Maven Based Java Project for Airline Management System

Name: Lande Sudam

Roll No: 94

# 1. Introduction

This documentation presents a comprehensive overview of a Java-based Airline Management System project developed using Apache Maven. The aim of this project is to simulate a simple airline application that includes functionalities such as adding flights, booking and canceling tickets, and displaying flight/passenger details. By using Maven, the project benefits from simplified build management, dependency control, and a standardized structure.

**2. Objectives**

* To develop a modular and maintainable airline system using Java.
* To utilize Apache Maven for efficient project management and build automation.
* To apply Object-Oriented Programming (OOP) principles in real-world applications.
* To implement basic airline functionalities through a console-based interface.
* To manage external libraries and dependencies using Maven.

**3. Technologies Used**

* Programming Language: Java
* Build Tool: Apache Maven
* IDE: IntelliJ IDEA / Eclipse
* JDK Version: 11 or higher
* Dependencies: JUnit, MySQL Connector (optional), Log4j (optional)

**4. Key Maven Concepts**

* **POM (Project Object Model)**: Central configuration file (pom.xml) containing project info and build settings.
* **Dependencies**: External libraries required for the project to compile and run.
* **Build Lifecycle**: Phases like validate, compile, test, package, verify, install, deploy.
* **Repositories**: Locations (local, central, remote) where Maven retrieves dependencies.

**5. Project Structure**

bash

CopyEdit

AirlineManagementSystem/

├── src/

│ ├── main/java/com/airline/ # Java source files

│ └── test/java/com/airline/ # Unit tests

├── target/ # Compiled bytecode and packaged files

├── pom.xml # Maven configuration file

**6. Implementation Details**

**Modules and Functionalities:**

1. **Flight Module**: Add a new flight, display flight details.
2. **Transaction Module**: Book ticket, cancel booking, view transaction history.
3. **Admin Module (Optional)**: Admin login, view all passengers and flights.
4. **Database Integration (Optional)**: Store and retrieve data using MySQL.

**Sample Java Class – Flight.java**:

java

CopyEdit

public class Flight {

private String flightId;

private String destination;

private String departureTime;

private int capacity;

public Flight(String flightId, String destination, String departureTime, int capacity) {

this.flightId = flightId;

this.destination = destination;

this.departureTime = departureTime;

this.capacity = capacity;

}

public void bookSeat() {

if (capacity > 0) capacity--;

else System.out.println("Flight fully booked!");

}

public void cancelSeat() {

capacity++;

}

public void displayDetails() {

System.out.println("Flight ID: " + flightId);

System.out.println("Destination: " + destination);

System.out.println("Departure Time: " + departureTime);

System.out.println("Available Seats: " + capacity);

}

}

**Main Class – Main.java**:

java

CopyEdit

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Flight flight = null;

while (true) {

System.out.println("\n--- Airline Management Menu ---");

System.out.println("1. Add Flight");

System.out.println("2. Book Seat");

System.out.println("3. Cancel Seat");

System.out.println("4. Display Flight Details");

System.out.println("5. Exit");

int choice = sc.nextInt();

switch (choice) {

case 1:

sc.nextLine();

System.out.print("Enter Flight ID: ");

String id = sc.nextLine();

System.out.print("Enter Destination: ");

String destination = sc.nextLine();

System.out.print("Enter Departure Time: ");

String time = sc.nextLine();

System.out.print("Enter Capacity: ");

int cap = sc.nextInt();

flight = new Flight(id, destination, time, cap);

break;

case 2:

if (flight != null) flight.bookSeat();

else System.out.println("Add flight first.");

break;

case 3:

if (flight != null) flight.cancelSeat();

else System.out.println("Add flight first.");

break;

case 4:

if (flight != null) flight.displayDetails();

else System.out.println("Add flight first.");

break;

case 5:

System.out.println("Thank you for using Airline Management System.");

System.exit(0);

break;

default:

System.out.println("Invalid choice!");

}

}

}

}

**7. Steps to Create Maven Project**

**Using Command Line:**

bash

CopyEdit

mvn archetype:generate -DgroupId=com.airline \

-DartifactId=AirlineManagementSystem \

-DarchetypeArtifactId=maven-archetype-quickstart \

-DinteractiveMode=false

cd AirlineManagementSystem

mvn compile

mvn package

**8. Sample pom.xml File**

xml

CopyEdit

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.airline</groupId>

<artifactId>AirlineManagementSystem</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**9. Conclusion**

The Maven-based Java Airline Management System provides a hands-on understanding of building Java applications with real-world functionalities. It showcases how to automate builds, manage dependencies, and maintain code in a clean, modular format. This project strengthens Java and Maven skills and promotes standard practices used in software development.

**10. References**

1. <https://maven.apache.org/guides/index.html>
2. <https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html>
3. <https://docs.oracle.com/en/java/>
4. <https://www.baeldung.com/maven>
5. <https://www.geeksforgeeks.org/apache-maven/>
6. <https://www.tutorialspoint.com/maven/index.htm>