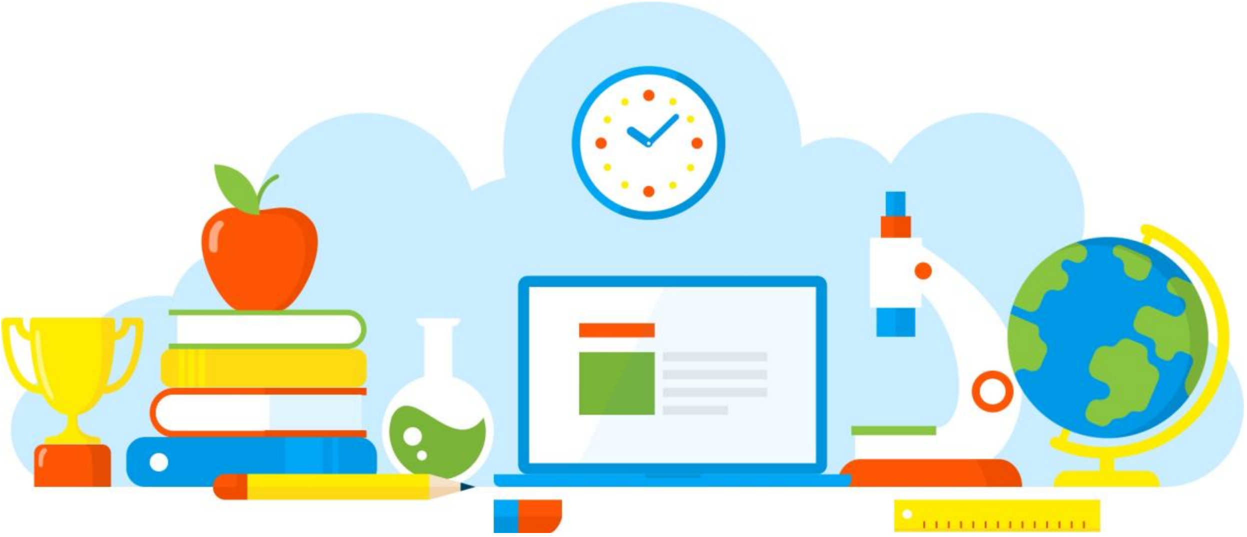


Class Scheduling System

Group#: 34

Github Repository URL:

https://github.com/TommyLi0511/OOP-Final-Project.git

Tommy Li

12/06/2023

System Analysis

Table of Work

	Tommy Li
Project Description	x
Uses Cases Diagram(s)	x
Sequence Diagrams	x
Class diagram(s)	x
Implementation	x
Conclusion	x

Project Description

- **General Description, Goals, Benefits:**

This project is implemented in Java language. It uses object-oriented programming design concepts to simulate a common class scheduling system. The users of this program, students, can complete their respective classes scheduling correctly and conveniently in the given database by using this program. This program automatically helps students to check if a class is full, if there are time conflicts, and if the credit requirements are met. At the same time, by saving and updating the data.bin file, data consistency and timeliness can be preserved in the database.

- **System input(s) and outputs(s):**

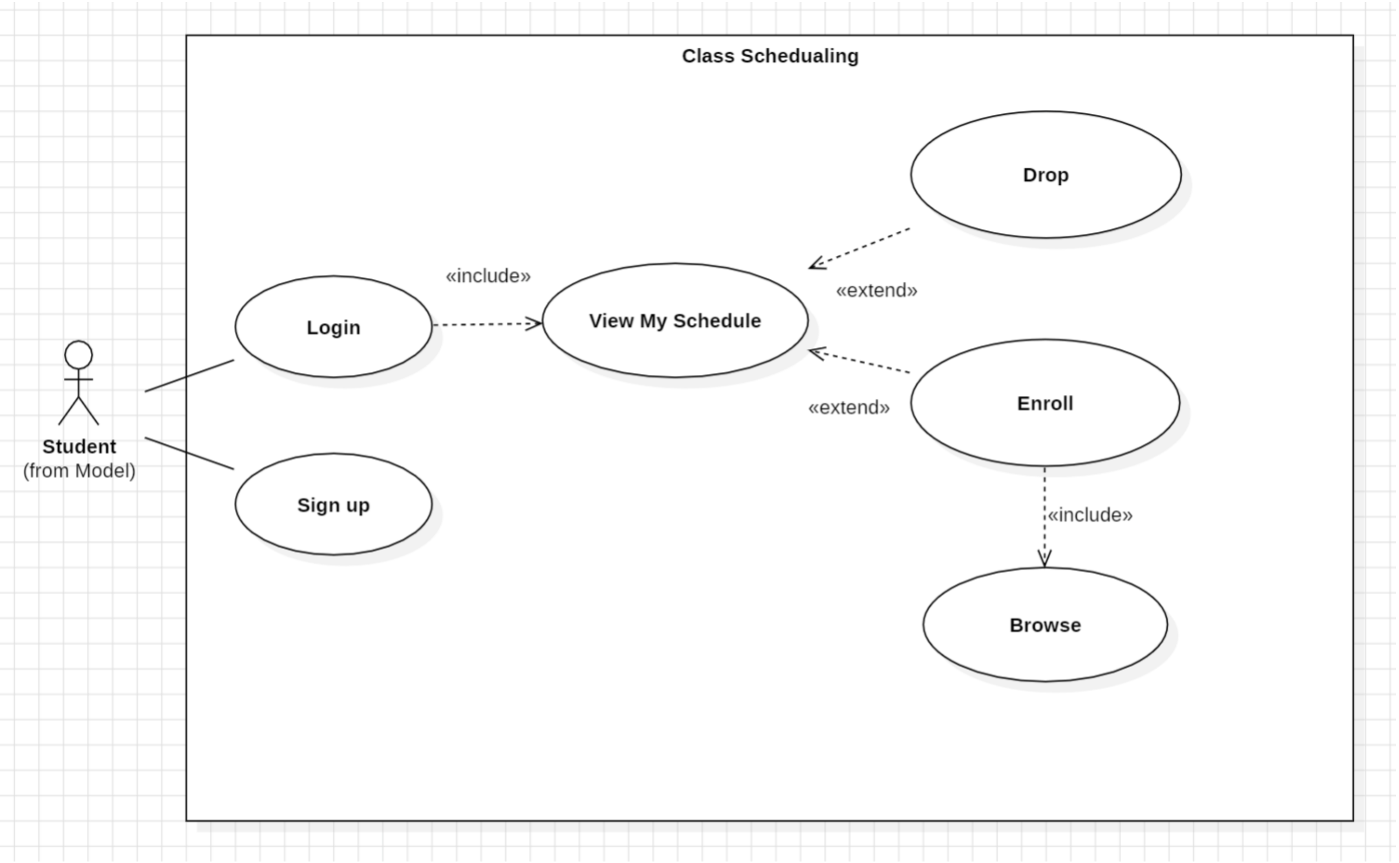
Inputs: User ID, Password, Course ID, information of courses(database).

Outputs: New users, new class schedules stored in the data.bin file.

- **Special requirements:**

This program doesn't allow managers, such as professors or advisors, to add or remove courses from the database by using some GUIs. This can only be achieved by directly modifying objects in code. At the same time, users have to have the updated version of data.bin file to complete their classes scheduling correctly. If they don't have it, the data consistency and timeliness can no longer be preserved in the database.

Use Cases Diagram

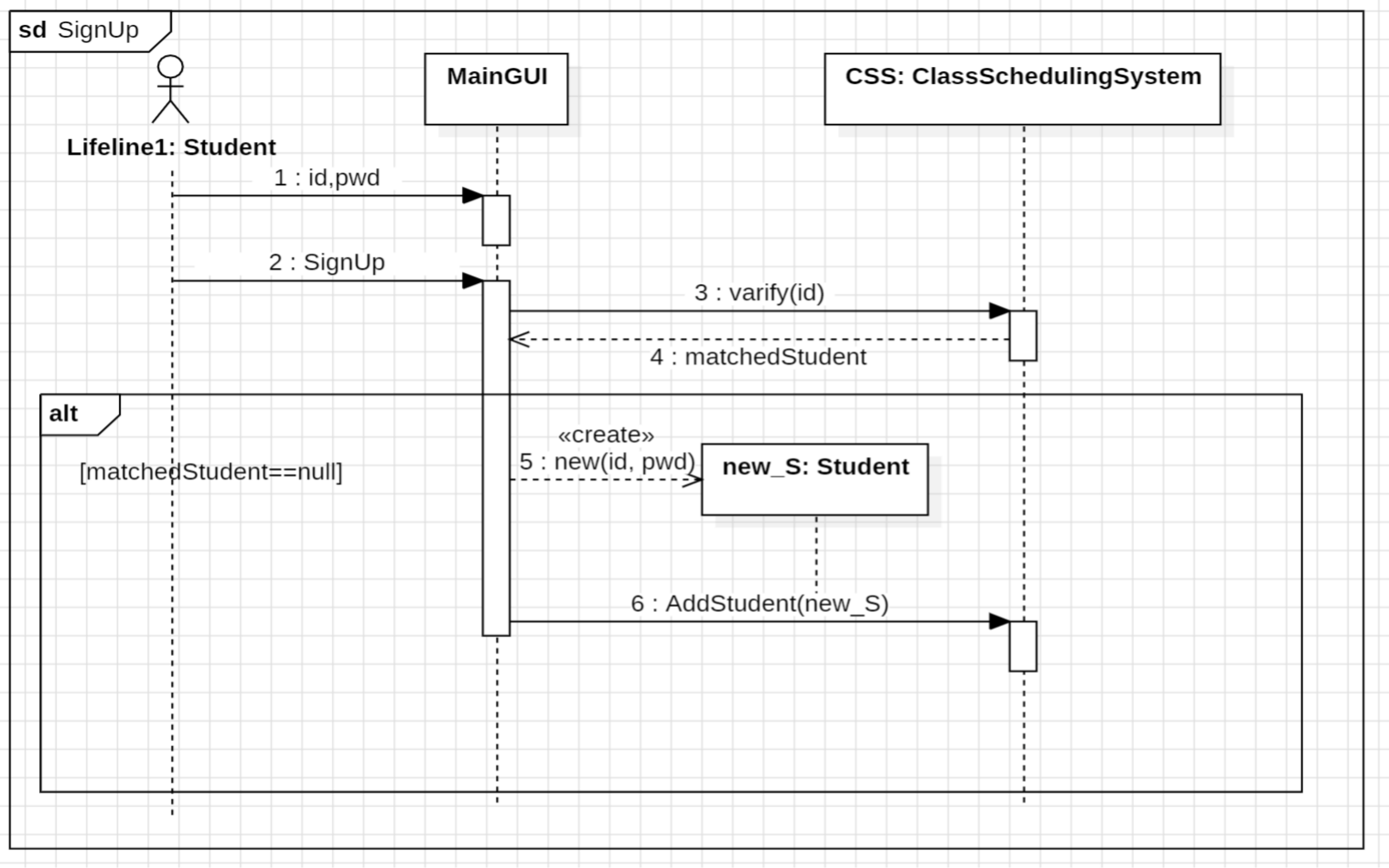


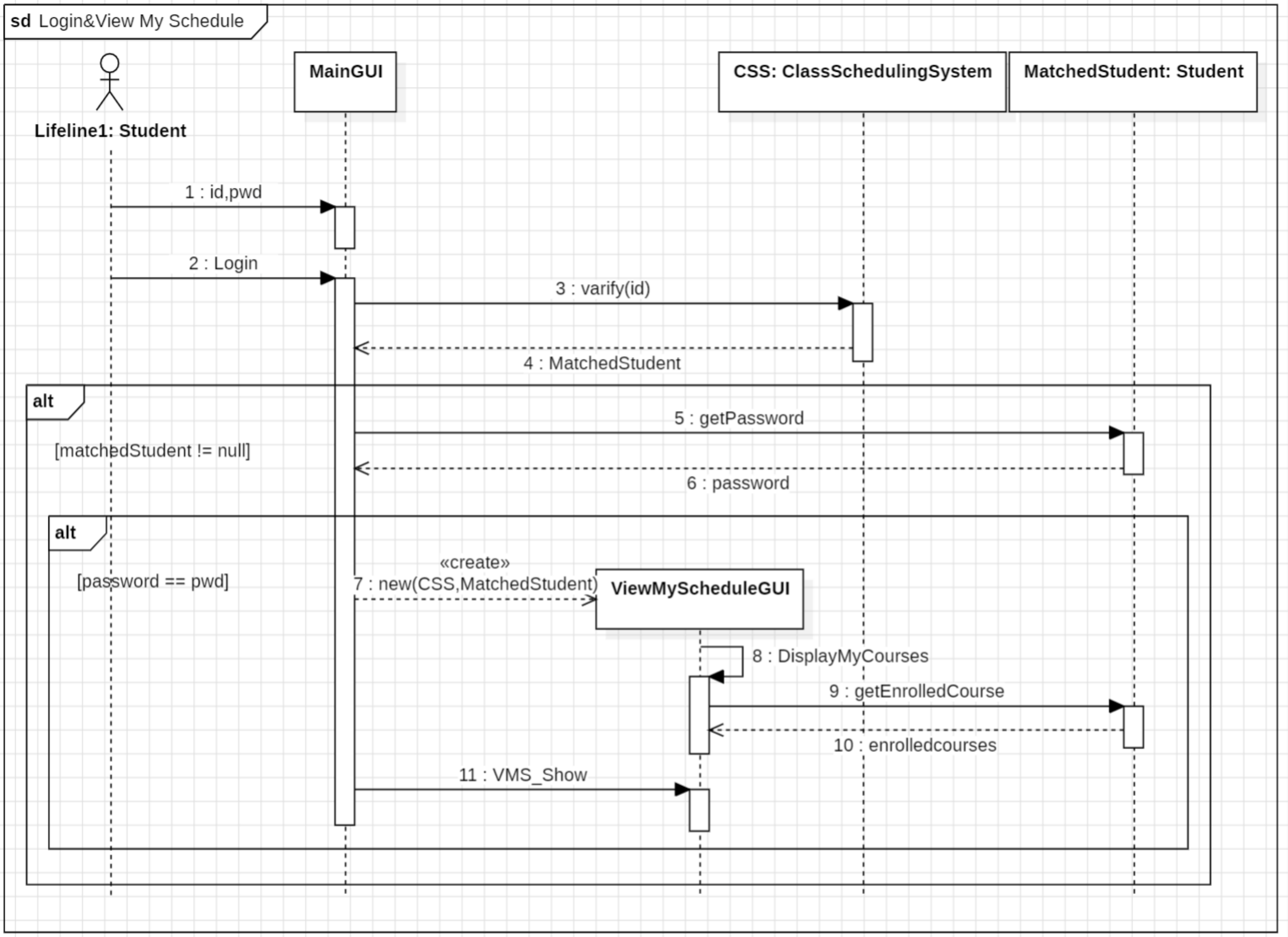
UC Reference Name/Number: Sign up (UC-1)	
Overview	Create a new student object without a complete class schedule using the entered input ID and Password
Related use cases:	N/A
Actors	Student
UC Reference Name/Number: Login (UC-2)	
Overview	Check if the entered ID and Password match any student's information in the database, if so, allow him/her to login in.
Related use cases:	View My Schedule

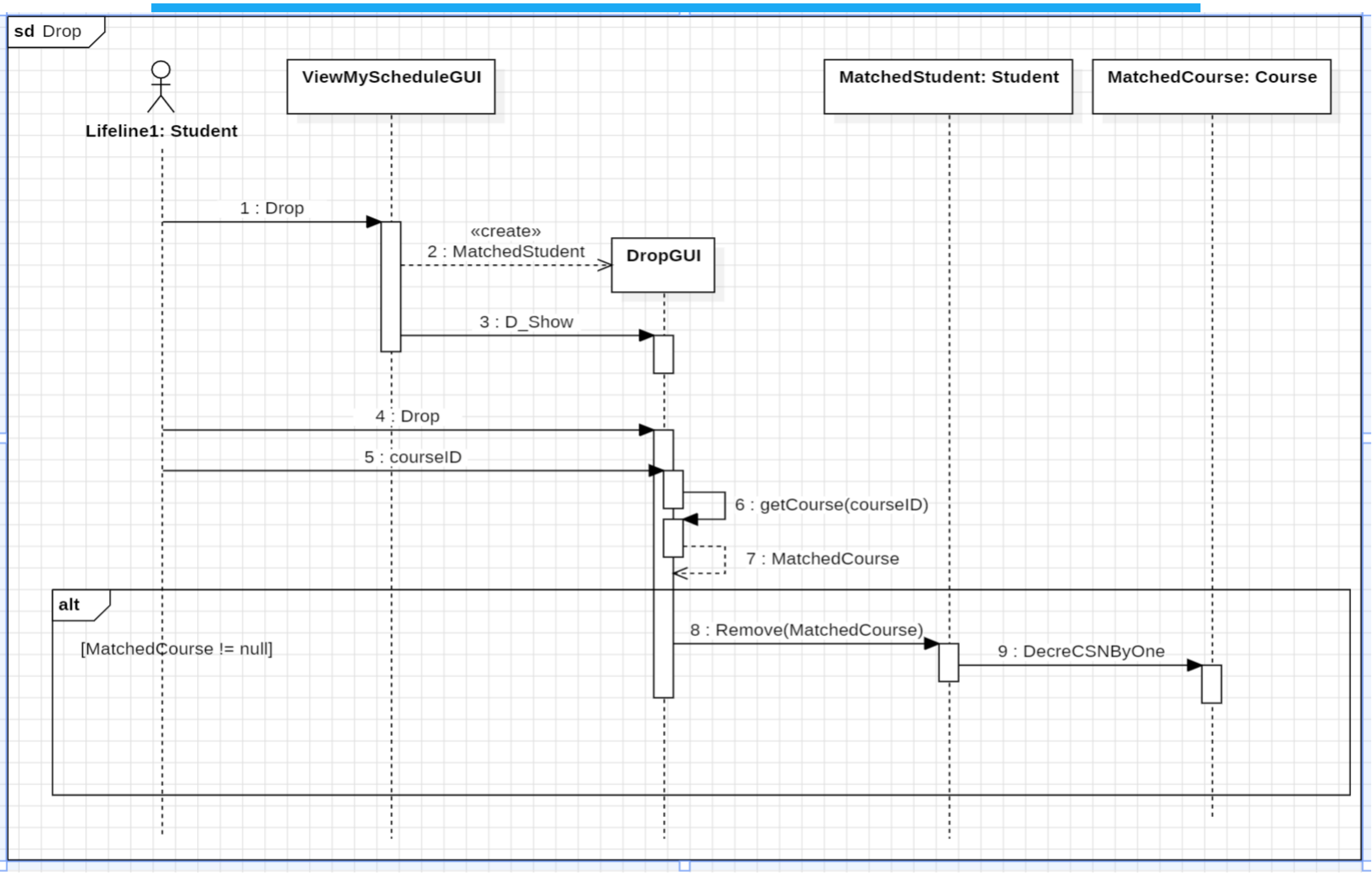
Actors	Student
UC Reference Name/Number: View My Schedule (UC-3)	
Overview	Display all courses the student who matches with the entered ID and Password has enrolled.
Related use cases:	Login, Drop, Enroll
Actors	Student
UC Reference Name/Number: Drop (UC-4)	
Overview	Drop a course that has been enrolled by the current user
Related use cases:	View My Schedule
Actors	Student
UC Reference Name/Number: Enroll (UC-5)	
Overview	Enroll a course that has not been enrolled by the current user and exists in the database
Related use cases:	View My Schedule, Browse
Actors	Student
UC Reference Name/Number: Browse (UC-6)	
Overview	Display all courses as well as their information that exist in the database. This provides the necessary information (course ID) the user needs to enroll in a course.
Related use cases:	Enroll
Actors	Student

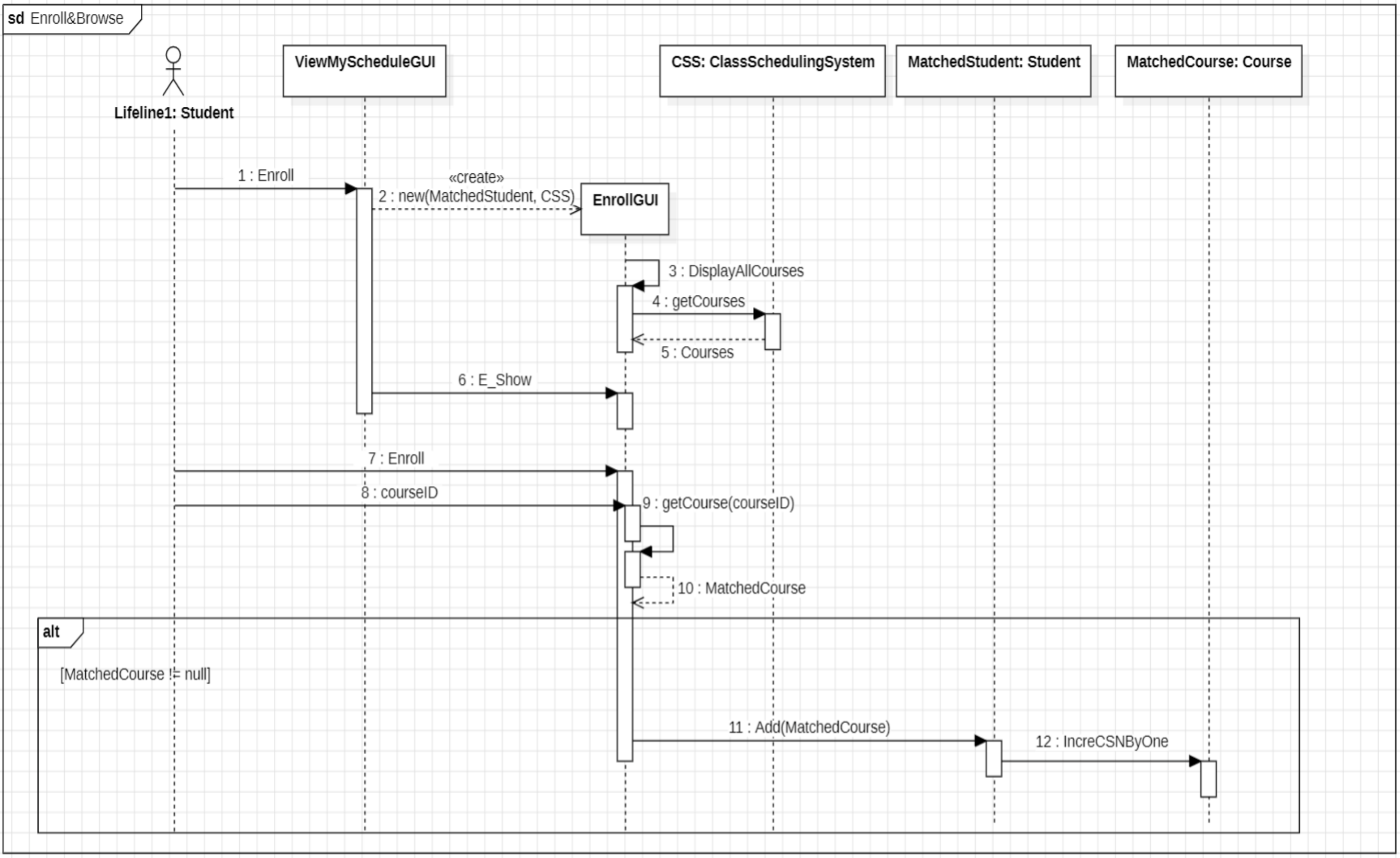
System Design

- Sequence Diagrams:

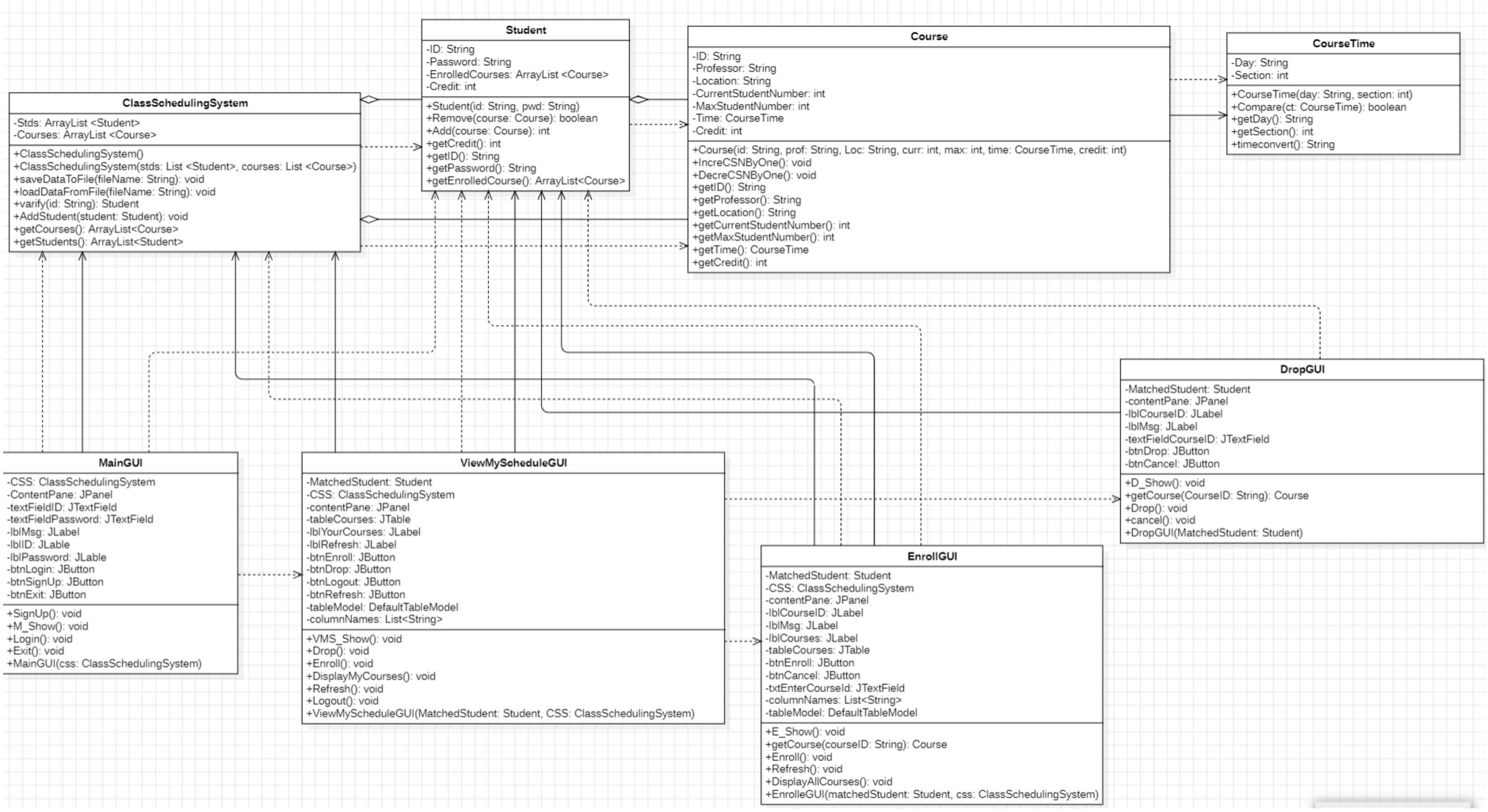



sd Login&View My Schedule
Lifeline1: Student
MainGUI
CSS: ClassSchedulingSystem
MatchedStudent: Student
1 : id,pwd
2 : Login
3 : varify(id)
4 : MatchedStudent
alt
[matchedStudent != null]
5 : getPassword
6 : password
alt
[password == pwd]
«create»
7 : new(CSS,MatchedStudent)
ViewMyScheduleGUI
8 : DisplayMyCourses
9 : getEnrolledCourse
10 : enrolledcourses
11 : VMS_Show


sd Drop
Lifeline1: Student
ViewMyScheduleGUI
MatchedStudent: Student
MatchedCourse: Course
1 : Drop
«create»
2 : MatchedStudent
DropGUI
3 : D_Show
4 : Drop
5 : courseID
6 : getCourse(courseID)
7 : MatchedCourse
alt
[MatchedCourse != null]
8 : Remove(MatchedCourse)
9 : DecreCSNByOne


sd Enroll&Browse
Lifeline1: Student
ViewMyScheduleGUI
CSS: ClassSchedulingSystem
MatchedStudent: Student
MatchedCourse: Course
1 : Enroll
«create»
2 : new(MatchedStudent, CSS)
EnrollGUI
3 : DisplayAllCourses
4 : getCourses
5 : Courses
6 : E_Show
7 : Enroll
8 : courseID
9 : getCourse(courseID)
10 : MatchedCourse
alt
[MatchedCourse != null]
11 : Add(MatchedCourse)
12 : IncreCSNByOne

- ## Class Diagram:


Student
-ID: String
-Password: String
-EnrolledCourses: ArrayList <Course>
-Credit: int
+Student(id: String, pwd: String)
+Remove(course: Course): boolean
+Add(course: Course): int
+getCredit(): int
+getID(): String
+getPassword(): String
+getEnrolledCourse(): ArrayList<Course>
Course
-ID: String
-Professor: String
-Location: String
-CurrentStudentNumber: int
-MaxStudentNumber: int
-Time: CourseTime
-Credit: int
+Course(id: String, prof: String, Loc: String, curr: int, max: int, time: CourseTime, credit: int)
+IncreCSNByOne(): void
+DecreCSNByOne(): void
+getID(): String
+getProfessor(): String
+getLocation(): String
+getCurrentStudentNumber(): int
+getMaxStudentNumber(): int
+getTime(): CourseTime
+getCredit(): int
CourseTime
-Day: String
-Section: int
+CourseTime(day: String, section: int)
+Compare(ct: CourseTime): boolean
+getDay(): String
+getSection(): int
+timeconvert(): String
ClassSchedulingSystem
-Stds: ArrayList <Student>
-Courses: ArrayList <Course>
+ClassSchedulingSystem()
+ClassSchedulingSystem(stds: List <Student>, courses: List <Course>)
+saveDataToFile(fileName: String): void
+loadDataFromFile(fileName: String): void
+varify(id: String): Student
+AddStudent(student: Student): void
+getCourses(): ArrayList<Course>
+getStudents(): ArrayList<Student>
DropGUI
-MatchedStudent: Student
-contentPane: JPanel
-lblCourseID: JLabel
-lblMsg: JLabel
-textFieldCourseID: JTextField
-btnDrop: JButton
-btnCancel: JButton
+D_Show(): void
+getCourse(CourseID: String): Course
+Drop(): void
+cancel(): void
+DropGUI(MatchedStudent: Student)
MainGUI
-CSS: ClassSchedulingSystem
-ContentPane: JPanel
-textFieldID: JTextField
-textFieldPassword: JTextField
-lblMsg: JLabel
-lblID: JLable
-lblPassword: JLable
-btnLogin: JButton
-btnSignUp: JButton
-btnExit: JButton
+SignUp(): void
+M_Show(): void
+Login(): void
+Exit(): void
+MainGUI(css: ClassSchedulingSystem)
ViewMyScheduleGUI
-MatchedStudent: Student
-CSS: ClassSchedulingSystem
-contentPane: JPanel
-tableCourses: JTable
-lblYourCourses: JLabel
-lblRefresh: JLabel
-btnEnroll: JButton
-btnDrop: JButton
-btnLogout: JButton
-btnRefresh: JButton
-tableModel: DefaultTableModel
-columnNames: List<String>
+VMS_Show(): void
+Drop(): void
+Enroll(): void
+DisplayMyCourses(): void
+Refresh(): void
+Logout(): void
+ViewMyScheduleGUI(MatchedStudent: Student, CSS: ClassSchedulingSystem)
EnrollGUI
-MatchedStudent: Student
-CSS: ClassSchedulingSystem
-contentPane: JPanel
-lblCourseID: JLabel
-lblMsg: JLabel
-lblCourses: JLabel
-tableCourses: JTable
-btnEnroll: JButton
-btnCancel: JButton
-txtEnterCourseId: JTextField
-columnNames: List<String>
-tableModel: DefaultTableModel
+E_Show(): void
+getCourse(courseID: String): Course
+Enroll(): void
+Refresh(): void
+DisplayAllCourses(): void
+EnrolleGUI(matchedStudent: Student, css: ClassSchedulingSystem)

Conclusion

This project is an application of object-oriented programming, and it also demonstrates the advantages of GUI. This program has achieved the functions I expected. On the other hand, He still has a lot of room for improvement: Advisor can be added to manage all courses in the system; Wait list mechanism and I can also take prerequisites into account. Most of course scheduling systems are online to facilitate data interaction (data.bin is unnecessary). Maybe after I learn more about networks and servers, I can modify my project into a website to better realize its functions and provide convenience to users. It makes me happy to be able to apply what I have learned into practice and create a meaningful project.