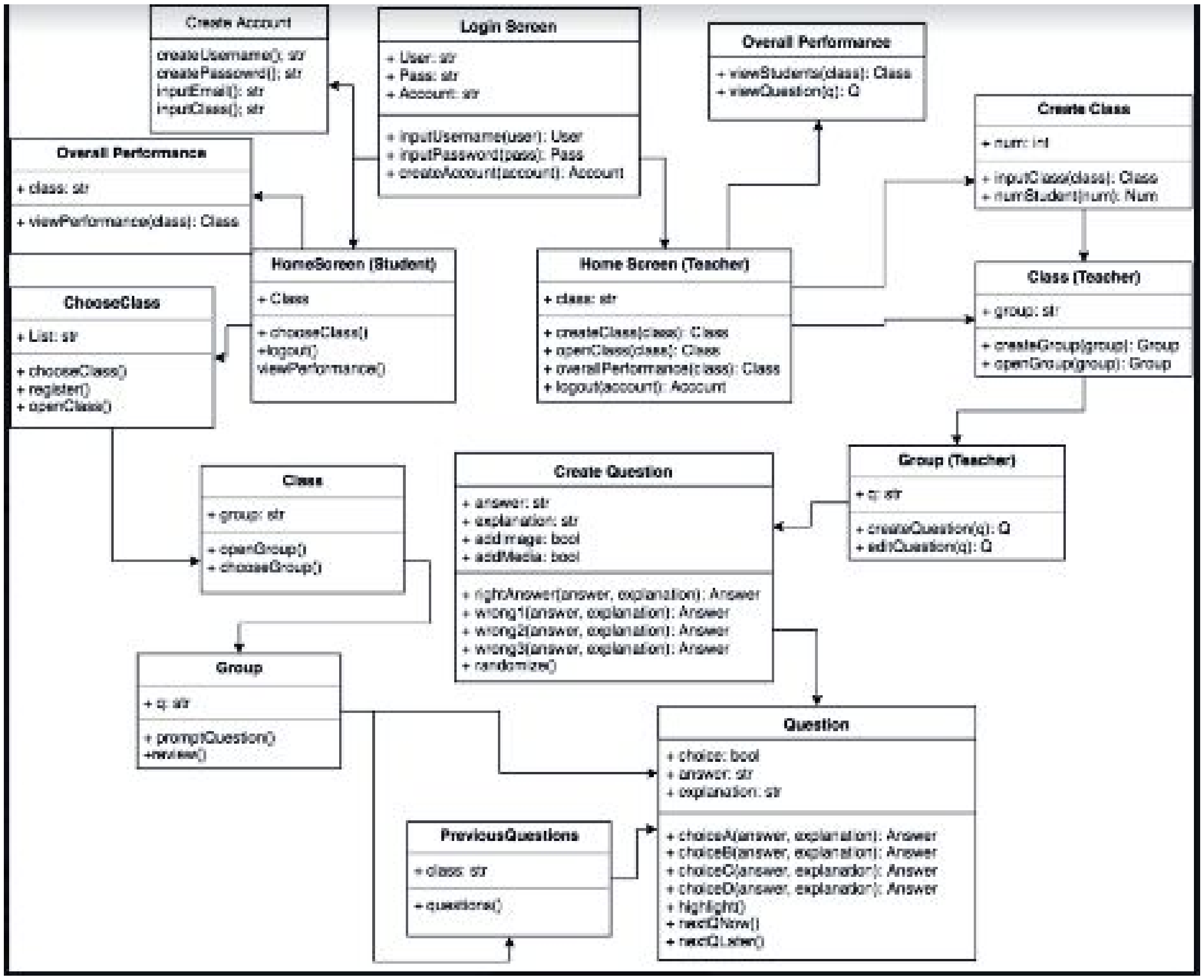
Medical Education Application
Developers: David Mattingly and Nathan Schmidt
Clients: Dr. Matt Broom and Dr. Chris Brownsworth
Supervisor: Dr. Jason Fritts
Deliverable #2
5/11/18

Design Document

UML Diagram:



Major Components for Finished Application:

The application will first bring up a login screen, which will prompt the user to either login with a username and password, or create a new account. Creating a new account gives the choice of whether the user is a student or a teacher, and this choice determines how the app behaves. Since the questions will be officially made by doctors for medical school, users of the app will also be prompted to an end user legal agreement before creating an account. When teachers successfully log in to the app, they will have the option to create classes, post questions (which will be sorted into groups based on what topic the question is for), control privacy restrictions of the content, and receive feedback reports of the students' performance. When students successfully log in to the app, they will be able to pull up a list of classes to register for (which the teacher will approve), filter questions by groups, highlight specific questions to review, and access reports for their overall scores and compare results.

Questions will be in a multiple choice format, and explanations will be given for each choice, regardless of if it's right or wrong. The teacher will have the option to add pictures or media to the questions as well as links to access more information related to the material. The questions will be peer reviewed, posted to private groups or all people using the app, and have the option to mark as a copyright violation. After answering the question, the students will be able to decide if questions will be posted either in blocks or over time. Overall, the app will be structured to be easy to use and able to access "teacher" or "student" functionality within the same app. The information and statistics from the app will be recorded for educational studies including demographic information if needed.

We will be developing the application for iOS devices, so we will be coding the application in Swift 3, using JavaScript for the screens, and mySQL for the database of questions. Our goal is to have a prototype for the question screens and login by Deliverable #2, an alpha for the application as a whole by Deliverable #3, and a finished product by Deliverable #4.

Timeline for remaining goals:

- Goals for Deliverable #3:
    - Create a user database so the data / progress can be saved
    - Since the student account is basically finished, implement the teacher account
    - Test new features, such as rolling out automated questions over time and having teachers create questions
    - Test entire app end to end
    - Complete user testing
    - Analyze user testing feedback form sponsor
- Goals for Deliverable #4:
    - Test final features
    - Prepare for application rollout
    - Complete user guide for the application
    - Obtain final feedback from the client
    - Roll out app to a small pilot group
    - Analyze and fix any issues found by the pilot group

- ○ Finalize the app to be available to all users

<u>What we have designed so far:</u>

When you open the app, it currently opens with the login screen. Since our main goal for this deliverable was to have the question screens working, we decided to not focus on creating a user database for the login info until the next deliverable. We do, however, have the screens for both logging into the app and creating the user. Since the database for the users hasn't been set up yet, clicking "create account" will go back to the login screen, but it won't actually create a new account yet. Likewise, clicking "Log In" on the login screen will take you to the home screen of the app, regardless of if there's any new information inputted.

Currently, the app is designed as if a student is logged in, so when the user "logs in", it shows the home screen for students, with buttons for choosing a class, viewing performance, and logging out. We decided to not implement the viewing performance screen yet since it relies on the user database (so clicking on "View Performance" doesn't do anything), but if you click on "Choose Class", it will take you to the class screen, where you choose a class, and once you choose a class, it has a list of groups to choose from. Currently, we have one class hard coded and one group, and the group has two questions to choose from.

Both of the questions we hard coded into the app take advantage of all of the possible features for making a question. They are multiple choice questions, where each question has 4 choices to choose from, and clicking on each choice goes to a "Correct" or "Incorrect" screen (with an explanation for whether it's right or wrong), depending on which choice you choose. The results screens also have a choice for highlighting the question for later, but again, the user database has not been implemented yet, so clicking on that doesn't do anything. One of the questions contains an image, and the other question contains a video.

<u>Any known bugs / difficulties:</u>

Since the user database has not been created yet, some of the buttons do not function as they're supposed to ("Create Account" on the Create Account screen doesn't actually create a new account, it just goes back to the login screen, and "Login" on the login screen goes to the Home Screen regardless of if anything is inputted), and some buttons don't do anything when you click on them (specifically the "View Performance" button on the Home Screen and the "Highlight Question for Later" button on the question results screens). We also still have not figured out how to connect the application to a server.

<u>Code for question screen</u>:

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var myWebVideo: UIWebView!


    @IBOutlet weak var textOptionA: UILabel!

    @IBOutlet weak var textOptionB: UILabel!

    @IBOutlet weak var textOptionC: UILabel!

    @IBOutlet weak var textOptionD: UILabel!

    func setAnswers(){
        self.textOptionA.text="5"
        self.textOptionB.text="0"
        self.textOptionC.text="3"
        self.textOptionD.text="15"
    }

    @IBOutlet weak var buttonA: UIButton!
    @IBOutlet weak var buttonB: UIButton!

    @IBOutlet weak var buttonC: UIButton!

    @IBOutlet weak var buttonD: UIButton!

    @IBOutlet weak var answerA: UITextField!

    @IBOutlet weak var answerB: UITextField!

    @IBOutlet weak var answerC: UITextField!

    @IBOutlet weak var answerD: UITextField!

    @IBOutlet weak var screenImage: UIImageView!

    @IBOutlet weak var screenQuestion: UILabel!

    @IBOutlet weak var questionNumber: UILabel!



    override func viewDidLoad() {
        super.viewDidLoad()
        getVideo(videoCode:"anJSrmTVLi8")
        setAnswers()

    }

    func getVideo(videoCode:String)
    {
        let url = URL(string:
"https://www.youtube.com/embed/\(videoCode)")
        myWebVideo.loadRequest(URLRequest(url:url!))

    }

    @IBAction func answerAselected(_ sender: Any) {

    }

    @IBAction func answerBselected(_ sender: Any) {
    }

    @IBAction func answerCselected(_ sender: Any) {
    }

    @IBAction func answerDselected(_ sender: Any) {
    }
```

Requirements fulfilled:

Requirement 1a.
Create a user application interface for IOS mobile devices.

Requirement 1b.
Have a login screen that will prompt the user for a username and password of an already enrolled user, or the option to create a new account for a new student or teacher.

Requirement 1c.
The new user screen should prompt the user for their email address, password, username, and whether they're a student or teacher. **- database for users not created yet**

Requirement 2a.
Have a home screen for the student account with the ability to access currently registered classes, join a new class, view their performance, and a logout option.

Requirement 4a.
When the student has the option to select a class and open the groups of questions available or request to register sending the request to the teacher.

Requirement 4b.
When the student has access to the class it will select a group of questions to open the content available.

Requirement 4c.
For every group of questions, the student will have the option to prompt a new question, or review previously answered questions, with the option to highlight questions they want to focus on.

Requirement 5b.
The question screen for the student will show the question, the multimedia, image or links added by the teacher, and the multiple possible answer for the student to choose from. When an answer is chosen, an explanation for whether the answer is right or wrong will be prompted.

Requirements remaining:

Above all else, create a user database, but besides that, the remaining requirements are as follows:

Requirement 2b.
Have a home screen for the teacher account with the ability to create a new class, open a existing class, access the overall performance of the students, and logout.

Requirement 3a.
The teacher account will have the create new class option, which will prompt the teacher to enter a class number and name.

Requirement 3b.
In an created class, the teacher will be able to create a new group of questions to provide to the class, or access a group of questions that already exists.

Requirement 3c.
For every existing group of question created, the teacher will be able to edit current questions, and create a new question to be posted.

Requirement 5a.
The teacher creating a new question will type the question, create multiple answer choices for the students marking which are the right answers, add an image, multimedia, or links if desired, and provide an explanation for each answer.

Requirement 5c.
The student will be given the choice to access the next new available question after answering a question, or be prompted to answer a new question in 24 hours.

Requirement 6a.
The overall performance screen for the teacher will show each of the students' performance in a question group, how successful the students are with a specific question,

Requirement 6b.
The overall performance screen for the student will show the percentage of answering the questions right for all the different group of questions they have answered.

Requirement 6c.
Have the app running and connected to the server using the Amazon Web Services tools.