Phase II report

Team X

Uzma Kapadia

John Jakobsen

Khuziama Rehman

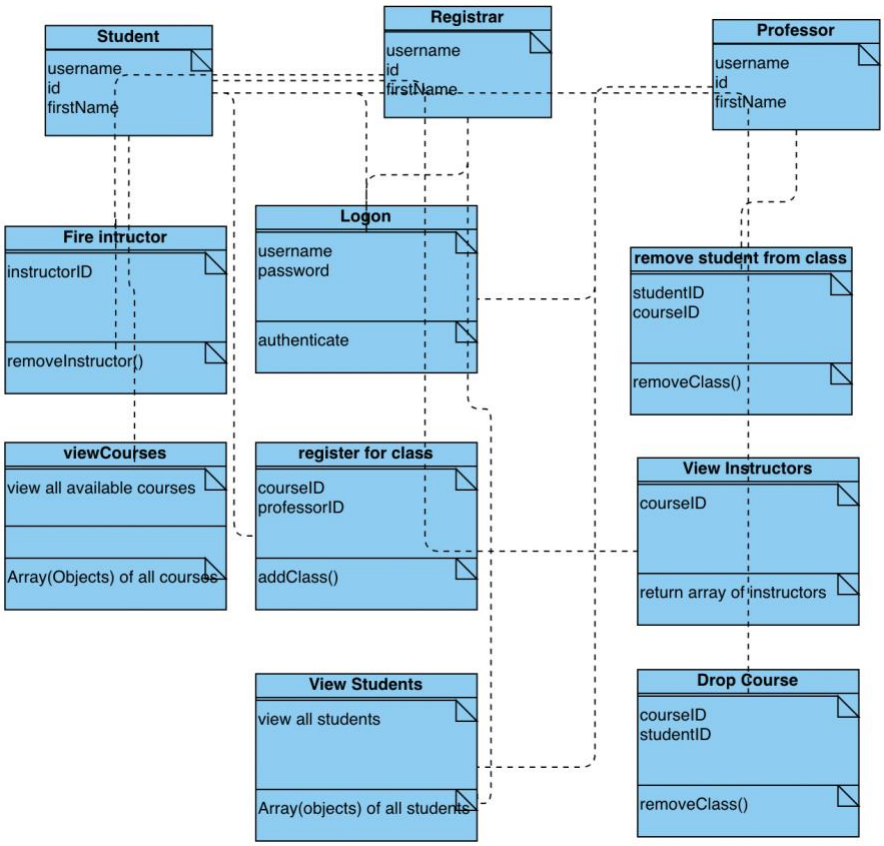
Shun Xie

Table of Contents

Introduction:

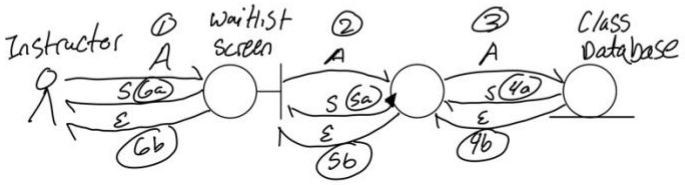
This project requires us to make an application called “GradSchoolZero”. This application is meant to serve as a portal for grad schools to use for their students, instructors, and registrar to maintain a record system for all. “GradSchoolZero” can be used by students to register for courses and check their course history throughout school as well for accessing other student data. Depending on the time of the semester students would be able to leave reviews for courses and see grades for all courses. As for instructors, they would be able see all the classes given to them by the registrar, grade students, receive warning when they have an average of less than 2 rating and some other functionalities. Registrars would be able to assign classes to professors and handle other registrar issues such as accepting students, reviewing graduations applications, hiring/ firing professors, and other such logistical matters.

Visitors can also use this application to see the courses being offered at the school, the top students as well as apply to the school as student or instructor. Below you can see the class diagram representing the overall structure of our version of the application “GradSchoolZero”.



Use Cases:

Instructor (Let student join from waitlist)

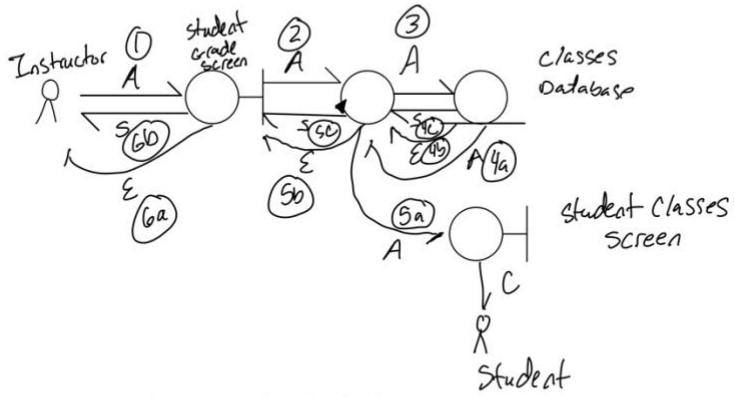


A – Student + course to Let student join

S – Success

ε – Error

Instructor (Assign Grade to Student)

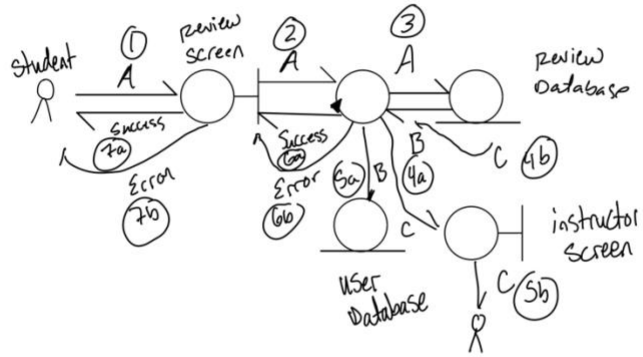


A- Student to apply Grade to + Grade + Class

C – Grade + Class

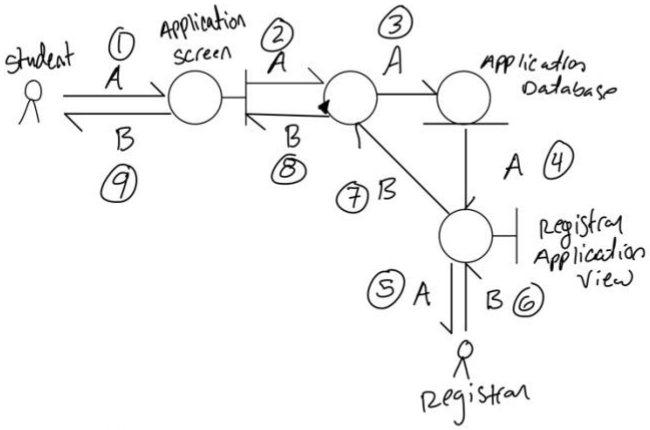
ε – Error , S – Success

Student (Review Class)



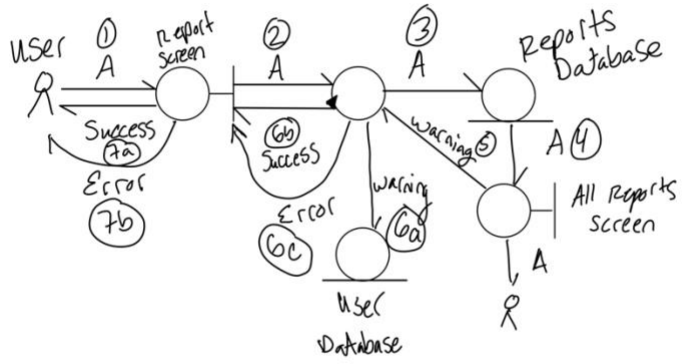
A – Review + Review Score
B – Warning
C – Suspend instructor

Student (Apply for Graduation)



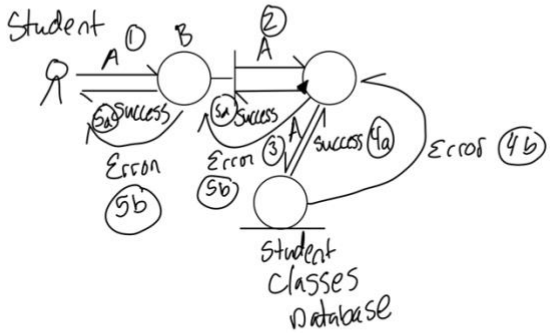
A – Graduation Application
B – Application Decision

User (Report student/instructor)



A- User to report + User info + Reason

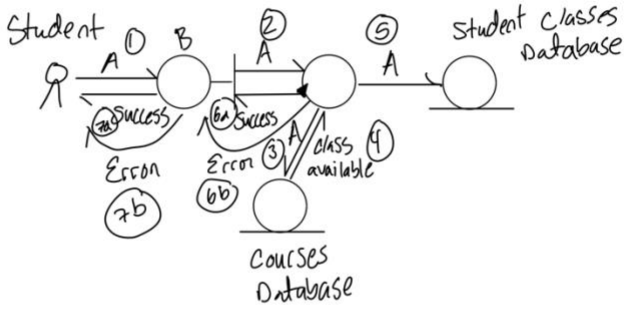
Student (Drop class)



A- Class to drop

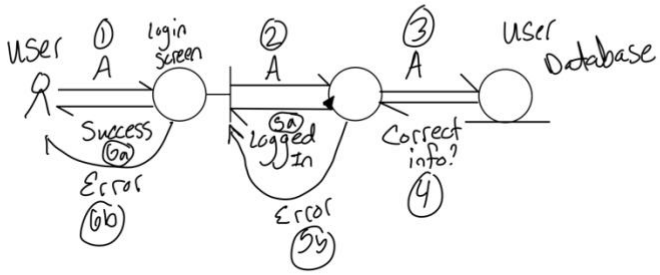
B- Student current classes screen

Student (Class register)



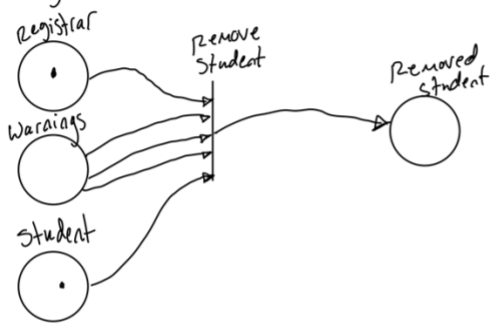
A- Class to register for
B- Class registration screen

Users (Login)

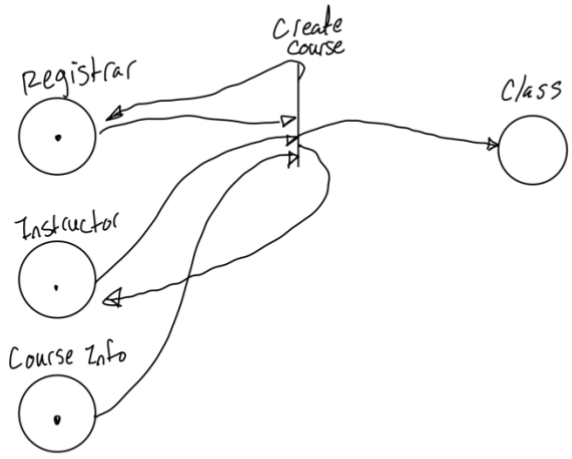


A- Login info

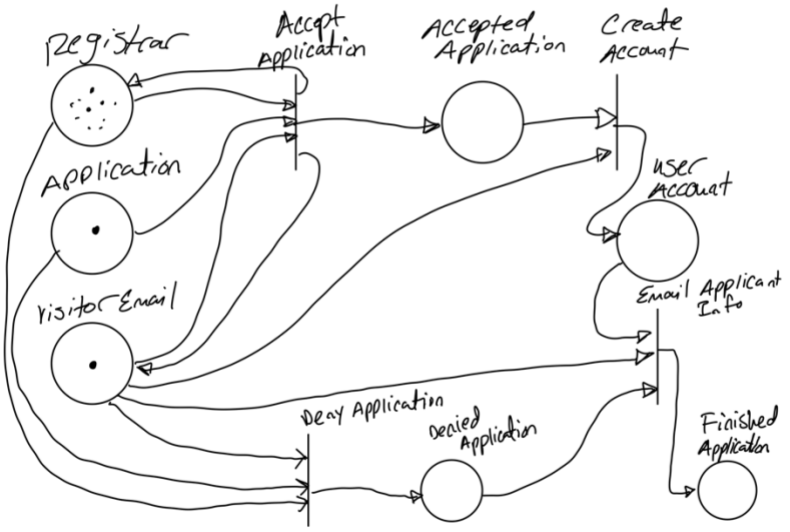
Registrar (Remove Student from program)



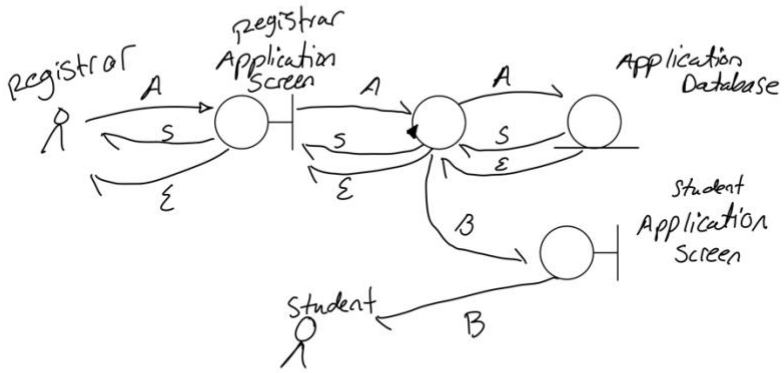
Registrar (create class)



Registrar (Accept/Deny Visitor Application)



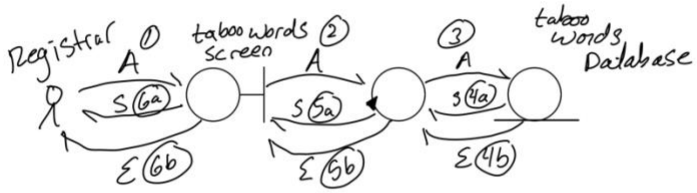
Registrar (Accept/Deny Graduation Application)



A - Application to consider S - Success

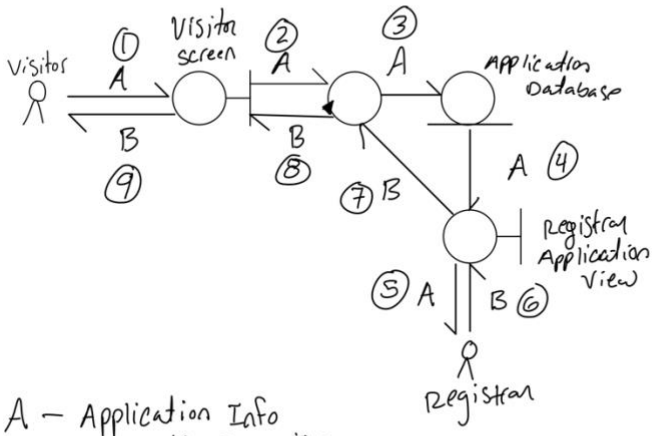
B - Application result ε - Error

Registrar (Create taboo word list)



A - List of taboo words
S - Success
E - Error

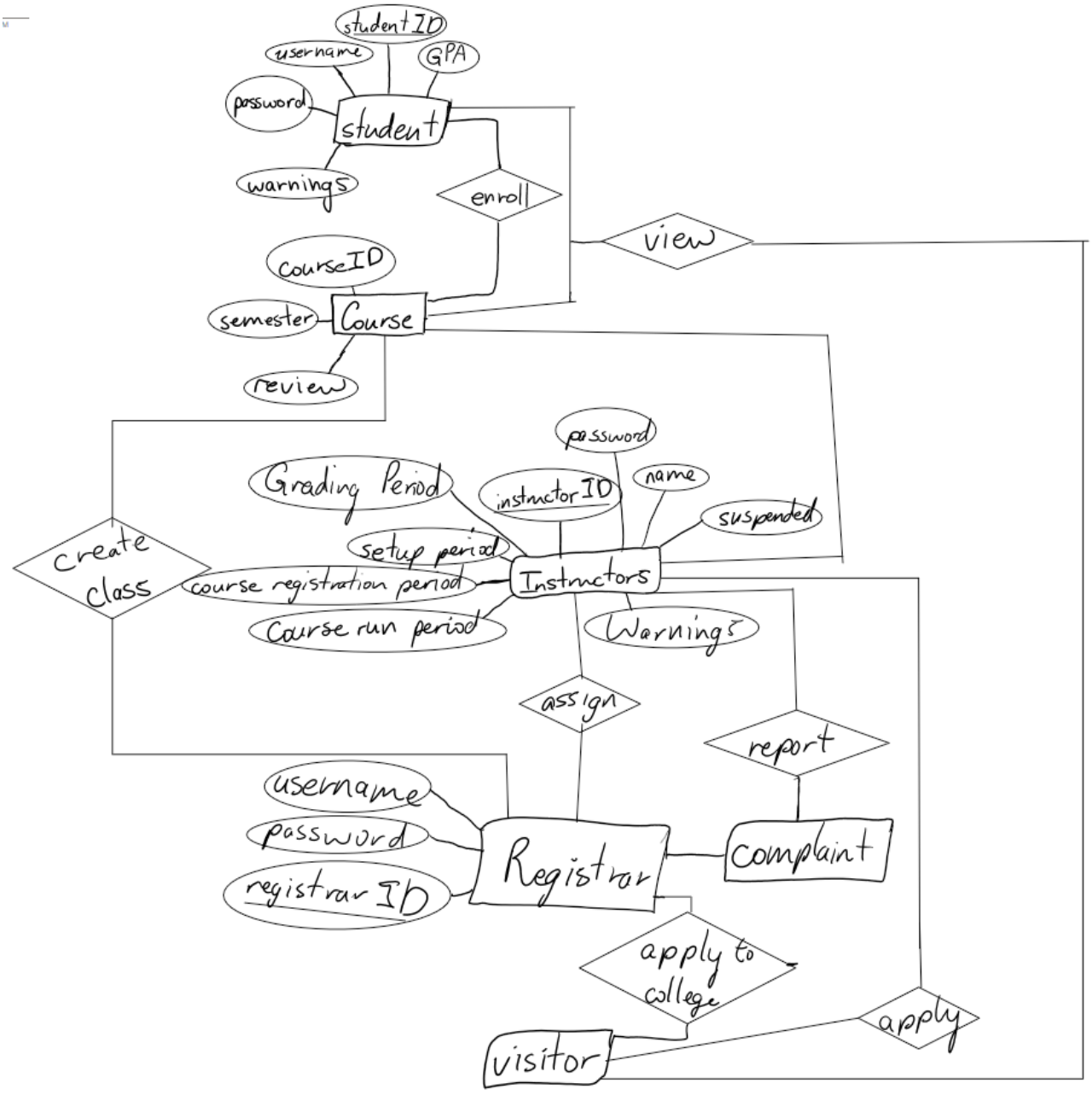
Visitor (Application)



A - Application Info
B - Application Decision

E-R Diagram:

This shows the E-R diagram for the application.



Detailed Design:

Psuedocode for functions

User login method finds all users in the type table, which can be either student, professor or registrar where the username and password match the inputs, if they do, we authenticate the user, else we return false.

Function userLogin(username,password,type){

Databaseclient.query(`Select * from type WHERE username = username AND password = password`).then((res) => if len(res) > 0 then return true else return false}

}

/////////

Register for course method will first check if user meets pre requisites for course, then inserts course into the users courses in the database and returns a success message

Function registerForCourse(userID,courseID,professorID){

If (ValidatePreRequisites(userID,courseID)) {

databaseclient.query("INSERT INTO userID Course(courseID,professorID) values (courseID, professorID)").then(console.log("successfully registered for course")

}

}

/////////

Function dropClass(userID,courseID){

Search for course we want to drop by searching through all courses user is taking

Then we can delete the course from the users course list,

Finally we need to remove the student from the course from the instructors side.

}

//////

Function graduationRequest(userID){

Check if user completed all required courses,

If yes, then return true and graduate student,

Else we return false and deny graduation.

}

////

Function reviewClass(){

Check if intructor has numerous complaints,

If complaints exceed the limit, then we run the fireInstructor method, else we give a warning to professor.

}

///

Function createClass(className,classProfessor){

NewClass = new Course(className,Professor)

Insert NewClass to database and notify all students of the new class

}

////

Function reportUser(userID){

Check if user is a student or professor.

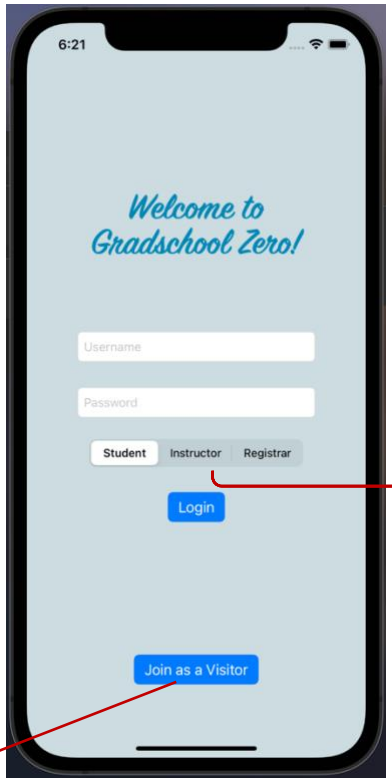
If user is student, check if they have previously been reported, if yes, then suspend student, else give them warning.

Else if user is professor, allow 3 warnings, if professor gets reported more than 3 times, then fire them.

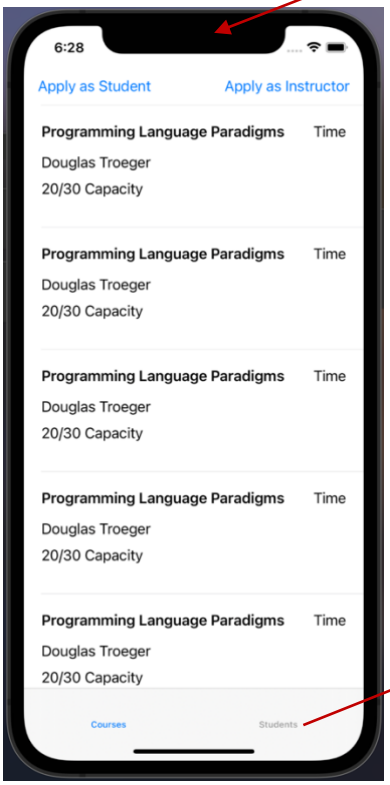
}

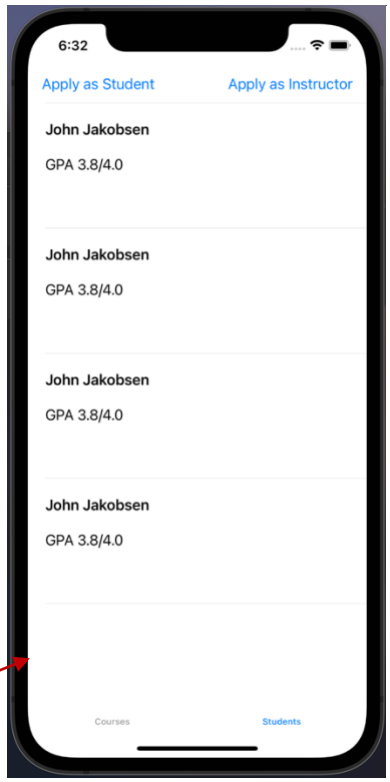
System Screens:

The main screen when the app is opened:



The user can select whether they are trying to log in as student, instructor or registrar.



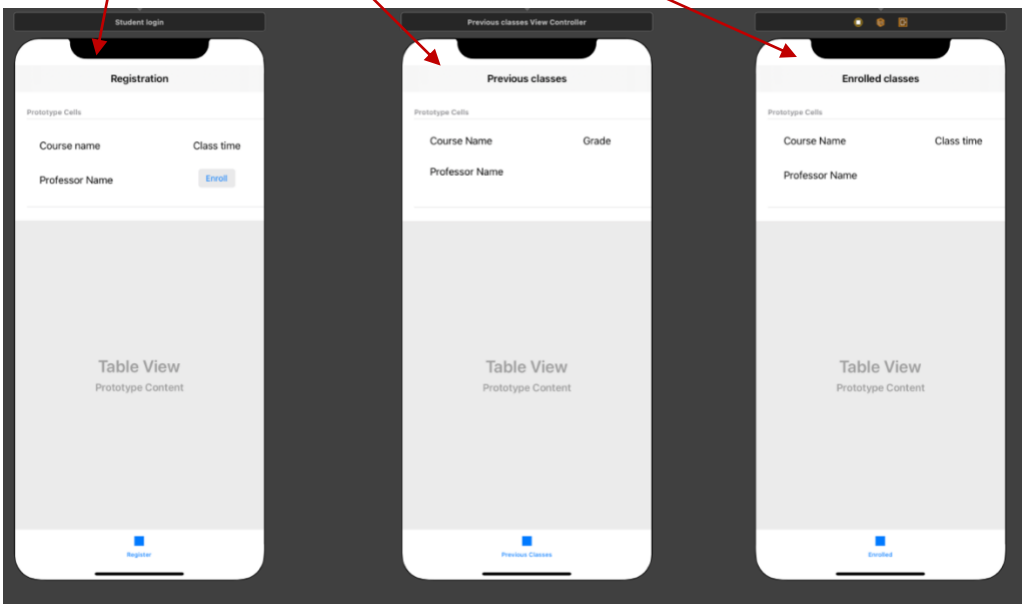


- These are the two screens with the courses and the students that the visitor Is able to see. The data that is shown at the moment is just some dummy data that will be changed so different classes and different students are shown.

- The visitor can click “Apply as Student” and “Apply as instructor” buttons if they would like to apply. For now, these buttons are functional but will just take you to a blank screen as there is application data added.

When a student logs in they are taken to their student profile/info page shown below. From here the student can press on the different on the tab bar at the bottom to register for classes, see previous classes and the classes they are currently enrolled in. These buttons can change depending on the time of the semester for example if registration period is over the registration button would not be there.

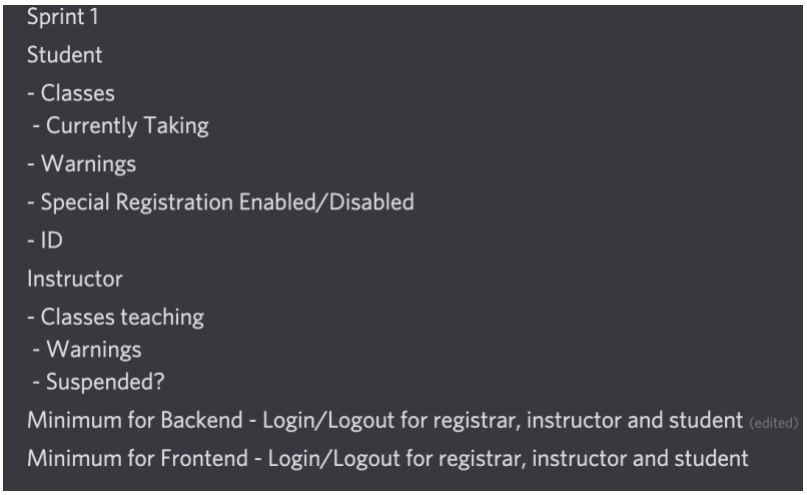
The three screens below for registration, previous classes, and enrolled classes would show a list of the courses for the different categories. As it is not yet connected to a database it does not show a list of data yet however, you can see the type data it would show such as course name, professor name, etc. as seen below in the cell configuration.



**Screens for instructors and registrars once logged in are in progress but have not yet been fully created hence not present here.

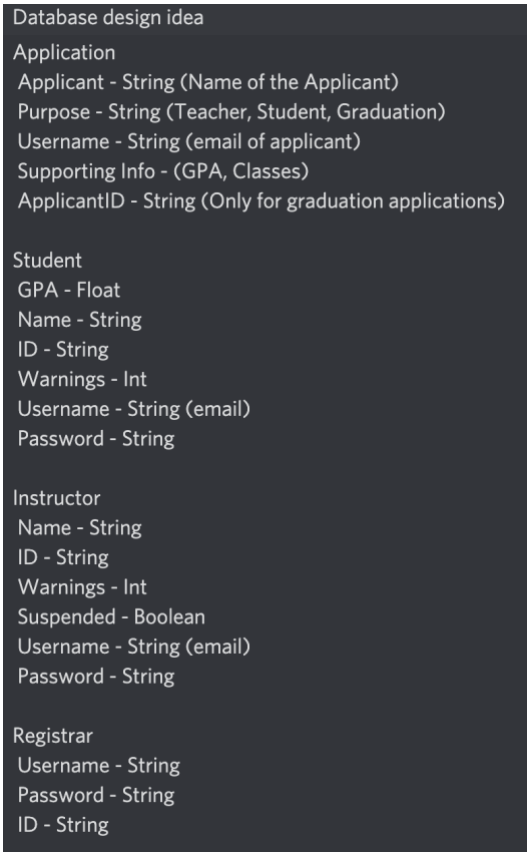
Group meeting memos:

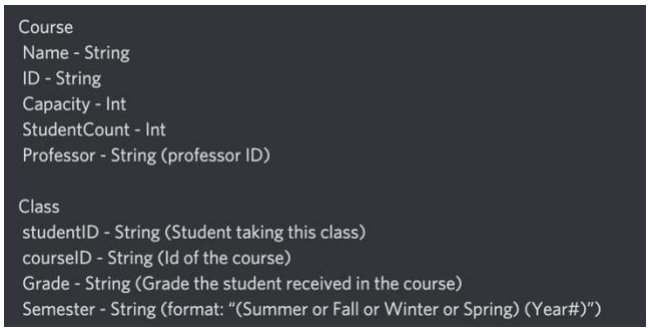
Throughout the span of this project, we have been meeting weekly through group calls to discuss the application and our plan of action. During our meetings we discuss our progress, what we still need to work on, work distribution, and if anyone is stuck on anything or struggling so we can help each other out. Throughout the week if we have updates about something we do it through group text and if there is info we need to remember it is usually just left in this group chat. We do not have written meeting memos from each of our meeting but below is some screenshots of what we do have.



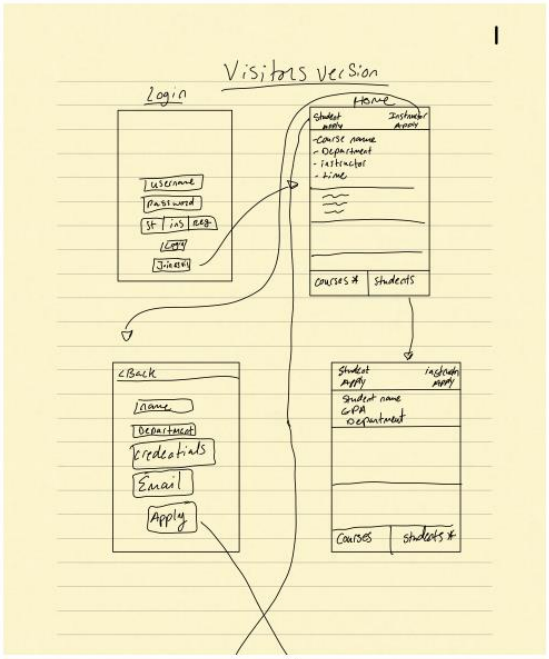
Link to the project specs description by one of our members to help get better breakdown and understanding of what we have to do:
https://docs.google.com/document/d/10aIbSW6pP7Fp2I4OcuElEeEHZLTHiGUiidM5gQNNwkM/edit

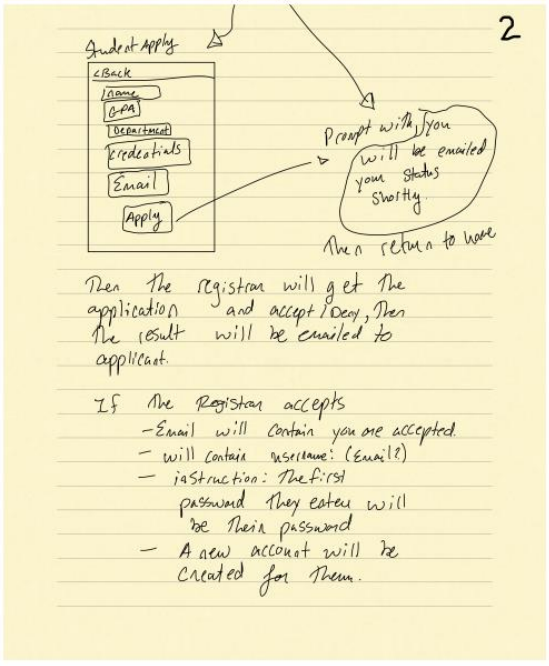
This was a discussion about the database design for the application:

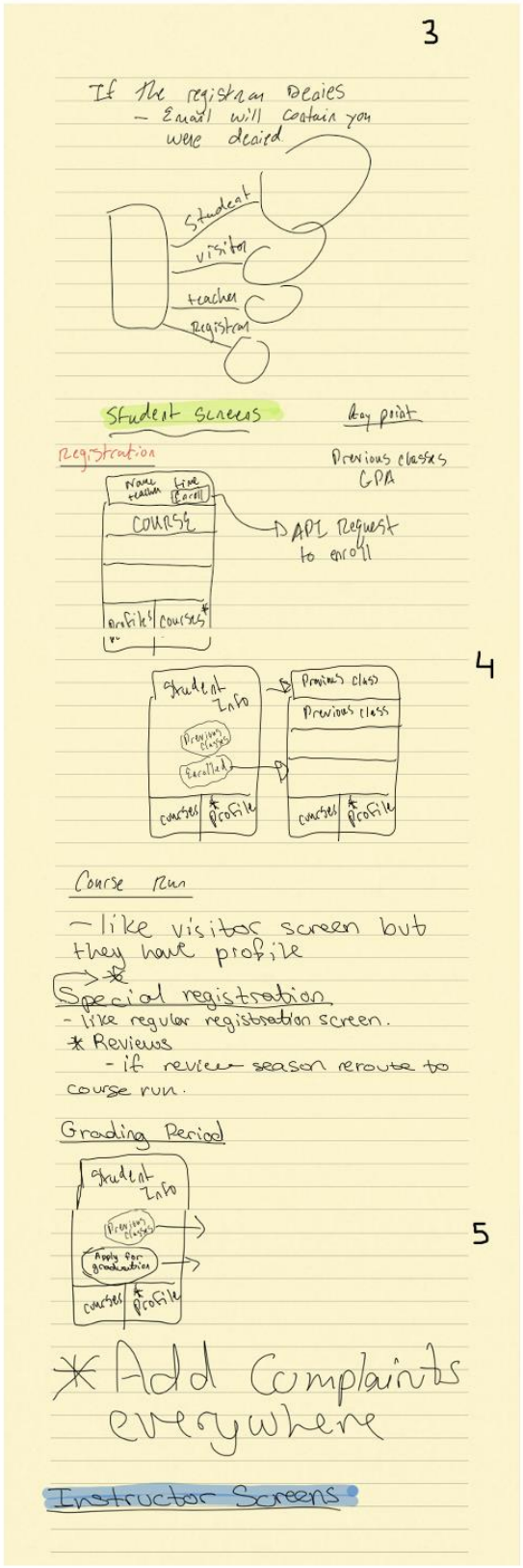




This shows the discussions on designing the app and the different fields we need to satisfy.







Git Repo Address:

Frontend iOS design(objective-C): https://github.com/jerikjakobsen/GradSchoolZero.git

Backend: https://github.com/KhuziamaR/GradSchoolZeroBackend.git