Detailed Design Document: F=MA

Nick Richardson
Kaylee Christie
Dafne Lazaro
Aidan Sheridan
Noah Cunningham

I. Table of Contents

Introduction

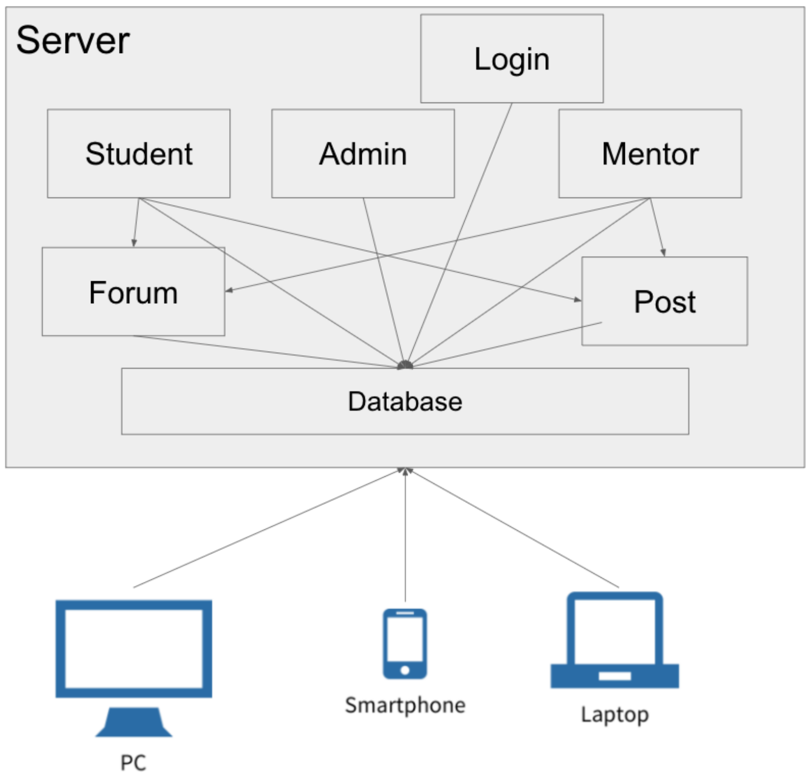
F=MA is the first social media platform designed specifically for college students with an emphasis on mental health and fighting imposter syndrome. At a very high level, there are three basic functionalities of this system. The first functionality of this system is that it will allow students to make posts about their academic or research projects, and other students can interact with these posts and comment or like them. The second feature allows the students to connect with mentors who do work that interests the student. Mentors can also initiate this connection for recruiting purposes for graduate school. Lastly there will be a forum where students can start a conversation about what they are concerned about, have a question about, etc. Other students can comment on these forums answers to the questions or just general advice.

The first task that we needed to do as system developers was to break down each of these features into its smallest, most basic elements. We ended up breaking down our modules into the most fundamental CRUD statements, and organized them into Student modules, Mentor modules, Forum modules, and Posts modules.

We additionally are creating a bottom up test plan where we will create each of the individual modules and test them and then build the system up from each of our individual tests.

Architecture

This application uses client server architecture. This allows multiple users and tasks to occur at the same time which is a nonfunctional requirement needed for this application when students and mentors alike will be using it. The distribution of data is also straightforward which is one of the advantages of this type of architecture.



Module Design

The modules section is broken up into 6 main modules (Student, Mentor, Administrator, Forum, Post, and Login) with each of these modules being broken down into smaller submodules that have individual functionality.

List Of Modules

1. Login
2. Administrator Module
    a. Enroll school module
    b. Edit school module
3. Student Module
    a. Create New Student Module
    b. Edit Student Profile Module
    c. Student Request Module
4. Mentor Module
    a. Create New Mentor Module
    b. Edit Mentor Profile Module
    c. Mentor Request Module
5. Post Module

    a. Caption Module
    b. Picture Module
    c. Comment Module
6. Forum Module
    a. Thread module
    b. Forum Subject Module

<u>Login Module</u>

Purpose of Module

The purpose of this module is to allow users (student, mentor, administrator) to login to their respective accounts and use this platform.

Provided Interface

No provided interface for this module

Required Interface

db.login(String Username, String Password)

Description: Allows the user (student, mentor, administrator) to login to their account.

Inputs: String Username, String Password

Returns: the account of the person

db.getAccount(String AccountID)

Description: Returns the account of the person who logged in

Inputs: String AccountID

Returns: the account connected to that AccountID

<u>Administrator Module</u>

Enroll School Module

Purpose of Module

The purpose of this module is to enroll individual schools into F=MA so the students can use this platform for free of charge.

Provided Interface

paymentMethod(String creditcardnumber, String expirationdate, String securityCode)

Description: Allows the administrator to pay for access of students from that school.

Inputs: String creditcardnumber, String expirationdate, String securityCode

Returns: None

Required Interface

db.setSchool(String College Name, String<list> majorsIncluded)

Description: Setter method to the database for the schools

Inputs: String College Name, String<list> majorsIncluded

Returns: None

db.getSchool(String schoolID)

Description: Getter Method from the database for the school

Inputs: String schoolID

Returns: The school from the database

Edit School Module

Purpose of Module

The purpose of this module is to edit individual schools into F=MA, allowing the school administrator to include additional majors used for the school.

Provided Interface

editMajors(String SchoolID, String<listOfNewMajors>)

Description: Allows administrators to edit the school profile and change which majors can use this system.

Inputs: String SchoolID, String<listOfNewMajors>

Returns: None

changePaymentMethod(String creditcardnumber, String expirationdate, String csv)

Description: Allows Administrator to edit the payment method for the school.

Inputs: String creditcardnumber, String expirationdate, String csv

Returns: None

Required Interface

db.setSchool(String College Name, String<list> majorsIncluded)

Description: Setter to the database for the school

Inputs: String College Name, String<list> majorsIncluded

Returns: None

db.getSchool(String schoolID)

Description: Getter from the database for the school

Inputs: String schoolID

Returns: The school connected to that school ID

Student Module

Create New Student Module

Purpose of Module

The purpose of this module is to create the student account for a new student user.

Provided Interface

deletestudent(String studentID)

Description: Allows the student user to delete their account

Inputs: String studentID

Returns: None

liststudents()

Description: Lists all the students using this platform

Inputs: None

Returns: List of all students

Required Interface

db.setstudent(Student student)

Description: Setter method for student into database

Inputs: Student student

Returns: none

db.getStudent(String studentID)

Description: Getter method for student from database

Inputs: String Student ID

Returns: Student object

createstudent(Student studentID, String password, String email)

Description: Creates student profile and puts it into database

Inputs: Student studentID, String password, String email

Returns: None

Edit Student Profile Module

Purpose of Module

The purpose of this module is to allow students to make changes to their profile page on F=MA

Provided Interface

changeProfilePicture(String image)

Description: Changes the profile picture of the student account

Inputs: String image

Returns: None

changeBioParagraph(String text)

Description: Changes the bio paragraph in the student profile

Inputs: String text

Returns: None

changePassword(String password)

Description: Changes the password of the student account

Inputs: String password

Returns: None

Required Interface

db.setstudent(Student student)

Description: Setter method for student into database

Inputs: Student student

Returns: none

db.getStudent(String studentID)

Description: Getter method for student from database

Inputs: String Student ID

Returns: Student object

Student Request Module

Purpose of Module

The purpose of this module is to search for potential friends and have the ability to add friends that you are interested in.

Provided Interface

sendRequest(Student newFriend)

Description: Sends a request from the student to another student or mentor.

Inputs: Student newFriend

Returns: None

acceptRequest()

Description: Allows the user to accept or reject the request

Inputs: None

Returns: True if accept false if reject

searchStudentByName(String studentName)

Description: Allows the student user to search for new students by name to follow.

Inputs: String studentName

Returns: None

searchStudentByInfo(String interests)

Description: Allows the student user to search for new students by their interest to follow.

Inputs: String interests

Returns: None

Required Interface

db.setNewFollower(StudentID)

Description: Setter for new follower into database

Inputs: StudentID

Returns: none

__**Mentor Module**__

Create New Mentor Module

Purpose of Module

The purpose of this module is to create an account for a new mentor who is joining the system.

Provided Interface

createsMentor(Mentor mentor)

Description: Allows user to create new mentor profile

Inputs: mentor

Returns: none

editsMentor(mentor)

Description: Allows mentor to edit their mentor profile

Inputs: Mentor mentor

Returns: none

deleteMentor(mentorID)

Description: Allows mentor to delete their profile

Inputs: Mentor ID

Returns: none

Required Interface

db.setMentor(mentor)

Description: Setter for mentor profile into database

Inputs: Mentor

Returns: none

db.getMentor(mentorID)

Description: Getter for mentor profile from database

Inputs: Mentor ID

Returns: Mentor profile

Edit Mentor Profile Module

Purpose of Module

The purpose of this module is to allow mentors to make changes to their profile page on F=MA

Provided Interface

changeProfilePicture(String image)

Description: Changes the profile picture of the mentor account

Inputs: String image

Returns: None

changeBioParagraph(String text)

Description: Changes the bio paragraph in the mentor profile

Inputs: String text

Returns: None

Required Interface

db.getMentor(MentorID)

Description: Getter for mentor profile from database

Inputs: MentorID

Returns: Mentor profile

Mentor Request Module

Purpose of Module

The purpose of this module is to allow mentors to search for students and then send a request for them to connect.

Provided Interface

searchStudentByName(String studentName)

Description: allows mentor to search students by their name

Inputs:String studentName

Returns: none

searchStudentByInterest(String interest)

Description: allows mentor to search students by their interests

Inputs:String interest

Returns: none

acceptRequest()

Description: Allows mentor to accept or reject request

Inputs: None

Returns: True if accept false if reject

Required Interface

db.setNewMentee(StudentID)

Description: Setter for student mentor connection

Inputs:StudentID

Returns: none

db.getMenteeList(Mentee<List>)

Description: Getter from database a list of students the mentor is connected with

Inputs: Mentee<List>

Returns: List of students

<u>**Post Module**</u>

Caption Module

Purpose of Module

The purpose of this module is to create the caption for a post that a student or mentor can upload onto their account. This post will then be displayed onto the feed of the student or mentor's followers.

Provided Interface

writeCaption(String text)

Description: Allows user to write caption onto a post

Inputs: String text

Returns: None

readCaption()

Description: Allows user to read caption from a post

Inputs: None

Returns: Caption

editCaption(String ID, String text)

Description: Allows user to edit their caption from a previous post

Inputs: String ID, String text

Returns: none

Required Interface

db.setCaption(caption, picture)

Description: Setter for caption into database

Inputs: caption, picture

Returns: None

db.getCaption(postID)

Description: Getter for caption from database

Inputs: postID

Returns: Caption

Picture Module

Purpose of Module

The purpose of this module is to upload a picture that is to be attached to the post that the student creates. The picture can be either taken at that moment using the camera on the phone or searched from the photo gallery on the phone.

Provided Interface

insertPicture(String pic)

Description: Allows user to put a picture onto a post

Inputs: String pic

Returns: None

deletePicture(String pic)

Description: Allows user to delete a picture from a post

Inputs: String pic

Returns: None

viewPicture()

Description: Allows user to view the picture from a post

Inputs: None

Returns: Picture

Required Interface

db.setPic(caption, picture)

Description: Setter for picture into ID

Inputs: caption, picture

Returns: None

db.getPic(picID)

Description: Gets picture from database

Inputs: picID

Returns: Picture

Comment Module

Purpose of Module

The purpose of this module is to allow other students to comment on the posts on their feed. There is functionality to view their comments and even edit or delete comments.

Provided Interface

writeComment(String text, String postID)

Description: Allows user to write a comment on a post

Inputs: String text, String postID

Returns: None

deleteComment(String commentID)

Description: Deletes comment on a post

Inputs: String commentID

Returns: None

Required Interface

db.setComment(String comment, Post postID)

Description: Setter for comment into database

Inputs: String comment, Post postID

Returns: None

db.getComment(commentID)

Description: Getter for comment from database

Inputs: CommentID

Returns: the comment

<u>Forum Module</u>

Thread Module

Purpose of Module

The purpose of this module is to create a new thread on the forums. This is for students who have a question or comment that has never been asked before and as a result are creating a new thread.

Provided Interface

createThread(String title)

Description: Creates the thread to allow users to write on forums.

Inputs: String title

Returns: none

Required Interface

db.setThread(String title)

Description: Setter for thread into database

Inputs: String title

Returns: None

db.getThread()

Description: Getter for thread from database

Inputs: none

Returns: Thread

Forum Subject Module

Purpose of Module

The purpose of this module is to create a new forum under the title or subject that was just created.

Provided Interface

writeForum(String text)

Description: Allows the user to write to the forum

Inputs: Sting text

Returns: none

readForum()

Description: Allows the user to read from the forum

Inputs: None

Returns: Forum

editForum(String text, String ForumID)

Description: Allows the user to edit their forum
Inputs: String text, String ForumID
Returns: none

Required Interface

db.setForum(String forum)

Description: Setter for forum from database
Inputs: String Forum
Returns: none

db.getForum()

Description: Getter for Forum from database
Inputs: String Forum
Returns: Forum

Abstract Data Types

Administrator Module	Type
name	String
userID	String
lastName	String
gradYear	Integer
collegeName	String
emailAddress	String
accepted	boolean
Student Module	Type
name	String
userID	String
lastName	String
gradYear	Integer
collegeName	String
emailAddress	String

accepted	boolean
Mentor Module	Type
name	String
userID	String
lastName	String
gradYear	Integer
collegeName	String
emailAddress	String
accepted	boolean
Post Module	Type
creator	String
picture	String
bodyText	String
datePosted	Date
numLikes	Integer
Likes	String[ ]
Forum Module	Type
Subject	String
bodyText	String
Comments	String[ ]
Picture	String
Caption	String
Author	String

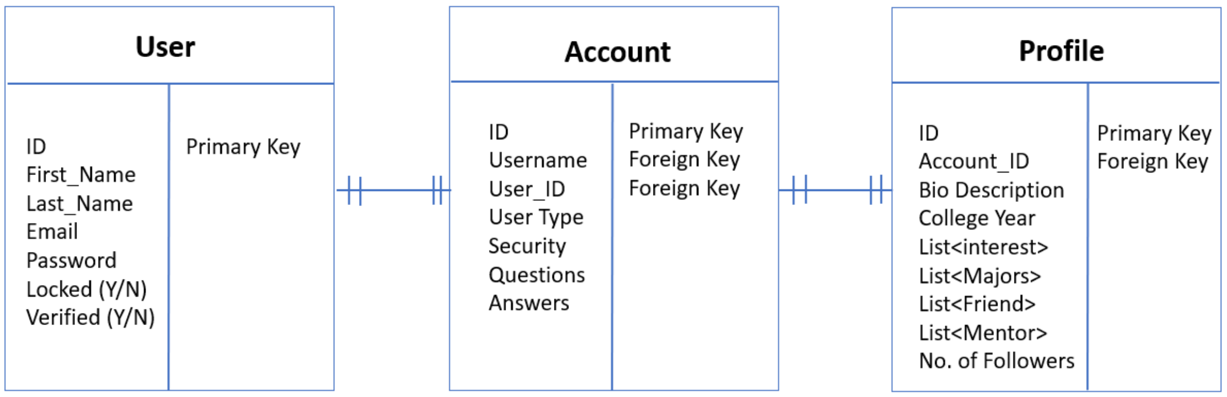
Database Designs

The database schema for a social media mobile application is designed to logically structure the information entered into the app by the users. F=MA uses databases to securely store the account information of our users and to link their accounts to their online profiles, as well as to ensure that the users are verified college students. The design was chosen to monitor the online activity of our users and their interaction with other members. Certain tables such as Forum, and Request handle the conversation logs between users via posts, comments and a list of contributors. The Request table handles the friend request feature of the application where a student can send a friend request to either a friend or mentor.

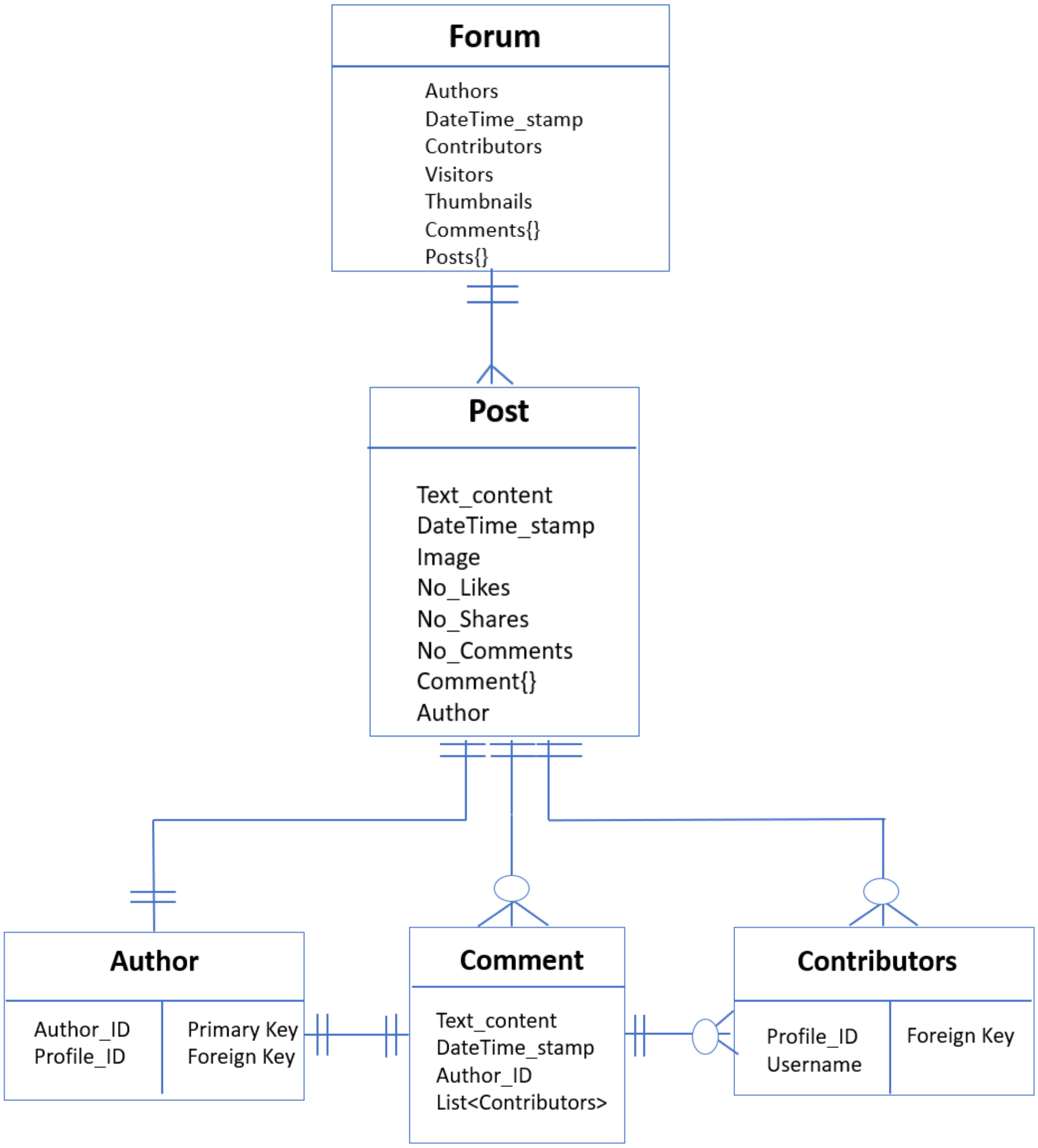
This database schema should:

- ➔ Organize the user's account and profile information
- ➔ Organize the functionality of the application's features
- ➔ Organize the user's online activity
- ➔ Reduce data redundancy

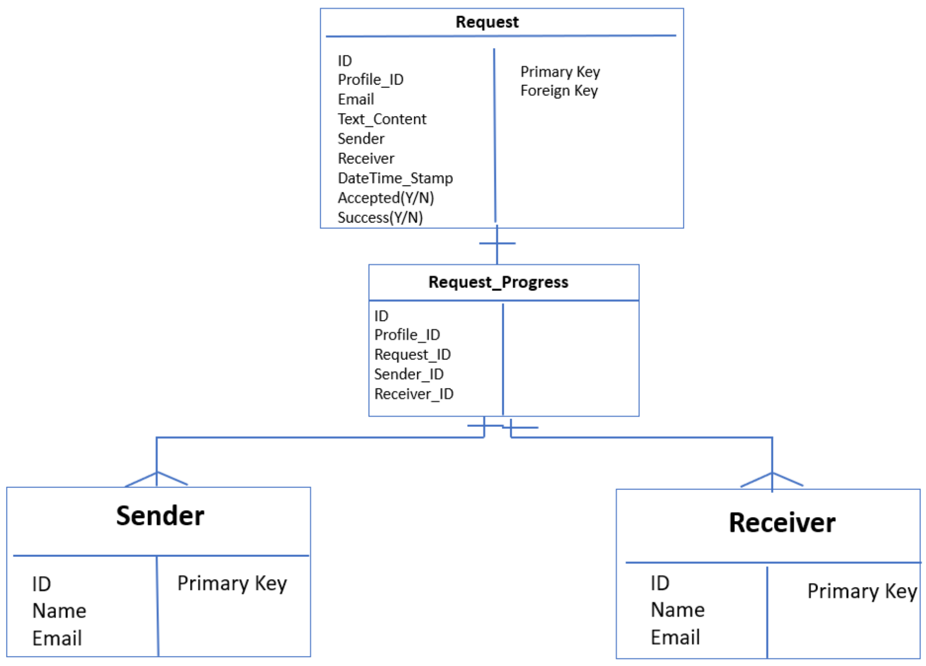
User Details:



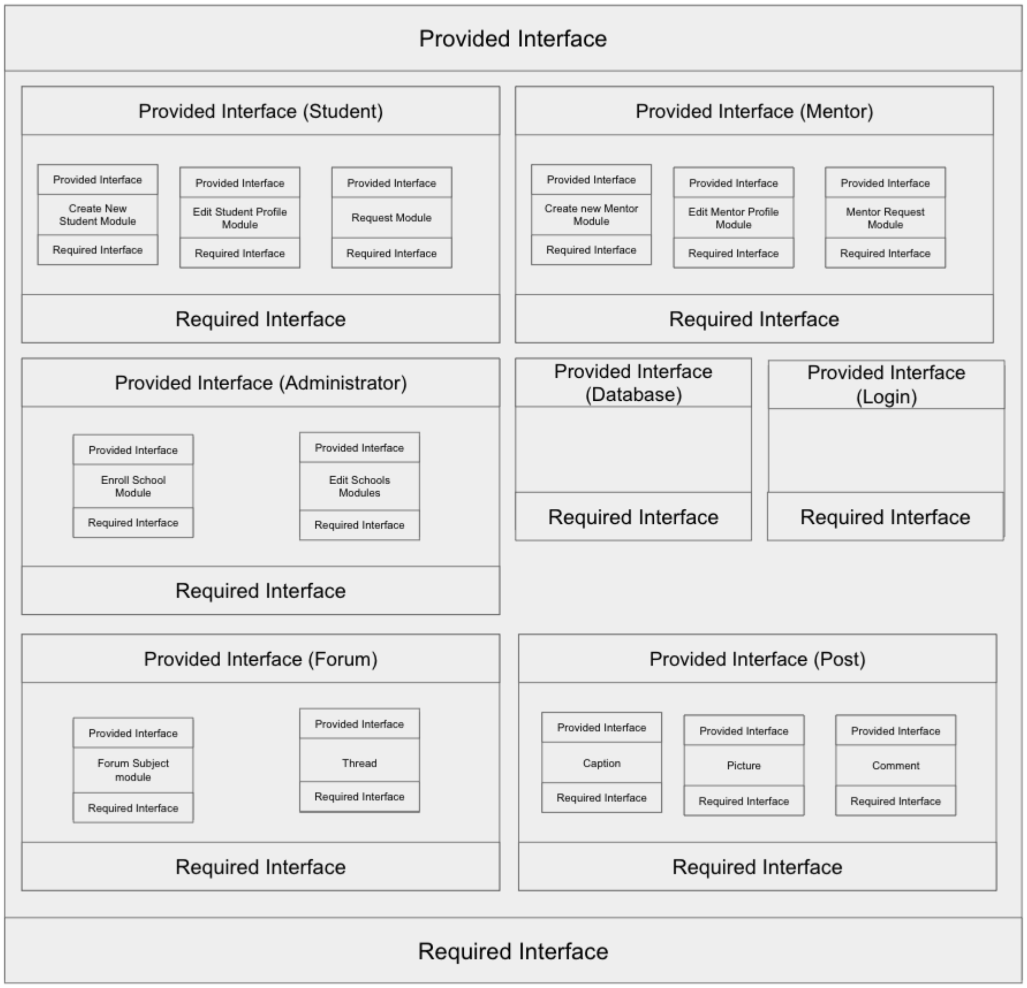
Database Table(s) for Forum:



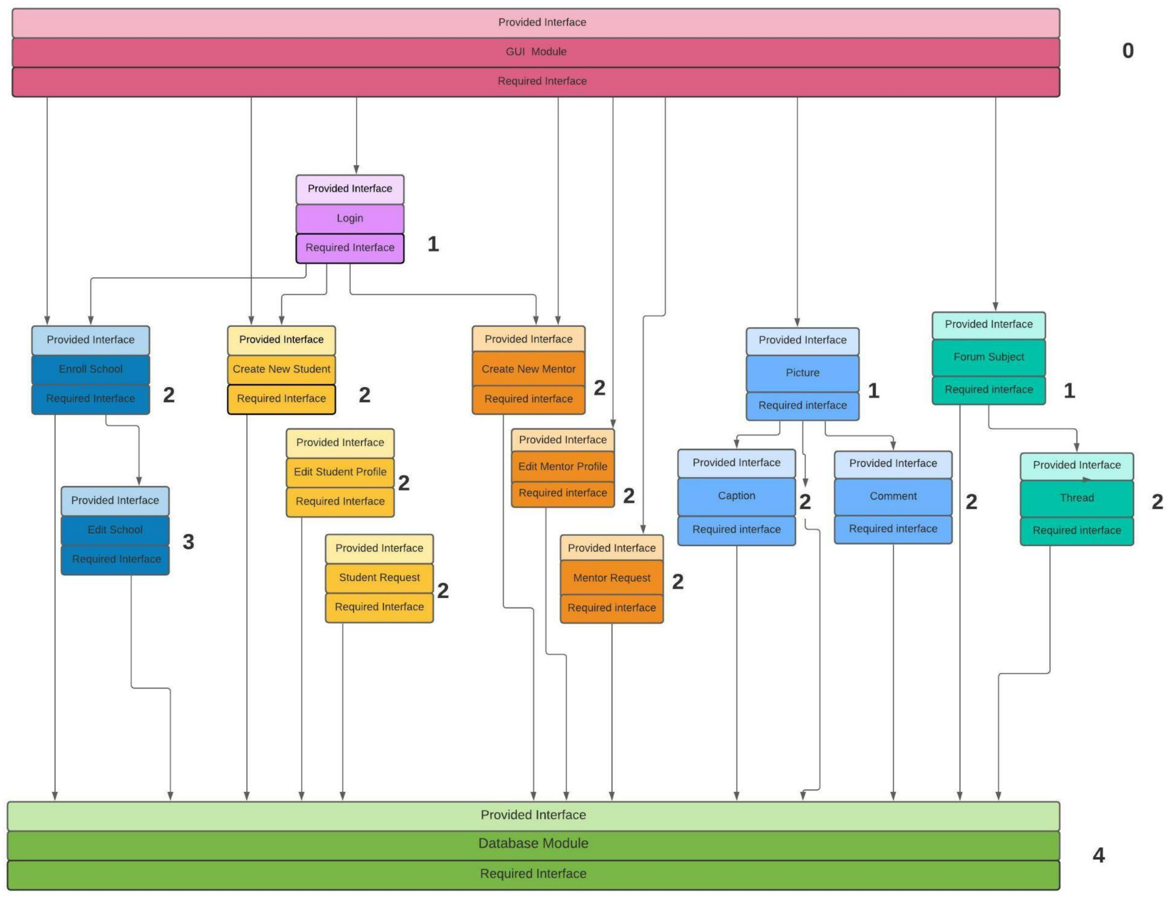
Database Table(s) for Friend/Mentor Request:



COMPRISES Diagram

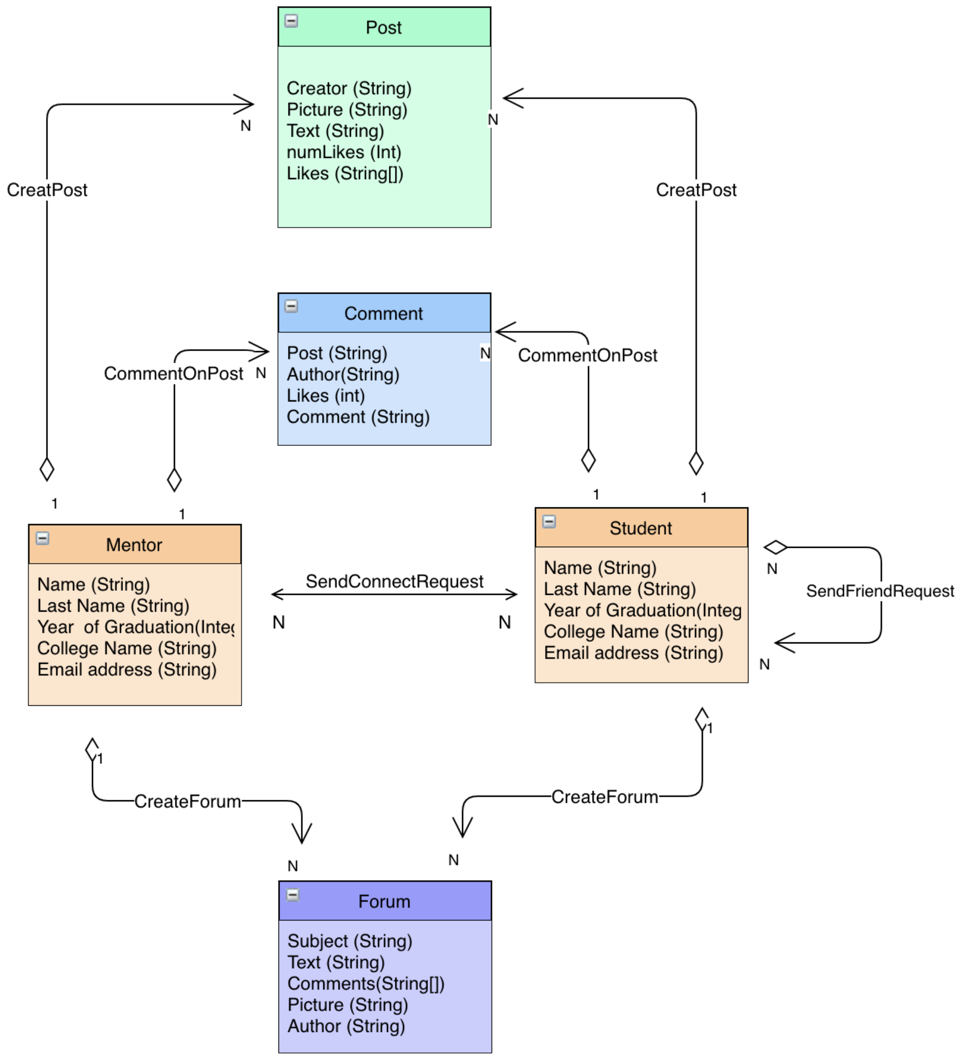


USES Diagram



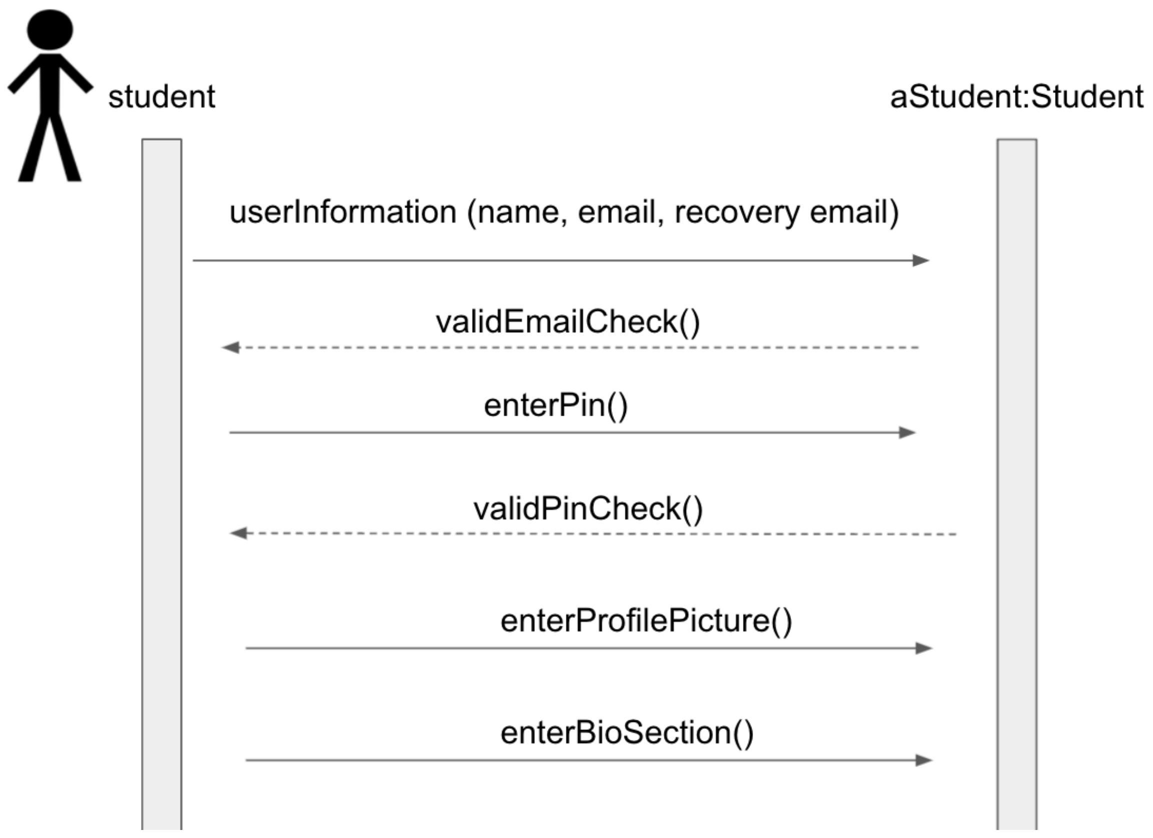
Other diagrams (Behavioral/ Structural modeling)

1. **Class Diagram**

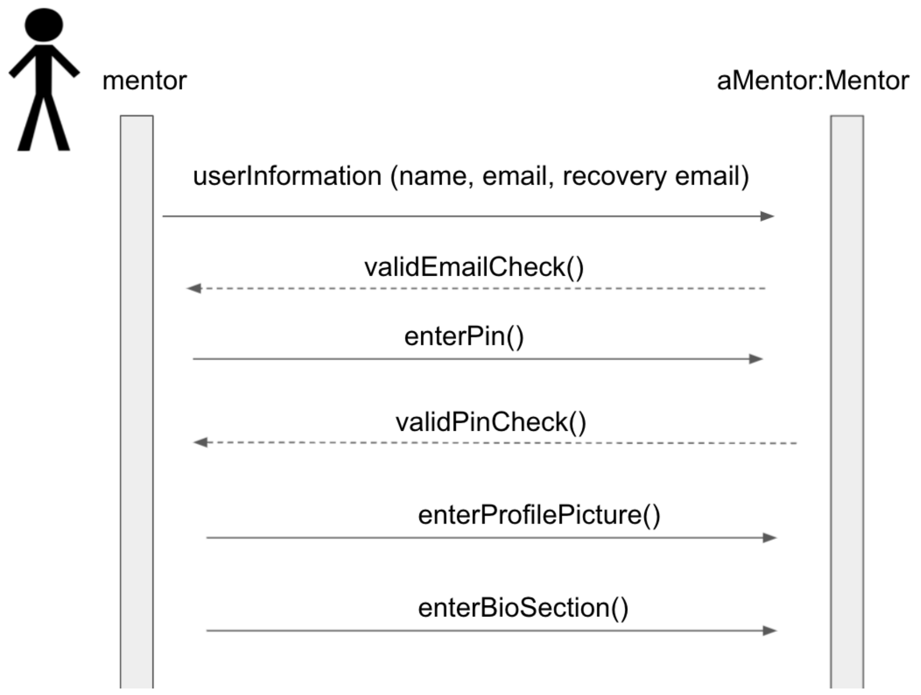


2. Sequence Diagrams

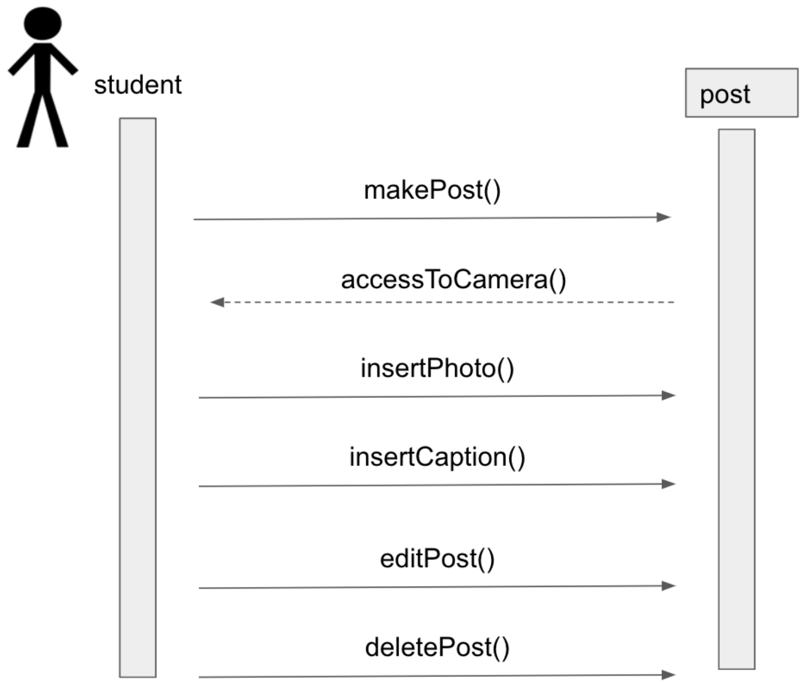
Sequence Diagram : Registration of New Student



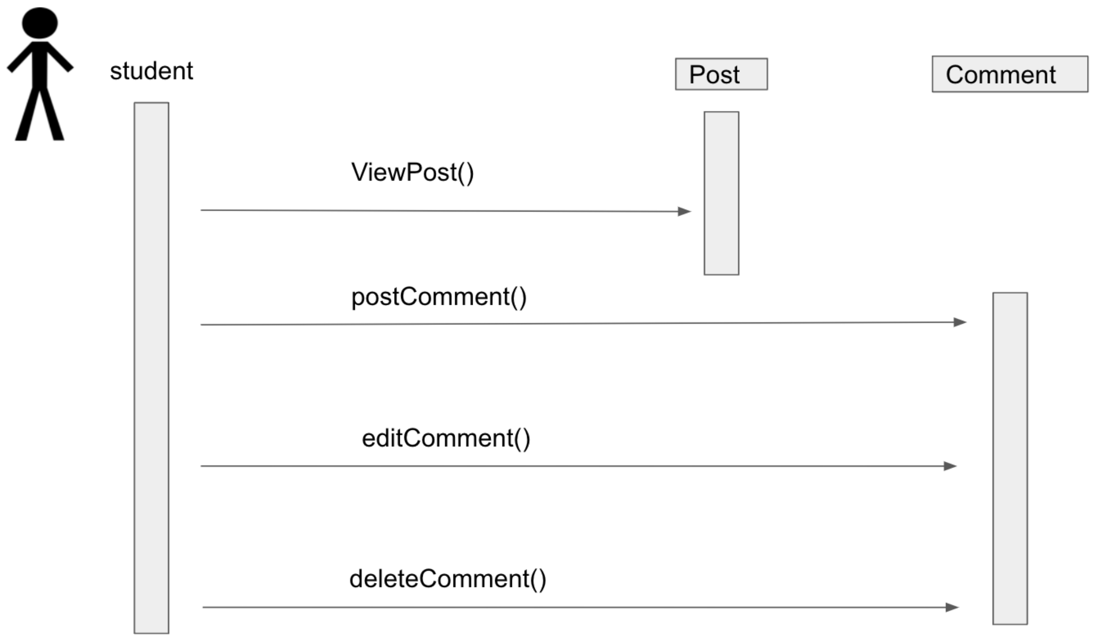
Sequence Diagram : Registration of New Mentor



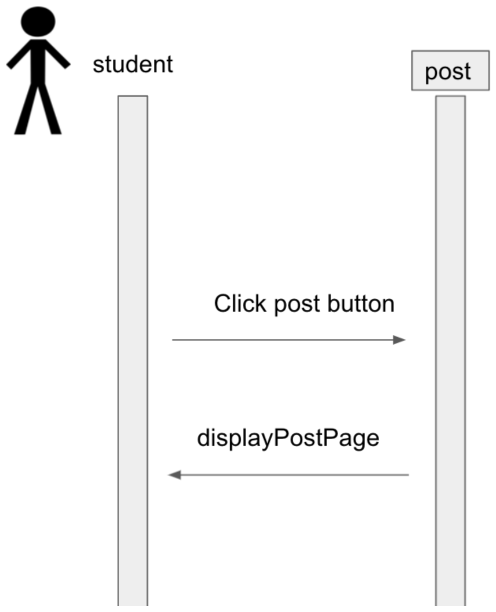
Sequence Diagram : Making a Post



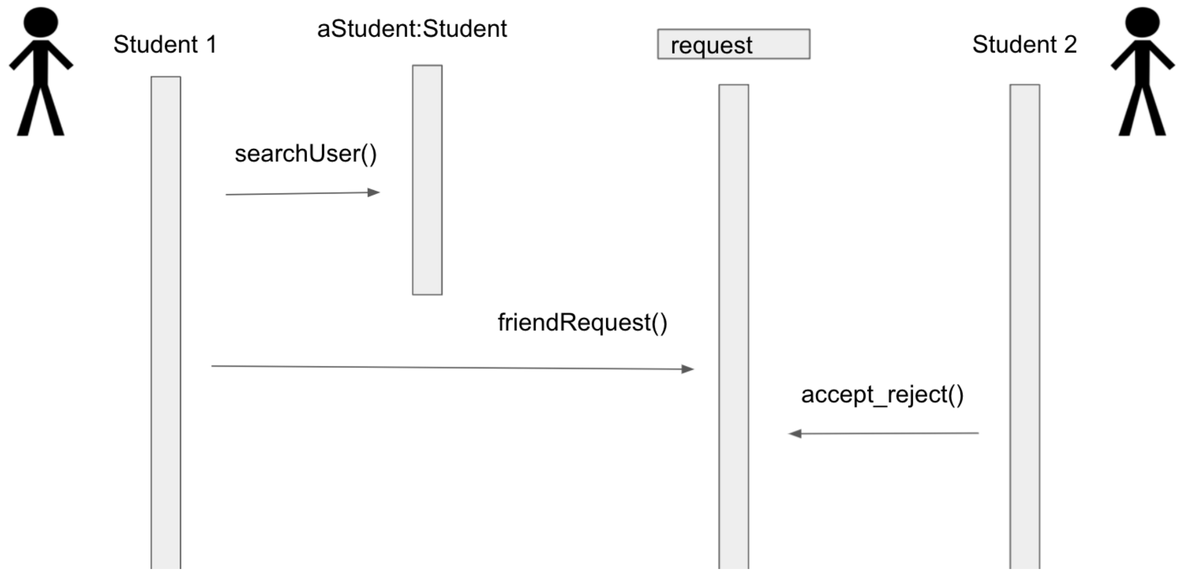
Sequence Diagram : Commenting a Post



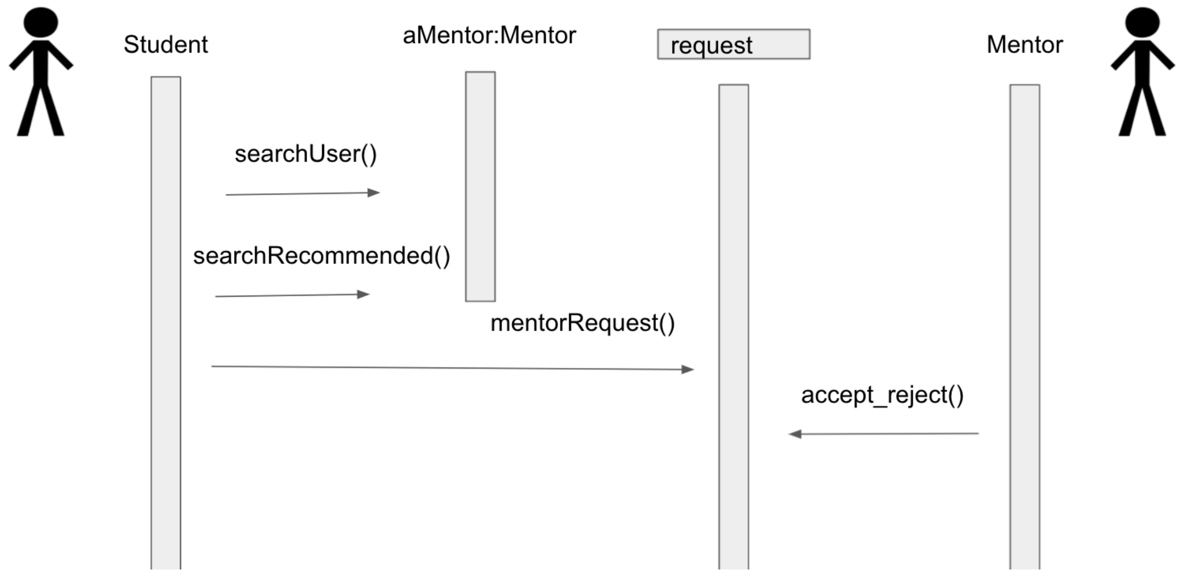
Sequence Diagram : Viewing Students Post



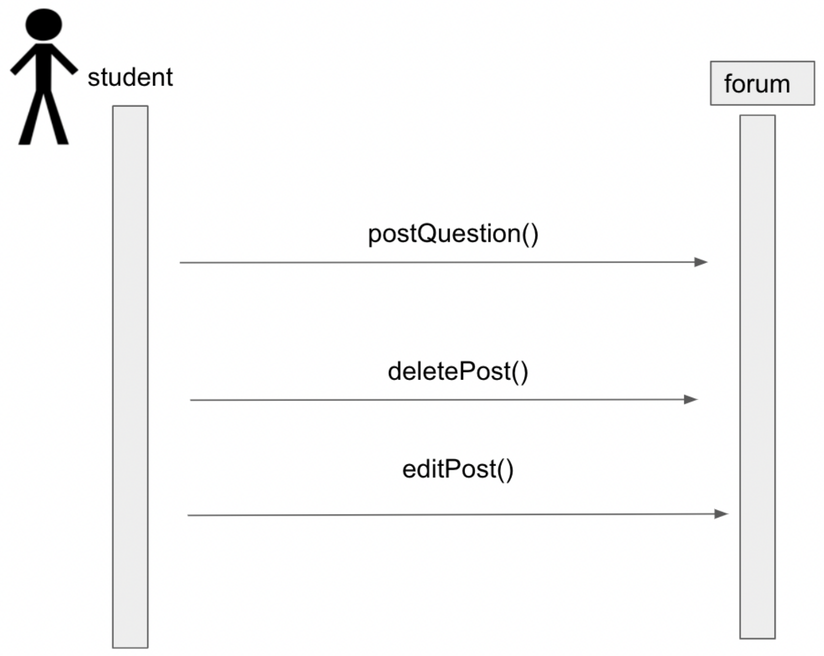
Sequence Diagram : Adding Friends



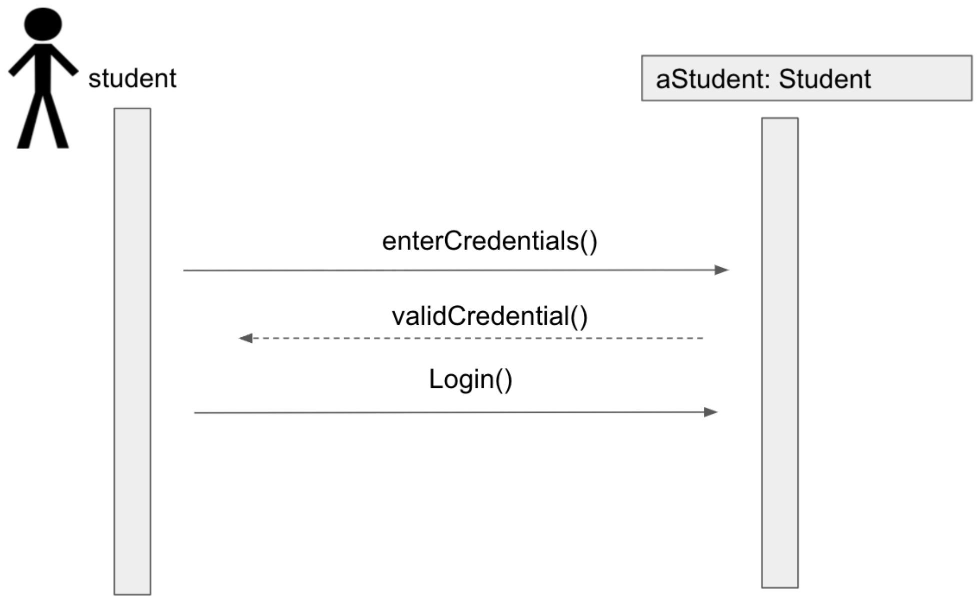
Sequence Diagram : Adding Mentor



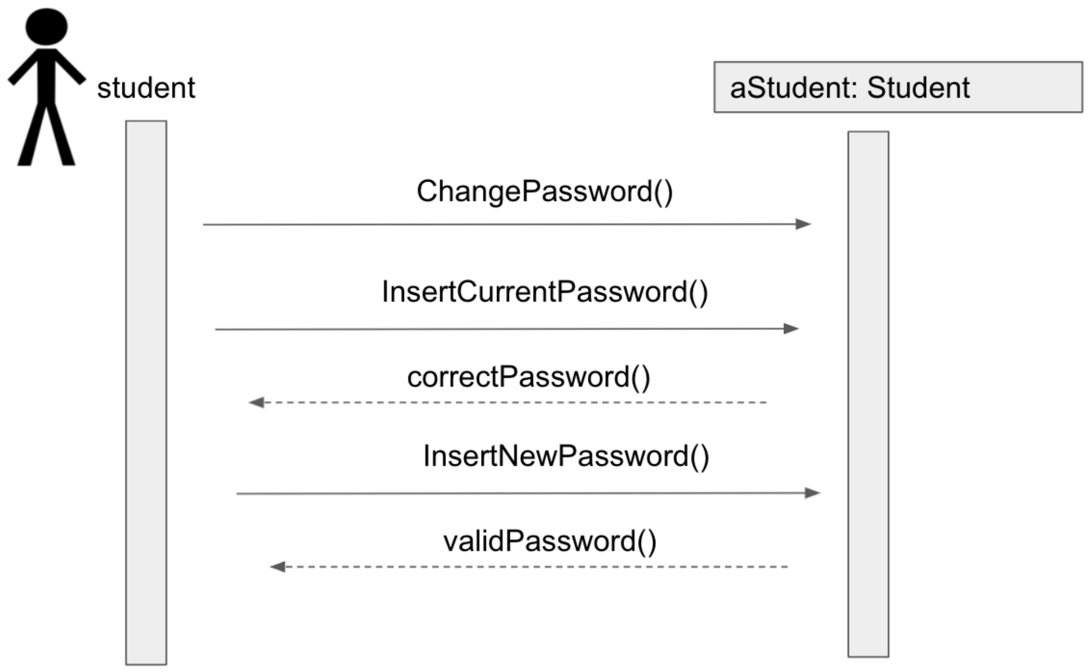
Sequence Diagram : Asking a question on the forum



Sequence Diagram : Login

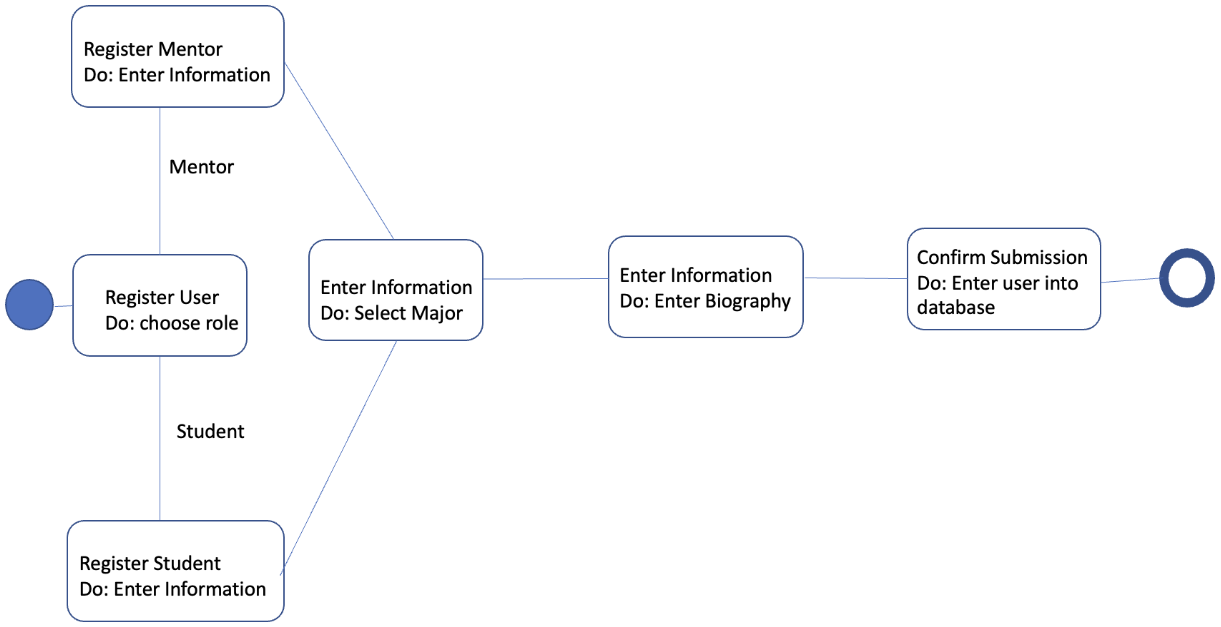


Sequence Diagram : Change Password

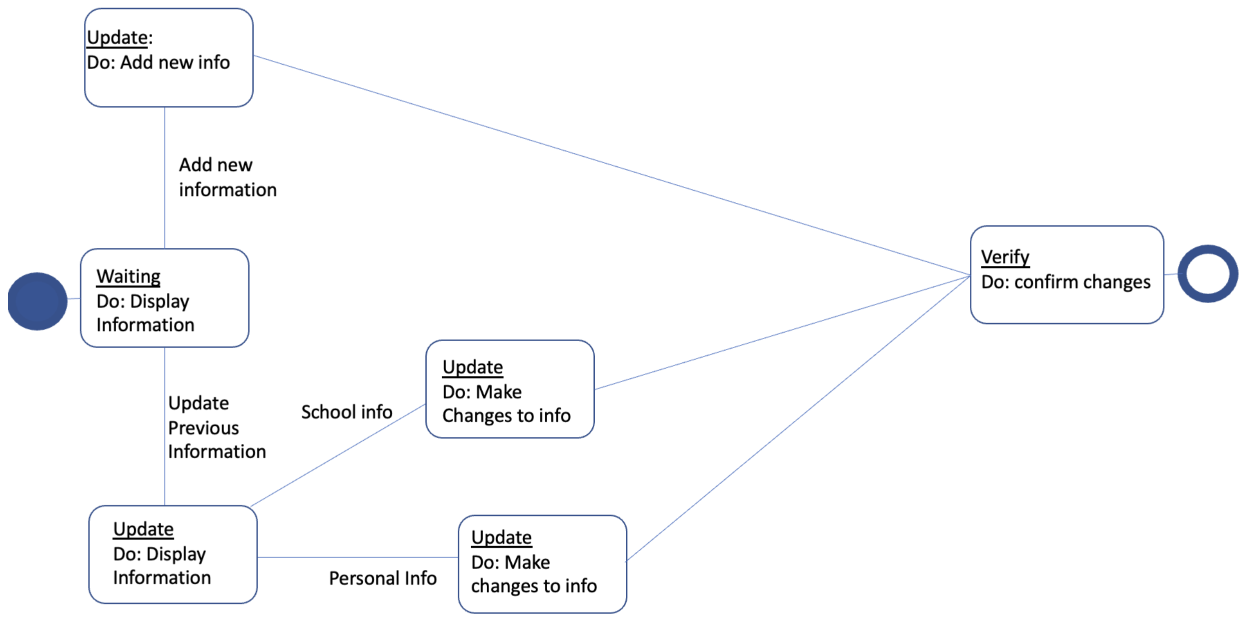


3. State Diagrams

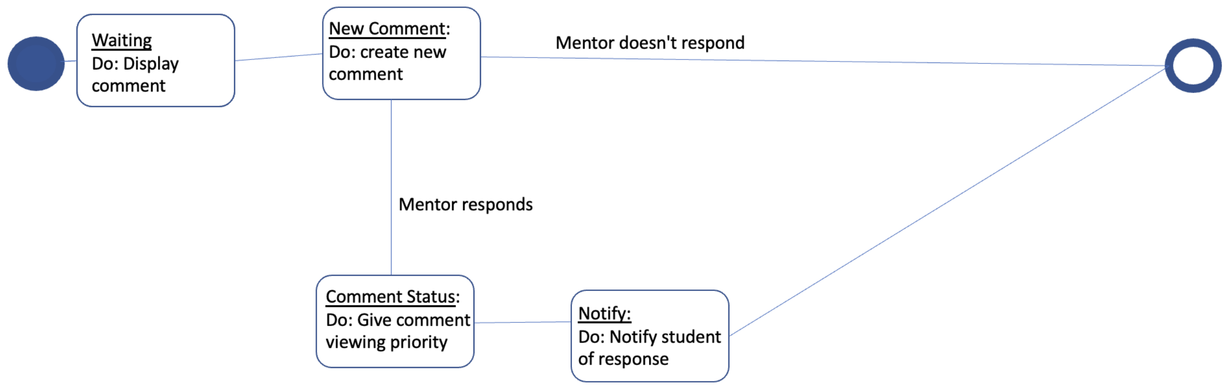
Register a User



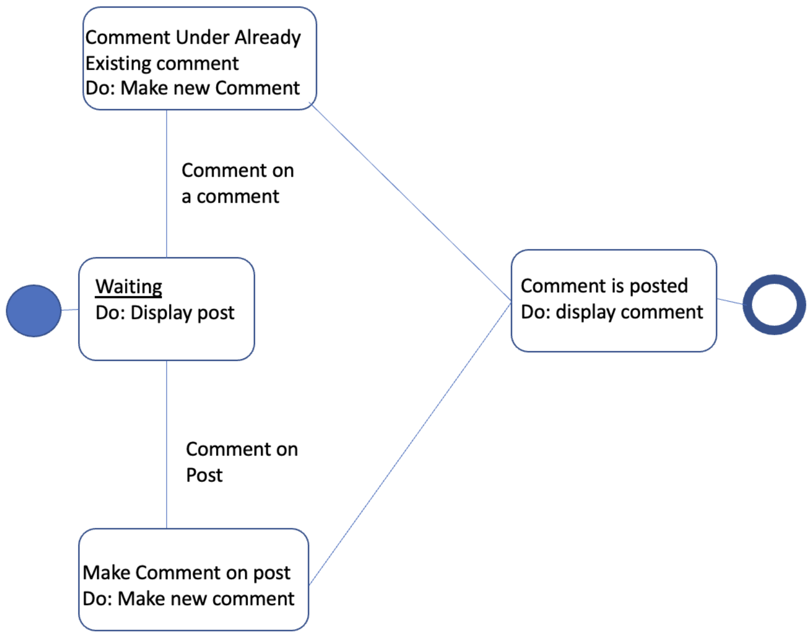
Update Student Info



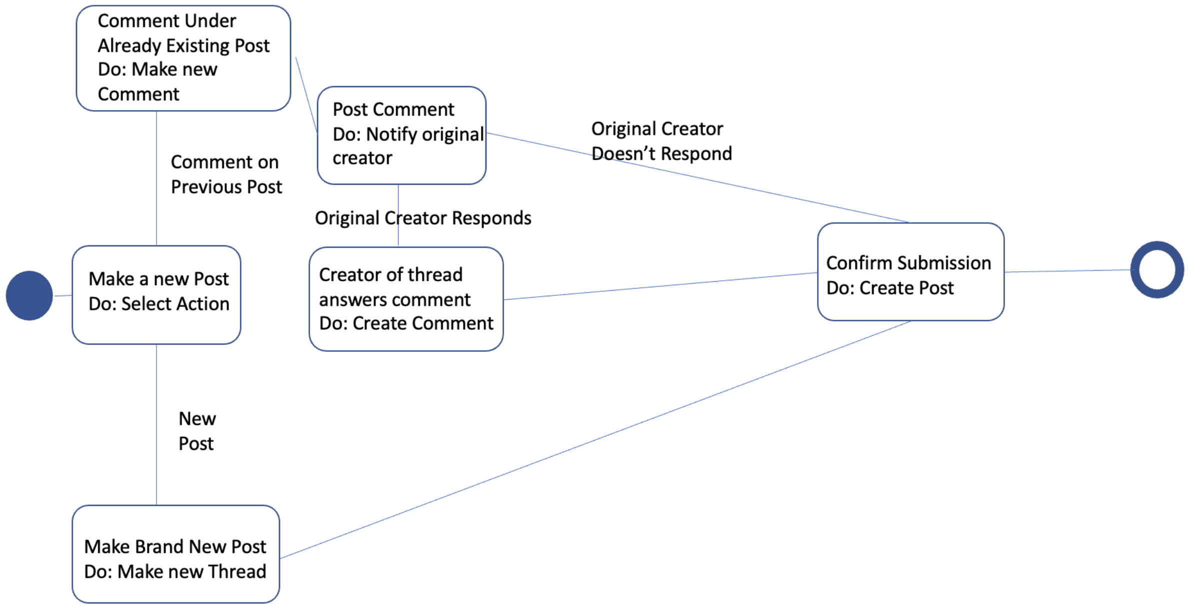
Adding Response to Comment



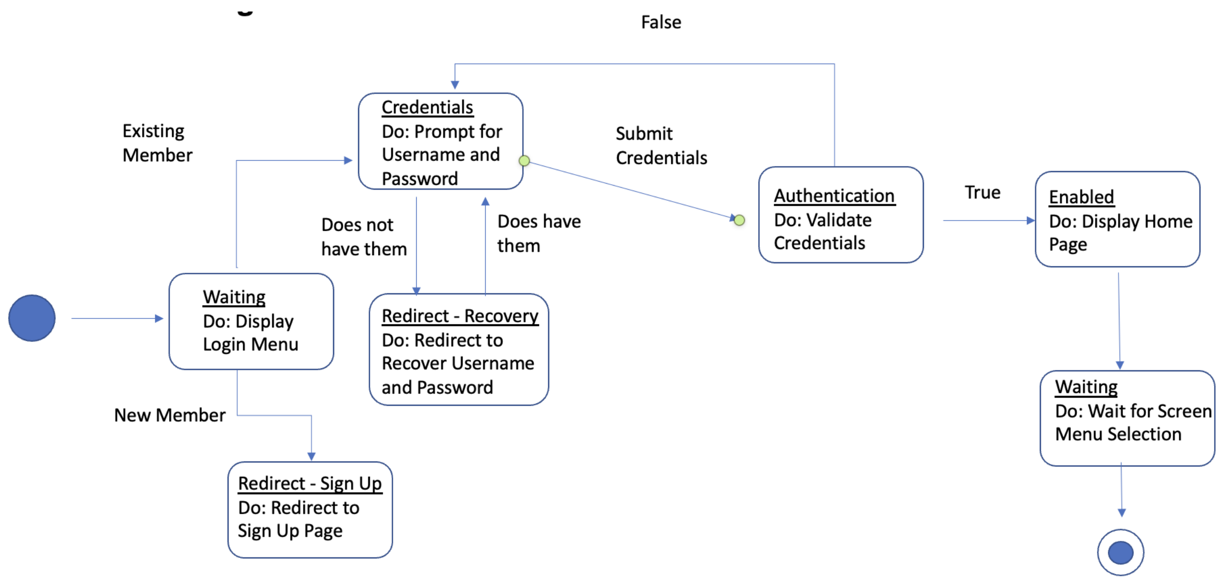
Comment on a post


Comment Under Already Existing comment Do: Make new Comment
Comment on a comment
Waiting Do: Display post
Comment on Post
Make Comment on post Do: Make new comment
Comment is posted Do: display comment

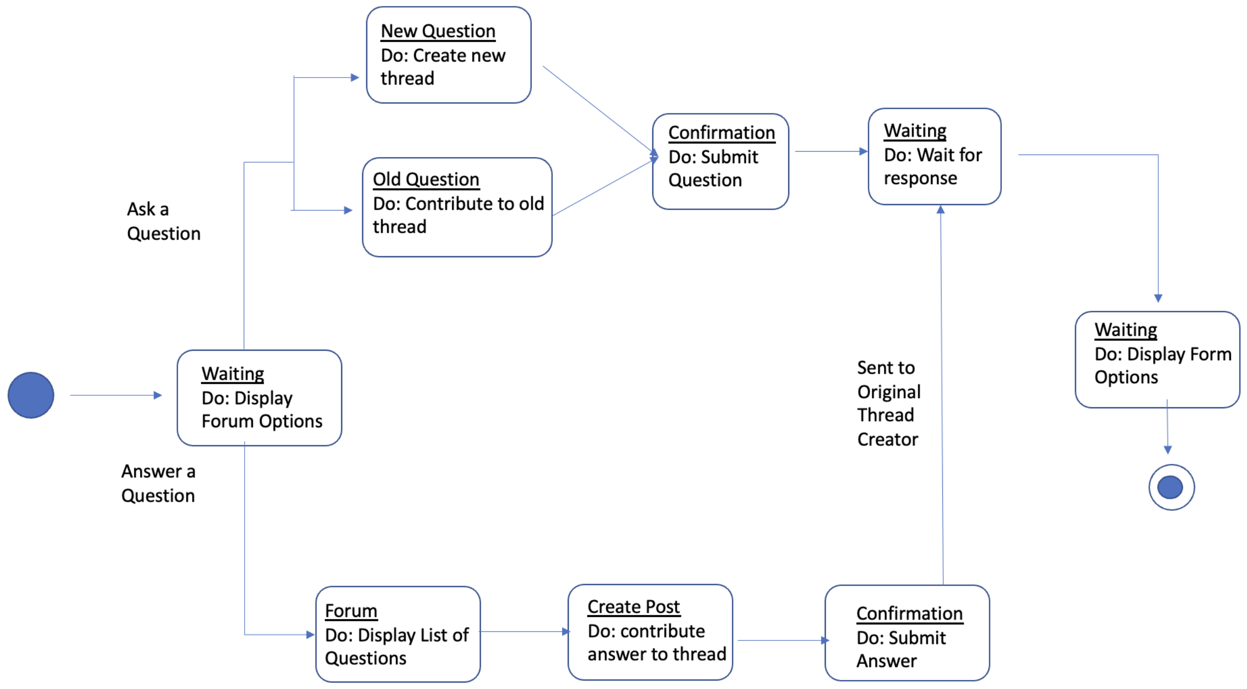
Make a Post


Comment Under Already Existing Post Do: Make new Comment
Comment on Previous Post
Post Comment Do: Notify original creator
Original Creator Doesn't Respond
Original Creator Responds
Make a new Post Do: Select Action
Creator of thread answers comment Do: Create Comment
Confirm Submission Do: Create Post
New Post
Make Brand New Post Do: Make new Thread

Account Login

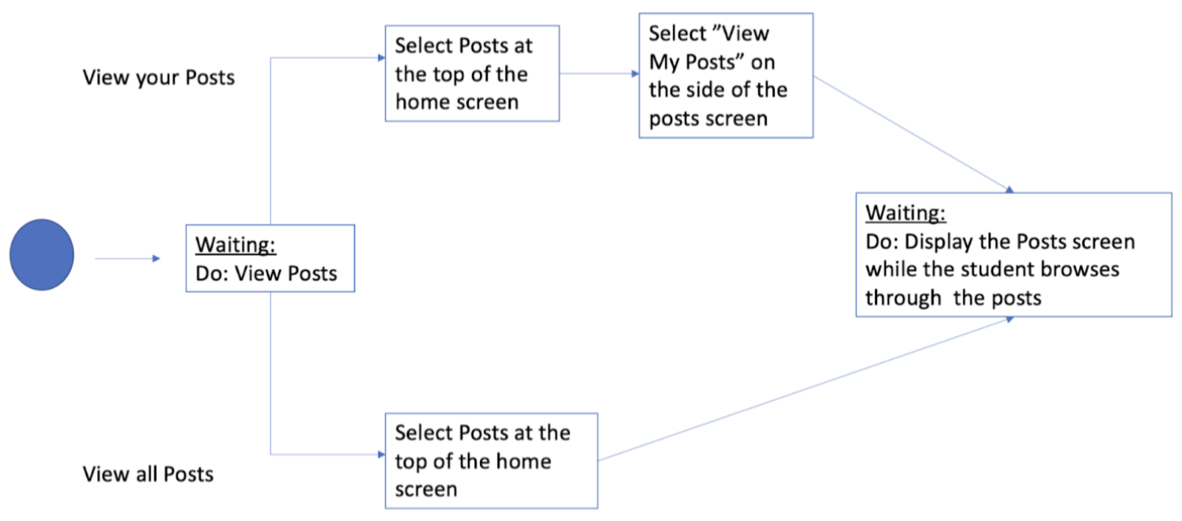


Discussion Board



Viewing a Students Post

Viewing a Students Post


View your Posts
Select Posts at the top of the home screen
Select "View My Posts" on the side of the posts screen
Waiting: Do: View Posts
Waiting: Do: Display the Posts screen while the student browses through the posts
View all Posts
Select Posts at the top of the home screen

Integration Test Plan

Method: Bottom Up testing

Reasoning: Because our service F=MA has already been broken down into many modules, we will employ bottom up testing to create these modules and connect them along the way. This way we will have very easy and simple to create test conditions along with an ease to observe test results. Additionally, our platform will have a lot of code reuse when we begin expanding to other college majors so we believe that the bottom up testing will make this easy to implement into the expanded major platforms later on.

Integration Group	**Drivers Needed**	**Stub(s)**
Phase I: Database Subsystem		
Database Information	Server Driver	None

Phase II: Data Entity Subsystem		
Admin Module	Data Driver	None
Student Module	Data Driver	None
Mentor Module	Data Driver	None
Post Module	Data Driver	None
Forum Module	Data Driver	None
Phase III: GUI Module		
Display Control	Display Driver	None

Driver Definitions:

Data Driver: Prints the retrieved data from the database to the screen

Display Driver: Displays the module and data

Student Modules Testing:

Test Case	Successful Login
Test case ID	1
Description	Check results on entering valid User Id & Password If no error User will be logged in successfully. User here has entered a valid user Id and Password
Input	Enter username and password
Output	Logged into account when input credentials are accepted
What is being tested?	Testing for the correct username and password combination from our database.
Testing constraints	Username and password of student logging in

Test Case	Username already in use
Test case ID	2
Description	This test case checks to make sure that the username of a student is not already in use. This is to prevent multiple people from having the same username.

Input	Username of student
Output	Error message saying that username is already taken; successful creation of username if it is original
What is being tested?	Username uniqueness to make sure there are no duplicates in the database
Testing constraints	Username of the student must be original

Test Case	Correct password on login
Test case ID	3
Description	This test case checks to ensure the correct password associated with that username on login
Input	Username and password of the student
Output	If the password is correct the student is brought to their profile page; if the password is incorrect error messages is displayed showing that the student must enter a correct password
What is being tested?	The correctness of the password associated with the username of an account
Testing constraints	Username and password must be correct

Test Case	Invalid Password
Test case ID	4
Description	This test case ensures that the student enters a password that fulfills the password security requirements
Input	The password that is created, along with the password used for the confirmation of the password.
Output	If the password meets the security requirements, the password is entered into the database, if the password does not meet the requirements, the message “Password must be at least 8 characters long and use one special character (!@#$%*)”
What is being tested?	The password that is created for the account meets the security requirements.
Testing constraints	Password must match the security requirements.

Mentor Modules Testing:

Test Case	Successful Login

Test case ID	1
Description	Check results on entering valid User Id & Password If no error User will be logged in successfully. User here has entered a valid user Id and Password
Input	Enter username and password
Output	Logged into account when input credentials are accepted
What is being tested?	Testing for the correct username and password combination from our database.
Testing constraints	Username and password of mentor logging in

Test Case	Username already in use
Test case ID	2
Description	This test case checks to make sure that the username of a mentor is not already in use. This is to prevent multiple people from having the same username.
Input	Username of mentor
Output	Error message saying that username is already taken; successful creation of username if it is original
What is being tested?	Username uniqueness to make sure there are no duplicates in the database
Testing constraints	Username of the student must be original

Test Case	Correct password on login
Test case ID	3
Description	This test case checks to ensure the correct password associated with that username on login
Input	Username and password of the mentor
Output	If the password is correct the mentor is brought to their profile page; if the password is incorrect error messages is displayed showing that the student must enter a correct password
What is being tested?	The correctness of the password associated with the username of an account
Testing constraints	Username and password must be correct

Test Case	Invalid Password

Test case ID	4
Description	This test case ensures that the mentor enters a password that fulfills the password security requirements
Input	The password that is created, along with the password used for the confirmation of the password.
Output	If the password meets the security requirements, the password is entered into the database, if the password does not meet the requirements, the message “Password must be at least 8 characters long and use one special character (!@#$%*)”
What is being tested?	The password that is created for the account meets the security requirements.
Testing constraints	Password must match the security requirements.

Posts Modules Testing:

Test Case	Posts viewable to students and mentors
Test case ID	1
Description	This test case ensures that the posts are readable from the database and is viewable to students and mentors that follow the student with the original post
Input	The post ID that is generated when the post is created
Output	The post on the screen of the student or mentor viewing it
What is being tested?	The readability of the post from the database
Testing constraints	Database must be readable

Test Case	Comments on posts under the character limit
Test case ID	2
Description	This test case tests to make sure that the character limit is under the limit that is established by the app. (200 max char.)
Input	The comment text
Output	Success message if under the limit
What is being tested?	The comment being under the character limit
Testing constraints	The length of the comment

Test Case	Inserting comments into the database
Test case ID	3
Description	This test case ensures that comments are properly inserted into the database.

Input	The comment
Output	Success message that the comment is in the database or failure message if not
What is being tested?	The writing of data to the database
Testing constraints	Data is writable to the database

Forum Modules Testing:

Test Case	Forum viewable to students and mentors
Test case ID	1
Description	This test case ensures that the Forums are readable from the database and is viewable to students and mentors that follow the student with the original post
Input	The forum ID that is generated when the forum is created
Output	The forum on the screen of the student or mentor viewing it
What is being tested?	The readability of the forum from the database
Testing constraints	Database must be readable

Test Case	Comments on forum under the character limit
Test case ID	2
Description	This test case tests to make sure that the character limit is under the limit that is established by the app. (200 max char.)
Input	The comment text
Output	Success message if under the limit
What is being tested?	The comment being under the character limit
Testing constraints	The length of the comment

Test Case	Inserting comments into the database
Test case ID	3
Description	This test case ensures that comments are properly inserted into the database.
Input	The comment
Output	Success message that the comment is in the database or failure message if not
What is being tested?	The writing of data to the database
Testing constraints	Data is writable to the database

GUI Modules Testing

Test Case	GUI Testing
Test case id	1
Description	User tests on Internet Explorer
Input	Searches documents
Output	Results returned
What is being tested?	Browser compatibility
Testing credentials	Open a Ie and search

Test Case	GUI Testing
Test case id	2
Description	User tests from Google Chrome
Input	Searches documents
Output	Results returned
What is being tested?	Browser compatibility
Testing credentials	Open a Chrome and search

Test Case	GUI Testing
Test case id	3
Description	User tests from Safari
Input	Searches documents
Output	Results returned
What is being tested?	Browser compatibility
Testing credentials	Open a Safari and search

Test Case	GUI Testing
Test case id	4
Description	User tests on iOS device App
Input	Searches documents
Output	Results returned
What is being tested?	Browser compatibility
Testing credentials	Open a Ie and search

Test Case	GUI Testing

Test case id	5
Description	User tests on Android app store app
Input	Searches documents
Output	Results returned
What is being tested?	Browser compatibility
Testing credentials	Open a Chrome and search