Developing a Backend Admin for Learner’s Academy

Developer: Kamar Zaghloul

1. Info

- FSD 2021 (Vodafone)
- **Developer**: Kamar Zaghloul
- **GitHub Repo**:

  https://github.com/kamarz01/Simplilearn_Phase2_Learners_Acadmy_Admin_Backend

2. Introduction

Project objective:

As a Full Stack Developer, design and develop a backend administrative portal for the Learner’s Academy. Use the GitHub repository to manage the project artifacts.

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.

3. Project Planning

- The project is to be completed in two sprints.
- The resources available for this project is (2):
  1 Developer & 1 Tester.

Sprint (1) will include:

- Create application design.
- Create project skeleton.
- Setup project configurations.
- Create database schema.
- Design project UI screens.
- Implement Classes CRUD feature.
- Implement Subjects CRUD feature.

- Implement Teachers CRUD feature.
- Implement Student CRUD feature.

Sprint (2) will include:

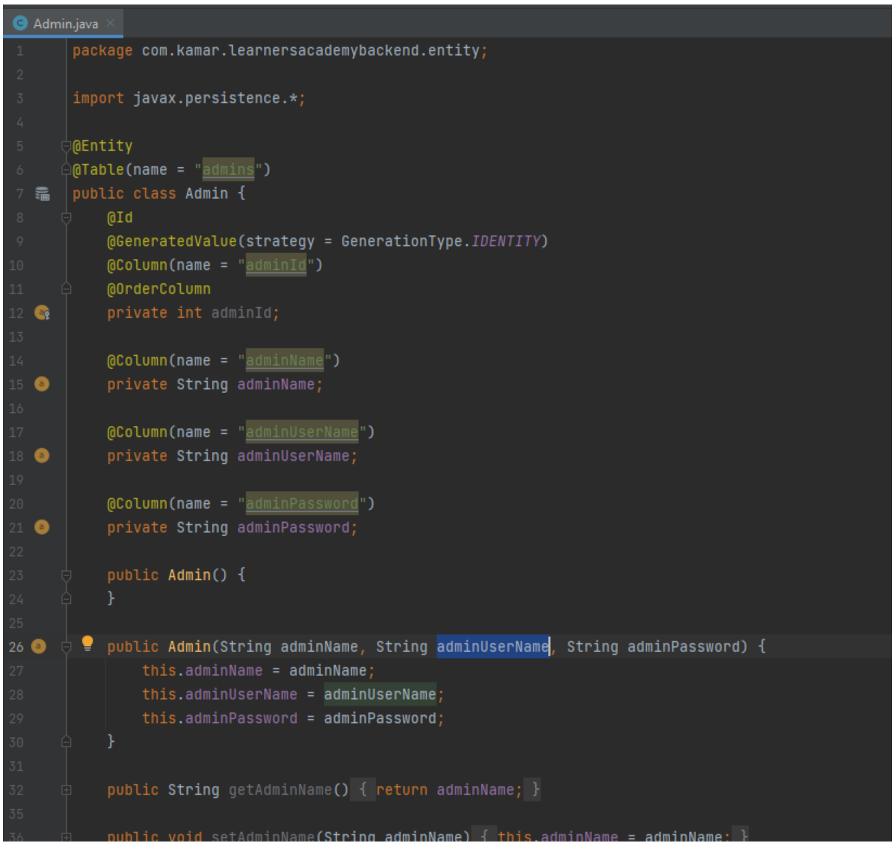
- Implement Assign subject to class feature.
- Implement Assign student to class feature.
- Implement Assign teacher to class feature.
- Implement Assign teacher to subjects feature.
- Create class report design.
- Implement Class Report feature.
- Creating documentations.
- Pushing project to GitHub.

4. Technologies & Concepts used in Project

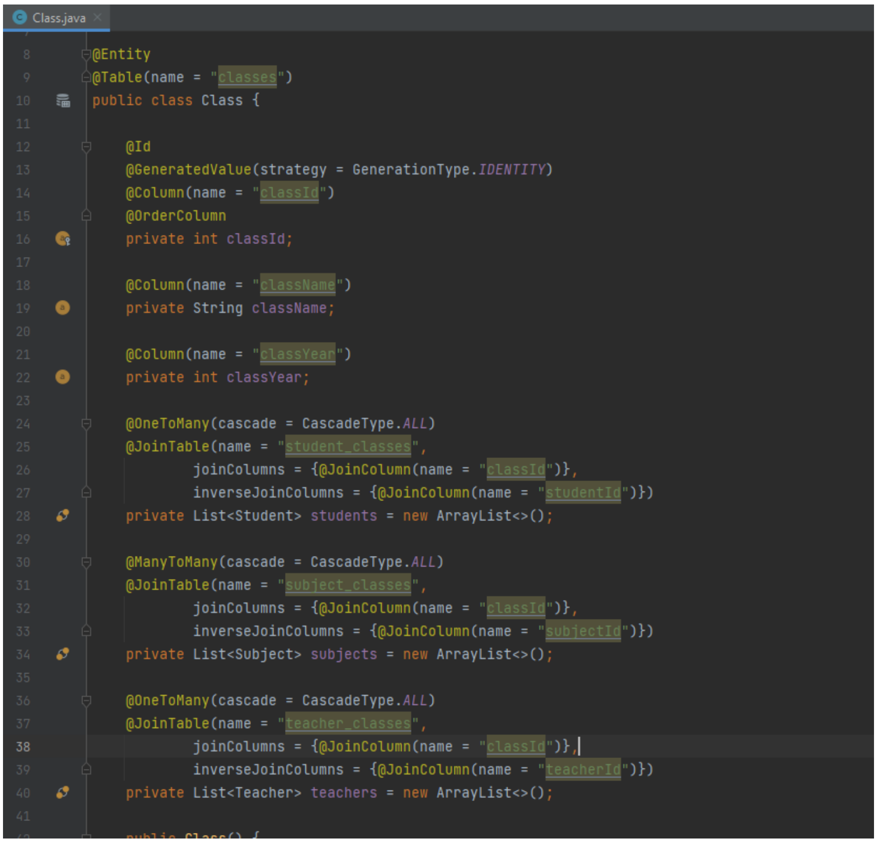
- Java EE
- Servlets
- JSP
- Hibernate
- MySql
- HTML5
- CSS3
- JSTL
- JSP EL
- Listeners
- Exception Handling
- OOP
- Collections
- Streams
- Data Structures

and more...

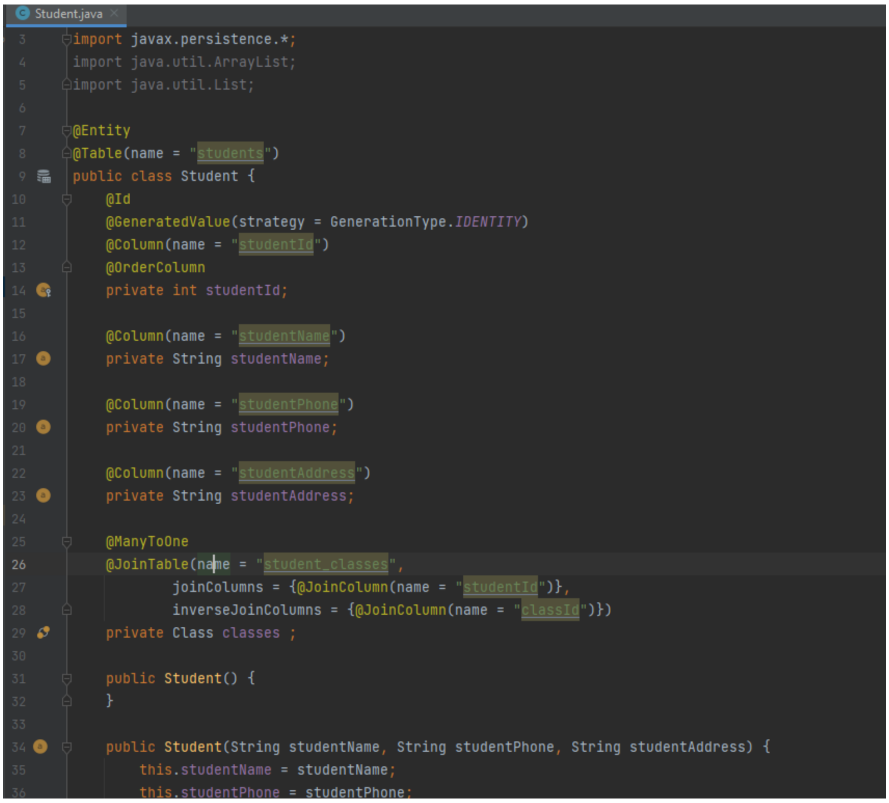
5. Application Entities



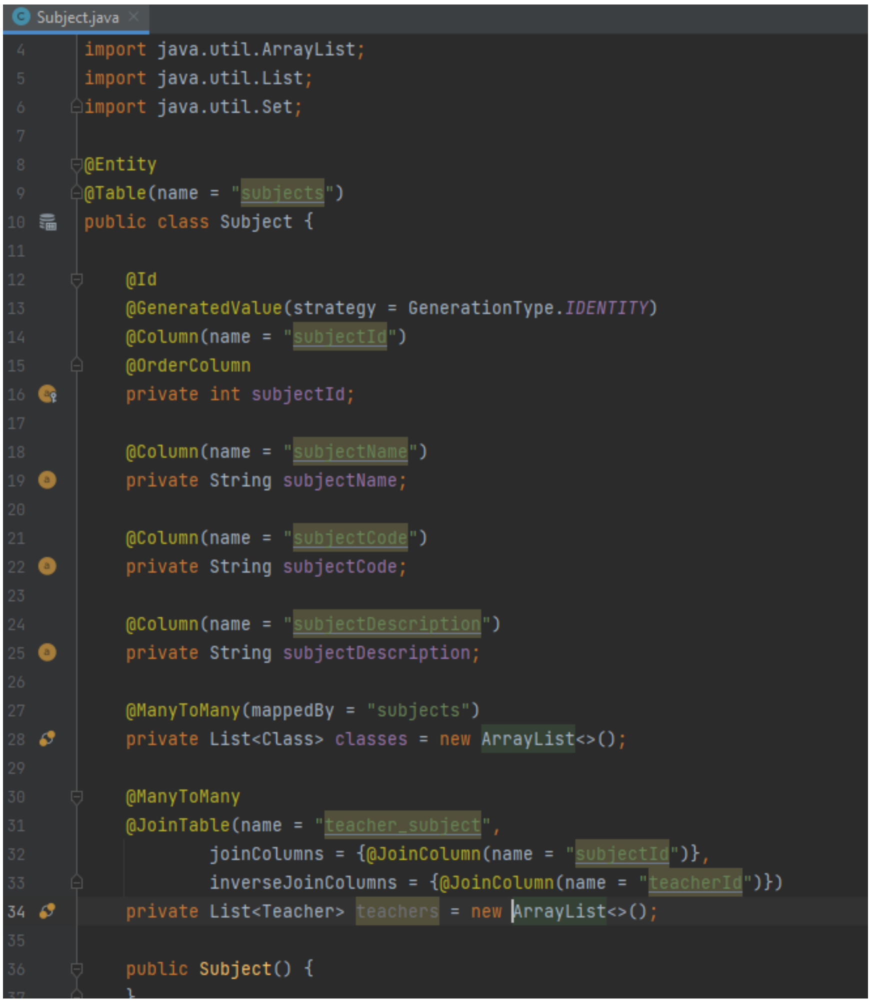
```java
package com.kamar.learnersacademybackend.entity;

import javax.persistence.*;

@Entity
@Table(name = "admins")
public class Admin {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "adminId")
    @OrderColumn
    private int adminId;

    @Column(name = "adminName")
    private String adminName;

    @Column(name = "adminUserName")
    private String adminUserName;

    @Column(name = "adminPassword")
    private String adminPassword;

    public Admin() {
    }

    public Admin(String adminName, String adminUserName, String adminPassword) {
        this.adminName = adminName;
        this.adminUserName = adminUserName;
        this.adminPassword = adminPassword;
    }

    public String getAdminName() { return adminName; }

    public void setAdminName(String adminName) { this.adminName = adminName; }
```



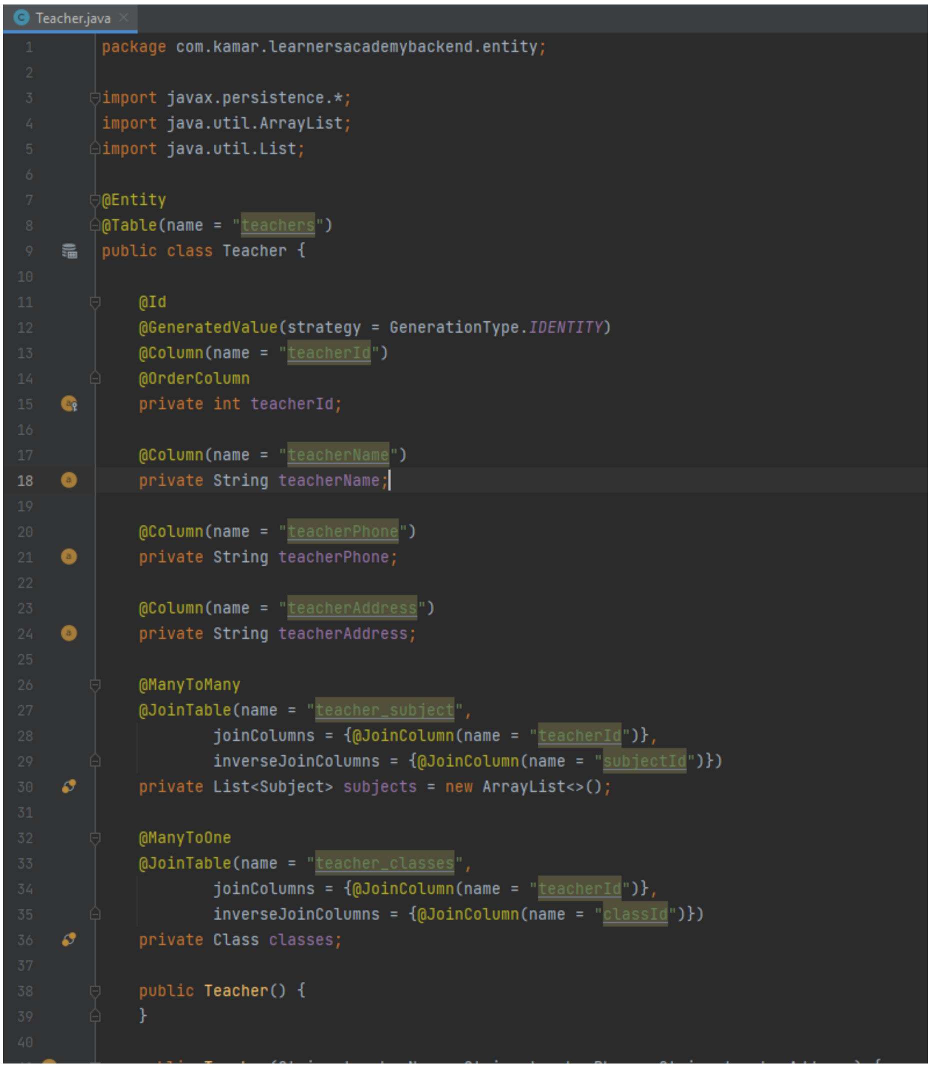
```java
@Entity
@Table(name = "classes")
public class Class {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "classId")
    @OrderColumn
    private int classId;

    @Column(name = "className")
    private String className;

    @Column(name = "classYear")
    private int classYear;

    @OneToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "student_classes",
            joinColumns = {@JoinColumn(name = "classId")},
            inverseJoinColumns = {@JoinColumn(name = "studentId")})
    private List<Student> students = new ArrayList<>();

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "subject_classes",
            joinColumns = {@JoinColumn(name = "classId")},
            inverseJoinColumns = {@JoinColumn(name = "subjectId")})
    private List<Subject> subjects = new ArrayList<>();

    @OneToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "teacher_classes",
            joinColumns = {@JoinColumn(name = "classId")},
            inverseJoinColumns = {@JoinColumn(name = "teacherId")})
    private List<Teacher> teachers = new ArrayList<>();

    public Class() {
```



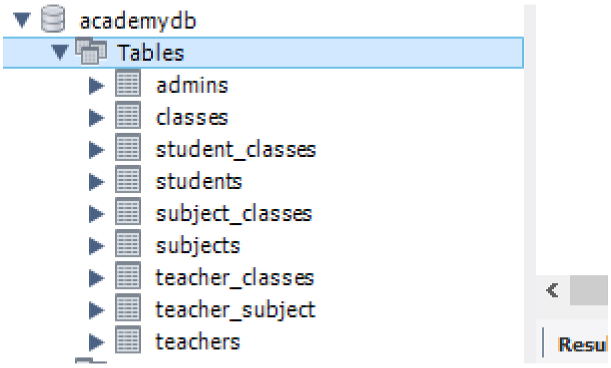
```java
import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
@Table(name = "students")
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "studentId")
    @OrderColumn
    private int studentId;

    @Column(name = "studentName")
    private String studentName;

    @Column(name = "studentPhone")
    private String studentPhone;

    @Column(name = "studentAddress")
    private String studentAddress;

    @ManyToOne
    @JoinTable(name = "student_classes",
            joinColumns = {@JoinColumn(name = "studentId")},
            inverseJoinColumns = {@JoinColumn(name = "classId")})
    private Class classes ;

    public Student() {
    }

    public Student(String studentName, String studentPhone, String studentAddress) {
        this.studentName = studentName;
        this.studentPhone = studentPhone;
```



```java
import java.util.ArrayList;
import java.util.List;
import java.util.Set;

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

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "subjectId")
    @OrderColumn
    private int subjectId;

    @Column(name = "subjectName")
    private String subjectName;

    @Column(name = "subjectCode")
    private String subjectCode;

    @Column(name = "subjectDescription")
    private String subjectDescription;

    @ManyToMany(mappedBy = "subjects")
    private List<Class> classes = new ArrayList<>();

    @ManyToMany
    @JoinTable(name = "teacher_subject",
            joinColumns = {@JoinColumn(name = "subjectId")},
            inverseJoinColumns = {@JoinColumn(name = "teacherId")})
    private List<Teacher> teachers = new ArrayList<>();

    public Subject() {
    }
```



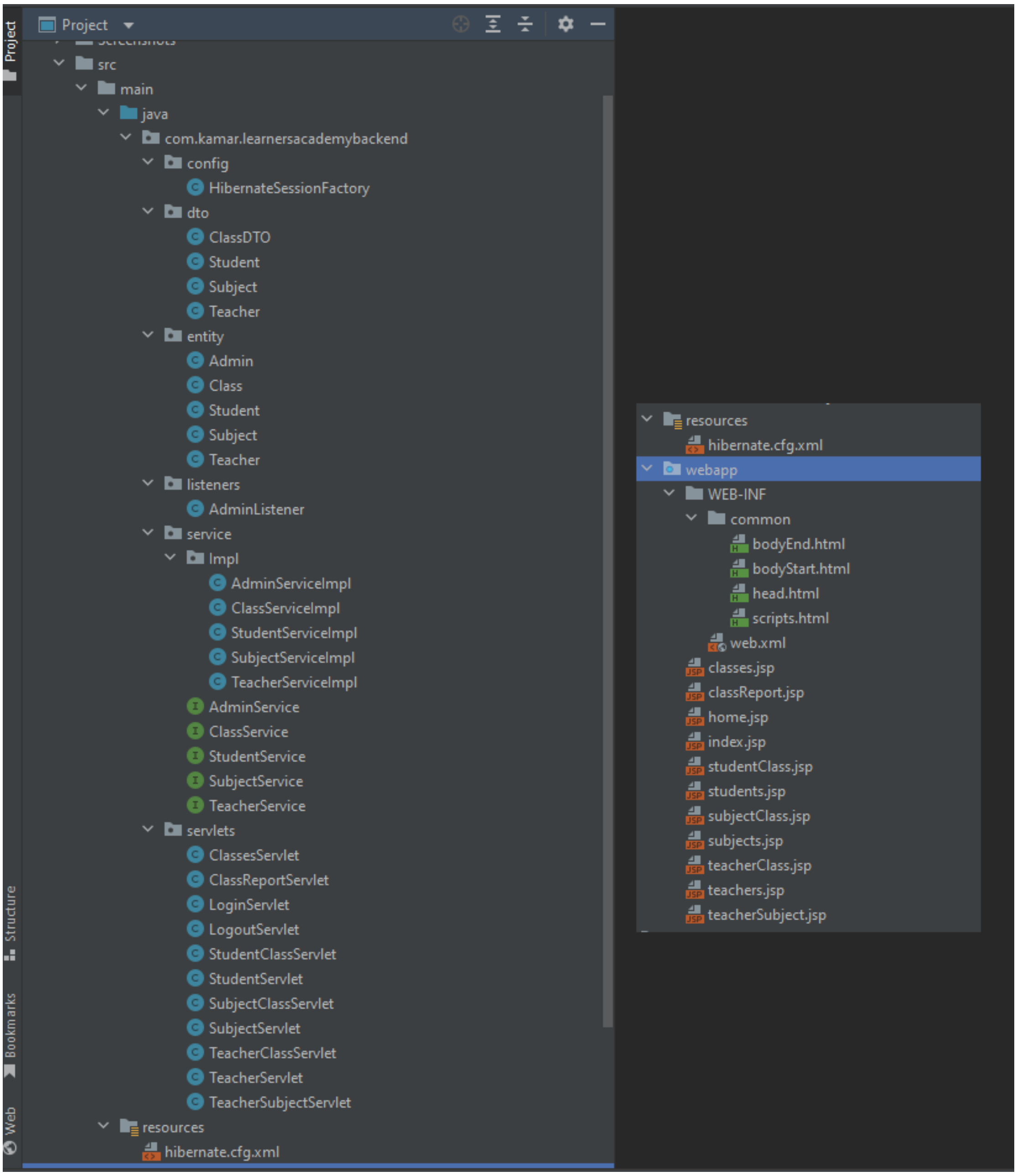
```java
package com.kamar.learnersacademybackend.entity;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

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

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "teacherId")
    @OrderColumn
    private int teacherId;

    @Column(name = "teacherName")
    private String teacherName;

    @Column(name = "teacherPhone")
    private String teacherPhone;

    @Column(name = "teacherAddress")
    private String teacherAddress;

    @ManyToMany
    @JoinTable(name = "teacher_subject",
            joinColumns = {@JoinColumn(name = "teacherId")},
            inverseJoinColumns = {@JoinColumn(name = "subjectId")})
    private List<Subject> subjects = new ArrayList<>();

    @ManyToOne
    @JoinTable(name = "teacher_classes",
            joinColumns = {@JoinColumn(name = "teacherId")},
            inverseJoinColumns = {@JoinColumn(name = "classId")})
    private Class classes;

    public Teacher() {
    }

```



```
academydb
  Tables
    admins
    classes
    student_classes
    students
    subject_classes
    subjects
    teacher_classes
    teacher_subject
    teachers
```

6. Project Structure

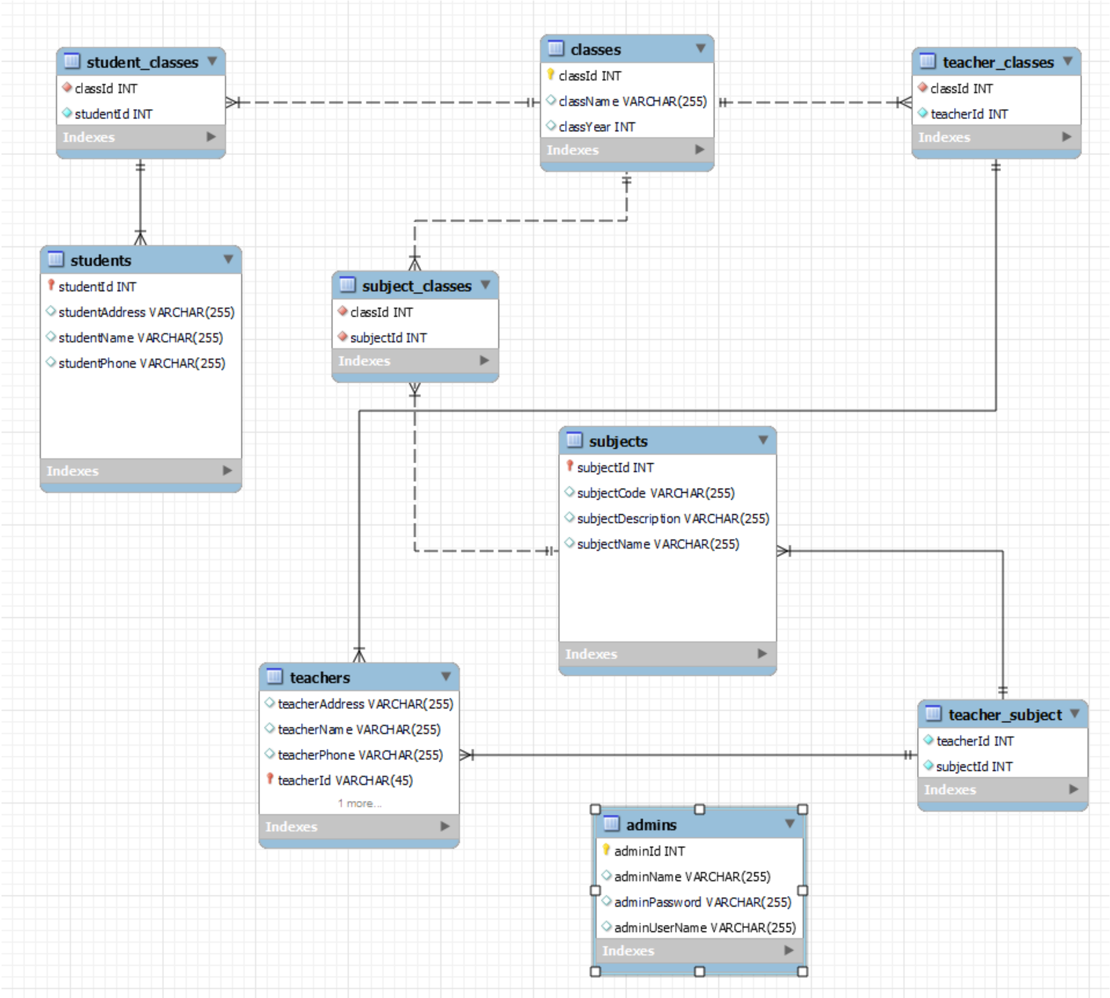
The project was been created following a good design structure to separate the logic from the user interface.



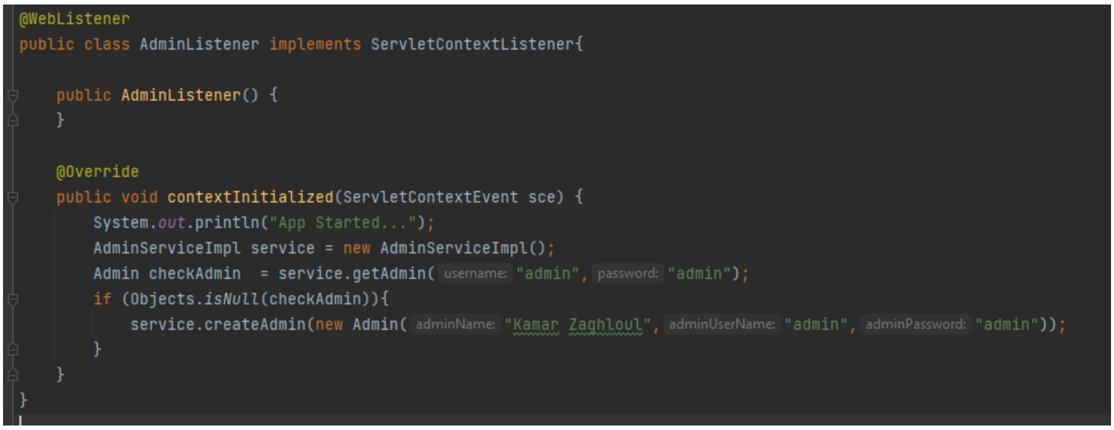
```
Project
  > screenshots
  v src
    v main
      v java
        v com.kamar.learnersacademybackend
          v config
              HibernateSessionFactory
          v dto
              ClassDTO
              Student
              Subject
              Teacher
          v entity
              Admin
              Class
              Student
              Subject
              Teacher
          v listeners
              AdminListener
          v service
            v Impl
                AdminServiceImpl
                ClassServiceImpl
                StudentServiceImpl
                SubjectServiceImpl
                TeacherServiceImpl
              AdminService
              ClassService
              StudentService
              SubjectService
              TeacherService
          v servlets
              ClassesServlet
              ClassReportServlet
              LoginServlet
              LogoutServlet
              StudentClassServlet
              StudentServlet
              SubjectClassServlet
              SubjectServlet
              TeacherClassServlet
              TeacherServlet
              TeacherSubjectServlet
      v resources
          hibernate.cfg.xml
```

```
v resources
    hibernate.cfg.xml
v webapp
  v WEB-INF
    v common
        bodyEnd.html
        bodyStart.html
        head.html
        scripts.html
      web.xml
    classes.jsp
    classReport.jsp
    home.jsp
    index.jsp
    studentClass.jsp
    students.jsp
    subjectClass.jsp
    subjects.jsp
    teacherClass.jsp
    teachers.jsp
    teacherSubject.jsp
```

7. ERD

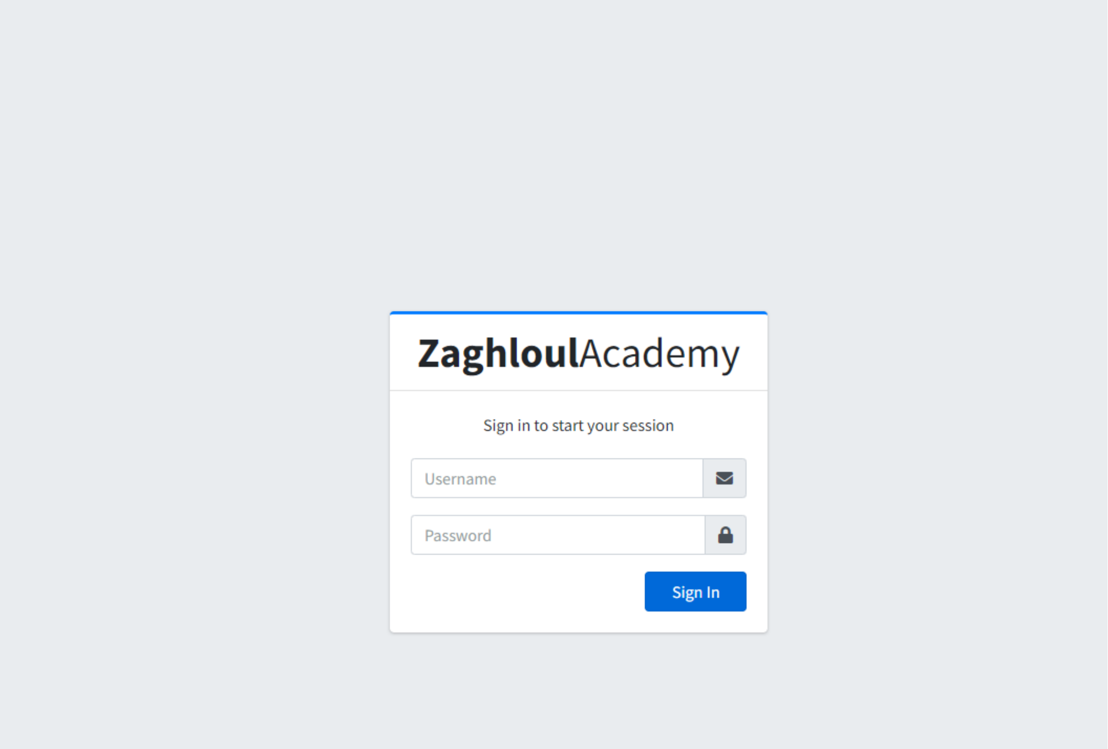


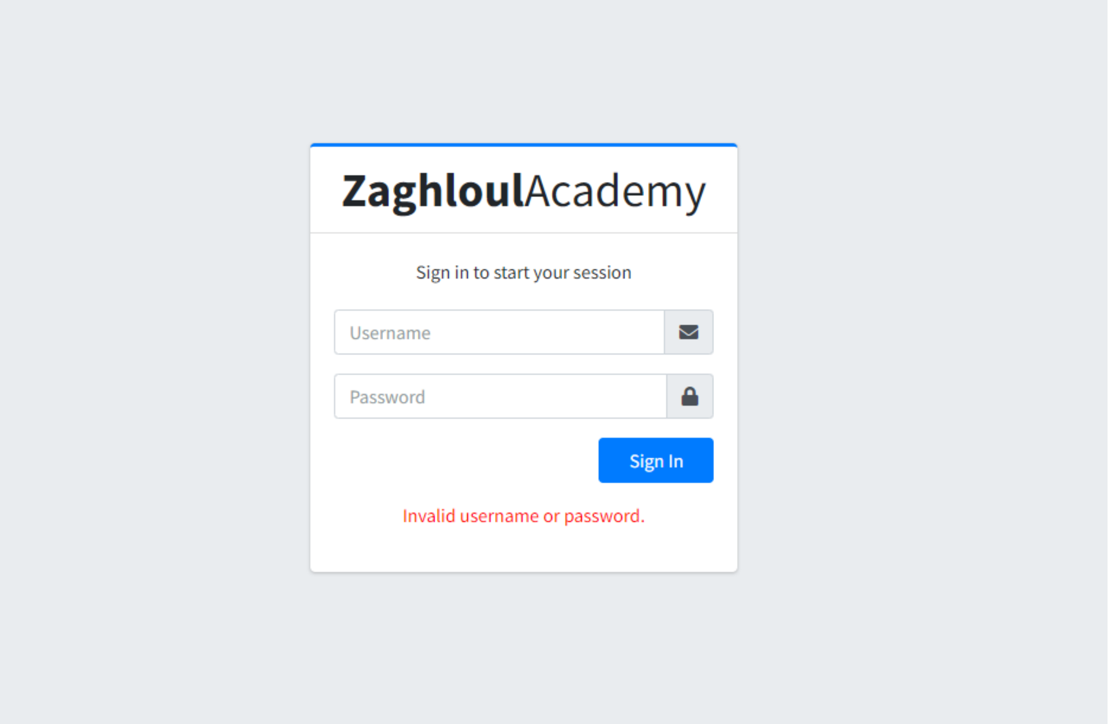
8. Create Admin user If does not exist



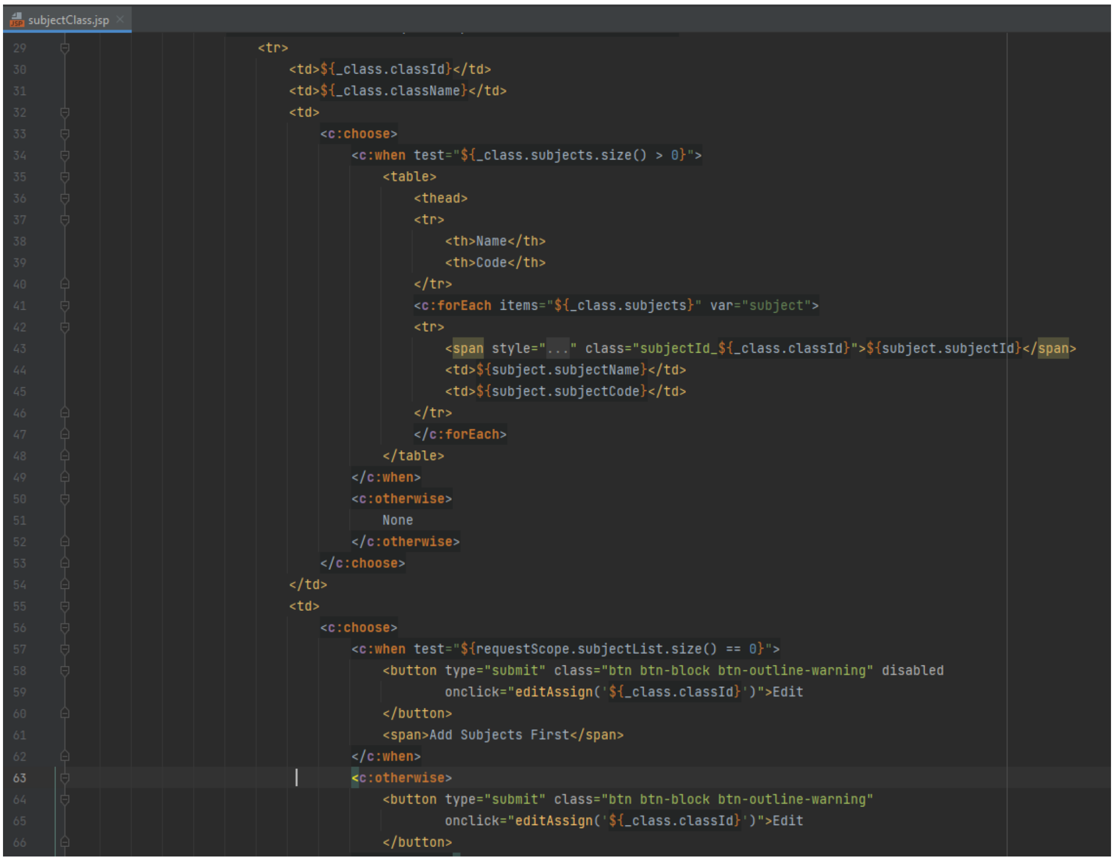
```
@WebListener
public class AdminListener implements ServletContextListener{

    public AdminListener() {
    }

    @Override
    public void contextInitialized(ServletContextEvent sce) {
        System.out.println("App Started...");
        AdminServiceImpl service = new AdminServiceImpl();
        Admin checkAdmin  = service.getAdmin( username: "admin", password: "admin");
        if (Objects.isNull(checkAdmin)){
            service.createAdmin(new Admin( adminName: "Kamar Zaghloul", adminUserName: "admin", adminPassword: "admin"));
        }
    }
}
```

9. Application Demo

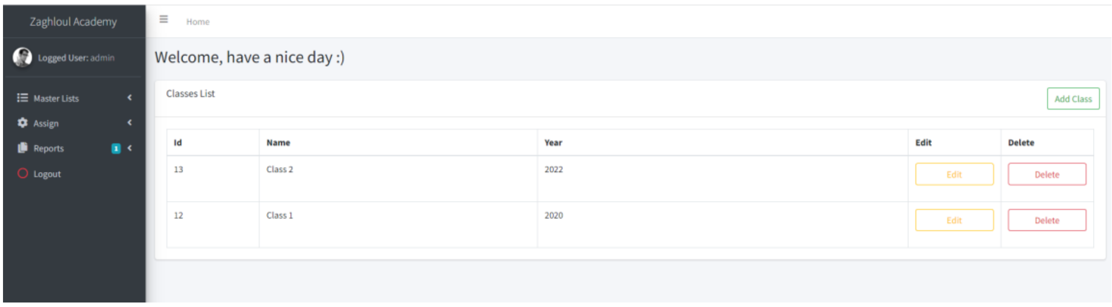




Invalid username or password.



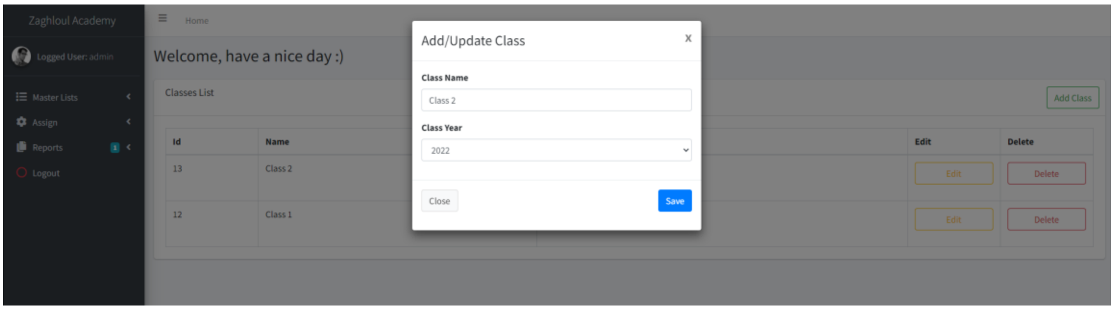
```
<tr>
    <td>${_class.classId}</td>
    <td>${_class.className}</td>
    <td>
        <c:choose>
            <c:when test="${_class.subjects.size() > 0}">
                <table>
                    <thead>
                    <tr>
                        <th>Name</th>
                        <th>Code</th>
                    </tr>
                    <c:forEach items="${_class.subjects}" var="subject">
                    <tr>
                        <span style="..." class="subjectId_${_class.classId}">${subject.subjectId}</span>
                        <td>${subject.subjectName}</td>
                        <td>${subject.subjectCode}</td>
                    </tr>
                    </c:forEach>
                </table>
            </c:when>
            <c:otherwise>
                None
            </c:otherwise>
        </c:choose>
    </td>
    <td>
        <c:choose>
            <c:when test="${requestScope.subjectList.size() == 0}">
                <button type="submit" class="btn btn-block btn-outline-warning" disabled
                        onclick="editAssign('${_class.classId}')">Edit
                </button>
                <span>Add Subjects First</span>
            </c:when>
            <c:otherwise>
                <button type="submit" class="btn btn-block btn-outline-warning"
                        onclick="editAssign('${_class.classId}')">Edit
                </button>
```



Welcome, have a nice day :)

Classes List

Id	Name	Year	Edit	Delete
13	Class 2	2022	Edit	Delete
12	Class 1	2020	Edit	Delete



Welcome, have a nice day :)

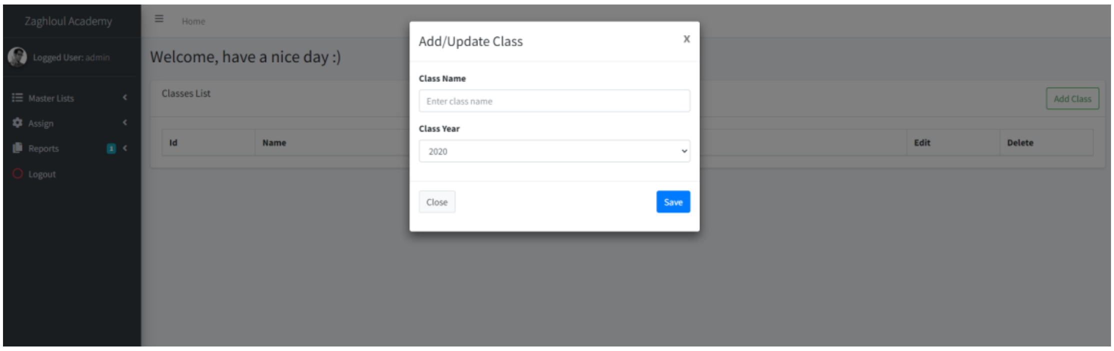
Classes List

Add/Update Class

Class Name: Class 2

Class Year: 2022

Id	Name	Edit	Delete
13	Class 2	Edit	Delete
12	Class 1	Edit	Delete



Welcome, have a nice day :)

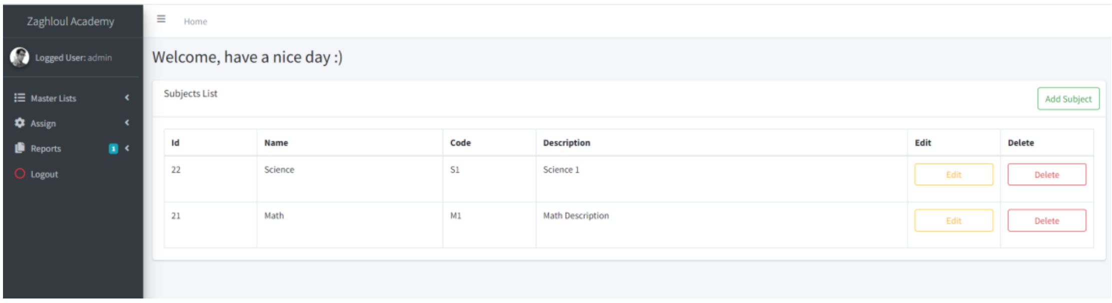
Classes List

Id	Name	Edit	Delete

Add/Update Class

Class Name: Enter class name

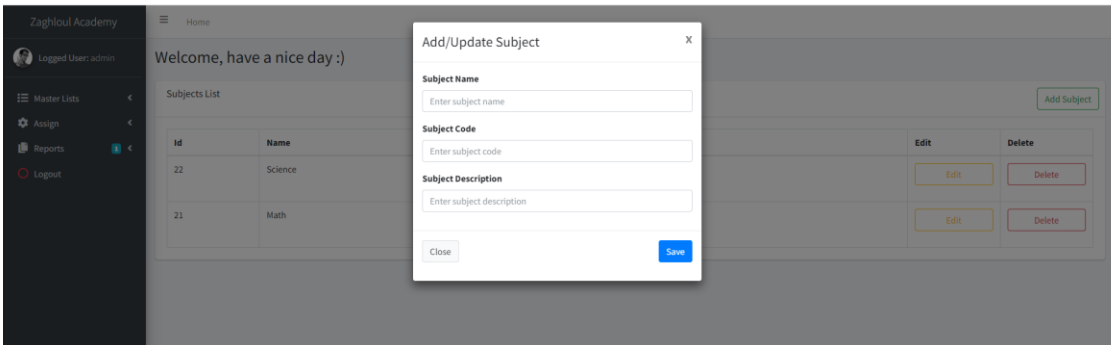
Class Year: 2020



Welcome, have a nice day :)

Subjects List

Id	Name	Code	Description	Edit	Delete
22	Science	S1	Science 1	Edit	Delete
21	Math	M1	Math Description	Edit	Delete



Welcome, have a nice day :)

Subjects List

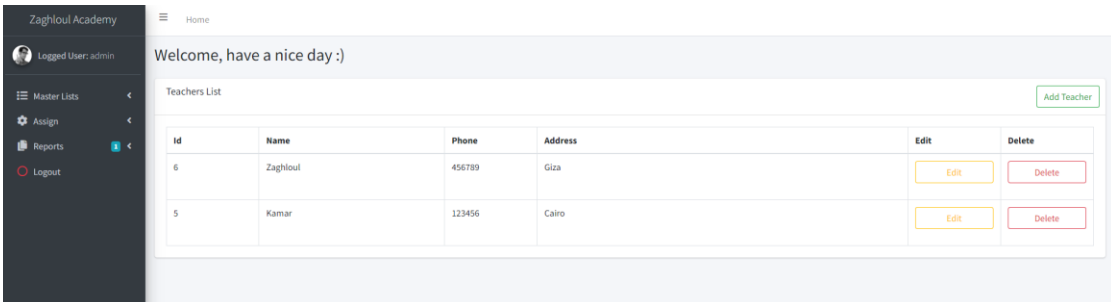
Id	Name	Edit	Delete
22	Science	Edit	Delete
21	Math	Edit	Delete

Add/Update Subject

Subject Name: Enter subject name

Subject Code: Enter subject code

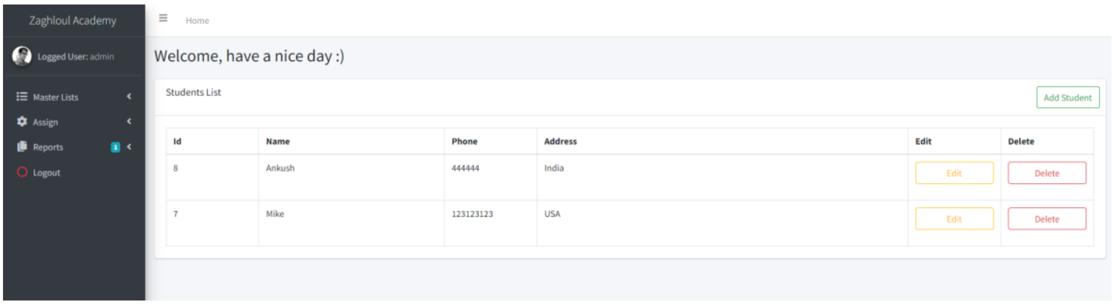
Subject Description: Enter subject description



Welcome, have a nice day :)

Teachers List

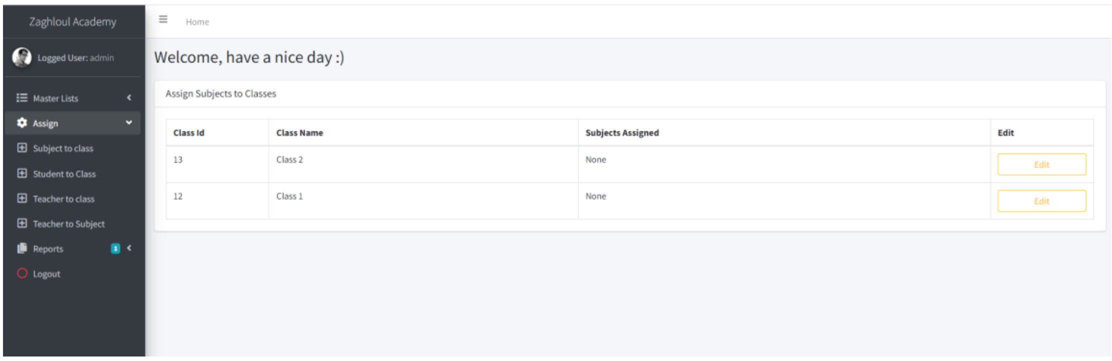
Id	Name	Phone	Address	Edit	Delete
6	Zaghloul	456789	Giza	Edit	Delete
5	Kamar	123456	Cairo	Edit	Delete



Welcome, have a nice day :)

Students List

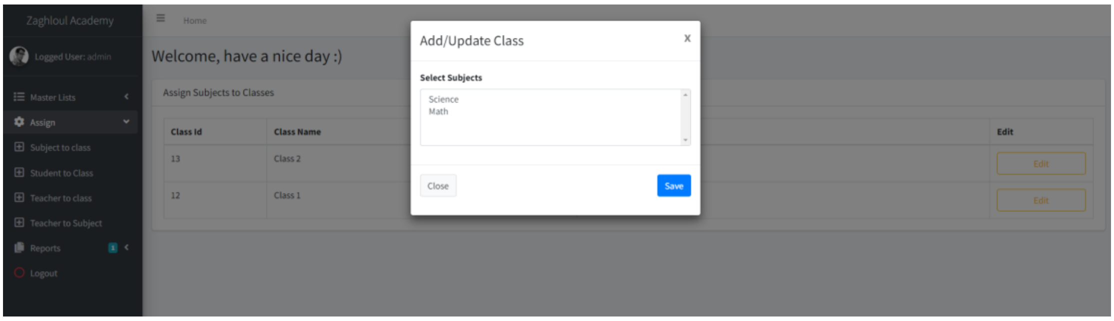
Id	Name	Phone	Address	Edit	Delete
8	Ankush	444444	India	Edit	Delete
7	Mike	123123123	USA	Edit	Delete



Welcome, have a nice day :)

Assign Subjects to Classes

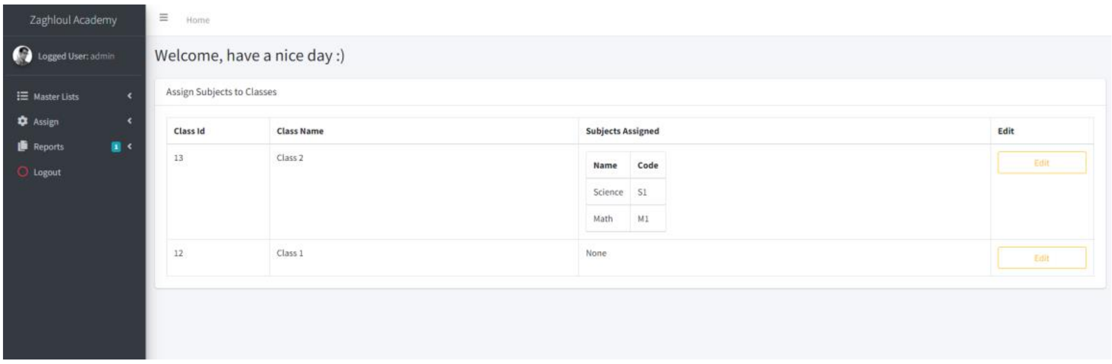
Class Id	Class Name	Subjects Assigned	Edit
13	Class 2	None	Edit
12	Class 1	None	Edit



Add/Update Class

Select Subjects

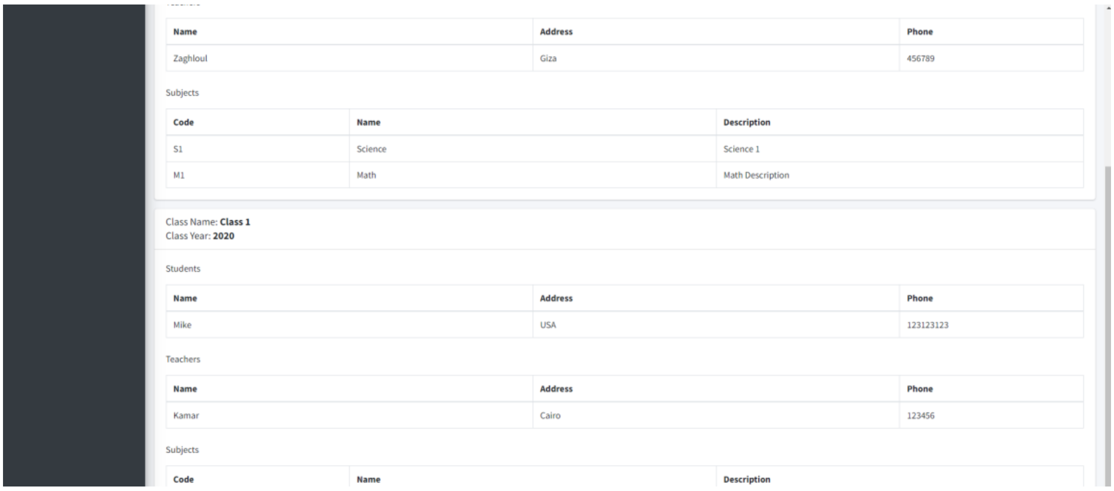
Science
Math



Welcome, have a nice day :)

Assign Subjects to Classes

Class Id	Class Name	Subjects Assigned	Edit
13	Class 2	Name / Code Science / S1 Math / M1	Edit
12	Class 1	None	Edit



Name	Address	Phone
Zaghloul	Giza	456789

Subjects

Code	Name	Description
S1	Science	Science 1
M1	Math	Math Description

Class Name: **Class 1**
Class Year: **2020**

Students

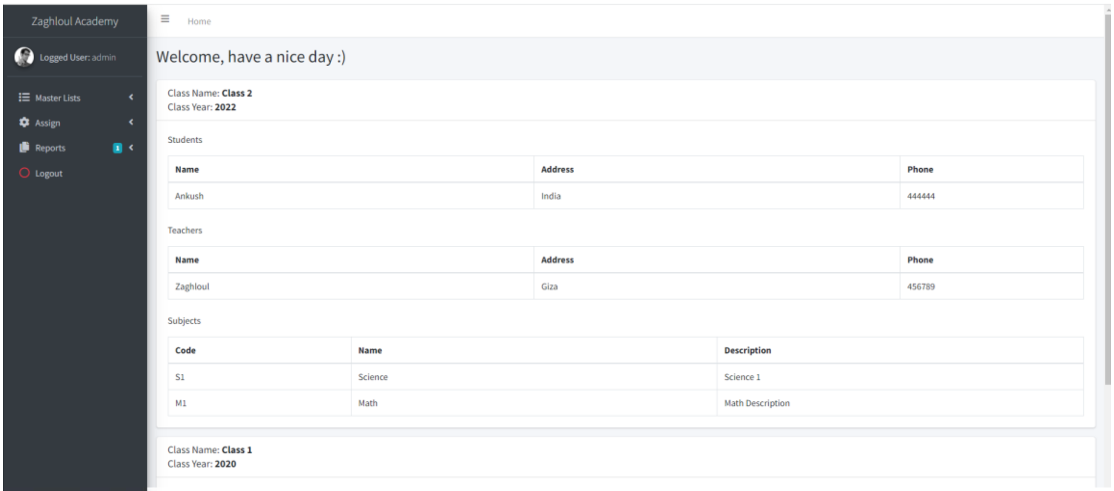
Name	Address	Phone
Mike	USA	123123123

Teachers

Name	Address	Phone
Kamar	Cairo	123456

Subjects

Code	Name	Description



Welcome, have a nice day :)

Class Name: **Class 2**
Class Year: **2022**

Students

Name	Address	Phone
Ankush	India	444444

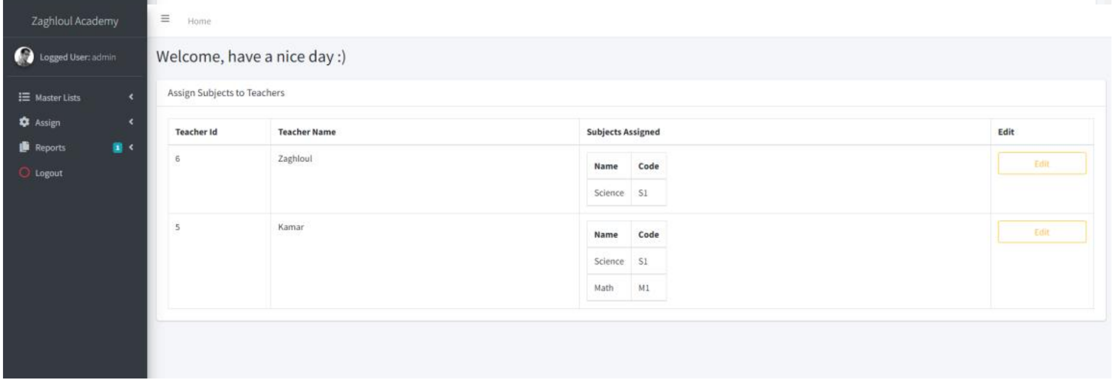
Teachers

Name	Address	Phone
Zaghloul	Giza	456789

Subjects

Code	Name	Description
S1	Science	Science 1
M1	Math	Math Description

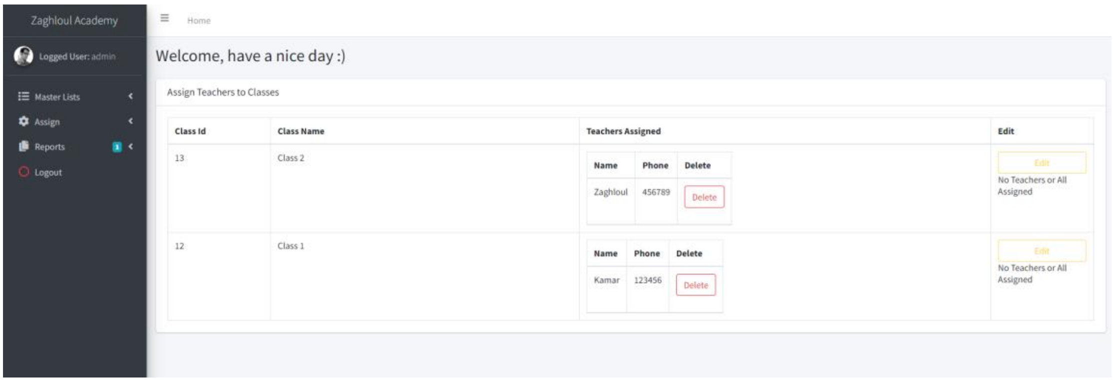
Class Name: **Class 1**
Class Year: **2020**



Welcome, have a nice day :)

Assign Subjects to Teachers

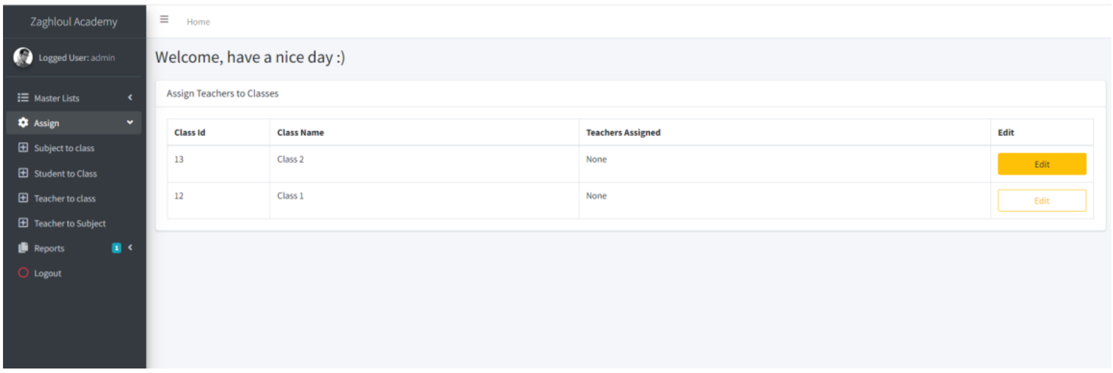
Teacher Id	Teacher Name	Subjects Assigned	Edit
6	Zaghloul	Name Code Science S1	Edit
5	Kamar	Name Code Science S1 Math M1	Edit



Welcome, have a nice day :)

Assign Teachers to Classes

Class Id	Class Name	Teachers Assigned	Edit
13	Class 2	Name Phone Delete Zaghloul 456789 Delete	Edit No Teachers or All Assigned
12	Class 1	Name Phone Delete Kamar 123456 Delete	Edit No Teachers or All Assigned



Welcome, have a nice day :)

Assign Teachers to Classes

Class Id	Class Name	Teachers Assigned	Edit
13	Class 2	None	Edit
12	Class 1	None	Edit

Rest of screenshots can be found here:

https://github.com/kamarz01/Simplilearn_Phase2_Learners_Acadmy_Admin_Backend/Screenshots

10.Github Repo

https://github.com/kamarz01/Simplilearn_Phase2_Learners_Acadmy_Admin_Backend