SW Engineering CSC648/848 Fall 2022

Project Name: New SFSU Student Center

Team Number: 05

Milestone 2

Date: 19 October 2022

Student Name	Roles	Email
Zhenyu Lin	Team Lead, GitHub Master	zlin4@mail.sfsu.edu
Christopher Alan	Backend Lead	cyee12@mail.sfsu.edu
Michael Harrison	Frontend Lead	mchang9@mail.sfsu.edu
Elisa Hsiao-Rou	Team Member	echih@mail.sfsu.edu
Steven Paul Fong	Team Member	sfong10@mail.sfsu.edu
Cameron	Team Member	cyee10@mail.sfsu.edu

Milestone/Version	Date
M1V1	09/21/2022
M1V2	10/5/2022
M2V1	10/19/2022

Contents

Data Definitions

1. Student - Our end users. Can enroll in course sections.
    1.1. Date Enrolled
    1.2. Month - Month in which they started enrollment
    1.3. Day - Day in which they started enrollment
    1.4. Year - Year in which they started enrollment
2. Transfer Student
    2.1. Current Year
    2.2. Transfer information - A student that came from another college. Has a special student record called ‘past transcripts’ for courses taken at other colleges.
        2.2.1. College Transferred From
            2.2.1.1. Name of previous college
            2.2.1.2. Dates they were enrolled at previous college
                2.2.1.2.1. Month - Month in which they started enrollment
                2.2.1.2.2. Day - Day in which they started enrollment
                2.2.1.2.3. Year - Year in which they started enrollment
    2.3. Dates Enrolled
        2.3.1. Month - Month in which they started enrollment at current university
        2.3.2. Day - Day in which they started enrollment at current university
        2.3.3. Year - Year in which they started enrollment at current university
3. Undergraduate Student - A student that has not yet graduated with a degree.
    3.1. Dates Enrolled - When the student started enrollment
        3.1.1. Month - Month in which they started enrollment at current university

3.1.2. Day - Day in which they started enrollment at current university

3.1.3. Year - Year in which they started enrollment at current university

4. Graduate Student - A student that has graduated with a degree.

4.1. Previous University

4.1.1. Name - Name of previous university

4.2. Dates Enrolled

4.2.1. Month - Month in which they started enrollment at current university

4.2.2. Day - Day in which they started enrollment at current university

4.2.3. Year - Year in which they started enrollment at current university

4.3. Graduate Program

4.3.1. Name of graduate program

5. Full-Time Student - A student that is taking enough units that would be equivalent to a full-time job.

5.1. Credits - Amount of credits being taken

6. Part-Time Student - A student that is taking less units than would be equivalent to a full-time job.

6.1. Credits - Amount of credits being taken

7. Personal Information

7.1. Name

7.1.1. First Name

7.1.2. Middle name

7.1.3. Last name

7.2. Address

7.2.1. City

7.2.2. State

7.2.3. Country

7.2.4. Zip Code

7.3. Age

7.3.1. DOB

7.3.1.1. Month

7.3.1.2. Year

7.3.1.3. Day

8. Course (sometimes referred to as “Class”) - The main product for the student.

8.1. Course Information

8.1.1. Times - Lists the duration of class length

8.1.1.1. Start time - When class begins

8.1.1.2. End time - When class ends

8.1.2. Start Date - Month and day when class begins

8.1.3. End Date - Month and day when class ends

8.1.4. Instructor - Lists name of instructor

8.1.4.1. Last name

8.1.4.2. First name

9. Credit (sometimes referred to as “Unit”) - A unit of measurement for the time/effort of a course. A student needs a certain amount to graduate.

10. Course Section (sometimes referred to as “Section”)

11. Online Section

11.1. Times - Lists the duration of class length

11.1.1. Start time - When class begins

11.1.2. End time - When class ends

11.2. Start Date - Month and day when class begins

11.3. End Date - Month and day when class ends

11.4. Instructor - Lists name of instructor

11.4.1. Last name

11.4.2. First name

12. Synchronous Section

12.1. Times - Lists the duration of class length

12.1.1. Start time - When class begins

12.1.2. End time - When class ends

12.2. Start Date - Month and day when class begins

12.3. End Date - Month and day when class ends

12.4. Instructor - Lists name of instructor

12.4.1. Last name

12.4.2. First name

13. Asynchronous Section

13.1. Times - Lists the duration of class length

13.1.1. Start time - When class begins

13.1.2. End time - When class ends

13.2. Start Date - Month and day when class begins

13.3. End Date - Month and day when class ends

13.4. Instructor - Lists name of instructor

13.4.1. Last name

13.4.2. First name

14. Hybrid Section

14.1. Times - Lists the duration of class length

14.1.1. Start time - When class begins

14.1.2. End time - When class ends

14.2. Start Date - Month and day when class begins

14.3. End Date - Month and day when class ends

14.4. Instructor - Lists name of instructor

14.4.1. Last name

14.4.2. First name

15. Subject

15.1. Major - The

16. Time Slot (sometimes referred to as “Date & Time”) - An attribute of course sections that details when it takes place.

16.1. Building

16.2. Room Number

17. Time Slot (sometimes referred to as “Date & Time”)

17.1. Start Time

17.2. End Time

17.3. Duration

18. Component - An attribute of a course section describing the kinds of learning activity the student would be doing. (e.g. Lecture, Lab, Self-Study, etc.)
    18.1. Lecture - Class where a professors convey information to the student
    18.2. Lab - Interactive setting where students have hands-on experience with the course material through experiments.
    18.3. Self-Study - Self regulated study, with oversight by professors to determine if a student fulfilled credit requirements.
19. Course Materials - An attribute of a course section describing the kinds of materials the student may need to purchase to take the class (e.g. textbooks)
    19.1. Textbook
20. Major - An attribute of a student. Can determine what types of classes the student can take.
21. Student Record - A type of form that also acts as an attribute of students. Having certain completed records affects how the student can interact with the system.
22. Transcripts - A type of student record. A list of previously enrolled courses, and the grades received from them.
    22.1. Currently enrolled courses- Courses in which the student is currently taking
    22.2. Previously enrolled courses - Courses in which the student has already taken
    22.3. Grades - The course evaluation give to the student by the professor
    22.4. Semester - The time period in which the class was taken by the student
23. Student Calendar - A list of dates & time slots.

23.1. Important dates

23.1.1. Final day to withdraw without a W - Lists the final day a student may drop a class without receiving a W grade.

23.1.2. Final day to drop and receive a refund - Allows a student to drop a class and receive a refund for said credits.

23.1.3. Final day to change to CR/NC - Lists the date a student must change their grading option by

23.1.4. Payment due dates - Lists the day which all payments for tuition are due

23.1.5. Final day to add classes - Lists the final day in which a student can add a class without requiring a permission number

23.1.6. Final day to add a class via permission number - Lists the final day in which a student can add a class from a professor given permission number. After this day the student may no longer enroll in the class

23.1.7. Final day for faculty drops - Lists the final day that faculty may drop students enrolled in their course.

23.1.8. Final day to withdraw from classes or university - Lists the final day in which a student may exist from the university and receive a refund.

23.1.9. Holidays - Lists days in which there will be no courses taught, in which they normally would be

24. Class Schedule - A type of Student Calendar containing only the dates & times of the student's course sections.

24.1. Name of class - The descriptive name of the course

24.2. Course ID - The specific alphanumeric value of the course.

24.3. Section number - Lists the more specific class the student will be in for classes which have multiple time slots.

24.4. Time slot - Lists when the class will run from

24.5. Professor - Lists the instructor teaching the course

25. Appointments - Has a date & time, and acts as an element in a calendar, just like course sections.

26. Prerequisite - A type, of course, that is also an attribute for another different course. The prerequisite must be completed concurrently with, or before the student takes the associated other courses.'

26.1. Courses the prerequisite unlocks

26.1.1. Name of the course the prereq unlocks

26.1.1.1. Name of course

26.1.1.2. Course ID

27. Professor - The teacher of a section. Can modify course section information.

27.1. Courses offered - Lists all courses that the specified teacher will be teaching.

27.2. First name

27.3. Last name

28. Advisor - An employee of the university that specializes in helping students with school-related questions.

28.1. Name of advisor

28.2. Time slots - Times in which the advisor is available

29. College (sometimes referred to as “University”) - The client organization. Uses the system to offer courses to students.
30. Department - A subsection of a college that facilitates learning for specific subject(s).
31. Campus Clinic - Site that students can visit for health-related issues.
    - 31.1. Hours - Lists the times that the clinic is open
    - 31.2. Location - Lists where you can find the Campus Clinic
32. Student Club - A school-recognized student organization. Associated with a faculty advisor.
    - 32.1. Faculty advisor Lists the faculty advisor of the student club
    - 32.2. Name - Lists the name of the specific club
33. Parking Permit - A form that grants students access to parking.
    - 33.1. Location - Lists the locations the permit is valid in.
34. Bill - A form that shows students' transactions after purchase.
    - 34.1. Amount paid
    - 34.2. Amount remaining - Tells the user how much of their bill they have yet to pay
35. Charge - An element on a bill. A specific amount the student owes for a specific purpose.
    - 35.1. Name of charge - Tells the user what type of charge is being levied on their account.
    - 35.2. Value of charge - Tells the user how much they must pay
36. Aid Form - A form that shows students’ financial aid information.

36.1. Loan amount granted - Tells the user the amount of money they may receive from a loan in dollars

36.2. Loan amount accepted - Tells the user how much of the granted loan they have accepted in dollars

36.3. Grant amount - The amount quantity in dollars of a federal pell grant given to a student

36.4. Grant amount accepted - The amount of a federal pell grant the student has accepted in dollars

37. Map - An image showing the geographical layout of a location.

37.1. Format – JPG

38. Schedule - an itinerary that you follow throughout the day.

39. Notification - an alert that notifies the student of an important message.

40. Holds - A property that prevents students from enrolling until resolved.

40.1. Name of hold - Tells the student the name of a hold so they can identify the type of hold they have

40.2. Date

40.2.1. Date created - Tells the student when their hold began

40.2.2. Due date - Tells the student when they must clear the hold by without incurring any penalty

41. Search Parameter - An attribute or specific value of an attribute that is used to filter/narrow down elements from a full list of elements.

41.1. Session

41.1.1. Semester - Lists the semester which class is offered E.g Summer, Fall, Spring

41.1.2. Year - Year of the course which is being offered

41.2. Subject - Specifies and searches for only courses of specified major E.g Statistics

41.3. Course number - The numerical ID attribute of the course

41.3.1. Contains - Returns courses in which the value searched appears in the course number.

41.3.2. Greater than or equal to - Returns courses with course number greater than or equal to the search parameters

41.3.3. Less than or equal to - Returns courses with course number less than or equal to the search parameters

41.3.4. Is exactly - Only searched the courses with the exact string parameter.

41.4. Additional criteria

41.4.1. Start time - Specifies when class begins.

41.4.1.1. Between - Lists courses beginning between the times specified

41.4.1.2. Greater than - Lists courses with start times after the times specified, but not going further than midnight of the next day

41.4.1.3. Greater than or equal to - Inclusively lists courses with start times of the times specified, but not going further than midnight of the next day

41.4.1.4. Is exactly - Lists courses only which begin exactly when specified.

41.4.1.5. Less than - Lists courses beginning before the times specified, but not going further than midnight of the current day

41.4.1.6. Less than or equal to - Inclusively lists courses that begin before the times specified, beginning at midnight of the current day.

41.4.2. End time - Specifies when class ends.

41.4.2.1. Between - Lists courses ending between the times specified.

41.4.2.2. Greater than - Lists courses with end times after the times specified, but not going further than midnight of the next day

41.4.2.3. Greater than or equal to - Inclusively lists courses with end times of the times specified, but not going further than midnight of the next day

41.4.2.4. Is exactly - Lists courses only which end exactly when specified.

41.4.2.5. Less than - Lists courses ending before the times specified, but not going further than midnight of the current day

41.4.2.6. Less than or equal to - Inclusively lists courses that end before the times specified, beginning at midnight of the current day.

41.4.3. Days of week

41.4.3.1. Exclude any of these days - Does not search for courses that only fall on the precisely specified days.

41.4.3.2. Exclude only these days - Does not search for courses that have a session on each of the precisely specified days.

41.4.3.3. Include any of these days - Search for courses that have a session on each of the precisely specified days.

41.4.3.4. Include only these days - Only search classes that have a session on the specified day.

41.4.4. Instructor name

41.4.4.1. Begins with - Searches instructors whose names begin with the string specified

41.4.4.2. Contains - Searches if a string parameter is contained within the instructor's name.

41.4.4.3. Is exactly - Searches for a professor whose last name is exactly the same as the search parameter.

41.4.5. Course attribute

41.4.5.1. GE - General education, needed by the state to graduate.

41.4.5.2. Upper division - Courses in which a student must have a graduate standing to take.

41.4.5.3. Lower division - Courses in which a student must have a undergraduate or higher standing to take

41.4.5.4. Graduation requirement - Courses in which are necessary to graduate, not necessarily a GE class.

41.4.6. Course keyword - Searches for a common keyword, E.g Computer science

41.4.7. Maximum units

41.4.7.1. Greater than - Lists courses with credits greater than the amount of course credits specified.

41.4.7.2. Greater than or equal to - Inclusively lists courses with credits greater than the amount of course credits specified.

41.4.7.3. Is exactly - Lists courses which are exactly the amount of course credits specified.

41.4.7.4. Less than - Lists courses with credits less than the amount of course credits specified.

41.4.7.5. Less than or equal to - Inclusively lists courses with credits less than the amount of course credits specified.

41.4.8. Minimum units

41.4.8.1. Greater than - Lists courses with credits greater than the amount of course credits specified.

41.4.8.2. Greater than or equal to - Inclusively lists courses with credits greater than the amount of course credits specified.

41.4.8.3. Is exactly - Lists courses which are exactly the amount of course credits specified.

41.4.8.4. Less than - Lists courses with credits less than the amount of course credits specified.

41.4.8.5. Less than or equal to - Inclusively lists courses with credits less than the amount of course credits specified.

41.4.9. Location

41.4.9.1. On campus - Courses provided on the main campus

41.4.9.2. Off campus - Courses not provided on the main campus. May be provided at a satellite location or an Annex.

41.4.9.3. Country - Courses provided in a country foregin from the main university’s country.

42. Course Requirements - a course that is required in order to take the next course.
    42.1. Prerequisite - A course that must first be taken before attempting to take another course
43. Health Records - records that inform the school/doctor about one person's health information.
    43.1. Name
        43.1.1. First Name
        43.1.2. Middle name
        43.1.3. Last name
    43.2. Address
        43.2.1. City
        43.2.2. State
        43.2.3. Country
        43.2.4. Zip Code
    43.3. Age
        43.3.1. DOB
        43.3.2. Month
        43.3.3. Year
        43.3.4. Day
    43.4. Allergies - Lists any potential allergies the student has to prevent an allergic reaction.
    43.5. Medical conditions - Lists any medical conditions the student has.

Prioritized Functional Requirements

Priority 1:

1. Student

1.1. Students shall log in before accessing the system.
1.2. Students shall be able to enroll in course sections.
1.3. Students shall not be able to enroll in a class that would cause the student to exceed the set unit limit.
1.4. Students shall not fully enroll in more than one section of the same class.
1.5. Students shall be automatically dropped from waitlists of course sections that are of the same class as one that the student is fully enrolled in.
1.6. Students shall be automatically dropped from waitlists of course sections that cause time conflicts with a course section that the student is fully enrolled in.
1.7. Students shall be notified when they are dropped from a course section.
1.8. Students shall be able to search for courses.
1.9. Students shall be able to add courses to a shopping cart, prior to enrolling.
1.10. Students shall have transcripts.
1.11. Students shall have a class schedule.
1.12. Students shall not fully enroll in multiple sections that overlap on the same date & time slot.
1.13. Students shall have a student calendar, showing the student's class schedule and the college's academic calendar.
1.14. Students shall be able to drop course sections.
1.15. Students shall receive a Hold/Alert if they have overdue charges.
1.16. Students shall be notified whenever new Holds/Alerts are created on their account.
1.17. Students shall be dropped from courses if they have overdue charges after the set deadlines.
1.18. Students shall be able to access their student records (including transcripts and payment receipts).
1.19. Students shall be able to access their payment histories.
1.20. Students shall be able to request to change their major. (changed to a request, cause the school and departments have to accept the request)
1.21. Students shall receive a grade, upon completing a course.
1.22. Students shall enroll in courses with one of two grading options: CR/NC or Letter Grade.
1.23. Students shall be able to switch between grading options within certain date & time slots.
1.24. Students shall be able to view their financial aid.
1.25. Students shall be able to receive Financial Aid.
1.26. Students shall be able to leave feedback reviews for professors of course sections that the student has taken before.
1.27. Students shall be able to contact the department of their major.
1.28. Students shall be able to search for clubs at the university.
1.29. Students shall be able to upload their health records.
1.30. Students shall be notified of payment due dates.
1.31. Students shall be able to schedule advising appointments.

1.32. Students shall be able to schedule financial aid appointments.
1.33. Students shall be able to make an appointment with the university clinic.
1.34. Students shall be notified of upcoming appointments.

2. Course
   2.1. Course sections shall have a number of seats.
   2.2. Course sections shall have a waitlist.
   2.3. Course sections that are full shall place enrolling students on the waitlist.
   2.4. Courses shall tell the students which classes are required as prerequisites.
   2.5. Courses shall belong to one (1) subject.
   2.6. Courses shall require prerequisites.
   2.7. Course sections shall have time slots.
   2.8. Course sections shall have a location. (can be online)
   2.9. Course sections shall have a list of the average grade received by students in past semesters.
   2.10. Courses shall tell the student if the class is online, in person, hybrid, synchronous or asynchronous.
3. Waitlist
   3.1. Waitlisted students shall be notified when they are able to fully enroll in the section.
   3.2. Waitlisted students shall be automatically enrolled if space is available.
   3.3. Waitlisted students shall be notified if they are dropped from the waitlist.
4. Class Schedules
   4.1. Class schedules shall show a student's enrolled courses.
   4.2. Class schedules shall show a student's waitlisted courses.
   4.3. Class schedules shall show courses currently in the student's shopping cart.
5. Professor Reviews
   5.1. Professor reviews made by students shall be anonymous.
   5.2. Professor reviews made by students shall show the grade of the student publishing the grade.
   5.3. Professor reviews shall only be made by students who have completed a course section that the professor has taught.
   5.4. Professor reviews shall be displayed under a professor's profile, as well as within the attributes of any course section taught by that professor.
6. Transcripts
   6.1. Transcripts shall list all courses taken in the past.
7. Searchs
   7.1. Searches shall have parameters, which filter the displayed courses.
      7.1.1. Searches can be filtered by student's eligibility to enroll in the course.

7.1.2. Searches can be filtered by professor.
7.1.3. Searches can be filtered by location.
7.1.4. Searches can be filtered by date & time.
7.1.5. Searches can be filtered by attribute. (online, asynchronous, lab, lecture)
7.1.6. Searches can be filtered by course name.
7.1.7. Searches can be filtered by course number. (not CRN)
7.2. Searches shall display a list of courses.
7.3. Searched course sections shall display all their important data in the listing. (CRN, professor, location, date & time, units, name)
7.4. Searched course sections shall display on mouse-over, less important data in the search listing. (description, past grade averages, professor ranking, etc.)
7.5. Searched courses shall be add-able to the student's shopping cart.

Priority 2:

1. Student
   1.1. Students shall be dropped from a course if they cannot prove they have first taken the course's prerequisites, or are currently taking the course's prerequisites.
   1.2. Student calendars shall recommend alternative course sections in order to resolve date & time conflicts on the class schedule.
   1.3. Students shall be notified if a course section on their calendar has any of its attributes changed. (students should know if there's a change of professor or change of location)
   1.4. Students shall be notified when a change of major is fully processed, regardless of whether it is accepted or rejected.
   1.5. Students who've recently changed majors shall have access to resources for their new major.
   1.6. Students shall be able to swap one course for another.
   1.7. Students shall be able to pay for courses.
   1.8. Students shall be able to generate What-If Reports.
   1.9. Students shall be able to save their What-If Reports.
2. Transferring
   2.1. Transfer students can have a transfer credit report.
   2.2. Transfer students shall be able to search for other colleges' courses to check if they count as being transferred.
3. What-If Report
   3.1. What-If Reports shall show students the required classes for hypothetical change of major, degree, or other academic career choices.
4. Professor Reviews
   4.1. The system shall notify reviewers that their received grade will be displayed along with their anonymous review.
   4.2. Professor reviews shall have tags to help students parameterize searches when filtering for professors with certain teaching styles.

Priority 3:

1. Student

1.1. Students shall be able to purchase parking permits.

1.2. Students shall be notified if their permit purchase was approved.
1.3. Students shall be able to open Google Maps in the student center.

2. Payment
   2.1. PayPal shall be supported as a payment method.
3. Advising

   3.1. Advisors shall be able to schedule appointment time slots.

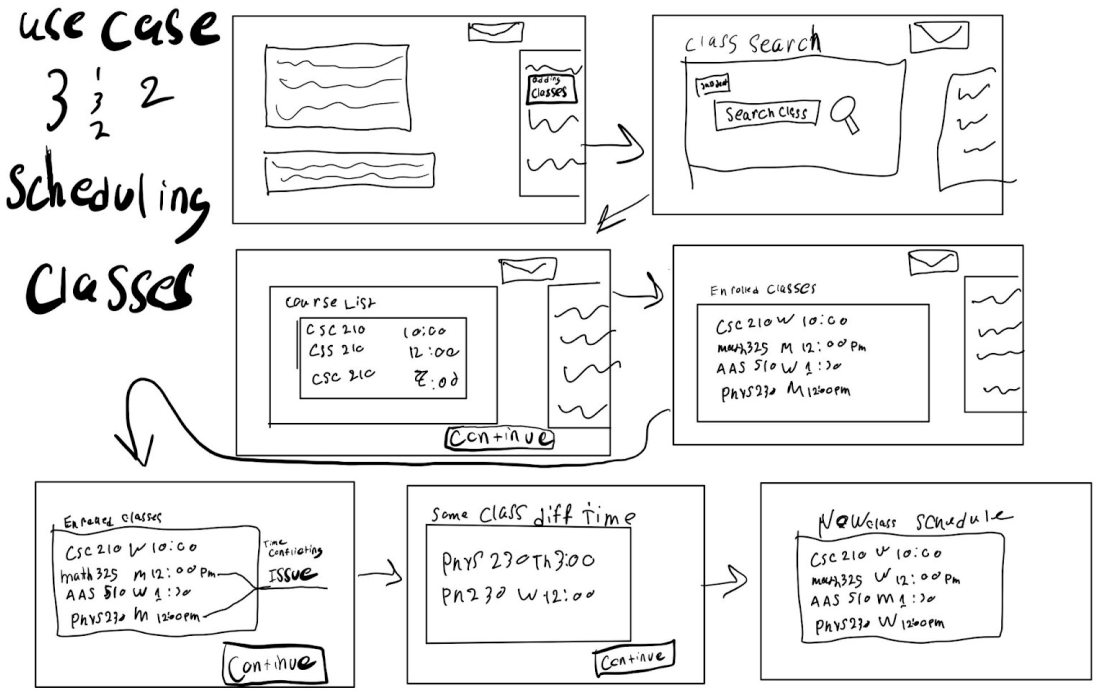
4. Clubs

   4.1. Clubs shall be displayed by the system.

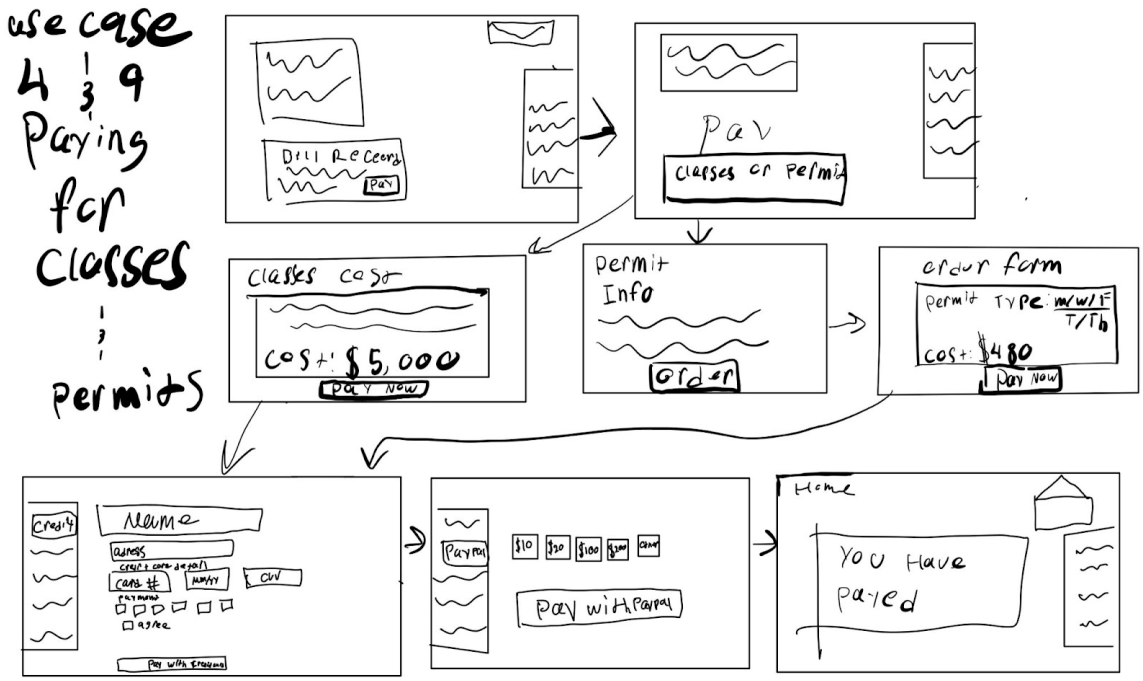
   4.2. Clubs can provide resources to the system for students to see.

UI Mockups and Storyboards (high level only)

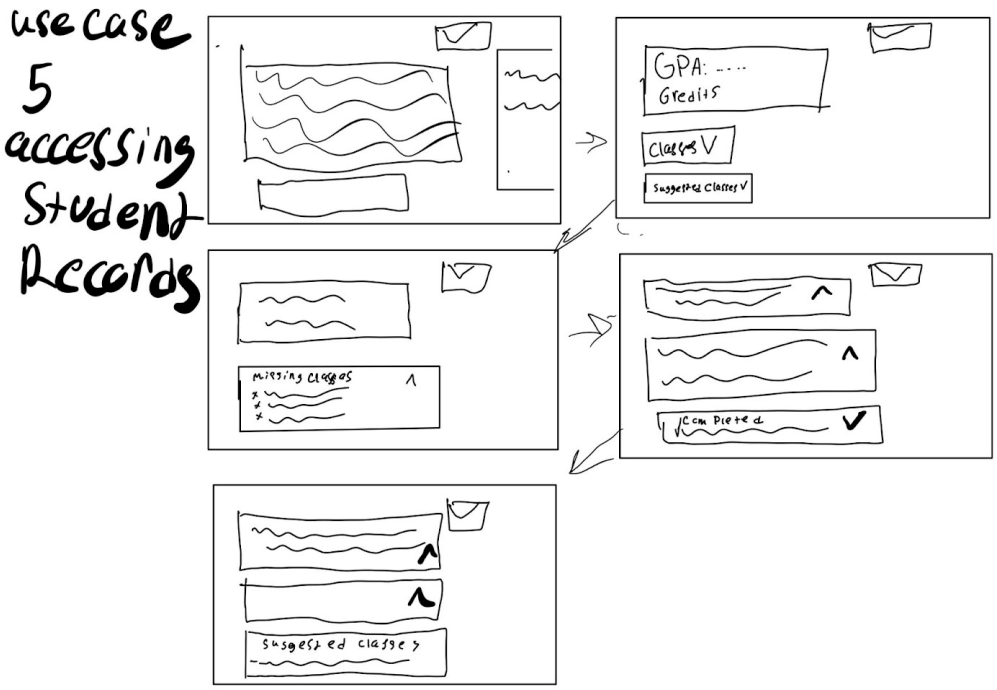
Use Case 2 and 3



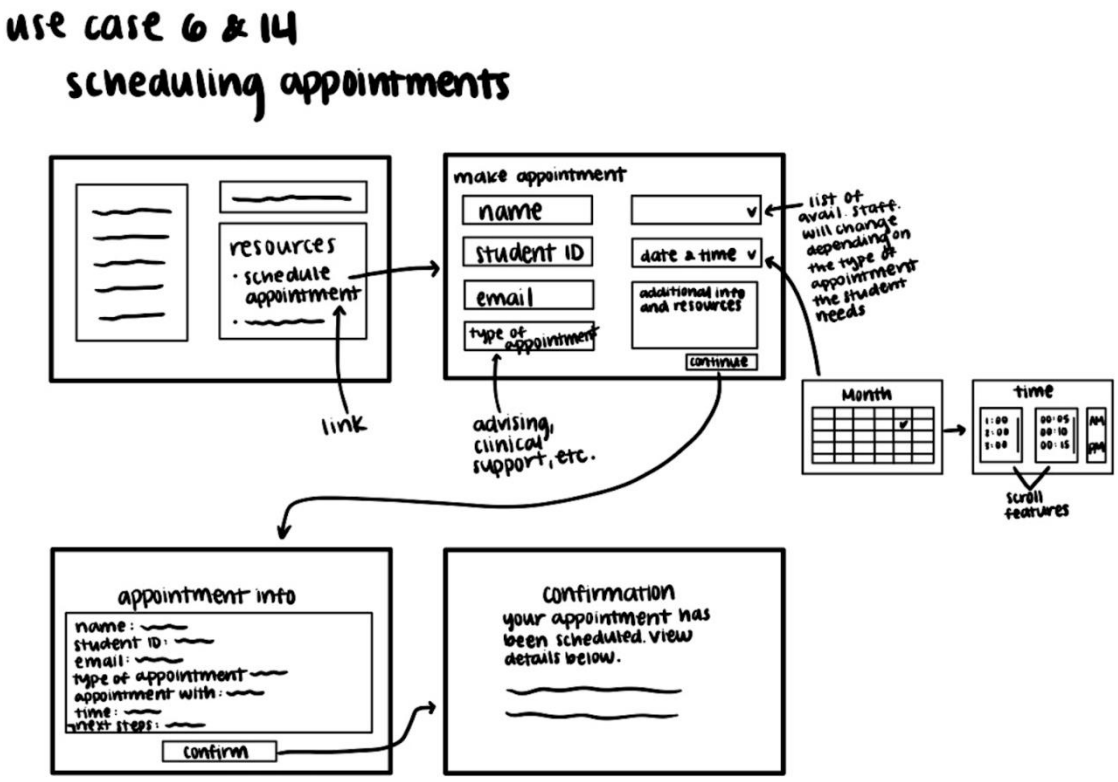
Use Case 4 and 9



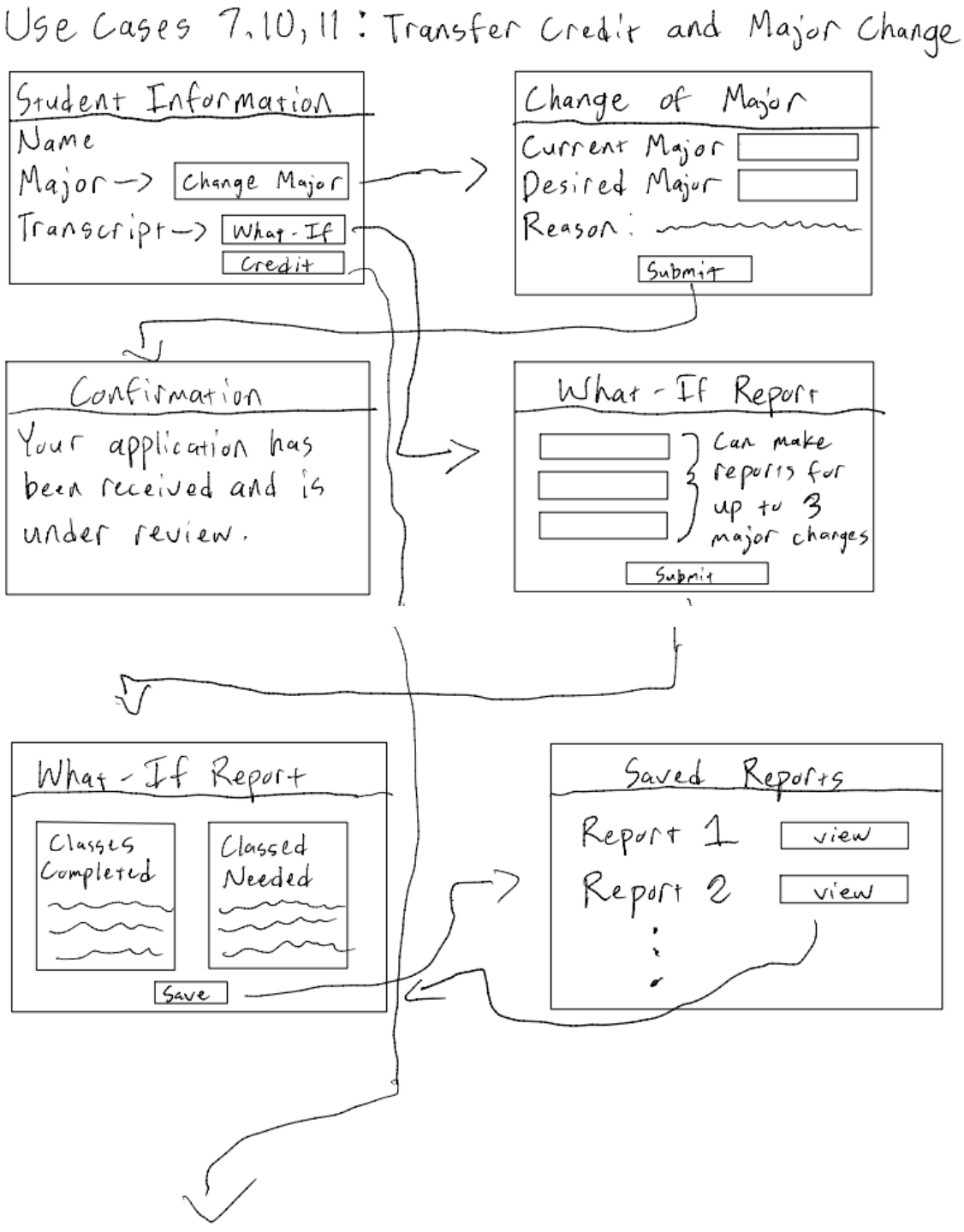
Use Case 5

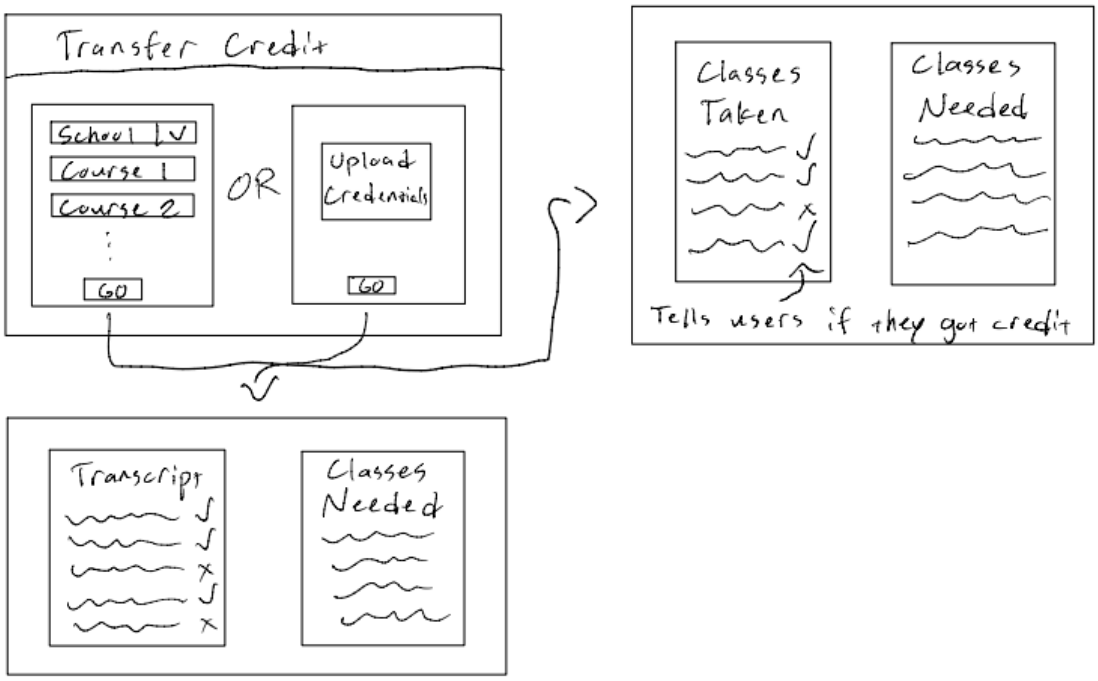

use case
5
accessing
Student
Records
GPA: ....
Credits
Classes V
Suggested classes V
missing classes
Completed
Suggested classes

Use Case 6 and 14

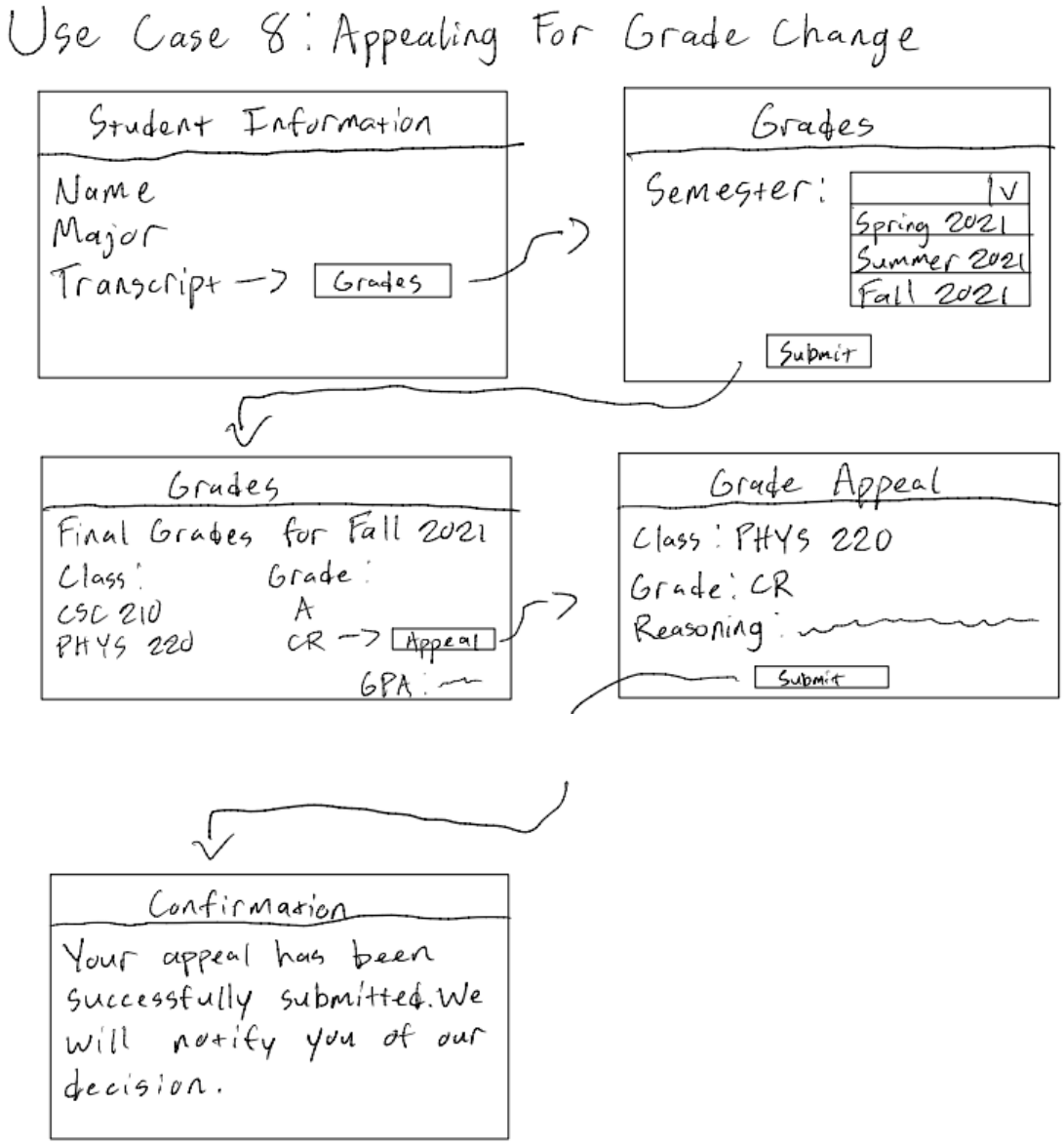


Use Case 7, 10 and 11

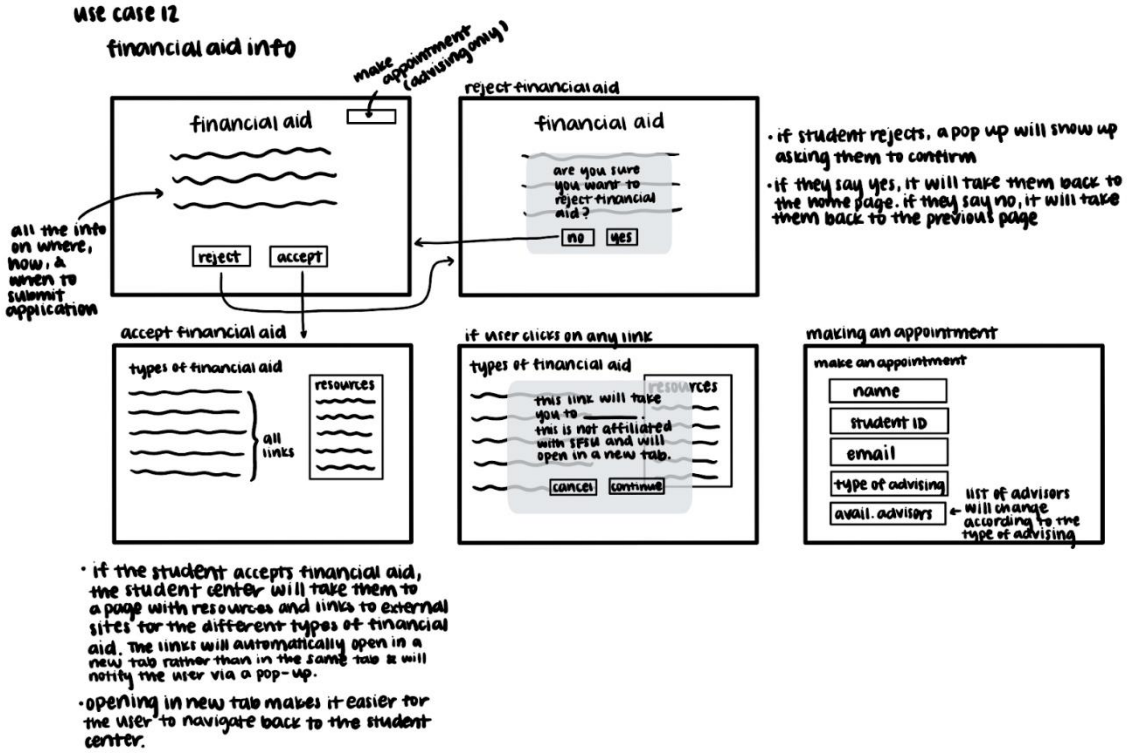



Transfer Credit
School
Course 1
Course 2
GO
OR
Upload Credentials
GO
Classes Taken
Classes Needed
Tells users if they got credit
Transcript
Classes Needed

Use Case 8


Use Case 8: Appealing For Grade Change
Student Information
Name
Major
Transcript -> Grades
Grades
Semester:
Spring 2021
Summer 2021
Fall 2021
Submit
Grades
Final Grades for Fall 2021
Class:
Grade:
CSC 210
A
PHYS 220
CR -> Appeal
GPA:
Grade Appeal
Class: PHYS 220
Grade: CR
Reasoning:
Submit
Confirmation
Your appeal has been successfully submitted. We will notify you of our decision.

Use Case 12

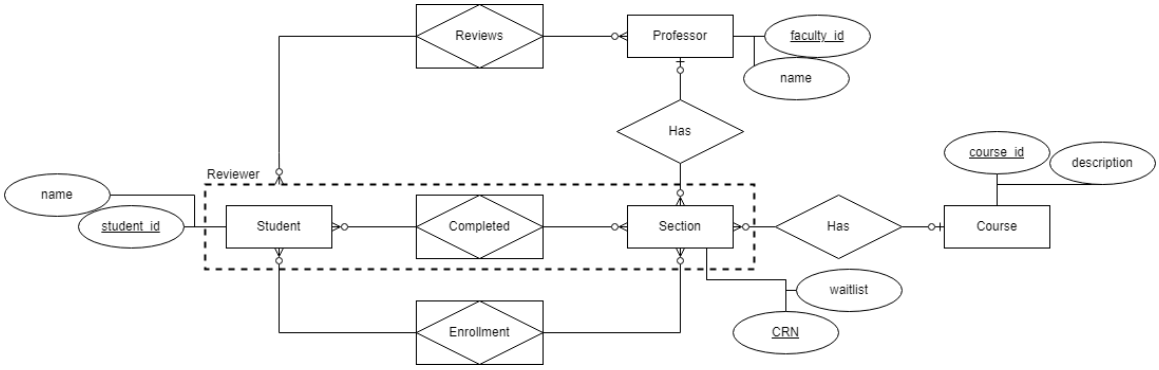

use case 12
financial aid info
make appointment (advising only)
financial aid
all the info on where, how, & when to submit application
reject
accept
reject financial aid
financial aid
are you sure you want to reject financial aid?
no
yes
• if student rejects, a pop up will show up asking them to confirm
• if they say yes, it will take them back to the home page. if they say no, it will take them back to the previous page
accept financial aid
types of financial aid
resources
all links
if user clicks on any link
types of financial aid
this link will take you to ____ this is not affiliated with SFSU and will open in a new tab.
cancel
continue
making an appointment
make an appointment
name
student ID
email
type of advising
avail. advisors
list of advisors will change according to the type of advising
• if the student accepts financial aid, the student center will take them to a page with resources and links to external sites for the different types of financial aid. The links will automatically open in a new tab rather than in the same tab & will notify the user via a pop-up.
• opening in new tab makes it easier for the user to navigate back to the student center.

High level database architecture and organization

Database Requirements:

1. Student
   1.1. A student can enroll in many course sections.
   1.2. A student can complete many course sections.
2. Course
   2.1. A course can have many course sections.
3. Course Section
   3.1. A course section belongs to one course.
   3.2. A course section can enroll many students.
   3.3. A course section is taught by one professor.
   3.4. A course section can have been completed by many reviewers.
4. Reviewer
   4.1. A reviewer is a student.
   4.2. A reviewer has completed many course sections.
   4.3. A reviewer can review many professors.
5. Professor
   5.1. A professor can teach many course sections.
   5.2. A professor can be reviewed by many reviewers.

ERD



Media storage Strategy:

Our system will store the media file in a file system. The database will store the location of media. One of the unique functionality of our project is professor ranking system. Students can rank their professor and give them feedbacks. Therefore, the most of storage content for our project will be text. Majority of our data will be store on the database. Although our first priority functional requirement has the majority of text data, our second and third functional requirements need to store large files such as student transcript pdf version and school club icons. The database will take huge amount of memory for backend to acquire the data from the database if we choose to encode the large file and store them into database .To effective use the memory and save our database bandwidth, we choose to store the large file like media file or pdf in a file system.

Search/filter architecture and implementation:

In our project, we will have one search bar that across all the pages. Therefore, the search function should be search across all of our database table. We will ask the user about the search category first. Then let them to type in the search content. The backend will base on the category and send the searching query to the database. For example, if the user want to search about the ranking for a specific professor, the user would first choose the searching category - ranking. Then type in the search content. The backend will first process the content by an algorithm. If the user type in the course number, the backend will send an query to the course table to get the professor id and check the professor ranking. Our search strategy is to send multiple search queries to different tables in the database. Then we will combine the search results and return to the front. We will ORM to manipulate data from the database. The benefit of ORM is that it can save lots of time in development. ORM avoids to write poorly-formed SQL and a lot of stuff is done automatically. Therefore, ORM significant reduce the development time.

High Level APIs and Main Algorithms

GET

/home

This API is used for querying the school's information. The front end sends the http request to the backend via this API. The backend will return the relevant school information.

/courses

This API is used for querying all of our course information. The front end sends the http request to the backend via this API. The backend will return the relevant course's information.

/university_calendar

This API is used for querying the university calendar. The front end sends the http request to the backend via this API, and the backend returns the university calendar info to the frontend.

/resource

This API is used for querying the university resources such as scholarship information. The front end sends the http request to the backend via this API, and the backend returns the resources to the frontend.

/professor

This API is used for querying a professor's profile page. The front end sends the http request to the backend via this API. The backend returns the professor's data, including ranking data, course sections the professor teaches, and any applicable reviews.

POST

/login

This API is used for the login function. The front end sends the user login info to this API by POST method. The backend will compare the given information with the relevant information stored in the database to see if the login is valid, then it will return the login status to the frontend to indicate whether the login was successful or not.

/register

This API is used for the registration function. The front end sends the registration info to this API by POST method. The backend will check if the information is all valid, then it will return the registration status to the frontend to indicate whether the registration was a success or not.

/rank

This API is used for the professor ranking system. The front end sends the user's id, login status, professor id, and the rank for the professor out of 5.0 that the user gave. The backend will update the professor's rank by adding this ranking to every rank the professor has received and taking the average. The backend will then return the operation status to the frontend to indicate whether the operation was a success or not.

/review

This API is used for the review function. The front end sends the user's review about the course and the professor to this API by POST method. The backend will insert the review into the database along with its tags, the date created, and the grade that the reviewer received for the

course, and then return the comment status to the frontend to indicate whether the comment operation was a success or not.

/course_operations

This API is used for adding, swapping, dropping, querying and changing the grading type of a course section. The front end sends the user identification, login status, course id, and operation type to the backend via this API, and the backend returns the corresponding operation status to the frontend.

/enrollment

This API is used for enrolling in courses. The front end sends the course id, user id, and login status to the backend via this API. The backend will return the enrollment status to the frontend to indicate whether the enrollment success or not.

/dashboard

This API is used for querying user information. The front end sends the user identification and login status to the backend via this API, and the backend returns the corresponding user info to the frontend

/shopping_cart

This API is used for querying user information. The front end sends the user identification and login status to the backend via this API, and the backend returns the corresponding shopping cart info to the frontend

/search

This API is used for the searching function. The front end sends the search category and the keyword to this API by the POST method. The backend then has to use database queries to obtain results of adequate similarity to the keyword. It will first decide which tables to search through depending on the chosen search category, then will use SQL and regex to search those tables for the specific elements.

/transcripts

This API is used for querying user information. The front end sends the user identification and login status to the backend via this API, and the backend returns the corresponding transcript info to the frontend.

/class_schedule

This API is used for querying the users' class schedules. The front end sends the user identification and login status to the backend via this API, and the backend returns the corresponding class schedule info to the frontend.

/notification

This API is used for querying the notification that the school sends to the users. The front end sends the user identification and login status to the backend via this API, and the backend returns the corresponding message to the frontend.

/checkout

This API is used for querying the total amount of tuition for the courses that the user has chosen. The frontend sends the user identification, login status, and course IDs to the backend via this API. The backend will add the student to the course, check if the enrollment caused any conflicts with other enrolled courses, create a tuition charge, and then return the corresponding price to the frontend.

/payment_history

This API is used for querying the user's payment history. The front end sends the user identification, login status, and the id of the course to the backend via this API, and the backend returns the payment history to the frontend.

/appointment

This API is used to schedule appointments. The frontend sends the user identification, login status and appointment info to the backend via this API, and the backend updates the user's calendar and returns the appointment schedule status to indicate whether it is scheduled success or not.

/major_operation

This API is used for apply, change, and querying the user's major. The front end sends the user identification, login status and operation type to the backend via this API, and the backend returns the corresponding operation status to the frontend.

/grade

This API is used to check the user's grades. The front end sends the user identification and login status to the backend via this API, and the backend returns the grades to the frontend.

/financial

This API is used to check the user's financial status. The front end sends the user identification and login status to the backend via this API, and the backend returns the financial status to the frontend.

/parking

This API is used to register the parking slot. The front end sends the user identification, login status, car information, time slot, and the user's requested parking position to the backend via this API, and the backend returns the registration status to the frontend to indicate whether the registration was a success or not.

/what_if

This API is used to generate What-If reports. The frontend sends the user identification, login status, and the hypothetical change of study program to the backend via this API. The backend will return a pdf report to the frontend.

/health

This API is used to keep track of the health status of the user. The frontend sends the user identification and login status. And the health info that's been entered by the user and clinical doctor to the backend via this API. The backend will return the operation status to the frontend to indicate whether the operation succeeded or not

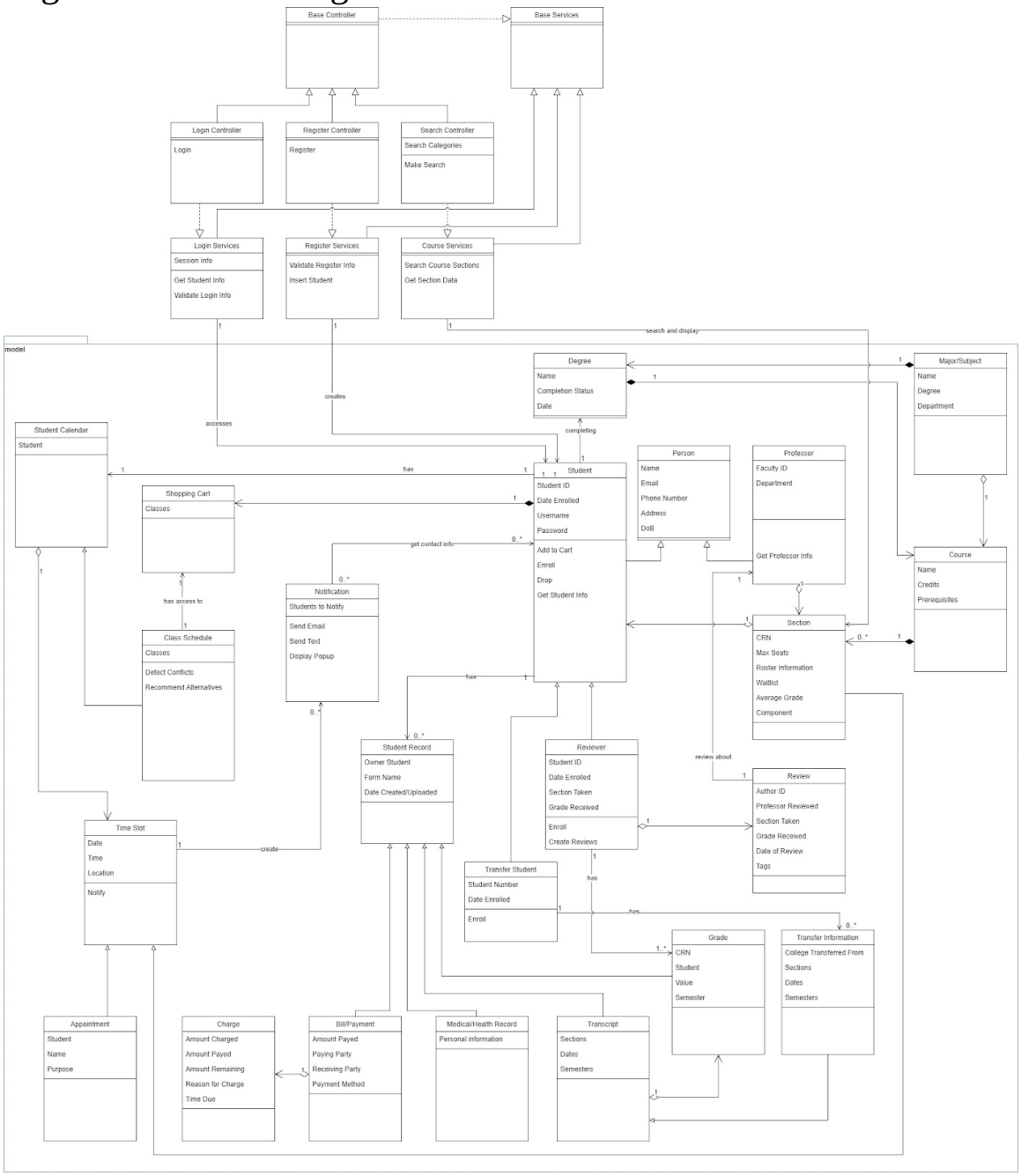
/transfer

This API is used to submit the user's transfer credit to the school. The frontend sends the user identification, login status, course number, and school code to the backend, the backend returns the credit transfer status to the frontend.

High-Level Algorithm

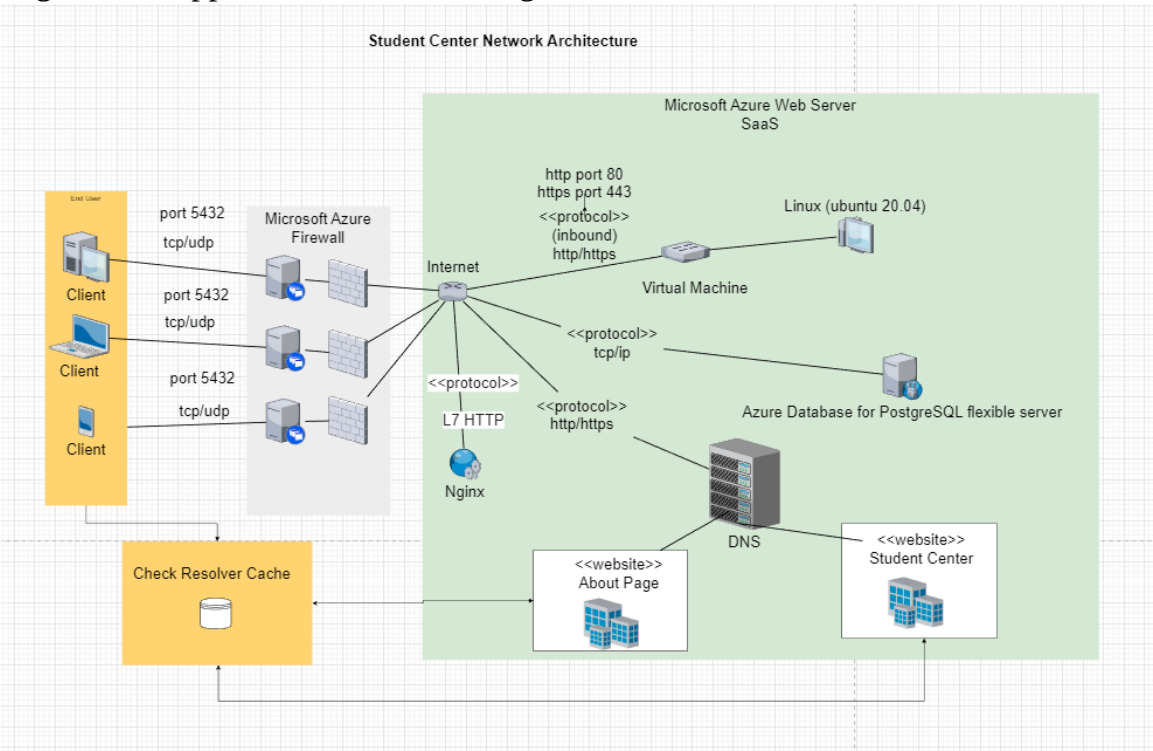
We design a professor ranking system based on students' feedback and the average grade of the courses. At the end of the semester, the students need to fill out a survey form about their class experience. The survey contains the following categories: Professor's explanation skills, Professor's class management, and Professor's teaching style and class feedback. The score of this will be posted on the professor's profile page and will be used to calculate the overscore of the professor. This ranking strategy can help students choose the most suitable professor for them. In addition, we also post-class feedback to help students succeed in class.

High Level UML Diagrams



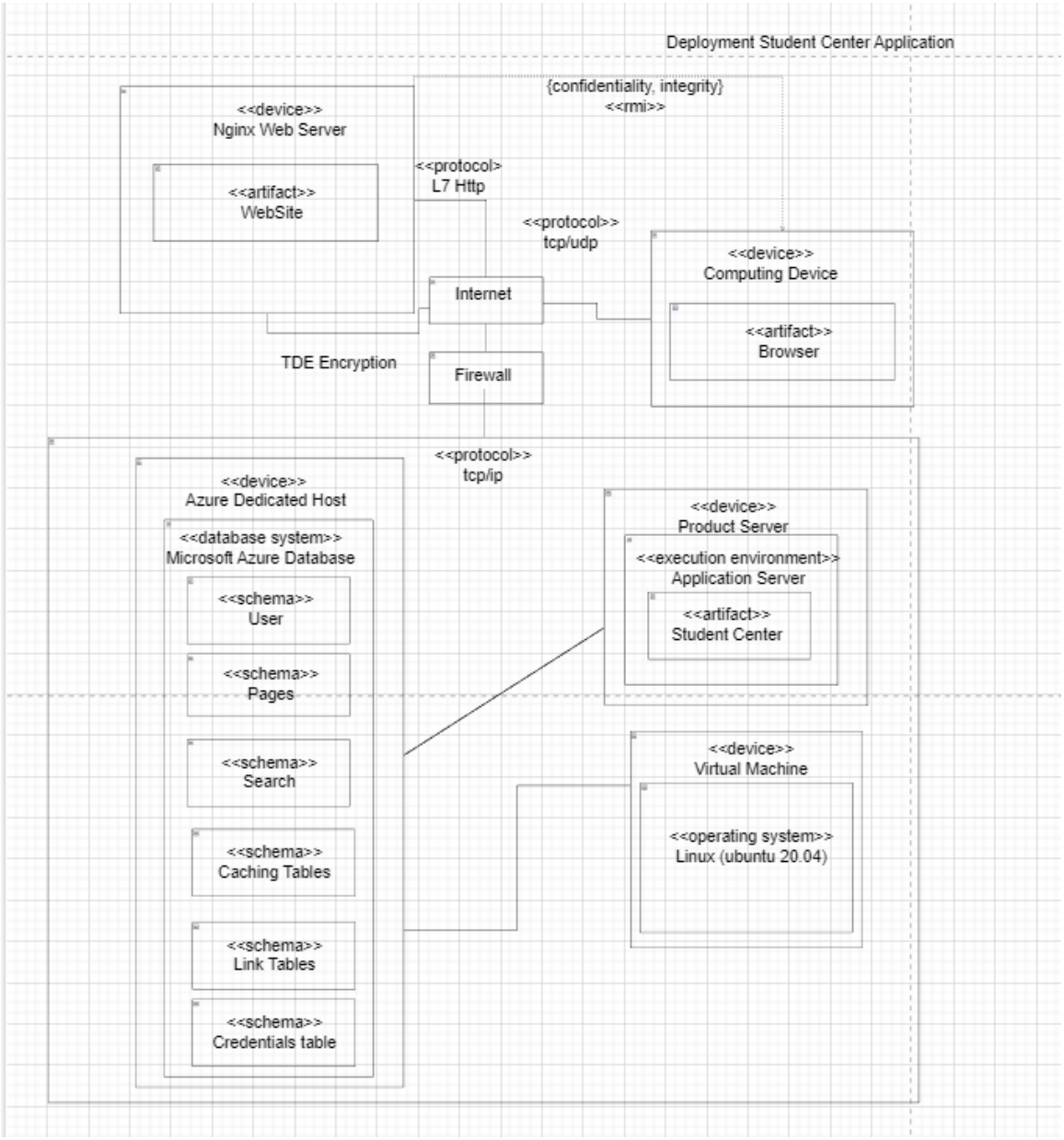
High Level Application Network and Deployment Diagrams

High Level Application Network Diagrams



Our student center network architecture lays out our secure way of connecting the client to our application. The client connects to our application via the internet through a tcp/udp connection. The Microsoft Azure Firewall ensures that only legitimate data is passed through, and ensures a secure connection. The user data requested is sent to our Azure server, and is processed. Depending on what type of information is required, a different protocol is used. To communicate with our VM, http/https is required, for our PostgreSQL server, tcp/ip is required and so on. After any request is made and accepted by the server, the relevant data is returned to the client. In cases where the relevant data is already loaded in our resolve cache, we can skip certain parts and go straight to the necessary webpage.

High Level Application Deployment Diagram



Our deployment diagram displays how our information will be connected in both its physical, and virtual components. One of the main concerns in our non-functional requirements is the guarantee that the client's information will be secure. We do so by using both a firewall, and TDE encryption for any data that is stored within our Azure server. Our Azure server will consist of our physical components, (servers, client computer) and the virtual components, (data tables, pages, search functions, operating systems, schemas and artifacts).

Identify actual key risks for your project at this time

- Time risks
    - Failure to reach deadlines can arise if we are to fall behind on our planned schedules. One deadline can snowball into another, meaning that if we get behind, it is likely we will stay behind for the remainder of the class. To prevent this from happening we will be using Trello and Discord to stay up to date with deadlines. By utilizing these two tools, we can communicate without the need to meet in person and be able to remotely remedy problems before class on Thursday.
    - Failure to arrive at planned meetings can cause a disruption with both the morale of the team and the flow of work. If even one member is behind, they will cause the whole team to fall behind. We will do our best to reach out and communicate to members who have not regularly attended our planned weekly meetings. If a member continually fails to attend meetings, we will inform the professor.
- Technical risks
    - Technical risks may arise with the availability of the services we are using. If a server is down, then our user will be unable to access our product. We will attempt to prevent this by only using established and reliable services, such as Microsoft Azure for our server.
- Teamwork risks
    - Our team may have issues when it comes to levels of commitment from each member. Levels of commitment may be seen as enough by one individual but not enough by another. We will remedy this by attempting to quantify levels of engagement and commitment. We will also inform the instructor of any individual that we do not believe are committed to the team.

- Legal/content risks
    - We are using data from schools in our model, therefore we might have legal problems by using their information without their explicit permission. We will remedy this issue by not using any personal student data that is protected under the Family Educational Rights and Privacy Act (FERPA) or Protection of Pupil Rights Amendment (PPRA).
    - We will be using the internet for much of our research. If we are to use a piece of code that is not ours and without the explicit approval of the writer, then our final product may be in jeopardy.
    - We will remedy this by either asking the original writer of the code for permission, or we will transform the code enough so it is detached from the original owner and is deemed as an original work of our own.
- Knowledge Risks
    - There is a differing level of understanding for fundamentals from each person on our team. If one member does not have the required information, then problems can arise in our implementation. We will attempt to remedy this by encouraging members to properly communicate where they are lacking in certain areas.
    - Further Knowledge Risks that may arise are the knowledge gaps between us and what we wish to implement, and what we are capable of implementing. Many of the frameworks we are utilizing are new to multiple members of the group. Therefore, time will be required to have a base level of understanding for how these frameworks operate. We intend to remedy these problems by learning the baseline fundamentals for our frameworks before we attempt to fully implement them.
- Experience Risks
    - Many of our members have yet to have worked on a large scale group project in their computer science career. Some of us will find it hard to properly communicate and convey our emotions and thoughts. We will remedy this by finding mediums which the individual is comfortable with communicating within. This may come in the form of Discord, Trello, Zoom or in class meetings,
    - Many of our members are still fairly new when it comes to large scale projects. If members attempt to spread themselves too thin, then the team as a whole will suffer. In their attempt to work on many things, they will find it difficult to stick and maintain a single task. To remedy this we will try to maintain a firm stance of separation between the backend and the frontend.
    - Maintaining a proper scope of the project is important to prevent us from getting side tracked. We will only implement what is required, namely the functional requirements. To ensure we stay on track we will frequently need to reference the documents we have laid out beforehand. While

things may change, it is important that we have a baseline fundamental guideline to go off of.

Project management

I divide my team to two small group for this milestone because this assignment has 2 major tasks. The first major task is documentation, and the second major task is the vertical prototype. We used Trello to manage the internal deadline. Trello helps my team to keep track of our milestone work progress. We integral the GitHub notification web hook to discord. The web hook can notify who and when submitted the code to the repository, then we can pull new code from GitHub. This webhook helps up avoid GitHub merge conflict. We use GitHub Action to automatically deploy the latest code to the server and check if the code is able to compile. The GitHub notification web hook and the GitHub Action greatly improved our work efficiency.

Knowing to use the proper tool is not enough to manage a team to finish such a big milestone. As a team leader, I asked every team member if they have any difficulty, or if the code had a bug. I would help or guide them to solve the issues they have. Because of our excellent management strategy, we are able to finish this milestone on time.

Detailed list of contributions

Name	Score	Contribution
Elisa Hsiao-Rou Chih	10	mockups for use cases 6 & 14 and 12, vertical prototype (styling, accessing the server, etc) contribute on every pages on vertical prototype.
Steven Paul Fong	10	mockups for use cases 7, 10, and 11 and use case 8, helped come up with design for home page/nav bar, worked on search results page/styling, worked on search parameter pulldown
Cameron Michael Yee	9	prioritized functional requirements, m1v2 revisions, database architecture ERD + requirements, api's and algorithms editing, UML diagram.
Michael Harrison Chang	10	worked on the login page, resizing the input for login, and register, created the nav, footer and errorpage, took the picture of the background, made the story board for case 2,3,4,5,9 , hard coded these sections classes, student info, important date, student records, finances and recourses, working on the pathing for the home page too
Christopher Alan Yee	9	Worked on UML diagram, Data Definitions, High level application network and deployment diagrams, identifying key risks, updated and created a new database Worked on the document Main Algorithm writing.
Zhenyu Lin	9	Worked on fixing vertical prototype bug. Worked on the document High Level APIs section Finished backend development Worked on part of document High level database architecture and organization