Maven Based Java Project for Library Management System

Name: Pawar Rohit S.

Roll No.: 115

1. Introduction

This documentation presents a comprehensive overview of a Java-based Library Management System project developed using Apache Maven. The aim of this project is to simulate a simple library application that includes functionalities such as adding books, issuing and returning books, and displaying book/member 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 library 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 library 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) that contains information about the project and configuration details used by Maven.
• Dependencies: External libraries that the project needs to compile and run.
• Build Lifecycle: Maven phases like validate, compile, test, package, verify, install, and deploy.
• Repositories: Sources where Maven searches for dependencies (local, central, remote).

5. Project Structure

```
LibraryManagementSystem/
├── src/
│   ├── main/java/com/library/          # Java source files
│   └── test/java/com/library/          # Unit tests
├── target/                             # Compiled bytecode and packaged files
├── pom.xml                             # Maven configuration file
```

6. Implementation Details

Modules and Functionalities:

1. Book Module: Add a new book, Display book details.
2. Transaction Module: Issue book, Return book, View transaction history.
3. Admin Module (Optional): Admin login, View all members and books.
4. Database Integration (Optional): Store and retrieve data using MySQL.

Sample Java Class - Book.java:

```
public class Book {
    private String bookId;
    private String title;
    private String author;
    private int quantity;

    public Book(String bookId, String title, String author, int quantity) {
        this.bookId = bookId;
        this.title = title;
        this.author = author;
        this.quantity = quantity;
    }

    public void issueBook() {
        if (quantity > 0) quantity--;
        else System.out.println("Book not available!");
    }

    public void returnBook() {
        quantity++;
    }

    public void displayDetails() {
        System.out.println("Book ID: " + bookId);
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Available Copies: " + quantity);
```

```
    }
}
```

Main Class - Main.java:

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Book book = null;

        while (true) {
            System.out.println("\n--- Library Management Menu ---");
            System.out.println("1. Add Book");
            System.out.println("2. Issue Book");
            System.out.println("3. Return Book");
            System.out.println("4. Display Book Details");
            System.out.println("5. Exit");
            int choice = sc.nextInt();

            switch (choice) {
                case 1:
                    sc.nextLine();
                    System.out.print("Enter Book ID: ");
                    String id = sc.nextLine();
                    System.out.print("Enter Title: ");
                    String title = sc.nextLine();
                    System.out.print("Enter Author: ");
                    String author = sc.nextLine();
                    System.out.print("Enter Quantity: ");
                    int qty = sc.nextInt();
                    book = new Book(id, title, author, qty);
                    break;
                case 2:
                    if (book != null) book.issueBook();
                    else System.out.println("Add book first.");
                    break;
                case 3:
                    if (book != null) book.returnBook();
                    else System.out.println("Add book first.");
                    break;
                case 4:
                    if (book != null) book.displayDetails();
                    else System.out.println("Add book first.");
                    break;
                case 5:
```

```
                System.out.println("Thank you for using Library Management System.");
                System.exit(0);
                break;
            default:
                System.out.println("Invalid choice!");
        }
    }
  }
}
```

7. Steps to Create Maven Project

Using Command Line:

```
mvn archetype:generate -DgroupId=com.library \
                -DartifactId=LibraryManagementSystem \
                -DarchetypeArtifactId=maven-archetype-quickstart \
                -DinteractiveMode=false

cd LibraryManagementSystem
mvn compile
mvn package
```

8. Sample pom.xml File

```
<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.library</groupId>
  <artifactId>LibraryManagementSystem</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 Library Management System provides a practical understanding of project development using Maven. It helps in automating builds, managing dependencies, and organizing the project in a standard way. This assignment enhances Java skills and promotes industry-standard practices 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