Software Requirements Specification

for

Student Course Notification Interface (may change name)

Prepared by Group 1
Joshua N.
Patrick M.
Tyler P.
Antoine M.
Daniel B.

CECS 343 - Spring 2024

5/3/2023

1. Introduction

1.1. Purpose

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>

Our product is a notification system that is meant to be in a similar style to the CSULB Student Center. The product emulates certain student and admin actions that would be performed by the Student Center alongside some added features that are not present in the current Student Center. We have implemented a notification system for when students in a waitlist are notified of being added into a course or a new section of a course has opened. Notifications of new sections go to all students regardless of whether or not the student is going to take that course.

1.2. Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

For our project, we try to get our documentation and source code to be concise, clear, and clean. To reiterate, we want our files relating to this project to be read and understood by anyone. That said, we do not have any ‘high-priority’ fonts or highlighting as we simply have everything with its own priority. Each function of our project is about as important as any other, so the priority at the time would be based on what we wanted to work on at the moment.

1.3. Intended Audience and Reading Suggestions

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

Currently, our intended audience are in the order of students, teachers, admin, and anyone else that would like to hear about how we can improve the current Student Center. The project documentations should contain our thought process in planning and creating the project, including the use cases, classes, and more. Each part should be organized into its

own section depending on where the title of the part suggests. For example, this part is ‘1.3. Intended Audience…’ we are notifying the reader of who the intended audience is.

1.4. Product Scope

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>

The project is an emulation of the CSULB Student Center that adds certain features that may be implemented into the current Student Center to improve the student experience. Most notably, we came up with this idea with the thought “what if the student was notified of new sections instead of having to check from time to time”, and that sparked the whole idea of creating that notification system ourselves. We believe CSULB may be interested in the idea, and this project will show how it can be done so that they may have their own implementation in the future.

1.5. References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

Our only references are the CSULB Student Center and the data structures that were taught at CSULB. The data structures that were used were the Python dictionaries and lists for keeping track of Student, Admin, and course information. An observer is used to looking at the notifications.

2. Overall Description

2.1. Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

The product is a result of us wanting more features from the current CSULB Student Center (Spring 2024), and so this product could be classified as a 'follow-on' member. However, as we have stated in previous sections, this project will likely only serve as a demonstration or example of how certain features can be implemented. But, since we are also emulating a Student Center of a sort, it could also be used to do that.

2.2. Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

There are two separate major sections of our project that the user can perform depending on if they are a student or admin. For the student side, once they are able to login, their functions include viewing classes, adding/removing classes, and viewing notifications. For the admin side, depending on their permissions, the admin should be able to create a course, add/remove a student from a course, and create/delete students.

2.3. Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

The software, in theory, should be able to operate how the current Student Center for CSULB operates now. Students and Admin should be able to access and do specific functions within the Student Center. If this project were to be fully implemented for use, then perhaps student, course, and admin data would be stored on a separate database instead of on Python dictionaries and lists. For now, those methods of handling data should be sufficient for demonstration purposes.

2.4. Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

Certain constraints that we had to work out are the speed, when running on Replit you can expect several seconds of delay before the screen responds to input. We have not tested the source code on anything else other than Replit. For the project to fully work, you would need to have PyGame and Python installed.

2.5. User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

The program itself is very easy to use, so a user manual does not seem necessary for the use of the program. However, we will consider adding a 'help' menu or a paper manual when presenting the project.

2.6. Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

The program, if implemented into a school or university setting, will likely depend on documentation and resources already present from that environment to enter necessary data for the program to work. Adding students and admins will take quite some time. Many of the functions in our program also depend on each other, for example the Admin, Student, and Course classes depend on each other to exist, or else many of their functions will simply not make sense.

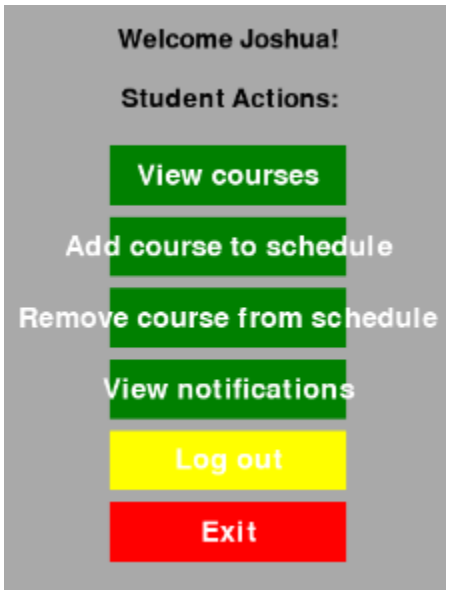
3. External Interface Requirements

3.1. User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface

is needed. Details of the user interface design should be documented in a separate user interface specification.>

The user interface is a graphical user interface, where the user of the program can click on buttons and type their inputs on the screen instead of the console. When first booting the program, the user should be met with three buttons with the prompt “Select your user”. The options are Student, Admin, and Exit. Once they select their user, they can login, and perform several actions that are also displayed as buttons. The “Exit” button always ends the program. Here are how the buttons appear for some of the Student functions:



3.2. Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

There are no specific or proprietary hardware interfaces for the software product, however, this product has mainly been tested on desktop and laptop computers, so it should work best on those devices.

3.3. Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and

the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

The software interfaces of this project include the Python dictionaries and lists containing the student data, admin data, and course data. Notifications also have their own lists. An observer is used to update the notifications list.

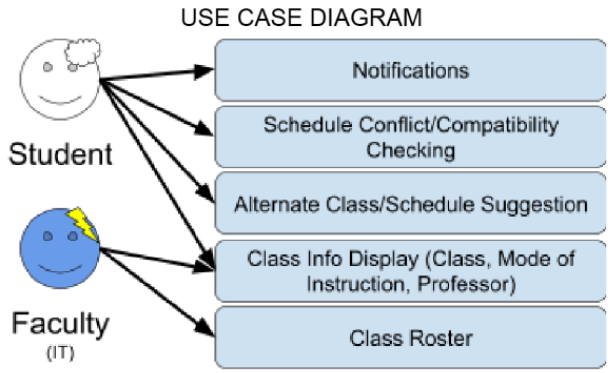
3.4. Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

Our product does not have any communication interfaces at this time. Communications interfaces can be implemented into our program and mock email addresses are used to log into the program. When a program like this is actually implemented for serious real-life applications, some level of encryption like HTTP would definitely be necessary.

4. System Features

*<This template illustrates organizing the **functional requirements** for the product by system features, the major services provided by the product. You may prefer to organize this section by **use case**, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>*



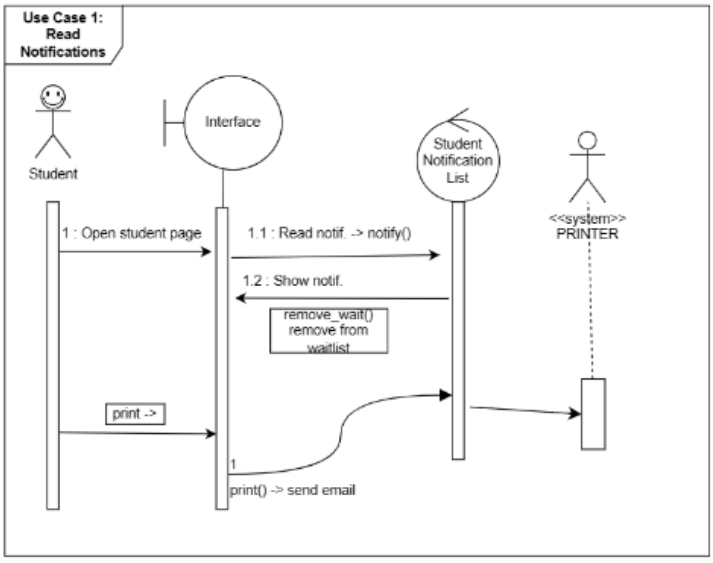
Use Case 1	Notifications
Actors	Students of CSULB
Description	Allows the user to be notified of new sections of desired classes.
Pre-Conditions	The user must be waitlisted for another section or registered in another section of the same class.
Flow of Control	1. User on their mobile device or desktop receives a notification (the message is received on the Canvas inbox folder)
Post-Conditions	User is sent an email notifying the user of the new section.
Error-Conditions	1. *Technical error*: unable to send a notification to users device. 2. Technical errors displaying incorrect information.
Nonfunctional Requirements	1. Login to Canvas and notifications on. 2. Performance: the notification should be displayed as soon as availability of class (Fast notification time).

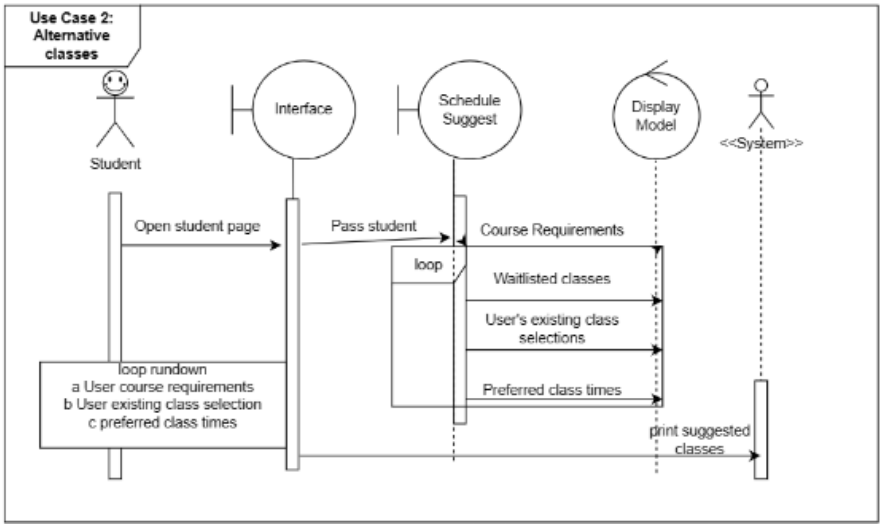
Use Case 2	Schedule Conflict/Compatibility Checking
Actors	Students of CSULB
Description	Informs the user whether a class they are about to sign up for will conflict with their schedule or their course requirements.
Pre-Conditions	The student must be logged into the Student Center and must navigate the schedule of classes.
Flow of Control	1. User navigates to the course selection page. 2. User selects a course. 3. If the course is not included in their major, minor, or general education requirements, inform the user by visually displaying a warning. 4. If the course conflicts with their existing schedule or the same class was already taken, display a warning before proceeding to Use Case 3 (Alternate Class/Schedule)
Post-Conditions	The student can see a notification whether the class they're selecting will conflict or not.
Error-Conditions	- User is not warned/informed in the case of lacking conflicts - User is warned/informed despite lacking any conflicts or identical courses
Nonfunctional Requirements	1. *Performance*: Calculated conflicts onto the schedule already made by the student. 2. *Usability*: The displayed information is clear and UI manageable when cross referencing to other viable classes.

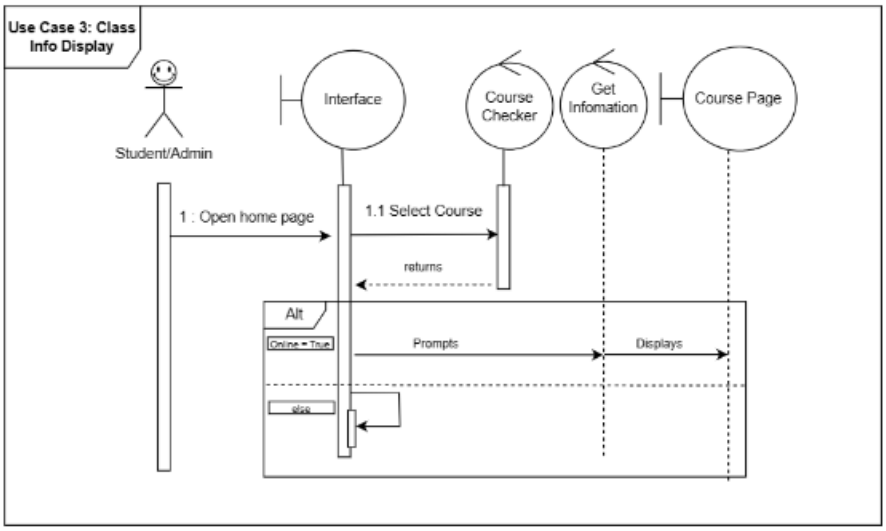
Use Case 3	Alternative Classes / Schedules
Actors	Students of CSULB
Description	Provides users a suggestion of courses to take or alternate sections to build a better schedule.
Pre-Conditions	Students must be logged into the Student Center and must be looking at their schedule of classes.
Flow of Control	1. User navigates to the course selection page. 2. User either selects a course or views their current schedule 3. The page should provide alternative courses and alternate sections relevant to the user, based on the following: a. User's course requirements b. User's waitlisted classes c. User's existing class selections d. User's preferred class times
Post-Conditions	The user is provided multiple classes and sections or alternate schedules that are relevant and do not provide conflict
Error-Conditions	- Suggestion of irrelevant/unrelated courses to the user in course requirements or already selected courses - Failure to provide suggestions - Suggestion of a schedule that includes conflicting courses (identical classes or conflicting times)
Nonfunctional Requirements	1. *Usability*: UI clear and organized to the user. 2. *Performance*: Navigation and loading is fast when navigating the Student Center.

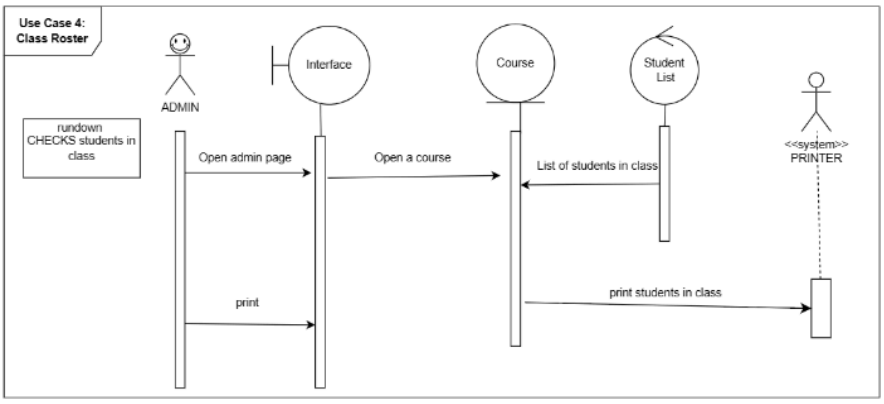
Use Case 4	Class Info Display (Class, Mode of Instruction, Professor)
Actors	Both the CSULB IT DIvision and the Students of CSULB
Description	Informs the user of general class information, such as the course title, department, mode of instruction, and professor.
Pre-Conditions	The user must select the course to view its information.
Flow of Control	1. User navigates to course selection page 2. User selects a course. 3. Displays all relevant information.
Post-Conditions	The user is able to read all relevant information pertaining to the course.
Error-Conditions	- Not being able to retrieve data from the database associated with it - Displaying the incorrect information (such as another class's information)
Nonfunctional Requirements	- *Structure*: Must be structured in a manner that is easy to read and phrased in an understandable manner. - *Relevancy*: Must be able to focus the user's attention on relevant information and avoid filling the page with "bloat" or irrelevant information

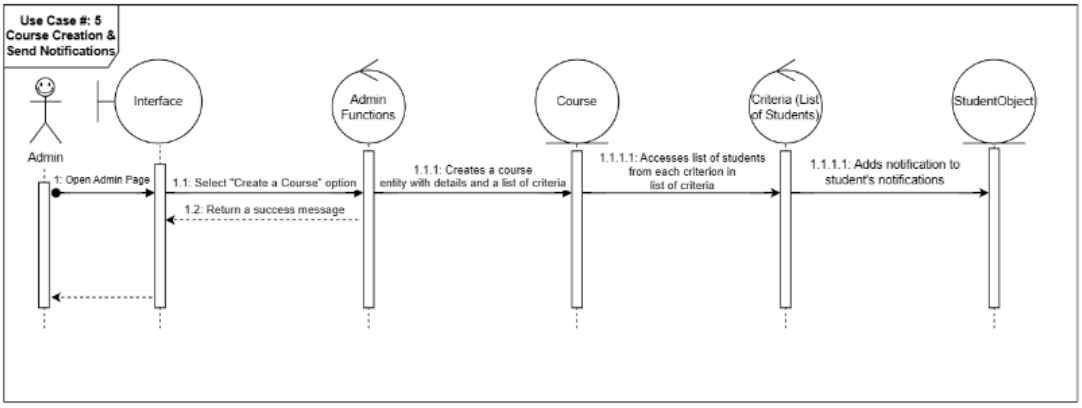
Use Case 5	Class Roster
Actors	CSULB Faculty (IT and Professors)
Description	Allows faculty to access a class roster upon request. Professors may only be able to access classes they are teaching, IT can access any roster.
Pre-Conditions	The user must have a CSULB account associated with staff and must have necessary permissions as an IT or a professor. Higher level IT or the program itself may assign permissions to users.
Flow of Control	The user must search for and select the class that they want to see the roster of: 1. User enters their log-in information and navigates to the Student Center. 2. Go to class courses and search, then click on the appropriate class. 3. Click “see class roster” including students and professors.
Post-Conditions	Users will see the class roster that includes the students and professors, the user may also see the available seats to the class.
Error-Conditions	*Incorrect Information*: Changes to the class roster may not update in a timely manner, for example, if a student enrolls it may take >30 minutes for the change to be seen by users.
Nonfunctional Requirements	*Order*: Class roster should be in a set order, either by time of enrollment or by name (first or last name alphabetical).




Use Case 2: Alternative classes
Student
Interface
Schedule Suggest
Display Model
<<System>>
Open student page
Pass student
Course Requirements
loop
Waitlisted classes
User's existing class selections
Preferred class times
loop rundown
a User course requirements
b User existing class selection
c preferred class times
print suggested classes


Use Case 3: Class Info Display
Student/Admin
Interface
Course Checker
Get Infomation
Course Page
1 : Open home page
1.1 Select Course
returns
Alt
Online = True
Prompts
Displays
else


Use Case 4: Class Roster
ADMIN
Interface
Course
Student List
<<system>>
PRINTER
rundown
CHECKS students in class
Open admin page
Open a course
List of students in class
print
print students in class



5. Nonfunctional Requirements

5.1. Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

N/A

5.2. Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

N/A

5.3. Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

When it comes to security or privacy, the user identity authentication requirements are an email and a password. When it comes to a regulation that other developer companies must face is

the Network security policy. This is a set of standardized practices and procedures that outlines rules of network access - ensures the confidentiality of company data.

5.4. Business Rules

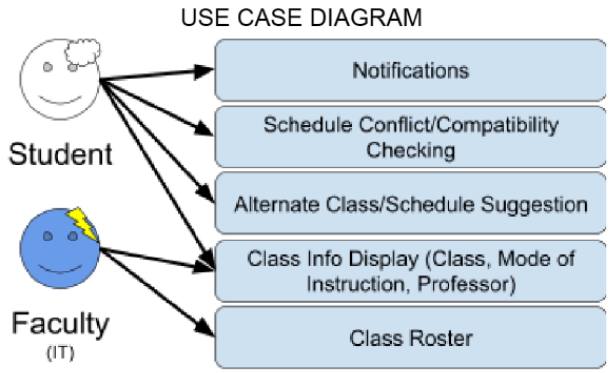
<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

There are 2 different roles when it comes to the implemented GUI system. That would be the admin and student role. When it comes to the student options, they cannot add a course to the schedule, forcefully remove a student, and make a new course. When it comes to the actions of the students, they are allowed to add/remove courses to themselves, and wishlist.

6. Preliminary Use Case Models and Diagrams

<This section presents a list of the fundamental diagrams and use cases that satisfy the system's requirements. The purpose is to provide an alternative, "structural" view of the requirements stated above and how they might be satisfied in the system. >

6.1. Use Case Diagram



Use Case 1	Notifications
Actors	Students of CSULB
Description	Allows the user to be notified of new sections of desired classes.
Pre-Conditions	The user must be waitlisted for another section or registered in another section of the same class.
Flow of Control	1. User on their mobile device or desktop receives a notification (the message is received on the Canvas inbox folder)
Post-Conditions	User is sent an email notifying the user of the new section.
Error-Conditions	1. *Technical error*: unable to send a notification to users device. 2. Technical errors displaying incorrect information.
Nonfunctional Requirements	1. Login to Canvas and notifications on. 2. Performance: the notification should be displayed as soon as availability of class (Fast notification time).

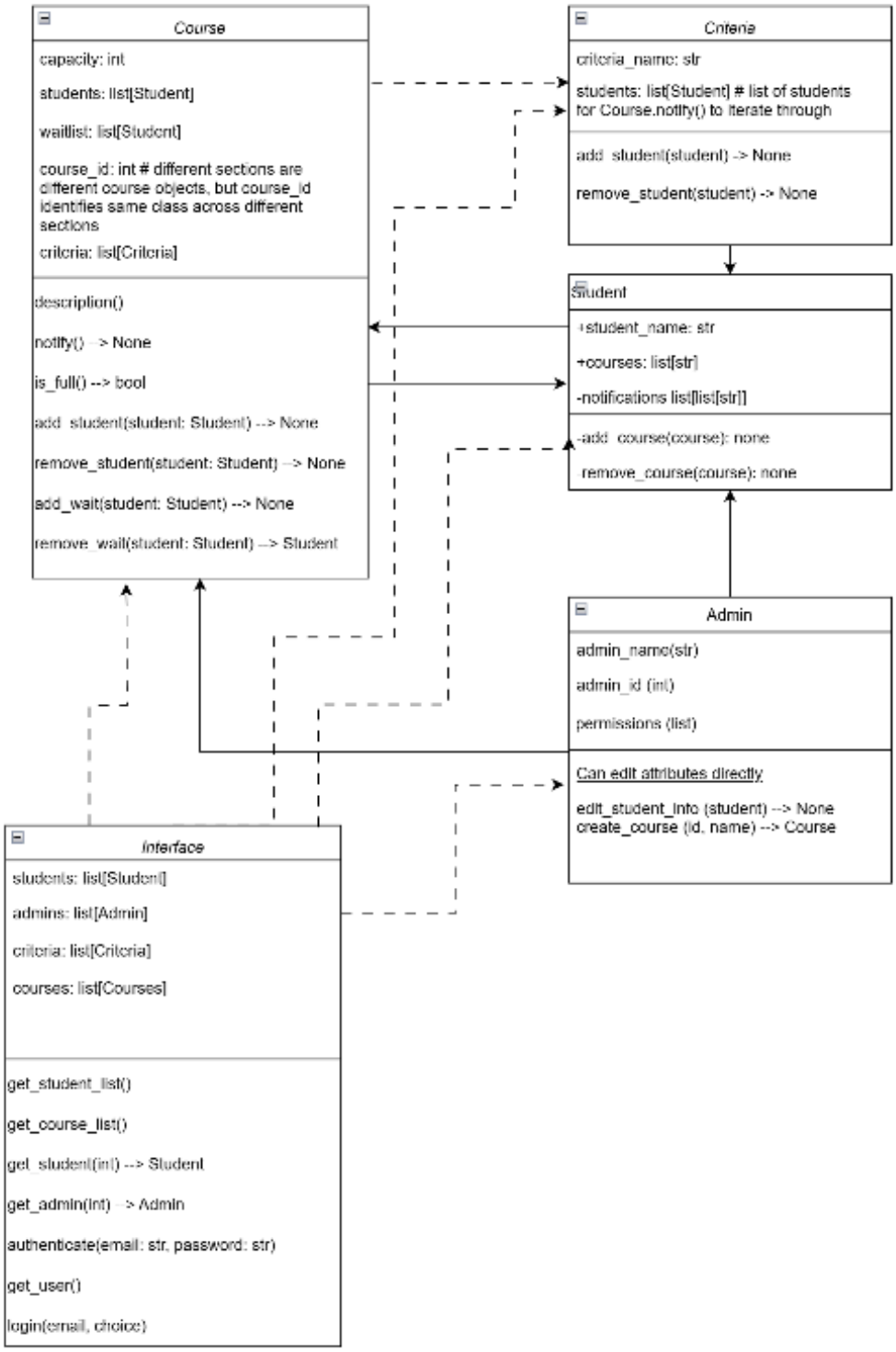
Use Case 2	Schedule Conflict/Compatibility Checking
Actors	Students of CSULB
Description	Informs the user whether a class they are about to sign up for will conflict with their schedule or their course requirements.
Pre-Conditions	The student must be logged into the Student Center and must navigate the schedule of classes.
Flow of Control	1. User navigates to the course selection page. 2. User selects a course. 3. If the course is not included in their major, minor, or general education requirements, inform the user by visually displaying a warning. 4. If the course conflicts with their existing schedule or the same class was already taken, display a warning before proceeding to Use Case 3 (Alternate Class/Schedule)
Post-Conditions	The student can see a notification whether the class they're selecting will conflict or not.
Error-Conditions	- User is not warned/informed in the case of lacking conflicts - User is warned/informed despite lacking any conflicts or identical courses
Nonfunctional Requirements	1. *Performance*: Calculated conflicts onto the schedule already made by the student. 2. *Usability*: The displayed information is clear and UI manageable when cross referencing to other viable classes.

Use Case 3	Alternative Classes / Schedules
Actors	Students of CSULB
Description	Provides users a suggestion of courses to take or alternate sections to build a better schedule.
Pre-Conditions	Students must be logged into the Student Center and must be looking at their schedule of classes.
Flow of Control	1. User navigates to the course selection page. 2. User either selects a course or views their current schedule 3. The page should provide alternative courses and alternate sections relevant to the user, based on the following: a. User's course requirements b. User's waitlisted classes c. User's existing class selections d. User's preferred class times
Post-Conditions	The user is provided multiple classes and sections or alternate schedules that are relevant and do not provide conflict
Error-Conditions	- Suggestion of irrelevant/unrelated courses to the user in course requirements or already selected courses - Failure to provide suggestions - Suggestion of a schedule that includes conflicting courses (identical classes or conflicting times)
Nonfunctional Requirements	1. *Usability*: UI clear and organized to the user. 2. *Performance*: Navigation and loading is fast when navigating the Student Center.

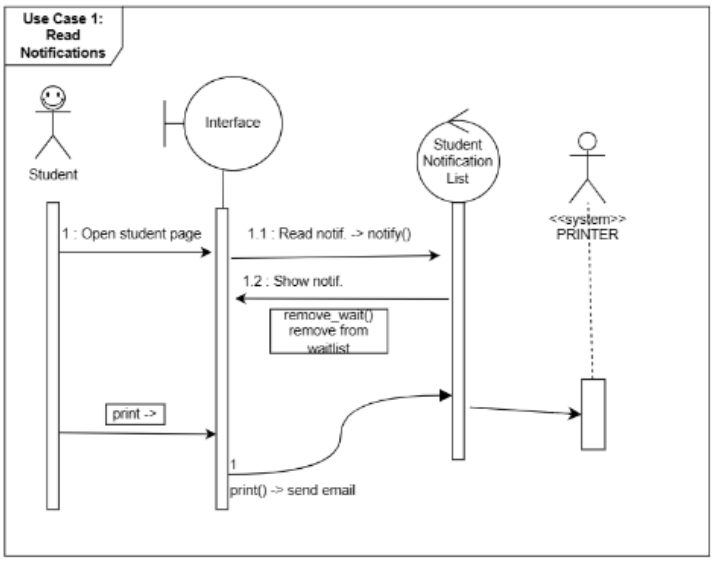
Use Case 4	Class Info Display (Class, Mode of Instruction, Professor)
Actors	Both the CSULB IT DIvision and the Students of CSULB
Description	Informs the user of general class information, such as the course title, department, mode of instruction, and professor.
Pre-Conditions	The user must select the course to view its information.
Flow of Control	1. User navigates to course selection page 2. User selects a course. 3. Displays all relevant information.
Post-Conditions	The user is able to read all relevant information pertaining to the course.
Error-Conditions	- Not being able to retrieve data from the database associated with it - Displaying the incorrect information (such as another class's information)
Nonfunctional Requirements	- *Structure*: Must be structured in a manner that is easy to read and phrased in an understandable manner. - *Relevancy*: Must be able to focus the user's attention on relevant information and avoid filling the page with "bloat" or irrelevant information

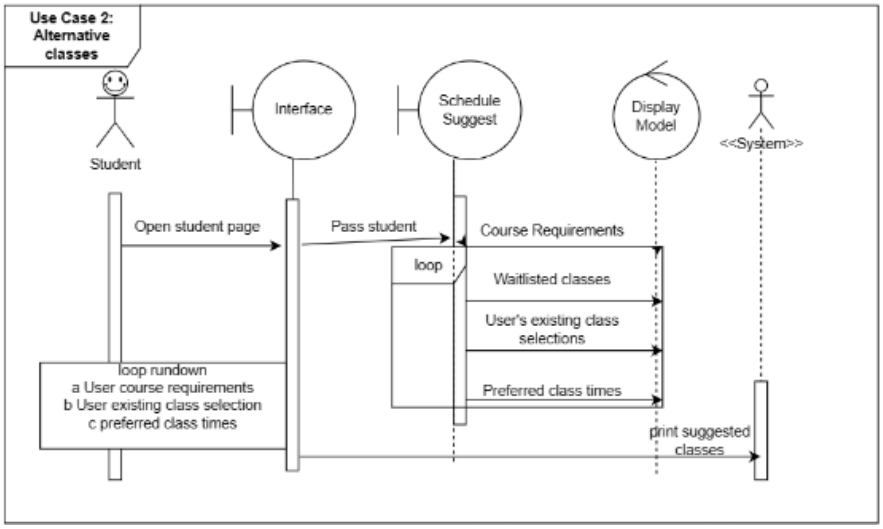
Use Case 5	Class Roster
Actors	CSULB Faculty (IT and Professors)
Description	Allows faculty to access a class roster upon request. Professors may only be able to access classes they are teaching, IT can access any roster.
Pre-Conditions	The user must have a CSULB account associated with staff and must have necessary permissions as an IT or a professor. Higher level IT or the program itself may assign permissions to users.
Flow of Control	The user must search for and select the class that they want to see the roster of: 1. User enters their log-in information and navigates to the Student Center. 2. Go to class courses and search, then click on the appropriate class. 3. Click “see class roster” including students and professors.
Post-Conditions	Users will see the class roster that includes the students and professors, the user may also see the available seats to the class.
Error-Conditions	*Incorrect Information*: Changes to the class roster may not update in a timely manner, for example, if a student enrolls it may take >30 minutes for the change to be seen by users.
Nonfunctional Requirements	*Orderly*: Class roster should be in a set order, either by time of enrollment or by name (first or last name alphabetical).

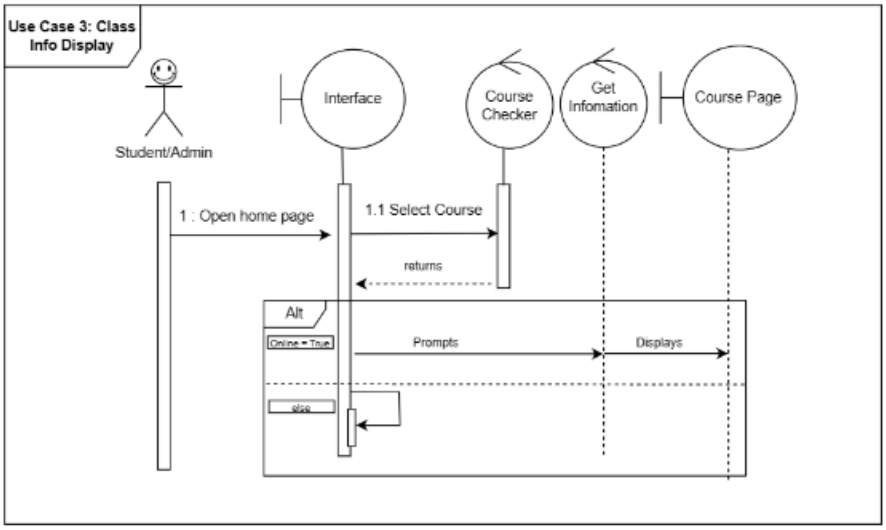
6.2. Class Diagram

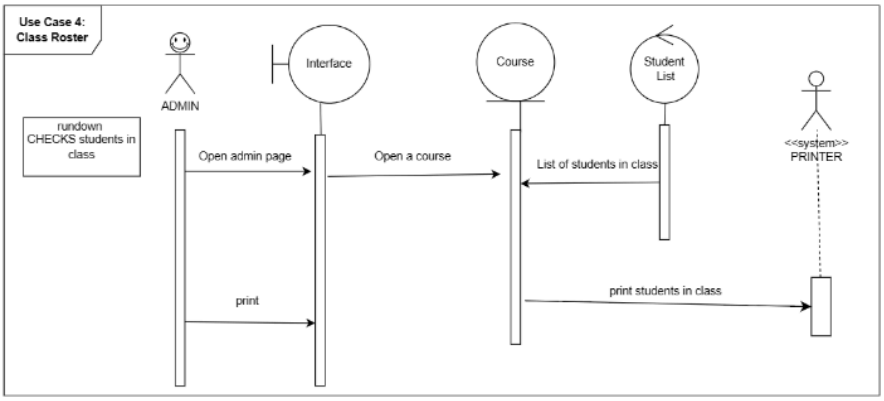


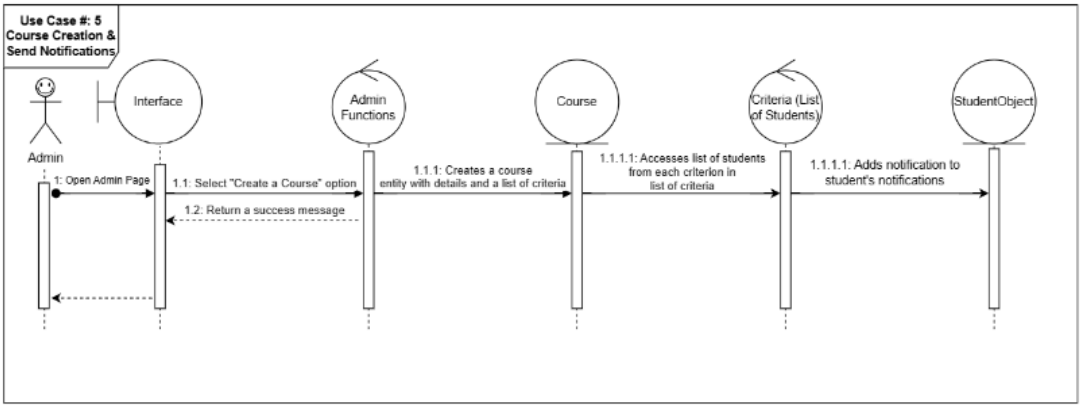
6.3. Sequence Diagrams


Use Case 1: Read Notifications
Student
Interface
Student Notification List
<<system>> PRINTER
1 : Open student page
1.1 : Read notif. -> notify()
1.2 : Show notif.
remove_wait() remove from waitlist
print ->
1
print() -> send email


Use Case 2: Alternative classes
Student
Interface
Schedule Suggest
Display Model
<<System>>
Open student page
Pass student
Course Requirements
loop
Waitlisted classes
User's existing class selections
Preferred class times
loop rundown
a User course requirements
b User existing class selection
c preferred class times
print suggested classes


Use Case 3: Class Info Display
Student/Admin
Interface
Course Checker
Get Infomation
Course Page
1 : Open home page
1.1 Select Course
returns
Alt
Online = True
Prompts
Displays
else


Use Case 4:
Class Roster
ADMIN
Interface
Course
Student
List
<<system>>
PRINTER
rundown
CHECKS students in
class
Open admin page
Open a course
List of students in class
print
print students in class


Use Case #: 5
Course Creation &
Send Notifications
Admin
Interface
Admin
Functions
Course
Criteria (List
of Students)
StudentObject
1: Open Admin Page
1.1: Select "Create a Course" option
1.2: Return a success message
1.1.1: Creates a course
entity with details and a list of criteria
1.1.1.1: Accesses list of students
from each criterion in
list of criteria
1.1.1.1: Adds notification to
student's notifications