# Become a Back-End Expert

# Learner’s Academy Admin Portal Project

Report content:

* Basic Info
* Specification document - Product’s capabilities, appearance, and user interactions
* Project objective
* Sprite planning
* Java concepts in use
* App workflow chart
* App sample snips
* Algorithms with comments
* Unique selling points
* Conclusion

## Basic Info

App name: Learner’s Academy Admin Portal

Developer: Drazen Drinic

GitHub: <https://github.com/Drazen-TheGRC/SimpliLearn_BackEnd_Project>

## Specification document - Product’s capabilities, appearance, and user interactions

This is a web application that gives many options to the user. Since it is a web app all interaction is done through the browser. User can register an Admin and then Login into the portal. Admins can be listed, edited and deleted.

Lerner’s Academy Admin Portal admin can manager students, teachers, subjects and classes. All of those can be created, edited and deleted by the Admin.

The look of the app is very simple and designed with only few lines of CSS since the goal here was the backend not the frontend.

Interactions is very simple and straight forward, each section of the page has a title that explains what is it about and the interaction is done by mouse and keyboard with minimal user imputs.

## Project Objectives:

### 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.

### The administrator can:

* Set up a master list of all the subjects for all the classes
* Set up a master list of all the teachers
* Set up a master list of all the classes
* Assign classes for subjects from the master list
* Assign teachers to a class for a subject (A teacher can be assigned to different classes for different subjects)
* Get a master list of students (Each student must be assigned to a single class)

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

The goal of the company is to deliver a high-end quality product as early as possible.

### The flow and features of the application:

* Plan more than two sprints to complete the application
* Document the flow of the application and prepare a flow chart
* List the core concepts and algorithms being used to complete this application
* Implement the appropriate concepts, such as exceptions, collections, and sorting techniques for source code optimization and increased performance

### You must use the following:

* Eclipse/IntelliJ: An IDE to code for the application
* Java: A programming language to develop the web pages, databases, and others
* SQL: To create tables for admin, classes, students, and other specifics
* Git: To connect and push files from the local system to GitHub
* GitHub: To store the application code and track its versions
* Scrum: An efficient agile framework to deliver the product incrementally
* Search and Sort techniques: Data structures used for the project
* Specification document: Any open-source document or Google Docs

### The following requirements should be met:

* The source code should be pushed to your GitHub repository. You need to document the steps and write the algorithms in it.
* The submission of your GitHub repository link is mandatory. In order to track your task, you need to share the link of the repository. You can add a section in your document.
* Document the process step-by-step starting from sprint planning to the product release.
* The application should not close, exit, or throw an exception if the user specifies an invalid input.
* You need to submit the final specification document which will include:
* Project and developer details
* Sprints planned and the tasks achieved in them
* Algorithms and flowcharts of the application
* Core concepts used in the project
* Links to the GitHub repository to verify the project completion

## Sprint Planning

There are eight sprites each taking 3-7 days. The goal here is to have a working web application which will communicate with the database and enable Admin options to an admin of a Learners Academy including adding, editing, listing and deleting students, teachers, subjects and classes. Admin can also manage admins inside the portal.

### Sprint 1:

Step Zero  
Refresh MySQL and Eclipse.

For the purpose of this project I am going to reinstall the MySQL and Eclipse so I can have a fresh start.

1. Fresh MySQL & Eclipse
2. Download MySQL Installer and use it to uninstall everything
3. In File Explorer use View and mark Hidden items and delete all MySQL related folders
   1. C:\Program Files\MySQL
   2. C:\Program Files (x86)\MySQL
   3. C:\ProgramData\MySQL
   4. C:\Users\<your-username>\AppData\Roaming\MySQL
4. Install fresh MySQL, <https://downloads.mysql.com/archives/installer/>
   1. I chose 8.0.13
5. My user and password for MySQL
   1. user: root   
      password: password
6. Install fresh Eclipse
   1. First uninstall
      1. Delete eclipse folder C:\Users\YourUserName\eclipse
      2. Delete eclipse cache C:\Users\YourUserName\.p2
      3. Delete shortcuts
      4. Delete eclipse workspace (if you want to delete all of your projects)
7. Than Install
   1. Download <https://www.eclipse.org/downloads/>
   2. Install now and chose Eclipse IDE for Enterprise Java and Web Developers

Step one

Create Dynamic Web Project inside eclipse-workspace,

* + create git repo for this project (RC on the project > Team > Share Project... >
  + check Use or create repository in parent folder of the project >
  + click on the project/select project >
  + Create Repository >
  + Finish),
  + refresh project,
  + delete .gitignore,
  + do initial commit
  + or
  + Creating project GitHub repo,
  + clone GitHub repo into eclipse-workspace,
  + create Dynamic Web Project inside cloned GitHub repo folder inside eclipse-workspace,
  + add files into staging area and do an initial commit.
  + or simply just create your eclipse project and use GitHub Desktop to take care of your VCS

Step two

Add essential documents into the project, add dependencies and build path for necessary JARs:

1. antlr-2.7.7
2. byte-buddy-1.8.17
3. classmate-1.3.4
4. dom4j-2.1.1
5. hibernate-commons-annotations-5.0.4.Fina
6. hibernate-core-5.3.7.Final
7. jandex-2.0.5.Final
8. javassist-3.23.1-GA
9. javax.activation-api-1.2.0
10. javax.persistence-api-2.
11. jaxb-api-2.3.1
12. jboss-logging-3.3.2.Final
13. jboss-transaction-api\_1.2\_spec-1.1.1.Fina
14. jsp-api-2.2
15. jstl-1.2
16. mysql-connector-java-8.0.13
17. protobuf-java-3.6.1
18. servlet-api (From apache-tomcat-8.5.78 lib folder)

### Sprint 2:

In the second sprite we will create project structure, Hibernateutil, and prepare the database using HeidiSQL. Afterwards we are going to create POJOs and DAOs with all the functionality for the ADMIN so we can work and make CRUD for ADMIN and a Login Page.

To be able to make it all work we will need to create SERVLETs for START app and another one to manage Learners Academy flow.

### Sprint 3:

In the third sprite we will create all the functionality and test it so we can make sure that the Admin options are working fine.

Inside POJOs we will have fields, getters, setters and few options for constructors. For DAOs we will make methods to validateAdmin(), saveAdmin(), updateAdmin(), deleteAdmin(), getAdmin(), getAllAdmin() in the case of Admin for other classes we will have same methods.

In SERVLETs we will have two sets: In the START SERVLET we will have a method to create an initial Admin registration and there we will also manage initial instances to fill the database.

In the Learners Academy SERVLET we will create methods which are going to first of all validate an admin when registering it. Afterwards we need methods to Login and to logout. And at the end we need methods to show register page, to register an instance, to list instances, to show edit form, to confirm edits and to delete an admin. Same methods are needed for all the instances of a Student, Subject, Teacher and Class.

We also need few JSPs:

admin-registration-initial.jsp,   
admin-registration.jsp,   
admin-list.jsp and   
admin-edit-form.jsp

portal.jsp  
login.jsp  
logout.jsp  
styles.css

### Sprint 4:

In the 4th sprite we will create all the functionality for the Teacher section of the portal.

We will also create all needed JSPs:

teacher-registration.jsp,   
teacher-list.jsp and   
teacher-edit-form.jsp

After creation it is very important to test everything.

### Sprint 5:

In the 5th sprite we will create all the functionality for the Subject section of the portal.

We will also create all needed JSPs:

subject-registration.jsp,   
subject-list.jsp and   
subject-edit-form.jsp

After creation it is very important to test everything.

### Sprint 6:

In the 6th sprite we will create all the functionality for the Student section of the portal.

We will also create all needed JSPs:

student-registration.jsp,   
student-list.jsp and   
student-edit-form.jsp

After creation it is very important to test everything.

### Sprint 7:

In the 7th sprite we will create all the functionality for the Class section of the portal.

We will also create all needed JSPs:

class-registration.jsp,   
class-list.jsp and   
class-edit-form.jsp

class-student-list.jsp  
welcome.jsp

After creation it is very important to test everything.

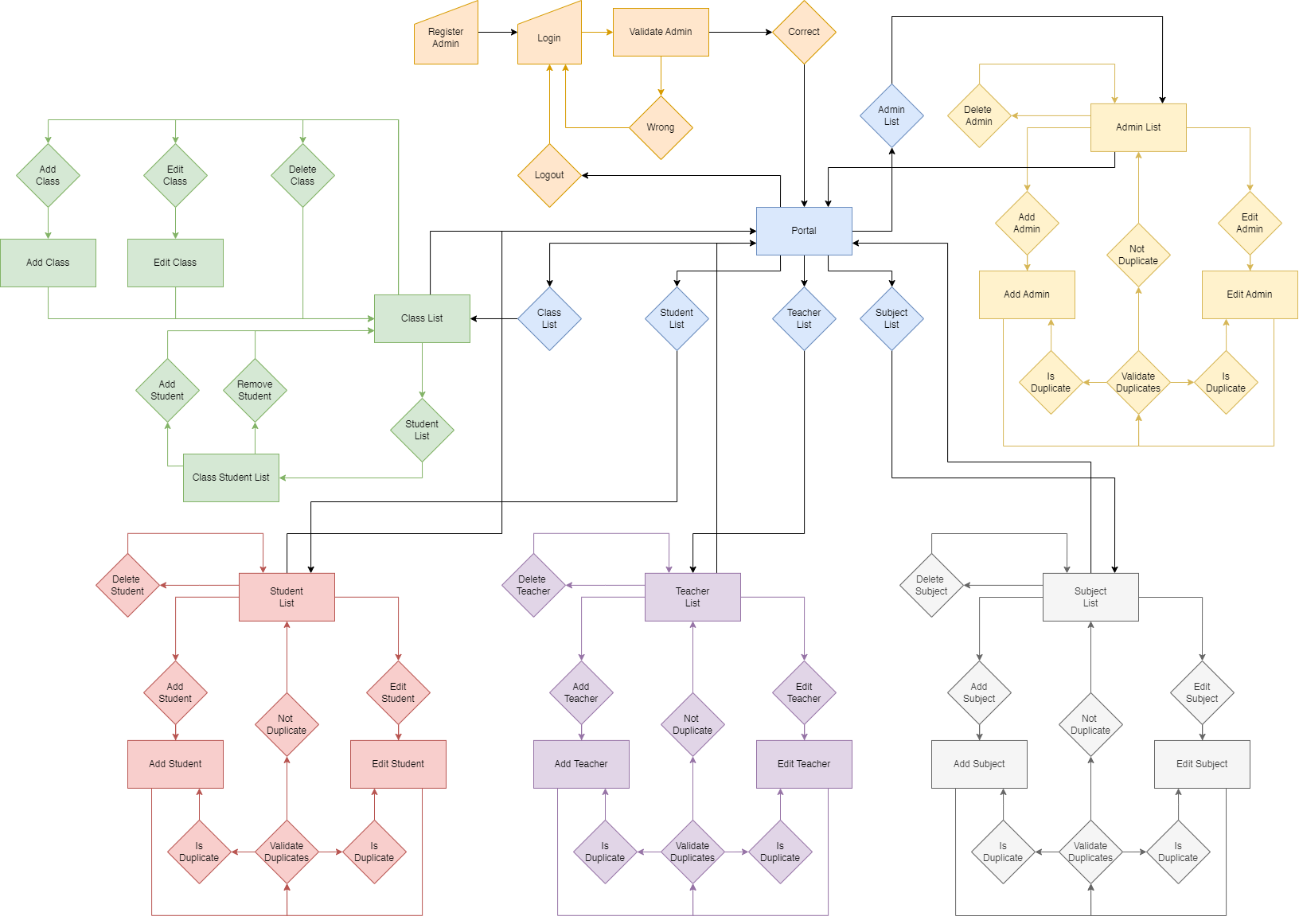
### Sprint 8:

In the last 8th sprite we are going to test the portal fully and to create functionality snips. After testing we are going to create necessary documentation.

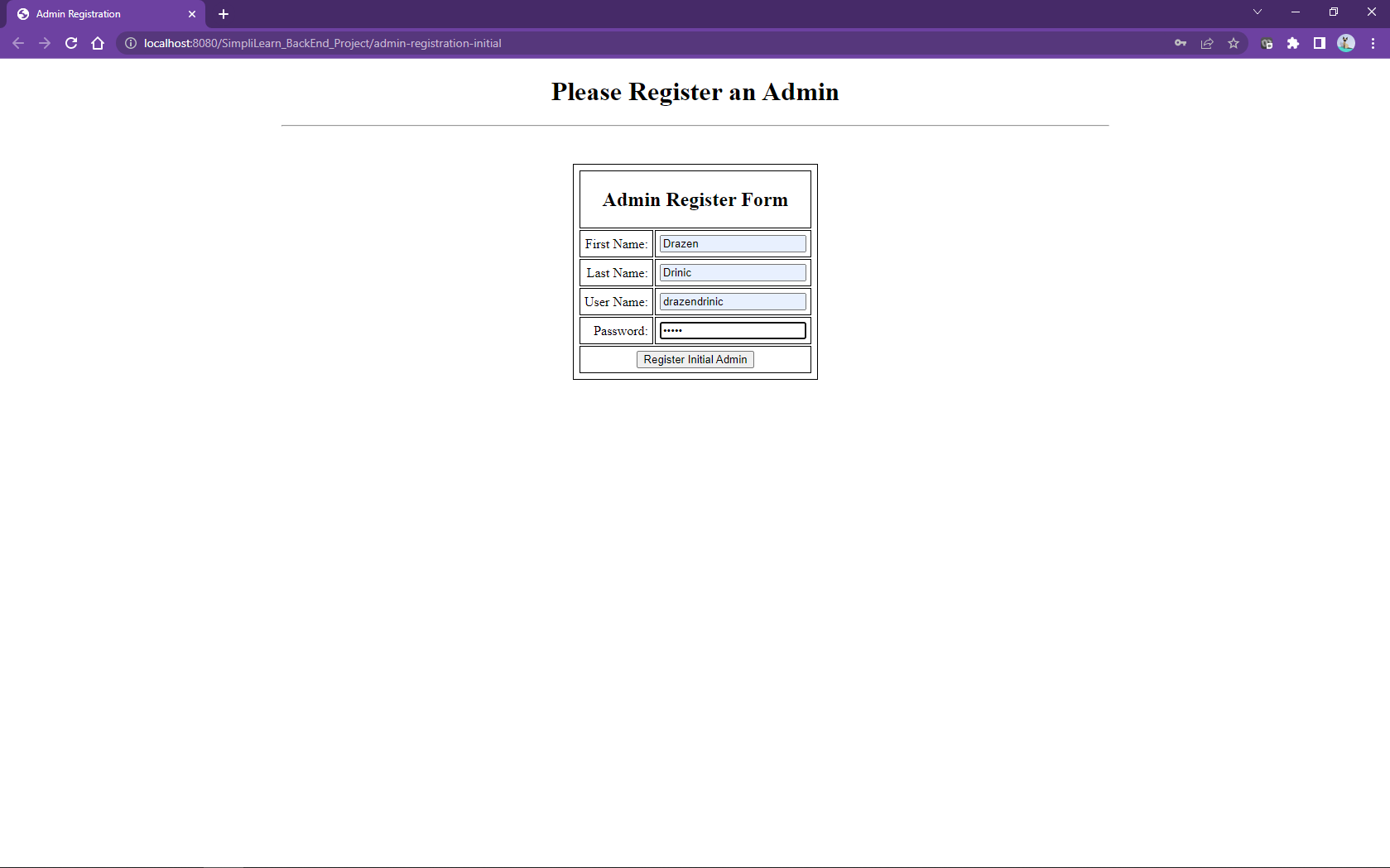
## Concepts in use

* Java core
* Hibernate and JPA
* DAO
* Servlet
* JSP
* JSTL
* CSS and HTML
* And other concepts

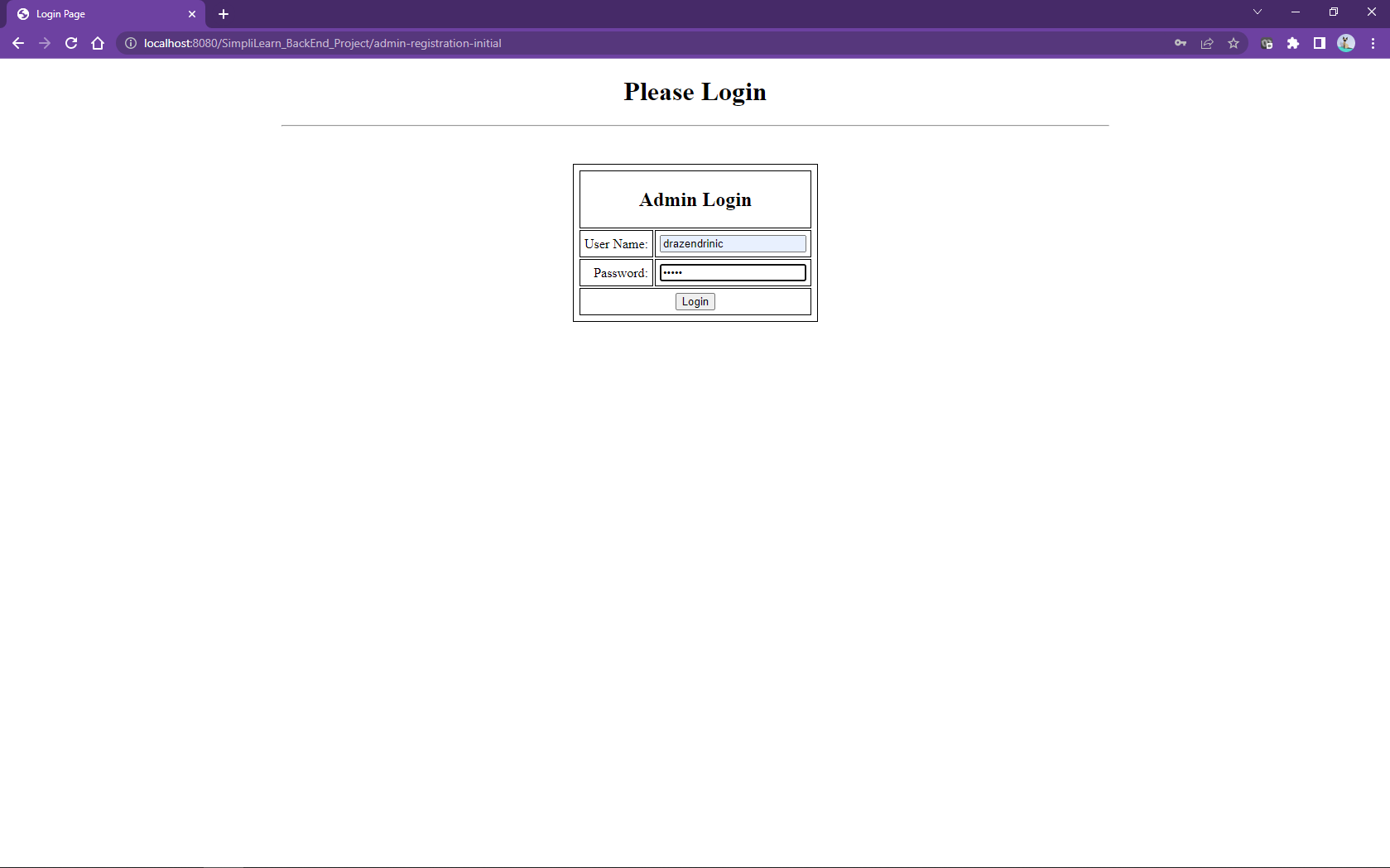
## App WorkFlow Chart



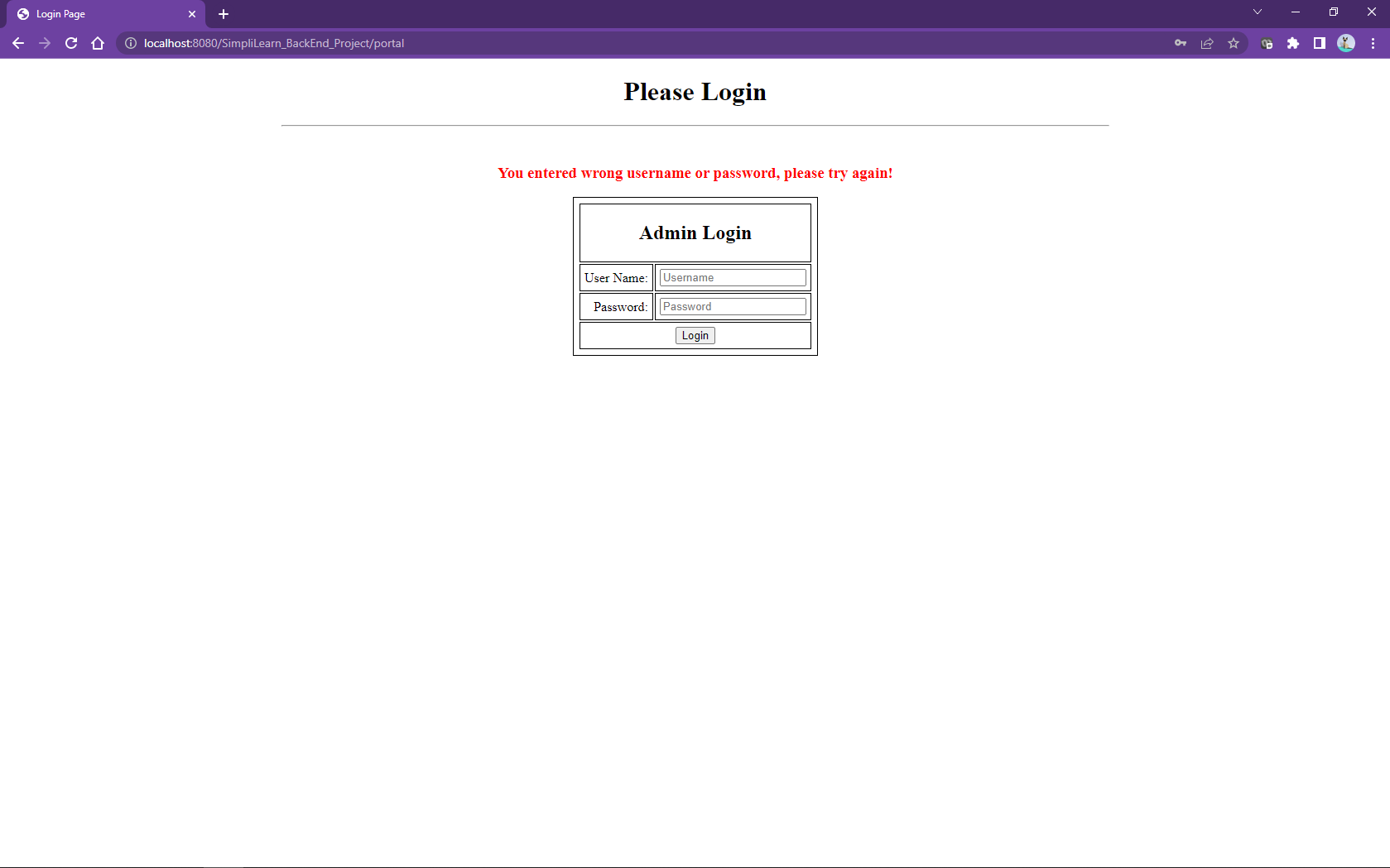
## App sample snips

This is the first interaction of the user with the Learners Academy Admin Portal. To see Admin registration in action we are going to register an Admin. 

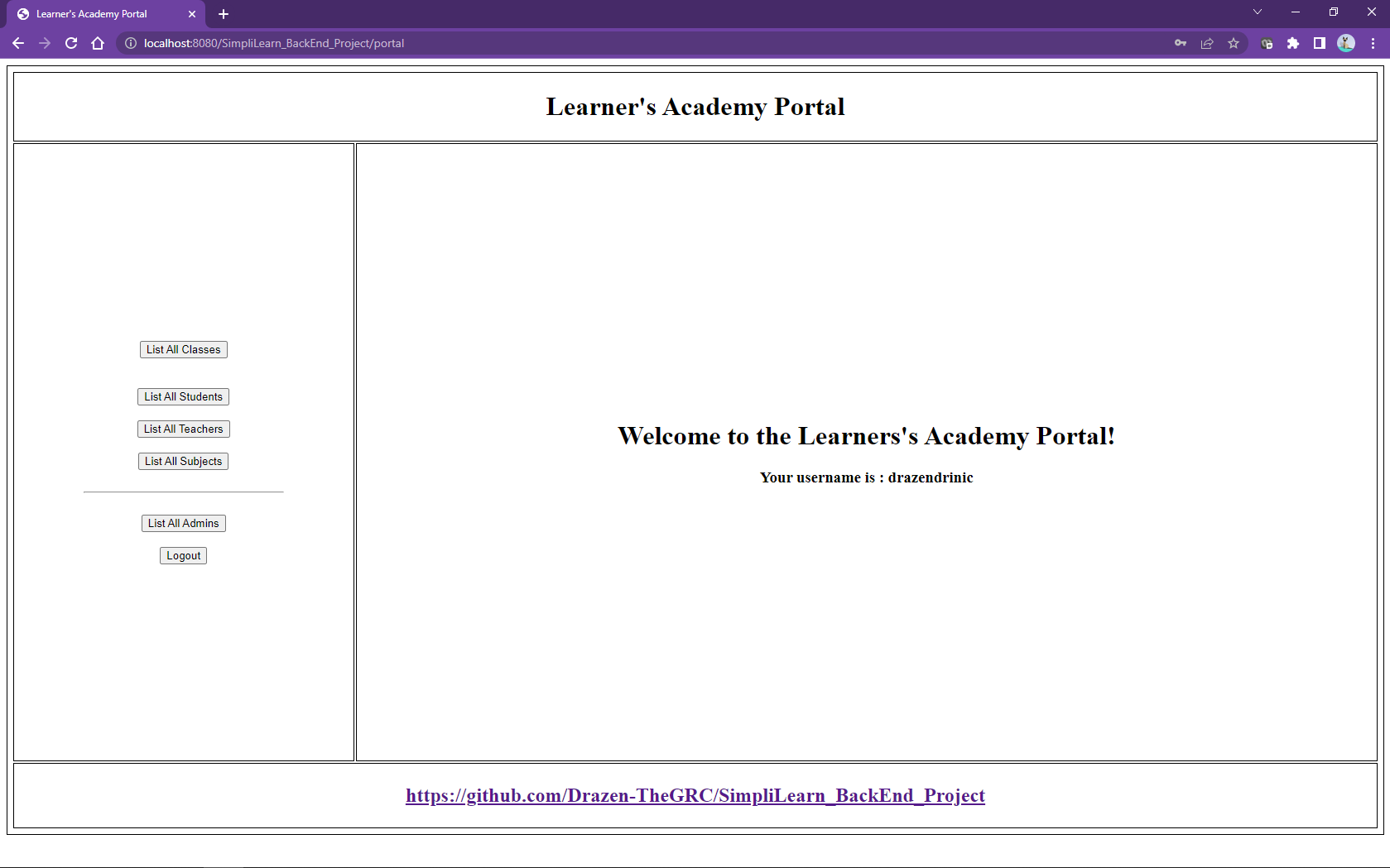
Login Page.



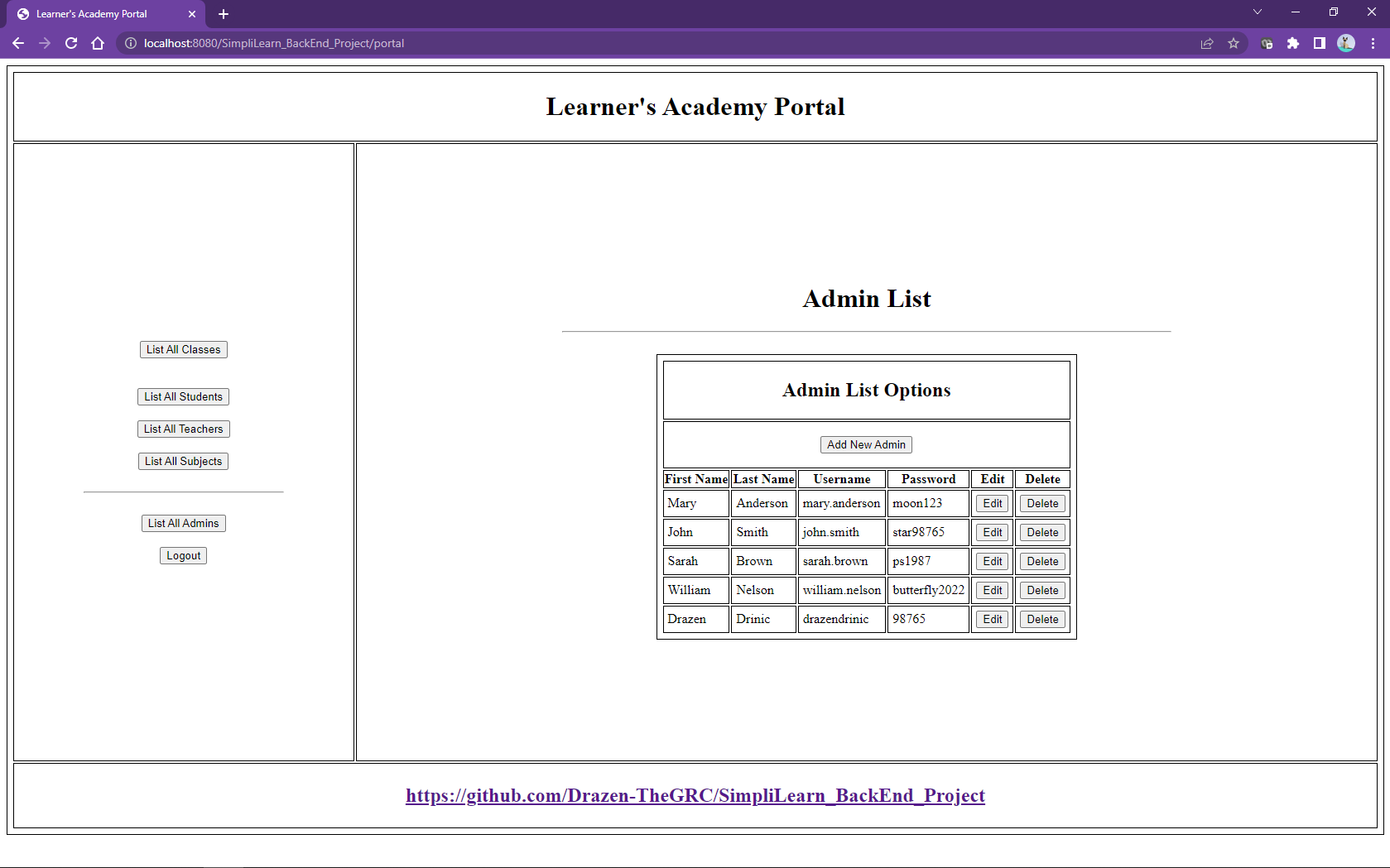
Login checks for username and password validation and prompts an error if there is no username and its related password in the database.



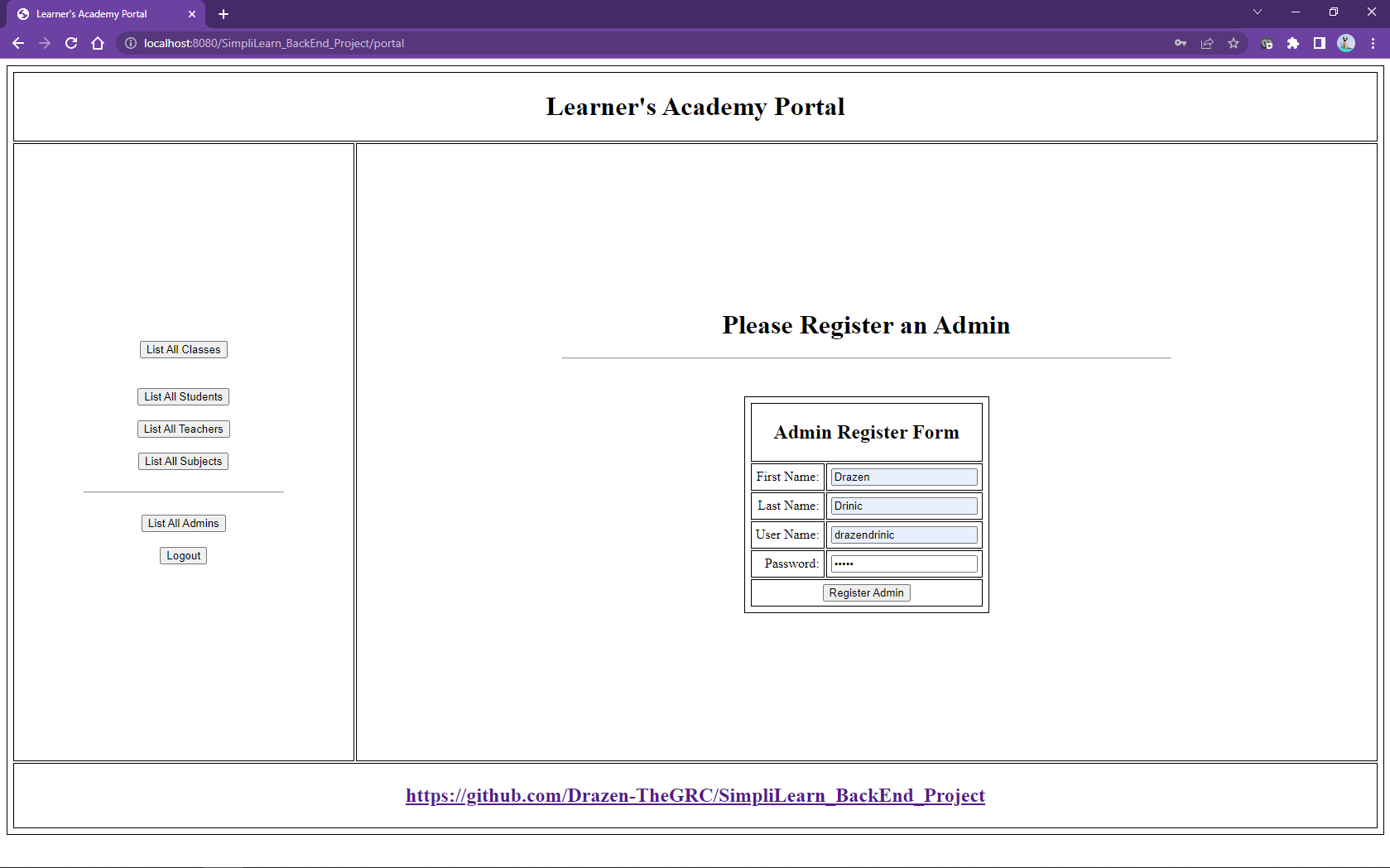
After successful Login we will se the Portal and its Welcome Page.



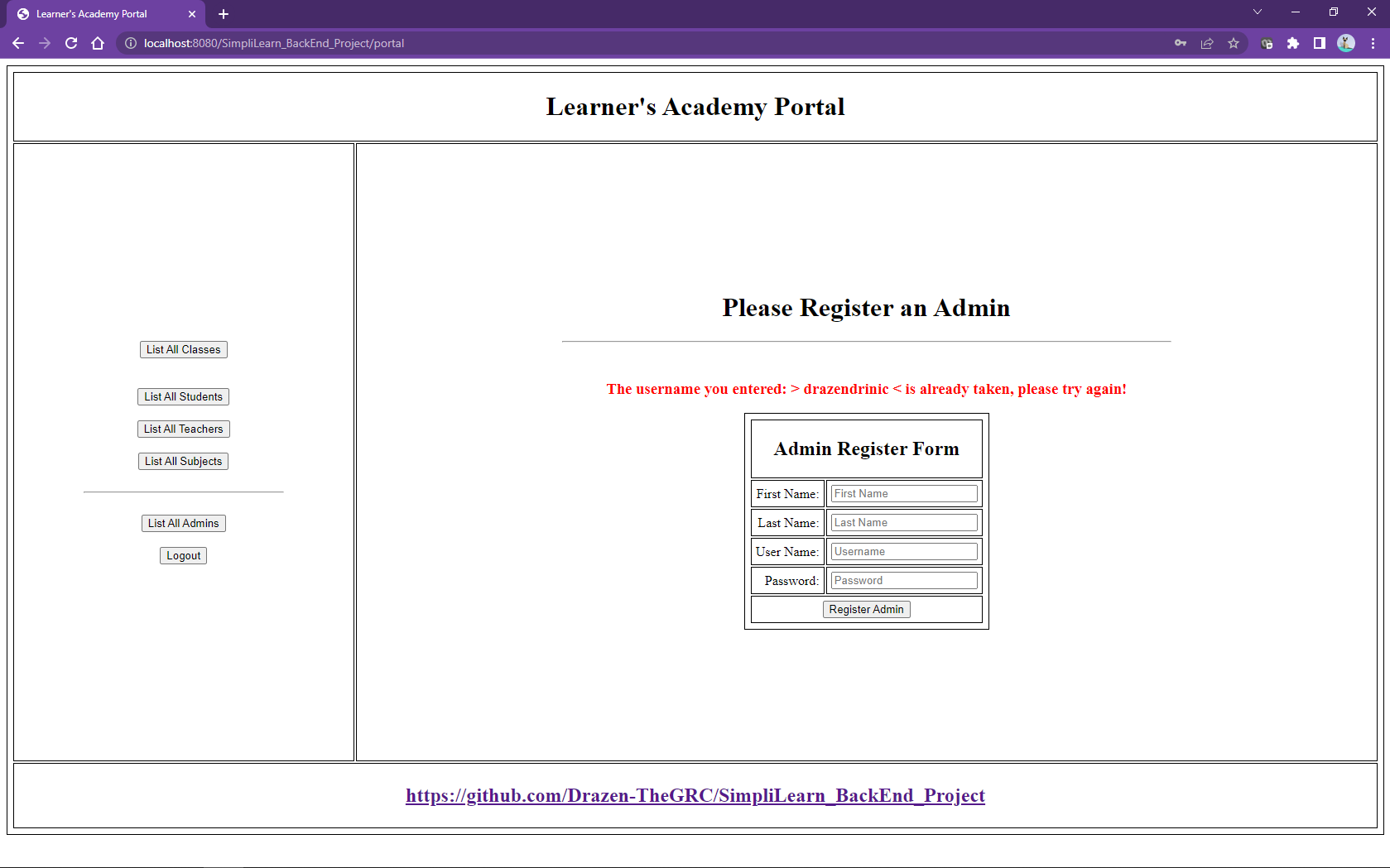
Admin can list all Admins and manage them from the Admin List Page.



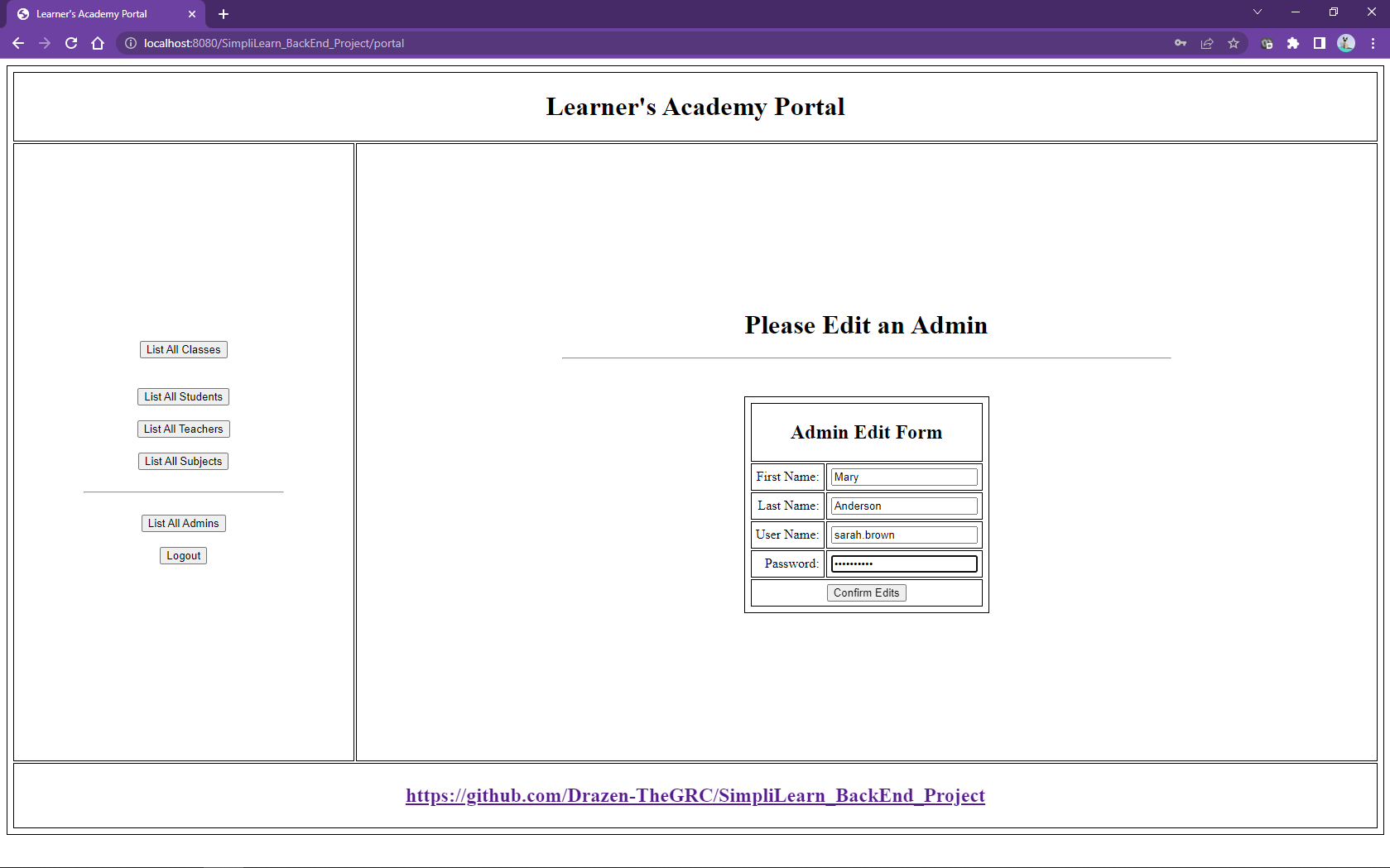
Adding another admin from the Portal.



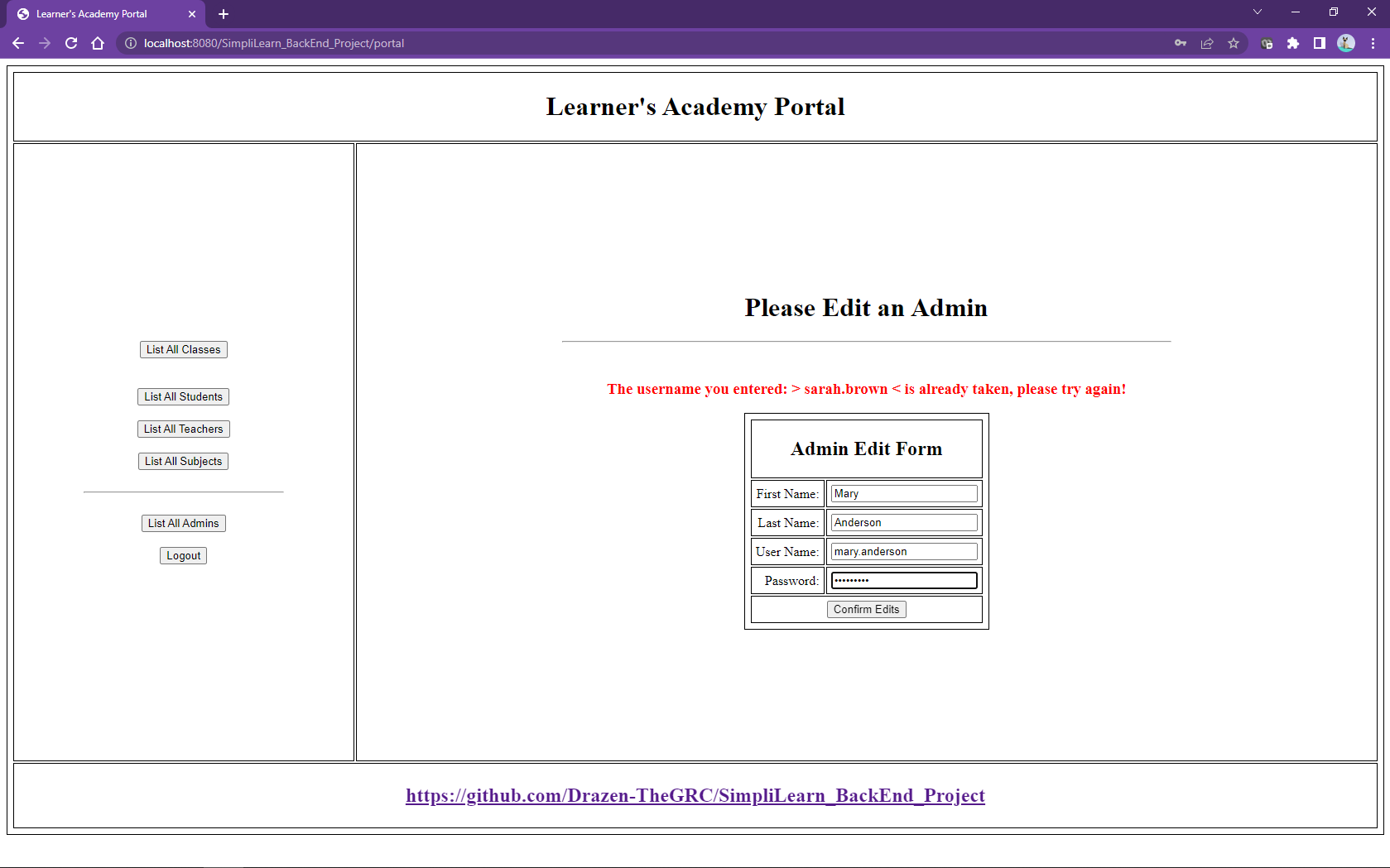
In case a username already exists in the database, an error message will be shown.



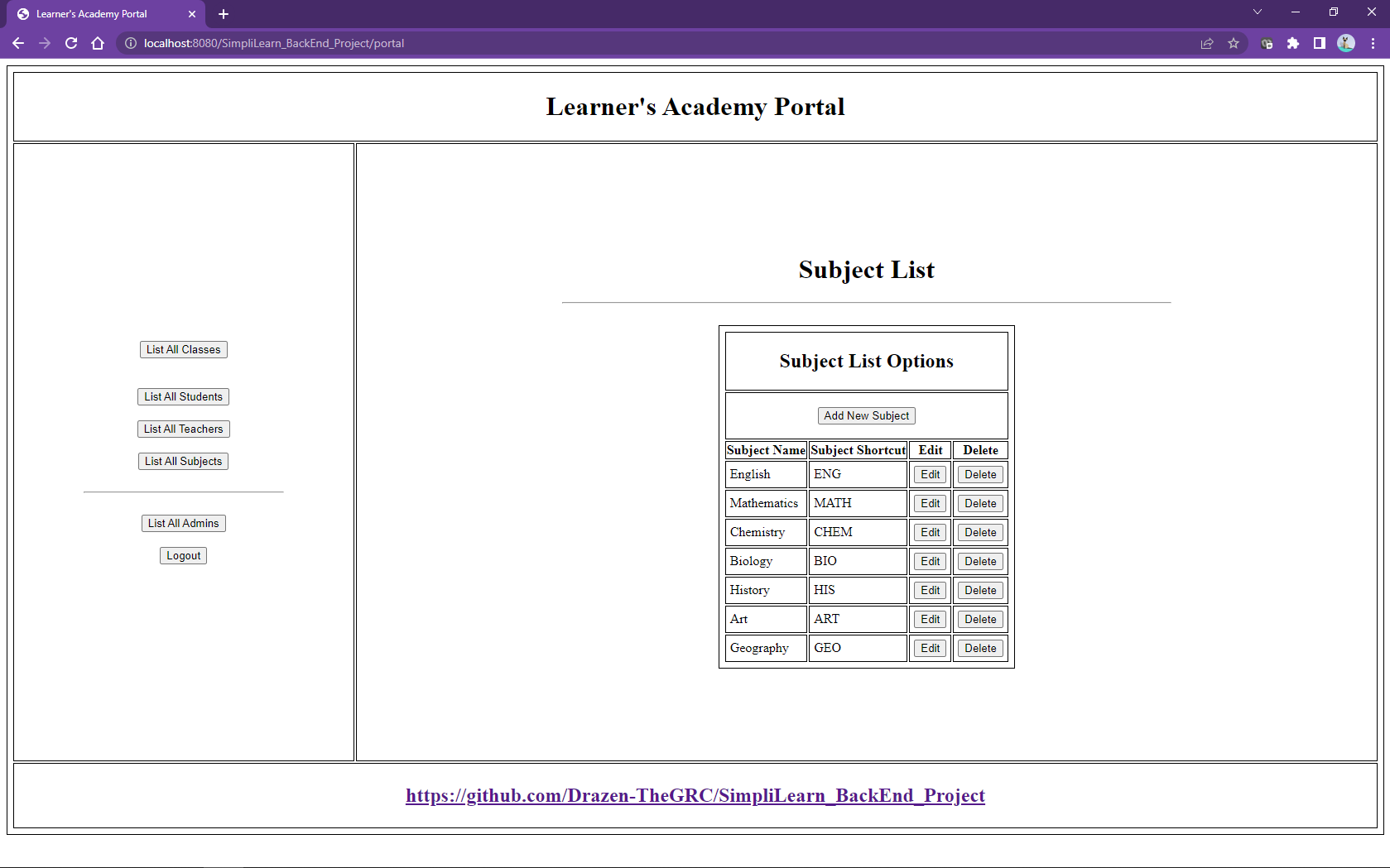
Admin can also edit existing Admins.



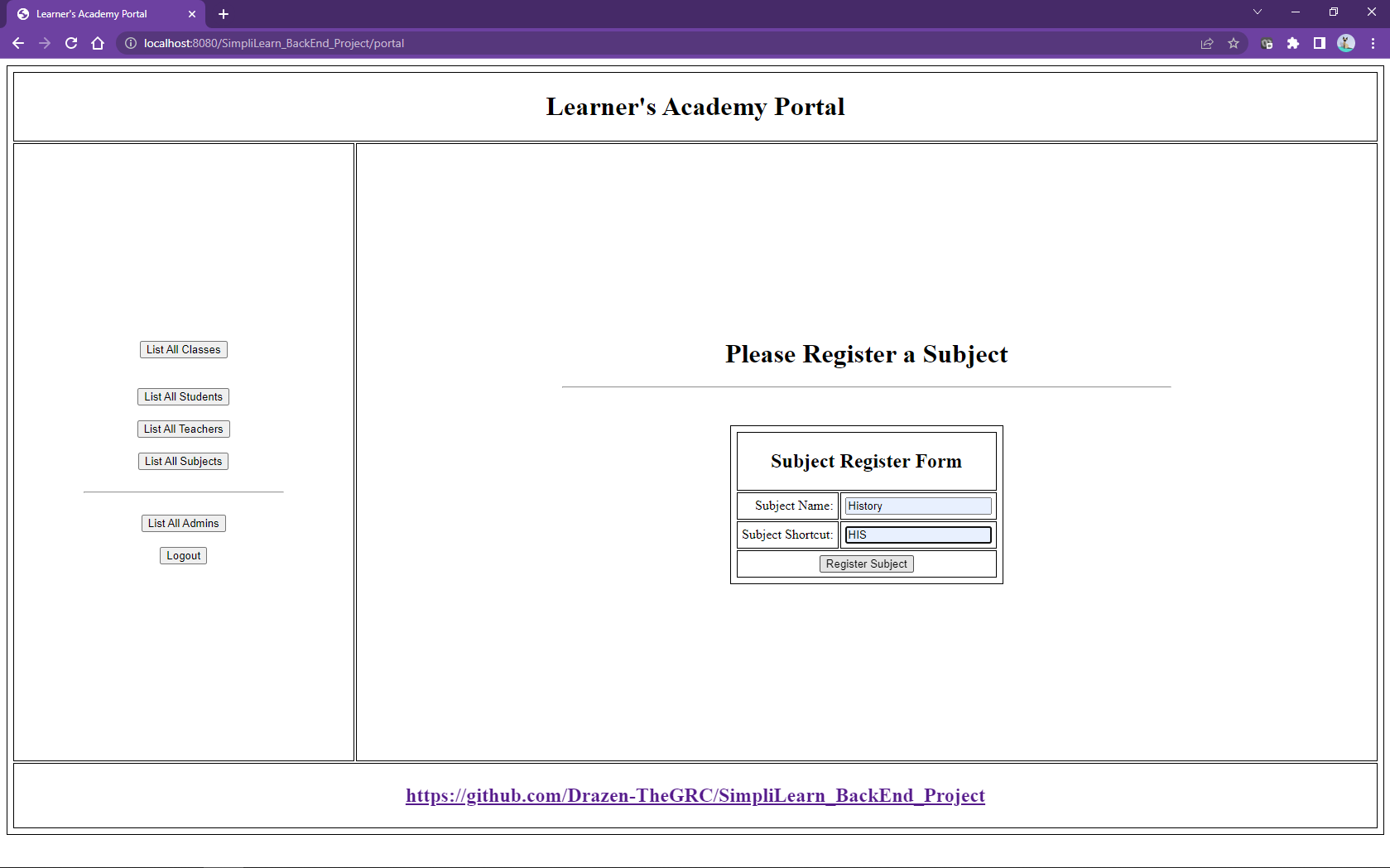
If during Admin Edit operations a duplicate username is edited we will get and error message.



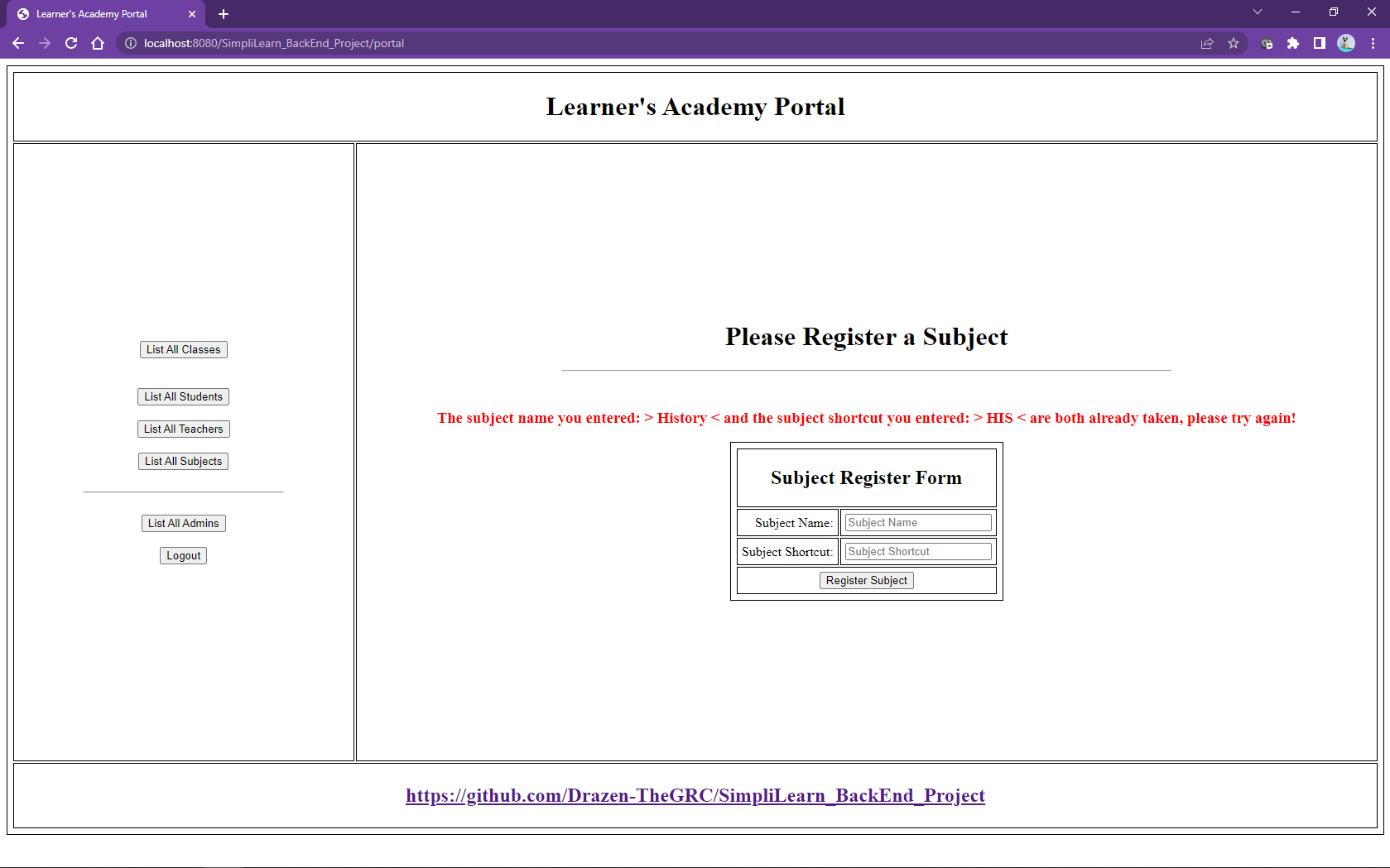
Amin can list all the Subject and manage them through the Portal.



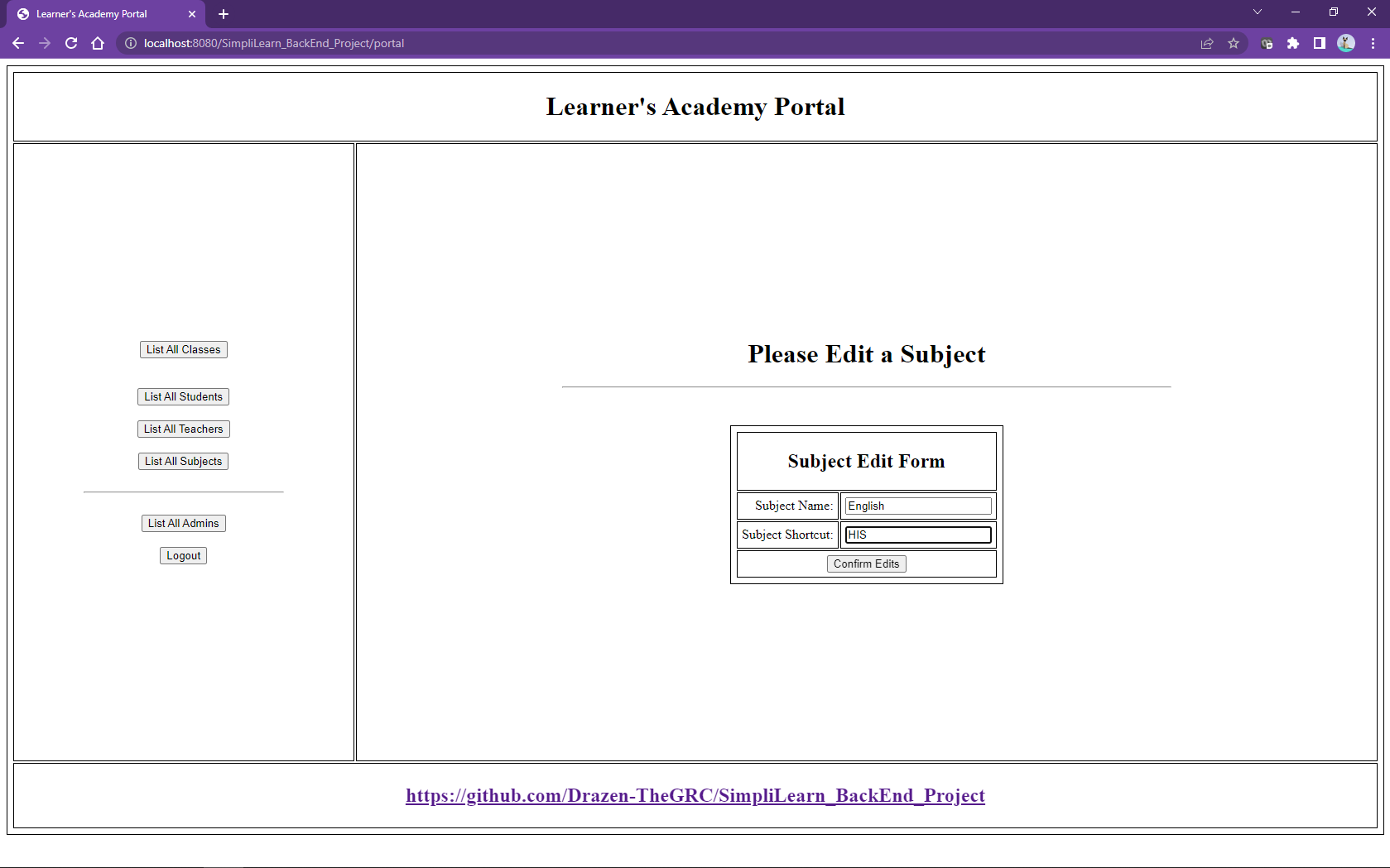
Admin can add new Subjects to the Portal



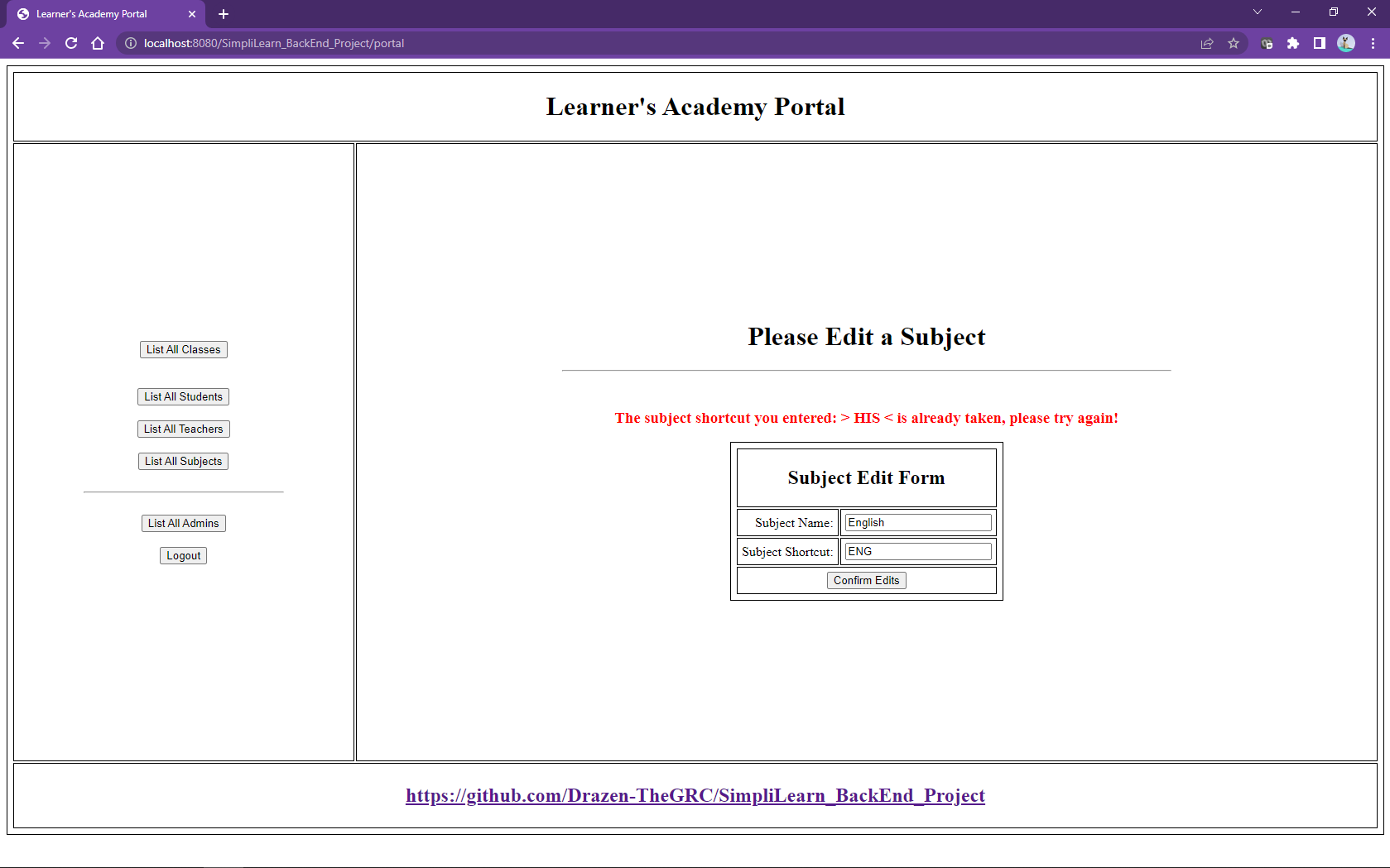
If Admin tries to add an existing Subject an error message will inform the user.



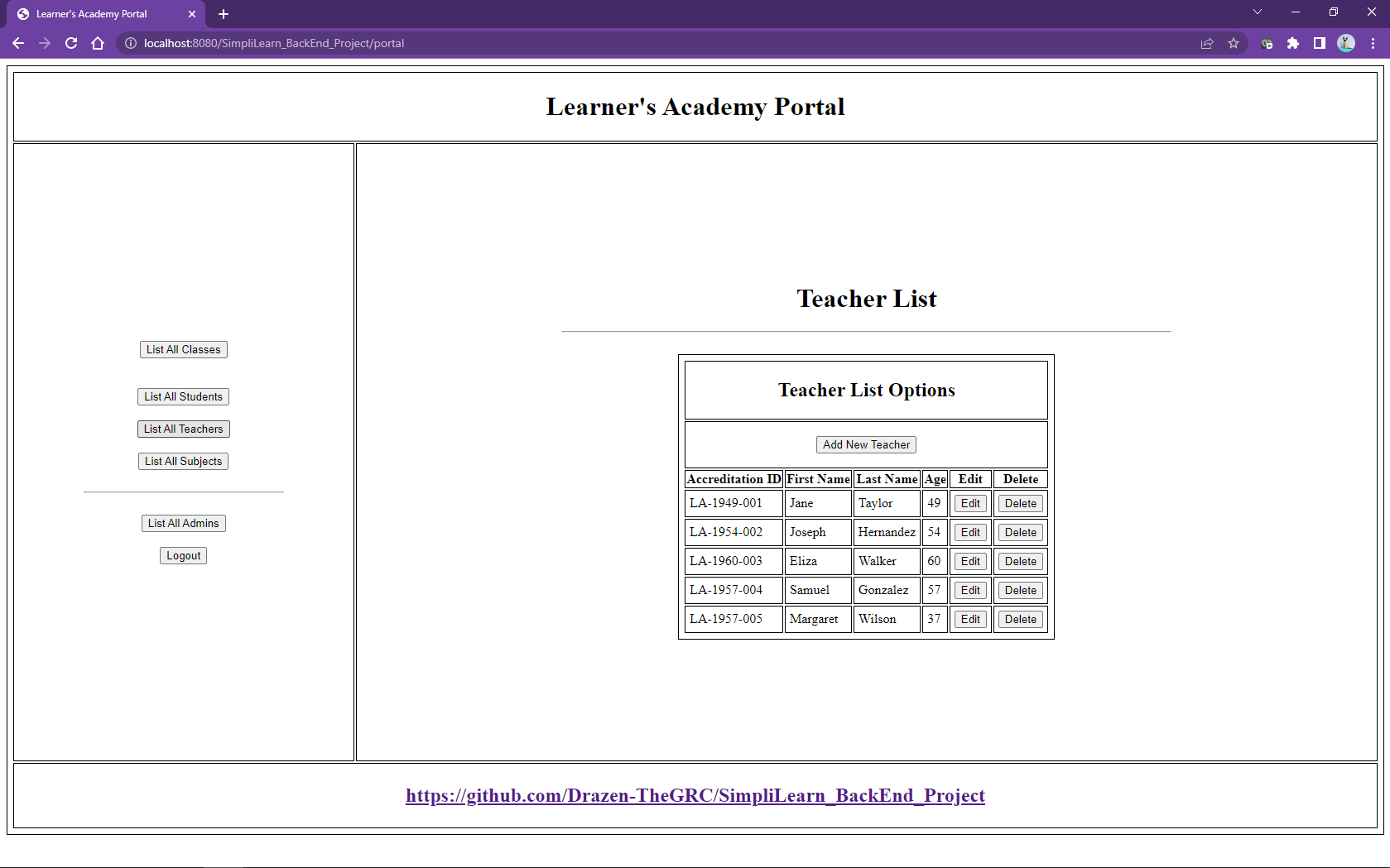
Subject can be also edited and edits are going to reflect the classes.



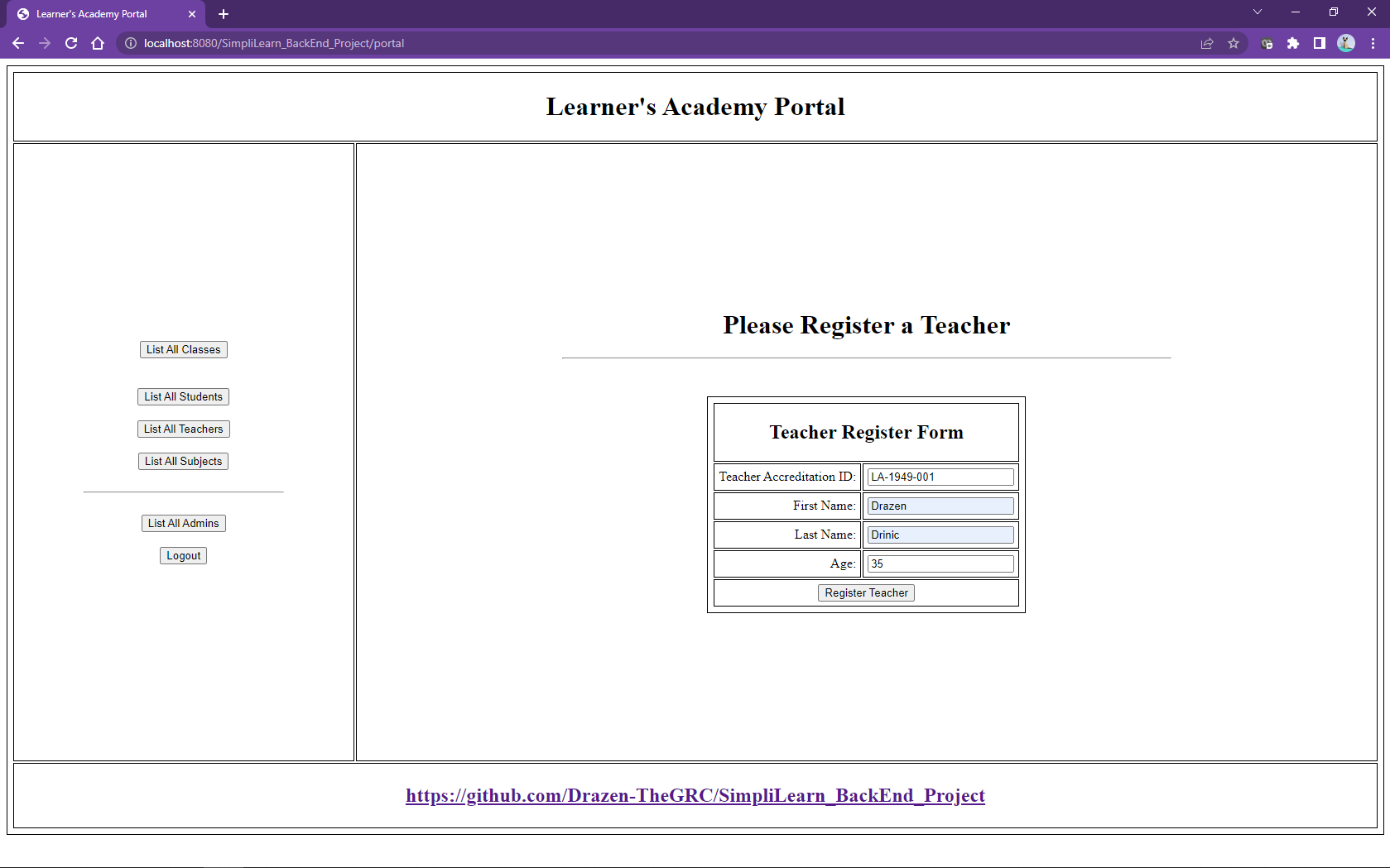
Again, if we try to edit an existing subject to another existing subject there will be an error.



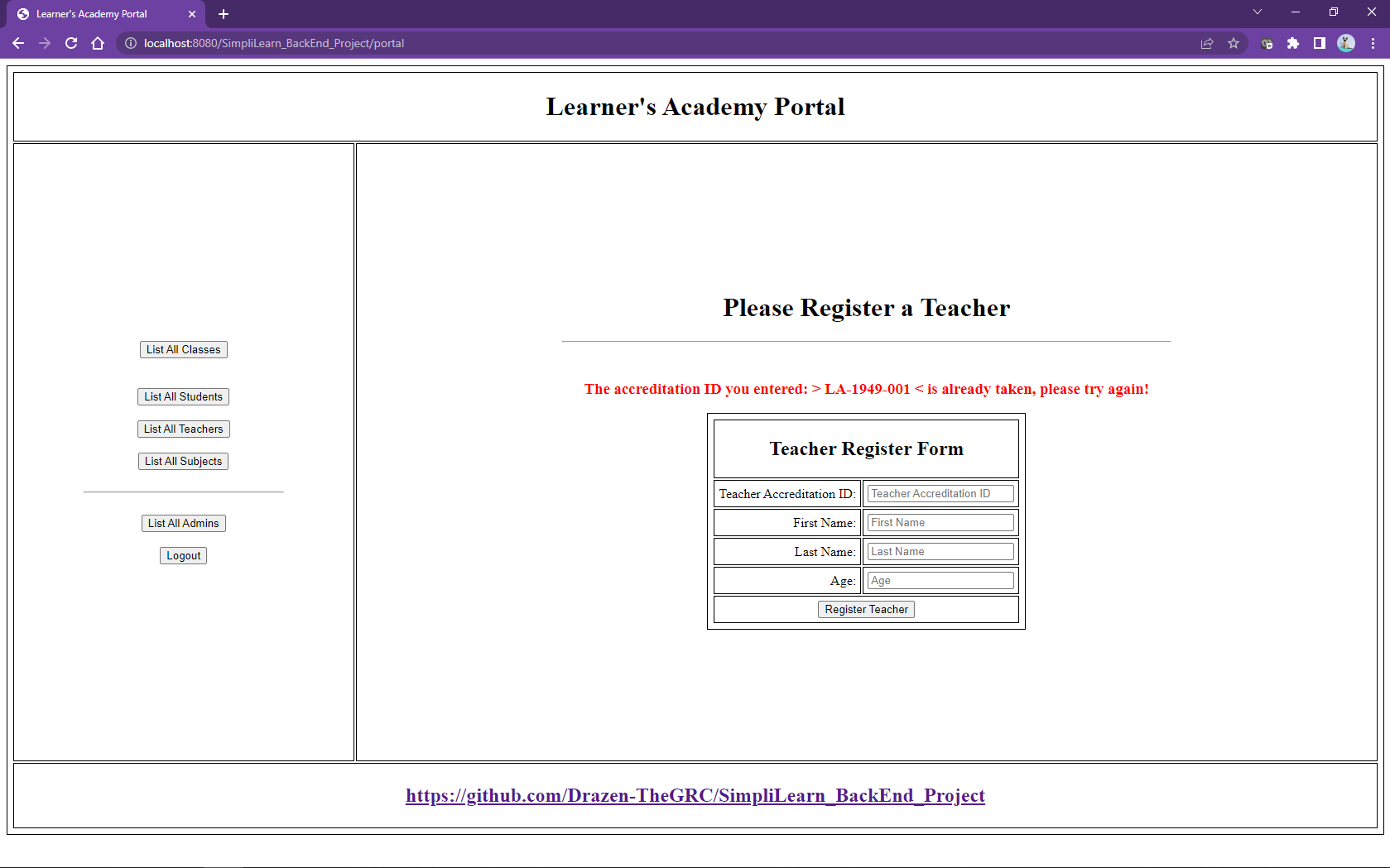
Admin can manage Teachers on the Portal viewing and manipulating Teacher List



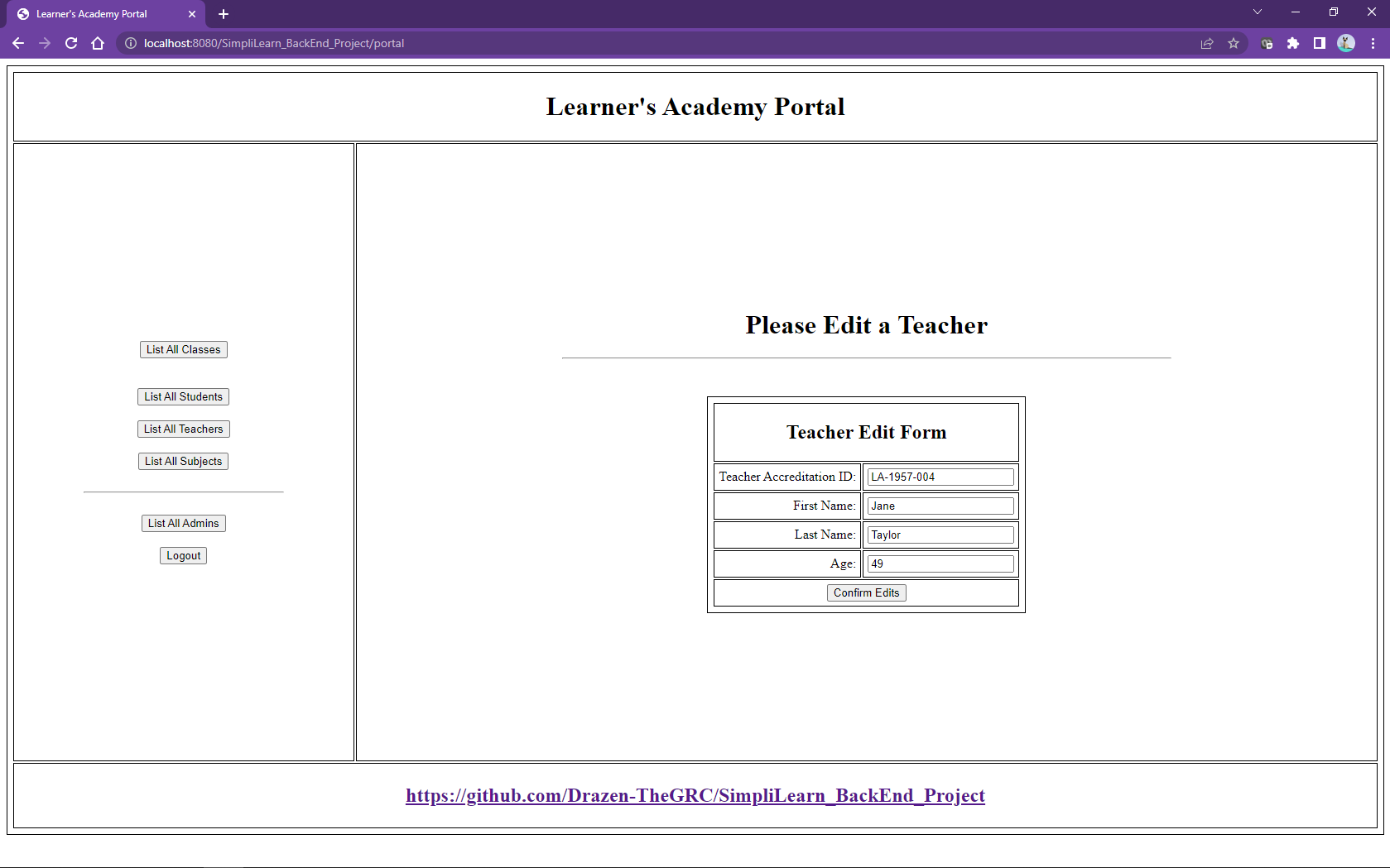
New Teachers can be added.



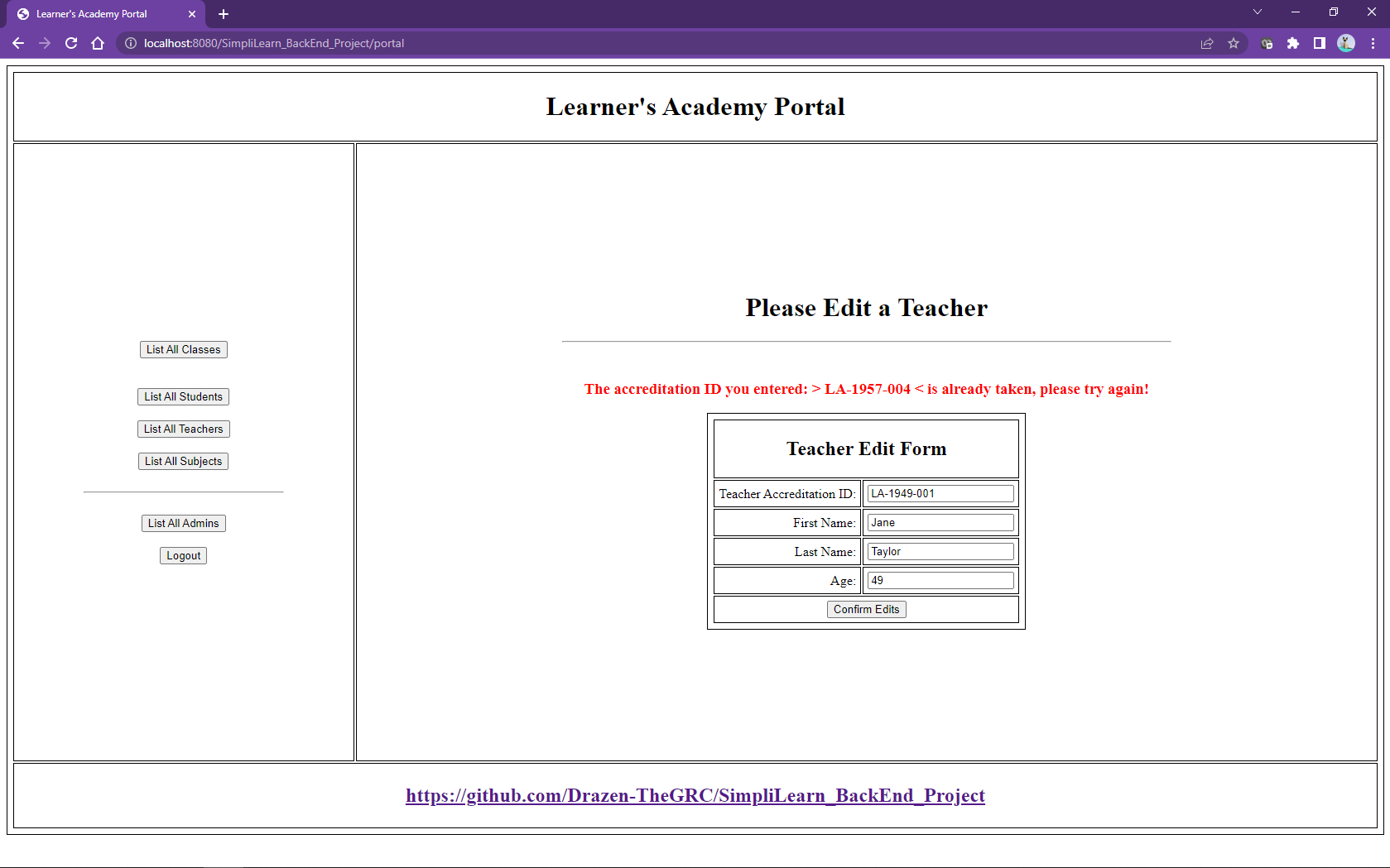
Each teacher has a unique teacher accreditation ID and in case we enter a duplicate there will be an error message preventing the Admin to enter duplicates.



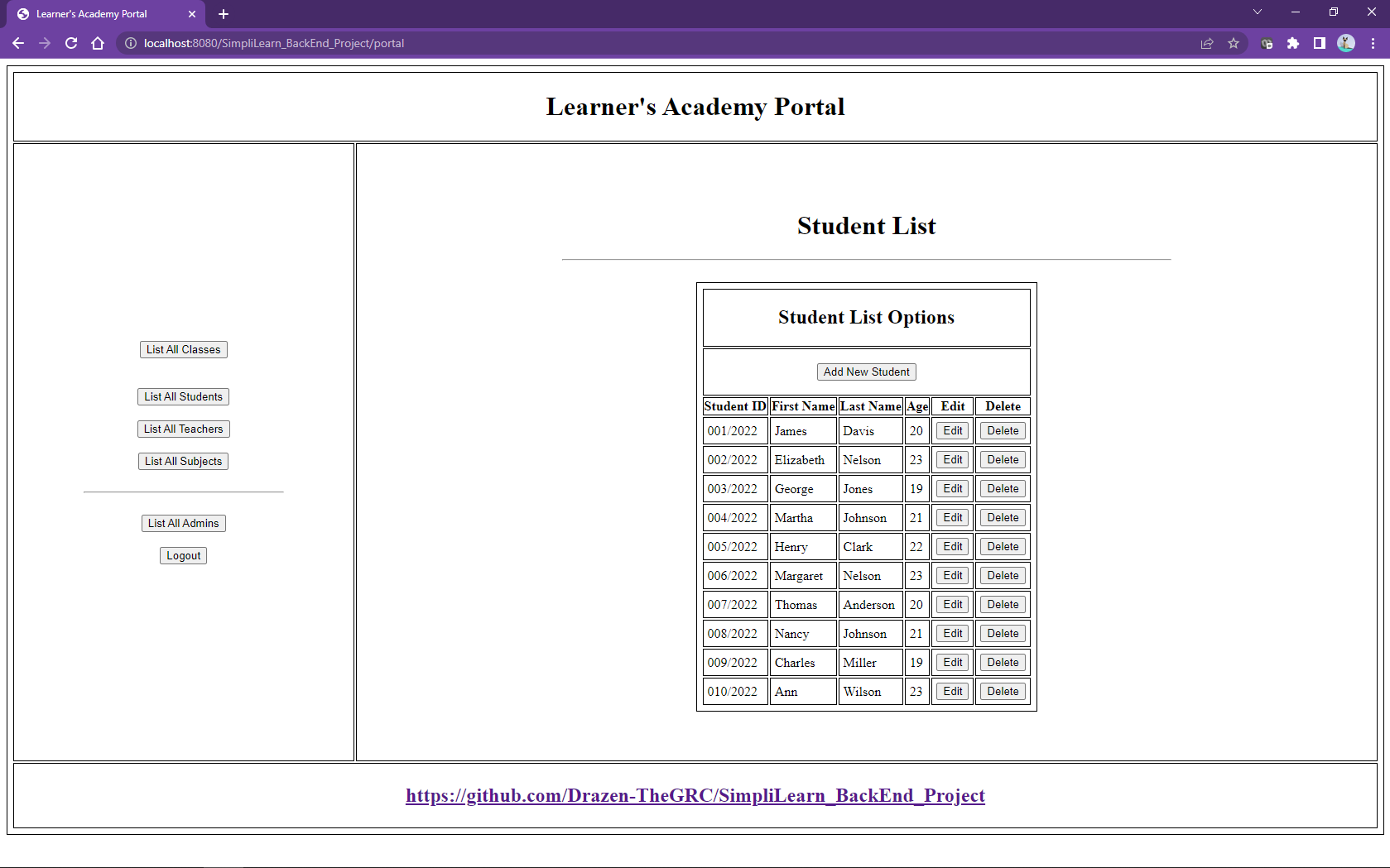
Teacher info can be edited.



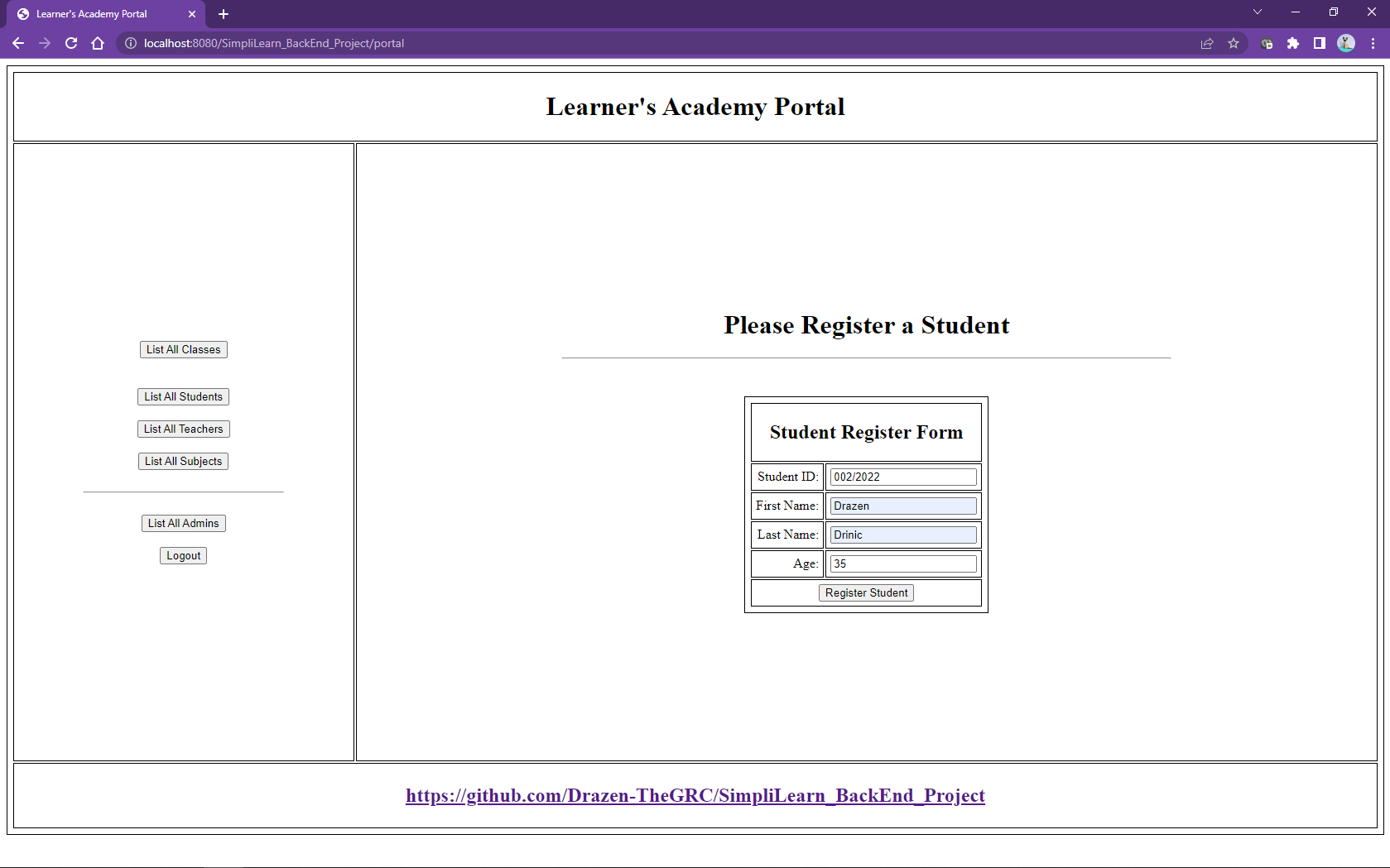
When editing teacher info, we can’t enter existing teacher accreditation IDs.



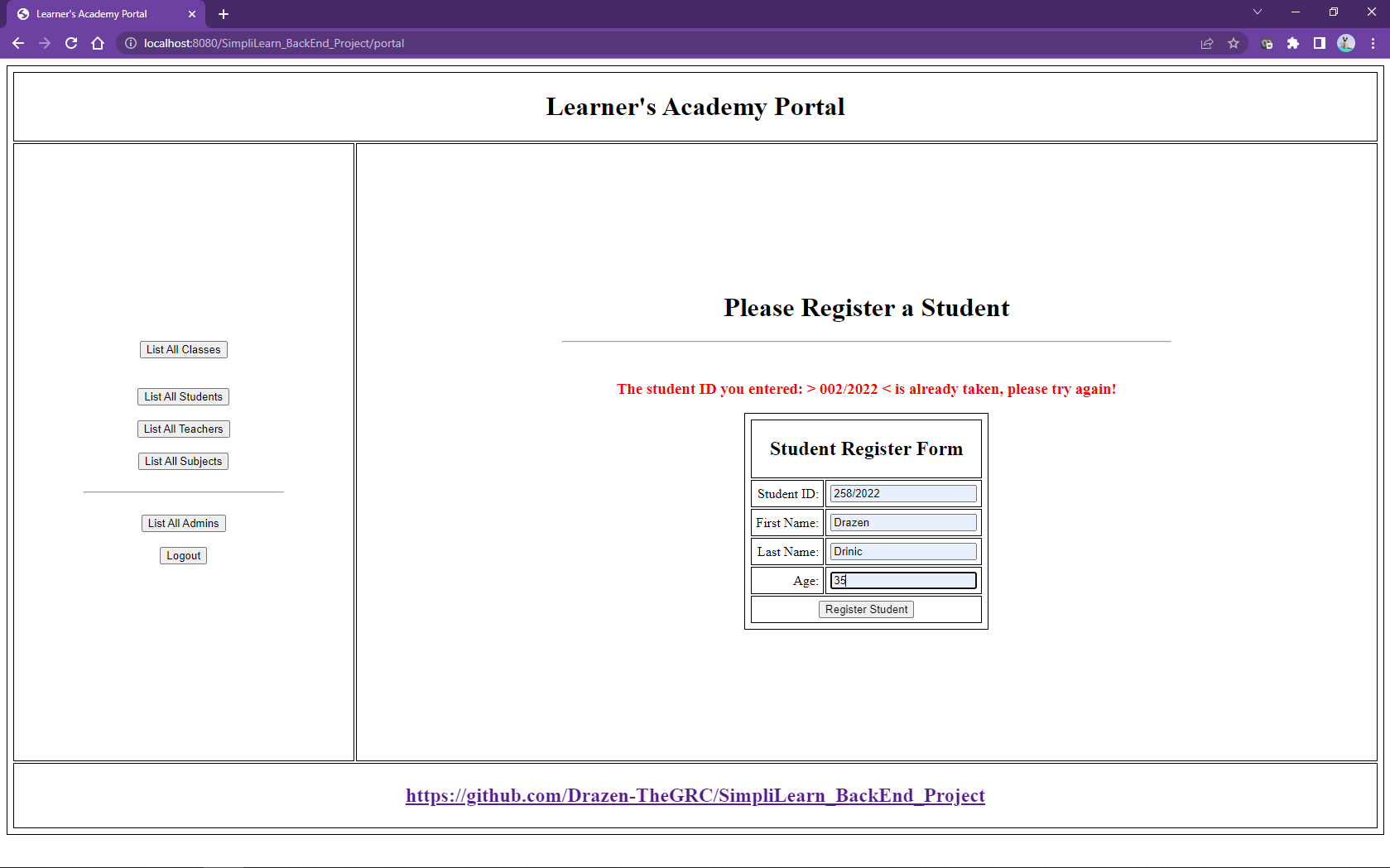
Without students there would not be a need for anything so Admin can also manage Students.



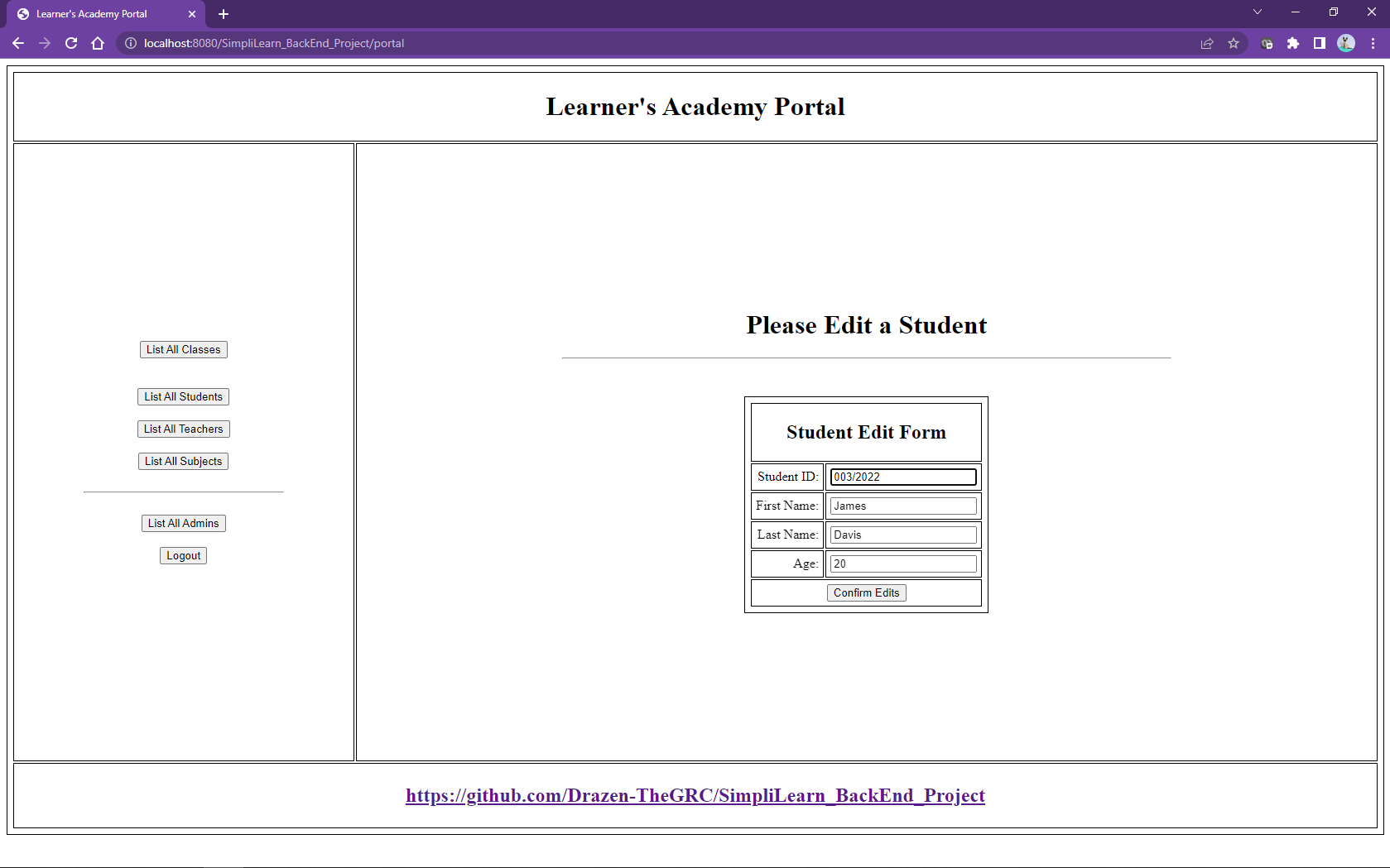
New students can be added.



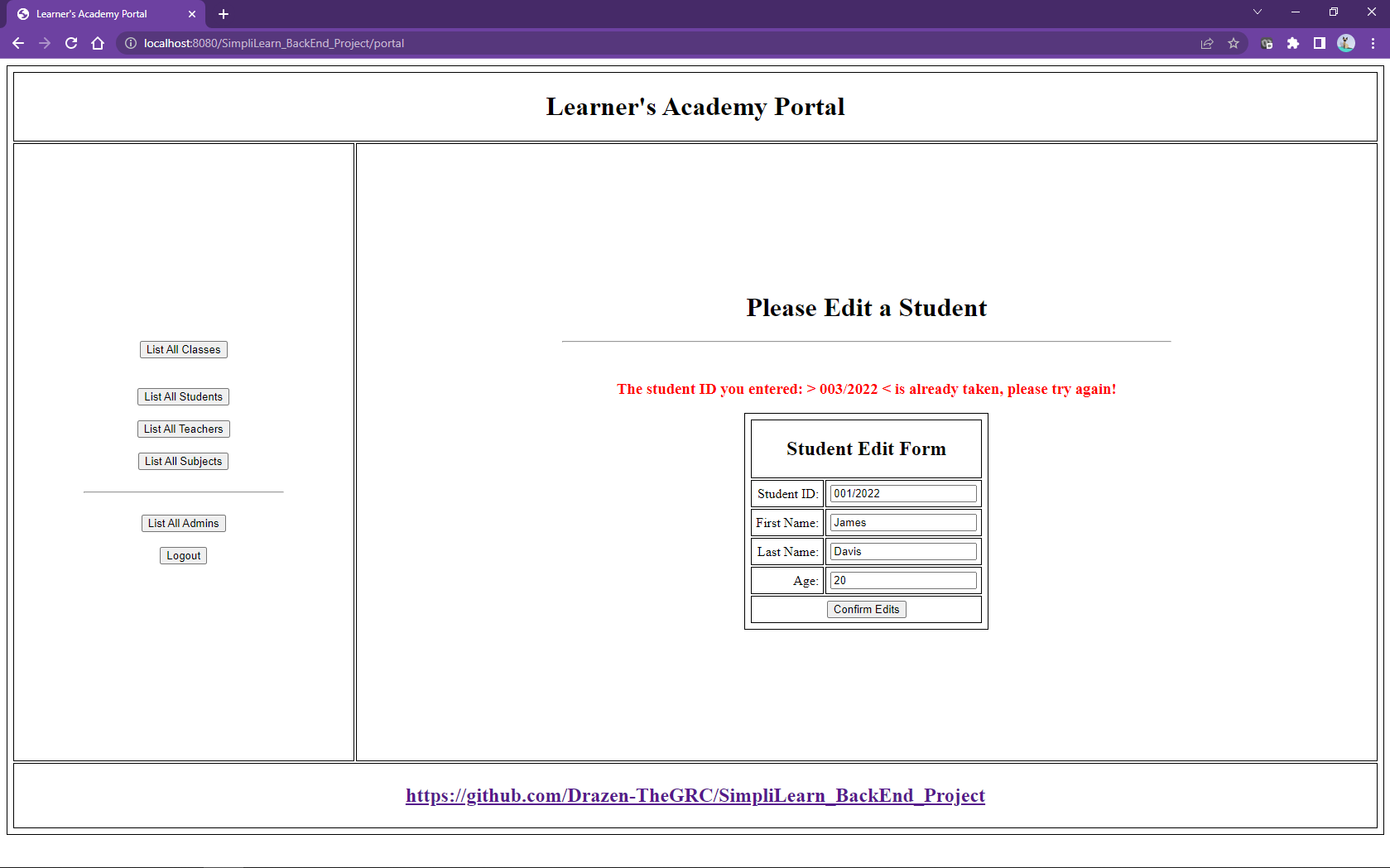
Same as with the Teacher unique accreditation ID students have their unique student IDs.



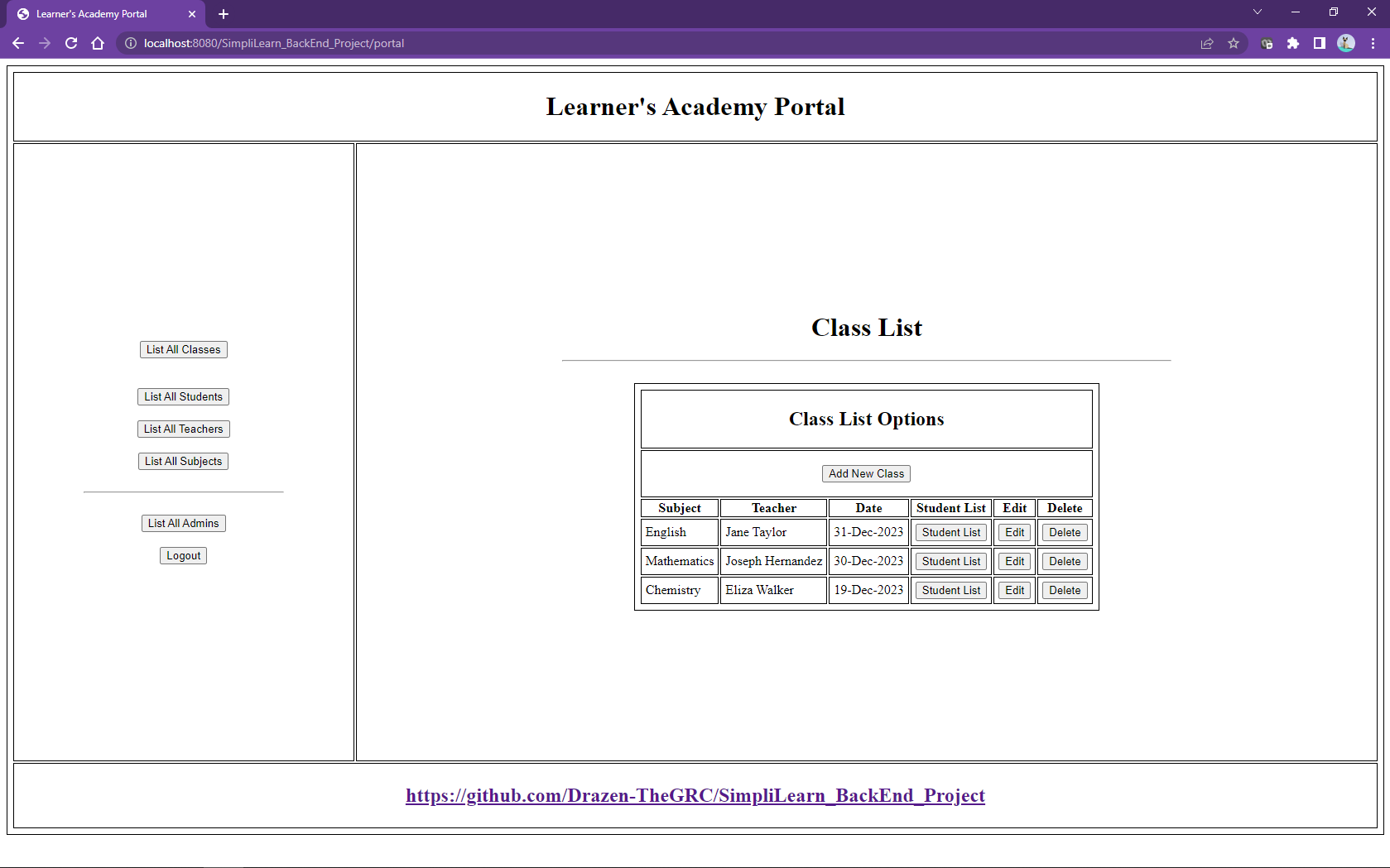
Student info can be edited.



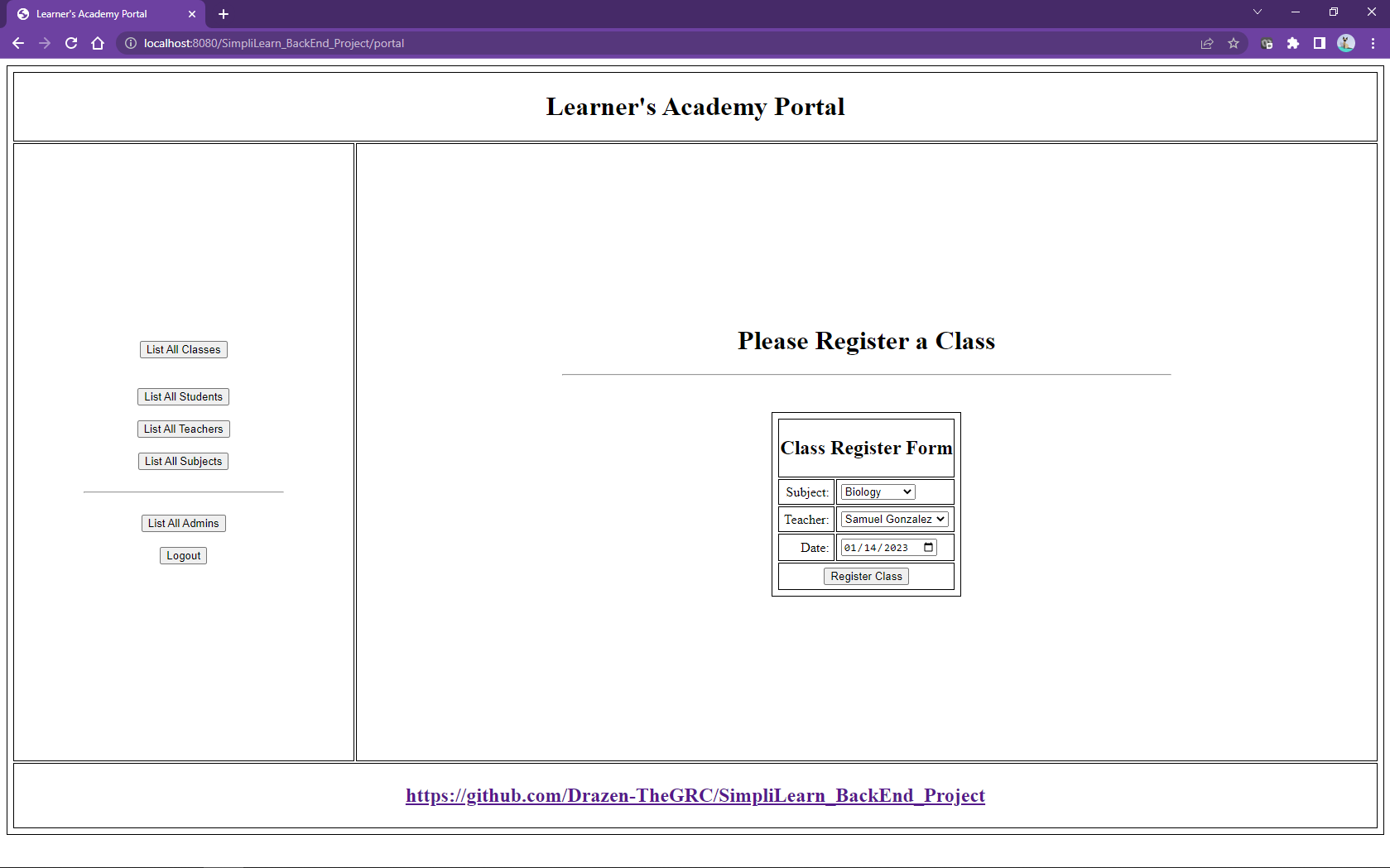
When editing students, it is important to prevent assigning already used Student IDs



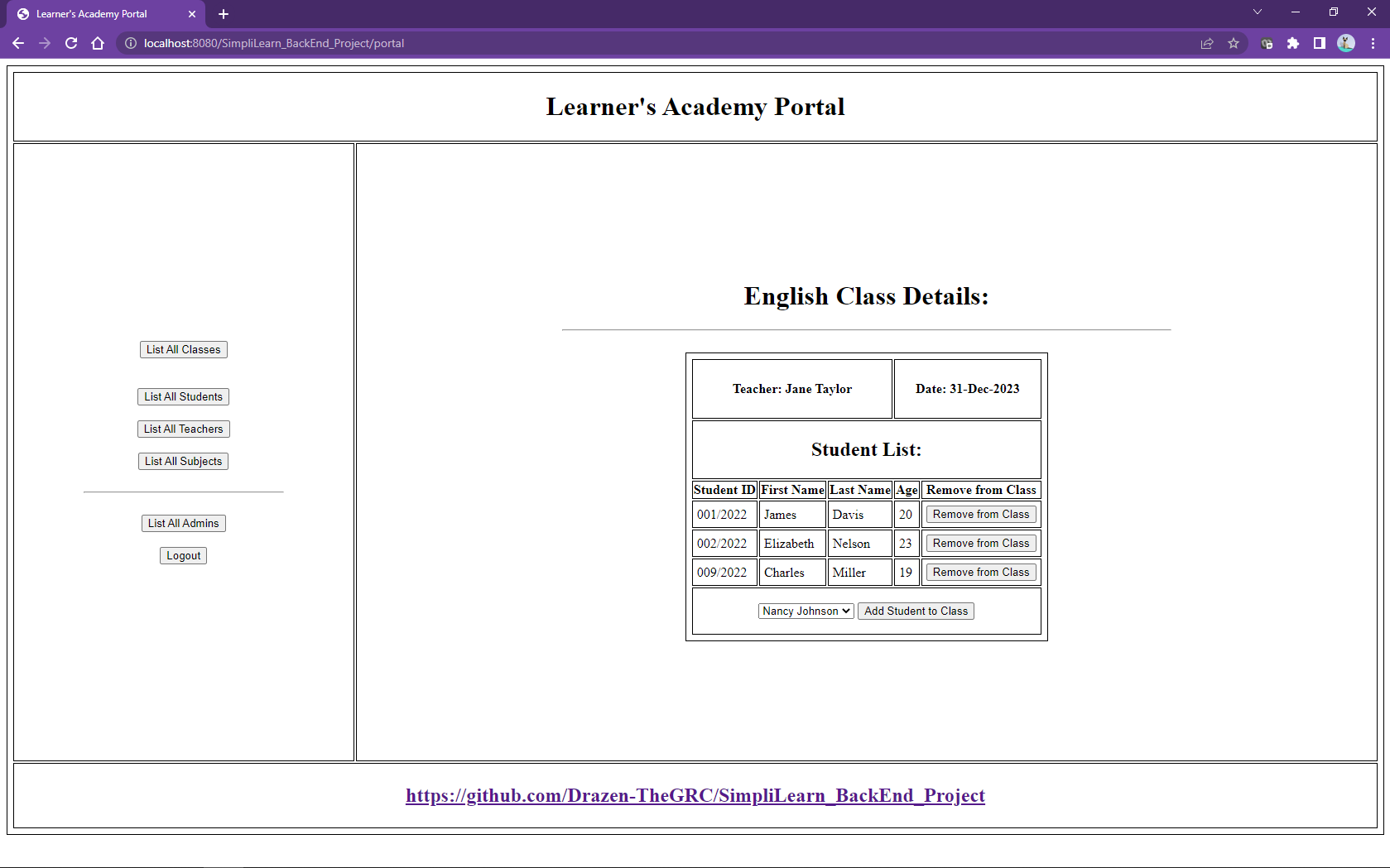
And at the end we come to Classes which are also managed based on available subjects, teachers and student.



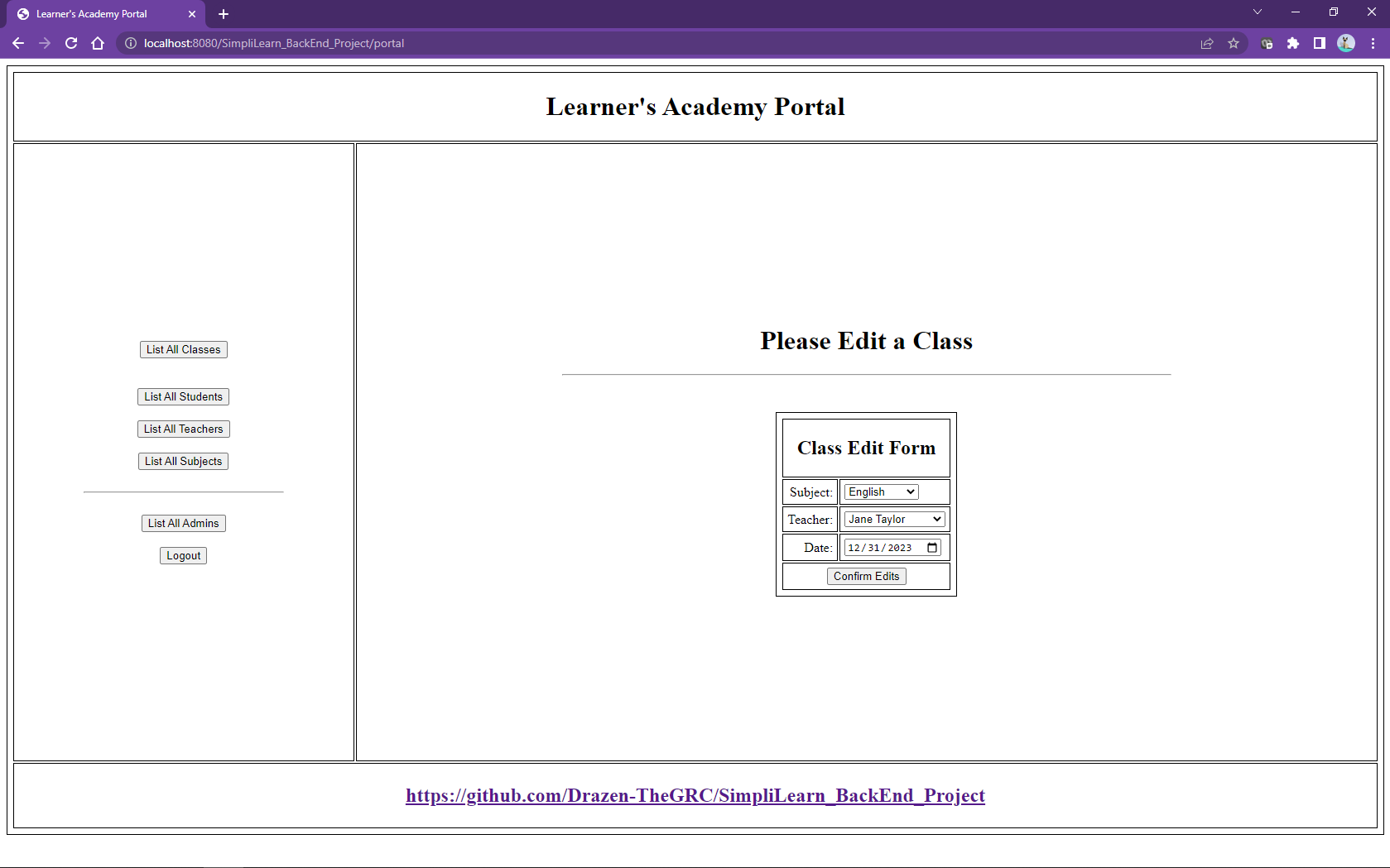
New classes can be added.



Students can be added or removed from classes.



And at the end it is important to be able to edit those classes.



## Algorithms with comments

For algorithms please visit the GitHub repository since the code is very large and it would take dozens of pages in this document.

<https://github.com/Drazen-TheGRC/SimpliLearn_BackEnd_Project>

## Unique selling points

* App has great presentation,
* App is very simple to use,
* User interaction is very simple with minimal input

## Conclusion

Application is prepared according to required features and tested. App project directory contains all the needed and extra documentation including a report, app use snips and workflow charts.