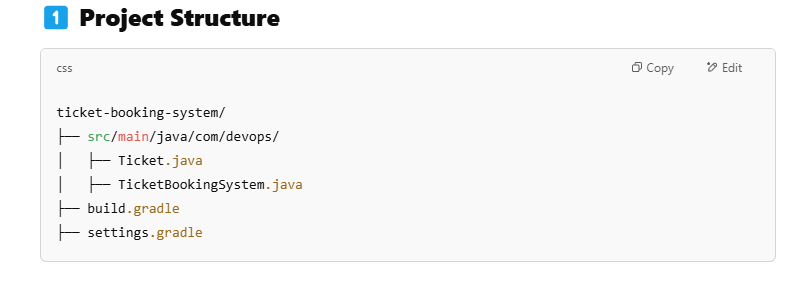
**MAVEN &**

**Java Code: Student Admission CLI**

**This Maven-based Java CLI application manages student records using file storage**



****

**Step 1:**

**1️⃣ Create pom.xml (Maven Configuration)**

|  |
| --- |
| **<?xml version="1.0" encoding="UTF-8"?>**  **<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.devops</groupId>**  **<artifactId>student-admission-cli</artifactId>**  **<version>1.0-SNAPSHOT</version>**  **<packaging>jar</packaging>**  **<dependencies>**  **<!-- Lombok for reducing boilerplate code -->**  **<dependency>**  **<groupId>org.projectlombok</groupId>**  **<artifactId>lombok</artifactId>**  **<version>1.18.26</version>**  **<scope>provided</scope>**  **</dependency>**  **</dependencies>**  **<build>**  **<plugins>**  **<!-- Maven Compiler Plugin -->**  **<plugin>**  **<groupId>org.apache.maven.plugins</groupId>**  **<artifactId>maven-compiler-plugin</artifactId>**  **<version>3.8.1</version>**  **<configuration>**  **<source>17</source>**  **<target>17</target>**  **</configuration>**  **</plugin>**  **</plugins>**  **</build>**  **</project>** |

**2️⃣ Java Code: Student Admission CLI**

**This Maven-based Java CLI application manages student records using file storage.**

package com.devops;

import java.io.\*;

import java.util.\*;

class Student {

private int id;

private String name;

private int age;

private String course;

public Student(int id, String name, int age, String course) {

this.id = id;

this.name = name;

this.age = age;

this.course = course;

}

@Override

public String toString() {

return id + ", " + name + ", " + age + ", " + course;

}

}

public class StudentAdmissionCLI {

private static final String FILE\_NAME = "students.txt";

private static final Scanner scanner = new Scanner(System.in);

public static void main(String[] args) {

while (true) {

System.out.println("\nStudent Admission System");

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

System.out.println("2. Display Students");

System.out.println("3. Search Student by ID");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

addStudent();

break;

case 2:

displayStudents();

break;

case 3:

searchStudentById();

break;

case 4:

System.out.println("Exiting...");

return;

default:

System.out.println("Invalid choice. Try again.");

}

}

}

private static void addStudent() {

try (FileWriter fw = new FileWriter(FILE\_NAME, true);

BufferedWriter bw = new BufferedWriter(fw);

PrintWriter out = new PrintWriter(bw)) {

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

int id = scanner.nextInt();

scanner.nextLine();

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

String name = scanner.nextLine();

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

int age = scanner.nextInt();

scanner.nextLine();

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

String course = scanner.nextLine();

Student student = new Student(id, name, age, course);

out.println(student);

System.out.println("Student added successfully!");

} catch (IOException e) {

System.out.println("Error writing to file.");

}

}

private static void displayStudents() {

try (BufferedReader br = new BufferedReader(new FileReader(FILE\_NAME))) {

String line;

System.out.println("\nList of Students:");

while ((line = br.readLine()) != null) {

System.out.println(line);

}

} catch (IOException e) {

System.out.println("No students found.");

}

}

private static void searchStudentById() {

System.out.print("Enter Student ID to search: ");

int searchId = scanner.nextInt();

scanner.nextLine();

try (BufferedReader br = new BufferedReader(new FileReader(FILE\_NAME))) {

String line;

while ((line = br.readLine()) != null) {

String[] data = line.split(", ");

if (Integer.parseInt(data[0]) == searchId) {

System.out.println("Student Found: " + line);

return;

}

}

System.out.println("Student with ID " + searchId + " not found.");

} catch (IOException e) {

System.out.println("Error reading file.");

}

}

}

3️⃣ Build & Run the Application

# Compile and package the JAR file

mvn clean package

# Run the CLI application

java -jar target/student-admission-cli-1.0-SNAPSHOT.jar

4️⃣ DevOps Pipeline

FROM openjdk:17

COPY target/student-admission-cli-1.0-SNAPSHOT.jar app.jar

ENTRYPOINT ["java", "-jar", "app.jar"]

docker build -t student-admission .

docker run -it student-admission

Jenkinsfile

pipeline {

agent any

stages {

stage('Build') {

steps {

sh 'mvn clean package'

}

}

stage('Test') {

steps {

sh 'mvn test'

}

}

stage('Deploy') {

steps {

sh 'docker build -t student-admission .'

sh 'docker run -d student-admission'

}

}

}

}

**DevOps Workflow**

1️⃣ **Gradle for Build Automation**  
2️⃣ **Git for Version Control**  
3️⃣ **Jenkins for CI/CD**  
4️⃣ **Docker for Containerization**  
5️⃣ **Kubernetes for Deployment**