implemented as a class here.
Examples:

- LoginForm.java
- Dashboard.java
- PackageView.java
- BookingForm.java

model/

This folder contains POJO classes representing the structure of key entities in the application. These classes are used to encapsulate data and share it across layers.
Example model classes:

- User.java
- Package.java
- Booking.java

- Payment.java
- Destination.java

services/

This package implements the business logic of the system. It acts as an intermediary between the gui and db packages, processing data before it's stored or displayed.
Examples:

- UserService.java
- BookingService.java
- PaymentService.java
- AdminService.java

---

lib/

This folder is used to store external libraries that the project depends on. For this project, the most common library here would be the **MySQL JDBC Driver (e.g., mysql-connector-java-x.x.x.jar)**, which enables Java to connect with MySQL databases.

---

tourism.sql

This file contains the full **MySQL schema** and **sample data** for the application. It includes the creation of all necessary tables (users, packages, bookings, etc.), along with INSERT statements to populate the tables with sample data.

This file should be executed in a MySQL database tool like:

- MySQL Workbench
- phpMyAdmin

- Command Line MySQL Client

README.md

This is a documentation file that includes basic setup instructions, usage guidelines, and contact information. It is especially helpful for users who download the source code from GitHub or any project-sharing platform.

This structure ensures a clean separation of concerns:

- **model**: Data representation
- **services**: Business logic
- **gui**: User interaction
- **db**: Persistence layer

5. Database Design

The **database** is the heart of the Tourism Management System. It stores all the essential information—user data, travel packages, booking records, payments, and destination details. The system uses **MySQL** as the relational database management system, ensuring data integrity, relationships, and optimized queries.

Database Name: tourism_db

All tables are created inside a database named tourism_db. Make sure this database is created in your MySQL environment before importing the tourism.sql file provided with the project.

Key Tables & Schema Design

Below is a list of essential tables with their fields, relationships, and descriptions:

5.1 users Table

Field	Type	Description
id	INT (PK)	Primary Key (auto-incremented user ID)
name	VARCHAR(100)	Full name of the user
email	VARCHAR(100)	Unique user email
password	VARCHAR(100)	Encrypted password (or plain in demo)
phone	VARCHAR(15)	Contact number
user_type	VARCHAR(20)	Role: admin or user

Field	Type	Description

- **Purpose**: Stores all system users and their roles.
- **Constraint**: Email must be unique.

5.2 packages Table

Field	Type	Description
id	INT (PK)	Unique ID for each tour package
name	VARCHAR(100)	Package title (e.g., “Goa Beach Tour”)
description	TEXT	Detailed itinerary or features
price	DECIMAL(10,2)	Total cost for the package
duration	VARCHAR(50)	Duration (e.g., “3 Nights 4 Days”)
location	VARCHAR(100)	Main destination (city/state)

- **Purpose**: Stores all available tour packages.
- **Relation**: Connected to bookings via package_id.

5.3 bookings Table

Field	Type	Description
id	INT (PK)	Booking ID
user_id	INT (FK)	Foreign Key → users.id
package_id	INT (FK)	Foreign Key → packages.id

Field	Type	Description
status	VARCHAR(20)	Booking status (e.g., Confirmed, Cancelled)
date	DATE	Booking date

- **Purpose**: Logs all booking transactions.
- **Relation**: Connects users to packages.

5.4 payments Table

Field	Type	Description
id	INT (PK)	Payment ID
booking_id	INT (FK)	Foreign Key → bookings.id
amount	DECIMAL(10,2)	Amount paid
method	VARCHAR(50)	Payment method (Cash, Card, Bank Transfer)
payment_date	DATE	Date of payment
status	VARCHAR(20)	Payment status (Success, Pending, Failed)

- **Purpose**: Manages all payment records.
- **Relation**: Tied to bookings via booking_id.

5.5 destinations Table

Field	Type	Description
id	INT (PK)	Unique ID for each destination
name	VARCHAR(100)	Name of the place (e.g., “Manali”)

Field	Type	Description
state	VARCHAR(100)	State or region
description	TEXT	Overview of the destination
image_url	TEXT	Link to an image (optional for GUI rendering)

- **Purpose**: Used to display tourist destinations.
- **Relation**: Linked to packages by shared location name.

Relationships and Keys

- **1-to-Many**: A user can have multiple bookings.
- **1-to-1**: Each booking has one associated payment.
- **1-to-Many**: A package can be booked by many users.

Sample SQL Setup

The project includes a tourism.sql file which:

- Creates all tables
- Adds sample admin and user accounts
- Adds a few packages, destinations, and test bookings

You can import it via:

- **MySQL Workbench**: File > Open SQL Script > Run
- **phpMyAdmin**: Import tab > Select tourism.sql
- **Command line**: mysql -u root -p tourism_db < tourism.sql

6. How to Run

This section will guide you step-by-step through setting up and running the **Tourism Management System** on your local machine using Java 17, MySQL, and Eclipse IDE. Whether you're a developer, student, or project evaluator, this guide ensures a smooth and complete setup.

6.1 Prerequisites

Before starting, ensure the following tools and environments are properly installed and configured:

Requirement	Description
Java 17 (JDK)	Install the Java Development Kit 17 (LTS version). Confirm installation with java -version in the terminal.
MySQL Server	Install MySQL Server (version 5.7 or 8.0). Ensure it's running and accessible on localhost:3306.
Eclipse IDE for Java Developers	Download and install Eclipse. The "Eclipse IDE for Java Developers" edition is preferred.
MySQL JDBC Driver	Download the MySQL Connector JAR (mysql-connector-java-8.x.x.jar) and place it in the lib/ folder.

6.2 Import the Project into Eclipse

1. **Launch Eclipse IDE.**
2. Go to File → Import…

3. Choose: Existing Projects into Workspace → Click **Next**.
4. Select root directory: Browse to the extracted TourismMS/ project folder.
5. Check the project box and click **Finish**.

Your project will appear in the Project Explorer pane.

6.3 Configure the Build Path

1. Right-click on your project → Build Path → Configure Build Path.
2. Go to the Libraries tab.
3. Click Add External JARs… → Select the MySQL Connector JAR from your lib/ folder.
4. Click **Apply and Close**.

This step is necessary for JDBC database access.

6.4 Set Up the MySQL Database

1. Open MySQL Workbench or any MySQL tool.
2. Create the database manually (if not already created):

sql

CopyEdit

CREATE DATABASE tourism_db;

3. Import the schema:
    - Open tourism.sql provided in the ZIP.
    - Execute it to create tables and insert sample data.

The database now contains:

- Users
- Tour packages
- Destinations
- Bookings
- Payments

6.5 Configure Database Connection in Code

1. Navigate to src/db/DBConnection.java (or similar class).
2. Edit the connection string as per your local MySQL credentials:

java

String url = "jdbc:mysql://localhost:3306/tourism_db";

String user = "root";

String password = "your_password"; // Change as per your MySQL root password

3. Save the file.

Make sure your MySQL server is running and accessible on port 3306.

6.6 Run the Project

1. Locate Main.java inside the src/ directory.
2. Right-click on Main.java → Run As → Java Application.

If everything is configured correctly, the GUI will launch showing either a login or dashboard screen depending on the default mode.

6.7 Default Login (Sample Data)

If using the sample SQL file:

- **Admin Login**:
  - Email: admin@example.com
  - Password: admin123
- **User Login**:
  - Email: user@example.com
  - Password: user123

(You can change these in the database or through the GUI.)

6.8 Common Issues & Fixes

Issue	Solution
"ClassNotFoundException for com.mysql.cj.jdbc.Driver"	Ensure the MySQL JDBC JAR is added to your classpath.
"Access denied for user"	Check MySQL username/password in DBConnection.java.
GUI not launching	Make sure Main.java contains a proper main() method initializing the GUI.

7.Sample Screenshots

7.1 Login / Register Screen

Description:
This is the first screen that users interact with. Users can either log in using their existing credentials or click the "Register" button to sign up. Validation is performed to check if the credentials are correct and match entries in the database.

7.2 User Registration Page

Description:
The registration form allows new users to create an account. Required fields include name, email, password, and phone number. Once submitted, the data is stored in the users table in the database.

7.3 Admin Dashboard

Description:
Once logged in as an admin, the dashboard provides full control over the system. Features include viewing users, managing packages, monitoring bookings, and generating reports. It serves as the control panel for the entire application.

7.4 Package Listing View

Description:
Displays a list of all available tour packages. Each package shows its name, price, duration, description, and destination. Users can browse through packages and click to view more details or initiate a booking.

7.5 Booking Window

Description:
Users can select a package and proceed to book it by choosing available dates. The booking is then stored in the bookings table, associated with the user's ID. A confirmation message is displayed after successful booking.

7.6 Payment Details Page

Description:
This section allows users (or admins on behalf of users) to enter payment details. It records the booking ID, amount paid, method (Cash/Card/Bank), and date. The payment status (Success/Pending) is also shown or updated.

7.7 Booking History for Users

Description:
Users can view their booking history, including booking status (Confirmed, Cancelled), package details, and payment info. This enhances transparency and user trust in the system.

7.8 Admin - Manage Packages Interface

Description:
Admins use this form to add, update, or delete tour packages. It includes input fields for name, price, duration, and destination. Once submitted, the package is added to the database and appears in the listing view.