

Course Management System Report

Introduction to Computer Science II

23APCS2 | Group 6



Last Modified Day:

May 2, 2024

Table of Contents

[A. Abstract 3](#_Toc165603121)

[1. Overview: 3](#_Toc165603122)

[2. Key features: 3](#_Toc165603123)

[B. Introduction 4](#_Toc165603124)

[1. Background: 4](#_Toc165603125)

[2. Objective: 4](#_Toc165603126)

[C. Group Information 5](#_Toc165603127)

[Group ID: 6 5](#_Toc165603128)

[Member: 5](#_Toc165603129)

[1. Nguyen Thu Uyen – 23125048 (**34%**) 5](#_Toc165603130)

[2. Tran Trong Nguyen – 23125087 (**36%**) 5](#_Toc165603131)

[3. Doan Duc Tuan – 23125021 (**30%**) 6](#_Toc165603132)

[D. Data Storage 7](#_Toc165603133)

[**Green:** School year management 7](#_Toc165603134)

[**Red:** Semester management 7](#_Toc165603135)

[**Purple:** Account management 8](#_Toc165603136)

[**Blue:** Class management 8](#_Toc165603137)

[**Orange:** Course management 8](#_Toc165603138)

[Rationalize data storage decisions: 8](#_Toc165603139)

[E. Project Architecture 10](#_Toc165603140)

[F. Implementation Detail 11](#_Toc165603141)

[Backend: 11](#_Toc165603142)

[FrontEnd: 17](#_Toc165603143)

[UI functions: 17](#_Toc165603144)

[Staff: 17](#_Toc165603145)

[Student: 19](#_Toc165603146)

[G. Technical Problems and Solutions: 19](#_Toc165603147)

[About user interface 19](#_Toc165603148)

[About major errors (reading access violations and other errors) 20](#_Toc165603149)

[H. Feature Description: 20](#_Toc165603150)

[Login screen: 20](#_Toc165603151)

[**Staff view:** Navigation bar and tool bar: 21](#_Toc165603152)

[School year management: 22](#_Toc165603153)

[Semester management: 23](#_Toc165603154)

[Course management: 23](#_Toc165603155)

[Class management: 25](#_Toc165603156)

[**Staff view:** Navigation bar and tool bar: 27](#_Toc165603157)

[Choose option 28](#_Toc165603158)

[Profile and change password 28](#_Toc165603159)

[I. Conclusion 29](#_Toc165603160)

[References 29](#_Toc165603161)

# Abstract

## Overview:

In this project, we chose to create an application using C++ and SFML libraries.

The project aims to support staff in organizing, creating, deleting, and modifying academic terms such as school years, semesters, courses, and to provide students with an overview of such contents.

We had developed and implemented all the functionality required, albeit not being able to optimize and having a clean code user interface.

## Key features:

* School year management:
  + View school year list.
  + Create, delete school year.
* Semester management:
  + View semester list.
  + Create, delete semester.
  + Modify semester’s information.
* Course management:
  + View course list.
  + Create, delete course, modify course’s information (Course ID, Course name, etc.)
  + Course’s Student Management: Enroll, withdraw student, modify student’s information, view student list.
  + Export student list.
  + Course’s Scoreboard Management: Import, view scoreboard, modify student’s scores.
* Class management:
  + View class list.
  + Create, delete class.
  + Class Student Management: Enroll, withdraw student, modify student’s information.
  + Class’s Scoreboard Management: View scoreboard, modify student’s scores.

# Introduction

Course Management System is a group project that requires significant effort on the part of the students. Students are required to implement this application using char[], char\*, string, linked lists, and dynamic allocated arrays. Database and vectors are not allowed.

This report aims to provide a detailed look at the resulting application with some analysis of the implemented design.

## Background:

At the beginning of the project, we had no previous experience with teamworking in coding and developing an application, apart from Tran Trong Nguyen prior Minesweeper solo project. Thus, a large portion of time was dedicated to studying and familiarizing with Github, SFML libraries, Figma for designing UI, class concepts, constructors, and destructors, etc. We had also looked for alternative implementations to build better solutions.

Ultimately, we ended up teaching ourselves SFML implementations, basic coding concepts, and designing, with Tran Trong Nguyen help in SFML. As a result, a significant part of the UI code are quite repetitive, thus optimization is needed in the future, and a majority of the functions we implemented are redundant. However, most of the project coding is made by us, with no adopted codes.

## Objective:

Over the last few years, due to the rise of technologies, digitalized paper documents and records are crucially needed to store, archive and access informations. The ability to manage and update learning content has become important for schools and universities to provide easy access for staff and students.

CMS is an application designed to bring academic staff members, students, and course information together in one place, reduce managing costs, and allow easy access and efficient/effective administration of information in a user-friendly environment. We strive to prevent human errors and provide comfortable, easy access working space for higher accuracy.

# Group Information

## Group ID: 6

## Member:

### Nguyen Thu Uyen – 23125048 (**34%**)

* Class implements:
  + Course
  + Semester
  + Schoolyear (manage semester list)
  + Additional Struct
* Feature:
  + Switch windows.
  + Design, align buttons, textboxes positions.
  + Application structure.
  + Write documents, README.md file, report, and organize files.
  + Planning, scheduling meetings, manage and assign tasks.
  + Design UI template (using Figma)
  + Design Data Storage
  + Beta tester
* UI:
  + Semester: staffSemesterLobby, staffSemesterLobby2, staffmanagesemesterdisplay
  + Course: staffmanagecourse, Staffaddcourse, staffviewstudentofcourse, staffCourseScoreboard
  + exportFile
  + staffChooseOption
  + studenthome
  + studentAboutUs

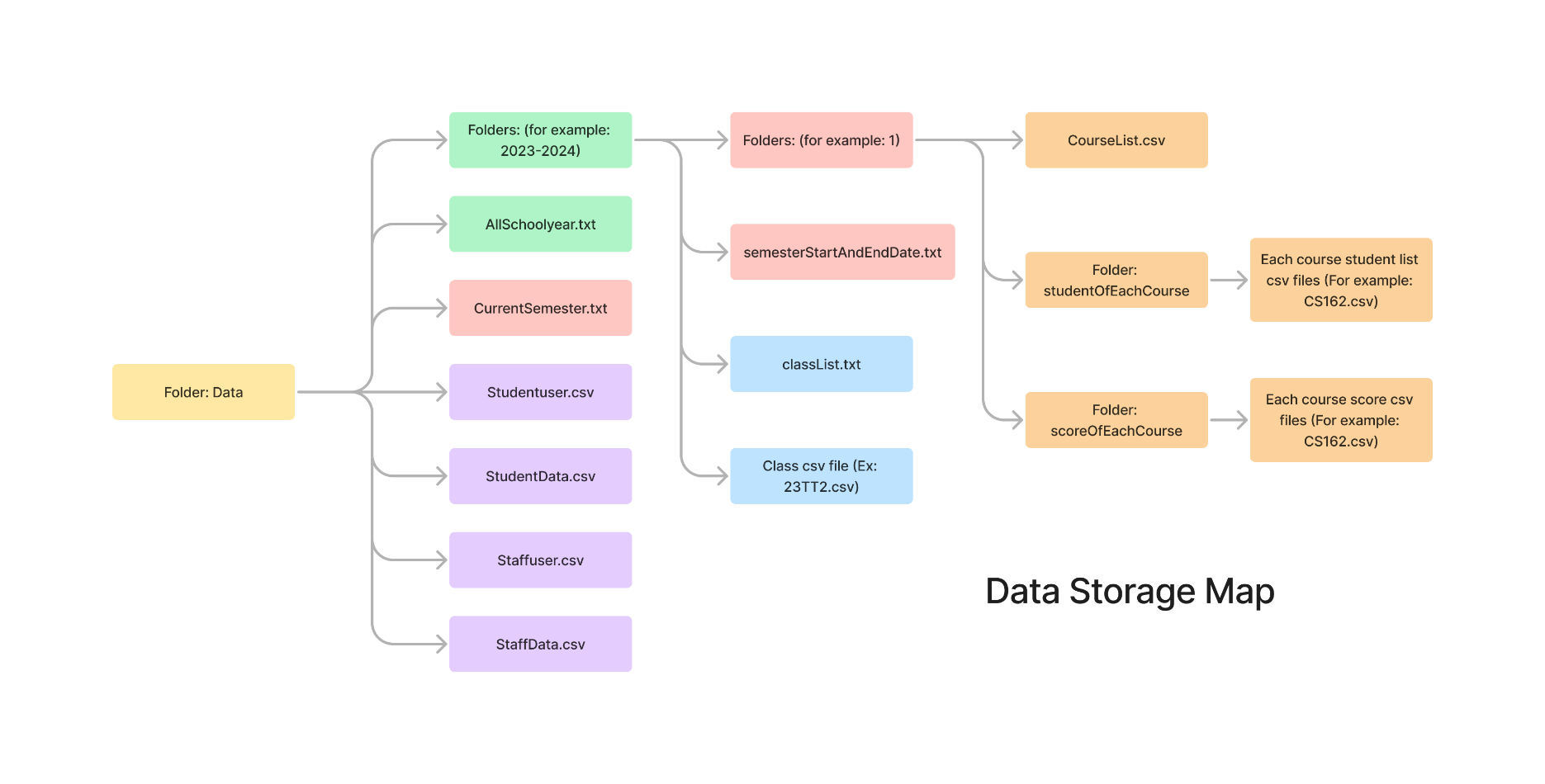
### Tran Trong Nguyen – 23125087 (**36%**)

* Class implements:
  + Dropdownlist
  + Button
  + Textbox, Output Textbox
  + Staff
  + Class
  + Student
  + Schoolyear (manage schoolyear list)
* Feature:
  + scroll wheel implementation.
  + Edit/view buttons, save/delete buttons.
  + Calculate overall GPA.
* UI:
  + Login: chooseRole, loginWindow
  + Announcement
  + About Staff: staffHome, staffviewprofile, staffaboutUs, staffChooseOption2
  + Class: staffaddclasses, staffmanageclass, staffaddfirstyearstudent, staffviewstudentinclass, staffViewStudentScoreboard, staffViewStudentScoreboard2
  + filenametoimport
  + Schoolyear: staffmanageschoolyeardisplay, staffmanageschoolyear2display
  + Student choose option: studentChooseOption, studentChooseOption2

### Doan Duc Tuan – 23125021 (**30%**)

* Class implements:
  + Linked List
  + User
  + Student
* Feature:
  + Hashing password
  + Innit classes
  + Calculate student GPA
  + Read Csv file functions
  + User manual videos
* UI:
  + staffChangePassword
  + studentChangePassword
  + studentCourse
  + studentScoreboard
  + studentprofileview

# Data Storage

Discuss how data is stored in the project. Explain the rationale behind the chosen data storage approach.

All data is stored in the “Data” folder. This will separate the data from the source code and other documentation.

## **Green:** School year management

* AllSchoolYear.txt: store all school years as a string (yyyy-yyyy)
* School year folders: folder name is the schoolyear string store in the AllSchoolYear.txt text file, contain all school year information, including its semesters, classes, courses, students.

## **Red:** Semester management

* CurrentSemester.txt: contain the latest semester and its year. This is the default semester.

School year folder contains (if at least one existed):

* semesterStartAndEndDate.txt: store all semesters and their start and end dates. The data is store in format: (semester dd mm yyyy dd mm yyyy\n)
* Semester folder: folder name can be either 1, 2, or 3 correspond to the semester information. Contains all semester information, including its courses and students.

## **Purple:** Account management

* userstaff.csv and studentuser.csv: store username and hashed password. Username is stored as a string (student username is his/her Student ID), and password is stored as a hashed string for security purposes.
* Studentdata.csv: store all student accounts’ information, in format: (No,studentID,lastName,firstName,dayOfBirth,gender,socialID)
* Studentdata.csv: store all staff accounts’ information, in format: (staffID,lastname,firstname,gender,socialID,dob)

## **Blue:** Class management

School year folder contains (if at least one class existed):

* classList.txt: store all classes as a string (Classname)
* class.csv: filename is the class string store in the classList.txt text file, contain all student information in a class, in format: (No,studentID,lastName,firstName,dayOfBirth,gender,socialID)

## **Orange:** Course management

Semester folder contains:

* CourseList.csv: store all courses and their information in the semester, in format: (No,courseID,courseNam,teacher\_lastName,teacher\_firstName,credits,max\_student,dayAndSession,classes)
* StudentOfEachCourse folder: store each course’s student list in a csv file, named CourseID.csv, in format: (No,studentID,lastName,firstName)
* scoreOfEachCourse: store each course’s student score list in a csv file, named CourseID.csv, in format: (No,studentID,lastName,firstName,midScore,finScore,otherScore,totalScore)

## Rationalize data storage decisions:

The data storage structure is quite simple and straightforward. Firstly, we store all data in a folder named Data. This is where we organize and modify our data, separate from other sources.

For the accounts’ information, we separate them into 2 main categories: staff and student. Each category has a file to store username and password, and the other store username and their account information. This is to speed up the login and view profile process and make it easier for passwords to be secured.

The general idea is with the school year and semester, we create a text file to control and manage a list of those objects, and each object has its own folders to store and access their information.

Since an academic school year contains all other objects, we start from this, and create a text file to store all existing school years, with their respective folder. The text file is to control and manage school years, and the folder is to access information. The semester work in similar fashion.

Since students are the smallest possible objects, we store students in csv file along with their information.

Originally, we aim to give staff 2 options of import: choose from file explorer/table or enter directory. Therefore, we propose StudentOfThisCourse folder idea. But since we are lack of man power, we cannot implement this functionality in time.

# Project Architecture

Folders:

* Doc Folder: To store our documents, including plans, report, implementation details, and Data Storage diagrams.
* Project: Store the Source code, Designs Screens, Fonts, and Data.
  + Data: to store and archive data.
  + SFML libraries: for GUI purposes.
  + Design UI: Screens that are used in the application.
  + Font: Store fonts needed for UI.

Main data structures:

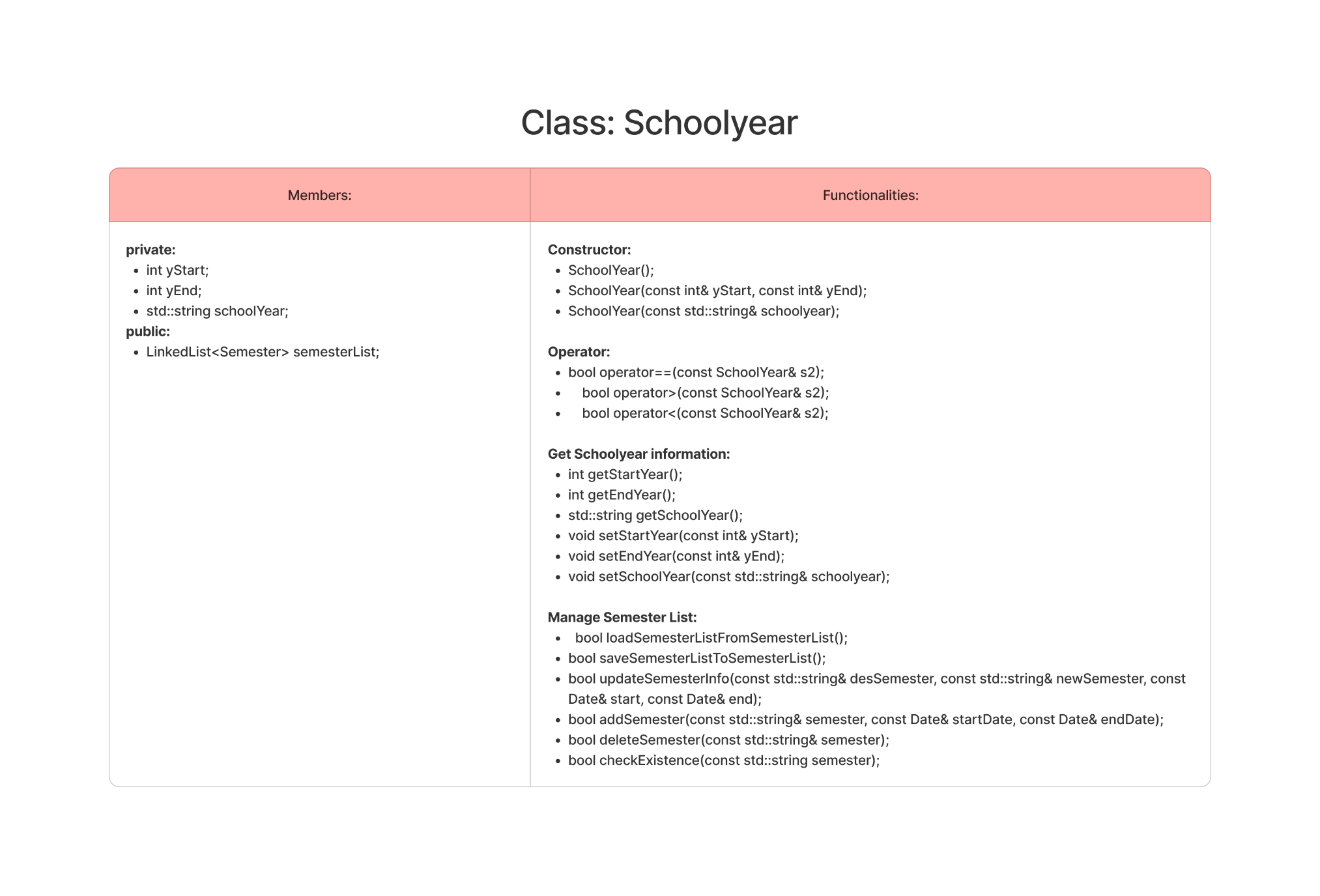
* Class User: managing login, logout functions, managing chooseRole function: either to be a staff or a student.
* Class Staff: allow users to enter staff view and manage contents.
* Class Student: allow users to enter student view to see information.
* Class Schoolyear: manage schoolyear and semesterList.
* Class Semester: manage semester and CourseList.
* Class Course: manage Course and StudentsOfCourse.
* Class Class: manage Class and StudentList.

The project aims to be encapsulated and is separated into each main categories, one that manage itself while controlling a list of smaller class member. Each class is stored in a .cpp file and have a .h header file.

The UI functions is divided and stored in files according to the class they manage.

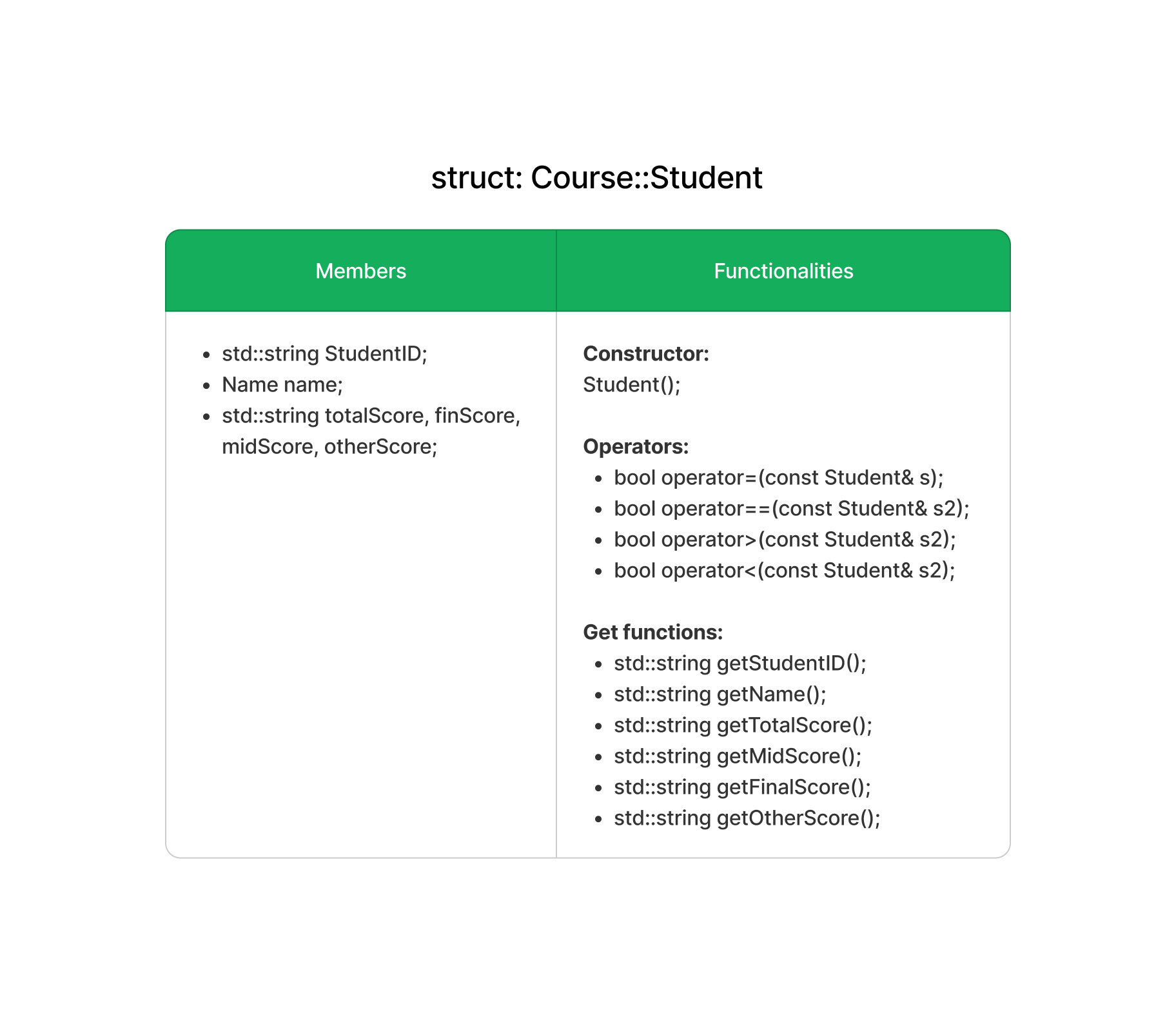
# Implementation Detail

## Backend:

**Schoolyear class**: have information of schoolyear, and list of semesters in this schoolyear, it use to manage schoolyear, and semester in schoolyear. It provides function to add, delete, load, and save semester for schoolyear.







**Class class:** have a information of class(class code) and list of Student in class.

It use to manage class: add 1 student, add student from csv file or delete student in class.

A screenshot of a class

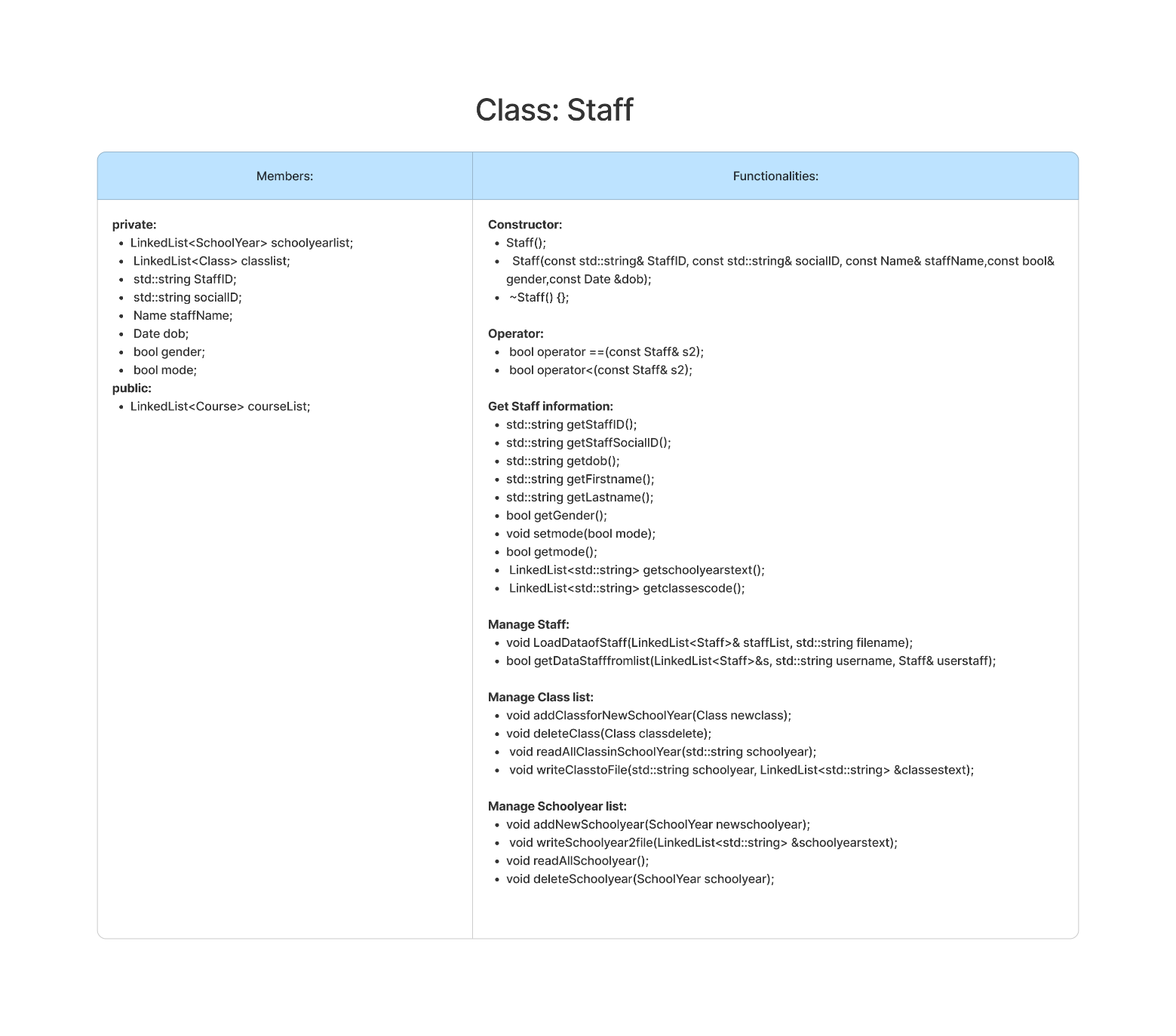
Description automatically generated

A screenshot of a phone

Description automatically generated

**Class Staff**: have information of staff, and a linked list of Schoolyear, Classes that need to manage.

It provides some function to add, delete, load and save, schoolyear and class, and get information if need to use.



A close-up of a student's information

Description automatically generated

A screenshot of a computer program

Description automatically generated

## FrontEnd:

* OutputTextbox: This class to draw a default text to display in window. It use to draw some text that it not require to edit, it only to use for display such as display schoolyear, semester in scoreboard.
* Button class: This class represents a clickable button with text. It has various methods for setting attributes such as position, color, text, etc., as well as methods for detecting clicks and mouse cursor hovering over the button. Overall, this class provides a convenient way to create and manage buttons in SFML applications. It used to in it button such as logout, save, delete button, …
* Textbox class: This class provides a flexible and customizable text input box component for SFML applications, with support for selection state, character limits, and text encoding. It’s used for inputting username password or someplace need to input text to add.
* Dropdownlist class: this class provides a list of buttons to draw in window and some function to manage this list. It’s used to draw a menu with a list of buttons. Or display classes schoolyears in system that require a lot of buttons to draw.

## UI functions:

**Login:**

* void chooseRole(sf::RenderWindow& window);
* void loginWindow(sf::RenderWindow& window, bool role);
* void announcement(std::string announcement);

### Staff:

**About Staff:**

* void staffHome(Staff& userstaff);
* void staffviewprofile(sf::RenderWindow& window, Staff& userstaff);
* void staffChangePassword(sf::RenderWindow& window, Staff& userstaff);
* void staffaboutUs(Staff& userstaff);

**Manage school year:**

* void staffmanageschoolyeardisplay(Staff& userstaff);
* void staffmanageschoolyear2display(sf::RenderWindow& window,Staff& userstaff, int Korderofbut);

**Manage semester:**

* void staffSemesterLobby(Staff& userstaff);
* void staffSemesterLobby2(sf::RenderWindow& window, Staff& userstaff, int Korderofbut);
* void staffmanagesemesterdisplay(sf::RenderWindow& window,Staff &userstaff, std::string schoolyear);

**Manage course:**

* void staffmanagecourse(Staff &userstaff, std::string schoolyear, std::string semester);
* void staffaddcourse(sf::RenderWindow& window, Staff& userstaff, std::string schoolyear, std::string semester);
* void staffviewstudentofcourse(sf::RenderWindow& window, Staff& userstaff, std::string schoolyear, std::string semester, std::string coursechosen);
* void staffCourseScoreboard(sf::RenderWindow& window, Staff& userstaff, std::string schoolyear, std::string semester, std::string coursechosen);

**Manage class:**

* void staffaddclasses(Staff& userstaff, std::string schoolyear)
* void staffmanageclass(sf::RenderWindow& window, Staff& userstaff, std::string schoolyear, int Korderbut);
* void staffaddfirstyearstudent(sf::RenderWindow& window, Staff& userstaff,std::string schoolyear, std::string addtoclass);
* void staffviewstudentinclass(sf::RenderWindow& window, Staff& userstaff, std::string schoolyearchoose, std::string classchosen);
* void staffChooseOption(sf::RenderWindow& window, Staff& userstaff, std::string classchosen);
* void staffChooseOption2(sf::RenderWindow& window, Staff& userstaff, std::string classchosen, int Korderofbut);
* void staffViewStudentScoreboard(sf::RenderWindow& window, Staff& userstaff, std::string classchosen, std::string schoolyear, std::string semeseter);
* void staffViewStudentScoreboard2(sf::RenderWindow& window, Staff& userstaff, std::string classchosen, std::string schoolyear, std::string semeseter, int kstudentchosen);s
* std::string filenametoimport();
* std::string exportFile();

### Student:

* void studenthome( Student& studentuser);
* void studentprofileview( Student& studentuser);
* void studentAboutUs( Student& studentuser);
* void studentChangePassword( Student& studentuser);
* void studentCourse(sf::RenderWindow& window, Student& studentuser, std::string schoolyear, std::string semester);
* void studentScoreboard(sf::RenderWindow& window, Student& studentuser, std::string schoolyear, std::string semester);
* void studentChooseOption(Student& studentuser, int viewType);
* void studentChooseOption2(sf::RenderWindow& window, Student& studentuser, int viewType,int korderbut);

# Technical Problems and Solutions:

For such an inexperienced team, we encountered some big technical challenges that we must resolve together, include:

## About user interface

* At first, we have a lot of ideas on how to implement the application. One of which was to design and develop graphic user interface. Since we are all inexperienced in this aspect, we agreed to display the application on console (which is acceptable for this project). But on second thought, we decided that this is the chance to study more about GUI since this project aimed to broaden our skills and introduce us to coding collaboration.
* To resolve this challenge, we do research and share academic documents with the help of Tran Trong Nguyen prior experience with SFML. Since we can’t organize offline meetings, we use Google Meeting to schedule sessions to assist studying GUI, while discussing how the design should be. We also want to try something new and agree to implement a navigation-and-tool bar on the left of the screen for better access.
* At first, when we began to study SFML, we mostly implemented back-end functionalities, classes, and structure our data storage. When we started to understand how SFML works, we then implemented our front-end functionalities.

## About major errors (reading access violations and other errors)

* In the beginning, we only coded back-end functionalities, divided into different classes for each member. Since one of our members also have to participate in another Math competition, the other two members have to share that workload and results in many conflicts.
* Thus, there are many redundancies and runtime errors undetected since we also didn’t test them properly in all cases. Only when we try merging them together that we can identify that there are errors everywhere.
* To resolve this alarming problem, we have to call an offline meeting to test, debug and find the errors together. After locating the root of the problem, we can then allow the original writer to fix the bug. We make a progress board to update our current progress and try to not overwrite other’s progress.
* After that, to avoid such problems, we occasionally organize offline meetings to show our current progress, assignment more works (since the workload now is uneven and a member hadn’t caught up with the progress), and to test for abnormalities in code. (In average once a week)

# Feature Description:

## Login screen:

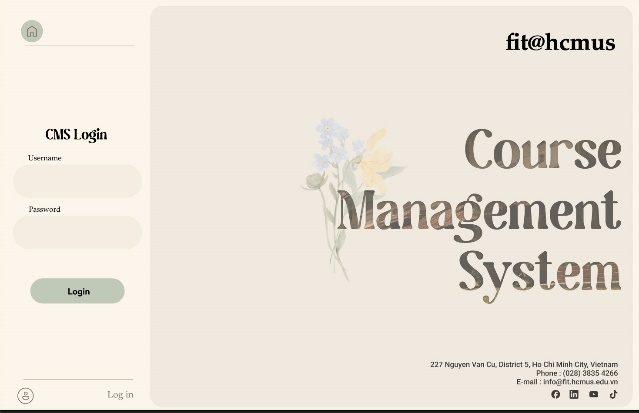


Figure 2. Login screen 2

Figure 1. Login screen 1

Step 1: After opening the application, we will be greeted with the login screen 1. We can choose to login as a student or a staff by pressing the green button. That will bring us to login screen 2.

Step 2: **Enter your username and password** into the textboxes on the left of the screen. If you wish to change your role as student or staff, you can **press the home button** (green, to the upper left of the screen) to come back to login screen 1.

After entering the right username and password in their respective textboxes, you can **press the green login button** to login to the app. Else, you will receive a message saying that you have entered wrong password or wrong username.

## **Staff view:** Navigation bar and tool bar:

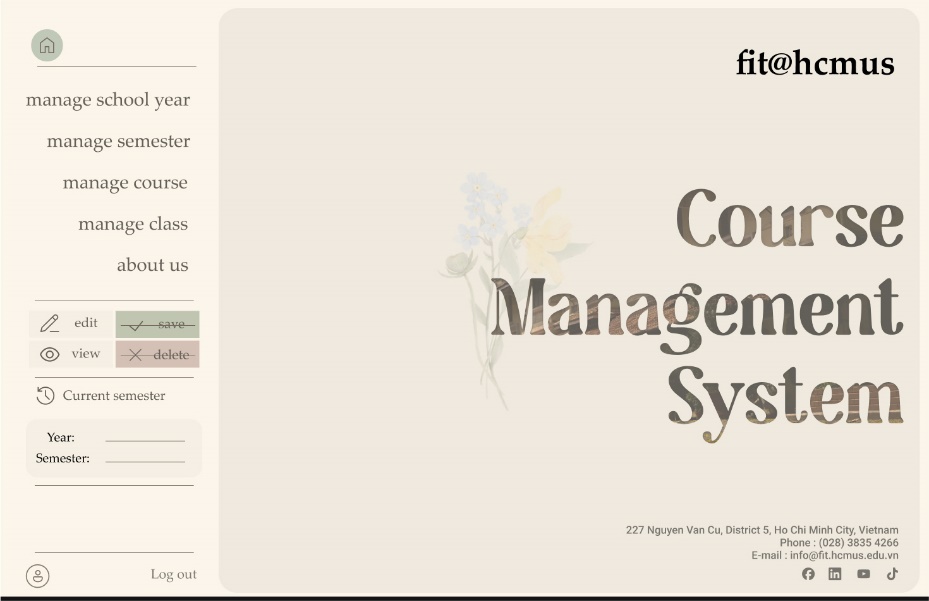
 The navigation bar is always on the left side of the screen.

Figure 3. Staff Home Screen

* Home button: return to Staff Home Screen (figure 3: left) anywhere in the app.
* Button: Manage schoolyear, manage semester, manage course, manage class, about us: will switch to their respective screen below. Their screen name match the object they manage.
* Edit/View button: changing mode between edit and view. We can only be in one mode. This mode will remain through all window unless being switch. Will be explained further below.
* Save/Delete button: can be use only in some screen where objects’ information can be edited. Use to save the edited information or delete objects.
* Current semester box: show the default/current semester that we are in. If the school year and semester were not chosen, all action performed in manage course and manage class that will be from this semester.
* Profile button: on the lower left of the screen. Switch to view profile screen.
* Log out button: on the right of the profile button. This will return you to the login screen 1 (Figure 1)

### School year management:

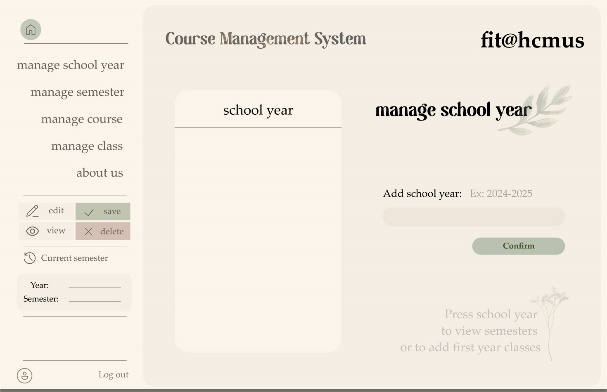
* **[Figure 4]** After pressing “manage school year” you will be switched to manage school year screen (Figure 4) and **view school year list**. Scroll down to see other schoolyears (if existed)
* **[Figure 4]** Here, you can **create a school year** by typing into the text box on the right and press the green button “Confirm” below.

Figure 4. manage school year screen.

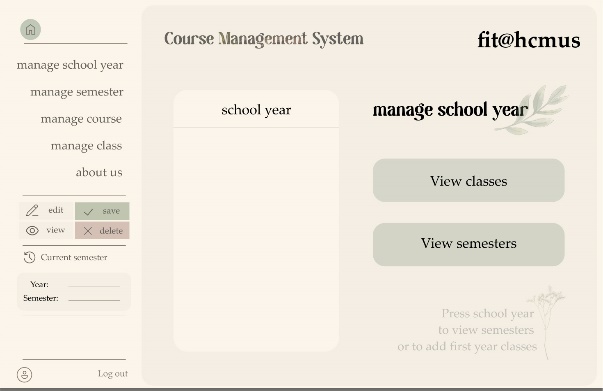
* **[Figure 4]** Or you can **delete a school year** by clicking delete in edit mode and choose school year to delete.
* **[Figure 5]** You can **choose a school year** by pressing it. This will switch you to manage school year screen 2 (Figure 5) You can still perform any actions in Figure 4 in this screen, only not adding more schoolyear to it.
* **[Figure 5]** If you wish to **come back** to add more school year, you can press the chosen school year again. This will switch you back to Figure 4.

Figure 5. manage school year screen 2.

* **[Figure 5]** You can either **view classes** or **view semesters** in the chosen school year by pressing one of the two green buttons on the right of the screen. This will switch you to their respective window below. (Semester management or Class management)

### Semester management:

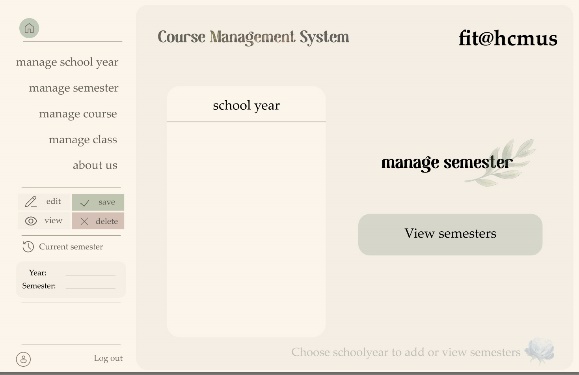
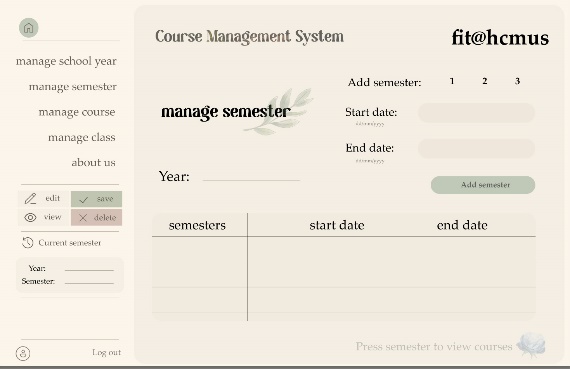
* If you choose manage semester from the navigation bar, you will be switched to the manage semester lobby (Figure 6), otherwise skip this paragraph.
  + You can choose school year like in Figure 4 and 5. Noted that you cannot edit school year in this screen.
  + Press the green view semesters button on the right to **view the chosen school year’s semesters.**

Figure 6. manage semester lobby.

* If you choose to view semesters from Figure 5 or 6, you will be switched to “manage semester” screen (Figure 7) to **view semesters list** in the year that is chosen.
* You can **add another semester** in the top right corner. Choose either semester 1, 2, or 3 by pressing it, add start date and end date in the text box below and press the green button “Add semester” to add. Note that if such semester exists, you cannot add another one. (Ex: You can’t add two semester 1)

Figure 7. manage semester.

* In edit mode, you can **delete a semester** by choosing it and press delete button on navigation bar, and you can **modify semester’s information** by clicking into it and type new information.
* In view mode, if you select a semester, it will show you the course list of that semester, switch to another screen below. (Figure 8)

### Course management:

* **[Figure 8]** If you choose manage course from the navigation bar, you will be switched to the manage course screen (Figure 8) **to view course list in default/current semester.**

Figure 8. manage course.

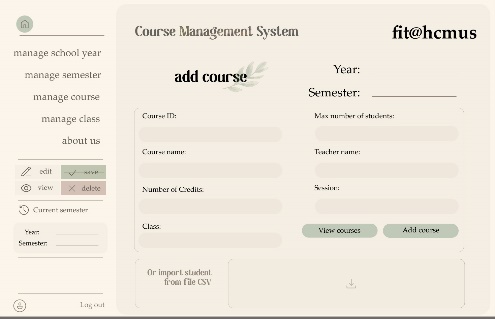
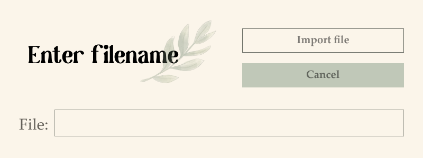
* **[Figure 8]** If you choose semester from Figure 7 to view course, you will be switched to the manage course screen (Figure 8) to **view the chosen semester’s course list.**
* **[Figure 8]** In edit mode, you can **edit course information** and **delete courses**.
* **[Figure 8]** You can **create a course** by choosing “Add course” green button at the bottom of the screen. You will be switched to add course screen (Figure 9)
* **[Figure 9]** Fill in all the blank textboxes to add a course. You can choose to **import course’s students** there by clicking the rectangle below or add later. If you choose to import here, you will open the Enter filename dialog.

Figure 9. add course.

* + **[Figure 9]** Enter filename in the textbox.
  + **[Figure 9]** Press import file to process forward and import student to course.
  + **[Figure 9]** Press cancel to close the dialog without importing.

Enter filename dialog.

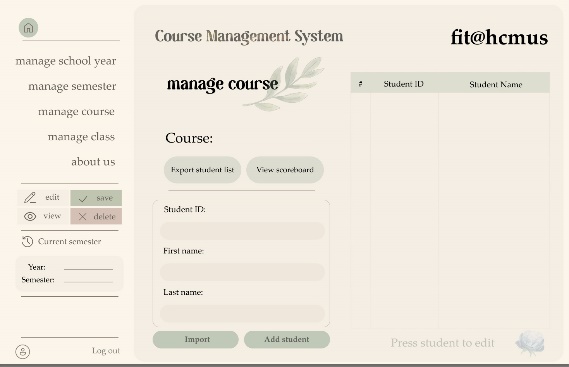
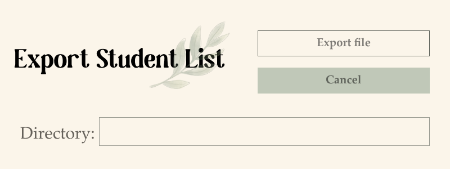
* **[Figure 9]** Press add course green button on the right to add the newly created course. If not, you will not be able to create it.
* **[Figure 9]** Press **view courses** to comeback to Figure 8.
* **[Figure 10]** To view a course’s student. First, select the course in view mode. This will switch you to manage course student screen (Figure 10) to **view student list in course**.

Figure 10. manage course student.

* **[Figure 10]** In edit mode, you can **modify students’ information** or **delete students** from course.
* **[Figure 10]** You can **enroll students** by typing student’s information to the three textboxes in the middle section and press add student green button below. You can also choose to **import student files** by pressing the green button import next to the add student green button.
* **[Figure 10]** You can **export student list** by pressing the green button Export student list in the upper part below course name. This will open export dialog (right)

Export dialog.

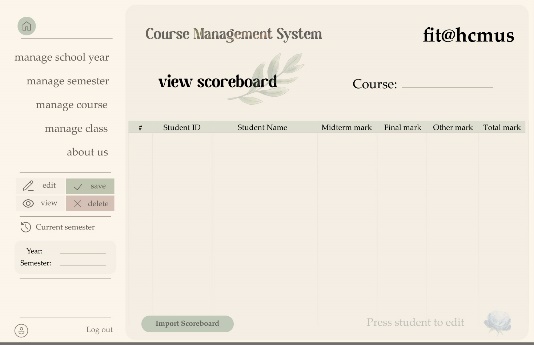
* + Enter the directory you wish to export.
  + Press import file to process forward and export student list.

Figure 11. manage course scoreboard.

* + Press cancel to close the dialog without exporting.
* **[Figure 10]** To **view a course’s scoreboard**, press view scoreboard green button next to the export button. You will be switched to Figure 11.
* **[Figure 11]** To **import a course’s scoreboard**, press import scoreboard green button at the bottom of the screen. You will open the Enter filename dialog. (See Course management – Figure 9)
* **[Figure 11]** In edit mode, you can **modify the students’ score** and information, or **delete score** from course.

### Class management:

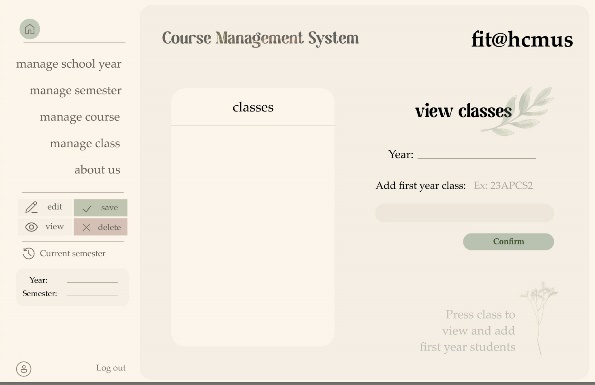
* **[Figure 12]** If you choose manage class from the navigation bar, you will be switched to the manage class screen (Figure 12) **to view class list in default/current year.**

Figure 12. manage class screen.

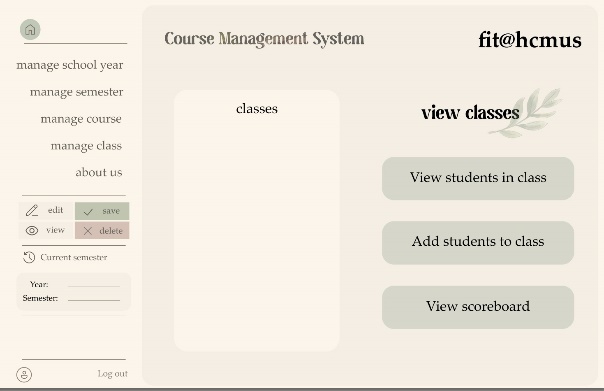
* **[Figure 12]** If you choose semester from Figure 5 to view class, you will be switched to the manage class screen (Figure 8) to **view the chosen year’s class list.**
* **[Figure 12]** Scroll down to see other classes (if existed)
* **[Figure 12]** Here, you can **create a class** by typing into the text box on the right and pressing the green button “Confirm” below.
* **[Figure 12]** Or you can **delete a class** by clicking delete in edit mode and choose school year to delete.
* **[Figure 12]** You can **choose a class** by pressing it. This will switch you to manage class screen 2 (Figure 13) You can still perform any actions in Figure 12 in this screen, only not adding more class to it.
* **[Figure 13]** If you wish to **come back** to add more class, you can press the chosen class again. This will switch you back to Figure 12.

Figure 13. manage class screen 2.

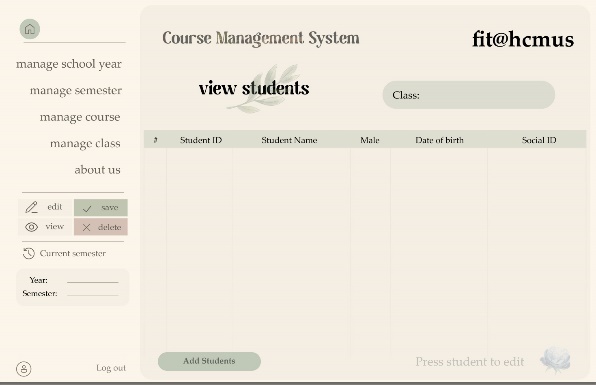
* **[Figure 13]** You can either **view students in class** or **adding students to class** or **view scoreboard** in the chosen class by pressing one of the three green buttons on the right of the screen. This will switch you to their respective window below.

Figure 14. view class’s student.

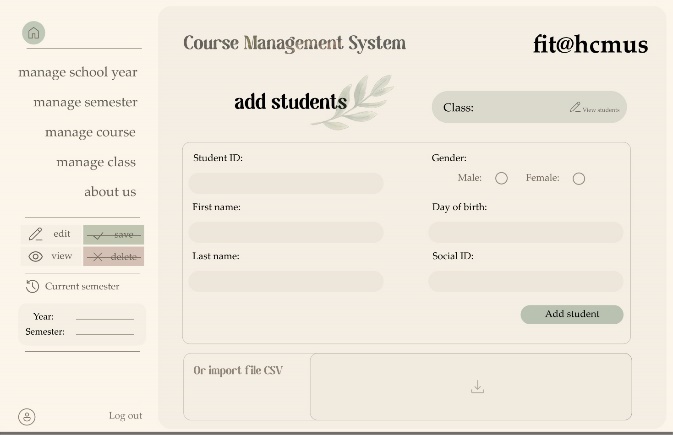
* **[Figure 13]** If you choose view student in class From Figure 13, you will be switched to Figure 14 **to view student list in class**
* **[Figure 14]** In edit mode, we can **modify the information** of each student or **delete student** from the class
* **[Figure 14]** We can add new student by pressing the green button “Add student” below to switch to Figure 15 to add new student.

Figure 15. add class’s student

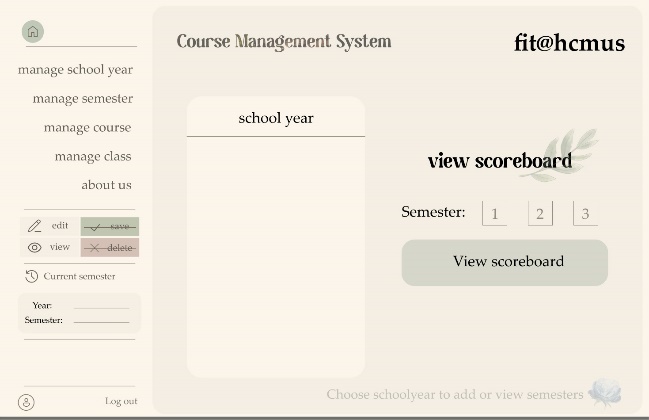
* **[Figure 13]** If you choose to **add student** from Figure 13 or Figure 14, you will be switched to add class’s student screen (Figure 15)
  + **[Figure 15]** Fill in all the blank textboxes to add a student. You can choose to **import class’s students** there by clicking the rectangle below. If you choose to import here, you will open the Enter filename dialog. (See Course management – Figure 9)
  + **[Figure 15]** Press add student green button on the right to add the new student. If not, you will not be able to add him/her.
  + **[Figure 15]** Press the pen right beside the class name to comeback to Figure 14 (view class’s student)
* **[Figure 13]** If you choose view scoreboard in class From Figure 13, you will be switched to Figure 16.

Figure 16. scoreboard option

* **[Figure 16]** Chose a school year to view scoreboard, then choose a semester by pressing to the chosen year and semester. Press the green button choose semester on the right. You will be switch to Figure 17.
* A screenshot of a computer

  Description automatically generated**[Figure 17]** Choose a student on the upper table, you will be displayed the scoreboard of that student on the lower table and his/her semester and overall GPA. In edit mode, you can modify the score of the student.

.

Figure 17. view class’s scoreboard.

## **Staff view:** Navigation bar and tool bar:

The navigation bar is always on the left side of the screen.

Figure 18. Student Home Screen

* Home button: return to Student Home Screen (figure 3: left) anywhere in the app.
* Button: my profile, change password, my course, my scoreboard, about ú: will switch to their respective screen below. Their screen names match the object they show.
* Current semester box: show the default/current semester that we are in.
* Profile button: on the lower left of the screen. Switch to view profile screen.
* Log out button: on the right of the profile button. This will return you to the login screen 1 (Figure 1)

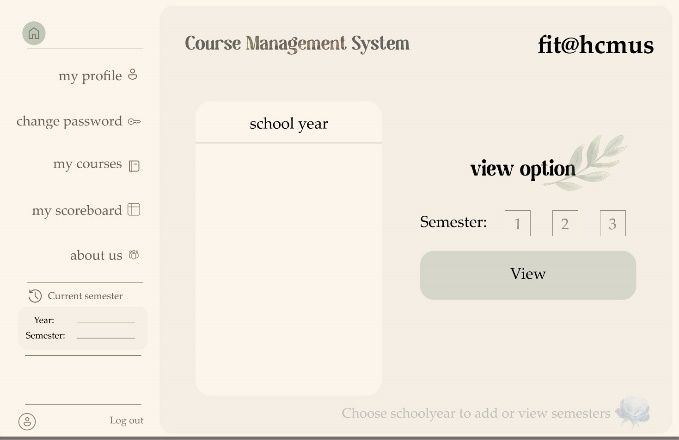
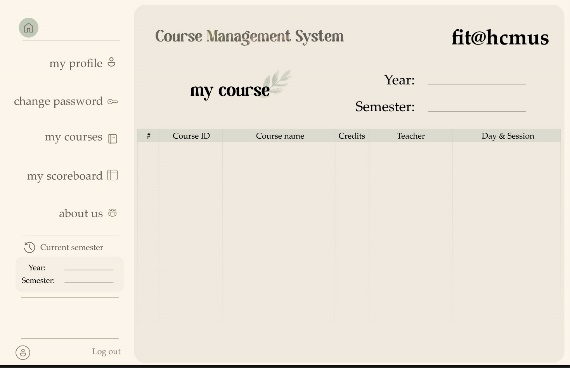
Choose option:

Figure 19. Choose option.

If we choose my scoreboard or my course, you will switch to choose option screen. You will choose the school year by clicking on the year presented in the list on the left and then choose available semester on the right. Press the green button “View” to view the course or scoreboard that you want to view.

We will be switched to either two screens below:



## Profile and change password

A screenshot of a computer

Description automatically generatedA screenshot of a login form

Description automatically generated

Figure 20. my profile.

Figure 21. Change password.

If we choose to view profile, we will be switched to Figure 20. There is also a change password button with a pen symbol on the right above gender textbox. If we choose change password, we will be switched to Figure 21. To change password, enter the correct current password on the top textbox, then the new password into the middle textbox. Finally, confirm the password once again by typing the new password to the confirm password textbox. If the current password is correct, and the confirm password match the new password, press change password and your password will be changed. Else, a message will appear to inform you about the error.

# Conclusion

The Course Management System was an exciting project to work on, and we gained a lot from it, above and beyond its original scope. We were able to study and research many different topics outside of our class, some of which were almost completely foreign to us.

Although the coding conventions, optimization and cleanliness of the code aren’t perfect, the application is working as we desired. The application organizes data structurally and is simple yet effective, and all the core and additional functionalities work successfully. We believe there is still a lot of potential enhancements for this application in the future, including but not limited to the UI’s switching window features and import file features (choose from file explorer).

# References

1. Chat GPT
2. SFML Tutorials: [2.6 Tutorials (SFML / Learn) (sfml-dev.org)](https://www.sfml-dev.org/tutorials/2.6/)
3. How to write Markdown file:
   1. [Include an SVG (hosted on GitHub) in MarkDown - Stack Overflow](https://stackoverflow.com/questions/13808020/include-an-svg-hosted-on-github-in-markdown/16462143#16462143)
   2. [Markdown Cheat Sheet – How to Write in Markdown with Examples (freecodecamp.org)](https://www.freecodecamp.org/news/markdown-cheat-sheet/)