INDIVIDUAL PROJECT FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title	WEBG301 - Project Web		
Submission date	09/01/2022	**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name	Nguyen Duc Anh	**Student ID**	GCH17051
Class	GCH0805	**Assessor name**	Nguyen Dinh Tran Long
Student declaration I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.			
		Student's signature	ducanh

Grade

❒Summative Feedback:	❒Resubmission Feedback:	
Grade:	**Assessor Signature:**	**Date:**
Signature & Date:		

Project Web

Student Management System

I. Introduction

The article is a individual report on the student management system, consisting of three parts. Part one introduces user requirements, ERD database design, and Site map. The second part shows the MVC pattern used in the project and screenshots of the website. The final part evaluates the site's strengths and weaknesses, and future improvements. In this project, the author only works with Student and Class function. Therefore, in this article, some part only introduces about things related to the above two functions.

II. User Requirement

A user story template is a common format used to write user requirements that help to define key pieces of information about that user requirements. One particular user story template, often related to as "As a ... (who wants to accomplish something), I want to ... (what they want to accomplish), So that ... (why they want to accomplish that thing)" is the most commonly recommended for teams and project owners starting to work and develop the product in general. A simple example:

- As a customer
- I want to purchase a product from a website
- So that I can add a product to the shopping cart and login to the page to purchase it.

The following table describes the user story template of the student management system starting with the user admin. Admin can manage courses, classes, classrooms, students, and teachers. While Teachers and students can browse and search for that information.

As a/an	I want to…<perform some task>	So that I can…<achieve some goal>
Admin	Manage all account of all user	Create account for all Student, Teacher
	Manage all Course	Create New, Update information, delete, search

	Manage all Class	Create New, Update information, delete, search
	Manage all Student	Assign and remove student to class Update information, delete, search, create new
	Manage all Teacher	Create New, Update information, delete, search
	Manage all Room	Create New, Update information, delete, search
	Manage all Category	Create New, Update information, delete, search
Teacher	Check Course, Class, Room, Course Category, Student	See list course, class, student, Category and see detail course, class, Course Category Search Course, Class, Room, Course Category, Student, Teacher
Student	Check Course, Class, Room, Course Category,	See list course, class, Category and see detail course, class, Course Category Search Course, Class, Room, Course Category,

Figure 1: User requirement

III. System Design

1. Site map

Following is the sitemap of the system, the first is the home page, which always appears before and even after the user logs in with any type of account. Next is the login page, the user can log in with student and teacher accounts. If they are the system administrator, they need an administrator account to log in. The administrator manages all the accounts of the system users, managing courses and classes such as adding, editing, and deleting. While the rest of the account types can only view and browse listings of courses, classes, and other related information.

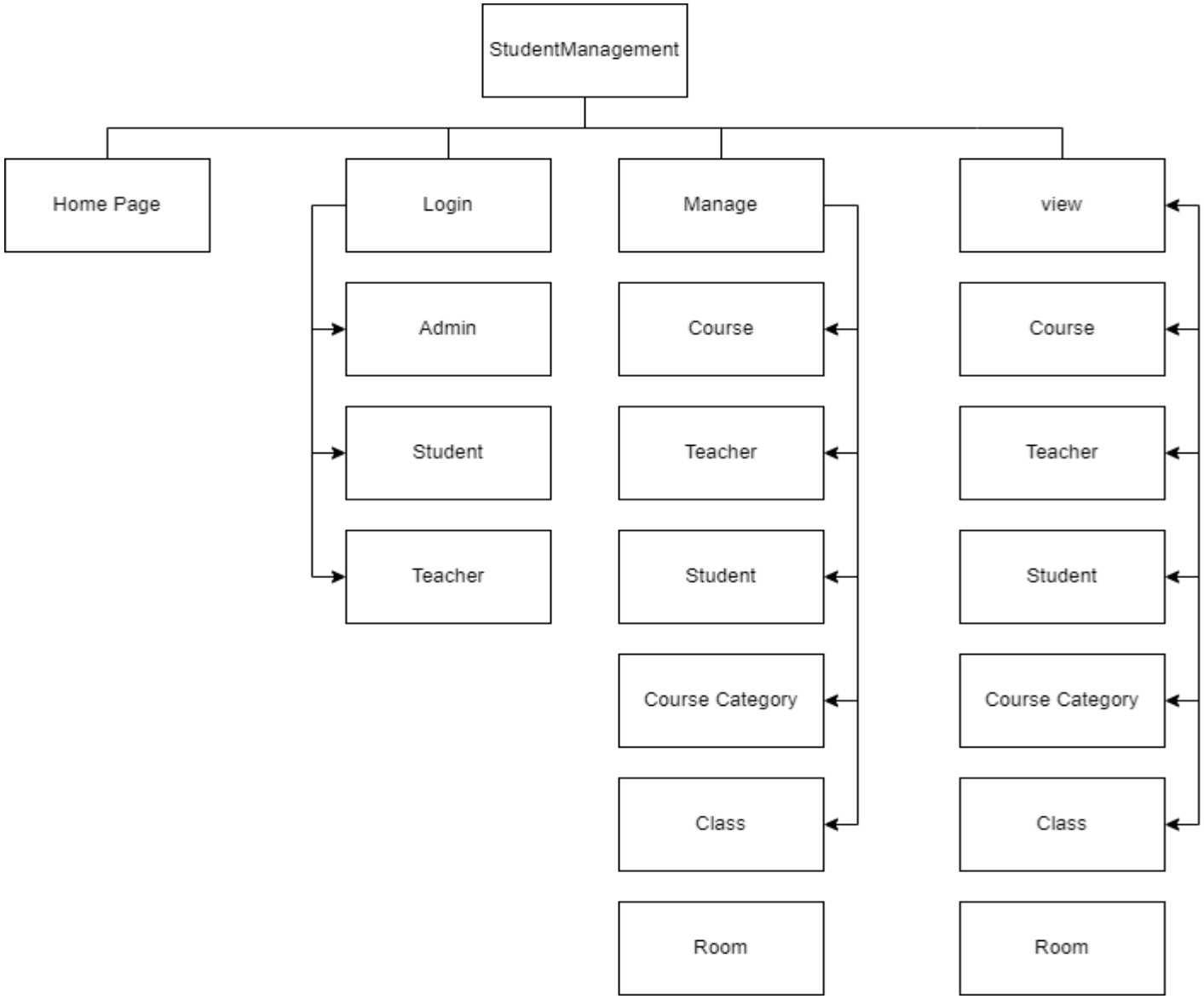

Figure 2: Site map

2. Entity Relationship Diagram

The following ERD shows the relationship between the tables in the database of the student management system. The system consists of seven tables, and all of these tables have a primary key of entity id and they are linked together through these keys. The first is the user table, which links one-to-one with the student and teacher tables because each user has only one account. The student table and the teacher table have a many-to-many relationship with the class table. This means that students can attend many classes and classes have many students, and so do teachers. The classes table has a many-to-one relationship with the courses table, and the courses table has a many-to-one relationship with the category table. This means that there are many courses in a category and a course has many classes. In addition, the teacher table also has a one-to-many link with the classroom table.

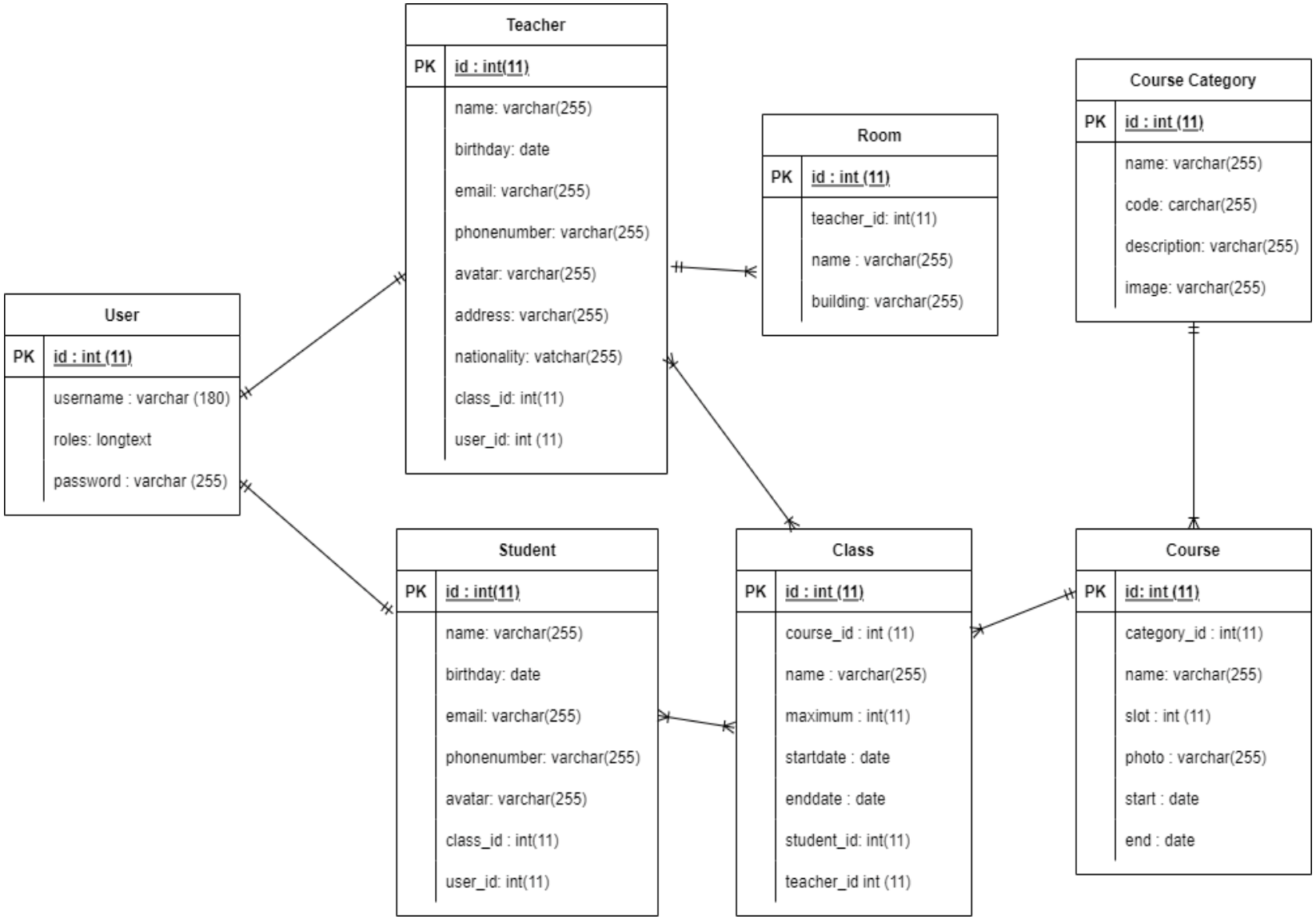

Figure 3: ERD diagram

3. Wire frame

In the project, the author builds student and class functionality, so in this section only relevant wireframes are presented. The first is the student index page, also known as the student view, which displays some common information about all students such as names and avatars.

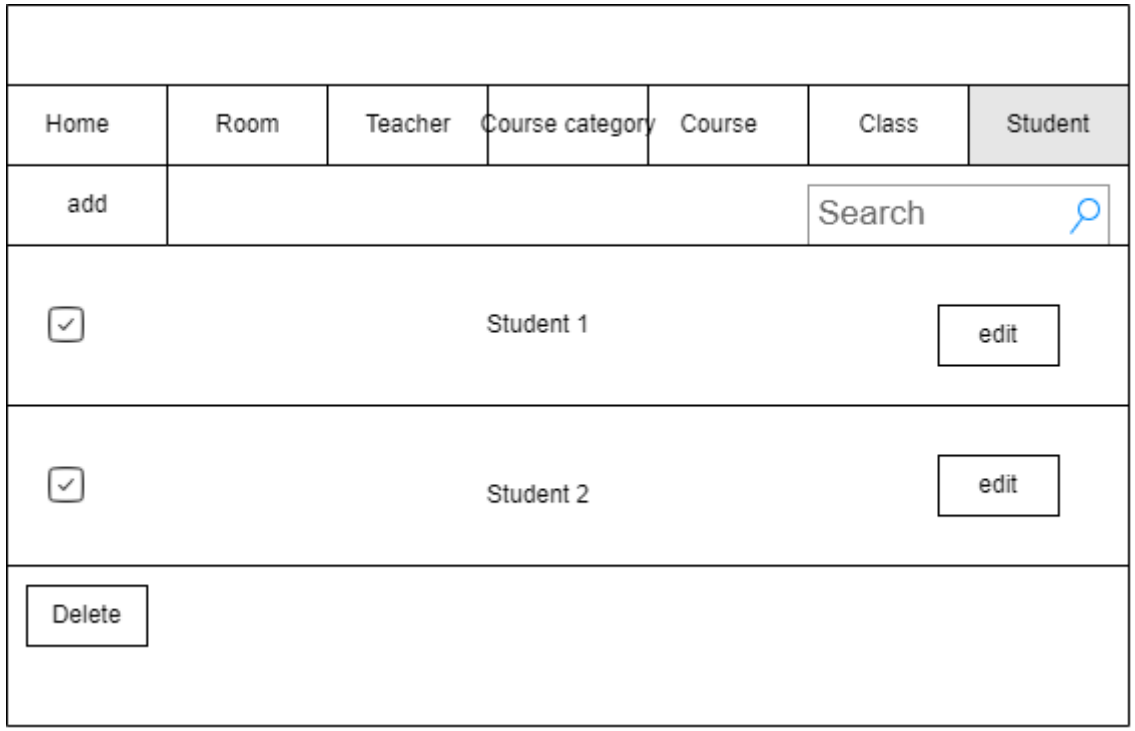

Figure 4: student view wireframe

Next is the student detail wireframe, this page displays more detailed information about the student such as personal information and classes participating.

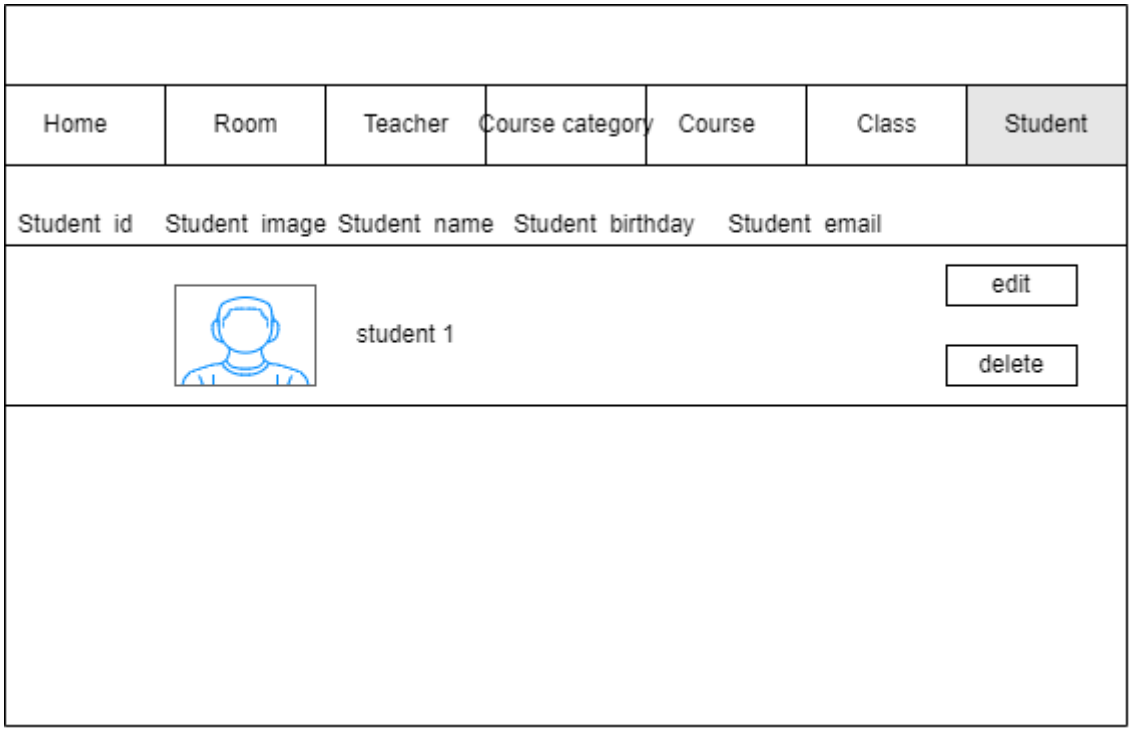

Figure 5: student detail wireframe

The add student wireframe page displays textboxes for users to enter data if they want to add a new student to the database.

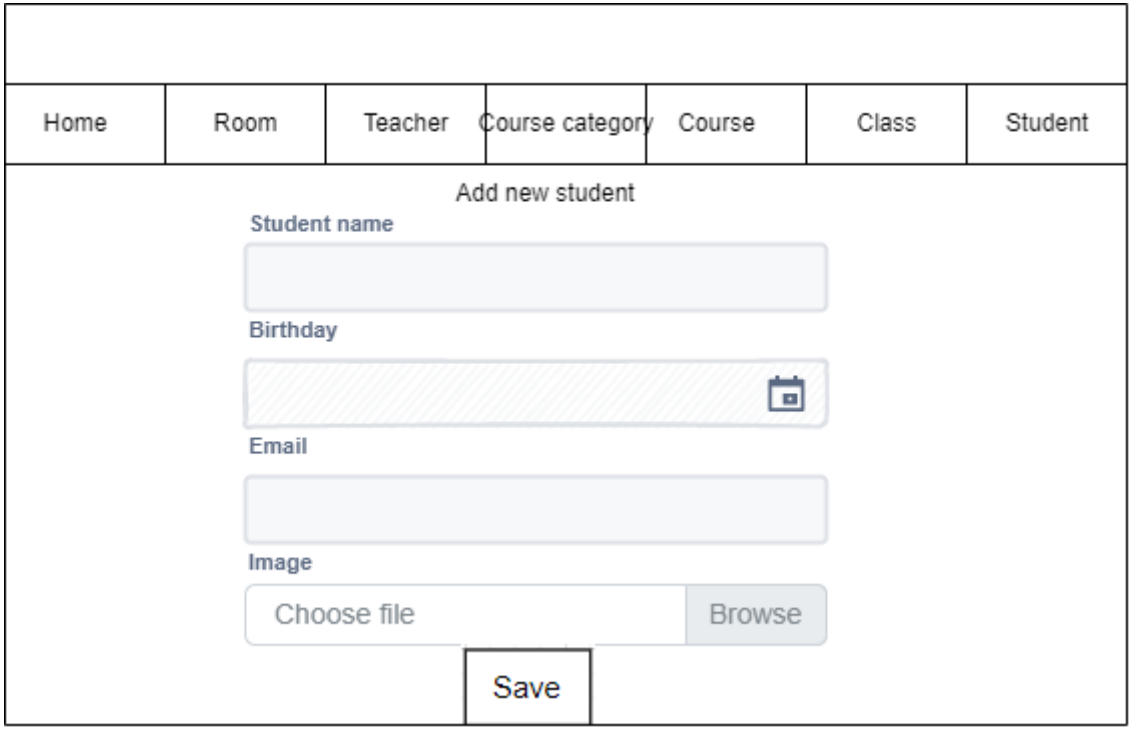

Figure 6: student add wireframe

The edit student wireframe page displays text boxes with student information for users to edit.

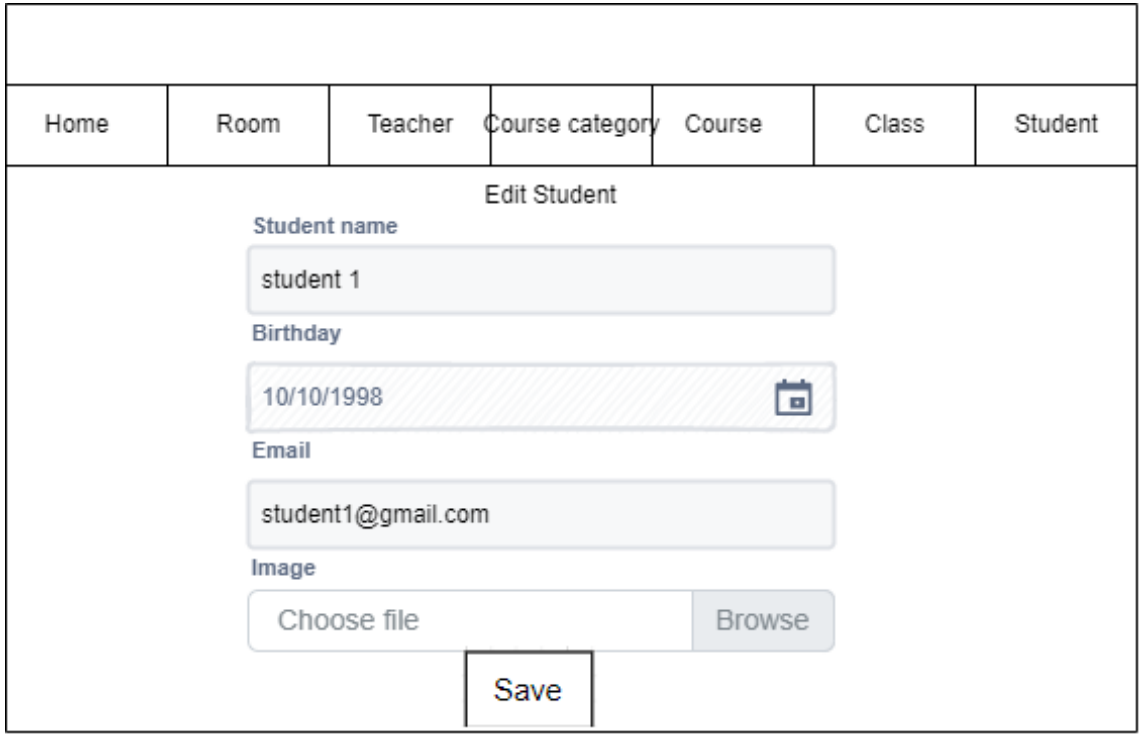

Figure 7: student edit wireframe

The following wireframes describe the pages related to the classroom interface. Basically, these wireframes also have the same functions as the students, so their design is also quite similar. Class wireframes include pages such as class view, class details, add class, and edit class. However, the class interface includes two additional pages that are the page to add students to the class and the page to remove students from the class. While the first page shows students who have joined the class so that administrators can remove them, the second page shows students who are not in that class so that they can be assigned in.

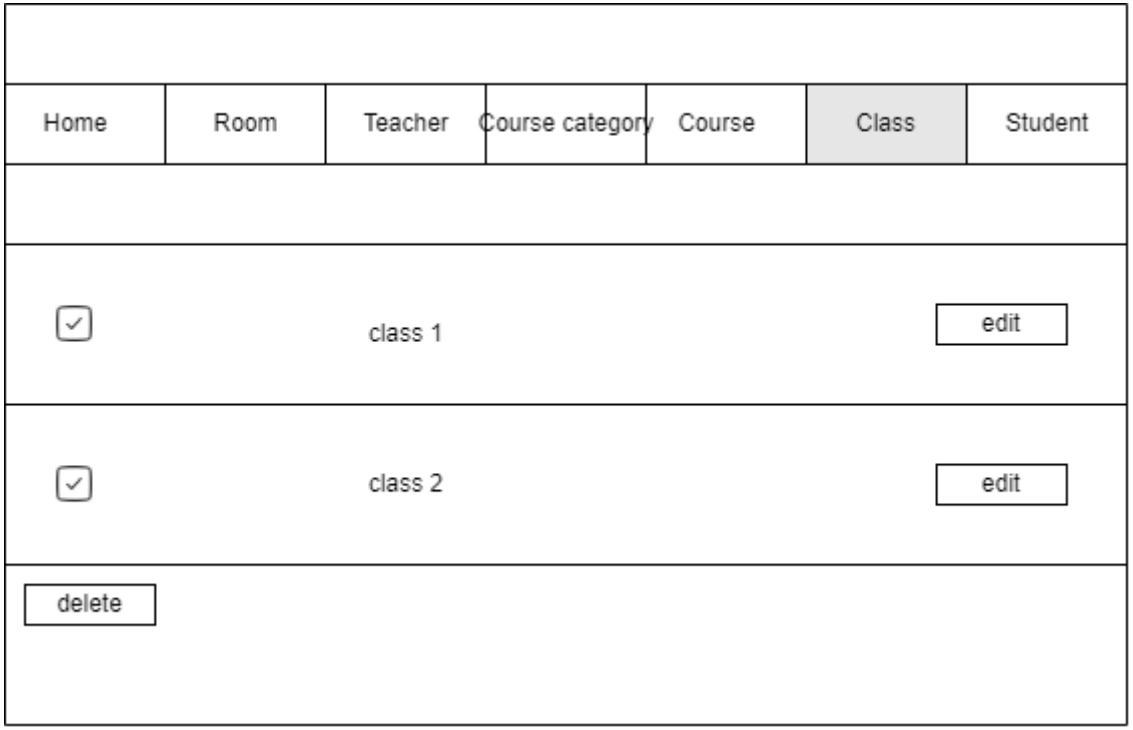

Figure 8: class index wireframe

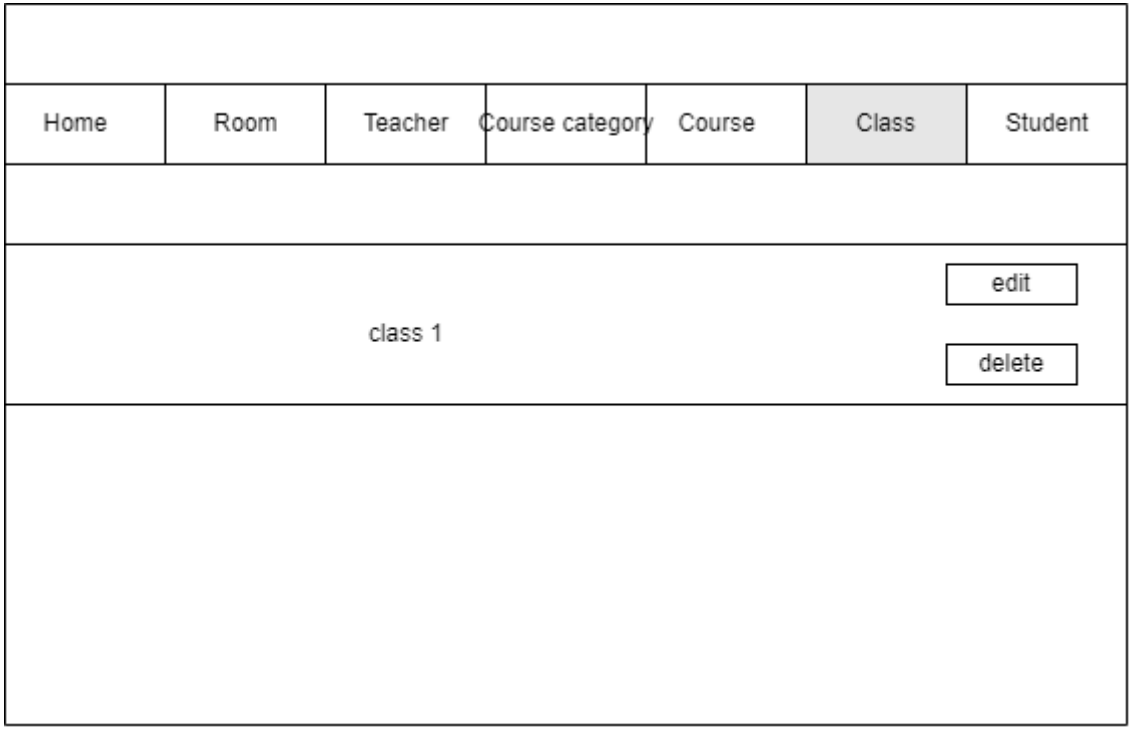

Figure 9: class detail wireframe

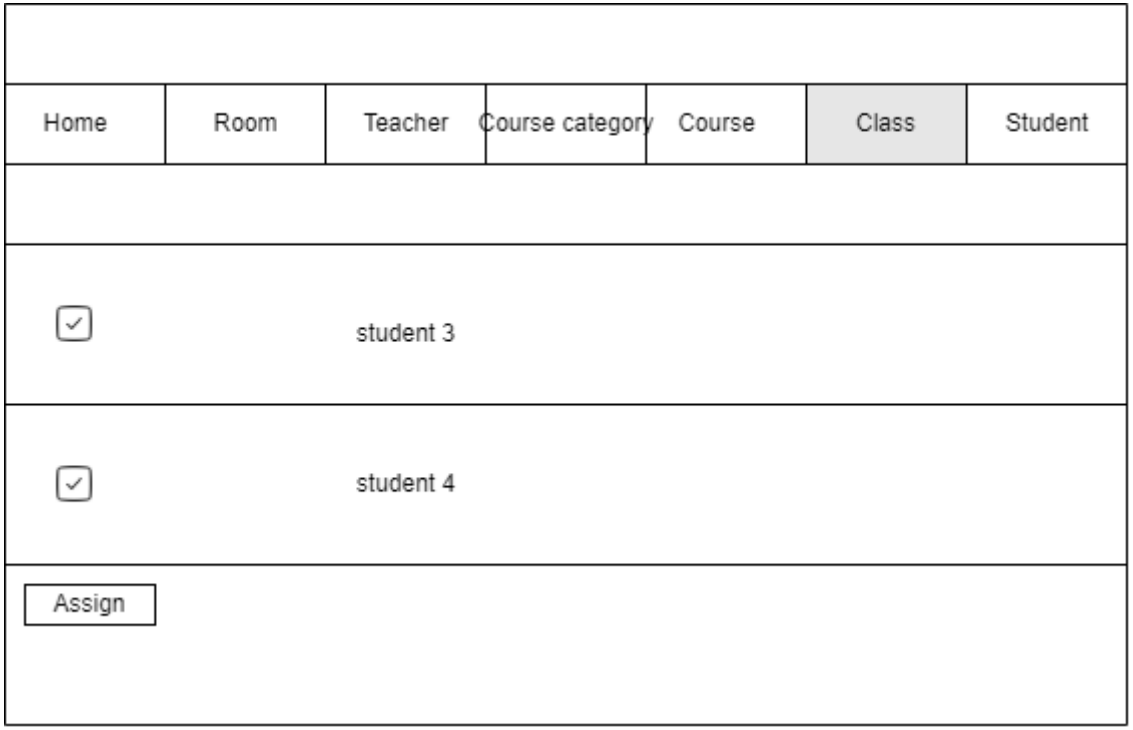

Figure 10: assign student to a class wireframe

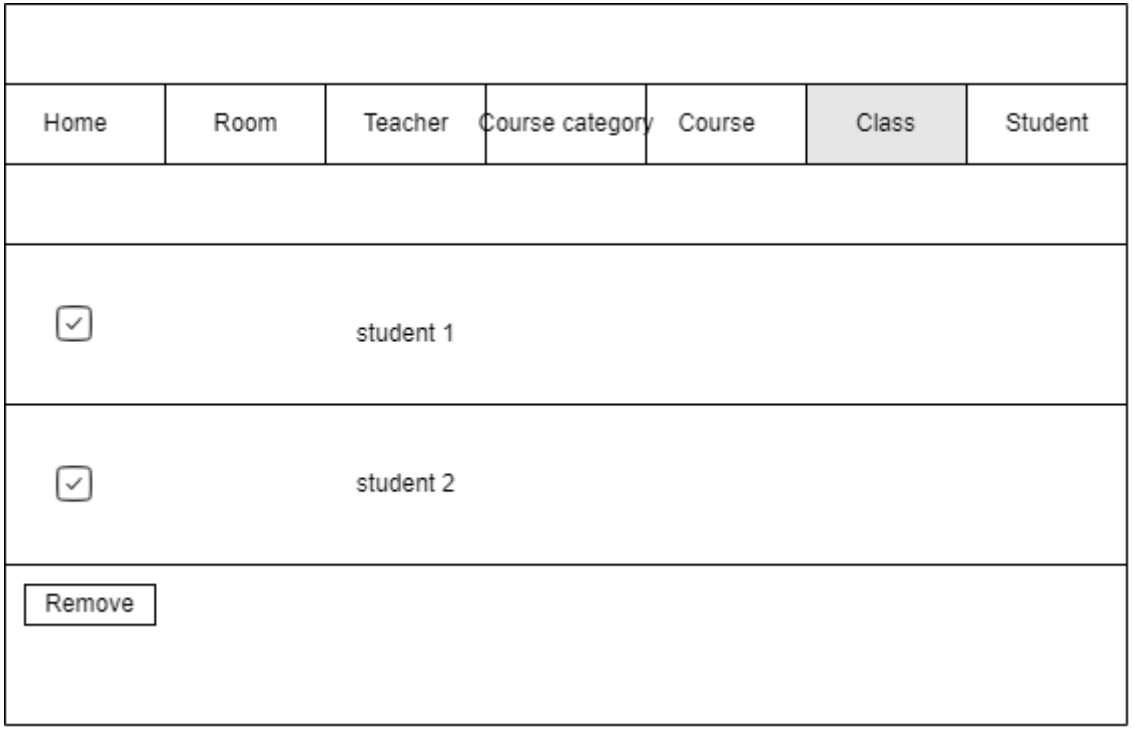

Figure 11: remove student from a class wireframe

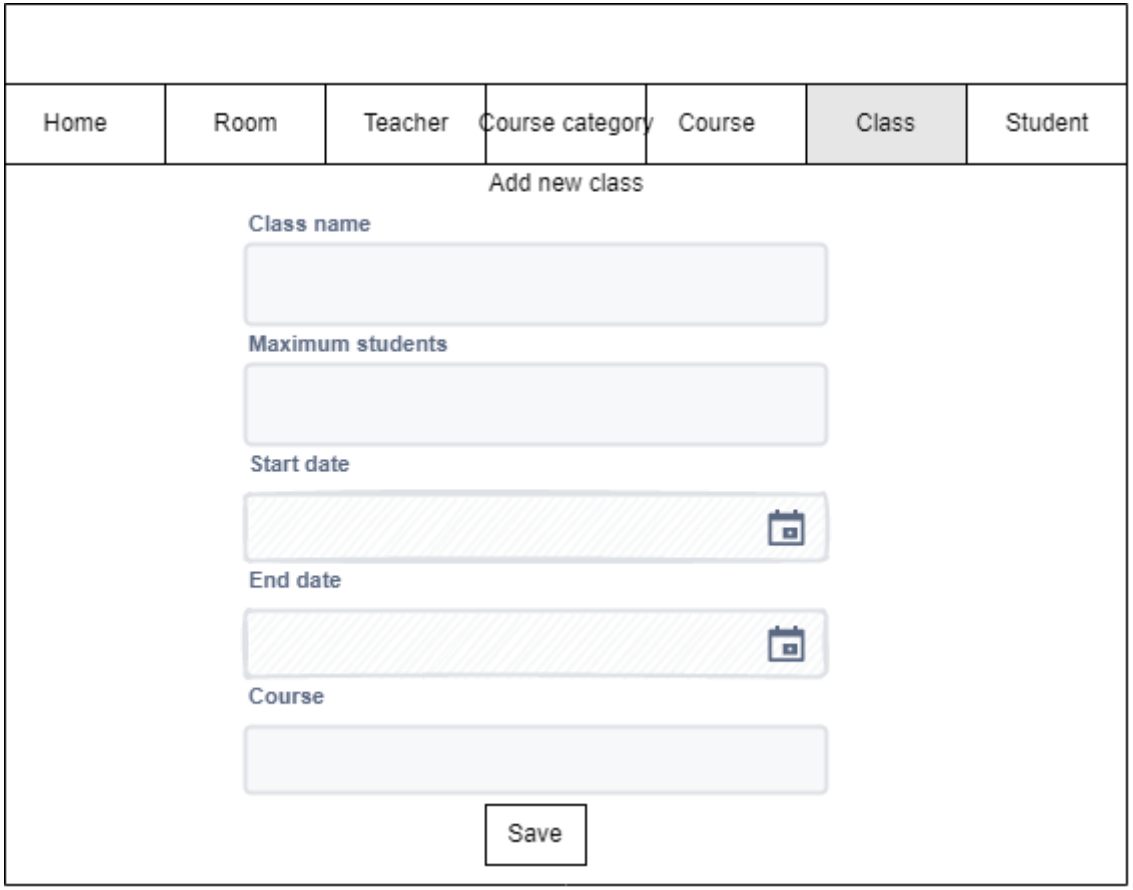

Figure 12: add new class wireframe

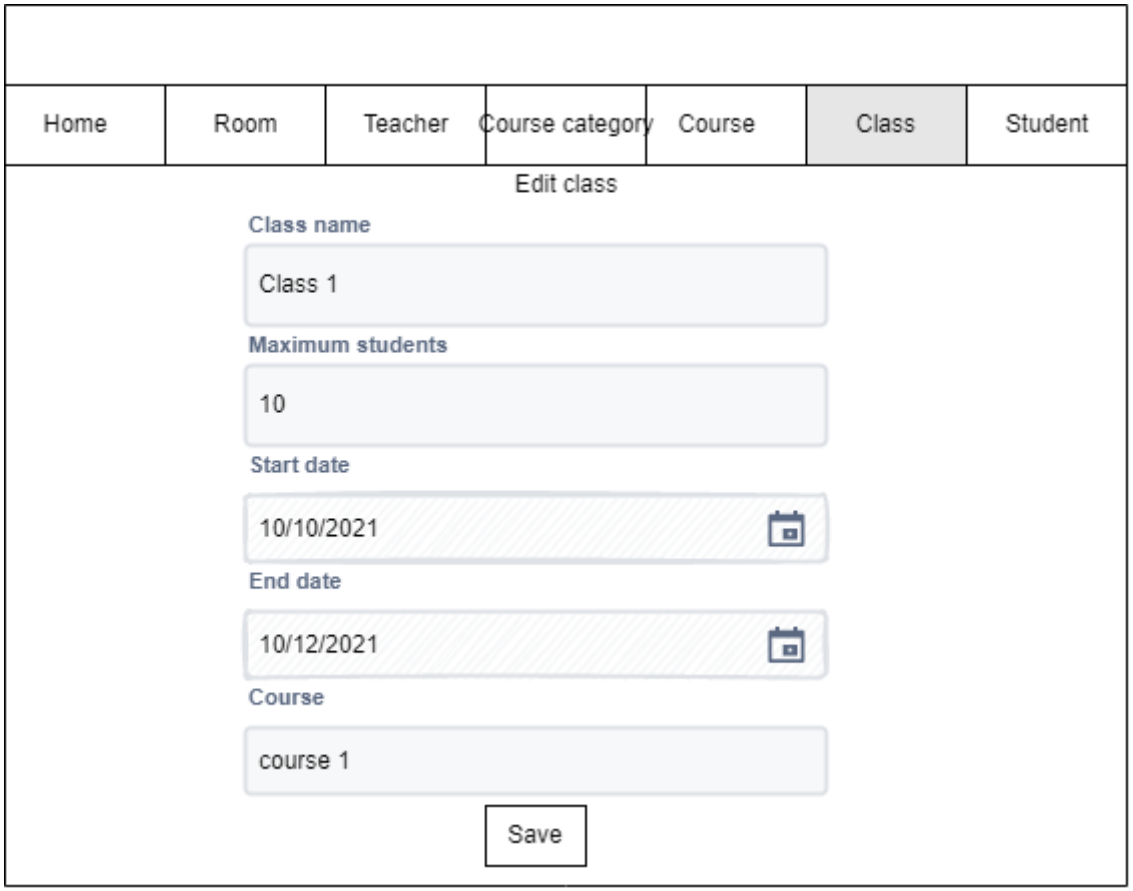

Figure 13: update class wireframe

IV. Implementation

1. Sample Source Code

The system uses the Symfony framework, so the image below shows the directory of the whole project that has been created with the Symfony command. Obviously, Symfony has divided folders according to the MVC pattern: model and controller in the src folder, and view in the templates folder.

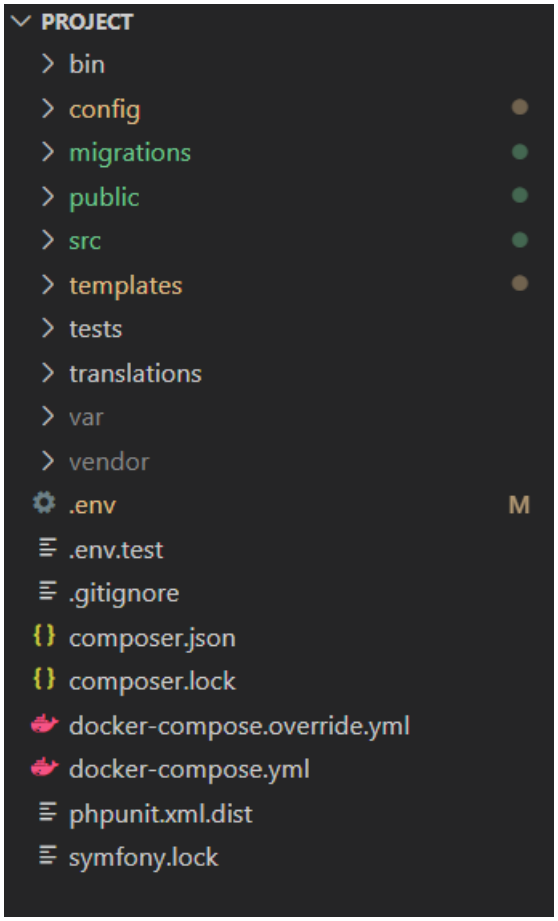

Figure 14: Source code folder

a. View

The MVC pattern, which is an acronym for 3 words Model - View - Controller. This is a design pattern used in software engineering. So, this model divides the source code into 3 parts, corresponding to each word of MVC. Each of these parts has a separate activity in the model. The first is the View, this is the part of the interface that helps users get or fill in data information through direct interactions when using the website. In symphony, this view is located in the "templates" folder. There are many twig files placed in each subfolder in the template folder. These subfolders are equivalent to their controllers. However, these twig files all carry the same functions such as displaying the index (rendering the home page

of an item in the navigation bar), displaying details (rendering the detailed information page of an entity such as a class or a course detail), add and edit (rendering a page that allows users to interact with and change the database).

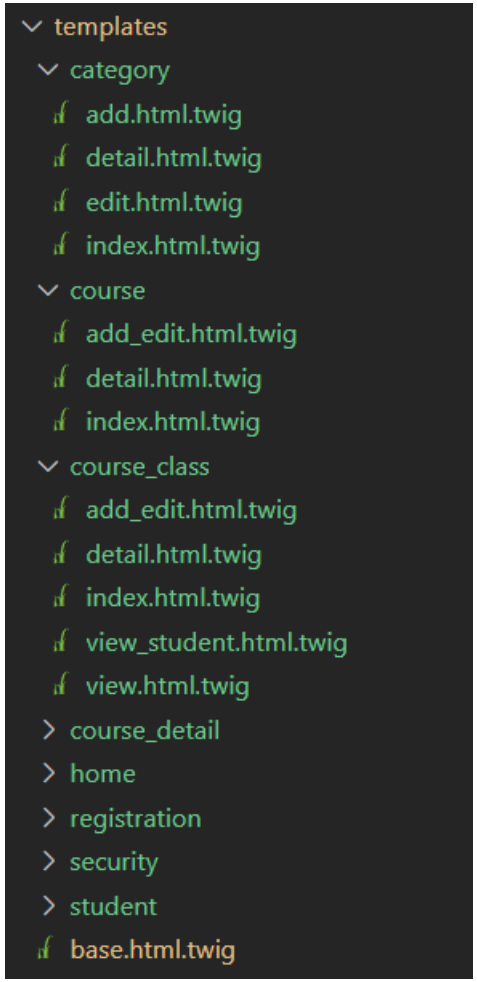

Figure 15: View folder

The following are some example of view. The code below is located in the file base.html.twig, which applies the bootstrap CSS and JavaScript libraries to all other twig files. All twig files can share the same CSS structures (called block) in this base file if it is invoked with the extents function. In addition, it is also possible to overwrite these blocks.

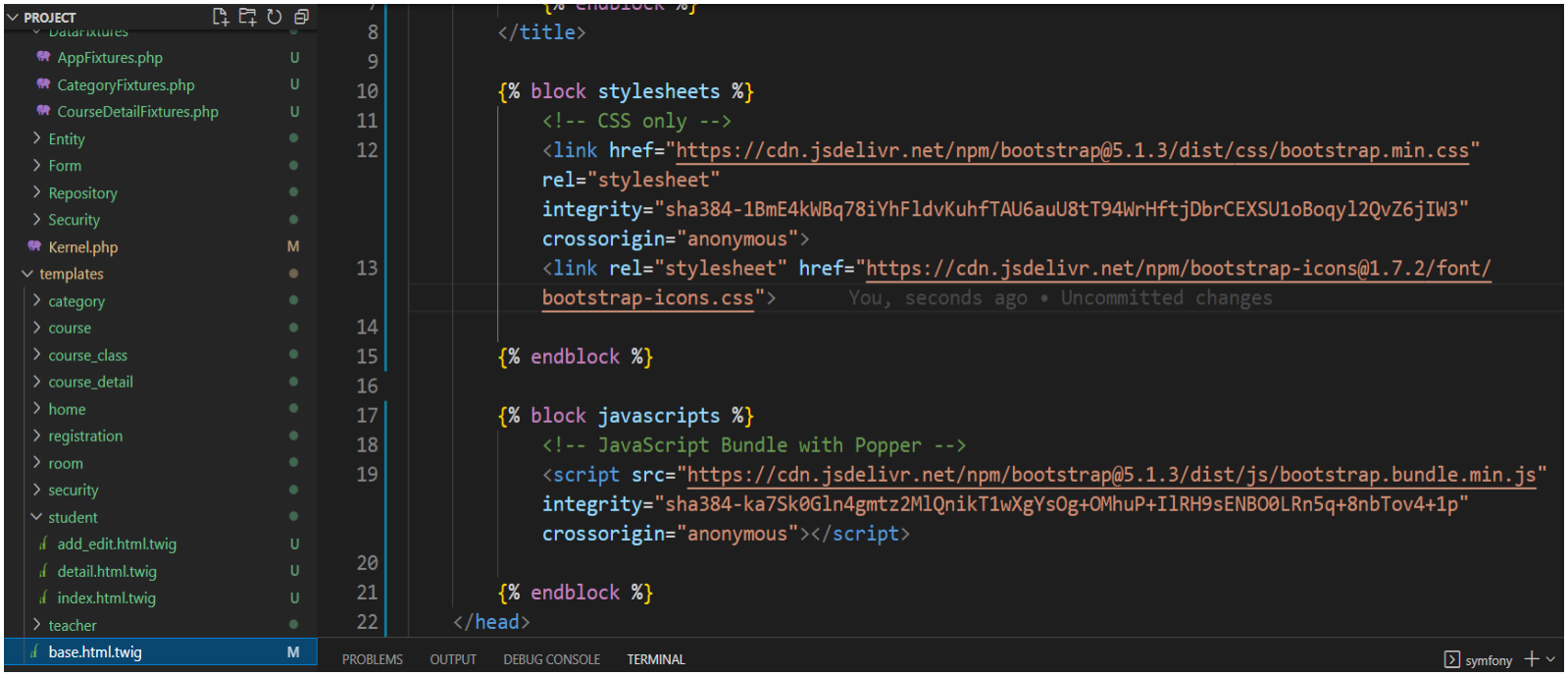

Figure 16: base.html.twig

The example below describes the function to display all students in the student home interface. Obviously, the controller has linked the view and model when it takes all the student data in the database and puts it in the array named student, and then renders it to the view. The student index page only needs to use the for loop to display each t student element in the array to the user.

Figure 17: Student index

b. Controller

Basically, in the MVC model, the controller acts as a middle layer between the view and the model, which is responsible for handling user requests through the view. The controller connects to the model and outputs the appropriate data to the user by receiving commands from the view and sends them to the model and vice versa. The project is divided into seven main controllers related to system functionality, located in the Controllers folder. These file controllers contain all control flows, called routers when the user uses functions like navigate, add, edit, delete, etc.

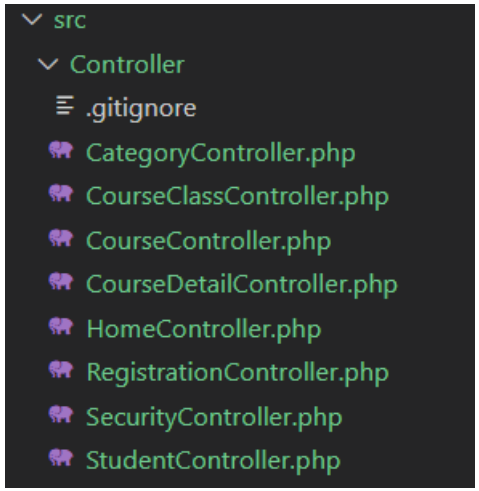

Figure 18: Controller folder

An example in the student's controller, two functions studentIndex and studentDetail are responsible for getting data from the Student table in the database and rendering it in the view.

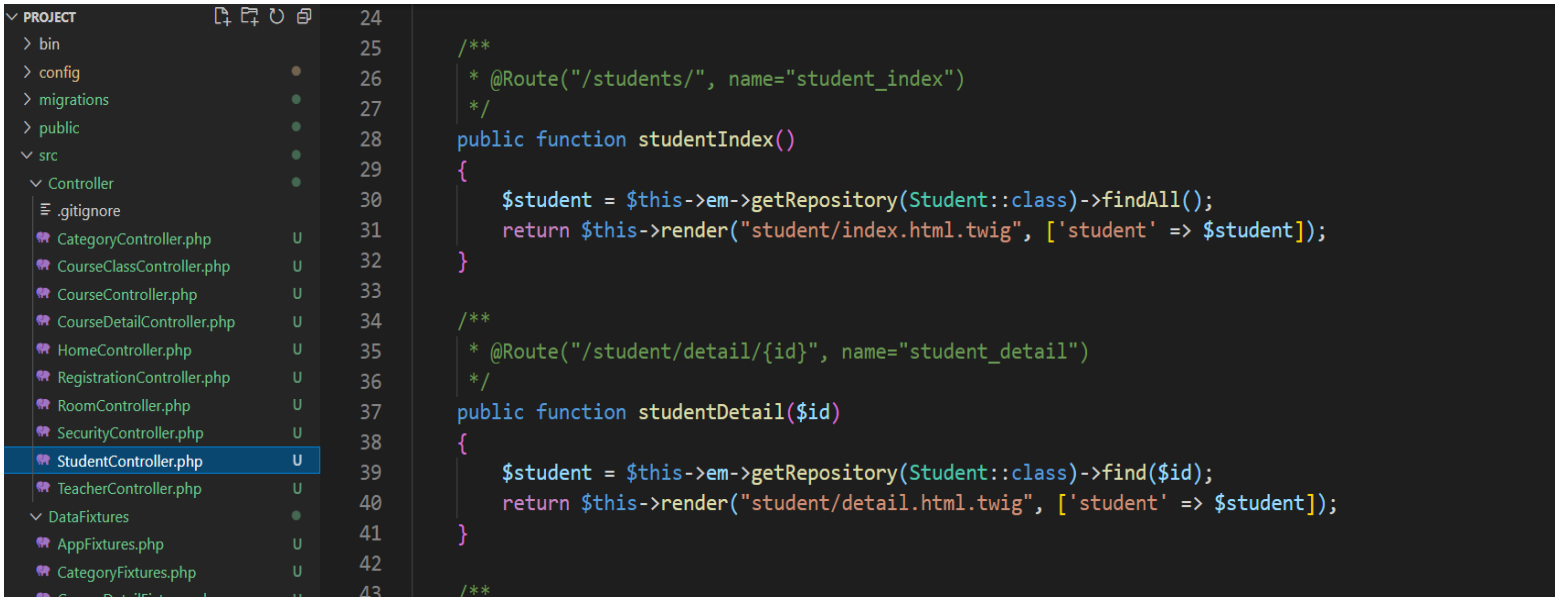

Figure 19: Student controller

c. Model

In simple terms, in the MVC pattern, this model handles and accesses operations in the database. In Symfony, the Model can be auto-generated with commands such as 'symfony make: entity'.

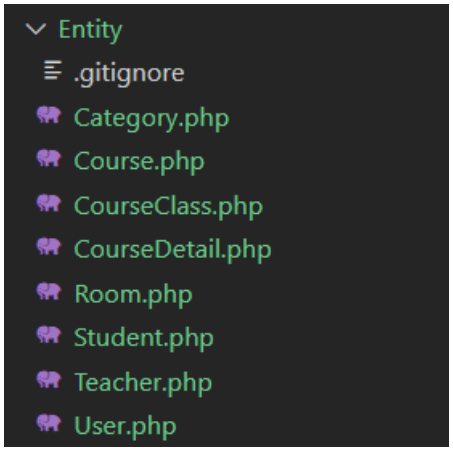

Figure 20.1: Entity folder

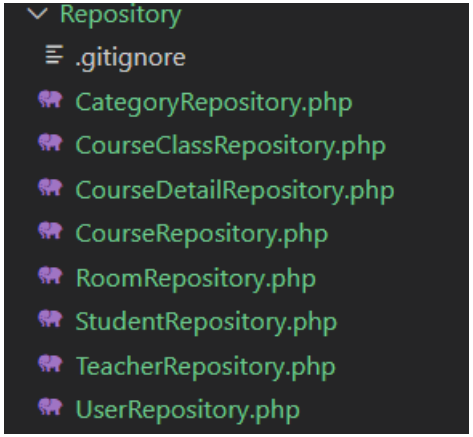

Figure 20.2: Repository folder

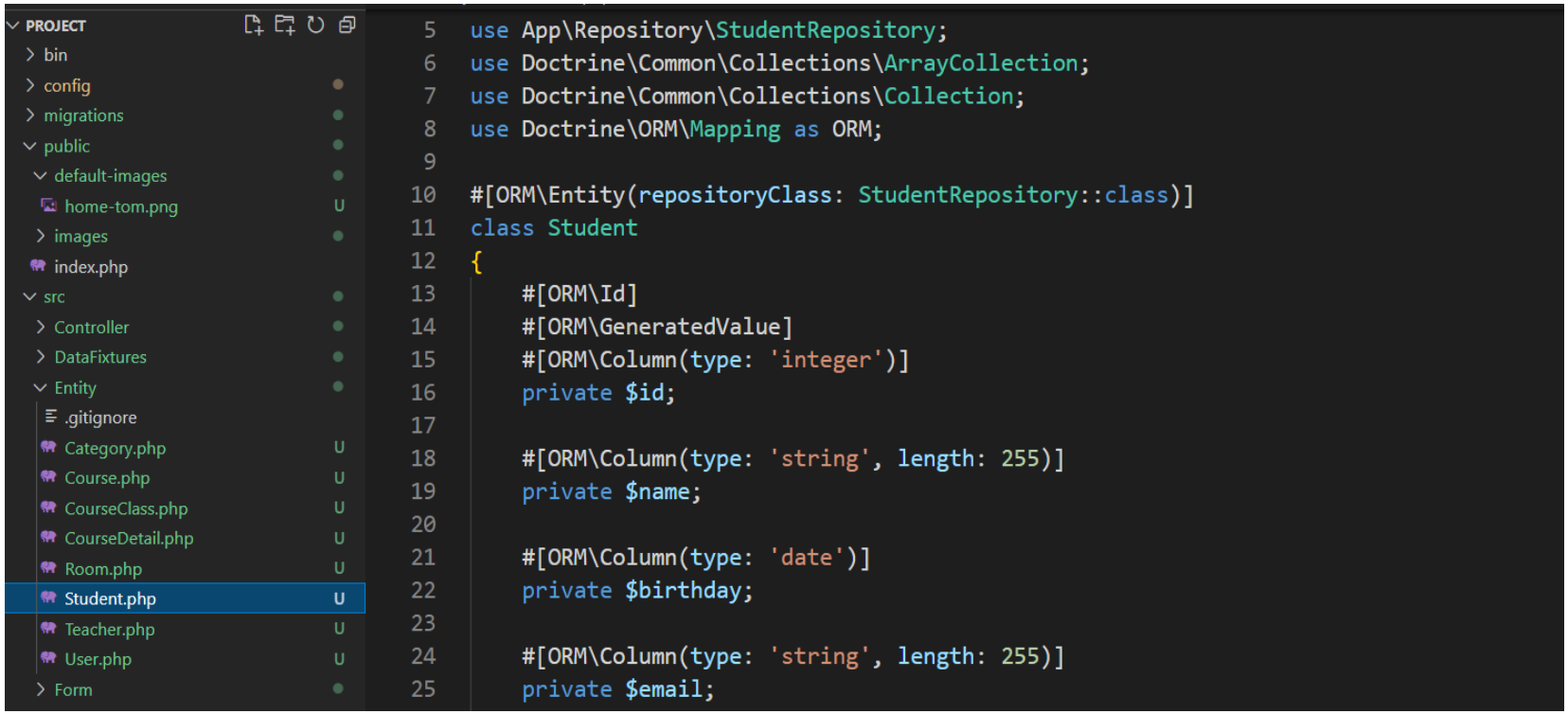

Figure 21: Student entity

Figure 22: Student orm

2. Web screenshots

The first page is the homepage, the user can move to other pages using the navigation bar.

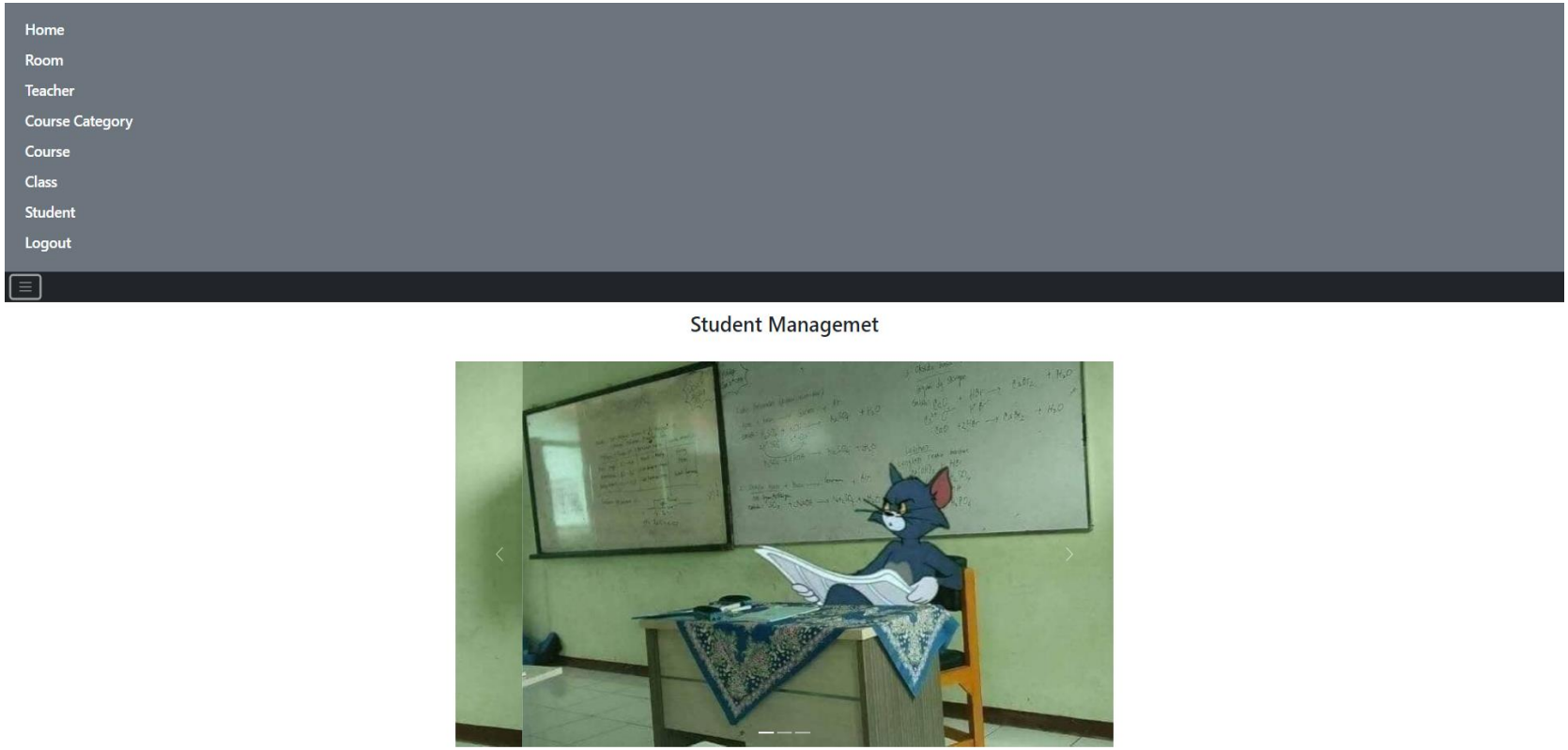

Figure 23: Home and navigation bar interface

After clicking on the student item in the menu bar, the system will redirect the user to the student index page.

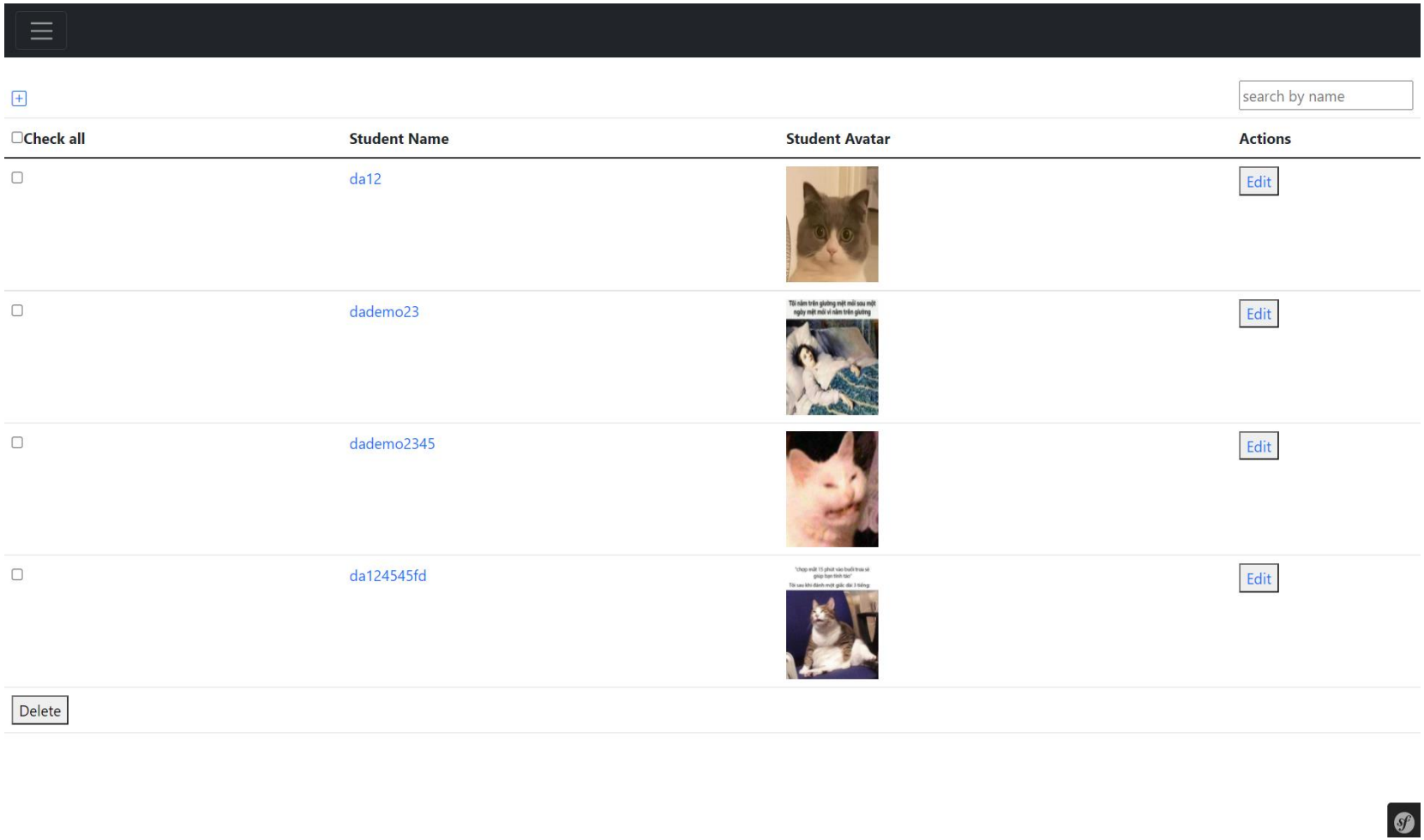

Figure 24: View Student

If the admin wants to add a new student to the database, click the plus icon in the upper left corner under the navigation bar, the system will move to the add student page. The user must fill in valid information otherwise the system will ask to re-enter. Some student information may be left blank as it can be added to the edit student page. After filling in the information and pressing the save button, the system will redirect the user back to the student index page.

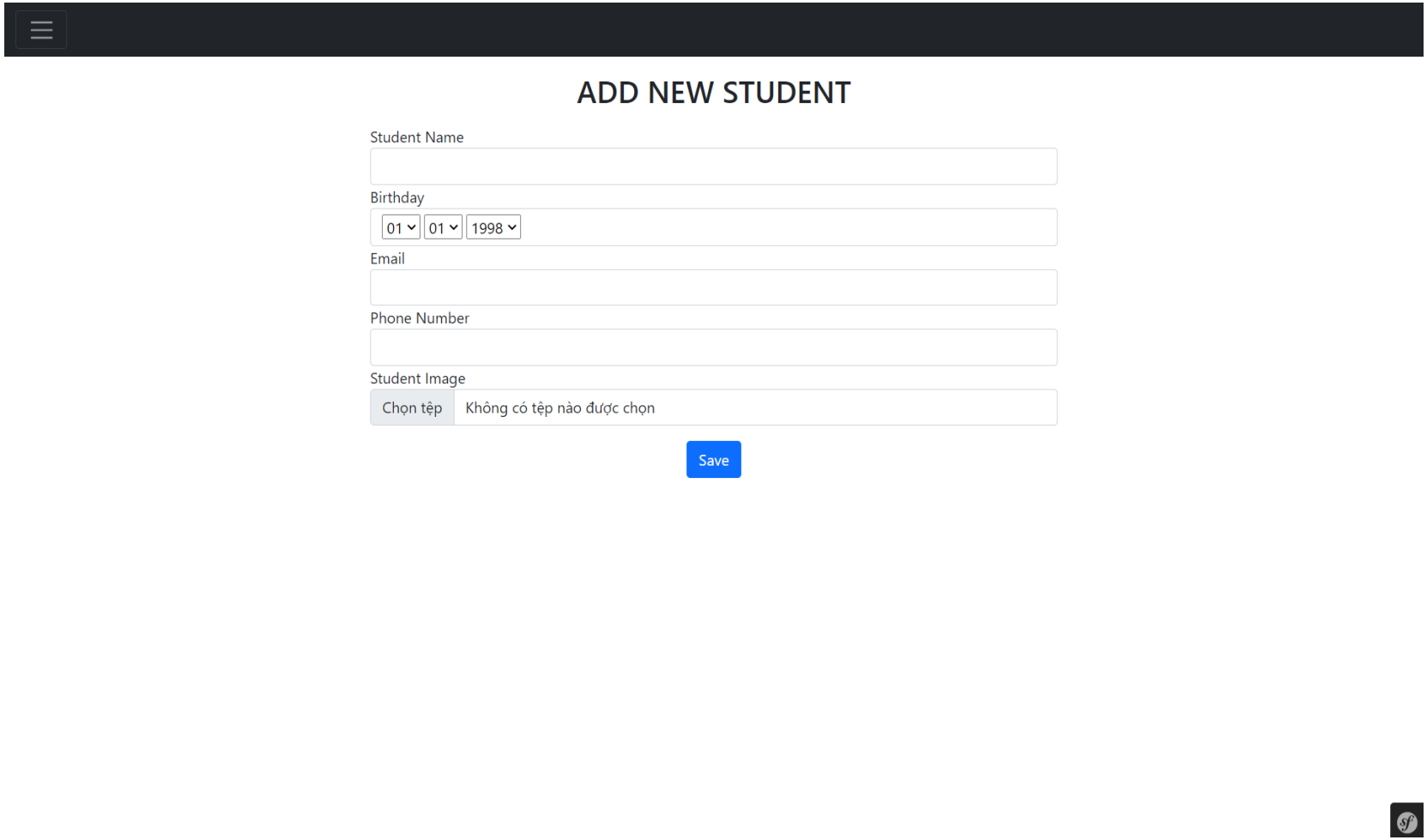

Figure 25: Add student

If the admin wants to update student information, click the edit button corresponding to that student. Similar to the add page, the edit page also includes a form with student information. Users edit and press save to change student data.

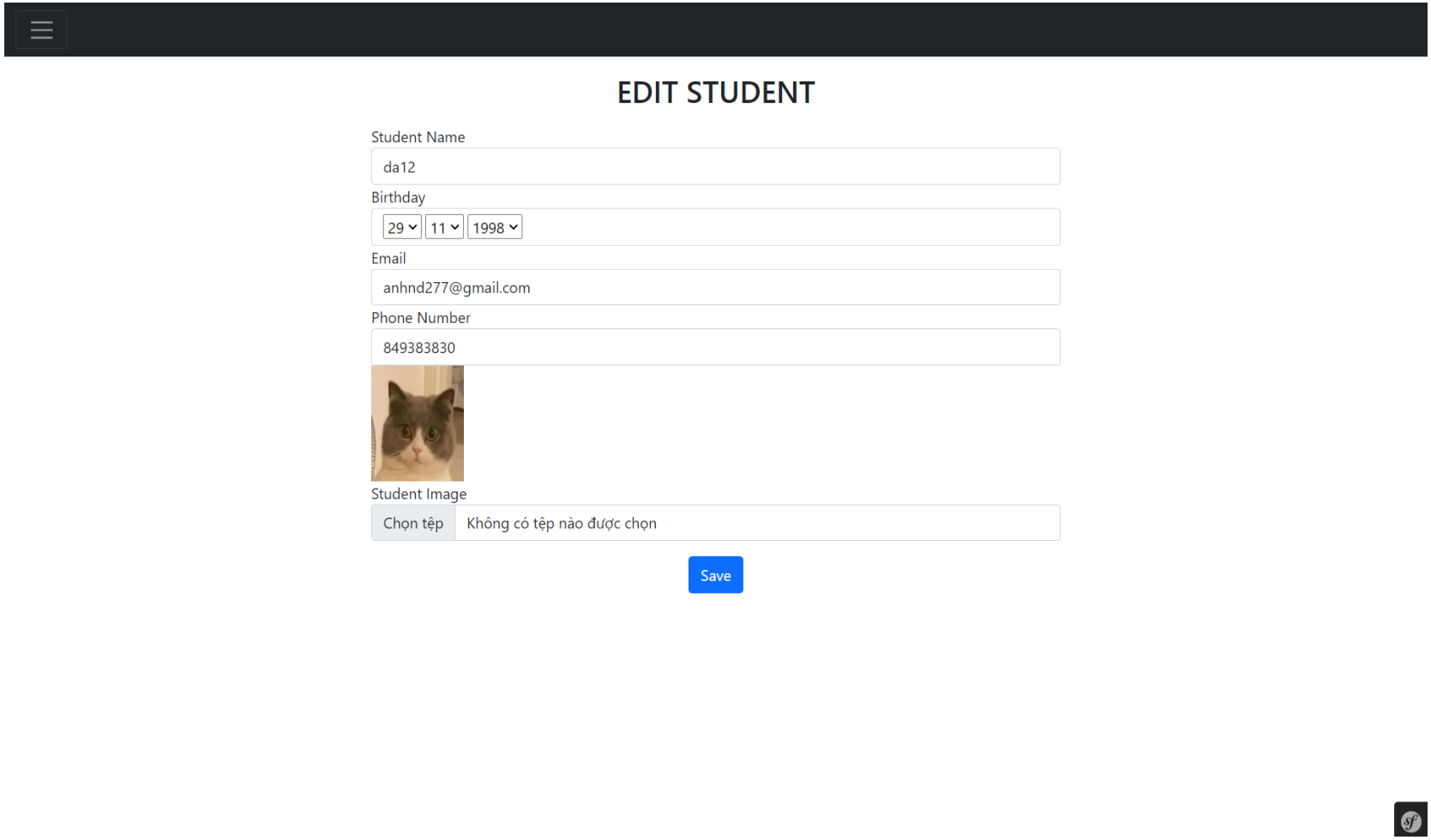

Figure 26: Update student

To view detailed information about a student, the administrator just needs to click on the student's name to view. The system will display a page with more detailed information about that student, including the class that the student is attending.

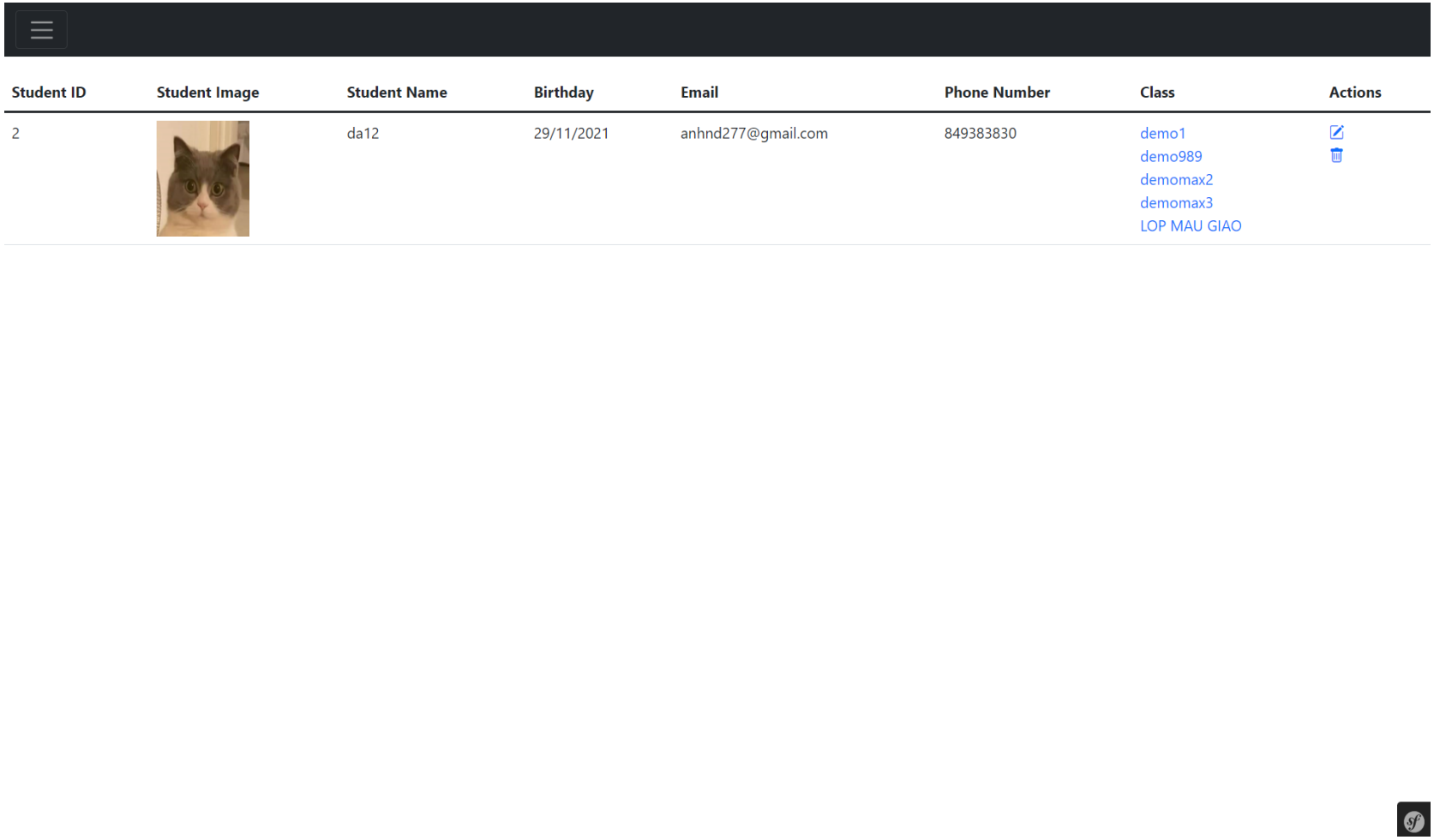

Figure 27: Student detail

Click on the class item in the navigation bar to move to the class index page, which displays all the classes in the database. Similar to the student page, this page also includes add, edit and delete functions.

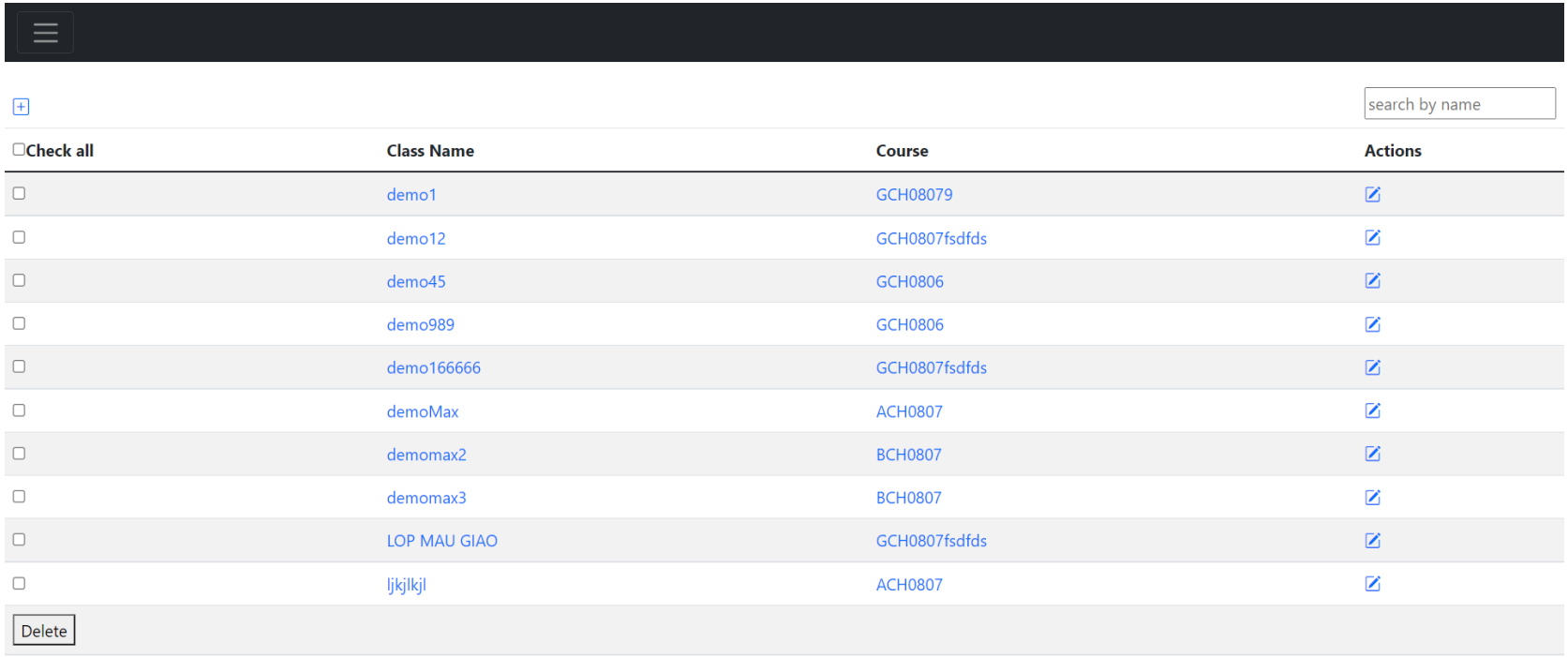

Figure 28: View class

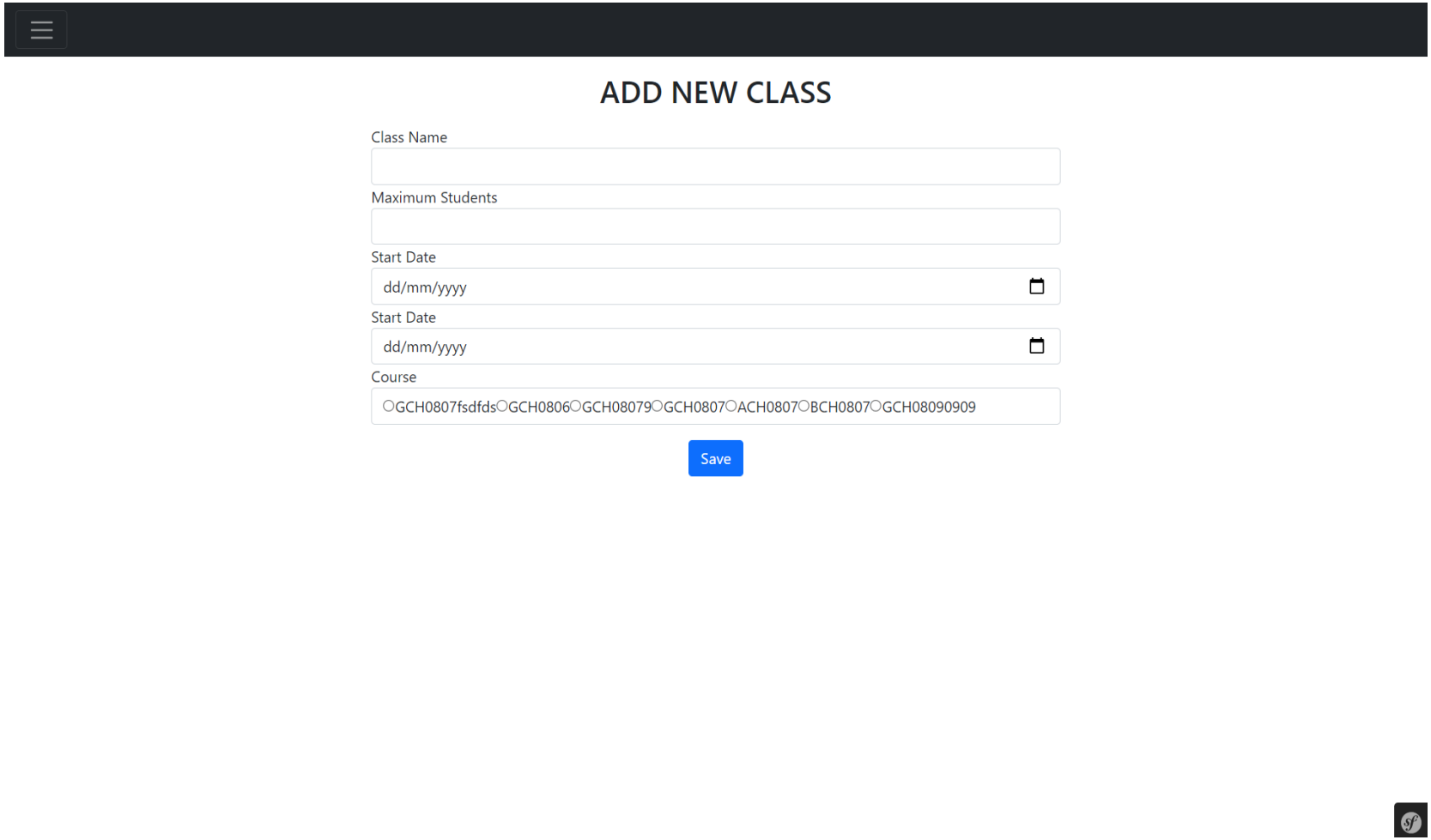

Figure 29: Add class

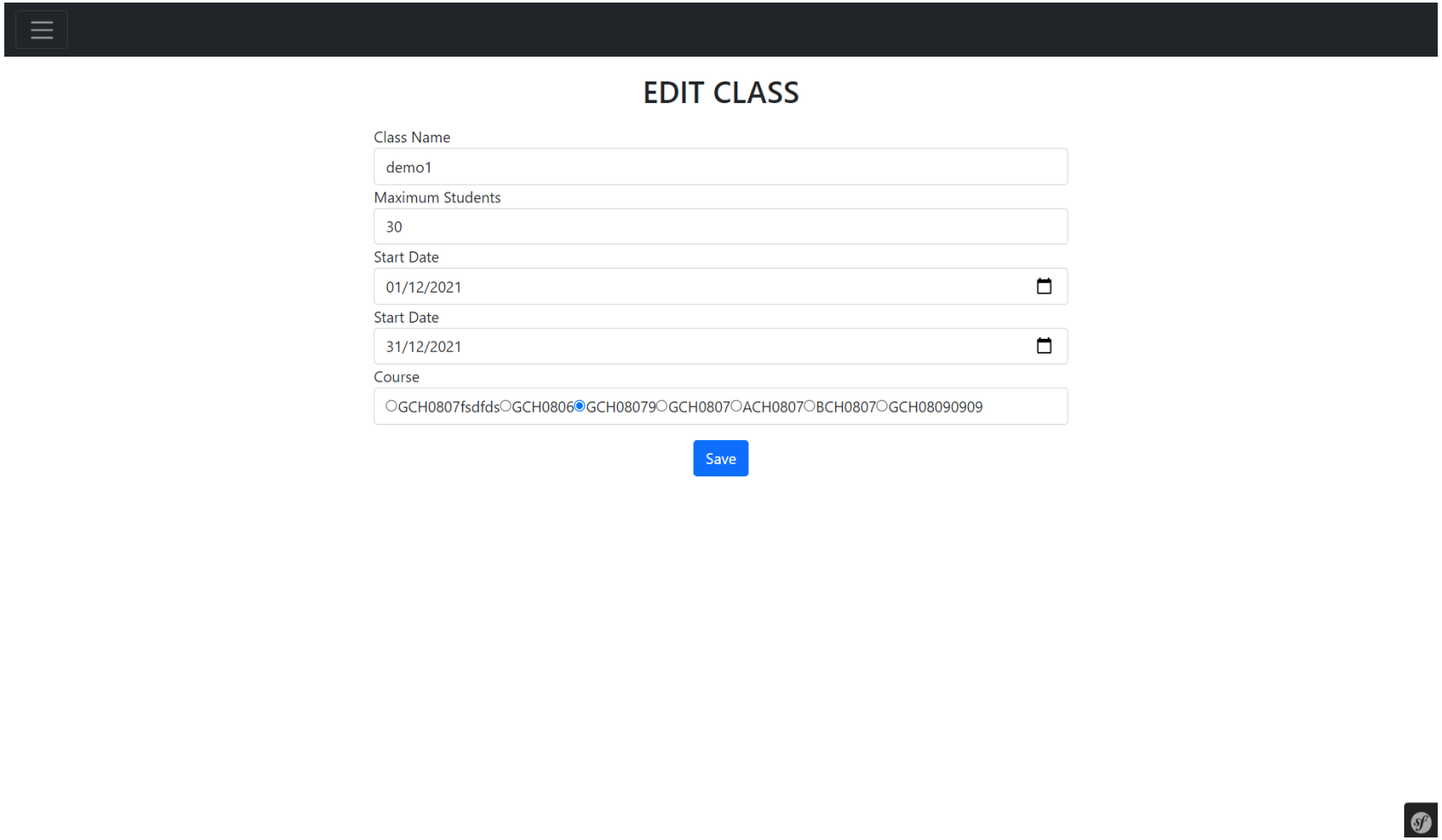

Figure 30: Update class

Click on the class name to display more detailed information about the class, including which course the class belongs to or the students in that class.

Class ID	Class Name	Course	Maximum Student	Number Students	Start Date	End Date	Students	Actions
4	demo1	GCH08079	30	3	01/12/2021	31/12/2021	view	

Figure 31: Class detail

In the student section of the class detail, click on the view link to move to the student management page in the class. This page displays the students in the class, the administrator can remove them by checking the checkbox and pressing the remove button.

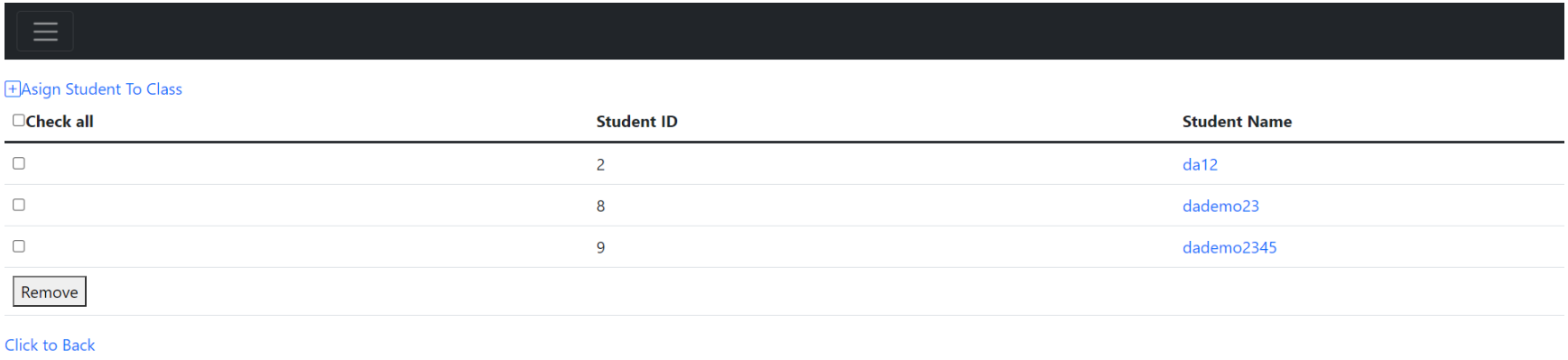

Figure 32: remove student to a class

To add a new student to that class, click the Assign Student to class button at the bottom of the navigation bar. Then a web page showing students who haven't attended the class appears. The user selects the student and then presses the assign to class button.

Check all	Student ID	Student Name	Student Course
☐	2	da12	demo1 demo989 demomax2 demomax3 LOP MAU GIAO
☐	8	dademo23	demo1 demomax3 LOP MAU GIAO
☐	9	dademo2345	demo1 LOP MAU GIAO
☐	11	da124545fd	

Assign To Class

Figure 33: assign student from a class

V. Conclusion

1. Evaluate

Strength	Weaknesses
• The website has met the minimum requirements of a student management system such as: adding, updating, editing, deleting searching. • The system is built according to the MVC model (Model, View, and Controller). • Website is easy to use, and all functions can be performed by no more than four operations. • In the process of running the website, there are no errors, the functions work very efficiently. • The website has authorization, access denial, and password encryption. • Easy to modify and improve in the code.	• Low security. • There are no special functions other than simple functions such as add, edit and delete. • Simple and boring interface. • Users other than admin can only view. • Some tables in the database are not connected properly. • Some features are not logical.

Figure 34: Pros and Cons

2. Future Improvement

In the future, the system will be added with more features such as grading function, student attendance, assignment submission, etc. In addition, the team will improve the user interface of the web to make them look better and more interactive. In the near future, the team will try to correct the shortcomings and irrationalities in the system to build a complete website.

VI. Appendix

Group member list	Role
Nguyen Duc Anh (GCH17051)	User Login, Student and Class function back-end and front-end
Pham Khue	Room and Teacher function back-end and front-end
Nghiem Cao Nhan	Course Category and Course function back-end and front-end

Figure 35: Member list

Link GitHub: https://github.com/paulpham98/projectWeb