**Project Name: Panini Academy**

**Introduction**

Panini Academy is a web-based application that allows students to register for a course, login, and view their personal information. It uses JSP, Hibernate, Servlet, Maven, and MySQL technologies to store and retrieve data from a database. The application has been developed to simplify the process of managing student information and to provide an easy-to-use interface for students.

**Features**

This application includes the following features:

**Registration**: New students can register by providing their personal information and selecting interested course.

**Login**: Registered students can login using email and date of birth to access their personal and academic information.

**Update**: Students can view and update their personal and course information.

**Delete**: Students can delete their data permanently.

**Architecture**

The application is built using a three-tier architecture:

**Presentation Layer:** The presentation layer is implemented using JSP and Html. It handles user interactions and generates dynamic web pages.

**Business Layer:** The business layer is implemented using Servlets and Hibernate. It handles data persistence and retrieval from the database.

**Data Layer**: The data layer is implemented using Hibernate and MySQL DBMS. It stores and retrieves data from the database.

**Technologies Used**

Java 8, JSP, Hibernate 5, Servlet, Maven, MySQL 8, Html, CSS, Bootstrap 4

**IDE**: Eclipse

**Server**: Apache Tomcat 9.0

**System Requirements**

Java 8 or higher

Apache Tomcat 9 or higher

MySQL 8 or higher

**Installation**

Clone the project from GitHub: <https://github.com/code-ravi/PaniniAcademy>

Open the project in your IDE (Eclipse or IntelliJ IDEA)

Configure the MySQL database by updating the hibernate.cfg.xml file with your MySQL database details.

Build the project using Maven: mvn clean install

Deploy the WAR file in Apache Tomcat.

Access the application at <http://localhost:8080/JalaAcademy/layout.html>

**Conclusion**

The Panini Academy web-application is a simple and efficient way to manage student information. It provides an easy-to-use interface for students to view and update their personal and course information. The application is built using JSP, Hibernate, Servlet, Maven, and MySQL technologies, making it a robust and scalable solution.

**Steps to build Full Stack Application**

**Step 1: Set up the Development Environment**

Install Java Development Kit (JDK) 8 or higher.

Install Eclipse IDE for Java EE Developers or any other IDE

Install Apache Tomcat or any other Server.

Install MySQL 8 or higher.

**Step 2: Create a Maven Project**

Maven provides benefits like Dependency management, Build automation, Standardization etc. Alternatively, we can use Gradle.

**Step 3: Add Dependencies**

To import libraries in our project, we need to add dependencies in our pom.xml file

**Step 4: Create the Database**

We have multiple options as database. I have worked on MySQL. In our MySQL client or workbench, we need to create a new schema/database mydatabase and we can create a table as well or hibernate will create a table for us.

**Step 5: Configure Hibernate**

We need to create a new file named hibernate.cfg.xml under src/main/resources and Add the required configurations/properties inside the <hibernate-configuration> tag.

**Step 6: Create the Model/POJO Class**

This class will have all the required attributes for database table.

**Step 7: Create the DAO Interface**

Here we can use JPA Interface.

**Step 8: Create the DAO Implementation**

This class will implement DAO interface which we created in above step.

**Step 9: Create User Service class:**

In Spring Framework, a service class is a class that contains the business logic of an application. It typically acts as an intermediary between the controller and the repository layers, and is responsible for processing data, executing business logic, and coordinating transactions.

**Step 10: Create the Controller**

A controller class is a class that handles HTTP requests and returns an HTTP response to the client. It acts as the intermediary between the user and the system, and manages the flow of information between them. Controller class does URL mapping.

**Step 10: Create the Views**

Here we design presentation layer for our application using technologies like html, css, javascript, React-Js, Angular etc.

These steps summarize the steps to build a full stack application using Spring framework. The steps may vary if we use different technologies and tools.