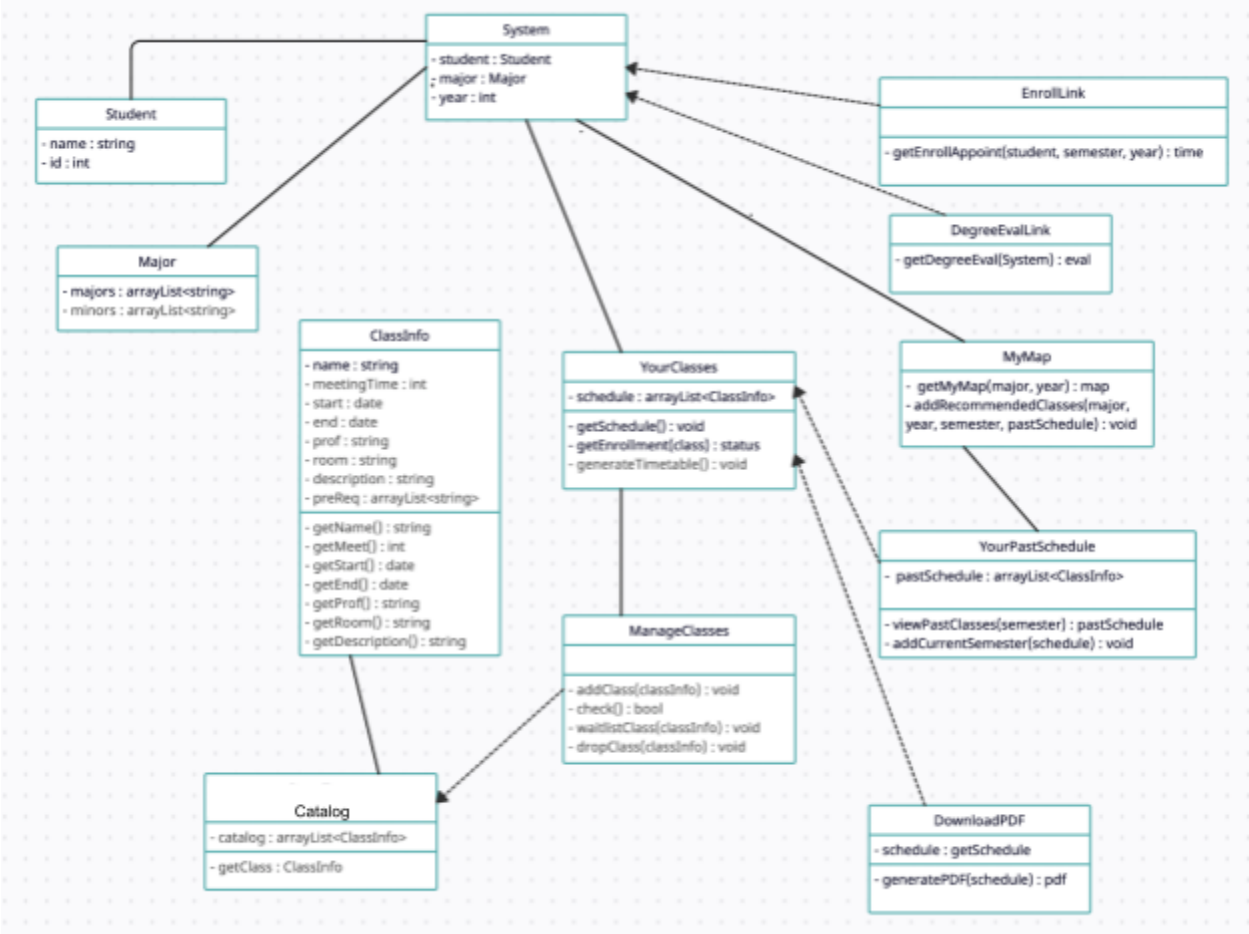
SDS Test Plan

By: Sophie Krivonosov
Quincy Kapsner
Rage Dizon

March 22, 2023

1 Software Design Specification

1.1 Updated Diagram



1.2 Description

This UML Class Diagram has the System as the main class. This class contains basic student information like the name, major, and year. The System has multiple associations and dependencies. The EnrollLink and DegreeEvalLink classes are dependencies that use the information from the system class. The student, major, yourClasses, and myMap classes are

associations that build off the information in the system class. Additionally, the classInfo class has all the information about a given class, that relates to the whole list of a student's classes, access to managing classes, and the catalog. Below is a list of all the classes, attributes, and operations and their functions.

- Classes
    - System: Overarching parent class that holds on the global variables
    - Student: Abstraction class holding student's name and ID
    - Major: Abstraction class holding a student's collection of majors and minors
    - ClassInfo: Class representation of all the characteristics of a course in the catalog
    - Catalog: A collection class that holds all the courses sorted by semester
    - ManageClasses: Management class that includes all the functions that students use to interact with the catalog of courses
    - YourClasses: A display class that provides images for schedules in different formats
    - DownloadPDF: Simple class that allows students to download pdf version of their course schedule
    - DegreeEvalLink: Link class that leads to the external tool for evaluating their own degree
    - MyMapLink: Link class that leads to the external tool for mapping out their courses based on major and year
    - EnrollLink: Link class that leads to external tool for checking when a student's enrollment appointment is
- Attributes
    - name: string containing the name
    - id : integer holding the student's id
    - Student: abstraction class object holding the name an did
    - meetingTime: custom variable holding a date and time for a class's meeting
    - start: date variable for when the course starts
    - end: date variable for when the course ends
    - prof: string containing the professor of the course's name
    - room: string containing the room name and number
    - description: cosmetic string containing an overview of the course
    - preReq: a list of strings that show which courses needed to be taken before the viewed course
    - major: class instance that contains a student's majors and minors
    - majors: list containing majors
    - minors: list containing minors
    - catalog: list of all possible classes
    - schedule: list containing all instances of ClassInfo to display courses being taken
    - year: integer that displays the year of the student

    - semester: string that indicates the seasonal semester they are currently on
- Operations
    - getName(): get method for name
    - getMeet(): get method for meetingTime
    - getStart(): get method for start
    - getEnd(): get method for end
    - getProf(): get method for prof
    - getRoom(): get method for room
    - getDesc(): get method for desc
    - getEnrollAppoint(): get method for the enrollment appointment time
    - getMyMap(): get method to get the external tool MyMap
    - getDegreeEval(): get method to get the external tool degree eval
    - getClass(ClassInfo): get method for a class
    - addClass(classInfo): get method to add a class
    - check(): convenience method to help addClass, in checking if a class can be added
    - dropClass(ClassInfo): use method for dropping a class
    - waitlist(classInfo): use method for waitlisting a class
    - getSchedule(): get method to display a schedule
    - getEnrollment(): get method to get the status of enrollment
    - generateTimetable(): use method to display a timetable view
    - generatePDF(): use method for getting a PDF of their schedule

2 Verification Test Plan

2.1 Unit Testing

Unit 1: EnrollLink: The EnrollLink class fetches the user's enrollment appointment for the selected semester.

- Test 1: Active student with appointment available. Should display the appointment time and date.
    - Student example = new Student ("Example Student", 000000001)
    - System.student = example
    - example.enrollAppointment = "April 21st, 2023, 2:30 pm"
    - getEnrollAppoint(example, "Fall", 2023) → "April 21st, 2023, 2:30 pm"

- Test 2: Student graduating the selected semester. Should show nothing because there is no appointment.
    - Student grad = new Student ("Graduating Student", 000000002)
    - System.student = grad
    - grad.enrollAppoint = null
    - getEnrollAppoint(grad, "Fall", 2023) → null

- Test 3: Appointment time is not available yet. Should show nothing because there is no appointment.
    - Student example2 = new Student (“Example2 Student”, 000000003)
    - System.student = example2
    - example2.enrollAppoint = null
    - getEnrollAppoint(example2, “Fall”, 2023) → null

Unit 2: YourClasses: YourClasses moderates all the schedule related functions of the system

- Test 1: getSchedule()
    - schedule = *nonempty schedule*
    - getSchedule() → null
        - *Should print a proper schedule*
    - schedule.clear()
    - getSchedule() → null
        - *Should print an empty schedule*
- Test 2: getEnrollment(testClass)
    - *test class will be open for enrollment in this case*
    - for(int i = 0; i < schedule.size(); i++)
        - if(schedule.at(i) == ClassInfo)
            - test = schedule.at(i)
            - test → enrolled
    - testClass = *change class to full class*
    - for(int i = 0; i < schedule.size(); i++)
        - if(schedule.at(i) == ClassInfo)
            - test = schedule.at(i)
            - test → waitlisted
- Test 3: generateTimetable()
    - schedule = *nonempty schedule*
    - generateTimetable() → null
        - *Should print a proper timetable*
    - schedule.clear()
    - generateSchedule() → null
        - *Should print an empty timetable*

2.2 Functional Testing

Feature 1: Past Schedule Testing: This test tests the catalog feature. The catalog feature is an array of class information that includes the course’s specifics, such as name, meeting time, start date, end date, etc. of the course. My first test assigns the name, meeting time, and start date to random integers and strings, and tests that when the getClass function is called, the same is outputted. For the second test, I set the name, meeting time, and start date to null to determine if the method would run even if no input was provided.

- Test 1:
    - name = “CS 250”
    - meetingTime = 5:30
    - start = 1/1/23
    - getClass(classInfo)
        - name = “CS 250”
        - meetingTime = 5:30
        - Start = 1/1/23
- Test 2:
    - name = null;
    - meetingTime = 0;
    - start = null
    - getClass(classInfo)
        - name = “ “
        - meetingTime = 0
        - start = “ “

Feature 2: Generate PDF Testing: This test tests the generate PDF feature. The generate PDF feature is responsible for outputting a file in the PDF format of the class schedule listed in the system. In order for this to occur, we must access the getSchedule function to generate the schedule that will be printed. For my first test, I set the classInfo to null, so that when I call the getSchedule function, it should return null. Then, when generatePDF is called, it should return an empty page as there are no classes. For my second test, I generated the PDF with the name and major accessed from the student function, and then used getSchedule to get the list of classes. The expected output should return the timetable of classes with the student’s name and major.

- getSchedule(classInfo)
- generatePDF(name, major, schedule)
    - Test Case: classInfo = null;
      getSchedule(classinfo) → return null
      generatePDF(name, major, schedule) → return empty page
    - Test Case: classInfo = yourClasses;
      getSchedule(classInfo) → return list of classes
      generatePDF(name, major, schedule) → return timetable of classes

2.3 System Testing

System 1: Quick-Adding the classes recommended by the user’s MyMap. SDSU provides a guide called “MyMap” that recommends what classes to take what semester based on your major. The Quick-Add feature will read the user’s already completed classes and add the selected semester’s recommended classes. The feature should be able to take user information from the

system or work with custom inputs for unofficial majors or minors. GE and Explorations classes are not automatically added, it is up to the student to select and add those.

- Test 1: Using system information
    - Student example = new Student (“Example Student”, 000000001)
    - System.student = example
    - Major compsci = new Major(<”Computer Science”>, <>)
    - System.major = compsci
    - System.year = 2023
    - YourPastSchedule takenClasses = new YourPastSchedule (<”MATH 150”, “CS 150”, “CS 150L”>)
    - example.addRecommendedClasses(example.major, example.year, “Spring”, takenClasses) → the classes CS 160, CS 160L, and MATH 151 should be added to the example student’s fall courses or waitlisted if they are full
- Test 2: Using custom information
    - Student example2 = new Student (“Example2 Student”, 000000002)
    - System.student = example2
    - example2.addRecommendedClasses(“Computer Science”, 2023, “Spring”, <”MATH 150”, “CS 150”, “CS 150L”>) → the classes CS 160, CS 160L, and MATH 151 should be added to the example student’s fall courses or waitlisted if they are full

System 2: This series of tests covers the “conventional” way of adding and dropping classes via the catalog, which is seen by many students as the main function of the system.

- Student sample = new Student (“Fred”, 000000002)
- System.student = sample
- Major sampMajor = new Major(<”Psych”>, <”Lang”>)
- Test 1: Add correct classes
    - addClass(PSY101)
    - addClass(RWS280)
    - schedule.size() → 2
    - schedule.clear()
- Test 2: Add incorrect classes
    - addClass(CS160)
    - addClass(MUSIC110)
    - schedule.size() → 0
    - schedule.clear()
- Dropping Classes
    - addClass(PSY101)
    - addClass(RWS280)
    - addClass(AFRAS101)
    - addClass(MATH150)

- Test 3: Drop nonexistent classes
    - dropClass(CS150)
    - dropClass(HIST110)
    - schedule.size() → 4
- Test 4: Drop proper classes
    - dropClass(AFRAS101)
    - dropClass(MATH150)
    - schedule.size() → 2