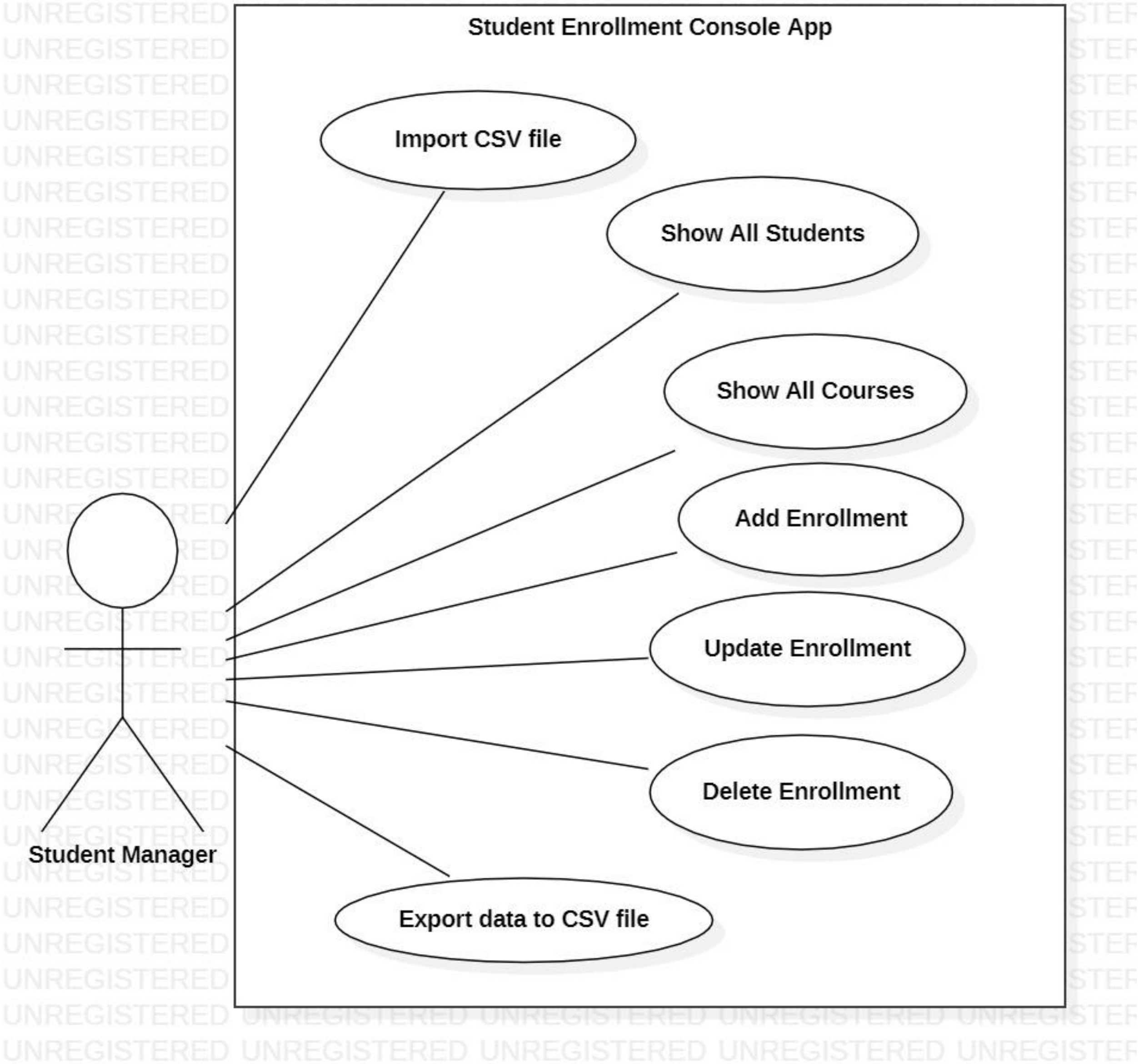
I. Function summary

This console app was created to help manage the student enrollment. To be more precise, this app is built for the managers who already know the detailed information of all courses in the semester as well as the background status of all the students. With that they can use this console app to easily import a CSV file and auto enroll students to the courses, then they can update or delete the enrollment as they want. After finishing the enrolment process, there are three options for the managers to export the data to a CSV file: Print all courses for a specific student in a specific semester, Print all Students for a specific Course in a specific semester and Print all courses in a specific semester .

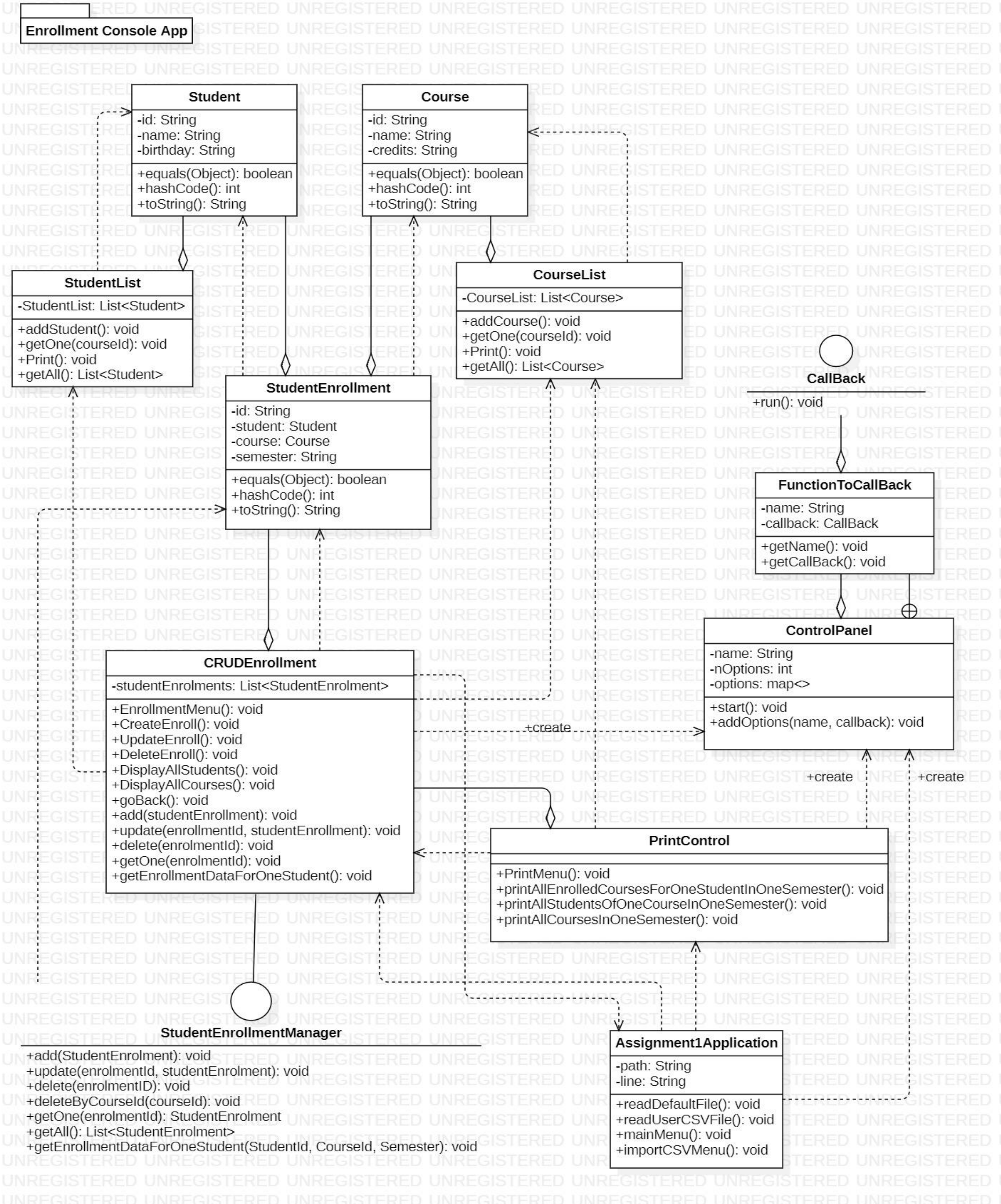
II. Use Case Diagram



III. Class Diagram

According to the client's requirement. We have these classes:

1. **Student**: display detailed information (include ID, name and birthday): auto added by reading a CSV file, do not have CRUD.
2. **StudentList**: store student data, display all info to a table.
3. **Course:** display detailed information (include courseID, name and Credits): auto added by reading a CSV file, do not have CRUD.
4. **CourseList:** store courses data, display all courses to a table.
5. **StudentEnrolment:** display detailed information (include Student name, course name and semester), need CRUD.
6. **CRUDEnrollment**:
    a. Add new enrollment. Enroll students into a course in a specific semester.
    b. Update old enrolment by changing the old course to a new course in that semester.
    c. Delete enrollment information.
    d. GetOne and GetAll Enrollment.
7. **PrintControl:** Print all the data to a CSV file according to the requirements.
    a. Print all enrolled courses for a student.
    b. Print all students in a course.
    c. Print all courses in a semester.
8. **ControlPanel:** To create a User menu. Allow users to pick options and run function immediately.
9. **Main:** Run program.


Enrollment Console App
Student
-id: String
-name: String
-birthday: String
+equals(Object): boolean
+hashCode(): int
+toString(): String
Course
-id: String
-name: String
-credits: String
+equals(Object): boolean
+hashCode(): int
+toString(): String
StudentList
-StudentList: List<Student>
+addStudent(): void
+getOne(courseId): void
+Print(): void
+getAll(): List<Student>
CourseList
-CourseList: List<Course>
+addCourse(): void
+getOne(courseId): void
+Print(): void
+getAll(): List<Course>
CallBack
+run(): void
StudentEnrollment
-id: String
-student: Student
-course: Course
-semester: String
+equals(Object): boolean
+hashCode(): int
+toString(): String
FunctionToCallBack
-name: String
-callback: CallBack
+getName(): void
+getCallBack(): void
ControlPanel
-name: String
-nOptions: int
-options: map<>
+start(): void
+addOptions(name, callback): void
CRUDEnrollment
-studentEnrolments: List<StudentEnrolment>
+EnrollmentMenu(): void
+CreateEnroll(): void
+UpdateEnroll(): void
+DeleteEnroll(): void
+DisplayAllStudents(): void
+DisplayAllCourses(): void
+goBack(): void
+add(studentEnrollment): void
+update(enrollmentId, studentEnrollment): void
+delete(enrolmentId): void
+getOne(enrolmentId): void
+getEnrollmentDataForOneStudent(): void
+create
+create
+create
PrintControl
+PrintMenu(): void
+printAllEnrolledCoursesForOneStudentInOneSemester(): void
+printAllStudentsOfOneCourseInOneSemester(): void
+printAllCoursesInOneSemester(): void
StudentEnrollmentManager
+add(StudentEnrolment): void
+update(enrolmentId, studentEnrolment): void
+delete(enrolmentID): void
+deleteByCourseId(courseId): void
+getOne(enrolmentId): StudentEnrolment
+getAll(): List<StudentEnrolment>
+getEnrollmentDataForOneStudent(StudentId, CourseId, Semester): void
Assignment1Application
-path: String
-line: String
+readDefaultFile(): void
+readUserCSVFile(): void
+mainMenu(): void
+importCSVMenu(): void

IV. System Design Analysis

1. **Import CSV file to read data.**

When running this console app. At the start, the user will have to choose between using a default CSV file to get data or importing their own CSV file by entering the path to that file. There is two thing to take notice:

- The user CSV file must be in the same direction with the app in order to successfully read the data.
- The data in the csv file must be written in this order: StudentID, Student Name, BirthDay, CourseID, Course Name, Credits, Semester.

After that all data will be read by using BufferReader and automatically added to the Student List, Course List and Enrollment List.

2. **Display All Students information.**

   There is no need for CRUD students, because all data will get from a CSV file. After successfully reading data, the console app will immediately display a table with full Student information (Include Student ID, Name and Birthday). There will be an option in the main menu to display this table again as well.

   After generating a constructor, getter, setter and toString for the Student class, we will implement StudentList with Array to store all the data and create GetOne and GetAll to easily access the data.

3. **Display All Courses information.**

   Same as Student class, there is no requirement for CRUD courses. All data will get from the CSV file that the user imports in and the console app will immediately display the Courses table with all the necessary information (include Course ID, Course name and Credits). There will be an option in the main menu to display this table again as well.

   After generating a constructor, getter, setter and toString for the Course class, we will implement CourseList with Array to store all the data and create GetOne and GetAll to easily access the data.

4. **CRUD Enrollment.**

When the user chooses the options Enrollment Menu. They can access these function:

Create new Enrollment: Enroll students to a new course in a specific semester, the user will be asked to input Student Id, Course Id and Semester to create an enrollment. If they enter invalid student id or course id, the system will ask them to do it again.

Update Enrollment: Change old course to a new course in that semester. The user will be asked to input Student Id and Semester they want to update. Then the app will display all the courses that students enrolled in that semester. Next, the user will choose the course Id they want to change, and enter a new Course ID to update the enrollment. If they enter invalid student id or course id, the system will ask them to do it again.

Delete Enrollment: Delete course that this student enrolled in a specific semester. The user will be asked to enter Student ID and semester first, then the app will display all courses that student enrolled. Next enter the course ID you want to drop, if valid the system will delete that enrollment. If they enter invalid student id or course id, the system will ask them to do it again.

Show all Enrollment: Display all the Enrollment to a table(include Enrollment ID, Student Name, Course Name and Semester).

5. **Export data to a CSV file.**
   When choosing options “Print Data to CSV files” on the main menu. They can access these function:
   - Print all student’s enrolled courses: Require to enter Student ID and semester.
   - Print all students in a course: Require to enter Course ID and semester.
   - Print all courses in a semester: Require to enter the semester.

   All data will automatically print to a csv file. The System will announce a path to access that file.

6. **Control Panel**
   This is where to create the system Menu. Using map to store the options(include key, function) and implement a callback interface to keep running the function when the user chooses it.
   To choose the options, the user will have to input the index value (which is also the key of the map options).
   For example, we have “1) Show all enrollment”, the user just needs to enter “1” to run that function.