|  |
| --- |
| RMIT University |
| Student Enrollment System |
| Software Architecture Design and Implementation |

|  |
| --- |
| Ngo My Quynh  S3836322 |

1. **Introduction**

This paper will demonstrate the implementation and design of the Student Enrollment Management System. This system was designed based on the CRUD software architecture which will help the user to create, delete, update and read the student enrollment. Besides from those, it will have more functions to support the viewing information of the student enrollment record. The document will contain the demonstration of the system’s workflow and also perform the detailed design of the class and use case diagram.

1. **Purpose**

* The main purpose of the Enrollment Management System report is to provide information about the project’s operation and functionalities and show the interaction between the system and the users. The system is focusing on managing the student enrollments, displaying the student enrollment and validating the input. Consequently, this CRUD system will help the user control the records more easily and at the same time help the user to prevent errors for example adding the student enrollment with the student id that may not be recognized by the system. Based on project requirements, the system will have the following intended users: academic assistant

1. **Scope**

* The academic assistance can enrol the student for a semester by providing three information: student id, course id and semester. We assume that all the student enrollment will be in 2021 and each year will have 3 semesters defined as 2021A, 2021B and 2021C. All the input will be verified and be added to the enrollment system.
* Moreover, the user can also delete a student enrollment record by giving three information the same as the adding function and all the input must be verified before the deleting.
* Furthermore, there is the updating function which will include 2 sub-functions: adding courses for students and delete courses for students if the student enrollment had existed before.
* Finally, there are many ways to display the student enrollment record such as viewing the student by course in one semester, displaying the course by a student in one semester, having all the courses in one semester and showing all the student enrollments.

**II. Functional Requirement**

1. **Methods description**
2. **Student Class**

* In the student class, we have a total of 9 methods, the first 6 methods are getter and setter methods for 3 attributes which are student id, student name and date of birth. In this project, we used getter methods to get the value of the attribute since the attributes are all set private, we didn’t use the setter methods. However, the setters methods were also important to later on using it on the unit testing for testing the getter methods.
* The toCSV() method was implemented from the Model class, this function was helping to combine all the attributes into the string separated by the comma to later be used for appending as a row to the CSV file.
* Besides that, we have two overdrive methods equals() and hashCode(). In the unit testing, we used the equals() to compare the student objects. We have to overdrive it since we want to check the object based on the property which the default function of equal() does not have. And since we override the equals method, we have also overdrive the hashCode since although they seem equals, they have to hash in the same bucket (same hashCode)

1. **Course Class**

* As the same as the Student Class, the Course class has 11 methods and 9 of them were also the getter and setter and the remaining methods were the same as Student Class: toCSV(), equals() and hashCode().
* The getter methods will return these attributes: course id, course name, number of credits and semester since all the attributes were set private and later on will be called in the StudentInMemory class. The setter methods were created for testing the getter methods in the test folder.
* The toCSV() as the student class, concat all the attributes into the string by comma which will support later in saving the data into CSV.
* Similar to the Student class, the equals() and hashCode() were overdriven to compare all the properties within the class for testing purposes.

1. **StudentEnrollment Class**

* The StudentEnrollment class has a total of 9 methods and 6 of them were getter and setter methods and the remaining were toCSV(), equals() and hashCode().
* The getter methods return the Student student, Course course and String semester which will be useful in the later methods and setter methods had the role on testing the getter methods,
* The toCSV() return the student id (by getting value from combining the getStudent() in StudentEnrollment Class and getStudentId() in Student Class), course id (by getting value from combining the getCourse() in StudentEnrollment Class and getCourseId() in Course Class) and semester (getSemester() in StudentEnrollment class) by comma and will be helpful in store the data in csv
* Similar to StudentEnrollment class, the equals() and hashCode() were overdriven to compare all the properties within the class for testing

1. **StudentManager Class**

* The StudentManager Class has 7 methods that are mainly used in managing the list of Student objects. The first 2 methods were getter and setter methods for the students’ list. The getter method will help to get the list of students since it will help get the private value and the setter method was used for testing the getter methods.
* The validStudent() will ask the user to enter the student id and then loop through the list of students to see if it matches any student id available in the list, if not, the system will display the error message and ask the user to enter the input again. The method will return the matching student id. This is the necessary method since it will check if the input of the user is valid so it will not display some weird data or return null point exception.
* The getStudentById(String studentId) will loop through the array list of student objects to return the student object. It will be used later on adding or making a change in StudentInMemory class since in these, it just requires the user to input student id but if you want to create the student enrollment, it must have the student object. For convenience, this method supports the system to get the student object more easily.
* showStudentList() will loop through the students’ list to display it all. Since there will be many students, when the users were required to enter the input, this list may help in preventing the error for the users in case they forgot the student id they want to enter
* readFile(String textFile) methods will read the file that the user specified, with the data obtained from the file, it will generate the Student object and add it into the array. This function mainly populated the data for the student list. If the file cannot be read, the error message will be displayed.

1. **CourseManager Class**

* The CourseManager Class has 7 methods and the first two are getter and setter, the remaining are validCourse(String semester), showCourseListBySemester(String semester), getCourseByIdAndSemester(String courseId, String semester), and readFile(String textFile)(the showCcoureList just used for test if the data is populate).
* The getter will return the list of Course objects since the list was set private and will be later on used for showCourseListBySemester(String semester). The setter methods were created to test the getter method.
* validCourse() function will display the list of course in the specific semester by called the showCourseListBySemester(String semester) and ask the user to enter the course id, With the input got from the user
* showCourseListBySemester(String semester) will loop through the list of courses with the matched semester since the user may not remember the course id when they are required to enter. Therefore, this function will help the user enter the input more accurately.
* getCourseByIdAndSemester(String courseId, String semester): The function will loop through the list of courses and return the course object that matches courseId and semester. This function will be used to create the StudentEnrollment since one of the attributes to create the student enrolment is Course. Therefore in creating the new student enrollment, the user will be asked to enter the course id so this function will make the creation easier.
* readFile(String textFile) methods will read the file that the user specified, with the data obtained from the file, it will generate the Course object and add it into the array. This function mainly populated the data for the course list. If the file cannot be read, the error message will be displayed.

1. **StudentMemoryManager Class**

* The StudentMemoryManager has 24 methods: getCourseByStudent(), addingCourseForStudent(), getOne(), delete(), getAll(), getCourseBySemester(), add(), deleteCourseForStudent(), execute(), ImportToCSV(), getStudentByCourse(), readFile(), update(), checkInput(), askingLoad(), validSemester(), checkStudentEnrollment(), createList(), getStudentById(), displayCourseOfStudent(), checkCourse(), display(), getStudentEnrollments().
* For the checkStudentEnrollment(String studentId, String semester, String courseId) methods, it will loop through the list of StudentEnrollment and return true if the student enrollment matches (the student enrollment exists). This will make sure that there is no duplicated student enrollment or mistake from the user.
* For the checkInput(int min, int max) method, the function will require the user to enter the input and check if it was a number and in the range of min to max. If the input does not match any conditions, the system will display the error message and repeat asking for input until it fulfils the requirement. After that, it returns the option number.
* For the add() method, the system will asked the user to enter the student id (called the method studentManager,validStudent() from StudentManger class to check if it valid), course id(called the method courseManager.validCourset() from CourseManager class to check if it valid) and the semester (also called the validSemester to validate). After that, with the given input, it will check if the student enrollment was added before, if not, it will display the error message and require the user to enter it again. The system will ask the user if they want to add another course. If the user enters Y/y, repeat the process, if the user enters N/n, the user will be back to the main menu. All the input from the user must be validated or else the system will ask the user to enter again and display the error message.
* For the delete() method, the system will display all the student enrollment lists and the user enters the enrol id that the user wants to delete. The input from the user will be validated using the checkOption() and then that student enrollment will be deleted from the array list. After that, it will ask the user if they want to repeat this option, if the user enters Y/y, it will repeat the process, if the user enters N/n, the user will be redirected to the main menu.
* For the update() method, the system will display the update menu which will be called in the Menu Class and ask the user to enter the option. Depending on the option addingCourseForStudent(), deleteCourseForStudent or exit update menu will be executed. The option will also be validated before that.
* For addingCourseForStudent(), this function is similar to the add() methods but the difference is it has the extra validation, the user had to enrol in the system before with any courses.
* For deleteCourseForStudent(), this function is similar to addingCourseForStudent(), the difference is the validation, the user had to enroll before.
* For getOne(), this function will display the list of student enrollment using the getAll() function. Then it will ask the user student id, course id and semester, all the input will be validated by using the studentManager.validStudent(), courseManager.validCourse() and validSemester(). After checking the input, the system will check if the student enrollment that the user wants to get the information exists. If the input meets all the conditions, all the information about the student enrollment will be displayed, else the error message is shown and requires the user to enter again.
* For the getAll() method, the function will loop through the student enrollment list and display it all. This function helps the user to prevent the users’ error when they enter the input related to student id, course id and semester
* For the createList() method, this function will create many student enrollments to add to the array list. This function will mostly be used for the testing part.
* For readFile(String textFile) methods will read the file that the user specified, with the data obtained from the file, it will generate the StudentEnrollment object and add it into the array. This function mainly populated the data for the course list. If the file cannot be read, the error message will be displayed.
* For the askingLoad() method, before going to the main menu, the system will ask the user wanted to use another file or default file to process. If the user enters Y/y, the default file will be read using the readFile() method and the data will be stored into the array list. If N/n, the user has to enter the file name that the user wants to use. Then the file will be read by using the readFile() and stored into the array list. And if the file does not exist, the error message will be displayed.
* For the display() method, the function will show the display menu called from the Menu class and with the option that the user chooses, the process will be executed included getCourseByStudent, getStudentByCourse and getCoursesBySemester(). This function just helps in displaying the menu for the user to choose and based on the condition, the method will be processed.
* For the validSemester(), the user will be asked to enter the semester and the input semester has to follow these patterns 2021A, 2021B, 2021C. We assume that the enrollment will occur just in 2021 and there are 3 semesters in one year. If the input does not meet the condition above, the system will display the error message and require the system to enter again until it meets the standard. The function will return the string semester. This function will mainly check if the semester is valid so it is quite necessary to, later on, not print out any weird data.
* For the getCourseByStudent() method, the system will ask the user to enter the student id which will be validated by process the getStudentById() and the semester which will be validated by validateSemester(). After the validation, it will loop through the array list of student enrolments and find the match student id and semester, and display it all. Then the system will ask if the user wants to save the list to the csv folder. If the user enters Y/y, the file will be created by using the ImportToCSV() methods. If the user enters N/n, the user will be redirected to the display menu.
* For the getStudentByCourse() method, the system will ask the user to enter the course id and semester. All the input will be validated by using the checkCourse() and validSemester(). After the validation, the system will loop through the list of the student enrollments and display the matched student enrollment. The system will ask if the user wants to save the record. If the user enters Y/y, the file will be created by using the ImportToCSV() methods. If the user enters N/n, the user will be redirected to the display menu.
* For the getCourseBySemester() method, the user will be asked to enter the semester and it will be checked by using the validSemester() method. The system will then go through the array list of student enrollment and display the matched student enrollment.
* For the execute() method, when the method was executed, the askingLoad() will be processed if the user wants to different file to store the data to the list of student enrollment and then the main menu will be displayed and based on the option, the method will be processed until the user choose the exit option to break the while loop.
* For the ImportToCSV(ArrayList<?> list, String textFileName, List<String> listOfProperties) method, first the system choose the file to write in by using the FileWriter by using the parameter textFileName to define. By using defined parameter listOfProperties, it will append the first row to csv as the column names. The remaining rows will be appended by looping through the list and use the toCSV() function. If the file cannot be created the error message will be displayed. This method is important since there are a total of 3 methods required to import to CSV file. Therefore, to make the code look cleaner, we create this function to prevent repetition.
* For getStudentById(), this function will ask the user to enter the student id then it will check if this student id has already enrolled in the system by looping through the array list of student enrollments. If it matched, it will return the student id. If it doesn't match, the system will display the error message and require the user to enter the input again until it is validated. This method is important since it will check if the student has enrolled before (using in the addingCourse() method and deleteCourse() method)
* For displayCourseOfStudent(), this function will take two parameters which is student id and semester to display the list of courses of the student in that semester by looping through the array list of student enrollments and display the matched one. This function will be used in the addingCourseForStudent and deleteCourseForstudent and the reason why it is necessary since it will help the user know what were the courses that they had enrolled in that semester and so that if they want to add course in their enrollment, it will not be duplicated course in same semester - preventing error from the users.
* For checkCourse() method, the function works as the same as the getStudentById method. It will ask the user to enter the course id and loop through the array list of student enrollment. If it matched, it will break the loop and return the course id. If don’t, the system will display the error message and require the user to enter again. This function is used in getStudentByCourse since it will check if the given course was already enrolled in the system which will prevent from displaying nothing for the output.
* For getStudentEnrollments() method, this is just the getter method to get the student enrollments list since it was set private. This is the necessary method since it will mostly be used in the testing part.

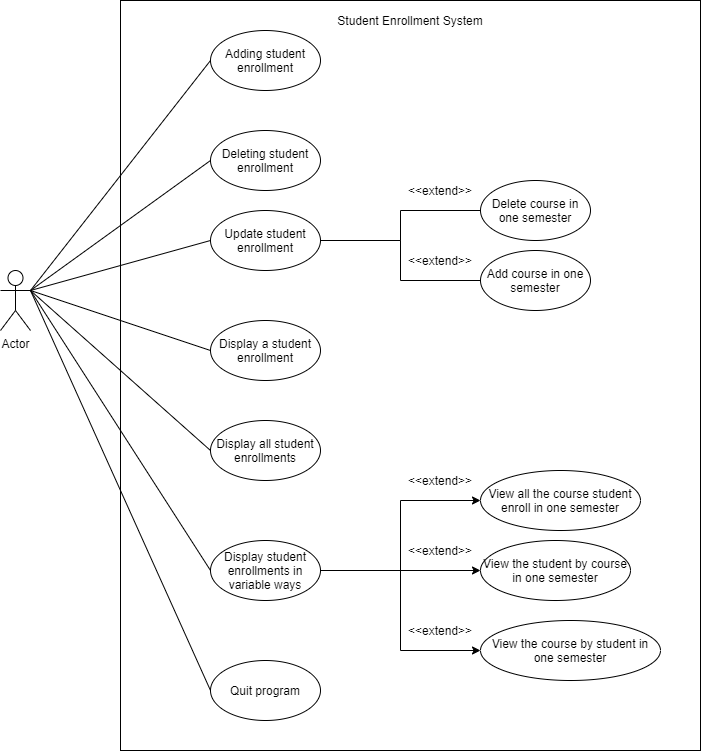
1. **Menu Class**

* The Menu class will have 3 methods: checkInput(), updateMenu(), displayMenu() and mainMenu().
* The checkInput() method will take two integer parameters min and max and ask the user to enter the option. With the given set in range min to max, the input will be checked if it is a number and in the range of it. If it does not meet the condition, it will display the error message and make the user enter again. This method is important since it can prevent the user from entering some mismatch option that is not available.
* The updateMenu() method will display the interface of the update menu and return the option of the user by using the checkInput() method that we had mentioned above. This method is important since it shows the interface of the update menu. Similar to the mainMenu() and the displayMenu() methods.

1. **Main Class**

* The main menu class has the role in executing the whole system

1. **Use case diagram**



1. Use case

|  |  |
| --- | --- |
| Use Case Name | UC01: Enroll student in one semester |
| Version | 1.0 |
| Goal | The information on student enrollment was added to the system |
| Summary | The user must input all the required information to accomplish adding the record |
| Actors | Admin Assistance |
| Precondition | Choosing the adding student enrollment option in the main menu |
| Trigger | - |
| Basic Course of event | 1. The system will display the main menu with 4 option and the user choose the first option 2. The user fulfilled all the information that has been required 3. The system then verified all the input from the user 4. After validation, with the provided information, the student enrollment will be added. 5. The users will be asked if they want to add to another course, if yes, the user will be asked to enter the course id, if not continue the process below. 6. The user will be asked if the user wants to add another student enrollment. If yes, the process will start over from here, if no, the user will be returned to the main menu 7. End of the use case |
| Alternative paths | Information is not valid   * The error message appeared * The system will require the user to enter the information again until it’s validated. |
| Post-conditions | * The student enrollment does not exist in the system * The system had already populated data from studentenroll.csv, student.csv and course.csv |
| Notes | Validation:   * The student id must be recognized in the system * The course id must be recognized in the system * The semester must be followed these pattern: 2021A, 2021B, 2021C * The student enrollment later add is not in the system before * In the main menu, the user only can enter in range 1 to 4 * In the action of continue, the user can only enter either Y/y or N/n |

|  |  |
| --- | --- |
| Use Case Name | UC02: Delete a student enrollment in one semester |
| Version | 1.0 |
| Goal | Delete the given information related to the record in the system |
| Summary | The user must input all the required information to accomplish deleting the record |
| Actors | Admin Assistance |
| Precondition | Choosing the delete option in the main menu |
| Trigger | - |
| Basic Course of event | 1. The system will display the main menu with 4 option and user choose the delete student enrollment option 2. The user fulfilled all the information related to the one that the user wants to delete 3. The system will verify the provided information 4. After validation, that student enrollment will be deleted from the system 5. The user be asked to continue the action or not, if yes the process will be repeated, if not, the user will be back to the meu 6. End of the use case |
| Alternative paths | Information is not valid   * The error message appeared * The system will require the user to enter the information again until it’s validated. |
| Post-conditions | * The student enrollment exist in the system anymore * The system had already populated data from studentenroll.csv, student.csv and course.csv |
| Notes | Validation:   * The student id must be recognized in the system * The course id must be recognized in the system * The semester must be followed these pattern: 2021A, 2021B, 2021C * The student enrollment later be deleted must be in the system before * In the main menu, the user only can enter in range 1 to 4 * In the action of continue, the user can only enter either Y/y or N/n |

|  |  |
| --- | --- |
| Use Case Name | UC03: Adding course for a student in one semester |
| Version | 1.0 |
| Goal | Adding the course to the student enrollment in one semester |
| Summary | The users must input valid information to successfully adding the course of the student in one semester |
| Actors | Admin Assistance |
| Precondition | Choosing the adding course option |
| Trigger | - |
| Basic Course of event | 1. The system will display the main menu and the user choose the update student enrollment option 2. The user will be redirected to the update menu and choose the adding course option 3. The user fulfilled the student id and the semester. This information will be validated by the system 4. After the validation, the system will display the list of course that the student already enrolled in that semester and required the user to enter the course id 5. After another validation, if the course is in the system, new student enrollment will be added to the system 6. The system then asked if the user wanted to repeat the system. If yes, repeat the whole process, if no, the system will redirect users to the main menu. 7. The user chooses the exit option in the menu 8. End of the use case |
| Alternative paths | Information is not valid.   * The error message appeared * The system will require the user to enter the information again until it’s validated. |
| Post-conditions | The adding course of student enrollment exists in the system |
| Notes | Validation:   * The student id must be recognized in the system * The course id must be recognized in the system * The semester must be followed these pattern: 2021A, 2021B, 2021C * The student enrollment has to be in the system before (same student id and same semester) * The course that later added must not appear in the same semester * In the main menu, the user only can enter in range 1 to 6 * In the update menu, the user only can enter in range 1 to 3 * In the action of continue, the user can only enter either Y/y or N/n |

|  |  |
| --- | --- |
| Use Case Name | UC04: Delete course for a student in one semester |
| Version | 1.0 |
| Goal | Delete the course for a student in one semester |
| Summary | The users enter some information related to the course to delete it successfully |
| Actors | Admin Assistant |
| Precondition | Choosing the delete for a course of a student in one semester |
| Trigger | - |
| Basic Course of event | 1. The system will display the main menu and the users choose the update student enrollment option 2. The user will be redirected to the update menu and choose the deleted course of a student in the one-semester option 3. The system then asked the user to enter the student id and the semester 4. After the validation, to see if the record exists. The list of courses that the student has already enrolled in the semester will be displayed 5. The user then is required to enter the course id and that information will be checked if it is valid. 6. After the second validation, the course will be removed from the system 7. The system then asks if the user wants to continue the action. If yes, the whole process will be started again. If no, the user will be back to the main menu and choose the exit option 8. End of the use case |
| Alternative paths | Information is not valid.   * The error message appeared * The system will require the user to enter the information again until it’s validated. |
| Post-conditions | The course of the student in that semester will not exist in the system |
| Notes | Validation:   * The student id must be recognized in the system * The course id must be recognized in the system * The semester must be followed these pattern: 2021A, 2021B, 2021C * The student enrollment has to be in the system before (same student id and same semester) * The course that later deletes must exist * In the main menu, the user only can enter in range 1 to 6 * In the update menu, the user only can enter in range 1 to 3 * In the action of continue, the user can only enter either Y/y or N/n |

|  |  |
| --- | --- |
| Use Case Name | UC05: Viewing the student by course in one semester |
| Version | 1.0 |
| Goal | The user view the lists of a student by course in one semester |
| Summary | The user enters some information about the student and the semester that they want to view the list of course |
| Actors | Admin Assistance |
| Precondition | Choosing the viewing the list of courses by a student in one semester option |
| Trigger | The user wants to view the student by course |
| Basic Course of event | 1. The system will display the main menu and the user chooses the display student enrollment option 2. The user then be redirected to the display menu and choose the “Course by a student in one semester” option 3. User fulfilled the student id and the semester and then the system will check if the student id and the semester are valid 4. After the checking, the system will display the list of course by that student in that semester 5. The display menu will appear, the user then choose the exit option 6. The main menu will be shown and the user then choose the exit option again 7. End of the use case |
| Alternative paths | Information is not valid.   * The error message appeared * The system will require the user to enter the information again until it’s validated. |
| Post-conditions | - |
| Notes | Validation:   * The student id must be recognized in the system * The course id must be recognized in the system * The semester must be followed these pattern: 2021A, 2021B, 2021C * The student enrollment has to be in the system before (same student id and same semester) * In the main menu, the user only can enter in range 1 to 6 * In the display menu, the user only can enter in range 1 to 4 |

|  |  |
| --- | --- |
| Use Case Name | UC06: Displaying the course by a student in one semester |
| Version | 1.0 |
| Goal | The user can view the list of course by the student in one semester |
| Summary | The user enters some information about the student and the semester that they want to view all the course |
| Actors | Admin Assistance |
| Precondition | The user goes to the display student enrollment in variable way option and then choose the display all the course by a student in one semester |
| Trigger | The user wants to view all the course of the student in the semester |
| Basic Course of event | 1. The system will display the main menu and the user chooses the display student enrollment option 2. The user will be redirected to the display menu and choose the display the course by a student in one semester 3. The user fulfilled the student id and course id 4. After the validation, the system will display the list of the courses by the student in the semester 5. The display menu will appear, the user then choose the exit option 6. The main menu will be shown and the user then choose the exit option again 7. End of the use case |
| Alternative paths | Information is not valid.   * The error message appeared * The system will require the user to enter the information again until it’s validated. |
| Post-conditions | The system displays all the course by the student in that semester |
| Notes | Validation:   * The student id must be recognized in the system * The course id must be recognized in the system * The semester must be followed these pattern: 2021A, 2021B, 2021C * The student enrollment has to be in the system before (same student id and same semester) * In the main menu, the user only can enter in range 1 to 6 * In the display menu, the user only can enter in range 1 to 4 |

|  |  |
| --- | --- |
| Use Case Name | UC07: Displaying all courses in one semester |
| Version | 1.0 |
| Goal | Viewing all the course that the student enroll in the semester |
| Summary | The user enters the semester about the list of course that the user wants to view |
| Actors | Admin Assistant |
| Precondition | The user goes to the display student enrollment in variable way option and then choose the display all the course in one semester |
| Trigger |  |
| Basic Course of event | 1. The system will display the main menu and the user chooses the display student enrollment variably in option 2. The system will be redirected to the display menu and choose the display all course that student enrol in one semester 3. The student fulfilled the semester 4. After the validation, the system will display the list of the courses in the semester 5. The display menu appeared and the user choose the exit option 6. The system will be redirected to the main menu and choose the exit option 7. End of the case |
| Alternative paths | Information is not valid.   * The error message appeared * The system will require the user to enter the information again until it’s validated. |
| Post-conditions | The system display all the courses in the semester |
| Notes | Validation:   * The student id must be recognized in the system * The course id must be recognized in the system * The semester must be followed these pattern: 2021A, 2021B, 2021C * The student enrollment has to be in the system before (same student id and same semester) * In the main menu, the user only can enter in range 1 to 6 * In the display menu, the user only can enter in range 1 to 4 |

|  |  |
| --- | --- |
| Use Case Name | UC08: Displaying all the student enrollment |
| Version | 1.0 |
| Goal | Viewing all the student enrollment |
| Summary | The user chooses the option |
| Actors | Admin Assistant |
| Precondition | The system can have the data from the file studentenroll.csv and the user have to choose the display all the student enrollment in the main menu |
| Trigger | - |
| Basic Course of event | 1. The system will display the main menu and then choose the display all the student enrollment option 2. The system will display all of the student enrollments 3. The system will direct the user to the main menu 4. The user then use the exit option 5. End of the use case |
| Alternative paths | Information is not valid.   * The error message appeared * The system will require the user to enter the information again until it’s validated. |
| Post-conditions | - |
| Notes | Validation:   * The student id must be recognized in the system * The course id must be recognized in the system * The semester must be followed these pattern: 2021A, 2021B, 2021C * The student enrollment has to be in the system before (same student id and same semester) * In the main menu, the user only can enter in range 1 to 6 * In the display menu, the user only can enter in range 1 to 4 |

|  |  |
| --- | --- |
| Use Case Name | UC09: Displaying a student enrollment |
| Version | 1.0 |
| Goal | The user wants to display the information about the student enrollment |
| Summary | The system displays information about student enrollment |
| Actors | Admin Assistance |
| Precondition | * The data must be manipulated from the studentenroll.csv * The user chooses the display a student enrollment |
| Trigger | - |
| Basic Course of event | 1. The system displays the main menu and the user chooses the display a student enrollment 2. The user fulfilled the student id, course id and semester 3. After the validation, the system will display all the information about the student and the course 4. The system will direct the user to the main menu and the user choose the exit option 5. End of the use case |
| Alternative paths | Information is not valid.   * The error message appeared * The system will require the user to enter the information again until it’s validated. |
| Post-conditions | - |
| Notes | Validation:   * The student id must be recognized in the system * The course id must be recognized in the system * The semester must be followed these pattern: 2021A, 2021B, 2021C * The student enrollment has to be in the system before (same student id and same semester) * In the main menu, the user only can enter in range 1 to 6 * In the display menu, the user only can enter in range 1 to 4 |

1. Activity Diagram

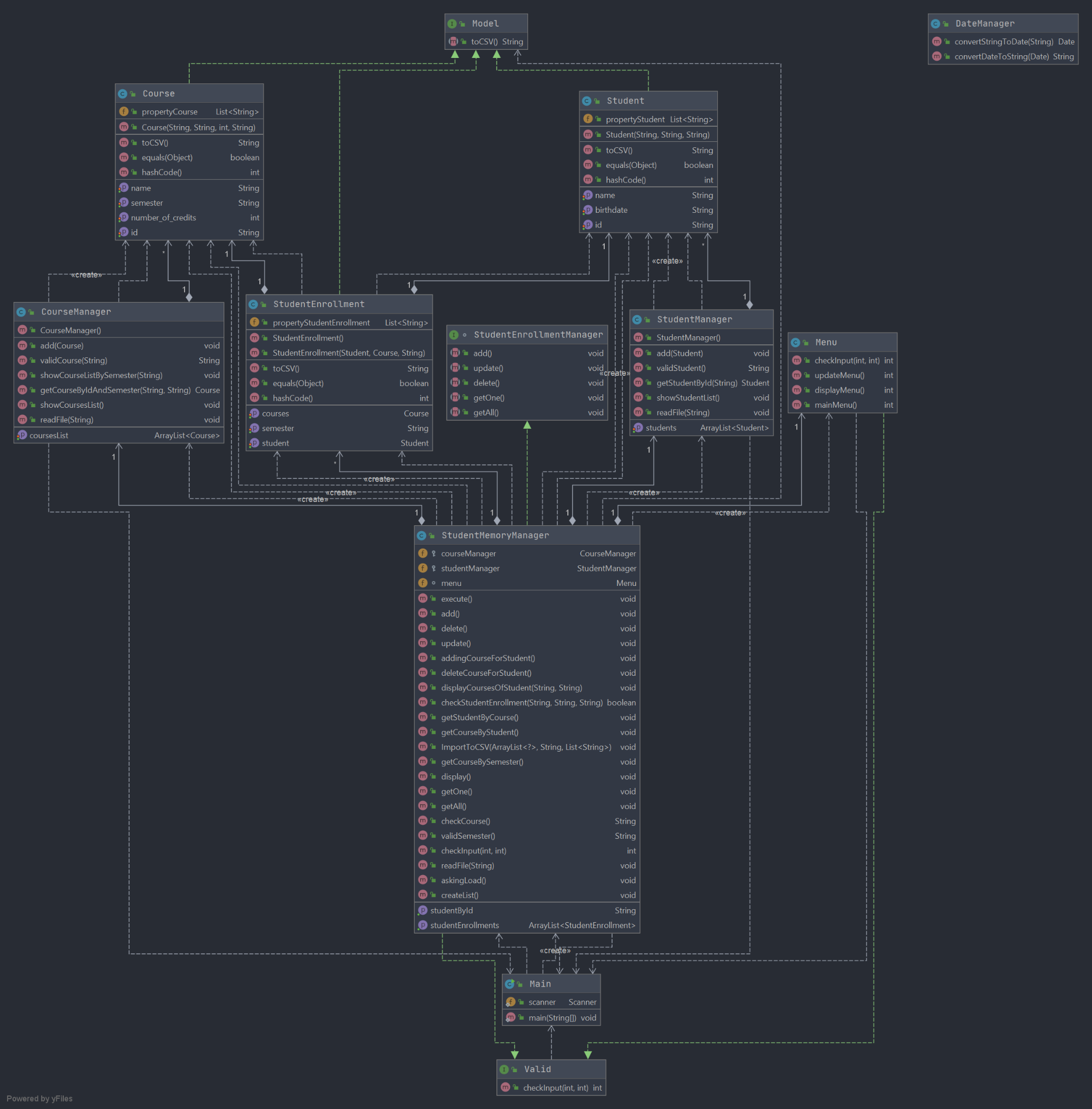
The following diagram demonstrates how users interact with the student enrollment system. First the system will display the main menu interface and list all the options for the user to choose. After entering their option, the system will check if the input is valid. If the input is not valid, the system will display the error and require the user to enter again. If it validates, depending on the choice of the user, the system will process the methods. Below are the activity diagrams for each feature of the users. Except for displaying all the student enrollments, and exit option, all the process needs the user to fulfill some information to continue the process. And all the input must be validated by the system. The system will only shut down when the user chooses the exit option in the main menu.

Activity Diagram for Student enrollmentGraphical user interface, application, Teams

Description automatically generated

III. Software Architecture

1. Class Diagram



* The following class diagram included all the classes and their attributes, also functionalities to process
* The Student has three attributes included id, name, birthDate in String. All the attributes were set private to prevent the customer from making the adjustment. The Student has the aggregation relationship - detaily composition aggregation (one to one relationship) with the StudentManager since the StudentManger contains the attribute students list which is defined as private ArrayList<Student>. If the Student class does not exist, the StudentManager cannot exist too. The Student class also implements the method toCSV() from the interface Model, this function will be helpful later on to generate the attributes of the Student and import to the csv file. The Student class also has the composition aggregation - detaily one to one - relationship with StudentEnrollment.
* The Course has 4 attributes which are id, name, number\_of\_credits and semester in String. The Course will have the toCSV() method which was implemented from the interface Model, this function will later on concating all the attributes into string and append as a row to the csv file. The Course has the aggregation relationship - detaily composition aggregation (one to one relationship) with the CourseManager since the CourseManager - similar to the StudentManager contains the list of courses - defined as private ArrayList<Course>. If the Course does not exist, the CourseManager will not exist too since the creation of the list is Course object. The Course class also has composition aggregation - detaily one to one - relationship with StudentEnrollment.
* The Student class has the associate with the Course class. Therefore, there is an associate class called StudentEnrollment to define the relationship between Student and Course class by providing the semester attributes in String related to the associate relationship. And also, the StudentEnrollment class implements the toCSV() method from the Model interface.
* The interface Model has one method called toCSV(). This interface class has the Realization relationship with the Student, Course and StudentEnrollment.
* StudentManager class has one attribute called students defined as private ArrayList<Student>. Besides the setter and getter, there are some main method that the StudentManager class included: validStudent(), showStudentList(), readFile(), getStudentById() and all the methods are set public since it will easier for the testing.
* CourseManager class has one attribute called courseList defined as private ArrayList<Course> since the customer cannot make the modification. Besides that, it have 6 methods which are add(Course), validCourse(String), showCourseListBySemester(String), getCourseByIdAndSemester(String, String), shoeCourseList(), readFile(String). All the methods was set in public to later used for the testing purpose
* StudentMemoryManager class has 4 aggregation relationships - detaily composition aggregation. Relationship with one to one with CourseManager, StudentManager, Menu since one StudentMemoryManager just has one course, student manager and menu, and one course, student manager and menu just belong to one student memory manager. Relationship one to many with StudentEnrollment since the Student MemoryManager can contain many StudentEnrollment and if the StudentEnrollment does not exist, the StudentMemoryManager cannot exist. Besides that, the class has total 21 methods which are execute(),, addingcourseForStudent(), deleteCourseForStudent(), displayCourseOfStudent(), checkStudentEnrollment(), getStudentByCourse(), getCourseByStudent(), ImportToCSV(), getCourseBySemester(), display(), checkCourse(), validSemester(), checkInput(), readFile(), askingLoad() and createList(). All the methods were set in public for easier testing. This class also implements methods from the StudentEnrollmentManger interface which are getOne(), getAll(), update(), delete() and add().
* StudentEnrollmentManager interface contains 4 methods which are getOne(), getAll(), add(), update(), delete(). The interface StudentEnrollment has the Realization relationship with the StudentMemoryManager.

The Menu has 3 methods which are mainMenu(), updateMenu(), displayMenu().All the methods were set public for testing purposes. The checkInput() method in Menu was overdrive from the Vali interface class.

* Valid interface contains one method checkInput(). This interface class has the Realization relationship with Menu.
* Main contains one public static final scanner and the main() to execute the whole system

Link github to the project : https://github.com/s3836322/ConsoleApp