Developing a Backend Admin for Learner's Academy

Background of the problem statement:
Learner's Academy is a school that has an online management system. The system keeps track of its classes, subjects, students, and teachers. It has a back-office application with a single administrator login.

Github repo	https://github.com/yujeshkc/Learnersadmin
Developed by	Yujesh KC

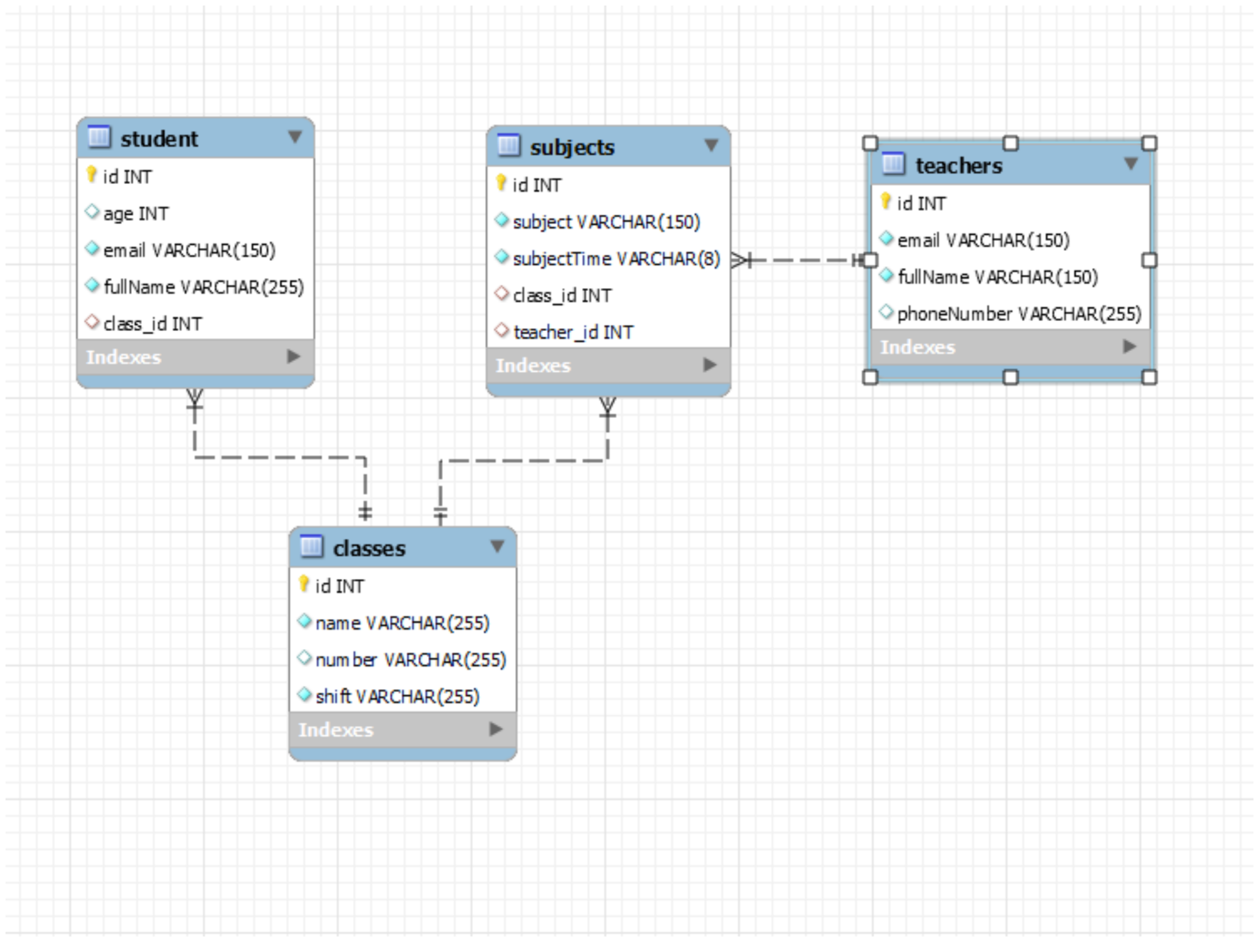
1. Sprints planning and Task completion

1.1. High-level solution design
1.2. Create Repo on Github
1.3. Write Program as per the requirement.
1.4. Testing
1.5. Lunch (publish on Github)

2. Core concepts used in the project

Hibernate: ORM is an object–relational mapping tool for the Java programming language.
Servlet : For business logic and works as a controller for the project.
Mysql: To Store data.
JSP: For handling the presentation view.
CSS: Styling to content
Eclipse: write and run the code.
JDBC: operations on the database for the project.
Tomcat: To run and deploy servlet applications.

3. EER Diagram



4. Setup and Initial data

Run the program then all tables will create and add initial data from MySqlData/data.sql. Run data.sql in mysql workbench or phpmyadmin by selecting default schema as project database.
*Important: project name must be **learnersadmin***

Edit the database' properties such as username, password and driverClassName to be suit to your database in **[4.2] HibernateUtil.java**

Username: **admin**, password: **password**

Url List of Teacher, classes, Student, subject

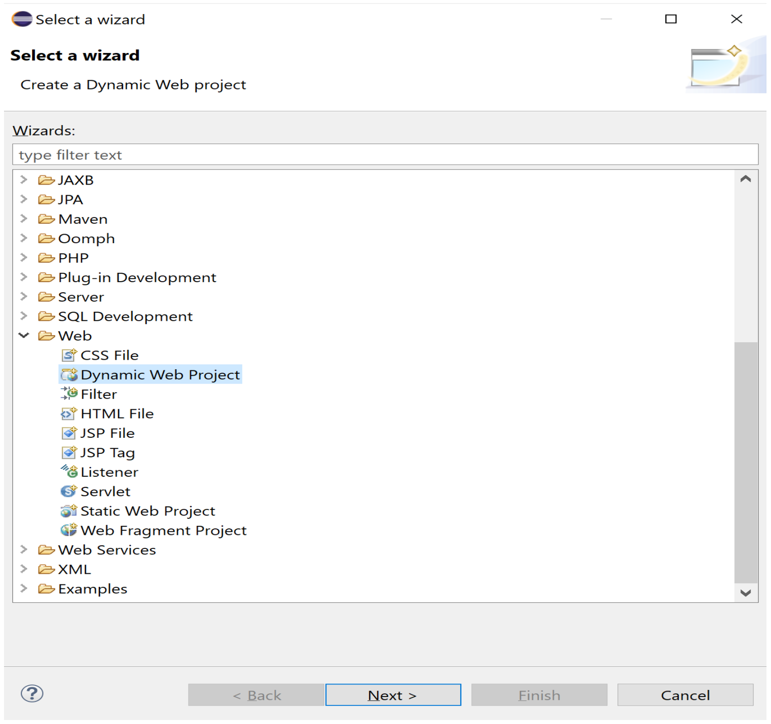
URL	***Detail***
http://localhost:8080/learnersadmin/login	Login
http://localhost:8080/learnersadmin/teacher/index	List of Teacher
http://localhost:8080/learnersadmin/classes/index	List of Classes
http://localhost:8080/learnersadmin/student/index?class_id=<class_id>	List of student of class
http://localhost:8080/learnersadmin/subject/index?class_id=<class_id>	List of subject of class

5. Tasks

5.1. Create New Project in Eclipse.

5.1.1. Open eclipse

5.1.2. Go to File -> Other-> Web-> Dynamic Web Project->Next


Select a wizard

Select a wizard

Create a Dynamic Web project

Wizards:

type filter text

- JAXB
- JPA
- Maven
- Oomph
- PHP
- Plug-in Development
- Server
- SQL Development
- Web
  - CSS File
  - Dynamic Web Project
  - Filter
  - HTML File
  - JSP File
  - JSP Tag
  - Listener
  - Servlet
  - Static Web Project
  - Web Fragment Project
- Web Services
- XML
- Examples

5.1.3. Add Project Name -> **learnersadmin**
Target runtime: **apache-tomcat-7.0.109**
Click **Finish**

5.1.4. Create 3 package
Learnersadmin.controller (all the business logic and incoming requests, manipulate data)
Learnersadmin.dao (objects that abstract away the data storage mechanism.)
Learnersadmin.model (Entities - POJO that represent data)

5.1.5. Learnersadmin.controller

ClassesController.java
LoginController.java
StudentController.java

SubjectController.java
TeacherController.java

5.1.6. Learnersadmin.dao

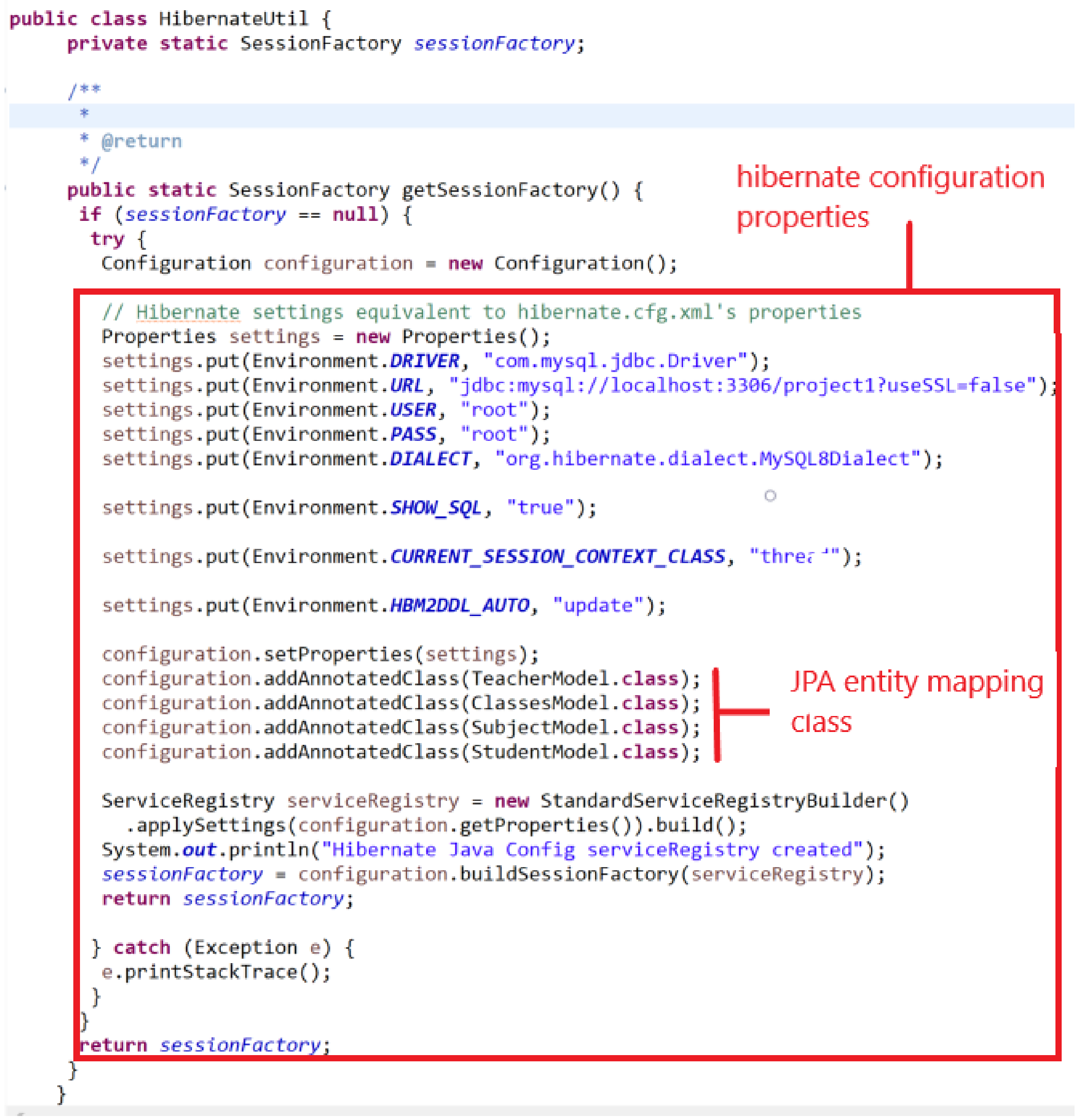
ClassesDao.java
HibernateUtil.java
LoginDao.java
StudentDao.java
SubjectDao.java
TeacherDao.java

5.1.7. Learnersadmin.model

ClassesModel.java
StudentModel.java
SubjectModel.java
TeacherModel.java

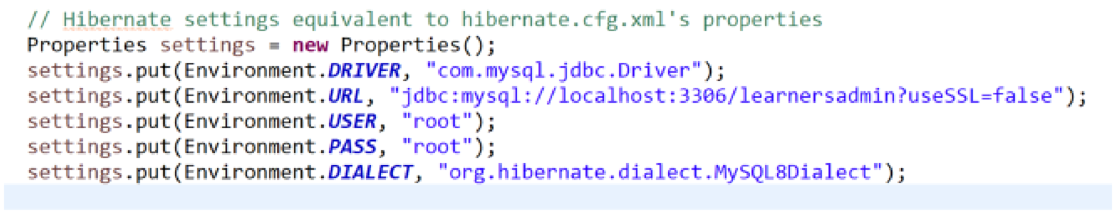
5.2. **HibernateUtil.java**

Hibernate Application using Java configuration to connect MySQL database. Hibernate settings equivalent to hibernate.cfg.xml properties.



```
public class HibernateUtil {
     private static SessionFactory sessionFactory;

     /**
      *
      * @return
      */
     public static SessionFactory getSessionFactory() {
      if (sessionFactory == null) {
       try {
        Configuration configuration = new Configuration();

        // Hibernate settings equivalent to hibernate.cfg.xml's properties
        Properties settings = new Properties();
        settings.put(Environment.DRIVER, "com.mysql.jdbc.Driver");
        settings.put(Environment.URL, "jdbc:mysql://localhost:3306/project1?useSSL=false");
        settings.put(Environment.USER, "root");
        settings.put(Environment.PASS, "root");
        settings.put(Environment.DIALECT, "org.hibernate.dialect.MySQL8Dialect");

        settings.put(Environment.SHOW_SQL, "true");

        settings.put(Environment.CURRENT_SESSION_CONTEXT_CLASS, "thread");

        settings.put(Environment.HBM2DDL_AUTO, "update");

        configuration.setProperties(settings);
        configuration.addAnnotatedClass(TeacherModel.class);
        configuration.addAnnotatedClass(ClassesModel.class);
        configuration.addAnnotatedClass(SubjectModel.class);
        configuration.addAnnotatedClass(StudentModel.class);

        ServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder()
          .applySettings(configuration.getProperties()).build();
        System.out.println("Hibernate Java Config serviceRegistry created");
        sessionFactory = configuration.buildSessionFactory(serviceRegistry);
        return sessionFactory;

       } catch (Exception e) {
        e.printStackTrace();
       }
      }
      return sessionFactory;
     }
    }
```

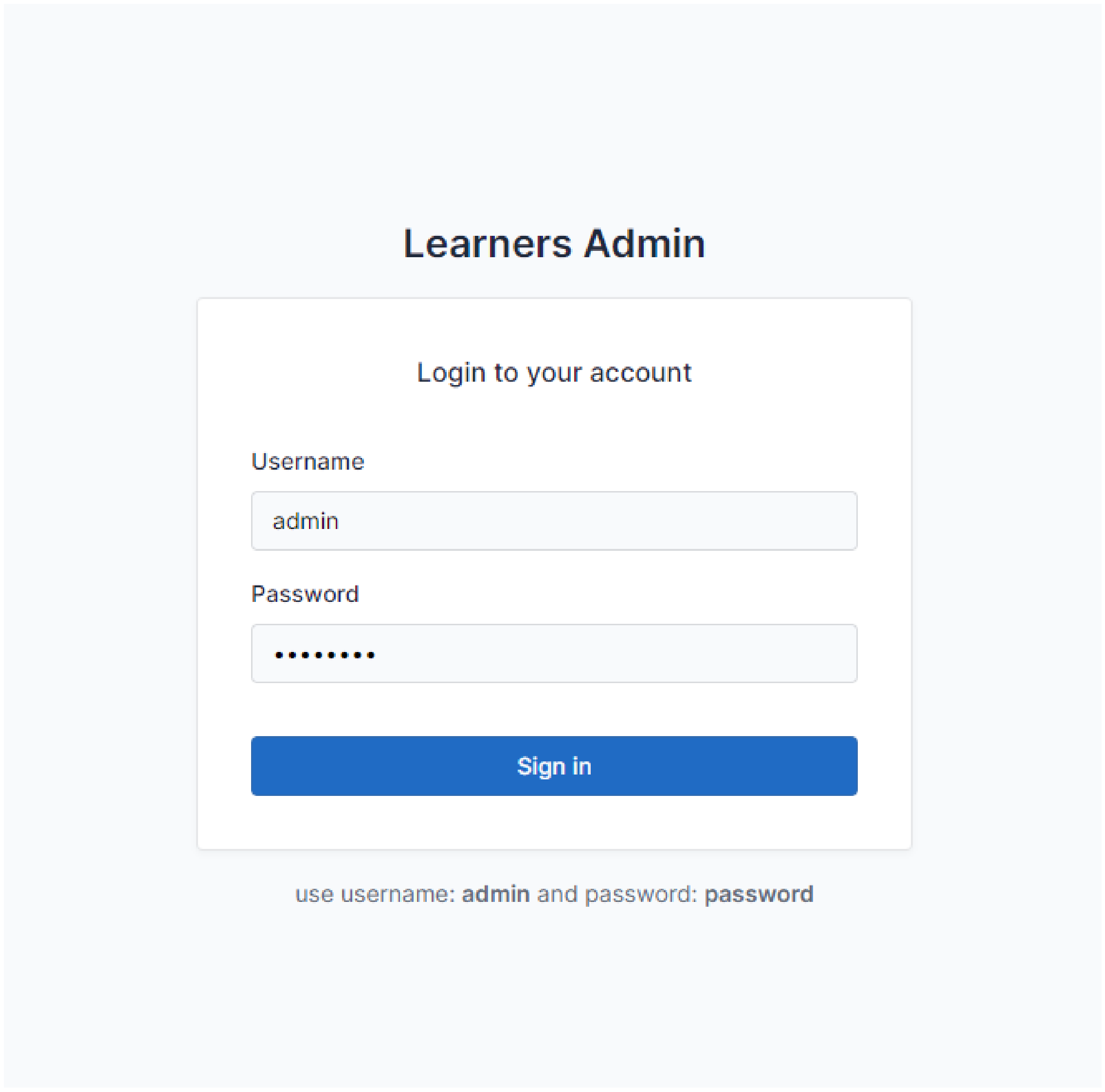
5.2.1. Hibernate settings



```
// Hibernate settings equivalent to hibernate.cfg.xml's properties
Properties settings = new Properties();
settings.put(Environment.DRIVER, "com.mysql.jdbc.Driver");
settings.put(Environment.URL, "jdbc:mysql://localhost:3306/learnersadmin?useSSL=false");
settings.put(Environment.USER, "root");
settings.put(Environment.PASS, "root");
settings.put(Environment.DIALECT, "org.hibernate.dialect.MySQL8Dialect");
```

Environment.DRIVER	com.mysql.jdbc.Driver
Environment.URL	jdbc:mysql://localhost:3306/learnersadmin?useSSL=false
Environment.USER	root
Environment.PASS	root
Environment.DIALECT	org.hibernate.dialect.MySQL8Dialect
Environment.SHOW_SQL	true
Environment.HBM2DDL_AUTO	update

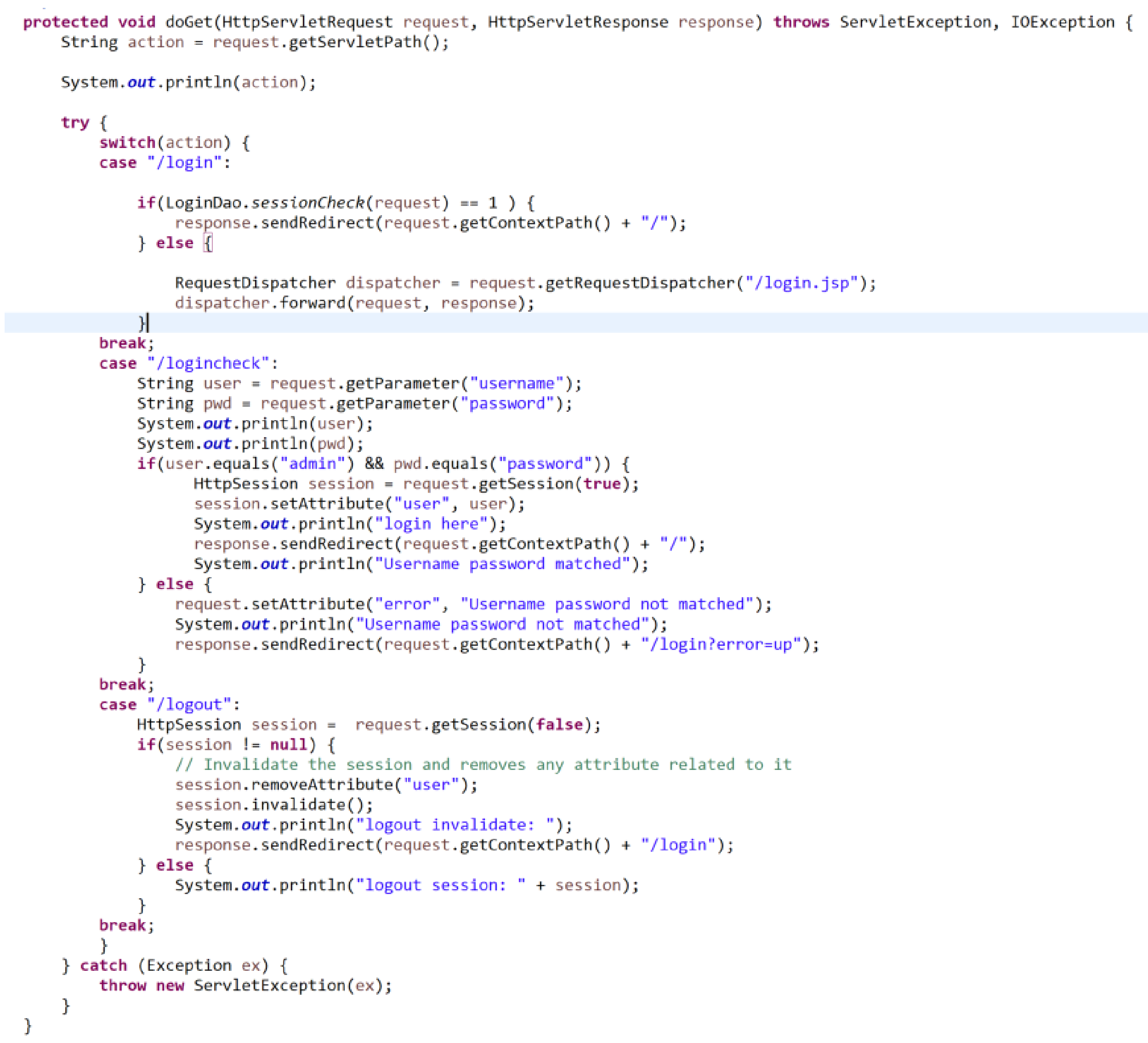
5.3. Login and session management



use username: **admin** and password: **password**

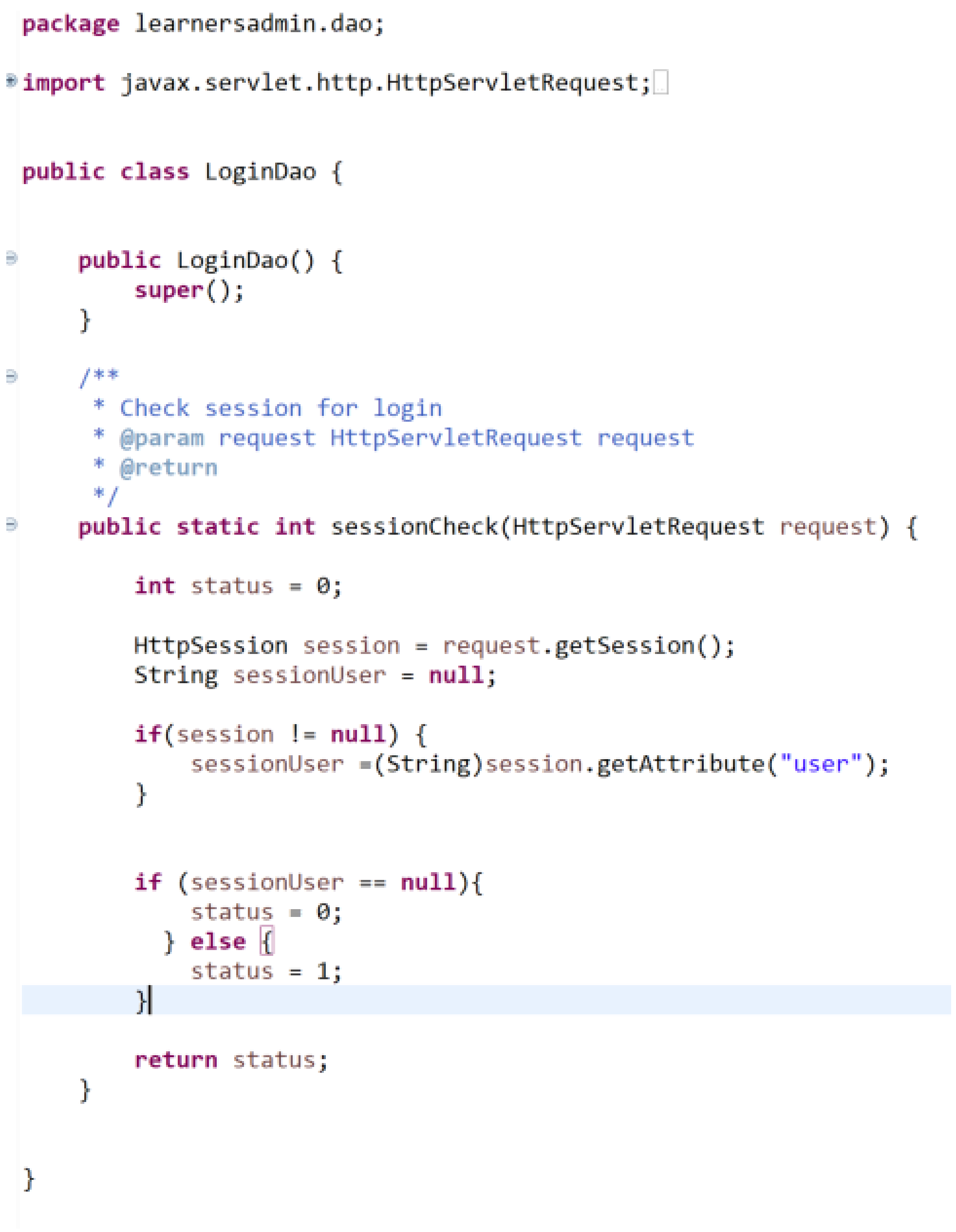
LoginController.java servlet file for login, logincheck, logout

@WebServlet(urlPatterns = {"/login","/logout","/logincheck"})



```java
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String action = request.getServletPath();

    System.out.println(action);

    try {
        switch(action) {
        case "/login":

            if(LoginDao.sessionCheck(request) == 1 ) {
                response.sendRedirect(request.getContextPath() + "/");
            } else {

                RequestDispatcher dispatcher = request.getRequestDispatcher("/login.jsp");
                dispatcher.forward(request, response);
            }
        break;
        case "/logincheck":
            String user = request.getParameter("username");
            String pwd = request.getParameter("password");
            System.out.println(user);
            System.out.println(pwd);
            if(user.equals("admin") && pwd.equals("password")) {
                  HttpSession session = request.getSession(true);
                  session.setAttribute("user", user);
                  System.out.println("login here");
                  response.sendRedirect(request.getContextPath() + "/");
                  System.out.println("Username password matched");
            } else {
                request.setAttribute("error", "Username password not matched");
                System.out.println("Username password not matched");
                response.sendRedirect(request.getContextPath() + "/login?error=up");
            }
        break;
        case "/logout":
            HttpSession session =  request.getSession(false);
            if(session != null) {
                // Invalidate the session and removes any attribute related to it
                session.removeAttribute("user");
                session.invalidate();
                System.out.println("logout invalidate: ");
                response.sendRedirect(request.getContextPath() + "/login");
            } else {
                System.out.println("logout session: " + session);
            }
        break;
        }
    } catch (Exception ex) {
        throw new ServletException(ex);
    }
}
```

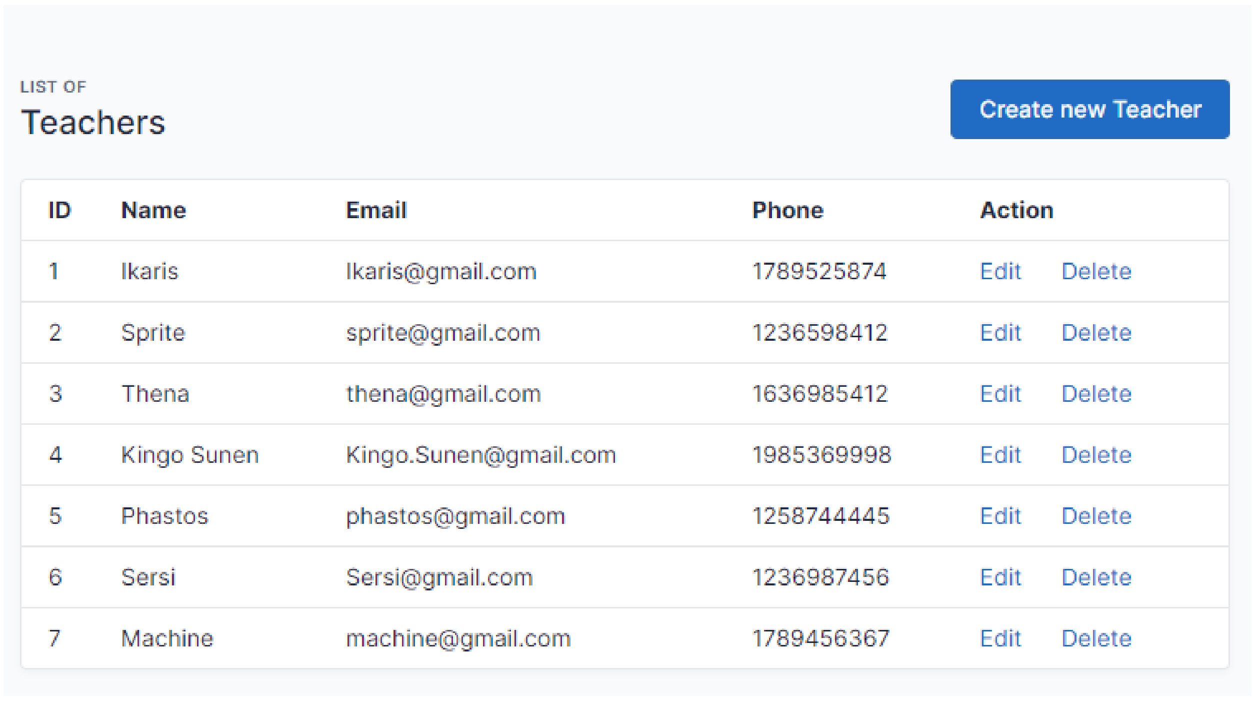
LoginDao.java - check session exist or not



```java
package learnersadmin.dao;

import javax.servlet.http.HttpServletRequest;


public class LoginDao {


    public LoginDao() {
        super();
    }

    /**
     * Check session for login
     * @param request HttpServletRequest request
     * @return
     */
    public static int sessionCheck(HttpServletRequest request) {

        int status = 0;

        HttpSession session = request.getSession();
        String sessionUser = null;

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


        if (sessionUser == null){
            status = 0;
          } else {
            status = 1;
        }

        return status;
    }


}
```

6. List of all the Teachers

http://localhost:8080/learnersadmin/teacher/index



LIST OF

Teachers

ID	Name	Email	Phone	Action
1	Ikaris	Ikaris@gmail.com	1789525874	Edit Delete
2	Sprite	sprite@gmail.com	1236598412	Edit Delete
3	Thena	thena@gmail.com	1636985412	Edit Delete
4	Kingo Sunen	Kingo.Sunen@gmail.com	1985369998	Edit Delete
5	Phastos	phastos@gmail.com	1258744445	Edit Delete
6	Sersi	Sersi@gmail.com	1236987456	Edit Delete
7	Machine	machine@gmail.com	1789456367	Edit Delete

TeacherController.java

```
/**
	 * List all Teacher
	 *
	 * @param request
	 * @param response
	 * @throws SQLException
	 * @throws IOException
	 * @throws ServletException
	 */
	private void listTeachers(HttpServletRequest request, HttpServletResponse response)
			throws SQLException, IOException, ServletException {

		List<TeacherModel> listTeacher = TeacherDao.getAllTeacher();
		request.setAttribute("listTeacher", listTeacher);

		RequestDispatcher dispatcher =   request.getRequestDispatcher("/teacher/index.jsp");
		dispatcher.forward(request, response);

	}
```

TeacherDao.java

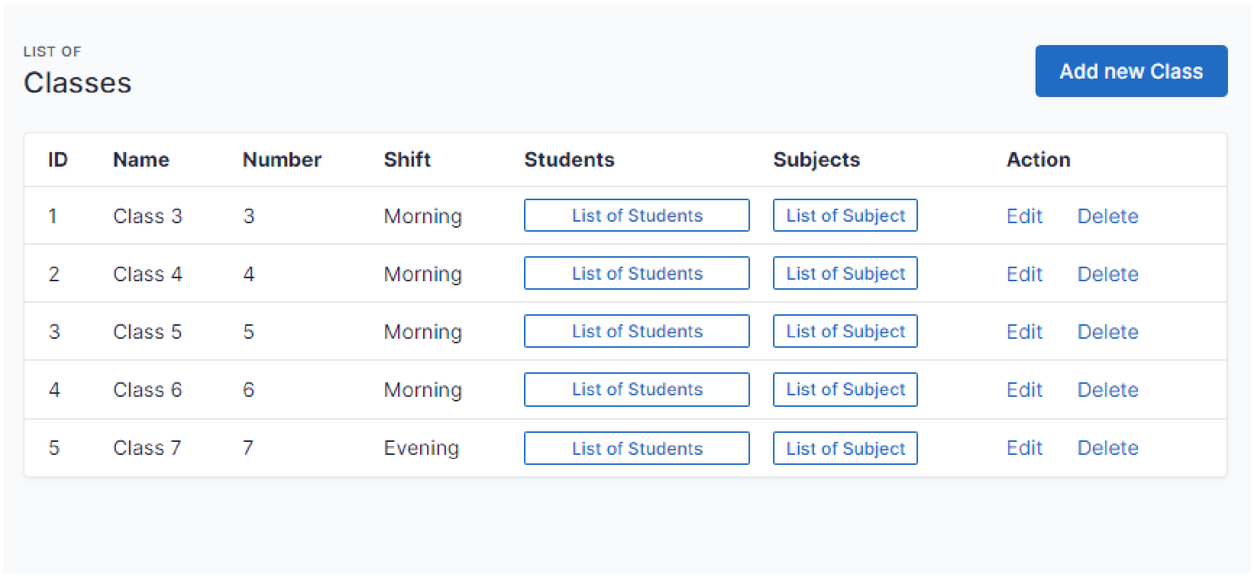
```
/**
         * Get all Teacher
         *
         * @return
         */
        @SuppressWarnings("unchecked")
        public static List<TeacherModel> getAllTeacher() {

                Transaction transaction = null;
                List<TeacherModel> listOfTeacher = null;
                Session session = HibernateUtil.getSessionFactory().openSession();

                try {
                        // start a transaction
                        transaction = session.beginTransaction();
                        listOfTeacher = session.createQuery("FROM teachers").getResultList();
                        transaction.commit();

                } catch (Exception e) {
                        transaction.rollback();
                        e.printStackTrace();
                }
                return listOfTeacher;
        }
```

TeacherModel.java

```java
package learnersadmin.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;


@Entity (name = "teachers")
@Table(name = "teachers")
public class TeacherModel {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;

	@Column(unique=false, nullable=false, length= 150)
	private String fullName;

	private String phoneNumber;

	@Column(unique=true, nullable=false, length= 150)
	private String email;


	public TeacherModel() {
		super();
		// TODO Auto-generated constructor stub
	}




	public TeacherModel(int id, String fullName, String phoneNumber, String email) {
		super();
		this.id = id;
		this.fullName = fullName;
		this.phoneNumber = phoneNumber;
		this.email = email;
	}


	public TeacherModel(String fullName, String phoneNumber, String email) {
		super();
		this.fullName = fullName;
		this.phoneNumber = phoneNumber;
		this.email = email;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}
	public String getFullName() {
		return fullName;
	}
	public void setFullName(String fullName) {
		this.fullName = fullName;
	}
	public String getPhoneNumber() {
		return phoneNumber;
	}
	public void setPhoneNumber(String phoneNumber) {
		this.phoneNumber = phoneNumber;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}

	@Override
	public String toString() {
		return "TeacherModel [id=" + id + ", fullName=" + fullName + ",
phoneNumber=" + phoneNumber + ", email=" + email
				+ "]";
	}

}
```

7. List of all the classes

http://localhost:8080/learnersadmin/classes/index



ID	Name	Number	Shift	Students	Subjects	Action
1	Class 3	3	Morning	List of Students	List of Subject	Edit Delete
2	Class 4	4	Morning	List of Students	List of Subject	Edit Delete
3	Class 5	5	Morning	List of Students	List of Subject	Edit Delete
4	Class 6	6	Morning	List of Students	List of Subject	Edit Delete
5	Class 7	7	Evening	List of Students	List of Subject	Edit Delete

ClassesController.java

```
/**
 * List all the classes for the index view
 *
 * @param request
 * @param response
 * @throws SQLException
 * @throws IOException
 * @throws ServletException
 */
private void listClasses(HttpServletRequest request, HttpServletResponse response)
                throws SQLException, IOException, ServletException {

        List<ClassesModel> listClasses = ClassesDao.getAllClasses();
        request.setAttribute("listClasses", listClasses);

        RequestDispatcher dispatcher = request.getRequestDispatcher("/classes/index.jsp");
        dispatcher.forward(request, response);

}
```

ClassesDao.java

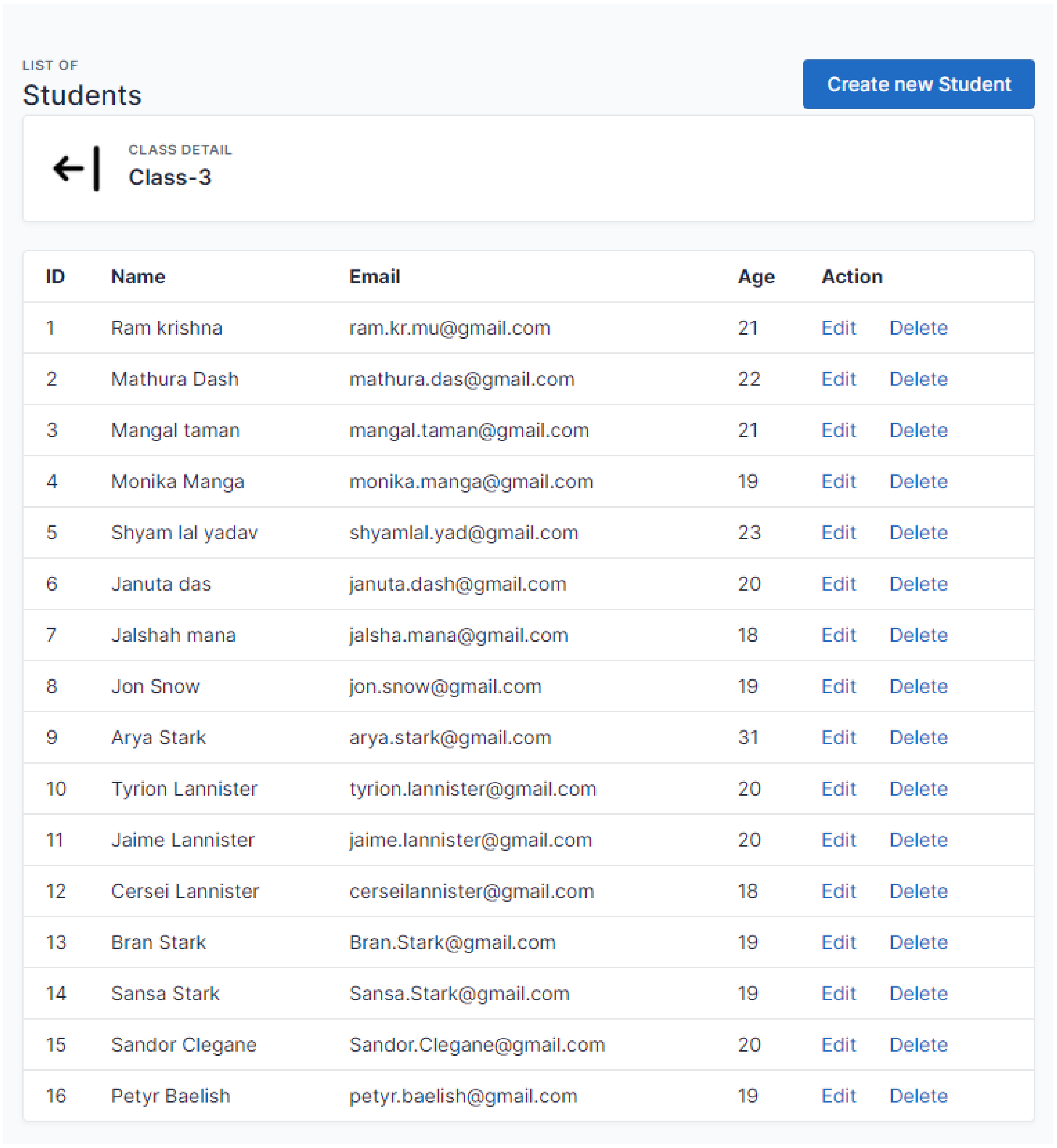
```java
/**
         * Get all Classes
         *
         * @return
         */
        @SuppressWarnings("unchecked")
        public static List<ClassesModel> getAllClasses() {

                Transaction transaction = null;
                List<ClassesModel> listOfClasses = null;
                Session session = HibernateUtil.getSessionFactory().openSession();

                try {

                        // start a transaction
                        transaction = session.beginTransaction();
                        listOfClasses = session.createQuery("FROM classes").getResultList();
                        transaction.commit();

                } catch (Exception e) {
                        transaction.rollback();
                        e.printStackTrace();
                }

                return listOfClasses;

        }
```

ClassesModel.java

```java
package learnersadmin.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

import javax.persistence.Table;

@Entity(name = "classes")
@Table(name = "classes")
```

```java
public class ClassesModel {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;

	@Column(nullable = false, name = "name")
	private String className;

	@Column(nullable = true, name = "number")
	private String classNumber;

	@Column(nullable = false, name = "shift")
	private String classShift;

	public ClassesModel() {
		super();
		// TODO Auto-generated constructor stub
	}

	public ClassesModel(int id) {
		super();
		this.id = id;
	}

	public ClassesModel(int id, String className, String classNumber, String classShift) {
		super();
		this.id = id;
		this.className = className;
		this.classNumber = classNumber;
		this.classShift = classShift;
	}

	public ClassesModel(String className, String classNumber, String classShift) {
		super();
		this.className = className;
		this.classNumber = classNumber;
		this.classShift = classShift;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
			this.id = id;
		}

		public String getClassName() {
			return className;
		}

		public void setClassName(String className) {
			this.className = className;
		}

		public String getClassNumber() {
			return classNumber;
		}

		public void setClassNumber(String classNumber) {
			this.classNumber = classNumber;
		}

		public String getClassShift() {
			return classShift;
		}

		public void setClassShift(String classShift) {
			this.classShift = classShift;
		}

}
```

8. Master list of students



LIST OF

Students

CLASS DETAIL

Class-3

ID	Name	Email	Age	Action	
1	Ram krishna	ram.kr.mu@gmail.com	21	Edit	Delete
2	Mathura Dash	mathura.das@gmail.com	22	Edit	Delete
3	Mangal taman	mangal.taman@gmail.com	21	Edit	Delete
4	Monika Manga	monika.manga@gmail.com	19	Edit	Delete
5	Shyam lal yadav	shyamlal.yad@gmail.com	23	Edit	Delete
6	Januta das	januta.dash@gmail.com	20	Edit	Delete
7	Jalshah mana	jalsha.mana@gmail.com	18	Edit	Delete
8	Jon Snow	jon.snow@gmail.com	19	Edit	Delete
9	Arya Stark	arya.stark@gmail.com	31	Edit	Delete
10	Tyrion Lannister	tyrion.lannister@gmail.com	20	Edit	Delete
11	Jaime Lannister	jaime.lannister@gmail.com	20	Edit	Delete
12	Cersei Lannister	cerseilannister@gmail.com	18	Edit	Delete
13	Bran Stark	Bran.Stark@gmail.com	19	Edit	Delete
14	Sansa Stark	Sansa.Stark@gmail.com	19	Edit	Delete
15	Sandor Clegane	Sandor.Clegane@gmail.com	20	Edit	Delete
16	Petyr Baelish	petyr.baelish@gmail.com	19	Edit	Delete

StudentController.java

```java
/**
 * List all the students
 *
 * @param request
 * @param response
 * @throws SQLException
 * @throws IOException
 * @throws ServletException
 */
private void listStudents(HttpServletRequest request, HttpServletResponse response)
            throws SQLException, IOException, ServletException {

    int class_id = Integer.parseInt(request.getParameter("class_id"));
    List<StudentModel> listStudents = StudentDao.getAllStudents(class_id);

    ClassesModel studentClasses = ClassesDao.getClasses(class_id);

    request.setAttribute("listStudents", listStudents);
    request.setAttribute("classes", studentClasses);
    RequestDispatcher dispatcher = request.getRequestDispatcher("/student/index.jsp");
    dispatcher.forward(request, response);

}
```

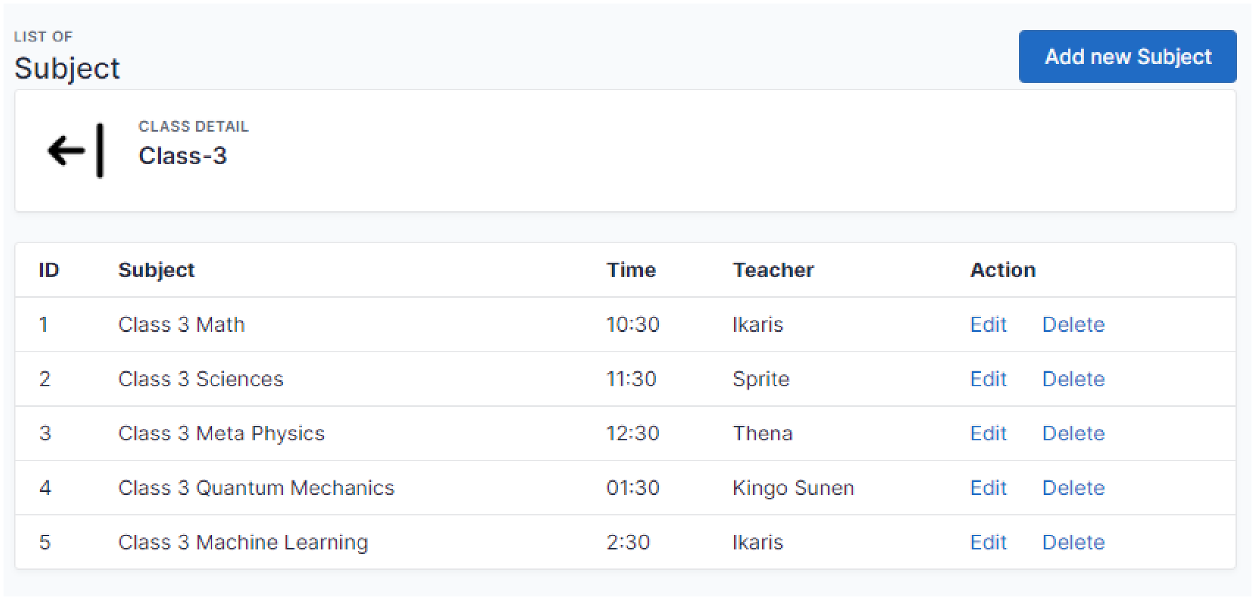
StudentDao.java

```java
@SuppressWarnings("unchecked")
    public static List<StudentModel> getAllStudents(int class_id) {

        Transaction transaction = null;
        List<StudentModel> listOfStudent = null;
        Session session = HibernateUtil.getSessionFactory().openSession();

        try { // start a transaction
            transaction = session.beginTransaction();
            listOfStudent = session.createQuery("FROM student WHERE class_id = " + class_id).getResultList();
            transaction.commit();
        } catch (Exception e) {
            transaction.rollback();
            e.printStackTrace();
        }
        return listOfStudent;
        }
```

StudentModel.java

```
package learnersadmin.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Table(name = "student")
@Entity(name = "student")
public class StudentModel {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;

	@Column(nullable = false, name = "fullName")
	private String fullName;

	@Column(nullable = true, name = "age")
	private int age;

	@Column(unique = true, nullable = false, length = 150, name = "email")
	private String email;

	@ManyToOne(targetEntity = ClassesModel.class)
	@JoinColumn(name = "class_id", referencedColumnName = "id")
	private ClassesModel classes;

	public StudentModel() {
		super();
	}

	public StudentModel(int id) {
		super();
		this.id = id;
	}
	public StudentModel(int id, String fullName, int age, String email, ClassesModel
classes) {
		super();
		this.id = id;
		this.fullName = fullName;
		this.age = age;
		this.email = email;
		this.classes = classes;
	}

	public StudentModel(String fullName, int age, String email, ClassesModel classes) {
		super();
		this.fullName = fullName;
		this.age = age;
		this.email = email;
		this.classes = classes;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getFullName() {
		return fullName;
	}

	public void setFullName(String fullName) {
		this.fullName = fullName;
	}

	public int getAge() {
		return age;
	}

	public void setAge(int age) {
		this.age = age;
	}

	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}

	public ClassesModel getClasses() {
		return classes;
	}

	public void setClasses(ClassesModel classes) {
		this.classes = classes;
	}

	@Override
	public String toString() {
		return "StudentModel [id=" + id + ", fullName=" + fullName + ", age=" + age + ",
email=" + email + ", classes="
					+ classes + "]";
	}

}
```

9. List of all the subjects



LIST OF
Subject

Add new Subject

CLASS DETAIL
Class-3

ID	Subject	Time	Teacher	Action
1	Class 3 Math	10:30	Ikaris	Edit Delete
2	Class 3 Sciences	11:30	Sprite	Edit Delete
3	Class 3 Meta Physics	12:30	Thena	Edit Delete
4	Class 3 Quantum Mechanics	01:30	Kingo Sunen	Edit Delete
5	Class 3 Machine Learning	2:30	Ikaris	Edit Delete

SubjectController.java

```
/**
 * List all the Subject
 *
 * @param request
 * @param response
 * @throws SQLException
 * @throws IOException
 * @throws ServletException
 */

private void listSubject(HttpServletRequest request, HttpServletResponse response)
                throws SQLException, IOException, ServletException {

        int class_id = Integer.parseInt(request.getParameter("class_id"));
        List<SubjectModel> listSubject = SubjectDao.getAllSubjects(class_id);
        ClassesModel subjectClasses = ClassesDao.getClasses(class_id);

        request.setAttribute("listSubject", listSubject);
        request.setAttribute("classes", subjectClasses);

        RequestDispatcher dispatcher = request.getRequestDispatcher("/subject/index.jsp");
        dispatcher.forward(request, response);

}
```

SubjectDao.java

```
/**
 * Get all Users
 * @return
 */
@SuppressWarnings("unchecked")
public static List<SubjectModel> getAllSubjects(int class_id) {

  Transaction transaction = null;
  List<SubjectModel> listOfSubjects = null;
  Session session = HibernateUtil.getSessionFactory().openSession();

  try {

    // start a transaction
```

```
        transaction = session.beginTransaction();
        listOfSubjects = session.createQuery("FROM subjects WHERE class_id = " + class_id).getResultList();
        transaction.commit();

    } catch (Exception e) {
          transaction.rollback();
        e.printStackTrace();
    }

    return listOfSubjects;

  }
```

SubjectModel.java

```
package learnersadmin.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity(name = "subjects")
@Table(name = "subjects")
public class SubjectModel {

        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private int id;

        @Column(unique = true, nullable = false, length = 150)
        private String subject;

        @Column(nullable = false, length = 8, name = "subjectTime")
        private String subjectTime;

        @ManyToOne(targetEntity = ClassesModel.class)
        @JoinColumn(name = "class_id", referencedColumnName = "id", updatable = false, nullable
= false)
```

```java
	private ClassesModel classes;

	@ManyToOne(targetEntity = TeacherModel.class)
	@JoinColumn(name = "teacher_id", referencedColumnName = "id", updatable = false,
nullable = false)
	private TeacherModel teacher;

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getSubject() {
		return subject;
	}

	public void setSubject(String subject) {
		this.subject = subject;
	}

	public String getSubjectTime() {
		return subjectTime;
	}

	public void setSubjectTime(String subjectTime) {
		this.subjectTime = subjectTime;
	}

	public ClassesModel getClasses() {
		return classes;
	}

	public void setClasses(ClassesModel classes) {
		this.classes = classes;
	}

	public TeacherModel getTeacher() {
		return teacher;
	}

	public void setTeacher(TeacherModel teacher) {
		this.teacher = teacher;
```

```
        }

        @Override
        public String toString() {
                return "SubjectModel [id=" + id + ", subject=" + subject + " time =" + subjectTime + ", ]";
        }

}
```

10. Publish your code to Github repository

Step 1

Open gitbash navigate to the project folder

CD <folder_path>

Step 2

Initialized git

git init

Add all the file to to repo by

git add .

Commit added files

git commit -m “<message>”

Add remote github link

git remote add origin <gitlink>

Push on git repo

git push -u origin main