## MEDICOLLAB

BY ARINDHAM ANEJA, YIRAN SHENG, PEIYU JIANG

## PROJECT OVERVIEW

The purpose of Medicollab is to create a platform where:

- Medical Students can collaborate with professors to research a topic on related subjects and gain work experience.
- Students can search for professors based on their expertise in the subject and send them a request with a brief description of their incentive to work with the professor.
- Professors can assign different tasks to students to help develop their knowledge and grade students for their completed assignments.

## SYSTEM OVERVIEW

This app will be developed using the **Model-View-Controller** architecture.

- ➤ The **Model** contains the pure application data and interacts with the controller by providing it the information and getting updated in return
- The **Controller** exists between view and model. It listens to events triggered by the user and executes the appropriate reaction by calling a method on the model and reflecting it in the view.
- The **View** presents the data to the user. It is responsible for the visual representation of the app.

ARCHITECTURAL STYLE & SYSTEM DIAGRAM

MC

**VIEW** 

- Professor data
- Student data
- Assignments list

MODEL Update result

Send input from user

Update the results to view

User

CONTROLLER

- Assign tasks
- Send requests

## ACTOR IDENTIFICATION

#### The actors include:



Uses the app to search for professors and work with them to gain experience and knowledge.



Uses the app to mentor students, acquaint them with concepts of a particular medical subject and grade them on the basis of knowledge they display.

### DESIGN PATTERN

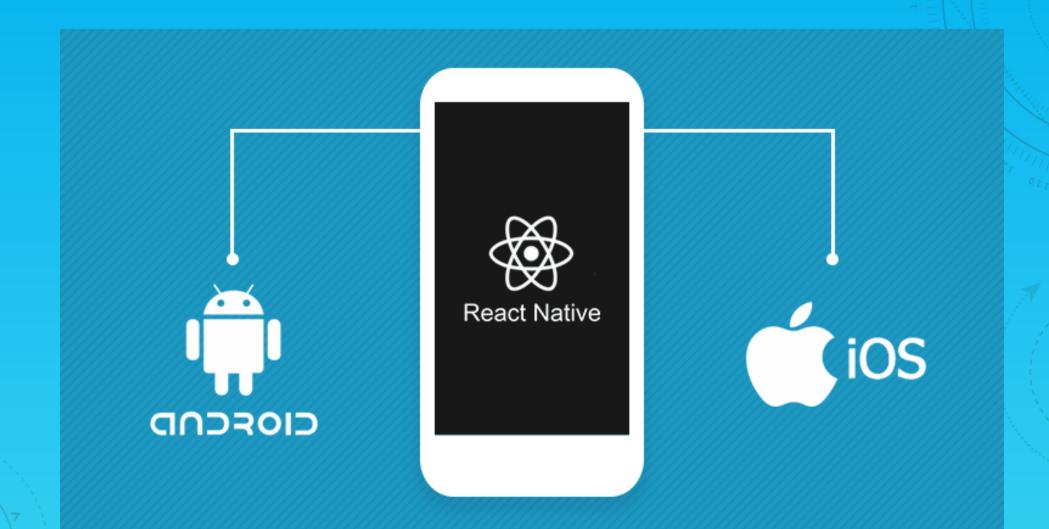
#### Singleton Method

It restricts a class to instantiate its multiple objects and is defined in such a way that only one instance of the class is created in the complete execution of a program. This pattern will be used in student and professor class to create only one instance of student and professor at one time.

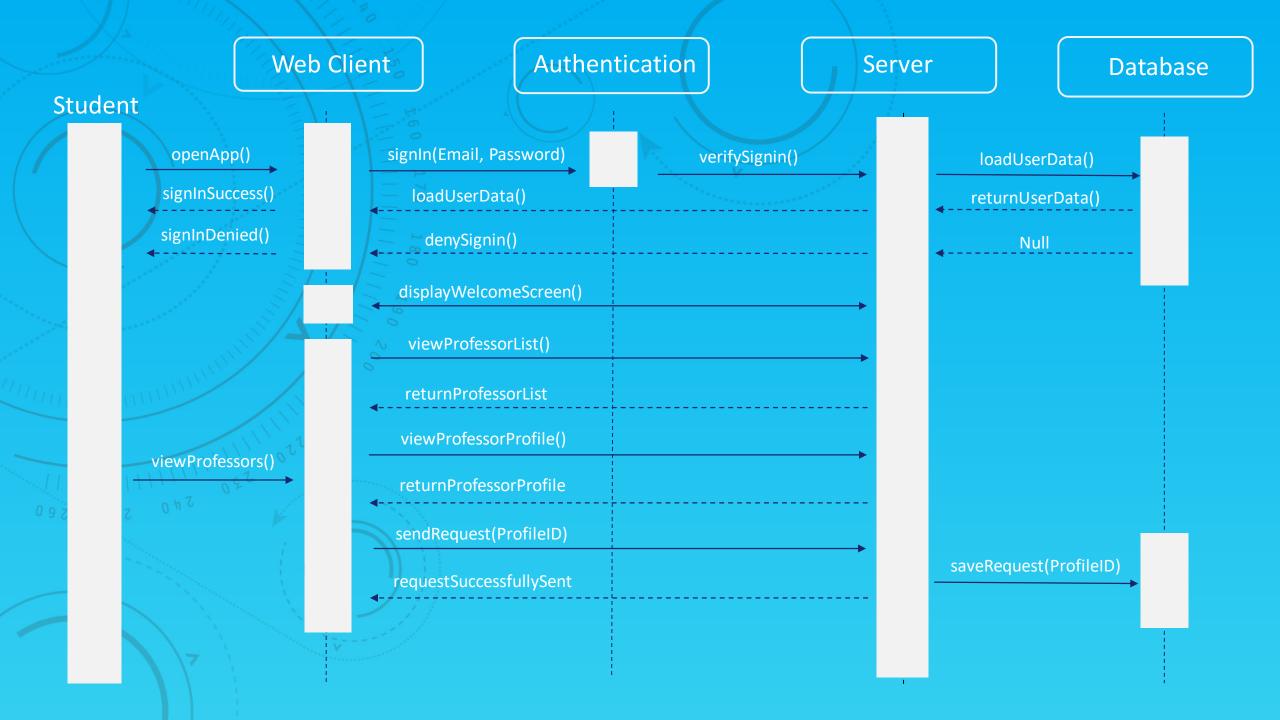
#### **Factory Method**

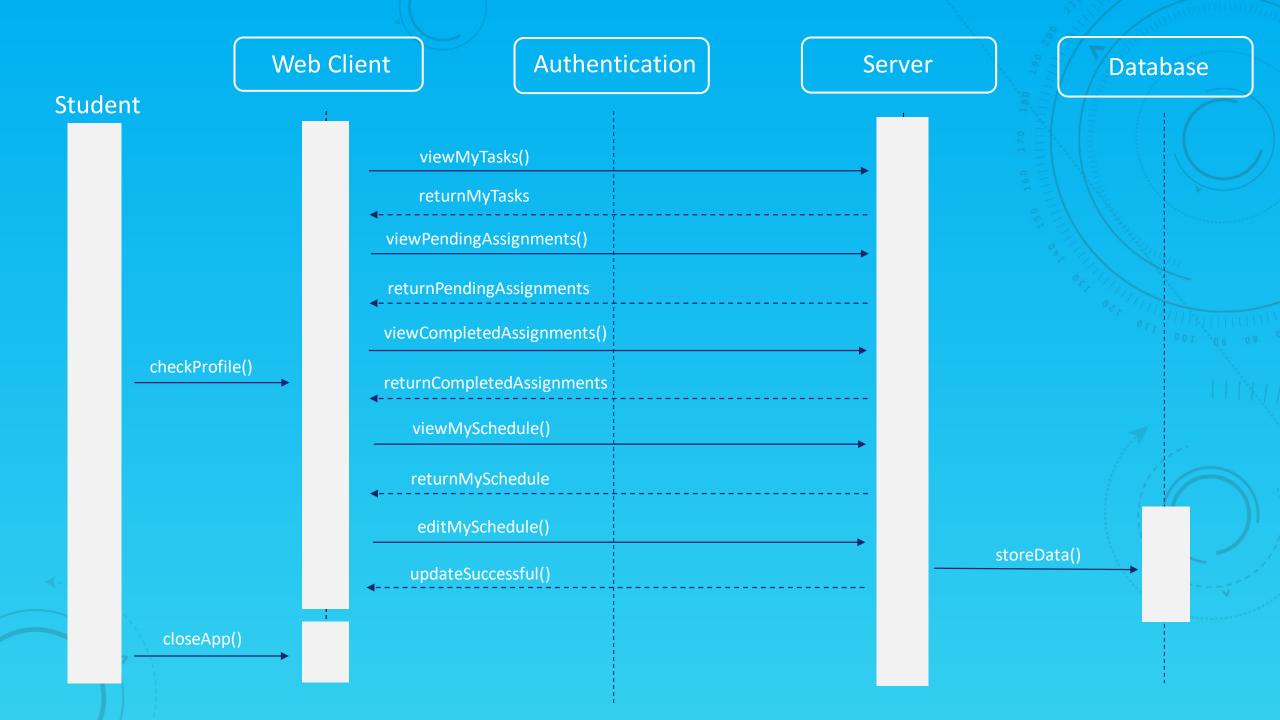
Provides approach to code for interface rather than implementation and abstraction between implementation and client classes through inheritance. It removes the instantiation of actual implementation classes from client code and keeps the code more robust, less coupled, and easy to extend. Because the student and professor classes share some common properties, we will create the abstract objects of student and professor classes in person class by using this method

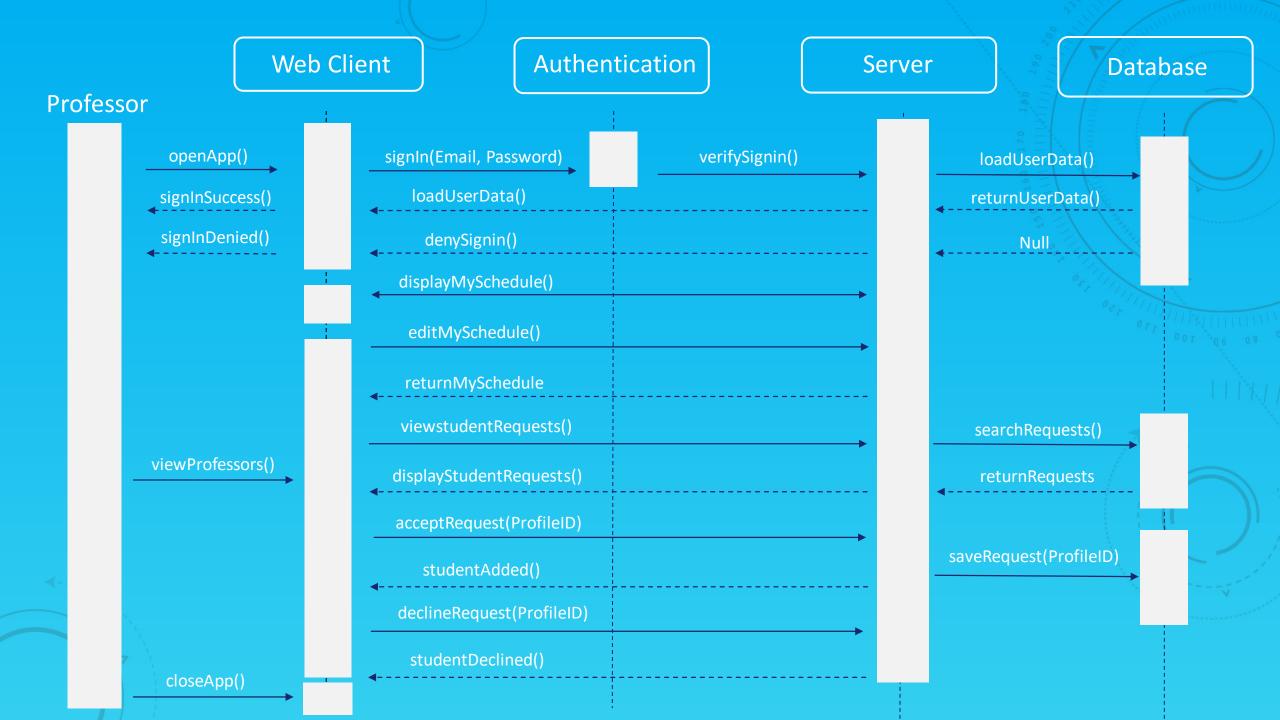
## FRAMEWORK



## SEQUENCE DIAGRAM







#### **CLASS DIAGRAM**

#### User

Name: String Email: String Password: String School: String Role: String

openApp()
signUp()
signIn()
resetPassword()
editProfile()
logout()

Manages

Settings

User Details:

editName()
changePassword()
changeSchool()

#### Student

StudentID: String

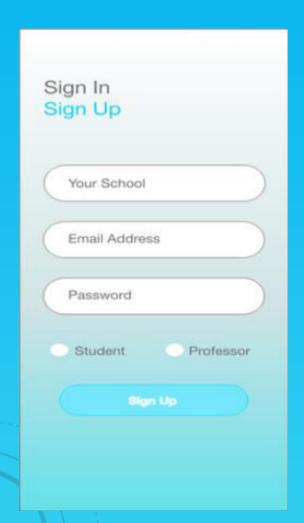
sendRequest()
viewProfessorList()
viewProfessorProfile()
viewMyTasks()
viewPendingAssignments()
viewCompletedAssignments()
viewMySchedule()
editMySchedule()

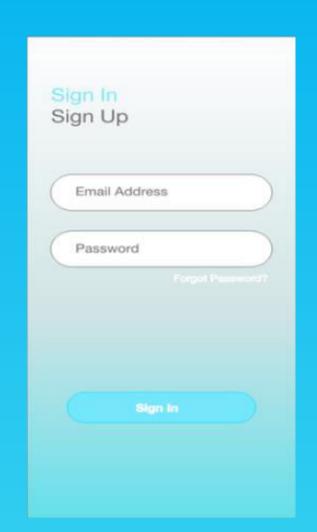
#### Professor

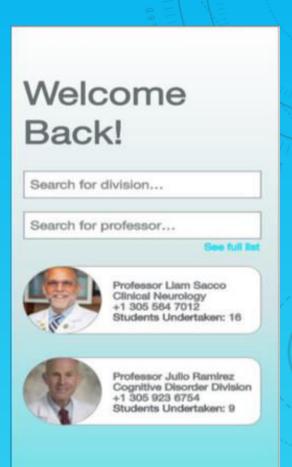
ProfID: String

viewAllStudents()
acceptRequest()
declineRequest()
viewStudentRequest()
assignAssignment()
viewMySchedule()
editMySchedule()

## **MOCK UP**









Professor Liam Sacco Clinical Neurology +1 305 564 7012 Students Undertaken: 16



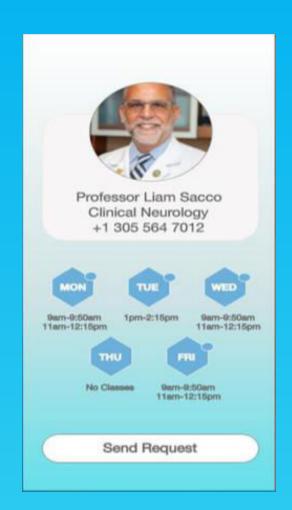
Professor Julio Ramírez Cognitive Disorder Division +1 305 923 6754 Students Undertaken: 9



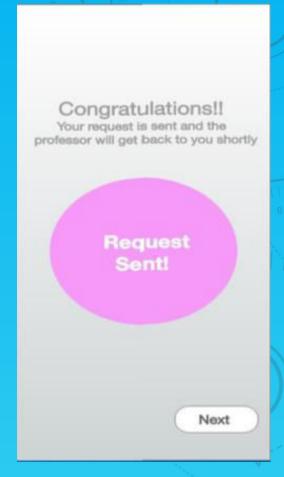
Professor Sean Bernard Dermatology Division +1 305 120 1221 Students Undertaken: 12

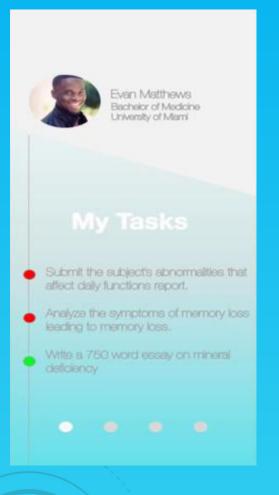


Professor Walter Gills Gastroenterology +1 305 666 1414 Students Undertaken: 21

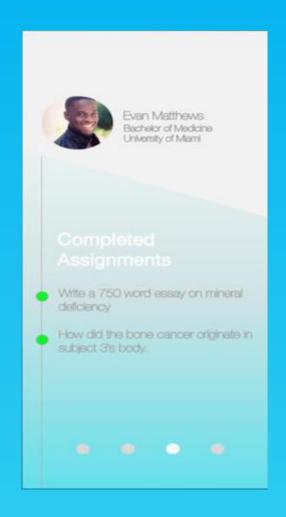




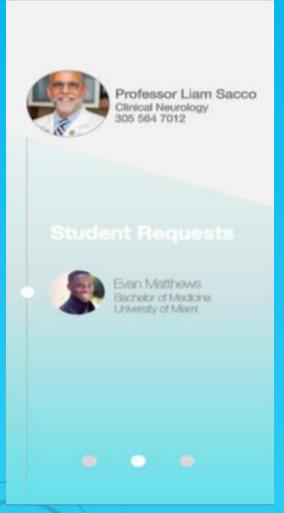




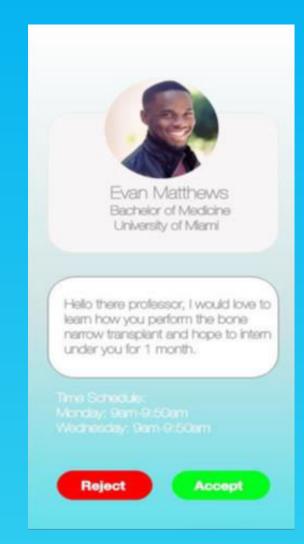














# THANK YOU FOR LISTENING ANY QUESTIONS?

https://github.com/arindham123/CSC431