LEARNER'S ACADEMY

(An online management system)

GITHUB LINK: https://github.com/AnishAugustin09/LEARNER-S-ACADEMY

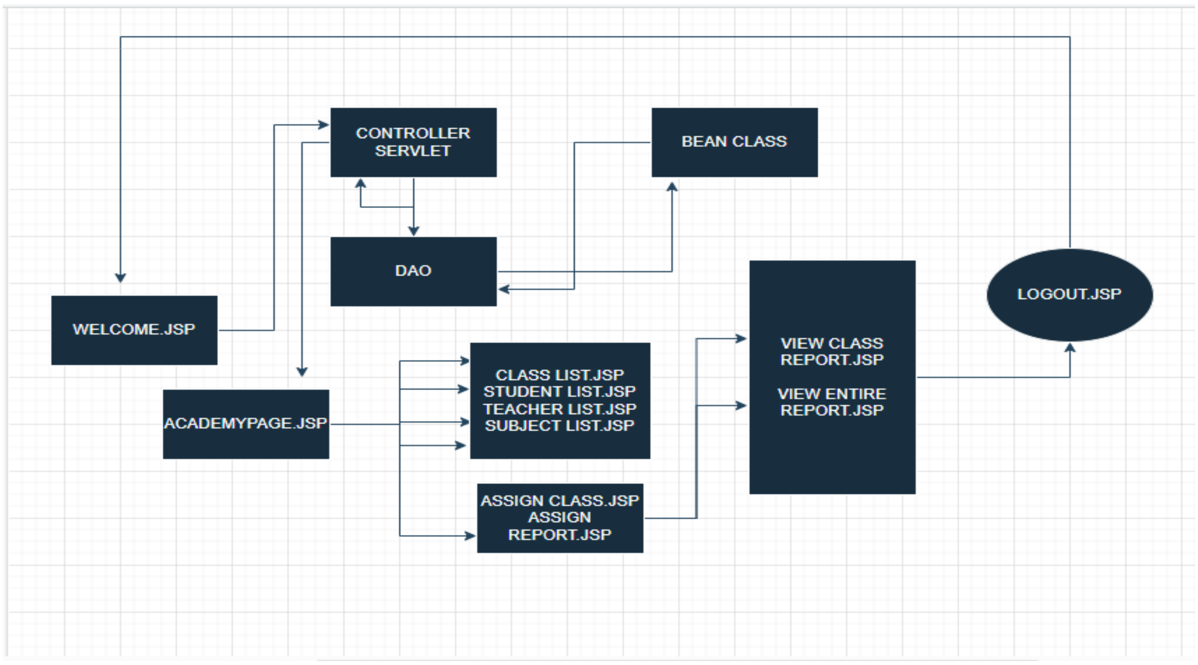
This document contains the:

- Sprint Planning
- Flow of Project
- Concepts used in Project
- Working on Eclipse IDE
- Conclusion

Sprint Planning:

For this project, I've planned to complete the project in 5 Sprints. And the task is completed according to that.

Flow of Project:



Concepts used in Project:

- JSP, HTML & CSS
- JAVA , SERVLET, JSP, JSTL
- MYSQL

Tools & Tech used in Project:

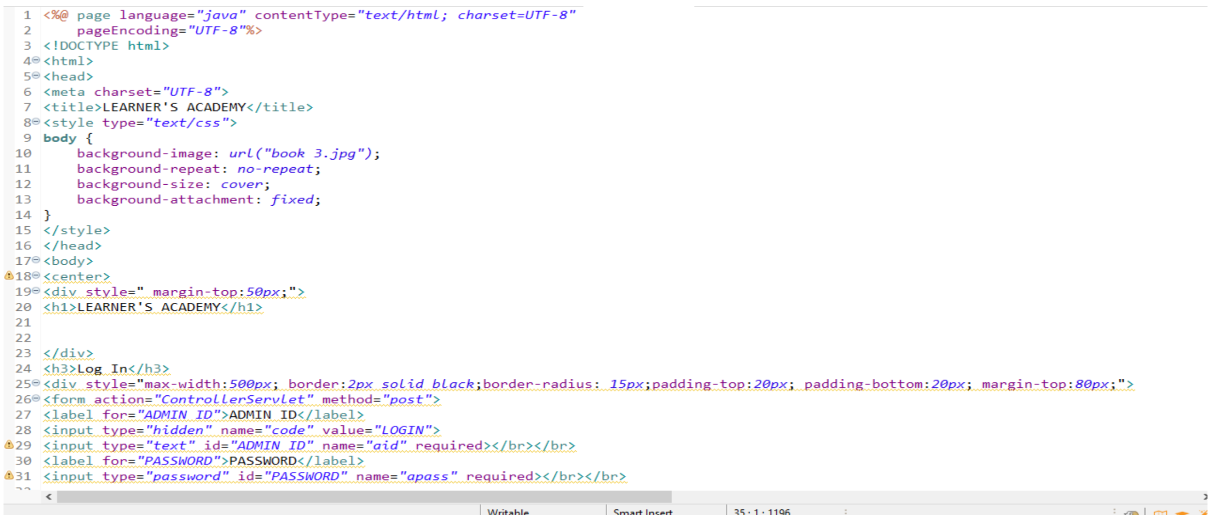
- Eclipse IDE
- Tomcat Server 9.0
- Git & Github

Working on Eclipse IDE:

Step 1: Creating a new Dynamic Web Project in Eclipse

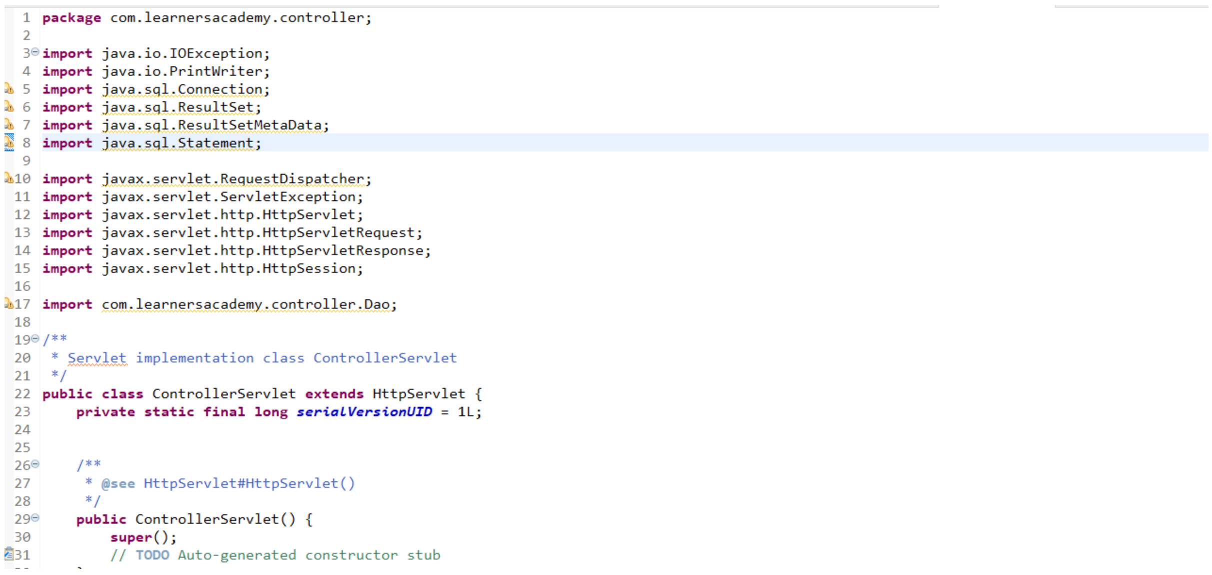
- Open Eclipse
- Go to File -> New -> Dynamic Web Project --> Next.
- Type in any project name and click on “Finish.”
- Select your project and go to WEB-INF -> New -> JSP.
- Enter JSP file name and click on “Finish.”

Step 2: Writing a program in JSP for the entry point of the application (**WELCOME.JSP).**



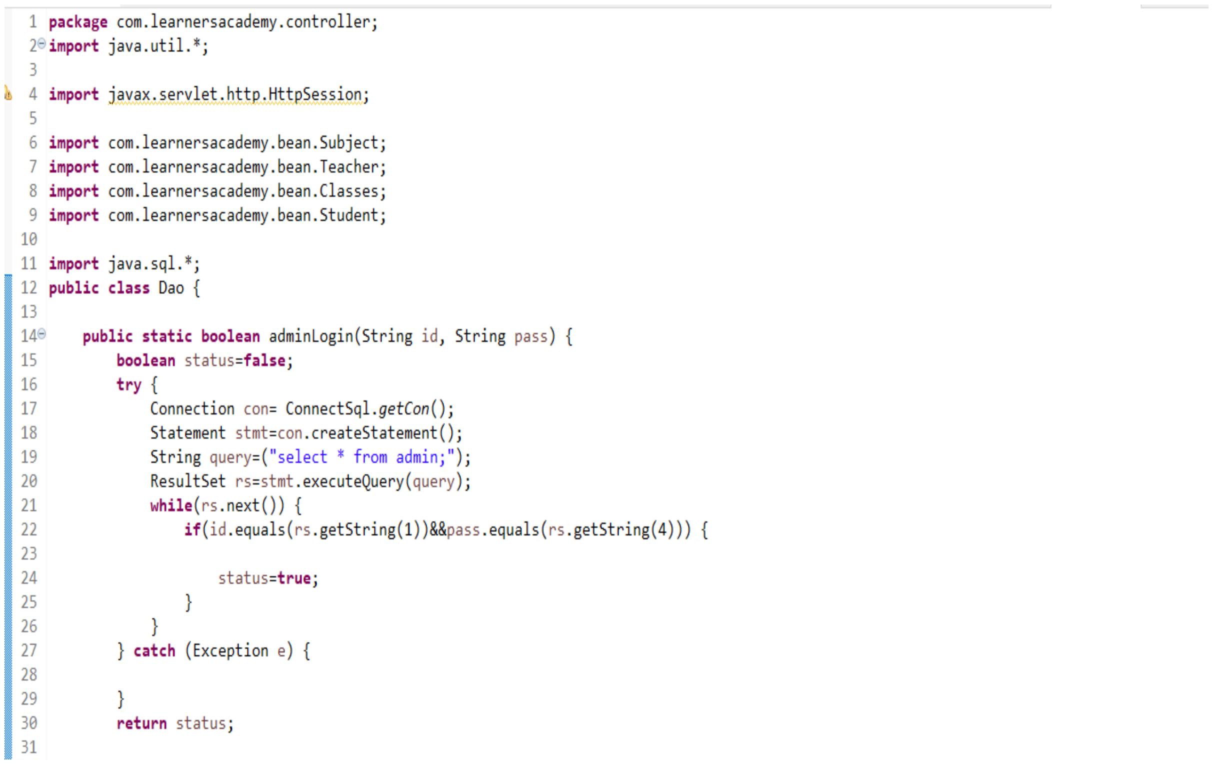
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>LEARNER'S ACADEMY</title>
<style type="text/css">
body {
    background-image: url("book 3.jpg");
    background-repeat: no-repeat;
    background-size: cover;
    background-attachment: fixed;
}
</style>
</head>
<body>
<center>
<div style=" margin-top:50px;">
<h1>LEARNER'S ACADEMY</h1>


</div>
<h3>Log In</h3>
<div style="max-width:500px; border:2px solid black;border-radius: 15px;padding-top:20px; padding-bottom:20px; margin-top:80px;">
<form action="ControllerServlet" method="post">
<label for="ADMIN ID">ADMIN ID</label>
<input type="hidden" name="code" value="LOGIN">
<input type="text" id="ADMIN ID" name="aid" required></br></br>
<label for="PASSWORD">PASSWORD</label>
<input type="password" id="PASSWORD" name="apass" required></br></br>
```

Step 3: Create a servlet page which get request and send response (**CONTROLLERSERVLET.Java**).



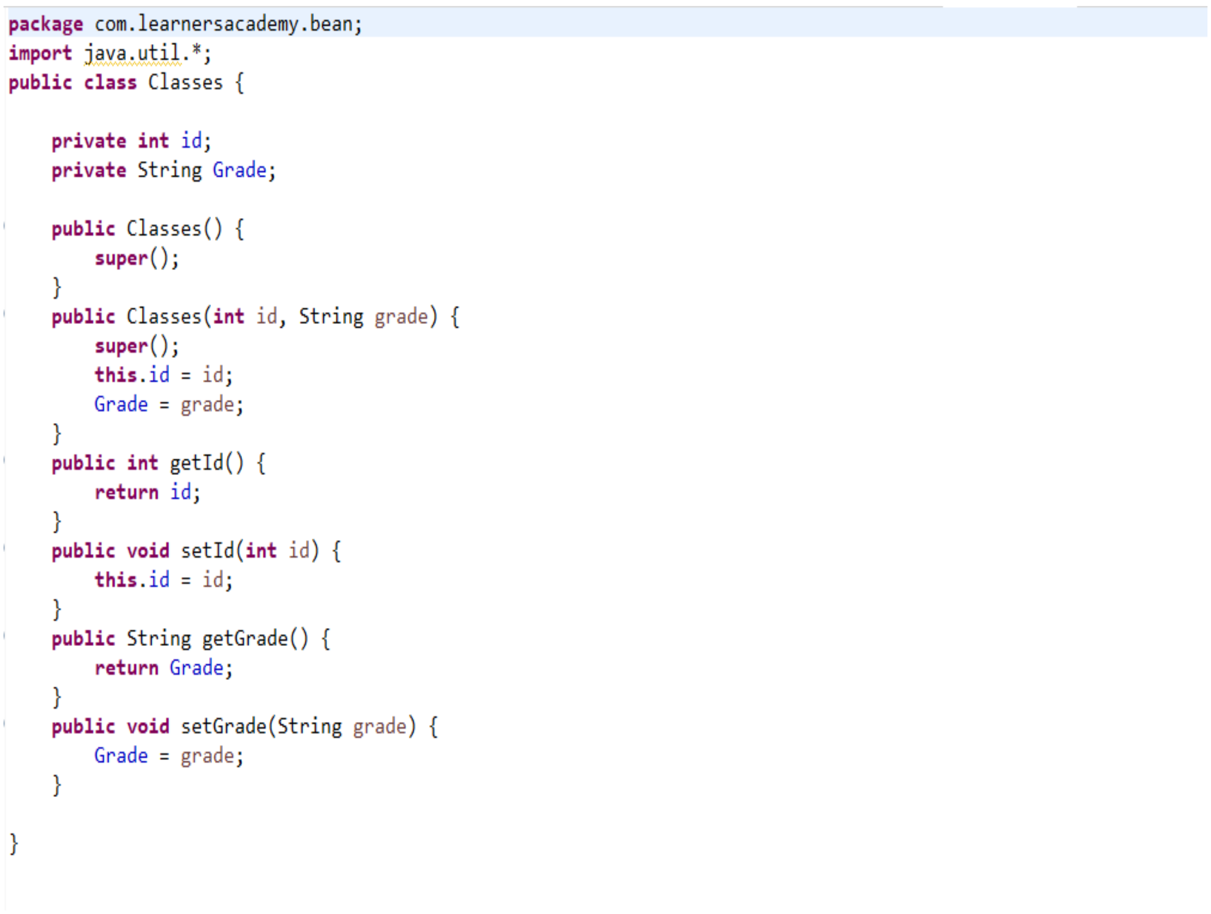
```
package com.learnersacademy.controller;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import com.learnersacademy.controller.Dao;

/**
 * Servlet implementation class ControllerServlet
 */
public class ControllerServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;


    /**
     * @see HttpServlet#HttpServlet()
     */
    public ControllerServlet() {
        super();
        // TODO Auto-generated constructor stub
```

Step 4: Create a JAVA class (**DAO.JAVA**) to connect mysql to jdbc.

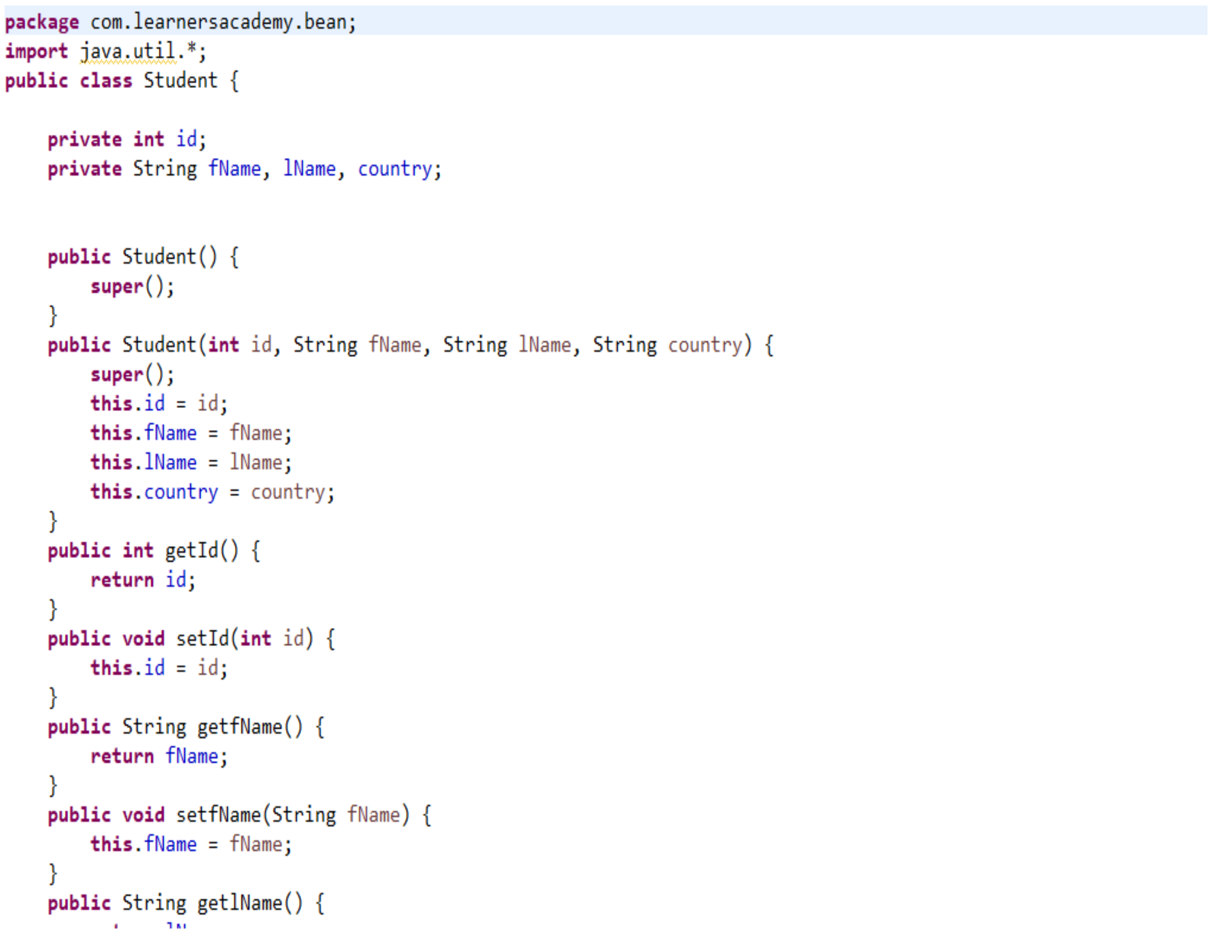


```
package com.learnersacademy.controller;
import java.util.*;

import javax.servlet.http.HttpSession;

import com.learnersacademy.bean.Subject;
import com.learnersacademy.bean.Teacher;
import com.learnersacademy.bean.Classes;
import com.learnersacademy.bean.Student;

import java.sql.*;
public class Dao {

    public static boolean adminLogin(String id, String pass) {
        boolean status=false;
        try {
            Connection con= ConnectSql.getCon();
            Statement stmt=con.createStatement();
            String query=("select * from admin;");
            ResultSet rs=stmt.executeQuery(query);
            while(rs.next()) {
                if(id.equals(rs.getString(1))&&pass.equals(rs.getString(4))) {

                    status=true;
                }
            }
        } catch (Exception e) {

        }
        return status;

```

Step 5: Create JAVA POJO Class to access the data from Mysql.



```java
package com.learnersacademy.bean;
import java.util.*;
public class Classes {

    private int id;
    private String Grade;

    public Classes() {
        super();
    }
    public Classes(int id, String grade) {
        super();
        this.id = id;
        Grade = grade;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getGrade() {
        return Grade;
    }
    public void setGrade(String grade) {
        Grade = grade;
    }

}
```



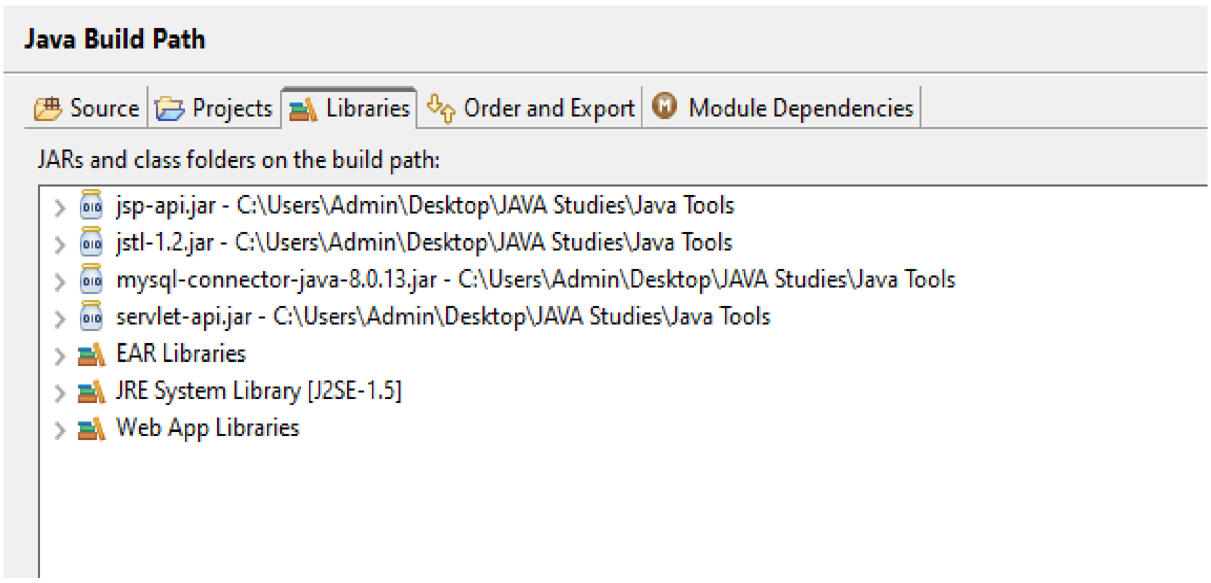
```java
package com.learnersacademy.bean;
import java.util.*;
public class Student {

    private int id;
    private String fName, lName, country;


    public Student() {
        super();
    }
    public Student(int id, String fName, String lName, String country) {
        super();
        this.id = id;
        this.fName = fName;
        this.lName = lName;
        this.country = country;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getfName() {
        return fName;
    }
    public void setfName(String fName) {
        this.fName = fName;
    }
    public String getlName() {
```

Step 6: Using the POJO class, the DAO can access the MYSQL database.

Step 7: Create table for the required fields

- Class
- Subject
- Student
- Teacher
- Admin
- Assign Class
- Assign Report

Step 8: Add the require JAR files in the **CONFIGURATION PATH & WEB-INF/LIB.**



JARs and class folders on the build path:

```
jsp-api.jar - C:\Users\Admin\Desktop\JAVA Studies\Java Tools
jstl-1.2.jar - C:\Users\Admin\Desktop\JAVA Studies\Java Tools
mysql-connector-java-8.0.13.jar - C:\Users\Admin\Desktop\JAVA Studies\Java Tools
servlet-api.jar - C:\Users\Admin\Desktop\JAVA Studies\Java Tools
EAR Libraries
JRE System Library [J2SE-1.5]
Web App Libraries
```

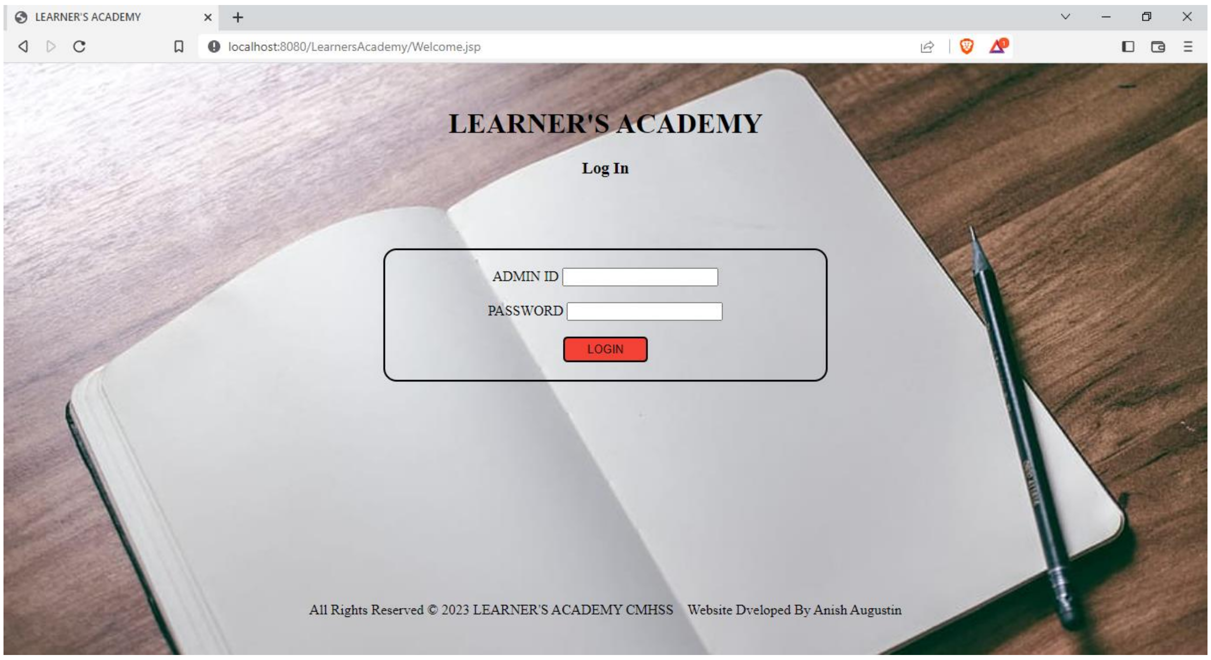
Step 9: Using these tables we can access the data’s from the Main Page(**ACADEMYPAGE.JSP**)

Step 10: By Creating all the requirements, we can achieve the entire goal of this Project.

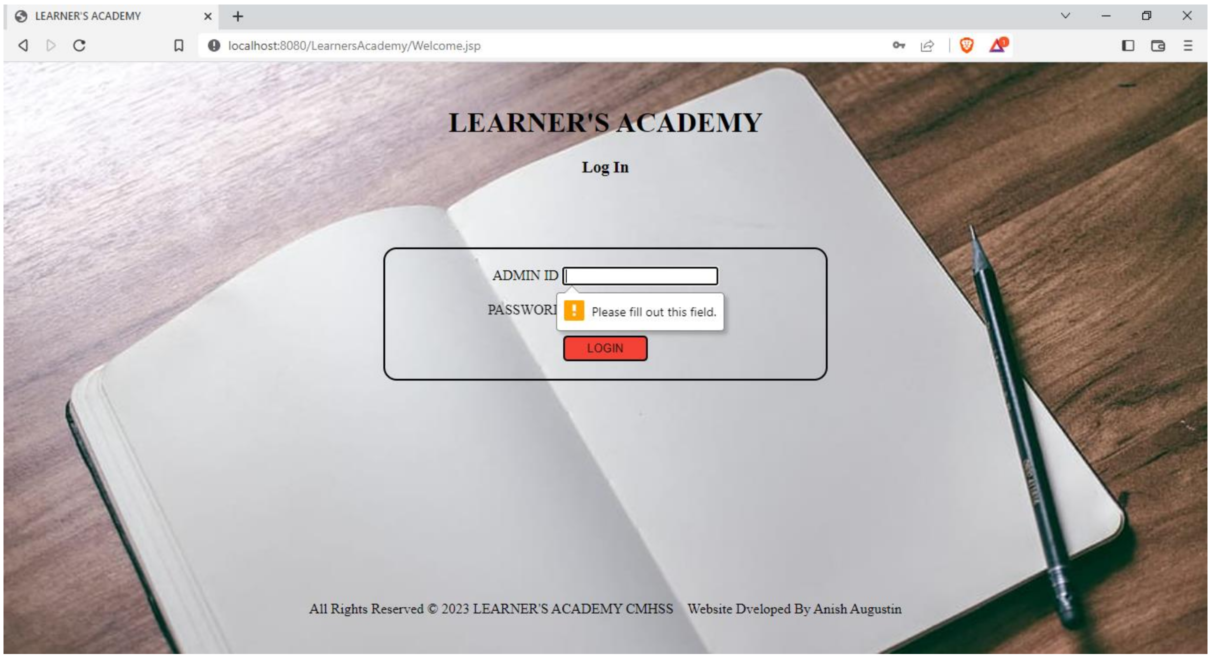
This Project met all the requirements:

1. It's a back-office application with a single administrator login.

Please use Admin ID: **LA1** & PASSWORD: **121212**

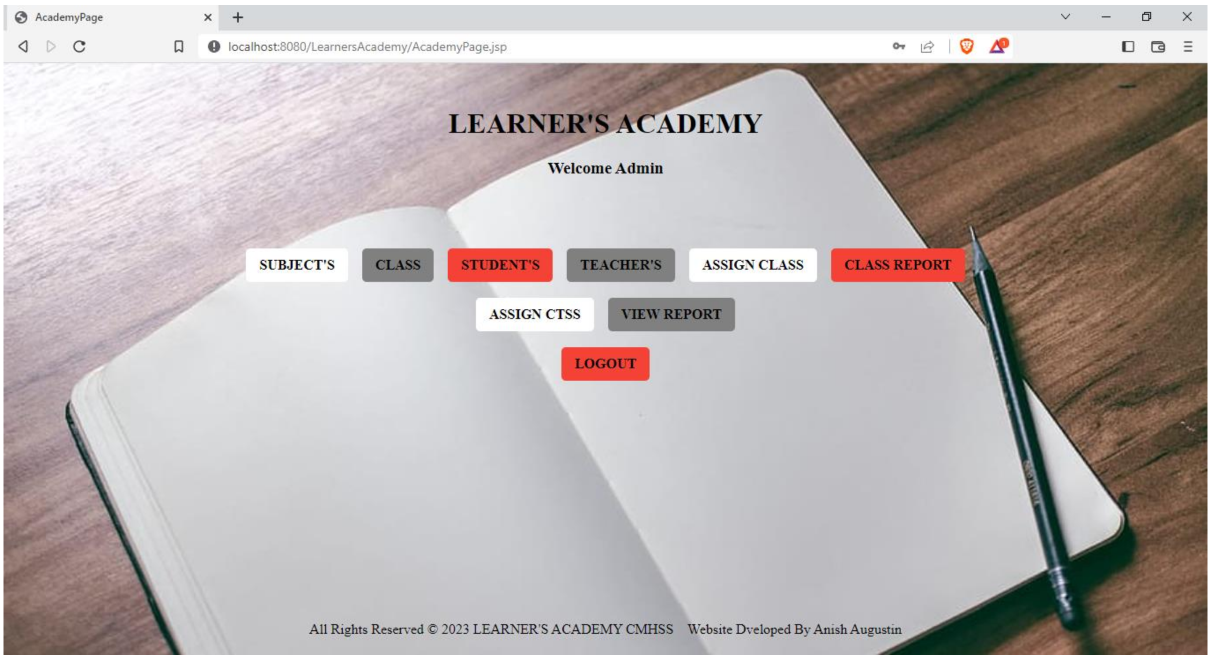

LEARNER'S ACADEMY
localhost:8080/LearnersAcademy/Welcome.jsp
LEARNER'S ACADEMY
Log In
ADMIN ID
PASSWORD
LOGIN
All Rights Reserved © 2023 LEARNER'S ACADEMY CMHSS Website Dveloped By Anish Augustin

Login Page


LEARNER'S ACADEMY
localhost:8080/LearnersAcademy/Welcome.jsp
LEARNER'S ACADEMY
Log In
ADMIN ID
PASSWORD
Please fill out this field.
LOGIN
All Rights Reserved © 2023 LEARNER'S ACADEMY CMHSS Website Dveloped By Anish Augustin

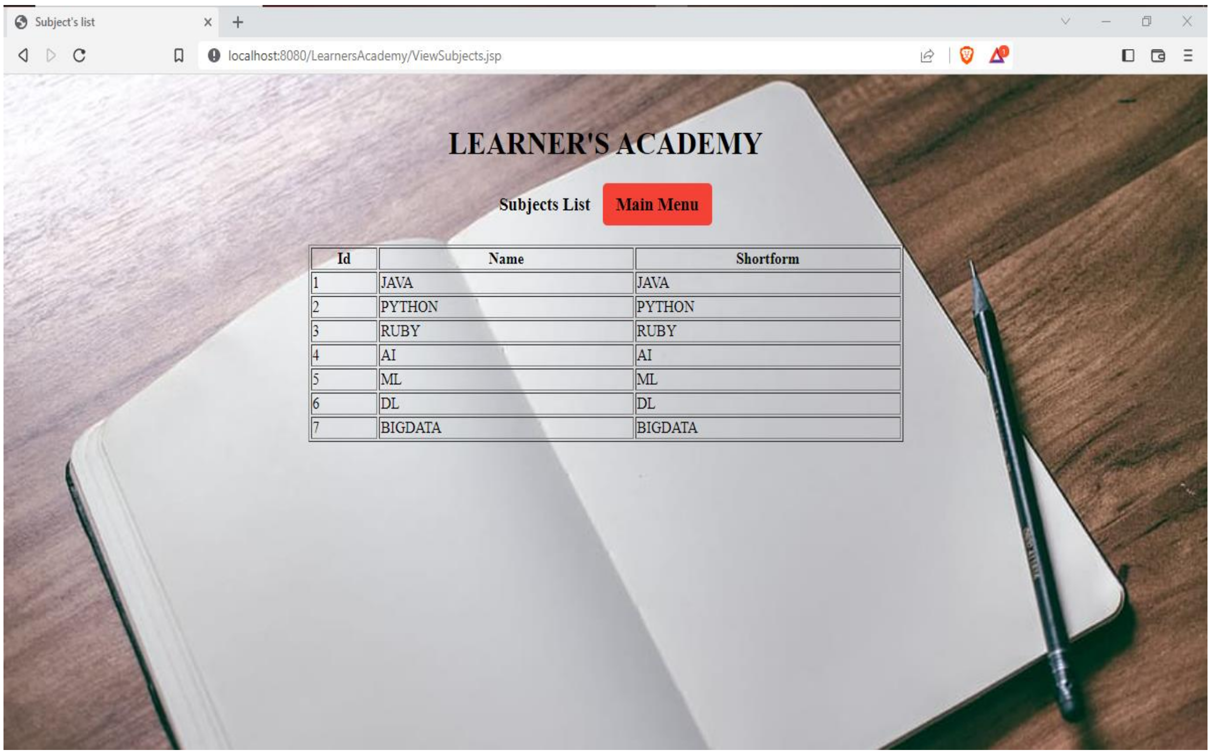
Login page will not move to next page without ID & PASS.

2.The system keeps track of its classes, subjects, students, and teachers..


AcademyPage
localhost:8080/LearnersAcademy/AcademyPage.jsp
LEARNER'S ACADEMY
Welcome Admin
SUBJECT'S
CLASS
STUDENT'S
TEACHER'S
ASSIGN CLASS
CLASS REPORT
ASSIGN CTSS
VIEW REPORT
LOGOUT
All Rights Reserved © 2023 LEARNER'S ACADEMY CMHSS Website Dveloped By Anish Augustin

ACADEMY PAGE

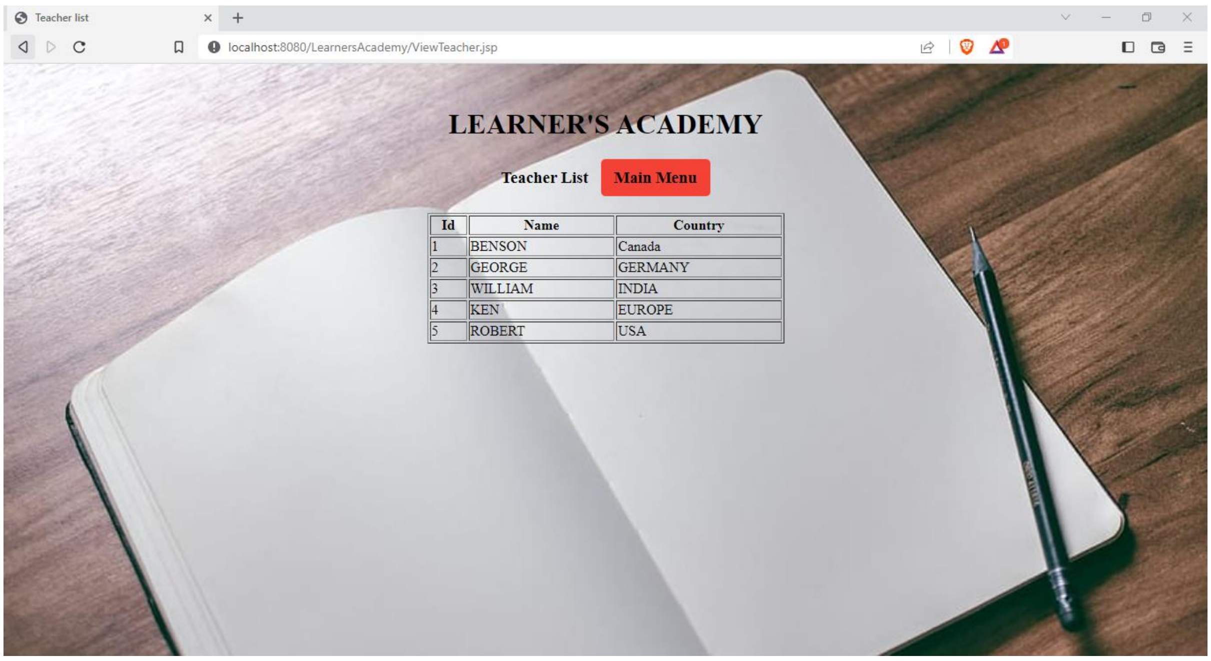
3. Set up a master list of all the subjects for all the classes



Id	Name	Shortform
1	JAVA	JAVA
2	PYTHON	PYTHON
3	RUBY	RUBY
4	AI	AI
5	ML	ML
6	DL	DL
7	BIGDATA	BIGDATA

MASTER SUBJECT LIST

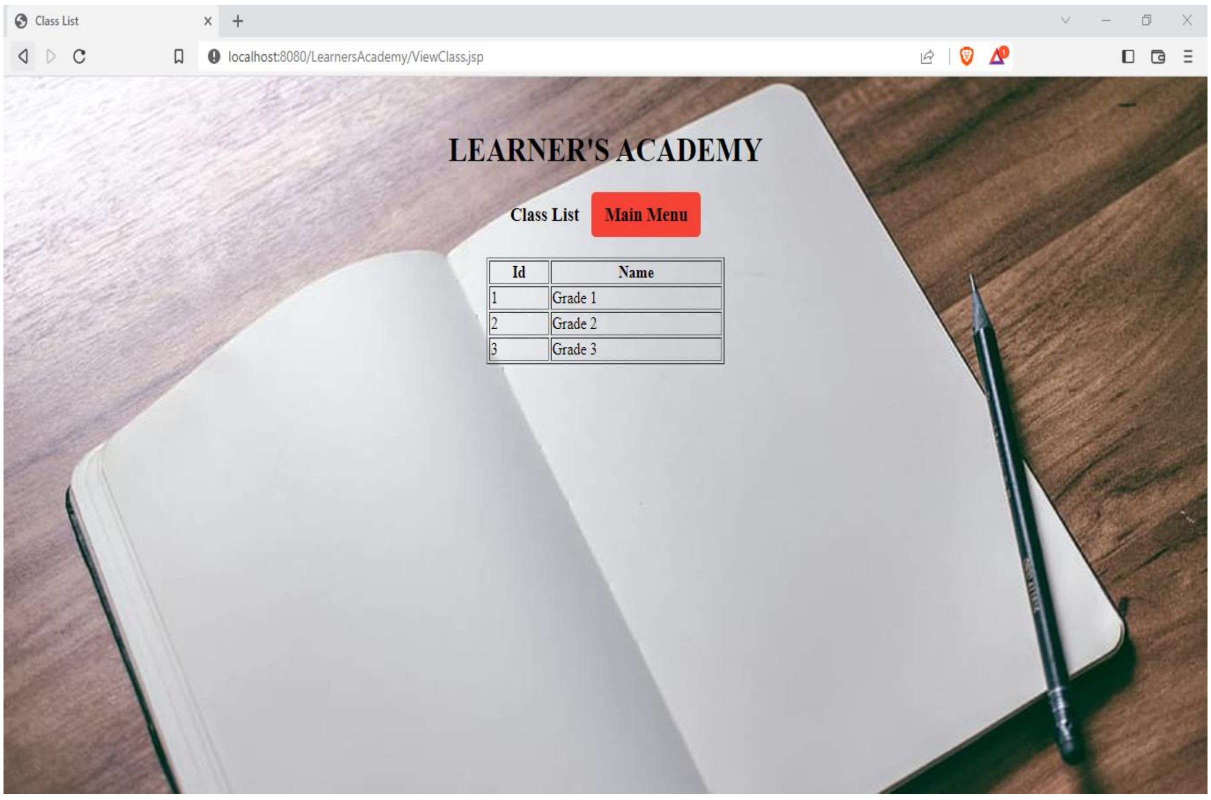
4. Set up a master list of all the teachers



Id	Name	Country
1	BENSON	Canada
2	GEORGE	GERMANY
3	WILLIAM	INDIA
4	KEN	EUROPE
5	ROBERT	USA

MASTER TEACHER'S LIST

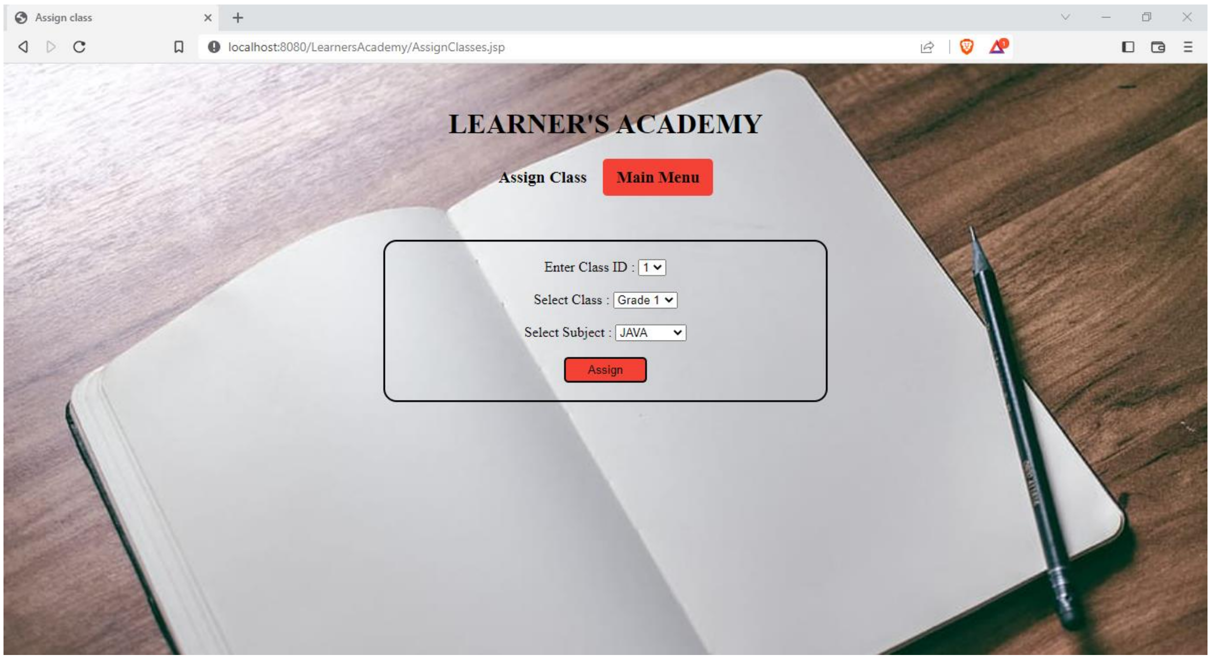
5. Set up a master list of all the classes



Id	Name
1	Grade 1
2	Grade 2
3	Grade 3

MASTER CLASS LIST

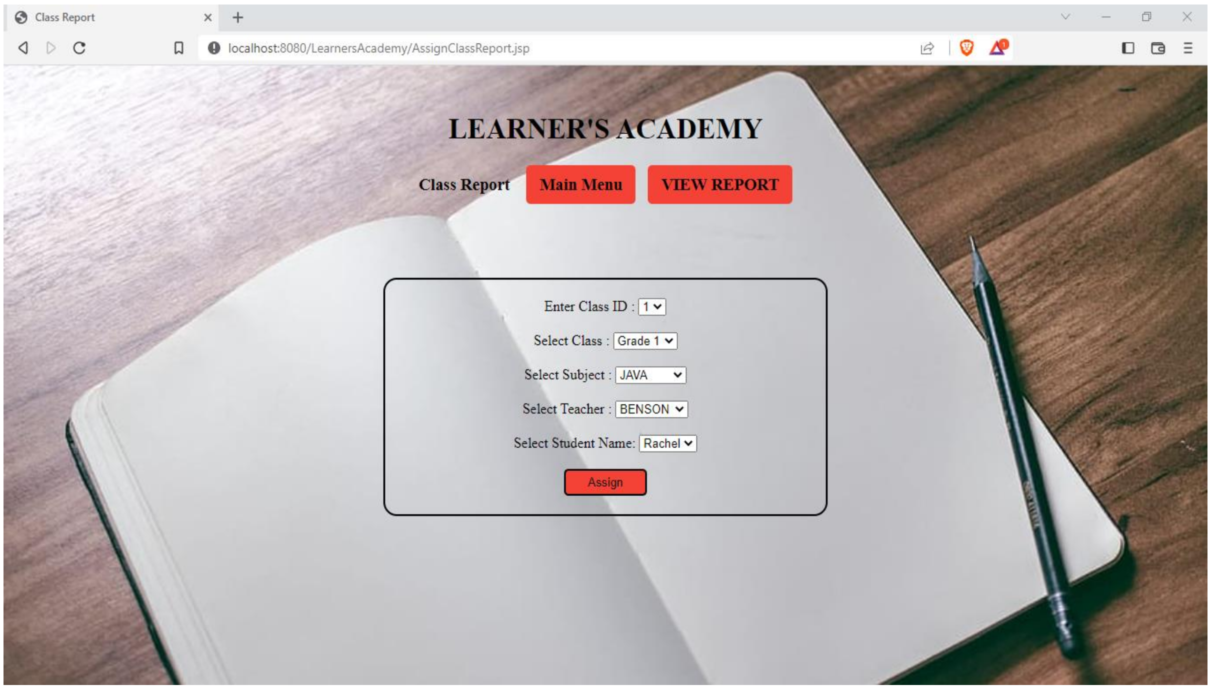
6. Assign classes for subjects from the master list



Enter Class ID : 1
Select Class : Grade 1
Select Subject : JAVA

ASSIGNING CLASS

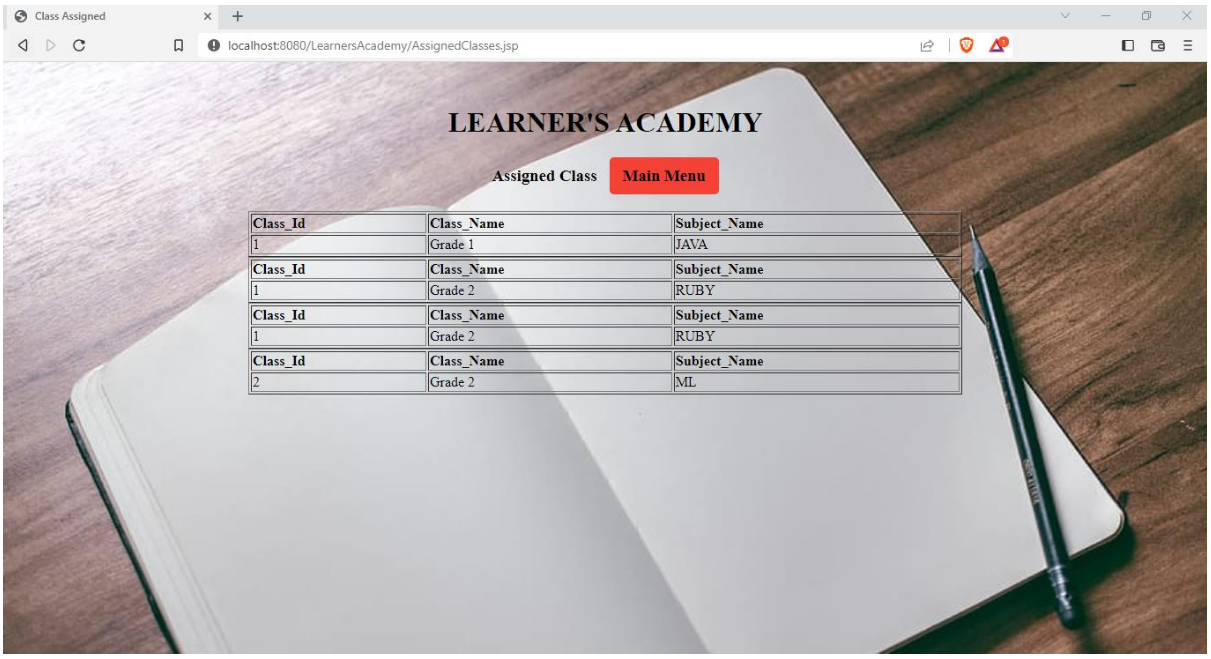
7. Assign teachers to a class for a subject (A teacher can be assigned to different classes for different subjects)



Enter Class ID : 1
Select Class : Grade 1
Select Subject : JAVA
Select Teacher : BENSON
Select Student Name: Rachel

ASSIGNING REPORT

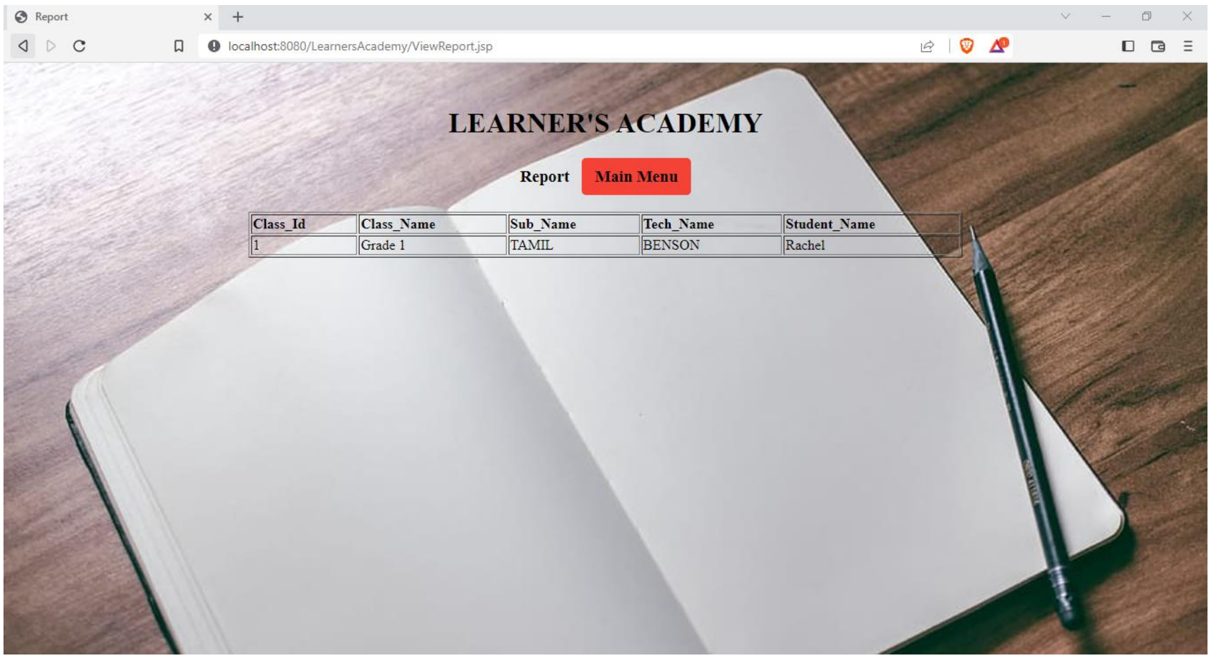
8. Get a master list of students (Each student must be assigned to a single class)



Class_Id	Class_Name	Subject_Name
1	Grade 1	JAVA
Class_Id	Class_Name	Subject_Name
1	Grade 2	RUBY
Class_Id	Class_Name	Subject_Name
1	Grade 2	RUBY
Class_Id	Class_Name	Subject_Name
2	Grade 2	ML

MASTERLIST CLASSES WITH SUBJECT

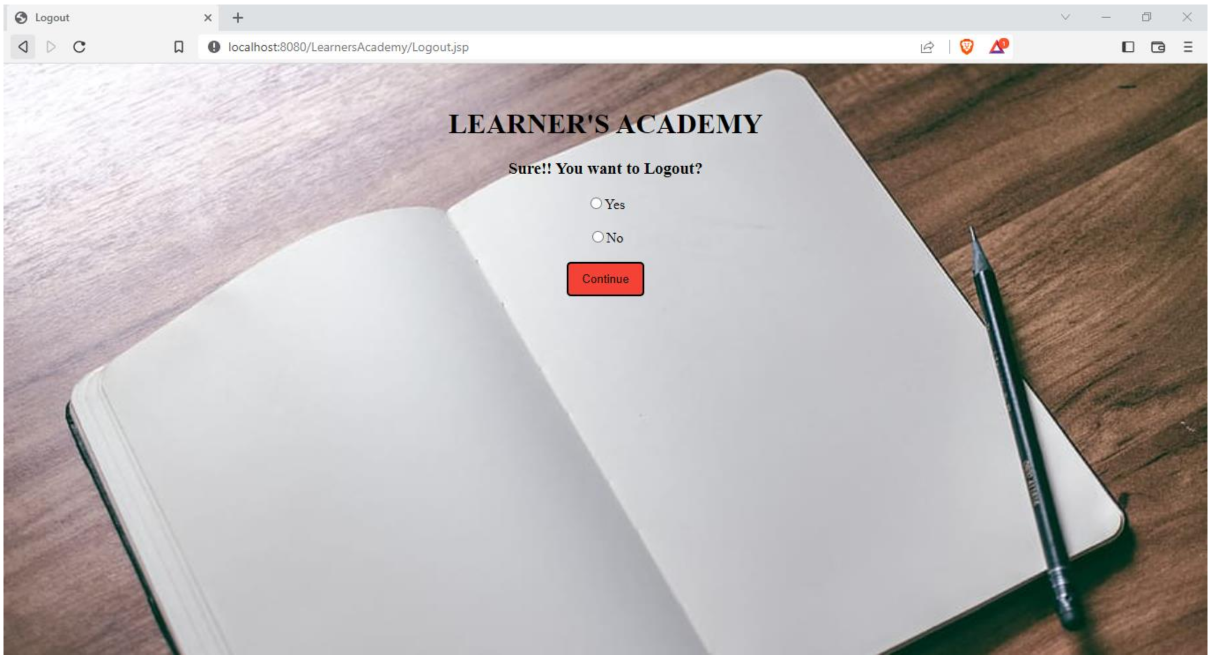
9. There will be an option to view a Class Report which will show all the information about the class, such as the list of students, subjects, and teachers



Class_Id	Class_Name	Sub_Name	Tech_Name	Student_Name
1	Grade 1	TAMIL	BENSON	Rachel

MASTERLIST OF REPORT

10. Additional settings (**LOGOUT**)



LEARNER'S ACADEMY

Sure!! You want to Logout?

○ Yes

○ No

LOGOUT PAGE

The goal of the company is to deliver a high-end quality product as early as possible. So, In this Project you can perform

- Admin login
- View Class list
- View Students list
- View Subjects list
- View Teachers list
- Assign Classes with Subjects
- Assign Classes with Subjects, Students & Teachers.

FINAL SETTING : Upload files to GitHub Repository

- Open your command prompt and navigate to the folder where you have created your files.
- cd <folder path>
- Initialize repository using the following command:
- git init
- Add all the files to your git repository using the following command:

- git add .
- Commit the changes using the following command:
- git commit . -m <commit message>
- Push the files to the folder you initially created using the following command:
- git push -u origin master

Unique Selling Points:

1. This application is use to manage & The system keeps track of its classes, subjects, students, and teachers.
2. This application will keep on running until user close the program.
3. It allows the admin to Assign classes for subjects from the master list, Assign teachers to a class for a subject (A teacher can be assigned to different classes for different subjects)
4. It allows the admin to view class, students, subjects & teachers list.

Conclusion:

Further enhancements to the application can be made which may include

Admin should able to add many data’s in the DB

THANK YOU