Learner's Academy

All Files are listed down as per the order

1. POM.xml :- (LearnersAcademy/pom.xml)

```
<?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.academy</groupId>
  <artifactId>learners-academy</artifactId>
  <version>1.1</version>

  <name>Learner's Academy</name>

  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <packaging>war</packaging>

  <dependencies>

    <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>4.0.1</version>
      <scope>provided</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>jstl</artifactId>
```

```
      <version>1.2</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-api</artifactId>
      <version>5.7.0</version>
      <scope>test</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <version>1.18.16</version>
      <scope>provided</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <version>42.2.12</version>
    </dependency>


  </dependencies>

  <build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
      <plugin>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.2.3</version>
        <configuration>
```

```
            <warSourceDirectory>WebContent</warSourceDirectory>
          </configuration>
        </plugin>
      </plugins>
    </build>

</project>
```

2. web.xml :- (LearnersAcademy/WebContent/WEB-INF/web.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://xmlns.jcp.org/xml/ns/javaee"
      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
      http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
      id="WebApp_ID" version="4.0">

  <display-name>LearnersAcademy</display-name>

  <welcome-file-list>
    <welcome-file>home</welcome-file>
  </welcome-file-list>


  <error-page>
    <!-- Missing login -->
    <error-code>401</error-code>
    <location>/error</location>
  </error-page>
  <error-page>
    <!-- Forbidden directory listing -->
    <error-code>403</error-code>
    <location>/error</location>
  </error-page>
  <error-page>
    <!-- Missing resource -->
    <error-code>404</error-code>
    <location>/error</location>
  </error-page>
  <error-page>
    <!-- Uncaught exception -->
    <error-code>500</error-code>
    <location>/error</location>
  </error-page>
```

```
    <error-page>
      <!-- Unsupported servlet method -->
      <error-code>503</error-code>
      <location>/error</location>
    </error-page>

</web-app>
```

3. Config.properties :- (LearnersAcademy/WebContent/WEB-INF/classes/config.properties)

```
db.password=root
db.user=postgres
db.url=jdbc:postgresql://localhost/learners_academy
db.driver=org.postgresql.Driver
```

4. ClassController.java :- (LearnersAcademy/src/main/java/com/academy/controller/ClassController.java)

```
package main.java.com.academy.controller;


import main.java.com.academy.dao.ClassDAO;

import main.java.com.academy.entity.Classes;


import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.util.List;


/**

 * Servlet implementation class ClassController

 * This class handles get request of url pattern '/classes'.

 */
```

```
@WebServlet(name = "classes", urlPatterns = {"/classes"})
public class ClassController extends HttpServlet {
   private static final long serialVersionUID = 1L;

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

   /**
    * This method takes get request of url pattern '/classes' and fetch the list of all class records
    * from {@link ClassDAO} and forward the request to classes.jsp file.
    *
    * @param request  - {@link HttpServletRequest}
    * @param response - {@link HttpServletResponse}
    * @throws ServletException, IOException
    * @jsp /WebContent/classes.jsp
    * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
    * response)
    */
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      // TODO Auto-generated method stub

      response.setHeader("Cache-Control", "no-cache, no-store, must-revalidate"); // HTTP 1.1.
      response.setHeader("Pragma", "no-cache"); // HTTP 1.0.
```

```
        response.setDateHeader("Expires", 0);

        List<Classes> classes = ClassDAO.getAllClassesWithStrength();

        request.setAttribute("classes", classes);

        request.getRequestDispatcher("classes.jsp")
                .include(request, response);
    }
}
```

5. ErrorController.java :- (LearnersAcademy/src/main/java/com/academy/controller/ErrorController.java)

```
package main.java.com.academy.controller;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * Servlet implementation class ErrorController This class handles get request
 * of url pattern '/error'.
 *
*/
@WebServlet(name = "error", urlPatterns = {"/error"})
public class ErrorController extends HttpServlet {
```

```
private static final long serialVersionUID = 1L;

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

/**
 * This method takes get request of url pattern '/error' and forward the request
 * to error.jsp file.
 *
 * @param request  - {@link HttpServletRequest}
 * @param response - {@link HttpServletResponse}
 * @throws ServletException, IOException
 * @jsp /WebContent/error.jsp
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
 * response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
   // TODO Auto-generated method stub
   processError(request, response);
}

/**
 * This method takes post request of url pattern '/error' and forward the
```

```
 * request to error.jsp file.
 *
 * @param request  - {@link HttpServletRequest}
 * @param response - {@link HttpServletResponse}
 * @throws ServletException, IOException
 * @jsp /WebContent/error.jsp
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
 * response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
      throws IOException, ServletException {
   processError(request, response);
}

/**
 * This method is internal implementation for doGet() and doPost() methods.
 *
 * @param request  - {@link HttpServletRequest}
 * @param response - {@link HttpServletResponse}
 * @throws ServletException, IOException
 * @jsp /WebContent/error.jsp
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
 * response)
 */
private void processError(HttpServletRequest request, HttpServletResponse response)
      throws IOException, ServletException {

   response.setHeader("Cache-Control", "no-cache, no-store, must-revalidate"); // HTTP 1.1.
   response.setHeader("Pragma", "no-cache"); // HTTP 1.0.
```

```
response.setDateHeader("Expires", 0);

String error = null;

String path = request.getContextPath();

Integer statusCode = (Integer) request.getAttribute("javax.servlet.error.status_code");

switch (statusCode) {

   case 500:
      error = "Internal Server Error";
      break;
   case 403:
      error = "Request Forbidden";
      break;
                        case 400:
      error = "Bad Request";
      break;
   case 401:
      error = "Unauthorized";
      break;
   default:
      error = "Page Not Found";
}

request.setAttribute("statusCode", statusCode);
request.setAttribute("error", error);
request.setAttribute("path", path);
```

```
            request.getRequestDispatcher("error.jsp").forward(request, response);
        }
    }
```

6. HomeController.java :- (LearnersAcademy/src/main/java/com/academy/controller/HomeController.java)

```
package main.java.com.academy.controller;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * Servlet implementation class HomeController
 * This class handles get request of url pattern '/home'.
 *

 */
@WebServlet(name = "home", urlPatterns = {"/home"})
public class HomeController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public HomeController() {
```

```
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * This method takes get request of url pattern '/home' and include the index.jsp file in response.
     *
     * @param request  - {@link HttpServletRequest}
     * @param response - {@link HttpServletResponse}
     * @throws ServletException, IOException
     * @jsp /WebContent/index.jsp
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
     * response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // TODO Auto-generated method stub

        request.getRequestDispatcher("index.jsp")
                .include(request, response);
    }

}
```

7. LoginController.java :- (LearnersAcademy/src/main/java/com/academy/controller/LoginController.java)

```
package main.java.com.academy.controller;

import main.java.com.academy.dao.Authenticate;
```

```
import main.java.com.academy.entity.Users;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

/**
 * Servlet implementation class LoginController
 * This class handles get and post request of url pattern '/login'.
 *
 * @author Riyaz J
 * @version 1.1
 */
@WebServlet(name = "login", urlPatterns = {"/login"})
public class LoginController extends HttpServlet {
   private static final long serialVersionUID = 1L;

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

```
  /**
   * This method takes get request of url pattern '/login' and forward the request to login.html file.
   *
   * @param request  - {@link HttpServletRequest}
   * @param response - {@link HttpServletResponse}
   * @throws ServletException, IOException
   * @jsp /WebContent/login.html
   * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
   * response)
   */
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    // TODO Auto-generated method stub
    request.getRequestDispatcher("login.html").forward(request, response);
  }

  /**
   * This method takes post request of url pattern '/login'.
   * This method takes 'username' and 'password' parameters and checks if user is present or not.
   * If user is present, the request will be redirected to {@link HomeController} with login success message.
   * If user is not present, the request will be redirected to {@link LoginController} with login failure message.
   *
   * @param request  - {@link HttpServletRequest}
   * @param response - {@link HttpServletResponse}
   * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
   * response)
   */
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
```

```
        throws IOException {
    // TODO Auto-generated method stub

    String user_name = request.getParameter("username");
    String password = request.getParameter("password");

    Users user = null;

    if (user_name.matches("^(.+)@(.+)$"))
        user = Authenticate.getUserWithEmail(user_name, password);
    else
        user = Authenticate.getUser(user_name, password);

    if (user != null) {
        HttpSession session = request.getSession();
        session.setAttribute("user", user.getEmail());

        response.sendRedirect("home?login-successful");
    } else
        response.sendRedirect("login?login-failed");
  }

}
```

8. LogoutController.java :- (LearnersAcademy/src/main/java/com/academy/controller/LogoutController.java)

```
package main.java.com.academy.controller;

import javax.servlet.annotation.WebServlet;
```

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

/**
 * Servlet implementation class LogoutController
 * This class handles get request of url pattern '/logout'.
 *

 */
@WebServlet(name = "logout", urlPatterns = {"/logout"})
public class LogoutController extends HttpServlet {
   private static final long serialVersionUID = 1L;

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

   /**
    * This method takes get request of url pattern '/logout' and removes the user from session object.
    * After removing the user from session object, the request will be redirected to {@link HomeController} with
    * logout success message.
```

```
     *
     * @param request  - {@link HttpServletRequest}
     * @param response - {@link HttpServletResponse}
     * @throws IOException
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
     * response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws IOException {
        // TODO Auto-generated method stub

        HttpSession session = request.getSession(false);
        session.setAttribute("user", "");
        session.removeAttribute("user");
        session.invalidate();

        response.sendRedirect("home?logout-successful");
    }
}
```

9. ReportController.java :- (LearnersAcademy/src/main/java/com/academy/controller/ReportController.java)

```
package main.java.com.academy.controller;

import main.java.com.academy.dao.ClassDAO;
import main.java.com.academy.entity.Classes;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
```

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * Servlet implementation class ReportController
 * This class handles get request of url pattern '/report'.
 *
 * @author Riyaz J
 * @version 1.1
 */
@WebServlet(name = "report", urlPatterns = {"/report"})
public class ReportController extends HttpServlet {
   private static final long serialVersionUID = 1L;

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

   /**
    * This method takes get request of url pattern '/request' and classId parameter and fetch the particular class
    * records with all the Subjects, Students and, Teachers from {@link ClassDAO} and forward the request to
    * report.jsp file.
```

```
 *
 * @param request  - {@link HttpServletRequest}
 * @param response - {@link HttpServletResponse}
 * @throws ServletException, IOException
 * @jsp /WebContent/report.jsp
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
 * response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
   // TODO Auto-generated method stub

   response.setHeader("Cache-Control", "no-cache, no-store, must-revalidate"); // HTTP 1.1.
   response.setHeader("Pragma", "no-cache"); // HTTP 1.0.
   response.setDateHeader("Expires", 0);

   int classId = 1;

   if (request.getParameter("classId") != null)
      classId = Integer.parseInt(request.getParameter("classId"));

   Classes cls = ClassDAO.getClassWithSubjectsTeachersStudents(classId);

   request.setAttribute("pages", ClassDAO.countOfClasses());

   request.setAttribute("cls", cls);

   request.setAttribute("currentPage", classId);
```

```
        request.getRequestDispatcher("report.jsp")
                .forward(request, response);
    }

}
```

10. StudentController.java :- (LearnersAcademy/src/main/java/com/academy/controller/StudentController.java)

```
package main.java.com.academy.controller;

import main.java.com.academy.dao.StudentDAO;
import main.java.com.academy.entity.Students;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.List;

/**
 * Servlet implementation class StudentController
 * This class handles get request of url pattern '/students'.
 *

 */
@WebServlet(name = "students", urlPatterns = {"/students"})
public class StudentController extends HttpServlet {
```

```
private static final long serialVersionUID = 1L;

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

/**
 * This method takes get request of url pattern '/students' and fetch the list of all Student records
 * from {@link StudentDAO} and forward the request to students.jsp file.
 *
 * @param request  - {@link HttpServletRequest}
 * @param response - {@link HttpServletResponse}
 * @throws ServletException, IOException
 * @jsp /WebContent/students.jsp
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
 * response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    // TODO Auto-generated method stub

    response.setHeader("Cache-Control", "no-cache, no-store, must-revalidate"); // HTTP 1.1.
    response.setHeader("Pragma", "no-cache"); // HTTP 1.0.
    response.setDateHeader("Expires", 0);
```

```
int from = 0;
int to = 10;

int currentPage = 1;

if (request.getParameter("page") != null)
   currentPage = Integer.parseInt(request.getParameter("page"));

if (currentPage > 1) {

   to *= currentPage;
   from = to - 10;
}

boolean showClass = true;

List<Students> students = StudentDAO.getFewStudentsWithOffset(30, from, showClass);

int count = StudentDAO.countOfStudents();

int pages = (int) Math.round(((double) count) / 30);

request.setAttribute("students", students);

request.setAttribute("showClass", showClass);

request.setAttribute("pages", pages);
request.setAttribute("currentPage", currentPage);
```

```
        request.getRequestDispatcher("students.jsp")
                .forward(request, response);
    }

}
```

11. SubjectController.java :- (LearnersAcademy/src/main/java/com/academy/controller/SubjectController.java)

```
package main.java.com.academy.controller;

import main.java.com.academy.dao.SubjectDAO;
import main.java.com.academy.entity.Subjects;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.List;

/**
 * Servlet implementation class SubjectController
 * This class handles get request of url pattern '/subjects'.
 *
*/
@WebServlet(name = "subjects", urlPatterns = {"/subjects"})
public class SubjectController extends HttpServlet {
    private static final long serialVersionUID = 1L;
```

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

/**
 * This method takes get request of url pattern '/subjects' and fetch the list of all Subject records
 * from {@link SubjectDAO} and forward the request to subjects.jsp file.
 *
 * @param request  - {@link HttpServletRequest}
 * @param response - {@link HttpServletResponse}
 * @throws ServletException, IOException
 * @jsp /WebContent/subjects.jsp
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
 * response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    // TODO Auto-generated method stub

    response.setHeader("Cache-Control", "no-cache, no-store, must-revalidate"); // HTTP 1.1.
    response.setHeader("Pragma", "no-cache"); // HTTP 1.0.
    response.setDateHeader("Expires", 0);

    List<Subjects> subjects = SubjectDAO.getAllSubjects();
```

```
        request.setAttribute("subjects", subjects);
        request.getRequestDispatcher("subjects.jsp")
                .forward(request, response);
    }
}
```

12. TeacherController.java :- (LearnersAcademy/src/main/java/com/academy/controller/TeacherController.java)

```
package main.java.com.academy.controller;

import main.java.com.academy.dao.TeacherDAO;
import main.java.com.academy.entity.Teachers;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.List;

/**
 * Servlet implementation class TeacherController
 * This class handles get request of url pattern '/teachers' and '/teacher/.
 *

 */
@WebServlet(name = "teachers", urlPatterns = {"/teachers", "/teacher"})
public class TeacherController extends HttpServlet {
```

```
private static final long serialVersionUID = 1L;

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

/**
 * This method takes get request of url pattern '/teachers' and '/teacher/.
 * If the servlet path is '/teachers' it will fetch the list of all Teacher records
 * from {@link TeacherDAO} and forward the request to teachers.jsp file.
 * <p>
 * If the servlet path is '/teacher' it will fetch the particular Teacher records
 * from {@link TeacherDAO} and forward the request to teacher.jsp file.
 *
 * @param request  - {@link HttpServletRequest}
 * @param response - {@link HttpServletResponse}
 * @throws ServletException, IOException
 * @jsp /WebContent/teachers.jsp, /WebContent/teacher.jsp
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
 * response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
   // TODO Auto-generated method stub
```

```
response.setHeader("Cache-Control", "no-cache, no-store, must-revalidate"); // HTTP 1.1.
response.setHeader("Pragma", "no-cache"); // HTTP 1.0.
response.setDateHeader("Expires", 0);

if (request.getServletPath().equals("/teacher")) {

   int teacherId = 1;

   if (request.getParameter("teacherId") != null)
      teacherId = Integer.parseInt(request.getParameter("teacherId"));

   Teachers teacher = TeacherDAO.getTeacherWithClassesAndSubjects(teacherId);

   request.setAttribute("teacher", teacher);

   request.getRequestDispatcher("teacher.jsp")
         .forward(request, response);

} else {

   int from = 0;
   int to = 10;

   int currentPage = 1;

   if (request.getParameter("page") != null)
      currentPage = Integer.parseInt(request.getParameter("page"));

   if (currentPage > 1) {
```

```
            to *= currentPage;
            from = to - 10;
        }

        List<Teachers> teachers = TeacherDAO
                .getFewTeachersWithLimitAndOffset(10, from, false, false);

        int count = TeacherDAO.countOfTeachers();

        int pages = (int) Math.round(((double) count) / 10);

        request.setAttribute("teachers", teachers);

        request.setAttribute("pages", pages);

        request.setAttribute("currentPage", currentPage);

        request.getRequestDispatcher("teachers.jsp")
                .forward(request, response);
      }
    }
}
```

13. Authenticate.java :- (LearnersAcademy/src/main/java/com/academy/dao/Authenticate.java)

```
package main.java.com.academy.dao;
import main.java.com.academy.entity.Users;
import java.sql.Connection;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/**
 * This class handles all the authentication related tasks.
*/
public class Authenticate {

   /**
    * This method takes username and password method arguments and checks if the user is present in the database
    * records with that password and return the user object with user details. If user is not present it will
    * return empty user object.
    *
    * @param username - String name of the user
    * @param password - String password of the user
    * @return Users
    */
   public static Users getUser(String username, String password) {

      return getUser("name", username, password);
   }

   /**
    * This method takes email and password method arguments and checks if the user is present with the given email id
    * in the database records with that password and return the user object with the user details. If user is not
    * present it will return empty user object.
```

```
    *
    * @param email    - String email of the user
    * @param password - String password of the user
    * @return Users
    */
   public static Users getUserWithEmail(String email, String password) {


      return getUser("email", email, password);
   }


   /**
    * This is method is internal implementation for the above two methods. It takes field, username
and, password
    * method arguments and check if the user is present with the given field in the database record
with that password
    * and return the user object with the user details. If user is not present it will return empty user
object.
    *
    * @param field    - String field with which user will be queried
    * @param username - String username of the user
    * @param password - String password of the user
    * @return Users
    */
   private static Users getUser(String field, String username, String password) {


      Users user = null;


      String sql = String.format("SELECT * FROM users WHERE %s = '%s' AND password = '%s' AND
role = 'ADMIN'",
            field,
            username,
```

```
            password);

        try (Connection connection = Database.getConnection();
            Statement statement = connection.createStatement();
            ResultSet set = statement.executeQuery(sql)) {

            if (set.next())
                user = new Users(set.getString("name"),
                        set.getString("password"),
                        set.getString("email"),
                        set.getString("role"));

        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        return user;
    }
}
```

14. ClassDAO.java :- (LearnersAcademy/src/main/java/com/academy/dao/ClassDAO.java)

```
package main.java.com.academy.dao;

import main.java.com.academy.entity.Classes;
import main.java.com.academy.entity.Students;
import main.java.com.academy.entity.Subjects;
import main.java.com.academy.entity.Teachers;
```

```
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

/**
 * This is a DAO class for the {@link Classes} entity.

 */
public class ClassDAO {

   /**
    * This method fetches all the classes with respective strength of students for that class from the database
    * and returns list of Classes. If there are no classes present in the database it will return empty
    * list of classes.
    *
    * @return List<Classes>
    */
   public static List<Classes> getAllClassesWithStrength() {

      List<Classes> classes = new ArrayList<>();

      String sql = "SELECT * FROM classes";

      try (Connection connection = Database.getConnection();
           Statement statement = connection.createStatement();
           ResultSet set = statement.executeQuery(sql)) {

         while (set.next()) {
```

```
            int class_id = set.getInt("class_id");

            classes.add(new Classes(class_id,
                    set.getString("class_name"),
                    set.getInt("seats"),
                    StudentDAO.getClassStrength(class_id)));
        }

    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    return classes;
}

/**
 * This method fetches all the Subjects, Teachers and, Students of a particular Class with class_id and
 * returns Classes object. If there is no class with the class_id it will return empty Classes object.
 *
 * @param class_id - int class id of the Classes
 * @return Classes
 */
public static Classes getClassWithSubjectsTeachersStudents(int class_id) {

    Classes classes = null;

    String sql = "SELECT * FROM classes WHERE class_id = " + class_id;
```

```
    try (Connection connection = Database.getConnection();
         Statement statement = connection.createStatement();
         ResultSet set = statement.executeQuery(sql)) {

        if (set.next()) {

            List<Students> students = StudentDAO.getStudentsWithClassId(class_id);

            List<Subjects> subjects = ClassSubjectsTeachersDAO.getSubjectsWithClassId(connection,
class_id);

            List<Teachers> teachers = new ArrayList<>();

            for (Subjects subject : subjects)

teachers.add(ClassSubjectsTeachersDAO.getTeachersWithClassAndSubjectId(connection, class_id,
                    subject.getSubjectId()));

            classes = new Classes(class_id,
                set.getString("class_name"),
                set.getInt("seats"),
                StudentDAO.getClassStrength(class_id),
                subjects, students, teachers);
        }

    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

```
        return classes;
    }

    /**
     * This method fetches the class with the given class_id and return the Classes object. If there is no class with
     * given class_id it will return empty Classes object.
     *
     * @param class_id - int class id of the Classes
     * @return Classes
     */
    public static Classes getClass(int class_id) {

        Classes classes = null;

        try (Connection connection = Database.getConnection()) {

            String sql = "SELECT * FROM classes WHERE class_id = ?";

            PreparedStatement statement = connection.prepareStatement(sql);

            statement.setInt(1, class_id);

            ResultSet set = statement.executeQuery();

            if (set.next())
                classes = new Classes(set.getInt("class_id"),
                        set.getString("class_name"),
                        set.getInt("seats"));
```

```
        statement.close();
        set.close();

    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    return classes;
}

/**
 * This method fetches the count of all the classes in the database and return the count of classes.
 *
 * @return count
 */
public static int countOfClasses() {

    int count = 0;

    String sql = "SELECT COUNT(class_id) AS count FROM classes";

    try (Connection connection = Database.getConnection();
         Statement statement = connection.createStatement();
         ResultSet set = statement.executeQuery(sql)) {

        if (set.next())
            count = set.getInt("count");
```

```
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        return count;
    }

}
```

15. ClassSubjectsTeachersDAO.java :-

(LearnersAcademy/src/main/java/com/academy/dao/ClassSubjectsTeachersDAO.java)

```
package main.java.com.academy.dao;

import main.java.com.academy.entity.Classes;
import main.java.com.academy.entity.Subjects;
import main.java.com.academy.entity.Teachers;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

/**
 * This class handles the queries for the database table class_subjects_teachers
```

```
*/
public class ClassSubjectsTeachersDAO {


   /**
    * This method takes connection and classId as method arguments and fetches all the subjects for the given classId
    * in the database and returns list of Subjects. If there are no subjects for the given classId it will return empty
    * list.
    *
    * @param connection - Connection object
    * @param classId    -  int classId of the Classes
    * @return List<Subjects>
    */
   public static List<Subjects> getSubjectsWithClassId(Connection connection, int classId) {


      List<Subjects> subjects = new ArrayList<>();


      String sql = "SELECT subject_id from class_subjects_teachers WHERE class_id = " + classId;


      try (Statement statement = connection.createStatement(); ResultSet set = statement.executeQuery(sql)) {


         while (set.next())
            subjects.add(SubjectDAO.getSubject(set.getInt("subject_id")));


      } catch (SQLException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      }
```

```
        return subjects;
    }

    /**
     * This method takes connection and teacher_id as method arguments and fetches all the distinct classes for the
     * given teacher_id and returns a list of Classes. If there are no classes for the given teacher_id it will return
     * an empty list.
     *
     * @param connection - Connection object
     * @param teacher_id - int teacher_id
     * @return List<Classes>
     */
    public static List<Classes> getClassesWithTeacherId(Connection connection, int teacher_id) {

        List<Classes> classes = new ArrayList<>();

        String sql = "SELECT DISTINCT class_id from class_subjects_teachers WHERE teacher_id = " + teacher_id;

        try (Statement statement = connection.createStatement(); ResultSet set = statement.executeQuery(sql)) {

            while (set.next())
                classes.add(ClassDAO.getClass(set.getInt("class_id")));

        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
```

```
        }

        return classes;
    }

    /**
     * This method takes connection and teacher_id as method arguments and fetches all the distinct subjects for the
     * given teacher_id and returns a list of Subjects. If there are no subjects for the given teacher_id it will return
     * an empty list.
     *
     * @param connection - Connection object
     * @param teacher_id - int teacher_id
     * @return List<Subjects>
     */
    public static List<Subjects> getSubjectsWithTeacherId(Connection connection, int teacher_id) {

        List<Subjects> subjects = new ArrayList<>();

        String sql = "SELECT DISTINCT subject_id FROM class_subjects_teachers WHERE teacher_id = " + teacher_id;

        try (Statement statement = connection.createStatement(); ResultSet set = statement.executeQuery(sql)) {

            while (set.next())
                subjects.add(SubjectDAO.getSubject(set.getInt("subject_id")));

        } catch (SQLException e) {
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        }

        return subjects;
    }

    /**
     * This method takes connection, class_id and, subject_id as method arguments and fetches all the teachers for the
     * given class_id and subject_id and returns Teacher object. If there are no teachers for the given class_id and
     * subject_id it will return empty teacher object.
     *
     * @param connection - Connection object
     * @param class_id   - int class_id
     * @param subject_id - int subject_id
     * @return Teachers
     */
    public static Teachers getTeachersWithClassAndSubjectId(Connection connection, int class_id, int subject_id) {

        Teachers teacher = null;

        String sql = "SELECT teacher_id FROM class_subjects_teachers WHERE class_id = " + class_id
                + " AND subject_id = " + subject_id;

        try (Statement statement = connection.createStatement(); ResultSet set = statement.executeQuery(sql)) {

            if (set.next())
                teacher = TeacherDAO.getTeacher(set.getInt("teacher_id"));
```

```
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        return teacher;
    }
}
```

16. Database.java :- (LearnersAcademy/src/main/java/com/academy/dao/Database.java)

```
package main.java.com.academy.dao;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.util.Properties;

/**
 * This class handles database connection
 *
*/
public class Database implements AutoCloseable {

    private static Connection connection = null;

    /**
     * This method creates database connection for the given driver, url, username and, password and returns the
```

```
 * connection objects.
 *
 * @return Connection
 */
public static Connection getConnection() {

   try {

      Properties prop = new Database().getProperties();

      final String driver = prop.getProperty("db.driver");

      final String url = prop.getProperty("db.url");

      final String username = prop.getProperty("db.user");

      final String pass = prop.getProperty("db.password");

      Class.forName(driver);

      connection = DriverManager.getConnection(url, username, pass);

   } catch (Exception e) {

      e.printStackTrace();
   }
   return connection;
}
```

```
    /**
     * This method get the properties from config.properties files and load the properties into the
Properties object
     * and returns the object.
     *
     * @return Properties
     * @throws IOException
     */
    public Properties getProperties() throws IOException {

        Properties prop = new Properties();
        prop.load(this.getClass().getClassLoader()
                .getResourceAsStream("/config.properties"));

        return prop;
    }

    /**
     * This method closes the database connection if the connection is null.
     *
     * @throws Exception
     */
    @Override
    public void close() throws Exception {

        if (connection != null)
            connection.close();
    }
}
```

17. StudentDAO.java :- (LearnersAcademy/src/main/java/com/academy/dao/StudentDAO.java)

```
package main.java.com.academy.dao;

import main.java.com.academy.entity.Classes;
import main.java.com.academy.entity.Students;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

/**
 * This is a DAO class for {@link Students} entity.
 *

 */
public class StudentDAO {

   /**
    * This method returns the list of Students for the given method arguments.
    *
    * @param limit       - int limit of records
    * @param offset      - int offset from which records to be fetched
    * @param need_classes - boolean need_classes
    * @return List<Students>
    */
   public static List<Students> getFewStudentsWithOffset(int limit, int offset, boolean need_classes)
{

      return getStudentsWithLimitAndOffset(limit, offset, need_classes);
```

```
}

/**
 * This method counts all the students in the database and return the count of students.
 *
 * @return int
 */
public static int countOfStudents() {

   int count = 0;

   String sql = "SELECT COUNT(student_id) AS count FROM students";

   try (Connection connection = Database.getConnection();
        Statement statement = connection.createStatement();
        ResultSet set = statement.executeQuery(sql)) {

      if (set.next())
         count = set.getInt("count");

   } catch (SQLException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
   }

   return count;
}

/**
```

```
    * This method fetches all the students for the given method arguments and returns a list of Students.
    * If there are not students then it will return empty list.
    *
    * @param limit       - int limit of records
    * @param offset      - int offset from which records to be fetched
    * @param get_classes - boolean get_classes
    * @return List<Students>
    */
   private static List<Students> getStudentsWithLimitAndOffset(int limit, int offset, boolean get_classes) {

       List<Students> students = new ArrayList<>();

       String sql = "SELECT * FROM students ORDER BY student_id";

       if (limit > 0)
           sql += " LIMIT " + limit;

       if (limit > 0 && offset > 0)
           sql += " OFFSET " + offset;

       try (Connection connection = Database.getConnection();
            Statement statement = connection.createStatement();
            ResultSet set = statement.executeQuery(sql)) {

           while (set.next()) {

               Classes classes = null;
```

```
            if (get_classes)
                classes = ClassDAO.getClass(set.getInt("class_id"));

            students.add(new Students(set.getInt("student_id"),
                    set.getString("name"),
                    set.getInt("age"),
                    set.getString("gender"),
                    set.getString("email_id"), classes));
        }

    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    return students;
}

/**
 * This method returns the list of students for the given class_id. If there are no students it will
return
 * empty list.
 *
 * @param class_id - int class_id
 * @return List<Students>
 */
public static List<Students> getStudentsWithClassId(int class_id) {

    List<Students> students = new ArrayList<>();
```

```
    String sql = "SELECT * FROM students WHERE class_id = " + class_id + " ORDER BY student_id";

    try (Connection connection = Database.getConnection();
         Statement statement = connection.createStatement();
         ResultSet set = statement.executeQuery(sql)) {

        while (set.next())
            students.add(new Students(set.getInt("student_id"),
                    set.getString("name"),
                    set.getInt("age"),
                    set.getString("gender"),
                    set.getString("email_id")));

    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    return students;
}

/**
 * This method fetches the count of all the students for the given class_id.
 *
 * @param class_id - int class_id
 * @return int
 */
public static int getClassStrength(int class_id) {
```

```
        int strength = 0;

        try (Connection connection = Database.getConnection()) {

            String sql = "SELECT COUNT(class_id) AS strength FROM students WHERE class_id = ?";

            PreparedStatement statement = connection.prepareStatement(sql);

            statement.setInt(1, class_id);

            ResultSet set = statement.executeQuery();

            if (set.next())
                strength = set.getInt("strength");

            statement.close();
            set.close();

        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        return strength;
    }
}
```

18. SubjectDAO.java :- (LearnersAcademy/src/main/java/com/academy/dao/SubjectDAO.java)

```
package main.java.com.academy.dao;

import main.java.com.academy.entity.Subjects;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

/**
 * This is a DAO class for {@link Subjects} entity.
*/
public class SubjectDAO {

   /**
    * This method fetches list of all subjects in the database and returns the list of subjects. If there
are no
    * subject it will return empty list.
    *
    * @return List<Subjects>
    */
   public static List<Subjects> getAllSubjects() {

       List<Subjects> subjects = new ArrayList<>();

       String sql = "SELECT * FROM subjects ORDER BY subject_id";

       try (Connection connection = Database.getConnection();
            Statement statement = connection.createStatement();
            ResultSet set = statement.executeQuery(sql)) {
```

```
            while (set.next())
                subjects.add(new Subjects(set.getInt("subject_id"), set.getString("name")));

        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        return subjects;
    }

    /**
     * This method fetches the subjects for the given subject_id. If there is no subject it will return
empty
     * subject object.
     *
     * @param subject_id - int subject_id
     * @return Subjects
     */
    protected static Subjects getSubject(int subject_id) {

        Subjects subject = null;

        try (Connection connection = Database.getConnection()) {

            String sql = "SELECT * FROM subjects WHERE subject_id = ? ORDER BY name";

            PreparedStatement statement = connection.prepareStatement(sql);

            statement.setInt(1, subject_id);
```

```
            ResultSet set = statement.executeQuery();

            if (set.next())
               subject = new Subjects(set.getInt("subject_id"), set.getString("name"));

            statement.close();

            set.close();

         } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
         }

         return subject;
      }
   }
```

19. TeacherDAO.java :- (LearnersAcademy/src/main/java/com/academy/dao/TeacherDAO.java)

```
package main.java.com.academy.dao;

import main.java.com.academy.entity.Classes;
import main.java.com.academy.entity.Subjects;
import main.java.com.academy.entity.Teachers;

import java.sql.Connection;
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

/**
 * This is a DAO class for {@link Teachers} entity.
 *
*/
public class TeacherDAO {

   /**
    * This method fetches list of all the Teachers for the given method arguments. If there are no Teachers it will
    * return empty list.
    *
    * @param limit        - int limit of records
    * @param offset       - int offset from which record to be fetched
    * @param need_classes  - boolean if need_classes
    * @param need_subjects - boolean if need_subjects
    * @return List<Teachers>
    */
   public static List<Teachers> getFewTeachersWithLimitAndOffset(int limit, int offset,
                                                                  boolean need_classes,
                                                                  boolean need_subjects) {

      return getTeachersWithLimitAndOffset(limit, offset, need_classes, need_subjects);
   }

   /**
```

```
    * This method fetches the count of teachers in the database and return the count of the
teachers.
    *
    * @return int
    */
   public static int countOfTeachers() {

      int count = 0;

      String sql = "SELECT COUNT(teacher_id) AS count FROM teachers";

      try (Connection connection = Database.getConnection();
           Statement statement = connection.createStatement();
           ResultSet set = statement.executeQuery(sql)) {

         if (set.next())
            count = set.getInt("count");

      } catch (SQLException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      }

      return count;
   }

   /**
    * This method is internal implementation of the method getFewTeachersWithLimitAndOffset
    *
    * @param limit       - int limit of records
```

```
 * @param offset       - int offset from which record to be fetched
 * @param get_classes  - boolean get_classes
 * @param get_subjects - boolean get_subjects
 * @return List<Teachers>
 */
private static List<Teachers> getTeachersWithLimitAndOffset(int limit, int offset,
                                                            boolean get_classes,
                                                            boolean get_subjects) {

    List<Teachers> teachers = new ArrayList<>();

    String sql = "SELECT * FROM teachers ORDER BY teacher_id";

    if (limit > 0)
        sql += " LIMIT " + limit;

    if (limit > 0 && offset > 0)
        sql += " OFFSET " + offset;

    try (Connection connection = Database.getConnection();
         Statement statement = connection.createStatement();
         ResultSet set = statement.executeQuery(sql)) {

        while (set.next()) {

            List<Subjects> subjects = new ArrayList<>();

            List<Classes> classes = new ArrayList<>();
```

```
            int teacherId = set.getInt("teacher_id");

            if (get_subjects)
               subjects.addAll(ClassSubjectsTeachersDAO
                     .getSubjectsWithTeacherId(connection, teacherId));

            if (get_classes)
               classes.addAll(ClassSubjectsTeachersDAO
                     .getClassesWithTeacherId(connection, teacherId));

            teachers.add(new Teachers(teacherId,
                  set.getString("name"),
                  set.getInt("age"),
                  set.getString("gender"),
                  set.getString("email_id"),
                  subjects, classes));
         }

      } catch (SQLException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      }

      return teachers;
   }

   /**
    * This method fetches the teacher for the given teacher_id and returns the Teacher object. If
there is no Teacher
    * it will return empty teacher object.
```

```
 *
 * @param teacher_id - int teacher_id
 * @return Teachers
 */
public static Teachers getTeacher(int teacher_id) {

   Teachers teacher = null;

   String sql = "SELECT * FROM teachers WHERE teacher_id = " + teacher_id;

   try (Connection connection = Database.getConnection();
       Statement statement = connection.createStatement();
       ResultSet set = statement.executeQuery(sql)) {

      if (set.next())
         teacher = new Teachers(set.getInt("teacher_id"),
               set.getString("name"),
               set.getInt("age"),
               set.getString("gender"),
               set.getString("email_id"));

   } catch (SQLException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
   }

   return teacher;
}
```

```
  /**
   * This method fetches the teacher with all the classes and subjects associated with the teacher and returns the
   * teacher object.
   *
   * @param teacher_id - int teacher_id
   * @return Teachers
   */
  public static Teachers getTeacherWithClassesAndSubjects(int teacher_id) {

    Teachers teacher = null;

    String sql = "SELECT * FROM teachers WHERE teacher_id = " + teacher_id;

    try (Connection connection = Database.getConnection();
        Statement statement = connection.createStatement();
        ResultSet set = statement.executeQuery(sql)) {

      if (set.next()) {

        int teacherId = set.getInt("teacher_id");

        List<Classes> classes = ClassSubjectsTeachersDAO
            .getClassesWithTeacherId(connection, teacherId);
        List<Subjects> subjects = ClassSubjectsTeachersDAO
            .getSubjectsWithTeacherId(connection, teacherId);

        teacher = new Teachers(teacherId,
            set.getString("name"),
            set.getInt("age"),
```

```
                set.getString("gender"),
                set.getString("email_id"),
                subjects, classes);
          }

      } catch (SQLException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      }

      return teacher;
   }
}
```

20. Classes.java :- (LearnersAcademy/src/main/java/com/academy/entity/Classes.java)

```
package main.java.com.academy.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.io.Serializable;
import java.util.List;

/**
 * This is an entity class for the database table public.classes
 *
* @version 1.1
```

```
 */
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Classes implements Serializable {

   /**
    *
    */
   private static final long serialVersionUID = 1L;
   private int classId;
   private String name;
   private int seats;
   private int strength;
   private List<Subjects> subjects;
   private List<Students> students;
   private List<Teachers> teachers;

   public Classes(int classId, String name, int seats) {
      super();
      this.classId = classId;
      this.name = name;
      this.seats = seats;
   }

   public Classes(int classId, String name, int seats, int strength) {
      super();
      this.classId = classId;
      this.name = name;
```

```
        this.seats = seats;
        this.strength = strength;
    }
}
```

21. Students.java :- (LearnersAcademy/src/main/java/com/academy/entity/Students.java)

```
package main.java.com.academy.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.io.Serializable;

/**
 * This is an entity class for the database table public.students
 *
*/
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Students implements Serializable {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private int studentId;
```

```
    private String name;
    private int age;
    private String gender;
    private String emailId;
    private Classes cls;

    public Students(int studentId, String name, int age, String gender, String emailId) {
        super();
        this.studentId = studentId;
        this.name = name;
        this.age = age;
        this.gender = gender;
        this.emailId = emailId;
    }
}
```

22. Subjects.java :- (LearnersAcademy/src/main/java/com/academy/entity/Subjects.java)

```
package main.java.com.academy.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.io.Serializable;

/**
 * This is an entity class for the database table public.subjects
 *
```

```
 */
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Subjects implements Serializable {

   /**
    *
    */
   private static final long serialVersionUID = 1L;
   private int subjectId;
   private String name;
}
```

23. Teachers.java :- (LearnersAcademy/src/main/java/com/academy/entity/Teachers.java)

```
package main.java.com.academy.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.io.Serializable;
import java.util.List;

/**
 * This is an entity class for the database table public.teachers
 *
```

```
 * @author Riyaz J
 * @version 1.1
 */
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Teachers implements Serializable {

   /**
    *
    */
   private static final long serialVersionUID = 1L;
   private int teacherId;
   private String name;
   private int age;
   private String gender;
   private String emailId;
   private List<Subjects> subjects;
   private List<Classes> classes;

   public Teachers(int teacherId, String name, int age, String gender, String emailId) {
      super();
      this.teacherId = teacherId;
      this.name = name;
      this.age = age;
      this.gender = gender;
      this.emailId = emailId;
   }
}
```

24. Users.java :- (LearnersAcademy/src/main/java/com/academy/entity/Users.java)

```
package main.java.com.academy.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.io.Serializable;

/**
 * This is an entity class for the database table public.users
 *
*/
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Users implements Serializable {

   /**
    *
    */
   private static final long serialVersionUID = 1L;
   private String user_id;
   private String name;
   private transient String password;
   private String email;
   private String role;
```

```
    public Users(String name, String password, String email, String role) {
        super();
        this.name = name;
        this.password = password;
        this.email = email;
        this.role = role;
    }
}
```

25. AuthenticationFilter.java :- (LearnersAcademy/src/main/java/com/academy/filter/AuthenticationFilter.java)

```
package main.java.com.academy.filter;

import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

/**
 * Servlet Filter implementation class AuthenticationFilter
 * This class checks for the authentication for all the request except '/home' and '/login'
 * before hitting the request to the actual controllers.
 *

 */
@WebFilter(filterName = "authentication", urlPatterns = {"/*"})
```

```
public class AuthenticationFilter implements Filter {

    /**
     * Default constructor.
     */
    public AuthenticationFilter() {
        // TODO Auto-generated constructor stub
    }

    /**
     * @see Filter#destroy()
     */
    public void destroy() {
        // TODO Auto-generated method stub
    }

    /**
     * This method checks for the authentication for different url request and allow the request to hit the
     * actual controller if proper condition are meet, if not will return the request with an error message.
     *
     * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
     */
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
            throws IOException, ServletException {
        // TODO Auto-generated method stub
        // place your code here

        HttpServletRequest request = (HttpServletRequest) req;
```

```
    HttpServletResponse response = (HttpServletResponse) res;
    HttpSession session = request.getSession(false);

    String user = null;

    if (session != null)
        user = (String) session.getAttribute("user");

    String servletPath = request.getServletPath();

    if (session == null || user == null || user.equals(" ")) {

        if (servletPath.equals("/home") || servletPath.equals("/login"))
            chain.doFilter(req, res);
        else
            response.sendRedirect("login?login-first");

    } else {

        if (servletPath.equals("/login"))
            response.sendRedirect("home?user-exists");
        else
            chain.doFilter(req, res);
    }
}

/**
 * @see Filter#init(FilterConfig)
 */
```

```
    public void init(FilterConfig fConfig) throws ServletException {
        // TODO Auto-generated method stub
    }

}
```

26. header.jsp:- (LearnersAcademy/WebContent/includes/header.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!-- Toast Begins -->
<div
  class="toast position-absolute col-lg-4 col-sm-12 border"
  role="alert"
  id="toast-alert"
  aria-live="assertive"
  aria-atomic="true"
  data-animation="true"
  style="top: 5rem; right: 5rem"
>
  <div class="toast-header">
    <strong class="mr-auto">Learner's Academy</strong>
    <small class="text-muted">just now</small>
    <button
      type="button"
      class="ml-2 mb-1 close"
      data-dismiss="toast"
      aria-label="Close"
    >
```

```
    <span aria-hidden="true">&times;</span>
   </button>
  </div>
  <div class="toast-body" id="toast-body">Login Success</div>
</div>
<!-- Toast End -->

<% String user = (String) session.getAttribute("user"); %>

<!-- Nav-Bar Beigns -->
<div
  class="d-flex flex-column flex-md-row align-items-center p-3 px-md-4 mb-3 bg-white border-bottom box-shadow"
>
  <h5 class="my-0 mr-md-auto font-weight-normal">
   <a href="home">Learner's Academy</a>
  </h5>
  <nav class="my-2 my-md-0 mr-md-3">
   <a class="p-2 text-dark" href="classes">Classes</a>
   <a class="p-2 text-dark" href="subjects">Subjects</a>
   <a class="p-2 text-dark" href="teachers">Teachers</a>
   <a class="p-2 text-dark" href="students">Students</a>
   <a class="p-2 text-dark" href="report">Class Report</a>
  </nav>

  <% if(user == null || user.equals("")) {%>
  <a class="btn btn-outline-success mx-1" href="login">Login</a>
  <%}else { %>
  <div class="dropdown mx-1">
   <button
```

```
    class="btn btn-outline-primary dropdown-toggle"
    type="button"
    id="dropdownMenu2"
    data-toggle="dropdown"
    aria-haspopup="true"
    aria-expanded="false"
   >
    <%=user %>
   </button>
   <div class="dropdown-menu" aria-labelledby="dropdownMenu2">
    <button class="dropdown-item d-inline-block" type="button">
     <a class="btn btn-outline-primary w-100 mx-auto" href="logout"
      >Logout</a
     >
    </button>
   </div>
  </div>
  <%} %>
</div>
<!-- Nav-Bar Ends -->
```

27. footer.jsp:- (LearnersAcademy/WebContent/includes/footer.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<footer class="my-sm-4 pt-md-2 my-lg-2 pt-lg-5 border-top">
 <div class="mx-2">
  <div class="col-12 col-md text-center">
   Learner's Academy
```

```
    <span class="d-inline-block text-muted">
        |   Designed & Developed By</span
    >
    <strong>Riyaz J</strong>
   </div>
  </div>
</footer>

<!-- Bootstrap core JavaScript
   ================================================== -->
<!-- Placed at the end of the document so the pages load faster -->
<script
  src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
  integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
  crossorigin="anonymous"
></script>
<script
  src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
  integrity="sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
  crossorigin="anonymous"
></script>
<script
  src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.min.js"
  integrity="sha384-
w1Q4orYjBQndcko6MimVbzY0tgp4pWB4lZ7lr30WKz0vr/aWKhXdBNmNb5D92v7s"
  crossorigin="anonymous"
></script>

<script>
```

```
 $(document).ready(function () {
  const url = window.location.href;

  if (url.includes("?login-successful"))
   toast("Login Successful", "text-success", "border-success");
  else if (url.includes("?logout-successful"))
   toast("Logout Successful", "text-success", "border-success");
  else if (url.includes("?user-exists"))
   toast(
    "You should Logout first to Login again.",
    "text-danger",
    "border-danger"
   );

  function toast(message, textColor, borderColor) {
   const alert = $("#toast-alert");
   const body = $("#toast-body");

   body.addClass(textColor);
   body.text(message);

   alert.addClass(borderColor);
   alert.toast({ delay: 5000 });
   alert.toast("show");
  }
 });
</script>
```

28. classes.jsp:- (LearnersAcademy/WebContent/classes.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%> <%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
<!DOCTYPE html>
<html>
 <head>
  <meta charset="ISO-8859-1" />
  <title>Classes | Learners's Academy</title>

  <!--  Bootstrap CSS-->
  <link
   rel="stylesheet"
   href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
   integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
   crossorigin="anonymous"
  />

  <style>
   a,
   a:hover {
    text-decoration: none;
   }
  </style>
 </head>

 <body>
  <!-- Container Begins -->
  <div class="container">
```

```
<!-- Header Begins -->
<jsp:include page="includes/header.jsp" />
<!-- Header Ends -->

<!-- Fluid-Container Begins -->
<div class="fluid-container">
 <div class="px-3 py-3 pt-md-5 pb-md-4 mx-auto text-center">
  <h3 class="display-5">Master List Of Classes</h3>
  <p class="lead mt-3">
   You can click on any class to view detailed report of the class.
  </p>
 </div>

 <!-- Table Content Beigns -->
 <table class="table table-hover">
  <thead>
   <tr>
    <th scope="col">S.No</th>
    <th scope="col">Class Name</th>
    <th scope="col">No.of Seats</th>
    <th scope="col">Total Strength</th>
   </tr>
  </thead>
  <tbody>
   <c:forEach items="${classes}" var="cls">
    <tr>
     <th scope="row">${cls.classId }</th>
     <td>
      <a href="report?classId=${cls.classId }">${cls.name }</a>
```

```
            </td>
            <td>${cls.seats }</td>
            <td>${cls.strength }</td>
           </tr>
          </c:forEach>
         </tbody>
        </table>
        <!-- Table Content Ends -->

        <!-- Footer Begins -->
        <jsp:include page="includes/footer.jsp" />
        <!-- Footer Ends -->
       </div>
       <!-- Fluid-Container Ends -->
     </div>
     <!-- Container Ends -->
   </body>
</html>
```

29. error.jsp:- (LearnersAcademy/WebContent/error.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
```

```
    <!-- The above 3 meta tags *must* come first in the head; any other head content must come
*after* these tags -->

    <title>Error | Learners's Academy</title>
    <!-- Google font -->
    <link
      href="https://fonts.googleapis.com/css?family=Montserrat:700,900"
      rel="stylesheet"
    />

    <!-- Custom stlylesheet -->

    <style>
      * {
        -webkit-box-sizing: border-box;
        box-sizing: border-box;
      }

      body {
        padding: 0;
        margin: 0;
      }

      #notfound {
        position: relative;
        height: 100vh;
        background: #030005;
      }

      #notfound .notfound {
```

```
  position: absolute;
  left: 50%;
  top: 50%;
  -webkit-transform: translate(-50%, -50%);
  -ms-transform: translate(-50%, -50%);
  transform: translate(-50%, -50%);
}

.notfound {
  max-width: 767px;
  width: 100%;
  line-height: 1.4;
  text-align: center;
}

.notfound .notfound-404 {
  position: relative;
  height: 180px;
  margin-bottom: 20px;
  z-index: -1;
}

.notfound .notfound-404 h1 {
  font-family: "Montserrat", sans-serif;
  position: absolute;
  left: 50%;
  top: 50%;
  -webkit-transform: translate(-50%, -50%);
  -ms-transform: translate(-50%, -50%);
```

```
  transform: translate(-50%, -50%);
  font-size: 224px;
  font-weight: 900;
  margin-top: 0px;
  margin-bottom: 0px;
  margin-left: -12px;
  color: #030005;
  text-transform: uppercase;
  text-shadow: -1px -1px 0px #8400ff, 1px 1px 0px #ff005a;
  letter-spacing: -20px;
}

.notfound .notfound-404 h2 {
  font-family: "Montserrat", sans-serif;
  position: absolute;
  left: 0;
  right: 0;
  top: 110px;
  font-size: 30px;
  font-weight: 700;
  color: #fff;
  text-transform: uppercase;
  text-shadow: 0px 2px 0px #8400ff;
  letter-spacing: 13px;
  margin: 0;
}

.notfound a {
  font-family: "Montserrat", sans-serif;
```

```
  display: inline-block;
  text-transform: uppercase;
  color: #ff005a;
  text-decoration: none;
  border: 2px solid;
  background: transparent;
  padding: 10px 40px;
  font-size: 14px;
  font-weight: 700;
  -webkit-transition: 0.2s all;
  transition: 0.2s all;
 }

 .notfound a:hover {
  color: #8400ff;
 }

 @media only screen and (max-width: 767px) {
  .notfound .notfound-404 h2 {
   font-size: 24px;
  }
 }

 @media only screen and (max-width: 480px) {
  .notfound .notfound-404 h1 {
   font-size: 182px;
  }
 }
</style>
```

```
 </head>

 <body>
  <div id="notfound">
   <div class="notfound">
    <div class="notfound-404">
     <h1>${statusCode}</h1>
     <h2>${error}</h2>
    </div>
    <a href="${path}">Home Page</a>
   </div>
  </div>
 </body>
</html>
```

30. index.jsp:- (LearnersAcademy/WebContent/index.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
 <head>
  <meta charset="ISO-8859-1" />
  <title>Learners's Academy</title>

  <!--  Bootstrap CSS-->
  <link
   rel="stylesheet"
   href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
```

```
    integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
    crossorigin="anonymous"
  />

  <style>
    a,
    a:hover {
      text-decoration: none;
    }
  </style>
</head>

<body>
  <!-- Container Begins -->
  <div class="container">
    <!-- Header Begins -->
    <jsp:include page="includes/header.jsp" />
    <!-- Header Ends -->

    <!-- Container Begins -->
    <div class="fluid-container">
      <div class="px-3 py-3 pt-md-5 pb-md-4 mx-auto text-center">
        <h1 class="display-5">Welcome To Learner's Academy</h1>
        <p class="lead mt-3">
          An online management tool to view and manage Classes, Subjects and
          Teachers.
        </p>
      </div>
```

```
<div class="card-deck text-center">
  <div class="card mb-4 box-shadow">
    <div class="card-body">
      <h6 class="card-title">
        <div
          class="alert alert-warning alert-dismissible fade show"
          role="alert"
        >
          <strong>Note!</strong> You must login to view and manage data.
          <button
            type="button"
            class="close"
            data-dismiss="alert"
            aria-label="Close"
          >
            <span aria-hidden="true">&times;</span>
          </button>
        </div>
      </h6>

      <!-- Row Begin -->
      <div class="row">
        <!-- Tabs Headers Begin -->
        <div class="col-4">
          <div class="list-group" id="list-tab" role="tablist">
            <a
              class="list-group-item list-group-item-action active"
              id="list-classes-list"
              data-toggle="list"
```

```
 href="#list-classes"
 role="tab"
 aria-controls="classes"
 >Classes</a
>
<a
 class="list-group-item list-group-item-action"
 id="list-subjects-list"
 data-toggle="list"
 href="#list-subjects"
 role="tab"
 aria-controls="subjects"
 >Subjects</a
>
<a
 class="list-group-item list-group-item-action"
 id="list-teachers-list"
 data-toggle="list"
 href="#list-teachers"
 role="tab"
 aria-controls="teachers"
 >Teachers</a
>

<a
 class="list-group-item list-group-item-action"
 id="list-students-list"
 data-toggle="list"
 href="#list-students"
```

```
      role="tab"
      aria-controls="students"
      >Students</a
    >
    <a
      class="list-group-item list-group-item-action"
      id="list-report-list"
      data-toggle="list"
      href="#list-report"
      role="tab"
      aria-controls="report"
      >Class Report</a
    >
  </div>
</div>
<!-- Tabs Headers End -->

<!-- Tabs Content Begin -->
<div class="col-8">
  <div class="tab-content my-3" id="nav-tabContent">
    <!-- Class Tab Begin -->
    <div
      class="tab-pane fade show active"
      id="list-classes"
      role="tabpanel"
      aria-labelledby="list-classes-list"
    >
      <div class="container">
        <h3 class="card-title pricing-card-title">
```

```
    Master List of Classes
   </h3>
   <strong class="text-muted my-4 d-block"
    >Here you can view the master list of all the Classes
    of the school.</strong
   >
   <div class="d-block text-muted">
    <a href="classes" class="btn btn-outline-success"
     >View Classes</a
    >
   </div>
  </div>
</div>
<!-- Class Tab End -->

<!-- Subjects Tab Begin -->
<div
 class="tab-pane fade"
 id="list-subjects"
 role="tabpanel"
 aria-labelledby="list-subjects-list"
>
 <div class="container">
  <h3 class="card-title pricing-card-title">
   Master List of Subjects
  </h3>
  <strong class="text-muted my-4 d-block"
   >Here you can view the master list of all the Subjects
   of the school.</strong
```

```
    >
    <div class="d-block text-muted">
      <a href="subjects" class="btn btn-outline-success"
        >View Subjects</a
      >
    </div>
  </div>
</div>
<!-- Subjects Tab Ends -->

<!-- Teachers Tab Begin -->
<div
  class="tab-pane fade"
  id="list-teachers"
  role="tabpanel"
  aria-labelledby="list-teachers-list"
>
  <div class="container">
    <h3 class="card-title pricing-card-title">
      Master List of Teachers
    </h3>
    <strong class="text-muted my-4 d-block"
      >Here you can view the master list of all the Teachers
      of the school.</strong
    >
    <div class="d-block text-muted">
      <a href="teachers" class="btn btn-outline-success"
        >View Teachers</a
      >
```

```
    </div>
  </div>
</div>
<!-- Teachers Tab End -->

<!-- Subjects Tab Begin -->
<div
  class="tab-pane fade"
  id="list-students"
  role="tabpanel"
  aria-labelledby="list-students-list"
>
  <div class="container">
    <h3 class="card-title pricing-card-title">
      Master List of Subjects
    </h3>
    <strong class="text-muted my-4 d-block"
      >Here you can view the master list of all
      Subjects.</strong
    >
    <div class="d-block text-muted">
      <a
        href="class-subjects"
        class="btn btn-outline-success"
        >View Subjects</a
      >
    </div>
  </div>
</div>
```

```
    <!-- Subjects Tab End -->

    <!-- Class Report Tab Begin -->
    <div
      class="tab-pane fade"
      id="list-report"
      role="tabpanel"
      aria-labelledby="list-report-list"
    >
      <div class="container">
        <h3 class="card-title pricing-card-title">
          Class Report
        </h3>
        <strong class="text-muted my-4 d-block"
          >Here you can view the Class Report. This tab contains
          all information about Classes, Subjects and Teachers
          for a perticular class.</strong
        >
        <div class="d-block text-muted">
          <a href="classes" class="btn btn-outline-success"
            >View Class Report</a
          >
        </div>
      </div>
    </div>
    <!-- Class Report Tab End -->
  </div>
</div>
<!-- Tabs Content End -->
```

```
            </div>
            <!-- Row End -->
          </div>
        </div>
      </div>
      <!-- Container Ends -->

      <!-- Footer Begins -->
      <jsp:include page="includes/footer.jsp" />
      <!-- Footer Ends -->
    </div>
  </div>
  <!-- Container Ends -->
 </body>
</html>
```

31. login.html:- (LearnersAcademy/WebContent/login.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
 <title>Login | Learner's Academy</title>

 <!--  Bootstrap CSS-->
 <link
      rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
      integrity="sha384-
TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
      crossorigin="anonymous"
```

```
 />
</head>
<body class="text-center">
<div class="container py-5">
 <!-- Alert Begins -->
 <div
       class="alert alert-warning alert-dismissible fade d-none col-8 mx-auto border"
       id="alert-ms"
       role="alert"
 >
  <strong>Oops!</strong>
  <span id="alert-message"
  >You should check in on some of those fields below.</span
  >
  <button
        type="button"
        class="close"
        data-dismiss="alert"
        aria-label="Close"
  >
    <span aria-hidden="true">&times;</span>
  </button>
 </div>
 <!-- Alert Ends -->
 <div class="col-sm-7 col-lg-5 mx-auto my-5">
  <form class="form-signin my-5" method="POST" action="login">
   <h3 class="text-center my-5">
     Login to
     <a class="text-decoration-none" href="home">Learner's Academy</a>
```

```
</h3>
<label for="user-name" class="sr-only">User Name or Email ID</label>
<input
      type="text"
      id="user-name"
      name="username"
      class="form-control my-3"
      placeholder="User Name or Email ID"
      required="required"
      autofocus="autofocus"
      minlength="4"
      maxlength="20"
/>
<label for="inputPassword" class="sr-only">Password</label>
<input
      type="password"
      id="inputPassword"
      name="password"
      class="form-control my-3"
      placeholder="Password"
      required="required"
      minlength="4"
/>
<div class="checkbox mb-3">
  <label>
    <input type="checkbox" value="remember-me" /> Remember me
  </label>
</div>
<button class="btn btn-lg btn-primary btn-block" type="submit">
```

```
      Sign in
    </button>
    <p class="mt-5 mb-3 text-muted">
      Designed & Developed <strong class="text-muted">Riyaz J</strong>
    </p>
  </form>
 </div>
</div>

<!-- JQuery and Bootstrap Javascript libraries
 ================================================= -->
<!-- Placed at the end of the document so the pages load faster -->
<script
      src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
      integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
      crossorigin="anonymous"
></script>

<script
      src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.min.js"
      integrity="sha384-
w1Q4orYjBQndcko6MimVbzY0tgp4pWB4lZ7lr30WKz0vr/aWKhXdBNmNb5D92v7s"
      crossorigin="anonymous"
></script>

<script>
    $(document).ready(function () {
      const url = window.location.href;
```

```
        if (url.includes("?login-failed"))
          showAlert("You should check in on some of those fields below.");
        else if (url.includes("?login-first"))
          showAlert("You must login first to view and manage data.");

        function showAlert(message) {
          const alert_ms = document.getElementById("alert-ms");
          alert_ms.classList.remove("d-none");
          alert_ms.classList.add("show");
          document.getElementById("alert-message").innerHTML = message;
        }
      });
    </script>
</body>
</html>
```

32. report.jsp:- (LearnersAcademy/WebContent/report.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%> <%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="ISO-8859-1" />
    <title>Report | Learners's Academy</title>

    <!--  Bootstrap CSS-->
    <link
```

```
    rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
    integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
    crossorigin="anonymous"
  />

  <style>
    a,
    a:hover {
      text-decoration: none;
    }
  </style>
</head>

<body>
  <!-- Container Begins -->
  <div class="container">
    <!-- Header Begins -->
    <jsp:include page="includes/header.jsp" />
    <!-- Header Ends -->

    <!-- Fluid-Container Begins -->
    <div class="fluid-container">
      <div class="px-3 py-3 pt-md-5 pb-md-4 mx-auto text-center">
        <h3 class="display-5">Detailed Report of ${cls.name }</h3>
        <p class="lead mt-3">
          You can click on the following links to view the respective details.
        </p>
      </div>
```

```
<!-- Card Beigns -->
<div class="accordion" id="classReport">
  <!-- Overview Card Begins -->
  <div class="card">
    <div class="card-header" id="headingOne">
      <h2 class="mb-0">
        <button
          class="btn btn-link btn-block text-left text-decoration-none"
          type="button"
          data-toggle="collapse"
          data-target="#overview"
          aria-expanded="true"
          aria-controls="overview"
        >
          Overview
        </button>
      </h2>
    </div>

    <div
      id="overview"
      class="collapse show"
      aria-labelledby="headingOne"
      data-parent="#classReport"
    >
      <div class="card-body">
        <div class="info text-right">
          <h5 class="card-title pricing-card-title text-info">
```

```
   Overview of ${cls.name }
  </h5>
  <strong class="text-muted my-4 d-block"
   >This tab shows brief info about the class.</strong
  >
 </div>
 <table class="table table-hover">
  <tbody>
   <tr>
    <th scope="row">Class Id</th>
    <td>${cls.classId }</td>
   </tr>
   <tr>
    <th scope="row">Class Name</th>
    <td>${cls.name }</td>
   </tr>
   <tr>
    <th scope="row">No.Of Seats</th>
    <td>${cls.seats }</td>
   </tr>
   <tr>
    <th scope="row">No.Of Students</th>
    <td>${cls.strength }</td>
   </tr>
   <tr>
    <th scope="row">No.Of Teachers</th>
    <td>${cls.teachers.size() }</td>
   </tr>
  </tbody>
```

```
    </table>
   </div>
  </div>
</div>
<!-- Overview Card Ends -->

<!-- Student Card Begins -->
<div class="card">
  <div class="card-header" id="headingTwo">
    <h2 class="mb-0">
      <button
        class="btn btn-link btn-block text-left text-decoration-none collapsed"
        type="button"
        data-toggle="collapse"
        data-target="#students"
        aria-expanded="false"
        aria-controls="students"
      >
        Students
      </button>
    </h2>
  </div>
  <div
    id="students"
    class="collapse"
    aria-labelledby="headingTwo"
    data-parent="#classReport"
  >
    <div class="card-body">
```

```
<div class="info text-right">
  <h5 class="card-title pricing-card-title text-success">
    List of Students
  </h5>
  <strong class="text-muted my-4 d-block"
    >This tab shows List of all Students belongs to this
    class.</strong
  >
</div>
<table class="table table-hover">
  <thead>
    <tr>
      <th scope="col">S.No</th>
      <th scope="col">Student Name</th>
      <th scope="col">Age</th>
      <th scope="col">Gender</th>
      <th scope="col">Email ID</th>
    </tr>
  </thead>
  <tbody>
    <c:forEach var="student" items="${cls.students}">
      <tr>
        <th scope="row">${student.studentId }</th>
        <td>${student.name }</td>
        <td>${student.age }</td>
        <td>${student.gender }</td>
        <td>${student.emailId }</td>
      </tr>
    </c:forEach>
```

```
      </tbody>
     </table>
    </div>
   </div>
  </div>
 </div>
 <!-- Student Card Ends -->

 <!-- Subjects and Teachers Card Begins -->
 <div class="card">
  <div class="card-header" id="headingThree">
   <h2 class="mb-0">
    <button
     class="btn btn-link btn-block text-left text-decoration-none collapsed"
     type="button"
     data-toggle="collapse"
     data-target="#subjectsTeachers"
     aria-expanded="false"
     aria-controls="subjectsTeachers"
    >
     Subjects & Teachers
    </button>
   </h2>
  </div>
  <div
   id="subjectsTeachers"
   class="collapse"
   aria-labelledby="headingThree"
   data-parent="#classReport"
  >
```

```
<div class="card-body">
  <div class="info text-right">
    <h5 class="card-title pricing-card-title text-danger">
      List of Subjects and respective Teachers
    </h5>
    <strong class="text-muted my-4 d-block"
      >This tab shows List of all Subjects and respective
      Teachers. Click on the Teacher name for more
      details.</strong
    >
  </div>
  <table class="table table-hover">
    <thead>
      <tr>
        <th scope="col">S.No</th>
        <th scope="col">Subject Name</th>
        <th scope="col">Teacher Name</th>
      </tr>
    </thead>
    <tbody>
      <c:forEach
        begin="0"
        end="${cls.teachers.size() - 1}"
        varStatus="loop"
      >
        <tr>
          <th scope="row">${loop.index + 1}</th>
          <td>${cls.subjects[loop.index].name }</td>
          <td>
```

```
                <a
                  href="teacher?teacherId=${cls.teachers[loop.index].teacherId }"
                  >${cls.teachers[loop.index].name }</a
                >
              </td>
            </tr>
          </c:forEach>
        </tbody>
      </table>
    </div>
  </div>
 </div>
 <!-- Subjects and Teachers Card Ends -->
</div>
<!-- Card Ends -->
<!-- Pagination Begins -->
<section class="mt-5">
 <nav aria-label="Page navigation example">
  <ul class="pagination justify-content-center">
   <li class="page-item ${1 == currentPage ? 'disabled' : ''}">
    <a
     class="page-link"
     href="report?classId=${currentPage - 1}"
     tabindex="-1"
     aria-disabled="true"
     >Previous</a
    >
   </li>
```

```
            <c:forEach begin="1" end="${pages }" varStatus="loop">
              <li
                class="page-item ${loop.index == currentPage ? 'disabled' : ''}"
              >
                <a class="page-link" href="report?classId=${loop.index}"
                  >${loop.index}</a
                >
              </li>
            </c:forEach>

            <li class="page-item ${pages == currentPage ? 'disabled' : ''}">
              <a class="page-link" href="report?classId=${currentPage + 1}"
                >Next</a
              >
            </li>
          </ul>
        </nav>
      </section>
      <!-- Pagination Ends -->

      <!-- Footer Begins -->
      <jsp:include page="includes/footer.jsp" />
      <!-- Footer Ends -->
    </div>
    <!-- Fluid-Container Ends -->
  </div>
  <!-- Container Ends -->
 </body>
</html>
```

33. students.jsp:- (LearnersAcademy/WebContent/students.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%> <%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
<!DOCTYPE html>
<html>
 <head>
  <meta charset="ISO-8859-1" />
  <title>Students | Learners's Academy</title>

  <!--  Bootstrap CSS-->
  <link
   rel="stylesheet"
   href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
   integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
   crossorigin="anonymous"
  />

  <style>
   a,
   a:hover {
    text-decoration: none;
   }
  </style>
 </head>

 <body>
```

```
<!-- Container Begins -->
<div class="container">
  <!-- Header Begins -->
  <jsp:include page="includes/header.jsp" />
  <!-- Header Ends -->

  <!-- Fluid-Container Begins -->
  <div class="fluid-container">
    <div class="px-3 py-3 pt-md-5 pb-md-4 mx-auto text-center">
      <h3 class="display-5">Master List Of Students</h3>
      <p class="lead mt-3">
        Here you have the master list of all the Students with respective
        Class with pagination support.
      </p>
    </div>

    <!-- Table Content Beigns -->
    <table class="table table-hover">
      <thead>
        <tr>
          <th scope="col">S.No</th>
          <th scope="col">Students Name</th>
          <th scope="col">Age</th>
          <th scope="col">Gender</th>
          <th scope="col">Email ID</th>
          <c:if test="${showClass }">
            <th scope="col">Class</th>
          </c:if>
        </tr>
```

```
  </thead>
  <tbody>
    <c:forEach items="${students}" var="student">
      <tr>
        <th scope="row">${student.studentId }</th>
        <td>${student.name }</td>
        <td>${student.age }</td>
        <td>${student.gender }</td>
        <td>${student.emailId }</td>
        <c:if test="${showClass }">
          <td class="d-flex px-0">
            <span class="mx-1 badge badge-pill badge-success p-2"
              >${student.cls.name }</span
            >
          </td>
        </c:if>
      </tr>
    </c:forEach>
  </tbody>
</table>
<!-- Table Content Ends -->

<!-- Pagination Begins -->
<section class="mt-5">
  <nav aria-label="Page navigation example">
    <ul class="pagination justify-content-center">
      <li class="page-item ${1 == currentPage ? 'disabled' : ''}">
        <a
          class="page-link"
```

```
        href="students?page=${currentPage - 1}"
        tabindex="-1"
        aria-disabled="true"
        >Previous</a
      >
    </li>

    <c:forEach begin="1" end="${pages }" varStatus="loop">
      <li
        class="page-item ${loop.index == currentPage ? 'disabled' : ''}"
      >
        <a class="page-link" href="students?page=${loop.index}"
          >${loop.index}</a
        >
      </li>
    </c:forEach>

    <li class="page-item ${pages == currentPage ? 'disabled' : ''}">
      <a class="page-link" href="students?page=${currentPage + 1}"
        >Next</a
      >
    </li>
  </ul>
 </nav>
</section>
<!-- Pagination Ends -->

<!-- Footer Begins -->
<jsp:include page="includes/footer.jsp" />
```

```
        <!-- Footer Ends -->
      </div>
      <!-- Fluid-Container Ends -->
    </div>
    <!-- Container Ends -->
  </body>
</html>
```

34. subjects.jsp:- (LearnersAcademy/WebContent/subjects.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%> <%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
<!DOCTYPE html>
<html>
 <head>
  <meta charset="ISO-8859-1" />
  <title>Subjects | Learners's Academy</title>

  <!--  Bootstrap CSS-->
  <link
   rel="stylesheet"
   href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
   integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
   crossorigin="anonymous"
  />

  <style>
   a,
   a:hover {
    text-decoration: none;
   }
  </style>
 </head>

 <body>
  <!-- Container Begins -->
```

```
<div class="container">
  <!-- Header Begins -->
  <jsp:include page="includes/header.jsp" />
  <!-- Header Ends -->

  <!-- Fluid-Container Begins -->
  <div class="fluid-container">
    <div class="px-3 py-3 pt-md-5 pb-md-4 mx-auto text-center">
      <h3 class="display-5">Master List Of Subjects</h3>
      <p class="lead mt-3">Here is the master list of all the subjects.</p>
    </div>

    <!-- Table Content Beigns -->
    <table class="table table-hover">
      <thead>
        <tr>
          <th scope="col">S.No</th>
          <th scope="col">Subject Name</th>
        </tr>
      </thead>
      <tbody>
        <c:forEach items="${subjects}" var="sub">
          <tr>
            <th scope="row">${sub.subjectId}</th>
            <td>${sub.name}</td>
          </tr>
        </c:forEach>
      </tbody>
    </table>
```

```
        <!-- Table Content Ends -->

        <!-- Footer Begins -->
        <jsp:include page="includes/footer.jsp" />
        <!-- Footer Ends -->
      </div>
      <!-- Fluid-Container Ends -->
    </div>
    <!-- Container Ends -->
  </body>
</html>
```

35. teacher.jsp:- (LearnersAcademy/WebContent/teacher.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%> <%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="ISO-8859-1" />
    <title>Teacher | Learners's Academy</title>

    <!--  Bootstrap CSS-->
    <link
      rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
      integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
      crossorigin="anonymous"
```

```
 />

 <style>
  a,
  a:hover {
   text-decoration: none;
  }
 </style>
</head>

<body>
 <!-- Container Begins -->
 <div class="container">
  <!-- Header Begins -->
  <jsp:include page="includes/header.jsp" />
  <!-- Header Ends -->

  <!-- Fluid-Container Begins -->
  <div class="fluid-container">
   <div class="px-3 py-3 pt-md-5 pb-md-4 mx-auto text-center">
    <h3 class="display-5">Details of ${teacher.name }</h3>
    <p class="lead mt-3">
     Here you will find complete details of the teacher.
    </p>
   </div>

   <!-- Table Content Beigns -->
   <table class="table table-hover">
    <tbody>
```

```
<tr>
  <th scope="col">S.No</th>
  <td>${teacher.teacherId }</td>
</tr>
<tr>
  <th scope="col">Teacher Name</th>
  <td>${teacher.name }</td>
</tr>
<tr>
  <th scope="col">Age</th>
  <td>${teacher.age }</td>
</tr>
<tr>
  <th scope="col">Gender</th>
  <td>${teacher.gender }</td>
</tr>
<tr>
  <th scope="col">Email ID</th>
  <td>${teacher.emailId }</td>
</tr>
<tr>
  <th scope="col">Subjects</th>
  <td class="d-flex px-0">
    <c:forEach var="subject" items="${teacher.subjects}">
      <span class="mx-1 badge badge-pill badge-primary p-2"
        >${subject.name }</span
      >
    </c:forEach>
  </td>
```

```
          </tr>
          <tr>
            <th scope="col">Classes</th>
            <td class="d-flex px-0">
              <c:forEach var="classes" items="${teacher.classes}">
                <span class="mx-1 badge badge-pill badge-success p-2"
                  >${classes.name }</span
                >
              </c:forEach>
            </td>
          </tr>
        </tbody>
      </table>
      <!-- Table Content Ends -->

      <!-- Footer Begins -->
      <jsp:include page="includes/footer.jsp" />
      <!-- Footer Ends -->
    </div>
    <!-- Fluid-Container Ends -->
  </div>
  <!-- Container Ends -->
 </body>
</html>
```

36. teachers.jsp:- (LearnersAcademy/WebContent/teachers.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%> <%@ taglib uri="http://java.sun.com/jsp/jstl/core"
```

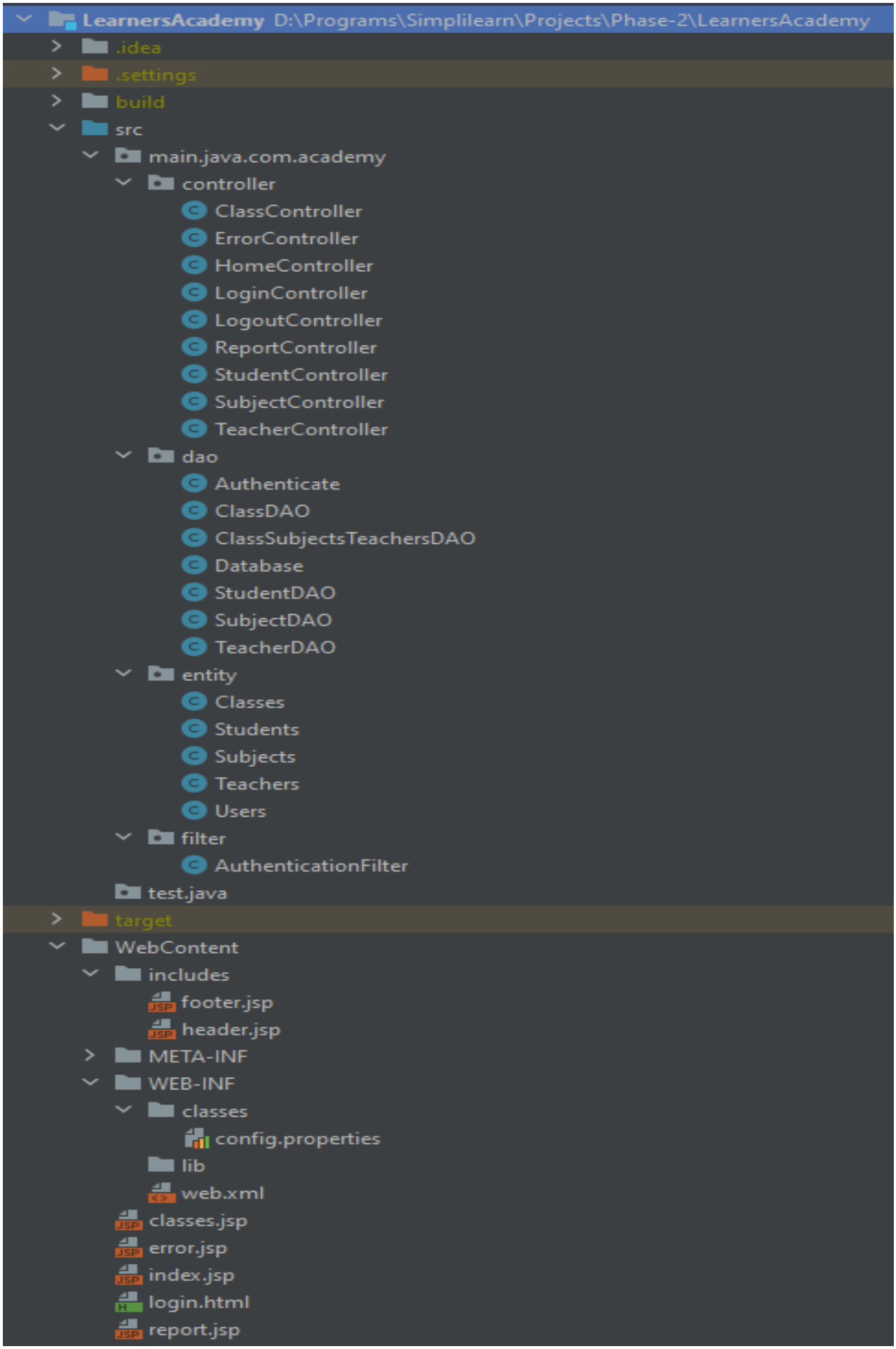
```
prefix="c"%>
<!DOCTYPE html>
<html>
 <head>
  <meta charset="ISO-8859-1" />
  <title>Teachers | Learners's Academy</title>

  <!--  Bootstrap CSS-->
  <link
   rel="stylesheet"
   href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
   integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
   crossorigin="anonymous"
  />

  <style>
   a,
   a:hover {
    text-decoration: none;
   }
  </style>
 </head>

 <body>
  <!-- Container Begins -->
  <div class="container">
   <!-- Header Begins -->
   <jsp:include page="includes/header.jsp" />
   <!-- Header Ends -->
```

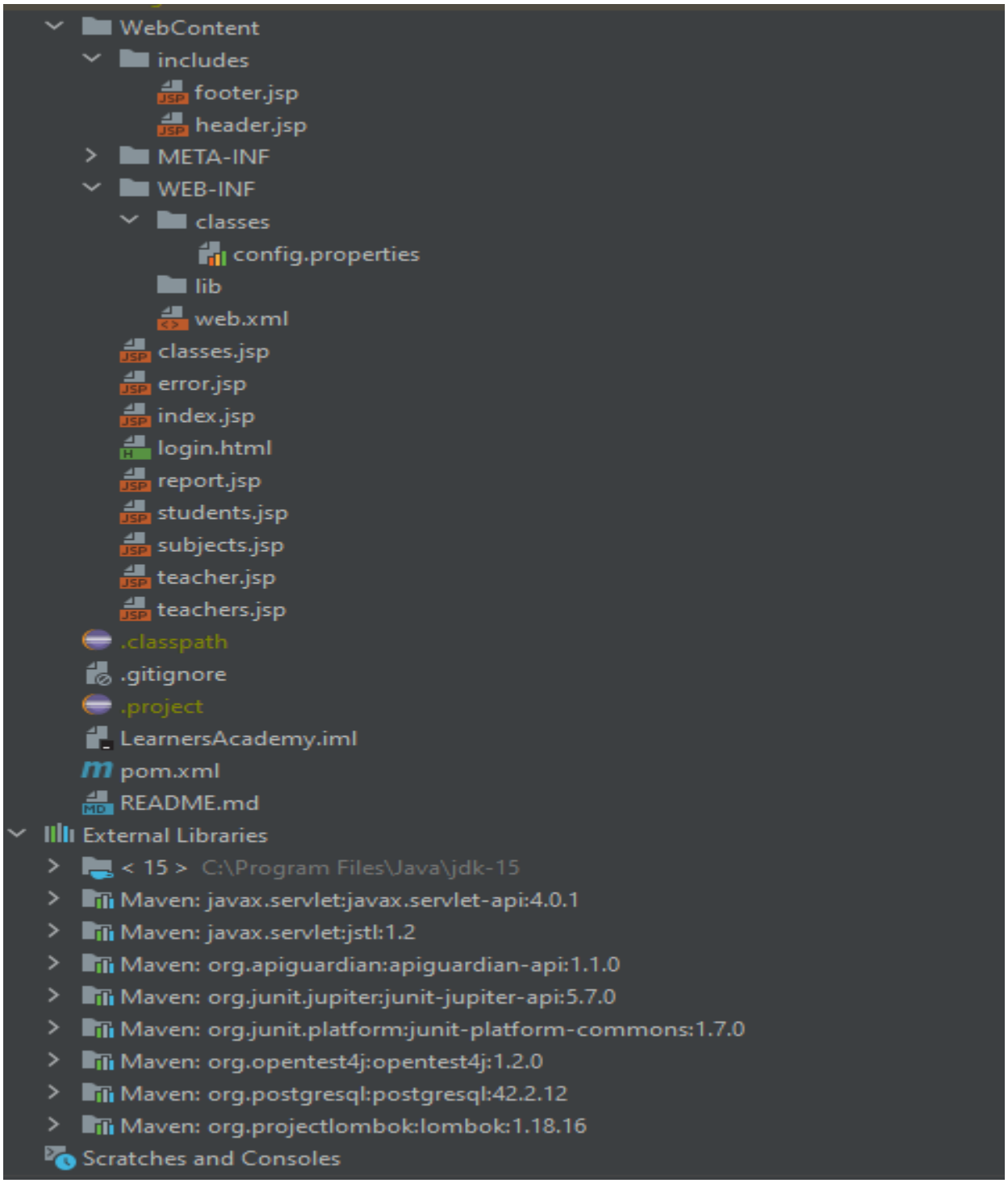
```
<!-- Fluid-Container Begins -->
<div class="fluid-container">
  <div class="px-3 py-3 pt-md-5 pb-md-4 mx-auto text-center">
    <h3 class="display-5">Master List Of Teachers</h3>
    <p class="lead mt-3">
      Here you have the master list of all the Teachers with respective
      Subjects with pagination support. Click on the teacher to find more
      details.
    </p>
  </div>

  <!-- Table Content Begin -->
  <table class="table table-hover">
    <thead>
      <tr>
        <th scope="col">S.No</th>
        <th scope="col">Teacher Name</th>
      </tr>
    </thead>
    <tbody>
      <c:forEach items="${teachers}" var="teacher">
        <tr>
          <th scope="row">${teacher.teacherId }</th>
          <td>
            <a href="teacher?teacherId=${teacher.teacherId }"
              >${teacher.name }</a
            >
          </td>
```

```
    </tr>
   </c:forEach>
  </tbody>
</table>
<!-- Table Content Ends -->

<!-- Pagination Begins -->
<section class="mt-5">
  <nav aria-label="Page navigation example">
    <ul class="pagination justify-content-center">
      <li class="page-item ${1 == currentPage ? 'disabled' : ''}">
        <a
          class="page-link"
          href="teachers?page=${currentPage - 1}"
          tabindex="-1"
          aria-disabled="true"
          >Previous</a
        >
      </li>

      <c:forEach begin="1" end="${pages }" varStatus="loop">
        <li
          class="page-item ${loop.index == currentPage ? 'disabled' : ''}"
        >
          <a class="page-link" href="teachers?page=${loop.index}"
            >${loop.index}</a
          >
        </li>
      </c:forEach>
```

```
            <li class="page-item ${pages == currentPage ? 'disabled' : ''}">
              <a class="page-link" href="teachers?page=${currentPage + 1}"
                >Next</a
              >
            </li>
          </ul>
        </nav>
      </section>
      <!-- Pagination Ends -->

      <!-- Footer Begins -->
      <jsp:include page="includes/footer.jsp" />
      <!-- Footer Ends -->
    </div>
    <!-- Fluid-Container Ends -->
  </div>
  <!-- Container Ends -->
 </body>
</html>
```

Picture of Project Structure


LearnersAcademy D:\Programs\Simplilearn\Projects\Phase-2\LearnersAcademy
.idea
.settings
build
src
main.java.com.academy
controller
ClassController
ErrorController
HomeController
LoginController
LogoutController
ReportController
StudentController
SubjectController
TeacherController
dao
Authenticate
ClassDAO
ClassSubjectsTeachersDAO
Database
StudentDAO
SubjectDAO
TeacherDAO
entity
Classes
Students
Subjects
Teachers
Users
filter
AuthenticationFilter
test.java
target
WebContent
includes
footer.jsp
header.jsp
META-INF
WEB-INF
classes
config.properties
lib
web.xml
classes.jsp
error.jsp
index.jsp
login.html
report.jsp


WebContent
includes
footer.jsp
header.jsp
META-INF
WEB-INF
classes
config.properties
lib
web.xml
classes.jsp
error.jsp
index.jsp
login.html
report.jsp
students.jsp
subjects.jsp
teacher.jsp
teachers.jsp
.classpath
.gitignore
.project
LearnersAcademy.iml
pom.xml
README.md
External Libraries
< 15 > C:\Program Files\Java\jdk-15
Maven: javax.servlet:javax.servlet-api:4.0.1
Maven: javax.servlet:jstl:1.2
Maven: org.apiguardian:apiguardian-api:1.1.0
Maven: org.junit.jupiter:junit-jupiter-api:5.7.0
Maven: org.junit.platform:junit-platform-commons:1.7.0
Maven: org.opentest4j:opentest4j:1.2.0
Maven: org.postgresql:postgresql:42.2.12
Maven: org.projectlombok:lombok:1.18.16
Scratches and Consoles