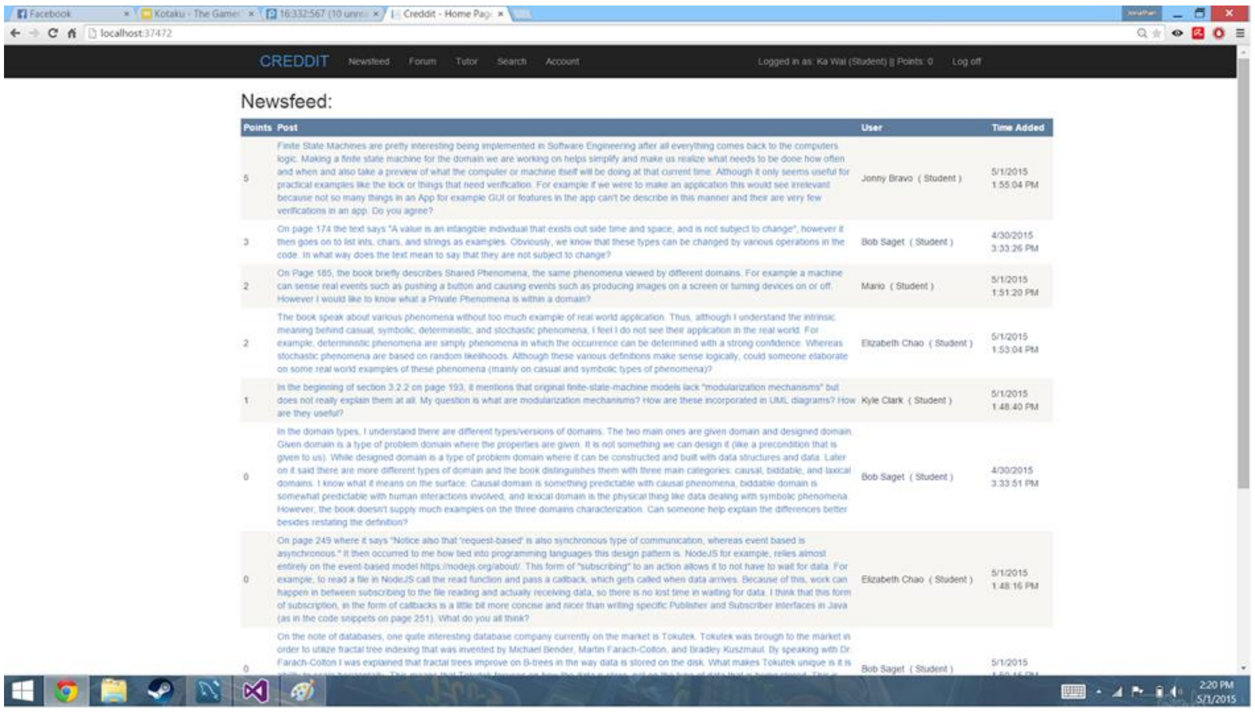
Educational Networking Tool for College Students



Newsfeed:

Points	Post	User	Time Added
5	Finite State Machines are pretty interesting being implemented in Software Engineering after all everything comes back to the computers logic. Making a finite state machine for the domain we are working on helps simplify and make us realize what needs to be done how often and when and also take a preview of what the computer or machine itself will be doing at that current time. Although it only seems useful for practical examples like the lock or things that need verification. For example if we were to make an application this would see irrelevant because not so many things in an App for example GUI or features in the app can't be describe in this manner and their are very few verifications in an app. Do you agree?	Jonny Bravo (Student)	5/1/2015 1:55:04 PM
3	On page 174 the text says "A value is an intangible individual that exists out side time and space, and is not subject to change", however it then goes on to list ints, chars, and strings as examples. Obviously, we know that these types can be changed by various operations in the code. In what way does the text mean to say that they are not subject to change?	Bob Saget (Student)	4/30/2015 3:33:26 PM
2	On Page 185, the book briefly describes Shared Phenomena, the same phenomena viewed by different domains. For example a machine can sense real events such as pushing a button and causing events such as producing images on a screen or turning devices on or off. However I would like to know what a Private Phenomena is within a domain?	Mario (Student)	5/1/2015 1:51:20 PM
2	The book speak about various phenomena without too much example of real world application. Thus, although I understand the intrinsic meaning behind casual, symbolic, deterministic, and stochastic phenomena, I feel I do not see their application in the real world. For example, deterministic phenomena are simply phenomena in which the occurrence can be determined with a strong confidence. Whereas stochastic phenomena are based on random likelihoods. Although these various definitions make sense logically, could someone elaborate on some real world examples of these phenomena (mainly on casual and symbolic types of phenomena)?	Elizabeth Chao (Student)	5/1/2015 1:53:04 PM
1	In the beginning of section 3.2.2 on page 193, it mentions that original finite-state-machine models lack "modularization mechanisms" but does not really explain them at all. My question is what are modularization mechanisms? How are these incorporated in UML diagrams? How are they useful?	Kyle Clark (Student)	5/1/2015 1:48:40 PM
0	In the domain types, I understand there are different types/versions of domains. The two main ones are given domain and designed domain. Given domain is a type of problem domain where the properties are given. It is not something we can design it (like a precondition that is given to us). While designed domain is a type of problem domain where it can be constructed and built with data structures and data. Later on it said there are more different types of domain and the book distinguishes them with three main categories: causal, biddable, and laxical domains. I know what it means on the surface. Causal domain is something predictable with causal phenomena, biddable domain is somewhat predictable with human interactions involved, and lexical domain is the physical thing like data dealing with symbolic phenomena. However, the book doesn't supply much examples on the three domains characterization. Can someone help explain the differences better besides restating the definition?	Bob Saget (Student)	4/30/2015 3:33:51 PM
0	On page 249 where it says "Notice also that 'request-based' is also synchronous type of communication, whereas event based is asynchronous." It then occurred to me how tied into programming languages this design pattern is. NodeJS for example, relies almost entirely on the event-based model https://nodejs.org/about/. This form of "subscribing" to an action allows it to not have to wait for data. For example, to read a file in NodeJS call the read function and pass a callback, which gets called when data arrives. Because of this, work can happen in between subscribing to the file reading and actually receiving data, so there is no lost time in waiting for data. I think that this form of subscription, in the form of callbacks is a little bit more concise and nicer than writing specific Publisher and Subscriber interfaces in Java (as in the code snippets on page 251). What do you all think?	Elizabeth Chao (Student)	5/1/2015 1:48:16 PM
0	On the note of databases, one quite interesting database company currently on the market is Tokutek. Tokutek was brough to the market in order to utilize fractal tree indexing that was invented by Michael Bender, Martin Farach-Colton, and Bradley Kuszmaul. By speaking with Dr. Farach-Colton I was explained that fractal trees improve on B-trees in the way data is stored on the disk. What makes Tokutek unique is it is	Bob Saget (Student)	5/1/2015

Report #3 - Specification & Design - Iteration 2

Software Engineering 332:452 - Group 9

https://github.com/kyleru/Creddit

Nathan Del Carmen, Ka Wai Chu, Jonathan Yang,
Elizabeth Chao, Daniel Lee, Kyle Clark, Kyujin Kim

Individual Contribution Breakdowns:

All team members contributed equally.

Report Distrubution	Nathan del Carmen	Elizabeth Chao	Kawai Chu	Kyle Clark	Kyujin Kim	Jon Yang	Daniel Lee	Percentage
Project Management (16 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.1: Interaction Diagrams (30 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.2: Class Diagram and Interface Specification (10 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.3: System Architecture and System Design (15 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.4: Algorithms and Data Structure (4 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.5: User Interface Design and Implemention (11 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.6: Design of Tests (12 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.7: Plan of Work (2 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.8: Refrences (-5 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Total Points that could be aquired				100				
Points Aquired	14.29	14.29	14.29	14.29	14.29	14.29	14.29	100.00

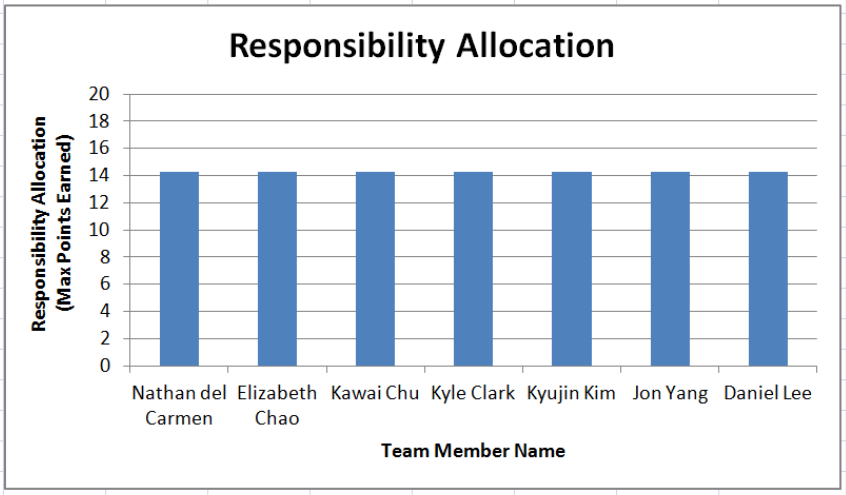


Table of Contents:

Summary of Changes

- Page maker has been deleted and its functionality has incorporated into interface page.
- Part 3 has been added
- Account types
    - Student
        - tutor
    - Faculty
    - Admin
    - Recruiter(deleted)

Itemized List of Changes

- Project Objectives
    - Optimize and debug the forum page
    - Optimize and debug the tutor page
    - Optimize and debug the sign-in page
    - Optimize and debug the sign up page
    - Optimize and debug the point system
    - Optimize and debug the comment page
    - Implement the search page
- User Interface Diagrams
    - Screenshots that represent the most updated design of the site are added
- Functional Requirements Specification
    - Actors and Goals: Faculty privileges have been updated
- Interaction diagrams
    - Design patterns have been updated with some of the ones from the lectures
- Class Diagrams and Interface Specification
    - The database tables have been updated to what they actually look in MySQL
    - Design patterns have been added
    - An OCL table has been added
- Algorithms and Data Structures
    - Updated the algorithm for the point system
    - Included an algorithm for the search logic
- User Interface Diagram and Implementation
    - Updated the images to be what the website currently looks like
- History of Work
    - Has been updated to what we have completed at this point and the future

1. Customer Statement of Requirements:

1.1 Problem Statement

There is a lack of intercollegiate communication currently and no clear way to establish this communication. With technology evolving at such a rapid pace, we are able to communicate with virtually anyone that has access to the internet. This has allowed users to share their plethora of knowledge and resources within the web. Unfortunately, when it comes to sharing knowledge and resources between colleges, communication is not optimal. Also, many students excel in certain courses but have no incentive to share that knowledge with their peers. There is no available centralized hub where insightful interaction and discussion is possible between college students about course subjects.

Currently there are websites such as Sakai or Piazza that connect students in their own class for discussions but these websites do not connect students and faculty across universities, nor do they provide access to available job listings. Frequently the forums on these websites are not even used unless required by the professor to participate, due to a lack of an incentive. As the material of the course starts to increase in difficulty, it may be hard for some students to learn the material independently. Of course, there are students who have a thorough understanding of the course but there is no online platform to communicate this knowledge outside of their respective schools. More students are graduating every year, but there is a continual struggle for students to find a place in the job market. It is evident that colleges are starting to find methods to improve upon this issue. According to the Economic Policy Institute,

> *"Economic Policy Institute reported that roughly 8.5 percent of college graduates between the ages of 21 and 24 were unemployed. That figure is based on a 12-month average between April 2013 and March 2014, so it's not a perfect snapshot of the here and now. Still, it tells us that the post-collegiate job market, just like the rest of the labor market, certainly isn't nearly back to normal. (For comparison, the unemployment rate for all college grads over the age of 25 is 3.3 percent, which is also still higher than normal.) More worrisomely, the EPI finds that a total of 16.8 percent of new grads are "underemployed," meaning they're either jobless and hunting for work; working part-time because they can't find a full-time job; or want a job, have looked within the past year, but have now given up on searching"* (**How Bad Is the Job Market for the College Class of 2014?**).

Furthermore, many professors struggle when they teach a new class or are unsure of how to structure the course material. Professors often rely solely on previous professors that taught the same course, or other faculty members. Frequently new professors are knowledgeable in the subject they are teaching but lack the ability to teach properly, and whether it is pride or a lack of resources, most professors do not reach out to other professors to acquire a better method for structuring their course. There lacks an easily accessible archive of knowledge that professors from various universities can go to to further their understanding of materials for teaching.

A myriad of professors are conducting research constantly and are looking for potential student researchers in the local area. This is very limited and cumbersome due to many variables, such as:

- -Limited resources
- -Very few avenues of communication
- -It is often difficult to advertise research opportunities for students, thus faculty tend to list them on their own websites, making it difficult for students to find.

1.2 Solution

Our proposed solution is to create an interface to allow intercollegiate students and faculty to communicate accessibly with ease. Creddit's purpose is to optimize currently existing methods of communication and integrating these methods into one platform. Some may argue that Facebook, Piazza, and other existing forums already serve these purposes. However, there does not seem to be one place for students to network educationally. Our plan is to create a website that unites students, professors, researchers, recruiters and others professionally. To achieve this standard, users will be separated into Student, Faculty, Guest, Recruiter, and Admin accounts to allow for an optimal experience while browsing Creddit.

> *"Learning doesn't stop when we graduate from high school or college. Teachers want and need to be lifelong learners and grow throughout their careers. And who better to guide that growth than experienced, expert teachers?...Peer assistance and review helps new teachers escape the "sink-or-swim" approach that too often mars entry into our profession. It provides guidance and support from accomplished colleagues when teachers struggle to master this highly complex endeavor" (*Randi Weingarten is President of The American Federation Of Teachers).

In order to address the problems listed above which plague currently utilized popular websites and networks, the following features will be implemented into Creddit to successfully carry out our proposed solutions.

1.2.1 Forum

A forum is necessary for faculty and students to communicate. They should be able to comment and post to the forums. These forums should be categorized by subjects, and then can be filtered even more. An alternative to using a filter can be using the search bar which will look for keywords in the forums and display those forums.

To ensure the forum is appropriate and relevant to the subjects, the administrators or faculty should have the power to report the post and have the post/comment removed. Each post in the forum will also have a rating system. If the post receives too many downvotes, it should be automatically deleted. If the post gains a surge in popularity and gets many upvotes, other users should be able to see the post readily in the newsfeed.

The newsfeed can be on the front page and should display posts from the forums that are trending/popular. Users can browse through the categories and indicate preferences of what they would like to have shown based on their accounts. A subscription feature would be useful so that students would be able to have access to these courses with ease.

The posting on the forum should range from questions, to interesting topics, to general discussion, to related article/site links, to trending document upload. The user should have the option to save a draft of their post at anytime. When a user is creating a new post, the posting page should include a listbox which should contain all previously saved drafts saved by the user to add to their post. Comments of the post should be automatically saved periodically into the user's browser's cache while the user is typing into the comment box. This should allow the user to reload the unfinished comment if the user exits the page or the system crashes. This feature should allow long message to be saved by a user and accessed at anytime if they wish to reuse. a message format in a different post or to finish a post for later.

Note that spell check should not be added as for education, some advanced/specialized vocabulary are not covered in the default dictionary. For example, programming lines, math equations, or uncommon class-related vocabulary in posts would constantly trigger auto check errors. This will become a hassle for users to work with instead of convenience.

1.2.2 Point System

Within the forum, there should be a point system that operates under a upvote and downvote system which number is saved in each individual account. Users can upvote and downvote comments or posts that they found useful/interesting, and there should be a reward system based on the number of points accumulated. Once an account reaches a certain threshold, the user would gain a title, for example: "trusted member." Each new threshold will present the member with a new title. Once a user has enough points, they can apply for positions such as "tutor", which will be discussed below. They can cash the points in, and the funding will come from advertisements and donations. If the post is downvoted to a certain point, then the original post should be automatically deleted to acquire professional environment. If a comment/reply is downvoted to a certain point, then the comment/reply will be marked as spam and will be hidden but will also have the option to be shown if desired by the user.

Points should decay over time after a period of inactivity so those who reach a threshold for rewards will have an incentive to keep contributing to the forum and documents. Accounts with under a threshold negative points should not be able to comment, but regain points over time regardless of inactivity until the account reaches back to zero. This will dissuade spammers while allowing those who wish to contribute to do so without being afraid of losing points.

1.2.3 Tutor

Students who reach a certain prestige in a subject should have the option to apply to teach that subject and become a part time peer to peer tutor. This will allow students to interact and seek help from reliable peers. To become a tutor you must reach a certain amount of points on the point system, as mentioned in the Points Section. An incentive for users to become tutors is that tutors will be paid hourly based on time spent tutoring in the chat room, and for every new threshold achieved there is a raise in salary. If the user is eligible for tutoring, and applies to become a tutor for the certain subject, he or she has to go through an online interview to make sure they are fit for the job. The quality of performance of each tutor should be based of the rating of the students. Tutors that get below a certain rating must be reviewed to see if his or her tutor rights should be revoked.

1.2.4 Career Page

This feature should allow faculty to post internships, research, or job opportunities to the job listing that is viewable by students. These opportunities should be found either from their respective schools or other schools that are affiliated with Creddit. Students will be able to refine their searches through the filtering system, where students can search opportunities based on their interests or major. For company postings a Creddit admins should confirm and check if the company is not a scam. The company must pay a monthly fee to Creddit for advertising themselves through job postings. They will be able to post their openings directly to the page asking for the required credentials.The opportunities will be listed by different categories: subject, location, deadline, or pay. These categories are set and cannot be altered unless the users require another category.

Students interested in applying to a recruiter's posting should have to contact the recruiter through the information posted on their page, such as private email. However, students may submit their resume (created through the resume builder or uploaded themselves), and write a short essay/cover page explaining why they are interested and why they should be chosen.

This page should include a resume builder in order to help those who have never created or seen a resume before. The resume builder will be easy to use and optimal. The resume builder will ask the student to input details containing their work experience, education, GPA, coursework, college, skills, awards, references, honors, etc. Then the resume builder should then auto-fill these into a predetermined template.

1.2.5 Faculty Advice Page

This feature should be accessible to faculty as tool for inexperienced professors. Faculty should be able to interact with other faculties nationwide to seek and share advice on teaching styles, or search questions that other faculties may have asked, and see other faculty's contributions. Experienced professors should be able to offer advice to inexperienced professors through this service. Professors should interact with each other to exchange homework/project ideas to improve the course teachings and increase course ratings. The idea is to allow Creddit to expand connections between faculty in order to increase communication between different universities/schools and to help improve their teachings of the course. It is a way to increase social-educational networks with others to share creativity and wisdom of their own interpretation of the course.

Advice postings should be categorized by course-subject or by school/university. Categorizing by course-subject will allow professors to easily obtain advices concentrated on that specific course. If a professor wants to access general and specific advices from within their own community, they should be able to find their own school/university name and all the postings pertaining to the school displayed in the subcategory of courses and general advices. They should be able to easily access advices in an organized manner. Faculty members should typically expected be to appropriately categorize the material they post, but a check must be implemented to prevent and remove accidental material posted under incorrect course categories. A search feature should also be implemented as an alternative to allow specific terms in material to be found within the faculty advice page. The main purpose of this page should be to allow professors to connect with other faculty's to aid them in enhancing their own teaching methods for students.

1.2.6 Private Chat

This plugin feature should allow all users to privately start a chat with any other user of all types besides guests. This should allow users to quickly interact with one another with a chat alert pop-up when one user messages another. This allows quick comments/questions between peers which will allow communication quicker than email, but less formal. There should be a list displayed on the site so starting a chat will be easy to initiate; the list should name others who the user commonly contacts frequently called "Favorites" and extended with store recent contacts. Chats can be initiated also in other features, for example anytime the user wishes to start a conversation with a user they found on the forum, they can click a button to start a chat. Specific users can also be searched up. This chat system makes it easy for users to professionally interact quickly and efficiently without the need of adding each other to a "friends list". This also allows discussions where two different students learning the same material to discuss problems outside their own network of recognized friends.

A "block select user-types" should be implemented so a faculty user can prevent multiple students from spamming them and vice-versa. User can also allow or block specific others to chat. Private chat does not only have to be one-on-one, but also allow multiple people for a group chat with an “invite-to-chat” button. The private chats will be stored in an “inbox” viewable only between the chat-ees, indexed chronologically, and will not be open to public. However, the user should be able to search for specific private messages in their private inboxes.

1.2.7 Inbox/Email

Each profile should have a inbox/email feature linked to it’s page, allowing students and faculty to send emails to each other. Faculty should be able to email one another for one-on-one advice while students should be able to send emails to the professors who posted the listing. This feature should be a more formal means of contacting someone over the chat system as chat should be mainly implemented to allow communicate with another for only brief discussions. Email should remain integrated into the website in the case a member does not feel comfortable giving a personal email to another member. These emails should be private to the two correspondents and should be archived in the correspondents’ inbox to be searched through later by using a provided search bar to pull out key terms or senders.

1.2.8 Documents

This feature should allow students to be able to post previous years notes, lectures, and syllabuses so that students would be able to have access and possibly help clarify certain concepts that could not be understood on their own. Uploading a useful document for the class should allow users another means to gain points as an incentive. This feature should be more geared towards documents being shared within the respective school to help students get a head start in a class or clarifications if the student ever gets stuck. However, if the student feels like reading lecture slides from another class to gain a different perspective on a certain topic, then he/she should be able to find documents from another university that would be similar in terms of content which could be implemented by using a search bar. Considering the fact that there could be handwritten notes uploaded, the uploaders should have to fill in things such as a title and maybe even hashtags so that these documents can be found even if they are not typed.

1.3 Summary

This project aims to create a website that allows university-level students and faculty across the country to become part of an academic community dedicated to education and learning from each other. However, there is currently a lack of intercollegiate communication and no simple way to fortify this communication. Our website seeks to provide a platform for college students of all majors across the nation to find help and resources related to their coursework. This will be accomplished by creating a website with several distinguishing features:

1. A forum where students can post questions about their courses
2. A newsfeed which will highlight the trending posts on the forum
3. A point system where students and faculty can upvote/downvote posts based on accuracy and appropriateness
4. A page for faculty to ask for advice on teaching if they do not have previous experience
5. A place for faculty to post their available research and internship positions for students to view and apply
6. A tutoring feature for established site members to tutor other students in topics they are familiar with
7. The ability for students to post resources and documents related to their courses
8. A private chat and e-mail for students and faculty to communicate with each other one-on-one

2. Glossary of Terms:

Administrators (Admins): Employees and creators of Creddit responsible in maintaining and managing the system in Creddit.

Advertisers: Companies who want to advertise their products/services that are relevant to Creddit users.

Chatting Station: A live listing of which users are online using Creddit.

Creddit: A website which allows college students as well as the faculty to network and assist one another through means of forums, chats, and document sharing.

Customers: All those who uses Creddit, which are including guests and users.

Faculty: Members of the school's faculty (i.e. professors), who wants to assist other faculty members and students with courses they need help on, will have access to most features of Creddit.

Forum: A place where users can post questions/answers about a topic in a course.

Guests: All non-Creddit users, who can only view forums.

NewsFeed: A feature that lists the most relevant/popular/recent postings from Creddit.

Point System: Allows student and faculty users to moderate the site by upvoting or downvoting posts in the forums. There are certain thresholds for points accumulation or reduction that may or may not benefit the users depending on the appropriate and relevant the posts are. For more details, please refer to the Point System section (1.2.2).

Recruiters: Companies who agreed to advertise their career openings through Creddit's Career page to student users. They will have to pay a fee to use Creddit's services.
Resume Builder: A feature that assist students to build a resume by using a given template as an outline.

Students: Undergraduates, Graduates, and Postgraduates, who want to share resources and discuss topics on course subjects with other peers, will have access to mostly all features of Creddit.

Tracking Number: This is a number key for an item/request by a user (ie. log in credentials, post, message, document) for reference in the database. For example user identification, course identification, etc.

Tutors: Student users who have reached a certain threshold and agreed to the option of assisting other student users in understanding the course through Creddit.

Users: All those who have access to Creddit. This includes those who have created an account with Creddit (Students, Faculty, Recruiters, Admins) and those who do not have an account (Guests).

User Base: All non-Creddit and Creddit users.

User Profile: An identity users use to display themselves as in Creddit. There is a page dedicated for every Creddit user to customize their personal data with personalized editable content (i.e. email, interests, courses, major, etc). It will also include name of attending college.

3. System Requirements:

Based upon the consumer needs, a list of requirements has been provided for the system to possess. For features that must be implemented by the system, "The system/user shall" is stated, whereas for features that are preferred, "The system/user should" is stated. For each requirement, there is an identifier in the form of REQ-x, as well as a priority weight from 1 to 5. A higher priority weight indicates that the corresponding requirement is more essential and crucial to the success of the project and the customer's needs.

3.1 Enumerated Functional Requirements

	Priority Weight	**Description**
REQ-1	**5**	The system shall award points to users based on how many upvotes/downvotes the post obtained.
REQ-2	**5**	The system shall deduct points from the user if the user has not received a upvote in a predetermined set of time.
REQ-3	**4**	The system shall allow students to apply for a tutor
REQ-4	**4**	The system shall reset points back to zero after a predetermined set of time.
REQ-5	**2**	The user shall be able to private chat with another user, except for recruiters and guest and can add multiple people to their chat.
REQ-6	**2**	The system shall allow users who achieve a predetermined amount of points the ability to apply to become a tutor.
REQ-7	**2**	The system should allow faculty and students to communicate through website email.
REQ-8	**5**	The user shall be allowed to read the forums and all but recruiters and guests will be allowed to post on forums.
REQ-9	**5**	The user shall be allowed to create an account.
REQ-10	**3**	The system should allow mistakes while entering the password to login. However, to resist "illegal hacking," the number of allowed failed attempts shall be small, say five, after which the system will

		lock the account and notify the user to reset the password to regain access to the account.
REQ-11	**4**	The user shall be able to upvote or downvote users, but is not allowed to downvote or upvote the same user within the the time frame of seven days if the user is a student.
REQ-12	**4**	The user shall be allowed to post job/research/internships if the user is a recruiter or faculty.
REQ-13	**1**	The system should log the user out if he is inactive for a predetermined amount of time.
REQ-14	**4**	The system shall store data in the database.
REQ-15	**4**	The system shall receive data from the database.

In regards with the point system, we will implement a system in which a user's upvote will follow the law of diminishing returns to prevent users from using their upvotes excessively. Upvotes are converted into points, however as a user gives out his upvotes without regard, his upvote will soon become worth very little in terms of how many points it gives the user. If the user is toxic to his community, his points will be deducted. If the user enters a negative point value, his privileges will be restricted, the user may not be allowed to comment or upvote/downvote. In order to keep the community active, we have implemented a decay system. The points will decay over a period of time if the user is inactive. The point system is the backbone of our website and the websites successes hinges on the how successful our implementation of the point system is. Our point system is the key function that keeps users using our website, with its plethora of rewards for obtaining a high point count. However in the worst case where our point system is not implemented correct, the website will have a very hard time getting its user base. With no user base, other companies will not want to post advertisements which leads to problem on how to raise funds to keep Creddit operating. In the case where a user figures out a way to abuse the point system, a swift and effective counter measure will be employed to combat the abuse. If however we are not able to effectively resolve an exploit of the point system, it will lead to the downfall of the Creddit. If the point system becomes irrelevant due to the exploit, users will not have an incentive to continue using the website (REQ-1 & REQ-2 & REQ-11).

The point system has tiers and rewards to encourage users to be proactive in the community and help fellow peers. The amount of points needed to become a tutor needs to be a number that is high but still obtainable. Those who qualify to become a tutor have established themselves as a member of the community who has contributed greatly to it and are thus rewarded for their service. It will take some time to fine tune the exact amount of pointed needed to become a tutor. Factors that will contribute to the exact amount will be the user base and how successful the point system is when it is implemented. If the user base of Creddit is not using the upvote/downvote system much, the amount of points needed to become a tutor will be lower than if the vote system is highly utilized (REQ-3 & REQ-4).

The Forum is a place where all Creddit users, except for Recruiters and Guests, can post questions and concerns about courses and topics they are unsure about. In response to that, Creddit users (Students and Faculty) can comment on the posts to answer the question in standing. Recruiters can only view the posts because there is no benefits or need for them to contribute to the forums. The system must assure that only Student and Faculty users can post and comment in the forum (REQ-8) or else the system has failed its purpose. Users that have access to Forum will be able to view, post, and comment on all forums in Creddit, regardless of school background and school courses. Forums are meant to be open with no restrictions between users. It is an open discussion between students and faculty from different schools to communicate (REQ-8).

3.2 Enumerated Nonfunctional Requirements

Non-functional requirements are a more descriptive than practical listing the qualities of our system. These requirements are based on FURPS+, which are functionality, usability, reliability, performance, supportability, and other various attributes. These requirements are mainly concerned with quality attributes such as capability, compatibility, security, responsiveness, availability, efficiency, and maintainability.

FURPS+

	Priority Weight	**Description**
REQ-16	5	The system shall have an uptime for almost 24/7.
REQ-17	4	Search engine shall be reliable and will return relevant and accurate results.
REQ-18	1	The system shall update the newsfeed based user's search preferences.
REQ-19	1	The system shall implement a captcha in order to detect if the user is a person.
REQ-20	2	The system should be easy to navigate and utilize.

REQ-21	**4**	The system should have a backup hard drive in case of failure of primary hard drive.
REQ-22	**2**	The system should return a search result for a search query of less than 10 seconds.
REQ-23	**1**	The system should be compatible with all browsers i.e. Chrome, Mozilla Firefox, Internet Explorer, Opera.
REQ-24	**3**	The system shall have downtime used for maintenance once a fortnight.
REQ-25	**4**	The system shall be able to handle heavy levels of traffic.

3.3 On-Screen Appearance Requirements

In this section, sketches of the user interface are provided for customers to fully visually understand how the website works. Do note that the arrangement and display is not finalized and is subjected to change. The purpose of the images are to give the user the essential information they need to understand the necessary features the system has. Here below are the list of requirements for the screen appearances of the system from the user's point of view. For visual representations, please refer to section 4.4 User Interface Design.

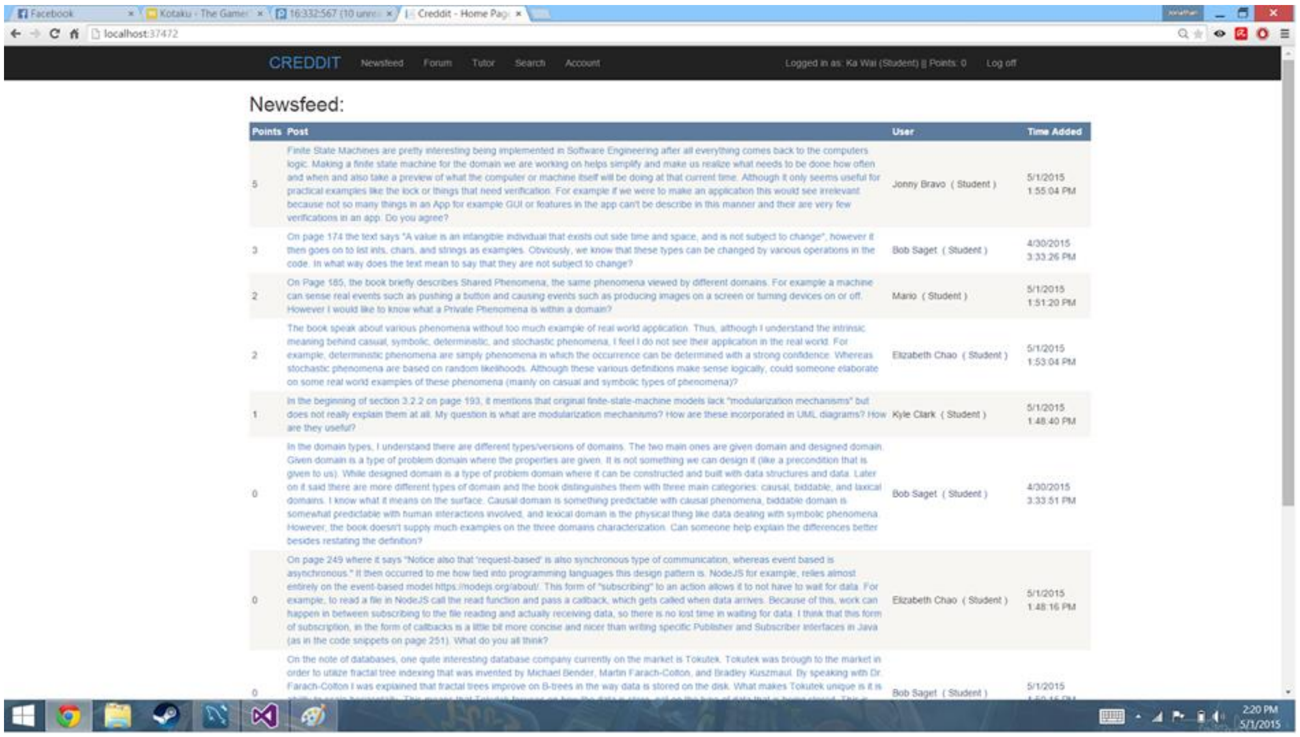
The standard format for the website is split into four sections: Creddit logo, Newsfeed, Chatting Station, and Main Features. The section on the Creddit logo will appear on the top of every webpage of Creddit along with the search bar. On the left side of the website is where the Newsfeed will be displayed with the most recent relevant postings from the forums. The right side of the website is where the Chatting Station is shown with the list of currently available users on Creddit. The central area of the website will be the Main Features of the website, such as the forums, tutors, and documents.

	Priority Weight	**Description**
OSA-1	**5**	Standard Default - Creddit logo will be on display on every webpage as well as the Login, Logout, and Sign Up link. The About Us and Contact Us link will also be provided on the top of the website. Newsfeed and Chatting Station will also be displayed on all webpages except for Chatrooms. The general search bar will displayed under the Creddit logo.

OSA-2	**5**	Home Page - Displays the default home page of Creddit. Newsfeed, Search bar, and Forums will be accessible. The option to create an account and/or to log in will be on screen in order to access other features available in Creddit.
OSA-3	**5**	Forums Page - Displays the categories of the forums by subject and topics.
OSA-4	**4**	Tutor Page - Displays the sessions by subject and topics.
OSA-5	**2**	Documents Page - Displays the documents by categories of subject, course number, and topics.
OSA-6	**1**	Career Page - Displays and lists Career opportunities, which can be filtered by interest of work concentration, location, pay, and deadline.
OSA-7	**4**	Student Dashboard - Displays a similar Home Page view with more features (Forums, Tutors, Documents, Resume Builder, Career, Newsfeed, Chat, and Inbox). It will also have the options to customize User Profile.
OSA-8	**4**	Faculty Dashboard - Displays a similar Home Page view with more features (Forums, Faculty Advice, Career, Newsfeed, Chat, and Inbox). It will also have the options to customize User Profile.
OSA-9	**3**	Recruiter Dashboard - Displays a similar Home Page view with more features (Forums, Career, Newsfeed, Chat, and Inbox). It will also have the options to customize User Profile.
OSA-10	**1**	Advertisements - Advertisements will be located at certain locations of the website. They must be relevant to the users in the academic sense.
OSA-11	**1**	Faculty Advice Page - Displays advices, which are organized by subject.

3.4 User Interface Design

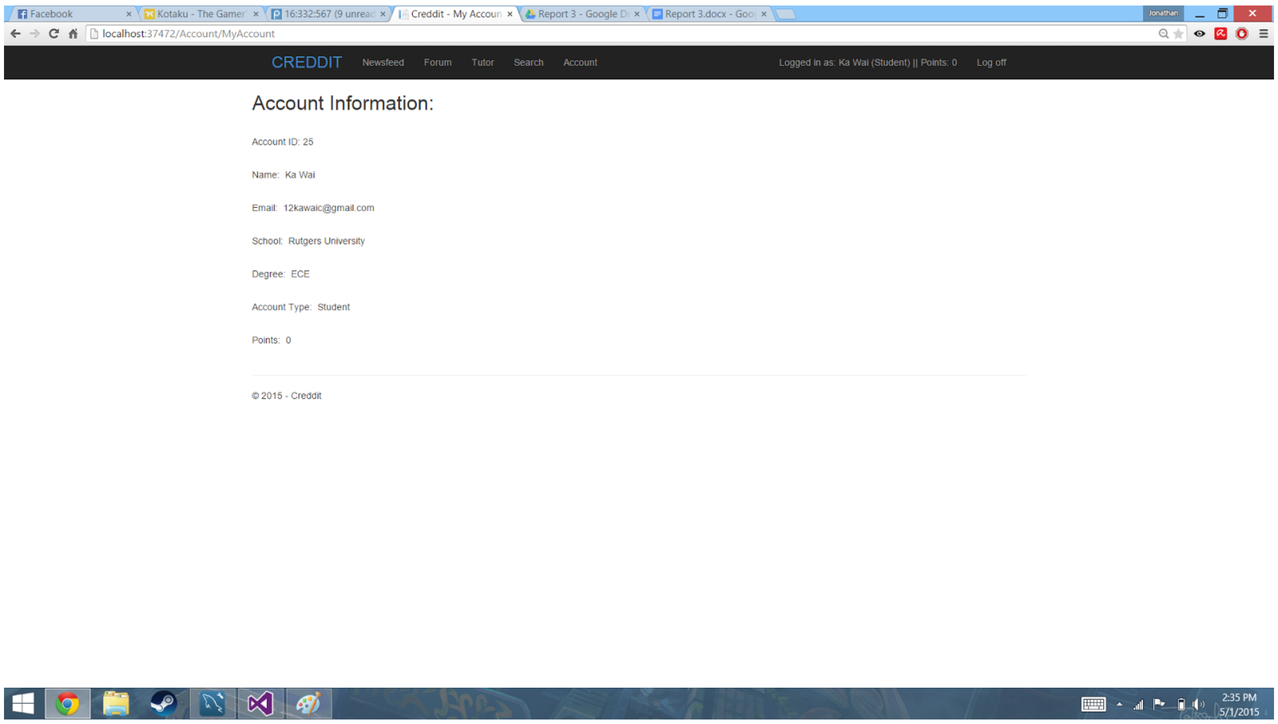
*Note: only selected Interface designs are presented below because they are the main functions of Creddit. The features that have actually been implemented are updated with screenshots of what the actual design of the website looks like.


CREDDIT Newsfeed Forum Tutor Search Account Logged in as: Ka Wai (Student) || Points: 0 Log off

Newsfeed:

Points	Post	User	Time Added
5	Finite State Machines are pretty interesting being implemented in Software Engineering after all everything comes back to the computers logic. Making a finite state machine for the domain we are working on helps simplify and make us realize what needs to be done how often and when and also take a preview of what the computer or machine itself will be doing at that current time. Although it only seems useful for practical examples like the lock or things that need verification. For example if we were to make an application this would see irrelevant because not so many things in an App for example GUI or features in the app can't be describe in this manner and their are very few verifications in an app. Do you agree?	Jonny Bravo (Student)	5/1/2015 1:55:04 PM
3	On page 174 the text says "A value is an intangible individual that exists out side time and space, and is not subject to change", however it then goes on to list ints, chars, and strings as examples. Obviously, we know that these types can be changed by various operations in the code. In what way does the text mean to say that they are not subject to change?	Bob Saget (Student)	4/30/2015 3:33:26 PM
2	On Page 185, the book briefly describes Shared Phenomena, the same phenomena viewed by different domains. For example a machine can sense real events such as pushing a button and causing events such as producing images on a screen or turning devices on or off. However I would like to know what a Private Phenomena is within a domain?	Mario (Student)	5/1/2015 1:51:20 PM
2	The book speak about various phenomena without too much example of real world application. Thus, although I understand the intrinsic meaning behind casual, symbolic, deterministic, and stochastic phenomena, I feel I do not see their application in the real world. For example, deterministic phenomena are simply phenomena in which the occurrence can be determined with a strong confidence. Whereas stochastic phenomena are based on random likelihoods. Although these various definitions make sense logically, could someone elaborate on some real world examples of these phenomena (mainly on casual and symbolic types of phenomena)?	Elizabeth Chao (Student)	5/1/2015 1:53:04 PM
1	In the beginning of section 3.2.2 on page 193, it mentions that original finite-state-machine models lack "modularization mechanisms" but does not really explain them at all. My question is what are modularization mechanisms? How are these incorporated in UML diagrams? How are they useful?	Kyle Clark (Student)	5/1/2015 1:48:40 PM
0	In the domain types, I understand there are different types/versions of domains. The two main ones are given domain and designed domain. Given domain is a type of problem domain where the properties are given. It is not something we can design it (like a precondition that is given to us). While designed domain is a type of problem domain where it can be constructed and built with data structures and data. Later on it said there are more different types of domain and the book distinguishes them with three main categories: causal, biddable, and laxical domains. I know what it means on the surface. Causal domain is something predictable with causal phenomena, biddable domain is somewhat predictable with human interactions involved, and lexical domain is the physical thing like data dealing with symbolic phenomena. However, the book doesn't supply much examples on the three domains characterization. Can someone help explain the differences better besides restating the definition?	Bob Saget (Student)	4/30/2015 3:33:51 PM
0	On page 249 where it says "Notice also that 'request-based' is also synchronous type of communication, whereas event based is asynchronous." It then occurred to me how tied into programming languages this design pattern is. NodeJS for example, relies almost entirely on the event-based model https://nodejs.org/about/. This form of "subscribing" to an action allows it to not have to wait for data. For example, to read a file in NodeJS call the read function and pass a callback, which gets called when data arrives. Because of this, work can happen in between subscribing to the file reading and actually receiving data, so there is no lost time in waiting for data. I think that this form of subscription, in the form of callbacks is a little bit more concise and nicer than writing specific Publisher and Subscriber interfaces in Java (as in the code snippets on page 251). What do you all think?	Elizabeth Chao (Student)	5/1/2015 1:48:16 PM
0	On the note of databases, one quite interesting database company currently on the market is Tokutek. Tokutek was brough to the market in order to utilize fractal tree indexing that was invented by Michael Bender, Martin Farach-Colton, and Bradley Kuszmaul. By speaking with Dr. Farach-Colton I was explained that fractal trees improve on B-trees in the way data is stored on the disk. What makes Tokutek unique is it is	Bob Saget (Student)	5/1/2015

Figure 3.3.1 - Creddit Homepage/Newsfeed


CREDDIT Newsfeed Forum Tutor Search Account Logged in as: Ka Wai (Student) || Points: 0 Log off

Account Information:

Account ID: 25

Name: Ka Wai

Email: 12kawaic@gmail.com

School: Rutgers University

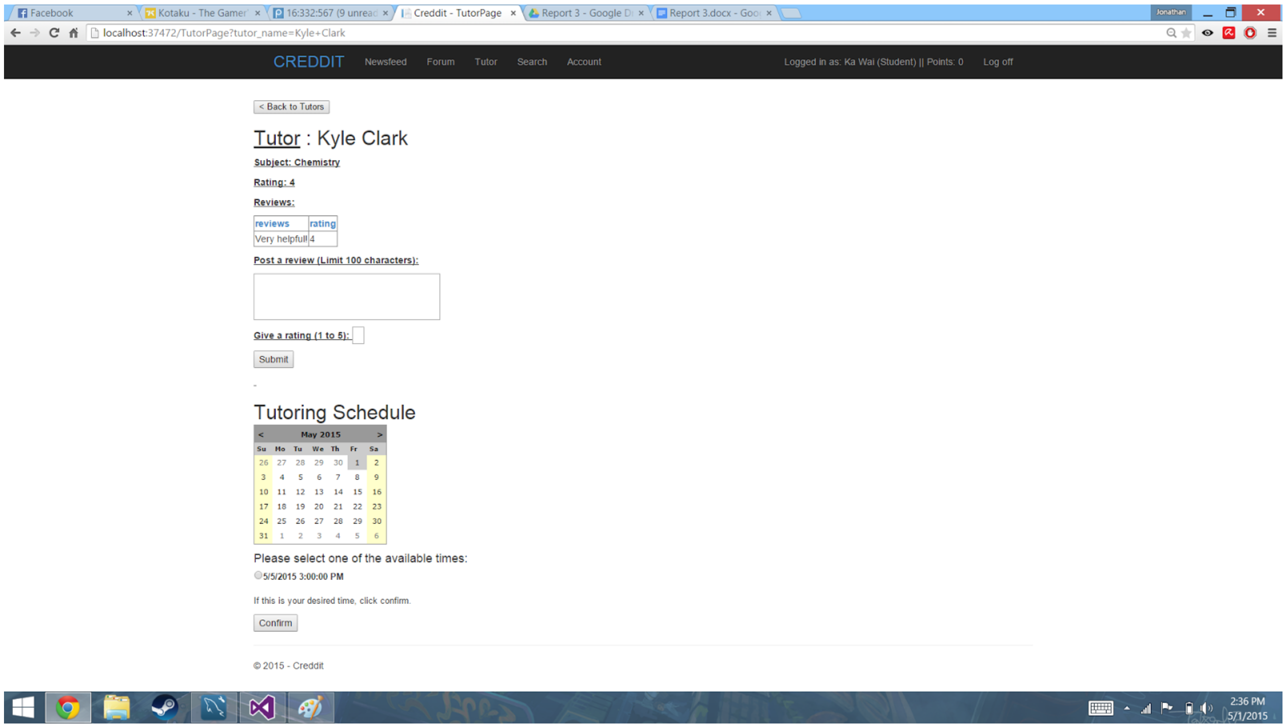
Degree: ECE

Account Type: Student

Points: 0

© 2015 - Creddit

Figure 3.3.2 - Account Page



CREDDIT Newsfeed Forum Tutor Search Account Logged in as: Ka Wai (Student) || Points: 0 Log off

< Back to Tutors

Tutor : Kyle Clark

Subject: Chemistry

Rating: 4

Reviews:

reviews	rating
Very helpfull	4

Post a review (Limit 100 characters):

Give a rating (1 to 5):

Submit

Tutoring Schedule

<			May 2015			>
Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Please select one of the available times:

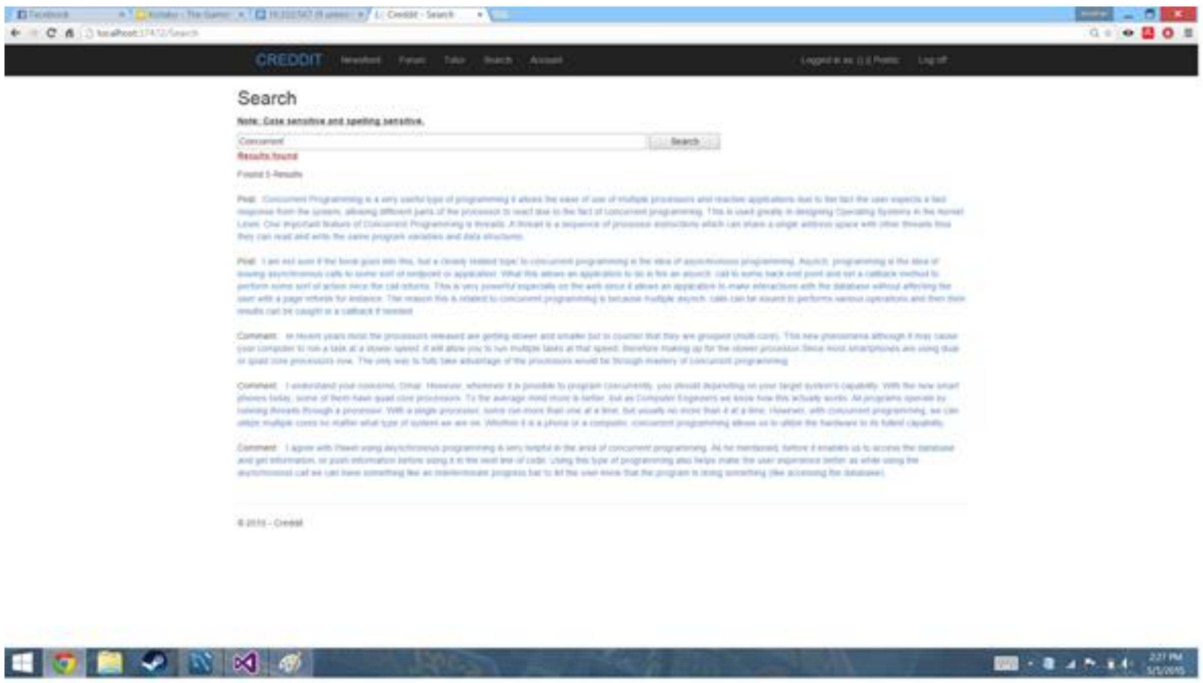
5/5/2015 3:00:00 PM

If this is your desired time, click confirm.

Confirm

© 2015 - Creddit

Figure 3.3.3 Tutor Session Sign-Up Page



CREDDIT Newsfeed Forum Tutor Search Account Logged in as: || Points: Log off

Search

Note: Case sensitive and spelling sensitive.

Concurrent Search

Results found

Found 5 Results

Post: Concurrent Programming is a very useful type of programming it allows the ease of use of multiple processors and reactive applications due to the fact the user expects a fast response from the system, allowing different parts of the processor to react due to the fact of concurrent programming. This is used greatly in designing Operating Systems in the Kernel Level. One important feature of Concurrent Programming is threads. A thread is a sequence of processor instructions which can share a single address space with other threads thus they can read and write the same program variables and data structures.

Post: I am not sure if the book goes into this, but a closely related topic to concurrent programming is the idea of asynchronous programming. Asynch. programming is the idea of issuing asynchronous calls to some sort of endpoint or application. What this allows an application to do is fire an asynch. call to some back end point and set a callback method to perform some sort of action once the call returns. This is very powerful especially on the web since it allows an application to make interactions with the database without affecting the user with a page refresh for instance. The reason this is related to concurrent programming is because multiple asynch. calls can be issued to performs various operations and then their results can be caught in a callback if needed.

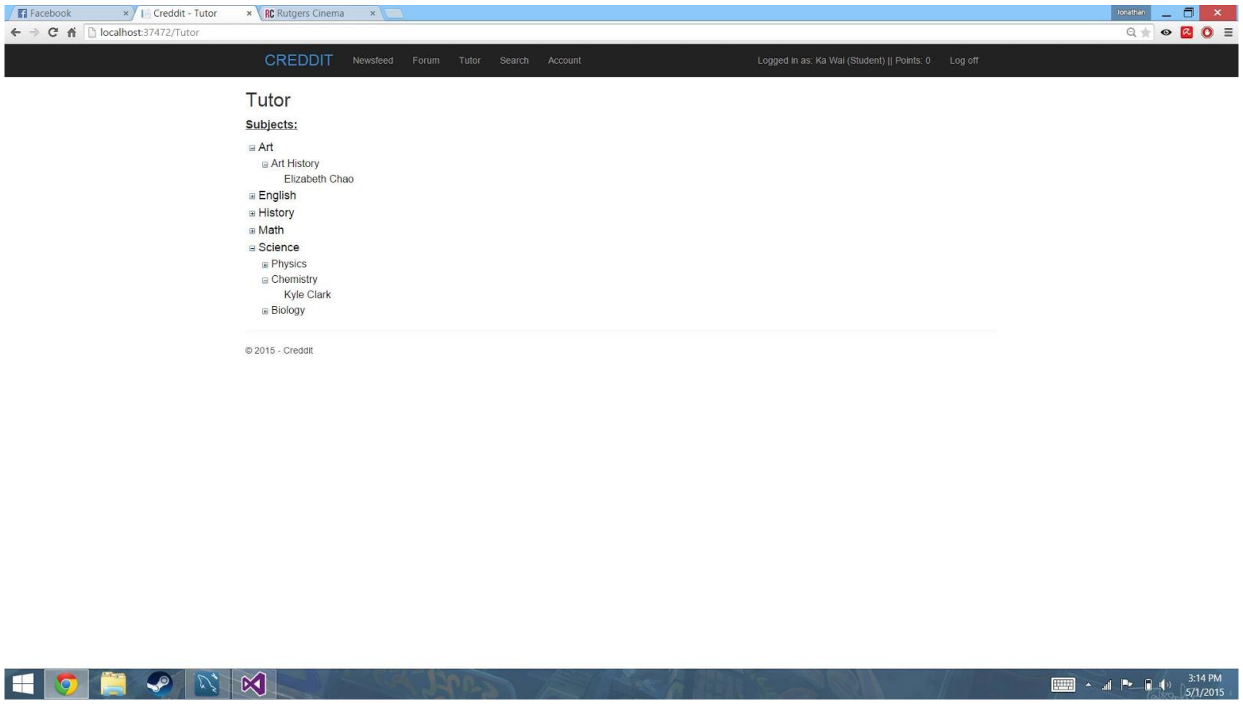
Comment: In recent years most the processors released are getting slower and smaller but to counter that they are grouped (multi-core). This new phenomena although it may cause your computer to run a task at a slower speed, it will allow you to run multiple tasks at that speed; therefore making up for the slower processor. Since most smartphones are using dual or quad core processors now. The only way to fully take advantage of the processors would be through mastery of concurrent programming.

Comment: I understand your concerns, [illegible]. However, whenever it is possible to program concurrently, you should depending on your target system's capability. With the new smart phones today, some of them have quad core processors. To the average mind more is better, but as Computer Engineers we know how this actually works. All programs operate by running threads through a processor. With a single processor, some run more than one at a time, but usually no more than 4 at a time. However, with concurrent programming, we can utilize multiple cores no matter what type of system we are on. Whether it is a phone or a computer, concurrent programming allows us to utilize the hardware to its fullest capability.

Comment: I agree with [illegible] using asynchronous programming is very helpful in the area of concurrent programming. As he mentioned, before it enables us to access the database and get information, or push information before using it in the next line of code. Using this type of programming also helps make the user experience better as while using the asynchronous call we can have something like an indeterminate progress bar to let the user know that the program is doing something (like accessing the database).

© 2015 - Creddit

Figure 3.3.4 Search Page



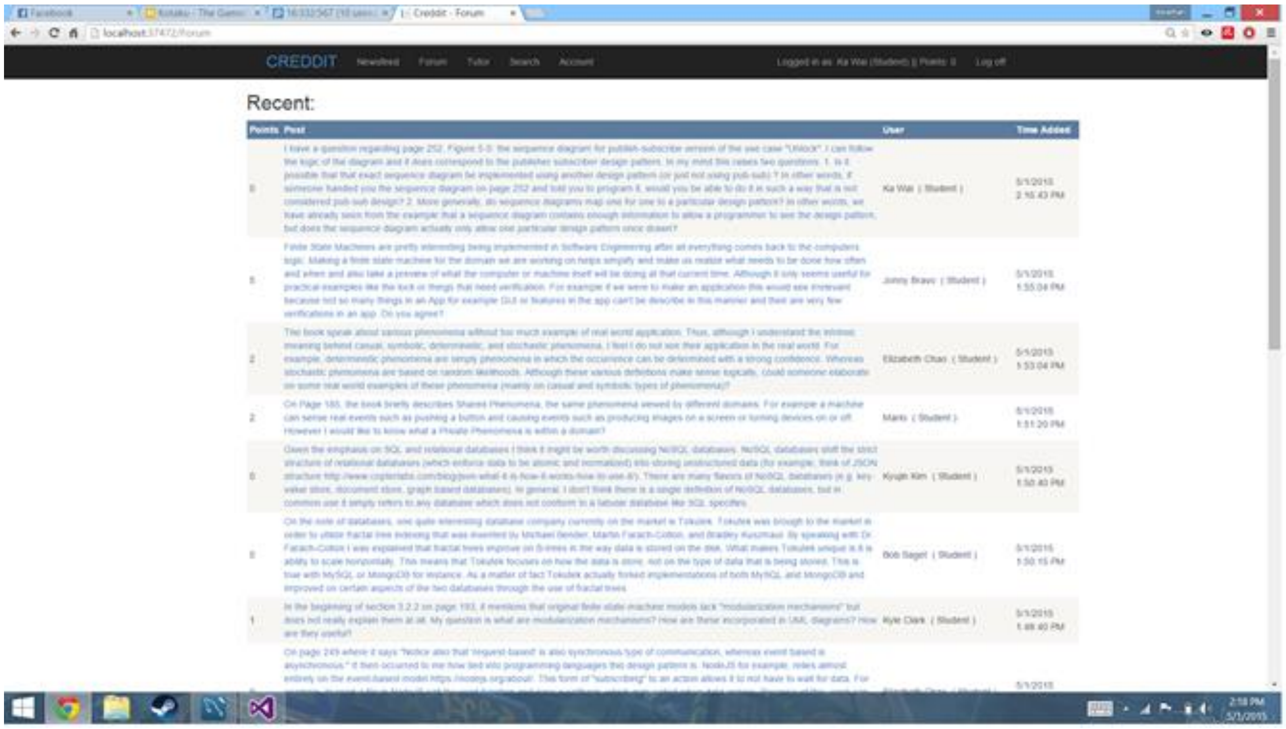
Tutor

Subjects:

- ⊟ Art
  - ⊟ Art History
    - Elizabeth Chao
- ⊞ English
- ⊞ History
- ⊞ Math
- ⊟ Science
  - ⊞ Physics
  - ⊟ Chemistry
    - Kyle Clark
  - ⊞ Biology

© 2015 - Creddit

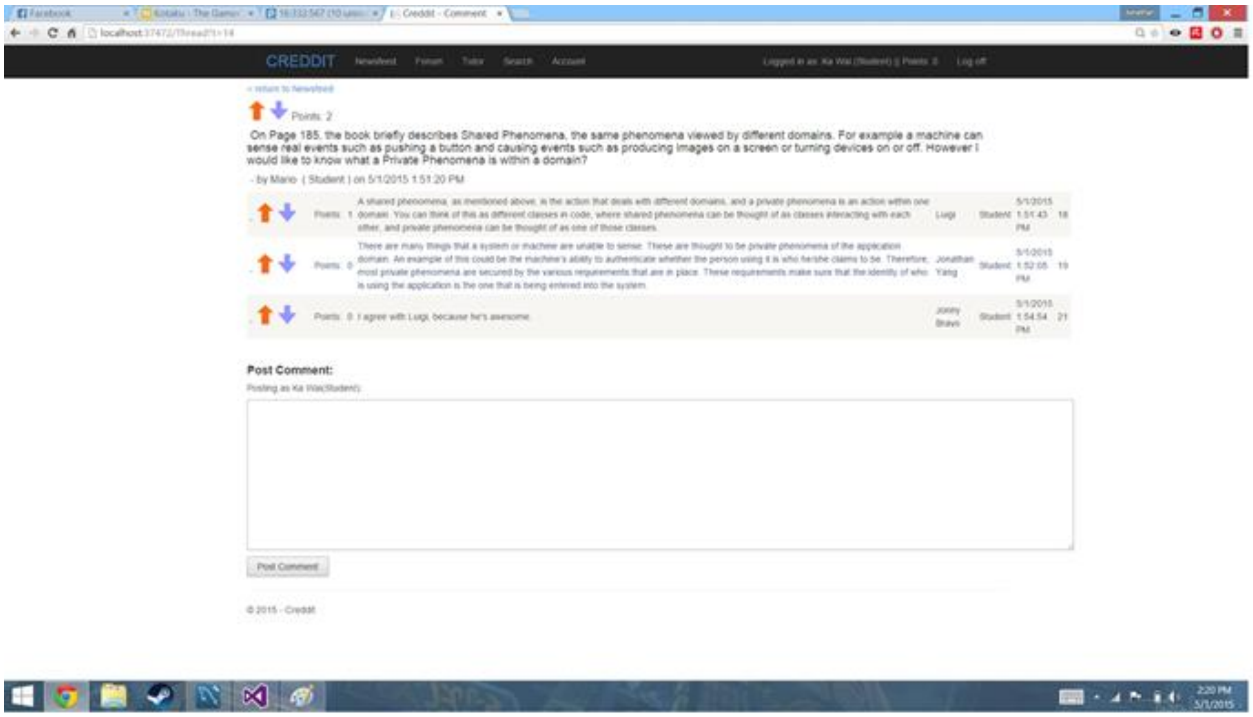
Figure 3.3.5 Tutor Page



Recent:

Points	Post	User	Time Added
0	I have a question regarding page 252, Figure 5-5: the sequence diagram for publish-subscribe version of the use case "Unlock". I can follow the logic of the diagram and it does correspond to the publisher subscriber design pattern. In my mind this raises two questions: 1. Is it possible that that exact sequence diagram be implemented using another design pattern (or just not using pub-sub) ? In other words, if someone handed you the sequence diagram on page 252 and told you to program it, would you be able to do it in such a way that is not considered pub-sub design? 2. More generally, do sequence diagrams map one for one to a particular design pattern? In other words, we have already seen from the example that a sequence diagram contains enough information to allow a programmer to see the design pattern, but does the sequence diagram actually only allow one particular design pattern once drawn?	Ka Wai (Student)	5/1/2015 2:16:43 PM
5	Finite State Machines are pretty interesting being implemented in Software Engineering after all everything comes back to the computers logic. Making a finite state machine for the domain we are working on helps simplify and make us realize what needs to be done how often and when and also take a preview of what the computer or machine itself will be doing at that current time. Although it only seems useful for practical examples like the lock or things that need verification. For example if we were to make an application this would see irrelevant because not so many things in an App for example GUI or features in the app can't be describe in this manner and their are very few verifications in an app. Do you agree?	Jonny Bravo (Student)	5/1/2015 1:55:04 PM
2	The book speak about various phenomena without too much example of real world application. Thus, although I understand the intrinsic meaning behind casual, symbolic, deterministic, and stochastic phenomena, I feel I do not see their application in the real world. For example, deterministic phenomena are simply phenomena in which the occurrence can be determined with a strong confidence. Whereas stochastic phenomena are based on random likelihoods. Although these various definitions make sense logically, could someone elaborate on some real world examples of these phenomena (mainly on casual and symbolic types of phenomena)?	Elizabeth Chao (Student)	5/1/2015 1:53:04 PM
2	On Page 185, the book briefly describes Shared Phenomena, the same phenomena viewed by different domains. For example a machine can sense real events such as pushing a button and causing events such as producing images on a screen or turning devices on or off. However I would like to know what a Private Phenomena is within a domain?	Mario (Student)	5/1/2015 1:51:20 PM
0	Given the emphasis on SQL and relational databases I think it might be worth discussing NoSQL databases. NoSQL databases shift the strict structure of relational databases (which enforce data to be atomic and normalized) into storing unstructured data (for example, think of JSON structure http://www.copterlabs.com/blog/json-what-it-is-how-it-works-how-to-use-it/). There are many flavors of NoSQL databases (e.g. key-value store, document store, graph based databases). In general, I don't think there is a single definition of NoSQL databases, but in common use it simply refers to any database which does not conform to a tabular database like SQL specifies.	Kyujin Kim (Student)	5/1/2015 1:50:40 PM
0	On the note of databases, one quite interesting database company currently on the market is Tokutek. Tokutek was brough to the market in order to utilize fractal tree indexing that was invented by Michael Bender, Martin Farach-Colton, and Bradley Kuszmaul. By speaking with Dr. Farach-Colton I was explained that fractal trees improve on B-trees in the way data is stored on the disk. What makes Tokutek unique is it is ability to scale horizontally. This means that Tokutek focuses on how the data is store, not on the type of data that is being stored. This is true with MySQL or MongoDB for instance. As a matter of fact Tokutek actually forked implementations of both MySQL and MongoDB and improved on certain aspects of the two databases through the use of fractal trees.	Bob Sagel (Student)	5/1/2015 1:50:15 PM
1	In the beginning of section 3.2.2 on page 193, it mentions that original finite state machine models lack "modularization mechanisms" but does not really explain them at all. My question is what are modularization mechanisms? How are these incorporated in UML diagrams? How are they useful?	Kyle Clark (Student)	5/1/2015 1:48:40 PM
[illegible]	On page 249 where it says "Notice also that 'request-based' is also synchronous type of communication, whereas event based is asynchronous." it then occurred to me how tied into programming languages this design pattern is. NodeJS for example, relies almost entirely on the event-based model https://nodejs.org/about/. This form of "subscribing" to an action allows it to not have to wait for data. For	Elizabeth Chao (Student)	5/1/2015

Figure 3.3.6 Forums Page



CREDDIT Newsfeed Forum Tutor Search Account Logged in as: Ka Wai (Student) || Points: 0 Log off

« return to Newsfeed

Points: 2

On Page 185, the book briefly describes Shared Phenomena, the same phenomena viewed by different domains. For example a machine can sense real events such as pushing a button and causing events such as producing images on a screen or turning devices on or off. However I would like to know what a Private Phenomena is within a domain?

- by Mario | Student | on 5/1/2015 1:51:20 PM

Points: 1	A shared phenomena, as mentioned above, is the action that deals with different domains, and a private phenomena is an action within one domain. You can think of this as different classes in code, where shared phenomena can be thought of as classes interacting with each other, and private phenomena can be thought of as one of those classes.	Luigi	Student	5/1/2015 1:51:43 PM	18
Points: 0	There are many things that a system or machine are unable to sense. These are thought to be private phenomena of the application domain. An example of this could be the machine's ability to authenticate whether the person using it is who he/she claims to be. Therefore, most private phenomena are secured by the various requirements that are in place. These requirements make sure that the identity of who is using the application is the one that is being entered into the system.	Jonathan Yang	Student	5/1/2015 1:52:05 PM	19
Points: 0	I agree with Luigi, because he's awesome.	Jonny Bravo	Student	5/1/2015 1:54:54 PM	21

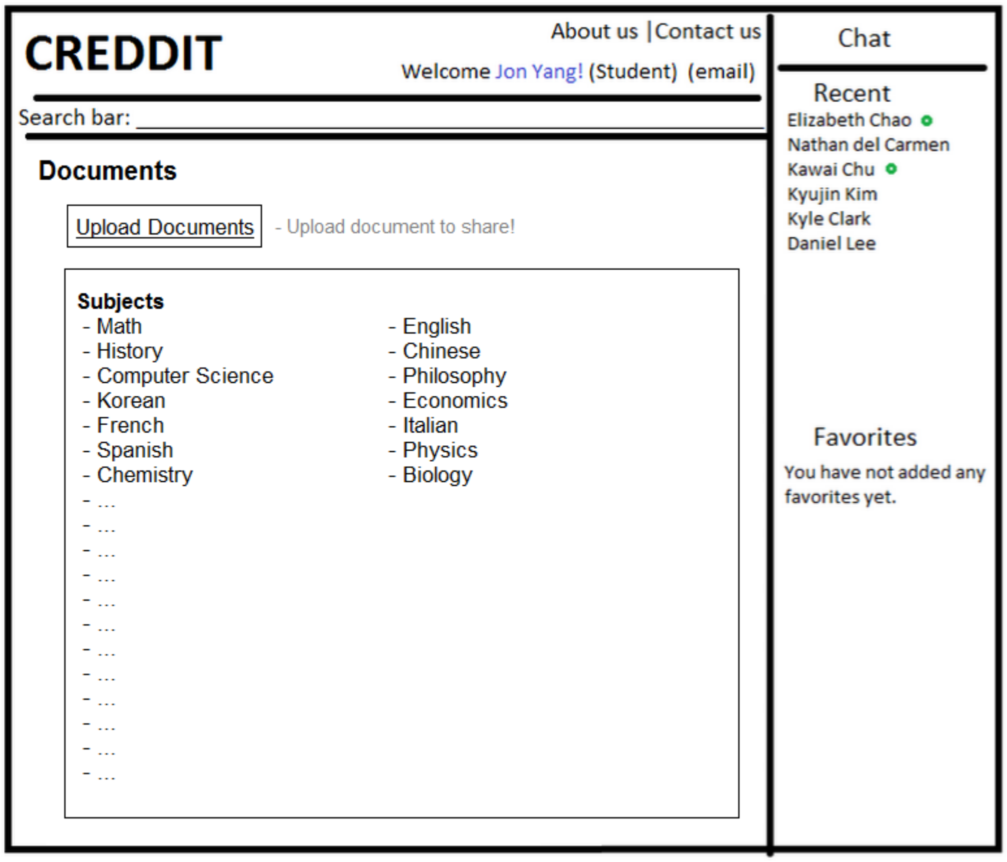
Post Comment:

Posting as Ka Wai(Student):

Post Comment

© 2015 - Creddit

Figure 3.3.7 Comment Page



CREDDIT

About us | Contact us

Welcome Jon Yang! (Student) (email)

Search bar:

Documents

Upload Documents - Upload document to share!

Subjects

- Math	- English
- History	- Chinese
- Computer Science	- Philosophy
- Korean	- Economics
- French	- Italian
- Spanish	- Physics
- Chemistry	- Biology
- …	
- …	
- …	
- …	
- …	
- …	
- …	
- …	
- …	
- …	
- …	
- …	

Chat

Recent

Elizabeth Chao

Nathan del Carmen

Kawai Chu

Kyujin Kim

Kyle Clark

Daniel Lee

Favorites

You have not added any favorites yet.

Figure 3.3.8 Documents Page

3.5 Acceptance Test Cases (ATC)

Acceptance tests are run by customers to check whether the system meets the demands and requirements of the customers. Note that the following test cases provided below are only coarse descriptions of how the requirements of the system will be tested. These acceptance test cases are designed for the website to take in account for the majority of the functions that can be carried out on the site. These test cases show the required input, pass/fail criteria, and the various scenarios that can occur.

Acceptance Test Cases for REQ-1:
ATC1.01: Gain a threshold points with student account. (Pass: Student obtains points)
ATC1.02: Gain a threshold points with tutor account. (Pass: Tutor obtains points)
ATC1.03: System awards the student a new title. (Pass: Student is granted another title)
ATC1.04: System awards the tutor a new title. (Pass: Tutor is granted another title)

Acceptance Test Cases for REQ-2:
ATC2.01: System deducts points from a student because s/he did not receive an upvote for a predetermined time. (Pass: Student's points are deducted)
ATC2.02: System deducts points from a tutor because s/he did not receive an upvote for a predetermined time. (Pass: Tutor's points are deducted)

Acceptance Test Cases for REQ-3:
ATC3.01: Student obtains enough points to be eligible to apply for a tutor. (Pass: Student is eligible to apply for a tutor)
ATC3.02: Student applies for a tutor. (Pass: System allowed the student to become a tutor)

Acceptance Test Cases for REQ-4:
ATC4.01: Student has had his point reset back to zero after waiting the predetermined set of time. (Pass: The system sets the student's points are now zero)

Acceptance Test Cases for REQ-5:
ATC5.01: Student is able to create a chat with another student. (Pass: The student is in a chat with another student)
ATC5.02: Chat is able to allow multiple people to be in it at once. (Pass: The chat has more than two students communicating with each other)

Acceptance Test Cases for REQ-6:
ATC6.01: Student obtains enough points to be eligible to apply for a tutor. (Pass: Student is eligible to apply for a tutor)
ATC6.02: Student applies for a tutor. (Pass: System allowed the student to become a tutor)

Acceptance Test Cases for REQ-7:
ATC7.01: A student sends an email to a faculty member through the website email system. (Pass: The email is successfully sent and received by the faculty member.)
ATC7.02: A student sends an email to a student through the website email system. (Pass: The email is successfully sent and received by the student.)
ATC7.03: A faculty member sends an email to a faculty member through the website email system. (Pass: The email is successfully sent and received by the faculty member.)
ATC7.04: A faculty member sends an email to a student through the website email system. (Pass: The email is successfully sent and received by the student.)

Acceptance Test Cases for REQ-8:
ATC8.01: A student views and posts/comments on the forums. (Pass: The student can see all content on the forum and the post/comment is posted successfully.)
ATC8.02: A faculty member views and posts/comments on the forums. (Pass: The faculty member can see all content on the forum and the post/comment is posted successfully.)
ATC8.03: An administrator views and posts/comments on the forums. (Pass: The administrator can see all content on the forum and the post/comment is posted successfully.)
ATC8.04: A recruiter views and posts/comments on the forums. (Pass: The recruiter can see all content on the forum but is not able to post or comment.)
ATC8.05: A guest views and posts/comments on the forums. (Pass: The guest can see all content on the forum but is not able to post or comment.)

Acceptance Test Cases for REQ-9:
ATC9.01: The user creates a student account. (Pass: The student email is verified and the account is created.)
ATC9.02: The user creates a faculty account. (Pass: The faculty email is verified and the account is created.)
ATC9.03: The user creates a recruiter account. (Pass: The recruiter email is verified and the account is created.)

Acceptance Test Cases for REQ-10:
ATC10.01: Enter correct password to successfully login upon first try. (Pass: Logged in)
ATC10.02: Enter incorrect password on first try, and correct password on second try and successfully login. (Pass: Logged in after second try)
ATC10.03: Enter incorrect password on first and second try, and correct password on third try and successfully login. (Pass: Logged in after third try)
ATC10.04: Enter incorrect password on first, second, and third try, and correct password on fourth try and successfully login. (Pass: Logged in after fourth try)
ATC10.05: Enter incorrect password on first, second, third, and fourth try, and correct password on fifth try and successfully login. (Pass: Logged in after fifth try)
ATC10.06: Enter in incorrect password five times, system will lock you out. (Pass: Notifies that you have been logged out and sends an email to your personal email account)

Acceptance Test Cases for REQ-11:
ATC11.01: Upvote a user, and upvote the same user again on the same day. (Pass: System did not allow the user to upvote again)
ATC11.02: Upvote a user, and upvote the same user again the next day. Do it every day for the next seven days. (Pass: System did not the same user to upvote another user anytime within the seven days)
ATC11.03: Upvote a user, and upvote the same user again after the seven days are up. (Pass: System allowed the user to upvote the same user)

Acceptance Test Cases for REQ-12:
ATC12.01: Post job/research/internship opportunity with a faculty account. (Pass: Allowed faculty to post)
ATC12.02: Post job/research/internship opportunity with a recruiter account. (Pass: Allowed recruiter to post)
ATC12.03: Post job/research/internship opportunity with a student account. (Pass: Does not allow student to post)

Acceptance Test Cases for REQ-13:
ATC13.01: System logs the faculty account out because s/he is inactive for a predetermined period of time. (Pass: Allowed faculty account to successfully log-out)
ATC13.02: System logs the recruiter account out because s/he is inactive for a predetermined period of time. (Pass: Allowed recruiter account to successfully log-out)
ATC13.03: System logs the student account out because s/he is inactive for a predetermined period of time. (Pass: Allowed student account to successfully log-out)
ATC13.04: System logs the tutor account out because s/he is inactive for a predetermined period of time. (Pass: Allowed tutor account to successfully log-out)

Acceptance Test Cases for REQ-14:
ATC14.01: Post to the chat system with a faculty account. (Pass: Allowed faculty to post and store it to the database)
ATC14.02: Post to the chat system with a recruiter account. (Pass: Allowed recruiter to post and store it to the database)
ATC14.03: Post to the chat system with a student account. (Pass: Allowed student to post and store it to the database)
ATC14.04: Post to the chat system with a guest account. (Pass: Does not allow the guest to post and is recommended to sign-up for an account)
ATC14.05: Post to the forum with a faculty account. (Pass: Allowed faculty to post and store it to the database)
ATC14.06: Post to the forum with a recruiter account. (Pass: Does not allow the guest to post)
ATC14.07: Post to the forum with a student account. (Pass: Allowed faculty to post and store it to the database)
ATC14.08: Post to the forum with a guest account. (Pass: Does not allow the guest to post and is recommended to sign-up for an account)

ATC14.09: Guest creates a new account. (Pass: Allows guest to create an account and store account information into the database)

Acceptance Test Cases for REQ-15:
ATC15.01: System receives information from the database after the user completes the login page correctly (Pass: Receives the input from the user and receives information from the database and checks whether or not the given input matches any from the database)
ATC15.02: System receives information from the database after the guest completes the signup page correctly (Pass: Receives the input from the guest and receives information from the database and checks whether or not the given input already exists within the database)
ATC15.03: System receives information from the database when a user opens up the chat box. (Pass: receives information from the database and finds corresponding history of the chat between users)

Acceptance Test Cases for REQ-16:
ATC16.01: Single user attempts to access the website at any time of the day. (pass: system remains up 24/7)
ATC16.02: Multiple users attempts to access the website at any time of the day. (pass: system remains up 24/7)

Acceptance Test Cases for REQ-17:
ATC17.01: Search by entering random sequence of keys that do not relate to any known keywords of relevant results. (pass: no significant results should be found)
ATC17.02: Search by entering nothing. (pass: no significant results should be found)
ATC17.03: Search by entering keywords of desired result. (pass: system will return any available relevant results)

Acceptance Test Cases for REQ-18:
ATC18.01: Open forum newsfeed without any user search preferences. (pass: system will return only the currently popular trending posts)
ATC18.02: Open forum newsfeed with user search preferences. (pass: system will return currently popular trending posts but with user preferred posts higher up)

Acceptance Test Cases for REQ-19:
ATC19.01: Load a captcha and verify if user input is equivalent to captcha. (Pass: captcha verified and continue with account.)
ATC19.02: Load a captcha and user input incorrect captcha. (Fail: cannot continue to move on and reloads the captcha.)
ATC19.03: Manually regenerate a new captcha if captcha cannot be read by user. (Pass: a new captcha image will appear.)

Acceptance Test Cases for REQ-20:
ATC20.01: Click a tab to navigate through the features shown in the website. (Pass: redirected to the correct page selected by user.)
ATC20.02: Click on the browser's back button to go back to the previous page. (Pass: redirected back to the page before.)

Acceptance Test Cases for REQ-21:
ATC21.01: Backup data into another hard drives. (Pass: data can be retrieved from the extra hard drives.)
ATC21.02: Encountered an error in backing up data into hard drive or the system crashes. (Fail: data cannot be retrieved from hard drives.)

Acceptance Test Cases for REQ-22:
ATC22.01: User searches using the search bar. (Pass: System returns the search result in less than 10 seconds).

Acceptance Test Cases for REQ-23:
ATC23.01: The website is compatible with Chrome. (Pass: The website is accessible with all its features on Chrome.)
ATC23.02: The website is compatible with Mozilla Firefox. (Pass: The website is accessible with all its features on Mozilla Firefox.)
ATC23.03: The website is compatible with Internet Explorer. (Pass: The website is accessible with all its features on Internet Explorer.)
ATC23.04: The website is compatible with Opera.(Pass: The website is accessible with all its features on Opera.)

Acceptance Test Cases for REQ-24:
ATC24.01: A user attempts to access the website at the designated downtime (2 am EST on Monday mornings, every other week). (Pass: The website is down and the system displays a maintenance message.)
ATC24.02: Multiple users attempt to access the website at the designated downtime (2 am EST on Monday mornings, every other week). (Pass: The website is down and the system displays a maintenance message.)

Acceptance Test Cases for REQ-25:
ATC25.01: Upload large number of documents at once. (Pass: System doesn't crash or freeze)
ATC25.02: Several users posting on one or several forum. (Pass: System doesn't crash or freeze)
ATC25.03: Several users using the chat function. (Pass: System doesn't crash or freeze)
ATC25.04: Several users on the same tutoring page. (Pass: System doesn't crash or freeze)
ATC25.05: Several users logged onto CREDDIT at the same time. (Pass: System doesn't crash or freeze)

4. Functional Requirements Specification:

4.1 Stakeholders

Stakeholders include individuals and organizations which are interested in the completion and use of a given product. The amount of stakeholders and different types of stakeholders relies on the versatility and ease-of-use of the product in question. This website has a targeted user base, and thus the number of stakeholders is small. Examples of potential stakeholders include:

- users (interested in the list of features that the service provides)
    - students
    - faculty
    - recruiters
    - guests
    - administrators
- universities (interested in a place where students can gain knowledge)
- businesses (interested in a place to advertise job positions to students)

4.2 Actors and Goals

- User (Student)
    - initiating type
    - goal: To connect to other students at different universities to get help in classes they are enrolled in. Also view available internships and research positions and apply for them.
- User (Tutor)
    - initiation type
    - goal: A special type of Student, Tutors can do everything that Students can but also have the added ability to teach Students in subjects of proven expertise.
- User (Faculty)
    - initiating type
    - goal: To connect to other faculty at different universities to discuss various topics of interest, as well as advertise available research positions for students to apply for, while helping monitor the forums to keep the material relevant and on topic.
    - Faculty will have a few more privileges than users such as a student or a tutor.
    - When the faculty upvotes/downvotes, their votes will carry more weight compared to that of regular students or tutors since it should be assumed that they have more knowledge in the subject. When they also post in the forum, it will be known that the type of account will be shown as a faculty member. However, these were not yet implemented yet in the second demo but we hope to have this in the future.
- User (Guest)
    - initiating type
    - goal: To view the forums and discussions posted on the website.
- User (Administrator)

    - initiating type
    - goal: To monitor forums, chats, and tutoring sessions to ensure that the material is relevant and appropriate.
- User (Recruiter)
    - initiating type
    - goal: To post available internship positions for students to view and apply for.
- Database
    - participating type
    - goal: To store posts from the forum and chat system, as well as store user profile information.
- Email Notification System
    - participating type
    - goal: Used for confirmation when creating a account, sends emails to help keep the user up to date when talking using the forums, and notify the recruiters or faculty when a student shows interest in their research/job/internship position via sending in their resume.

4.3 Use Cases

4.3.1 Casual Description

Use Case	**Action**	**Description**
UC-1	**Posting in Forum**	There will be various topics that the user will be able to view within the forum page. When the user sees a topic relevant to what the user is looking for, the user would be able to ask a question regarding the corresponding topic and/or post a response to another user. Guests (users that are not logged in) will be able to view the contents of the forum, but they will be unable to post anything within the forum unless they are logged in.
UC-2	**Signing up for Tutoring**	The student or a tutor signs up for a tutor. This means that a faculty, recruiter, and the administrator cannot sign up for tutoring. However, when a tutor is eligible to teach a subject, that tutor is only able to teach that corresponding subject. If the same tutor desires to teach another subject, then he/she would have to go through the sign-up process again for another subject.
UC-3	**Sign up**	The system creates an account for the user. To sign up the new user will be prompted to enter a list of information along with a .edu email account. The user will then receive a confirmation email that holds a confirmation code that the user will have to

		enter to successfully confirm their email account to successfully complete the sign-up processes.
UC-4	**Log in**	The user signs into his/her account on Creddit by entering the email account they used to sign up and their password.
UC-5	**Career Post**	The user posts Job/Internship/Research opportunities, excluding Students and Guests.
UC-6	**Using the Search Bar**	The user searches on Creddit. Through the search bar, it will make it much easier for the user to navigate through the website. For instance, if the user would like to find a specific forum post regarding a question that the user has, the search would filter through all of the posts within the database and extract the posts that are relevant to what the user searched. It would also be able to help filter tutors that the user would want to specifically find.
UC-7	**Creating a Chat**	The user creates a chat with another user.
UC-8	**Using the Email system**	The user emails another user.
UC-9	**Upvoting / Downvoting**	The user upvotes or downvotes a post on the forums by clicking the up or down arrow next to the left of the post. It is important to note that a user will not be able to upvote/downvote the same user within the span of seven days.
UC-10	**Becoming a tutor**	Once a student exceeds the 500 point threshold, he/she will be eligible to apply to become a tutor.
UC-11	**Point decay**	If the user votes too many times within a set amount of time, the worth of each new vote will decay in worth on a logarithmic scale, approaching zero.
UC-12	**Point reset**	The user has been toxic to the community of Creddit, has gain a large amount of negative point, and lost his privileges. After a long period of time, has gained the chance to regain those lost privileges and has his point counter reset.
UC-13	**View Forum**	The user is accessing the forums and sees the contents of the forum. Guests (users that are not logged in) will also be able to view the contents of the forum, but they will be unable to post within the forum.

UC-14	**Saving Draft Forum Post**	The user have the option to save their posting as a draft, which will be saved in a listbox for later access.
UC-15	**Autosave Comments**	The system will automatically save the drafts of the comments incase of any system downs or window exits.
UC-16	**Update Schedule**	The Tutor updates their schedule based on available times for teaching.
UC-17	**Check Tutoring Signups**	The Tutor views the list of Students signed up for a specific tutoring session.

Use cases 2,5,7, 8, 10, 14, 15, 16, and 17 will be considered for future work due to limited amount of time

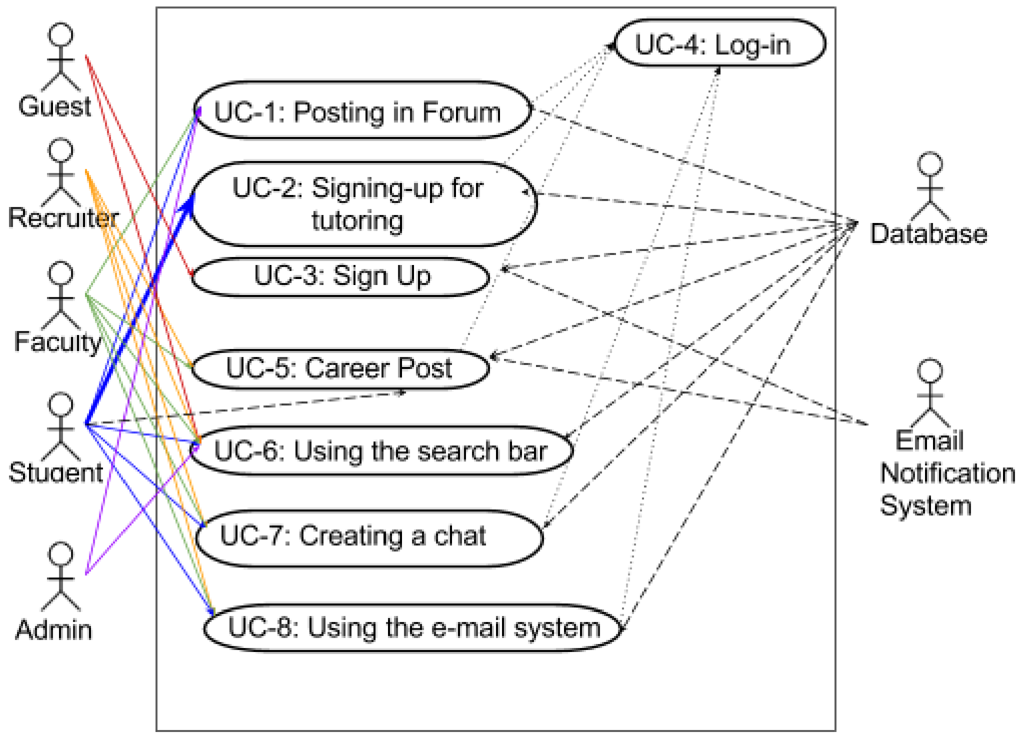


Figure 4.3.1(a)
This figure shows the interactions between the actors and use cases 1-8.

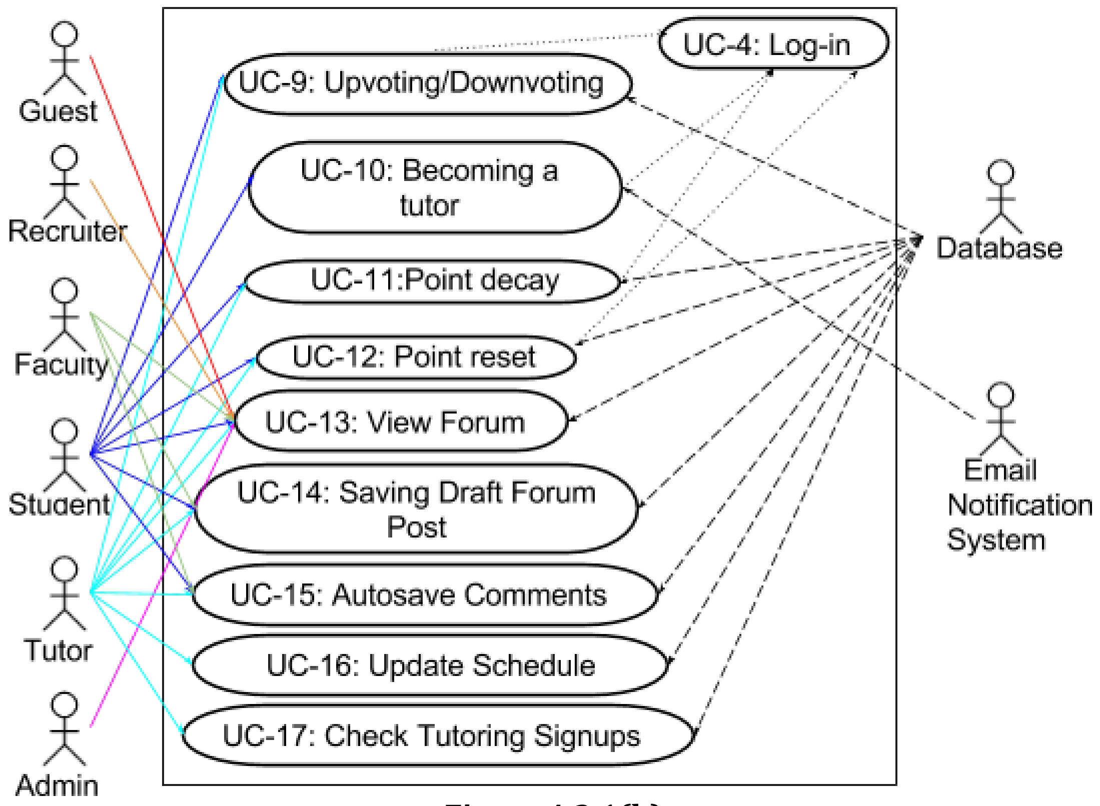


Figure 4.3.1(b)

This figure shows the interactions between the actors and use cases 9-17.

Dotted line = <<include>>
Long dotted line = <<participate>>
Bolded solid arrow = <<participate + initiate>>

4.3.3 Traceability Matrix

The Traceability Matrix allows the reader to cross the functional and non-functional requirements described earlier with the use cases. This demonstrates which use cases fulfill each requirement, and the total priority weight of each use case will determine which cases are the most important. If an X is present at any point in the column for a Use Case, then the corresponding requirement's priority weight must be added to the sum. The remaining Xs in the column are similarly considered, and the total priority weight for the Use Case is listed at the bottom of the column.

	Priority Weight	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9
REQ -1	5									x
REQ -2	5									
REQ -3	4		x							
REQ -4	4									

REQ -5	2							x		
REQ -6	2							x		
REQ -7	2								x	
REQ -8	5	x								
REQ -9	5			x						
REQ -10	3				x					
REQ -11	4									x
REQ -12	4					x				
REQ -13	3				x					
REQ -14	4	x	x	x	x	x	x	x	x	x
REQ -15	4	x	x		x	x		x	x	
REQ -16	5	x	x	x	x	x	x	x	x	x
REQ -17	4	x					x			
REQ -18	1	x					x			
REQ -19	1			x						
REQ -20	2	x	x	x	x	x	x	x	x	x
REQ -21	4	x								
REQ -22	2	x				x	x			
REQ -23	1	x	x	x	x	x	x	x	x	x
REQ -24	3	x	x	x	x	x	x	x	x	x
REQ -25	4	x	x	x	x	x	x	x	x	x
Total Weight		39	27	25	29	29	26	27	25	28

	UC-10	UC-11	UC-12	UC-13	UC-14	UC-15	UC-16	UC-17
REQ -1								
REQ -2		x						
REQ -3	x							
REQ -4			x					
REQ -5								

REQ -6								
REQ -7								
REQ -8				x	x			
REQ -9								
REQ -10								
REQ -11				x				
REQ -12								
REQ -13								
REQ -14	x	x	x	x	x	x	x	
REQ -15	x	x	x	x	x	x	x	x
REQ -16	x	x	x	x	x	x	x	x
REQ -17				x				
REQ -18				x				
REQ -19								
REQ -20	x	x	x	x	x	x	x	x
REQ -21	x	x	x	x	x	x	x	
REQ -22				x				x
REQ -23	x	x	x	x	x	x	x	x
REQ -24	x	x	x	x	x	x	x	x
REQ -25	x	x	x	x	x	x	x	x
Total Weight	31	32	31	43	32	27	27	21

4.3.4 Fully- Dressed Description

The following Use Cases below are selected to have a fully dressed description because they are more relevant and important to the system. These are the main use cases that are crucial to the system's functions. We chose the following "Posting in Forum", "Signing up for Tutoring", "Sign-Up", "Log In" and "Career Post" due to those cases being integral to this site. The core backbone of Creddit is the point system, the features that incorporate some element of the point system are vital features that we have chosen to focus our attention on. These features are the tutoring feature, the forums, and the career post feature. The tutoring and forums utilizes the points system while the career post feature helps retain a constant flow of users.

Another feature, “Spell Check” was initially given some thought to be implemented, but after heavy discussion, we decided that it would be impractical to cover all the advanced/specialized vocabulary that does not exist in the default dictionary. For example, programming lines, math equations, organic compounds, or uncommon class-related vocabulary in posts would constantly trigger auto check errors. This would become a hassle for users to work with instead of convenience.

Use Case UC1: Posting in Forum
Related Requirements: REQ-8, REQ-14, REQ-15, REQ-16, REQ-17, REQ-18, REQ-20, REQ-21, REQ-22, REQ-23, REQ-24, REQ-25 **Initiating Actor:** Student, Faculty, Administrator **Actor’s Goal:** To submit a post/comment to the forum on a new or ongoing discussion.. **Participating Actor:** Database **Preconditions:** The user must be a register account of Faculty, Administrator, or Student. **Postconditions:** The post has been updated into the forum database. **Main Success Scenario:** → 1. The **Student/Faculty** inputs and submits their post. ← 2. The **System** checks if all required fields have been answered. ← 3. **System** adds the post to the Database and displays the post in the forum page to all **Users**. **Extensions (Alternate Scenarios):** 1a. Student/ Faculty comments on an existing post. → 1. The **Student/Faculty** inputs a comment. ← 2. The **System** checks if all required fields have been answered. ← 3. **System** adds the comment onto the post and displays the comment on the post’s page. 2a. Student/Faculty attempts to submit an empty post. ← 1. **System** (a) notifies Student/Faculty with an error message that the form is incomplete, and (b) reloads the posting form. → 2. **Student/Faculty** (a) refill the posting form, and (b) submit the form. ← 3. **System** continues where it left off at step 2 in Main Success Scenario. 3a. System fails to archive and display the postings in the page. ← 1. **System** (a) notifies Student/Faculty an error message that the posting did not successfully transfer through, and (b) reloads the posting form. → 2. **Recruiters/Faculty** (a) refill the posting form, and (b) submit the form. ← 3. **System** continues where it left off at step 2 in Main Success Scenario.

Figure 4.3.2
Use Case diagram for posting in the forum.

Use Case UC2: Signing up for Tutoring

Related Requirements: REQ-3, REQ-14, REQ-15, REQ-16, REQ-20, REQ-23, REQ-24, REQ-25
Initiating Actor: Student
Actor's Goal: To receive an insightful tutoring session from another student.
Participating Actor: Database, Student.
Preconditions: User must be logged in as a Student or Tutor.
Postconditions: User will have signed up for a tutor.
Main Success Scenario:
→ 1. The **Student** enters the tutoring page and applies for a tutor.
→ 2. The **System** asks the database for list of tutors.
← 3. The **Database** sends the list of tutors.
← 4. The **System** displays the list of available tutors.
→ 5. The **Student** selects a tutor.
→ 6. The **System** ask the database for tutor's reviews.
← 7. The **Database** sends the tutor's reviews.
→ 8. The **system** asks the database for list of tutor's available time slots.
← 9. The **Database** sends the list of tutor's available time slots.
→ 10. The **Student** selects desired time slot.
→ 11. The **System** register student for time slot into the database and checks space availability.
← 12. The **System** shows the student a confirmation message stating he is registered for the time slot.

Extensions (Alternate Scenarios):
11a. The **System** checks the database for space availability and it is full.
← 1. The **Database** sends back that the time slot is full.
← 2. The **System** alerts the student that the time slot is full.
← 3. The **System** asks the student to select another time slot.

Figure 4.3.3
Use Case diagram for signing up for a tutor.

Use Case UC3: Sign-up

Related Requirements: REQ-9, REQ-14, REQ-16, REQ-19, REQ-20, REQ-23, REQ-24, REQ-25.
Initiating Actor: Guest.
Actor's Goal: To create an account that has access to the website's functionalities.
Participating Actor: Database, Email Notification System, Administrator

Preconditions: The user is not yet registered and has a valid college .edu email address. For recruiters, they will provide their respective company e-mail (i.e. johndoe@yelp.com, stevejobs@apple.com, etc.) For faculty, proof of employment in the university/ college is required.
Postconditions: The user has an account type. If the account type was a faculty or recruiter account, an administrator has checked their company email or proof of employment.
Main Success Scenario:
→ 1. The **Guest** clicks on the sign-up link.
← 2. The **System** displays the application page to sign-up.
→ 3. The **Guest** selects his account type.
→ 4. The **Guest** inputs his/her credentials, the .edu email address, and submits the application. If the account type is faculty or recruiter, they are required to input their proof of employment at their respective college/university or professional work email.
← 5. The **System** checks the basic requirements of the password.
← 6. The **System** checks to make sure email address is not in the **Database**, adds them to the **Database.**
← 7. Relocates the **Guest** to the page that says "Check Email and Activate Account"
→ 8. The **System** signals the **Email Notification System** to send the confirmation code to user's email.
→ 9. The **Guest** activates the account by entering the confirmation code in the confirmation email.
→ 10. The **System** activates the account by granting it access in the **Database** and displays the page that says, "Registration Successful!"

Extensions (Alternate Scenarios):
3a. The **Guest** enters invalid password information.
← 1. **System** (a) notifies guest with an error message that the form is incomplete, and (b) reloads the application page.
→ 2. **Guest** (a) must refill the posting form, and (b) submit the form.
← 3. **System** continues where it left off at step 5 in Main Success Scenario.
5a. The **System** displays an error when email is already in **Database**.
← 1. **System** (a) notifies guest with an error message that the email is already taken.
→ 2. **Guest** (a) must refill the posting form, and (b) submit the form.
← 3. **System** continues where it left off at step 6 in Main Success Scenario.
7a. The account type selected was faculty or recruiter.
← 1. If the account type is a faculty or recruiter type, the **System** sends the user an email telling him that his account will become active after verification by an **Administrator**.
← 2. The **System** sends the **Administrator** a notification alerting him that his attention is required to verify the proof of employment, or professional work email.
→ 3. The **Administrator** will verify the credentials and updates the system.
← 4. The **System** notifies the **Email Notification System**

← 4. The **Email Notification System** sends an email to notify their user that their account has been activated.

Note: The administrators will not sign-up through the sign-up page, but will be created by another administrators. The first administrators are the founders of Creddit.

Figure 4.3.4
Use Case Diagram for signing up for an account on Creddit.

Use Case UC4: Log-in
Related Requirements: REQ-10, REQ-13, REQ-14, REQ-15, REQ-16, REQ-20, REQ-23, REQ-24, REQ-25 **Initiating Actor:** Students, Faculty, Recruiters, Administrators. **Actor's Goal:** To log-in and access the website. **Participating Actor:** Database **Preconditions:** The user has an existing account. **Postconditions:** The user will log-in and be granted access the website. **Main Success Scenario:** → 1. The **User** enters the User ID and password. → 2. The **System** checks for the validity of the credentials to prevent "illegal hacking." ← 3. The **System** verifies the credentials by checking the **Database** and grants the user log-in privileges. **Extensions (Alternate Scenarios):** 1a. The **User** enters an invalid User ID and password. → 1 The **System** prompts an error message. 1b. The **User** enters the wrong credentials 3 times. ← 1. The **System** locks the the account and notifies the **User** to reset the password.

Figure 4.3.5
Use Case diagram for logging into Creddit.

Use Case UC5: Career Post (revision)
Related Requirements: REQ-12,REQ-14, REQ-15, REQ-16, REQ-20, REQ-22, REQ-23, REQ-24, REQ-25 **Initiating Actor:** Recruiters, Faculty

Actor's Goal: To post any career openings (jobs, research, internships) for students to apply to.
Participating Actor: Students, Database, Email Notification System
Preconditions: User is logged in as Faculty or Recruiter
Postconditions: A new career posting is added to the system's database
Main Success Scenario:
→ 1. **Recruiters/Faculty** fills out the posting form.
→ 2. **Recruiters/Faculty** submits the posting form.
← 3. **System** checks if all required fields have been answered.
← 4. **System** adds the posting to the Database and displays the posting in the Career page to Users.
← 5. **Email Notification System** notifies Recruiters/Faculty when Students apply to the posting.

Extensions (Alternate Scenarios):
2a. Recruiters/Faculty failed to answer required fields.
← 1. **System** (a) notifies Recruiters/Faculty with an error message that the form is incomplete, and (b) reloads the posting form.
→ 2. **Recruiters/Faculty** (a) refill the posting form, and (b) submit the form.
← 3. **System** continues where it left off at step 3 in Main Success Scenario.

4a. System fails to archive and display the postings in the page.
← 1. **System** (a) notifies Recruiters/Faculty an error message that the posting did not successfully transfer through, and (b) reloads the posting form.
→ 2. **Recruiters/Faculty** (a) refill the posting form, and (b) submit the form.
← 3. **System** continues where it left off at step 3 in Main Success Scenario.

Figure 4.3.6
Use Case diagram for posting a career/internship post.

4.4 System Sequence Diagrams

*Note: only selected Sequence Diagrams are presented below because they are the main functions of Creddit.

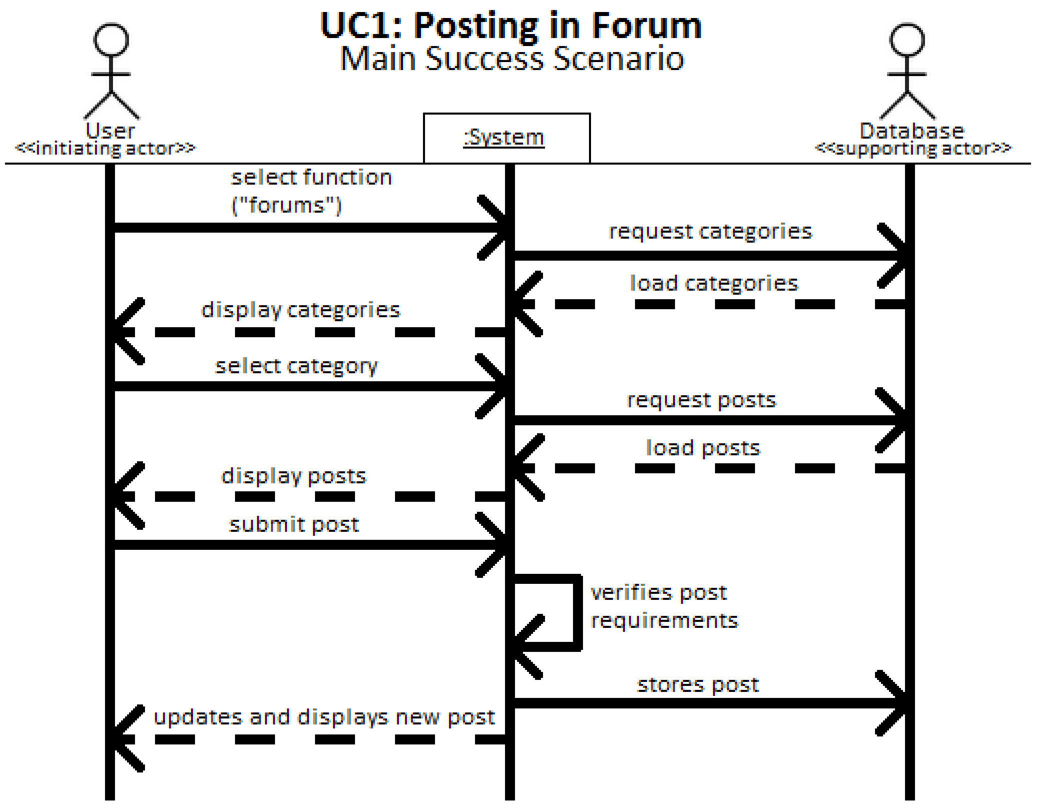


Figure 4.4.1 - UC1: Posting in Forum (Main Success Scenario)

Valid users will go to the Forum Page and the system will load and display the forums into categories. Users will select a certain category of their choice, which will notify the system to load the requested category with its corresponding posts. Users will write a post and submit a post request. After the system verifies the posting requirements, it will store the post into the database and displays the new post in the Forum page.

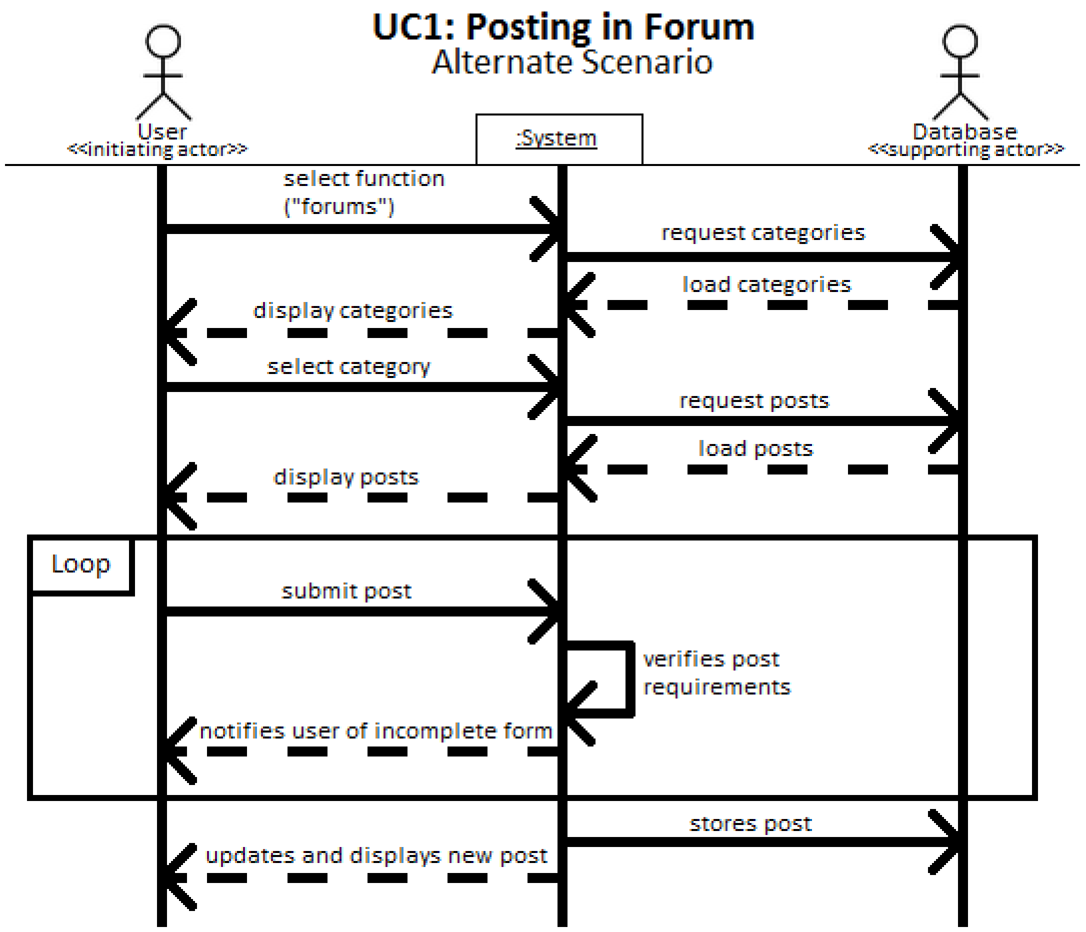


Figure 4.4.2 - UC1: Posting in Forum (Alternate Scenario 1 - incomplete form)

Valid users will go to the Forum Page and the system will load the forums into categories. Users will select a certain category of their choice, which will notify the system to load the corresponding posts. Users will write a post and submit a post request. If the user did not complete the required files of the posting form, then the system will notify the user the incomplete form. The user will then have to refill in the posting form. Once the required fields are filled, the system will store the post into the database and displays the new post in the Forum page.

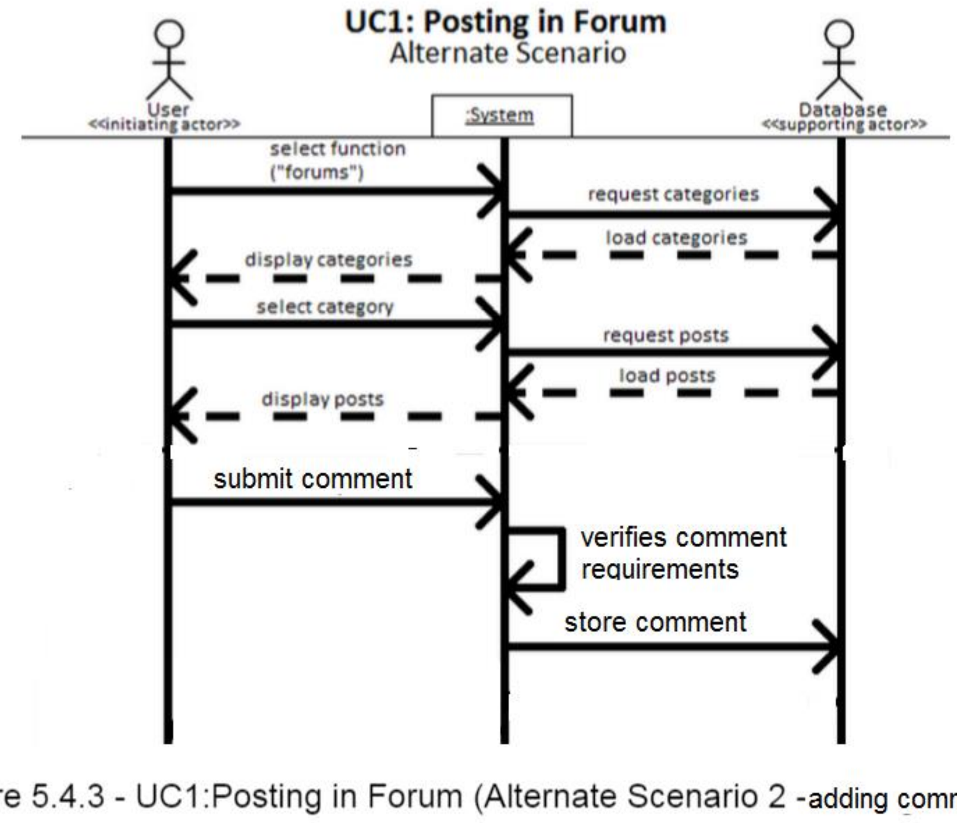


Figure 4.4.3 - UC1: Posting in Forum (Alternate Scenario 2 - adding comment)

Similar to the main scenario of posting in forum as seen in Figure 4.4.1, valid users will post a comment instead of a new forum in the Forum page. User will go to the Forum Page and select a certain category of their choice, which will notify the system to load the corresponding posts. Users will write a comment on an existing forum. Once the required fields are filled, the system will store the comment into the database and displays it below the specified forum post.

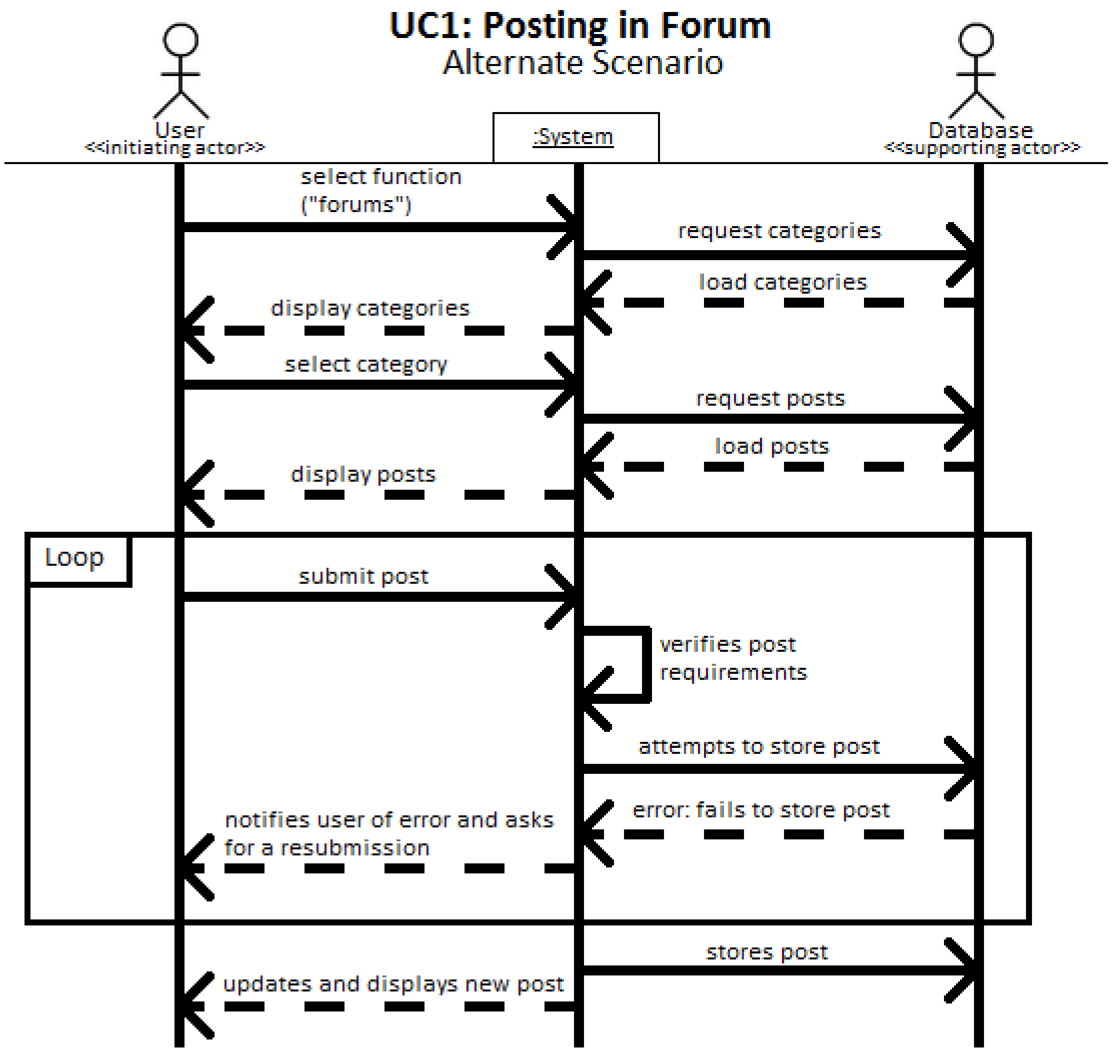


Figure 4.4.4 - UC1: Posting in Forum (Alternate Scenario 3 - storing failure)

Valid users will go to the Forum Page and the system will load the forums into categories. Users will select a certain category of their choice, which will notify the system to load the corresponding posts. Users will write a post and submit a post request. Once the required fields are filled, the system will attempt to store the post into the database. However, in this scenario, the system fails to store the post. An error message will display to the user and will ask for a resubmission.

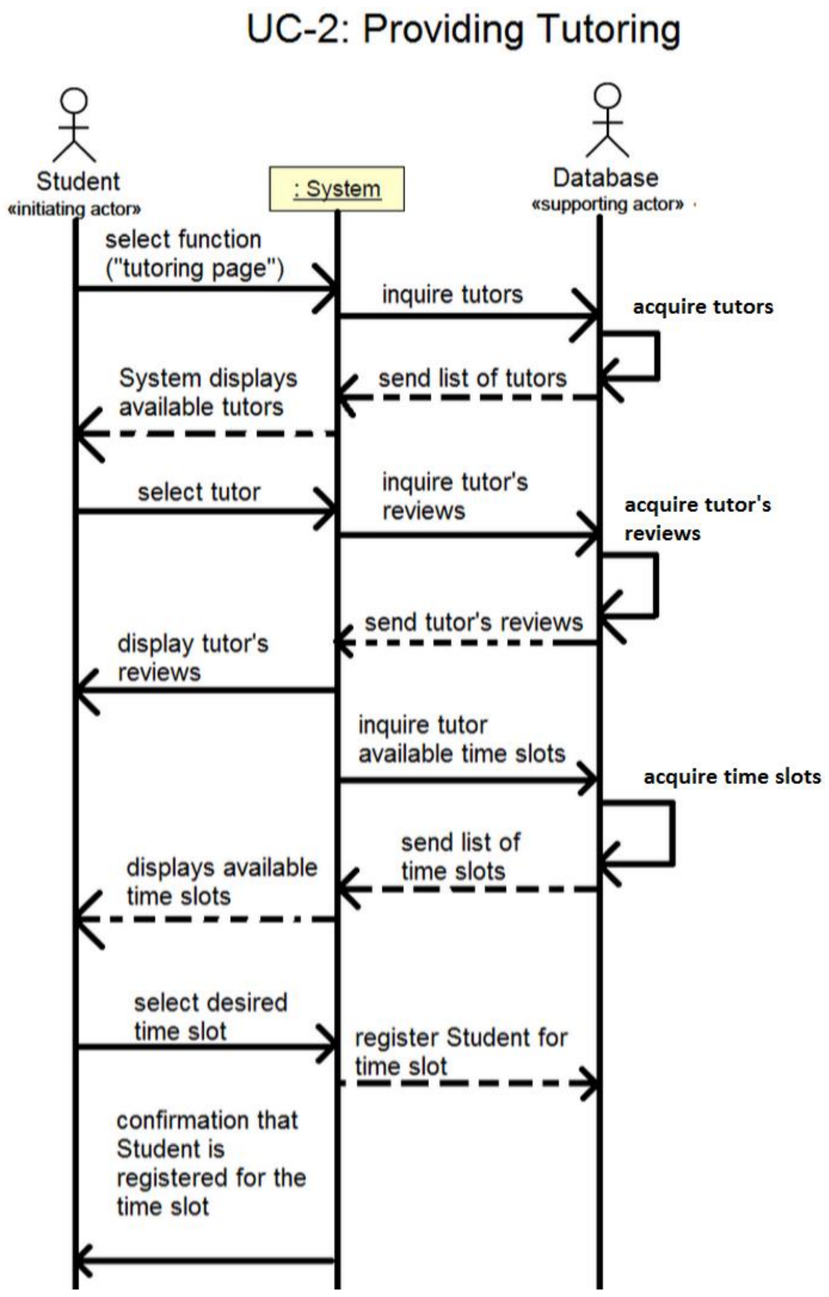


Figure 4.4.5 - UC2: Providing Tutoring (Main Success Scenario)

Valid users will access the Tutor page. The system will collect the available tutors and displays the list of tutors from the database onto the Tutor page. User will select a tutor to display the tutor's review once the system acquire the tutor's review information from the database. The system will also obtain the available time slots from the database and displays the availability to the user. User will select the desired time slot and the system will register the user to the tutor and update the database with the new information. A confirmation will be send to the user when tutoring appointment is successfully saved.

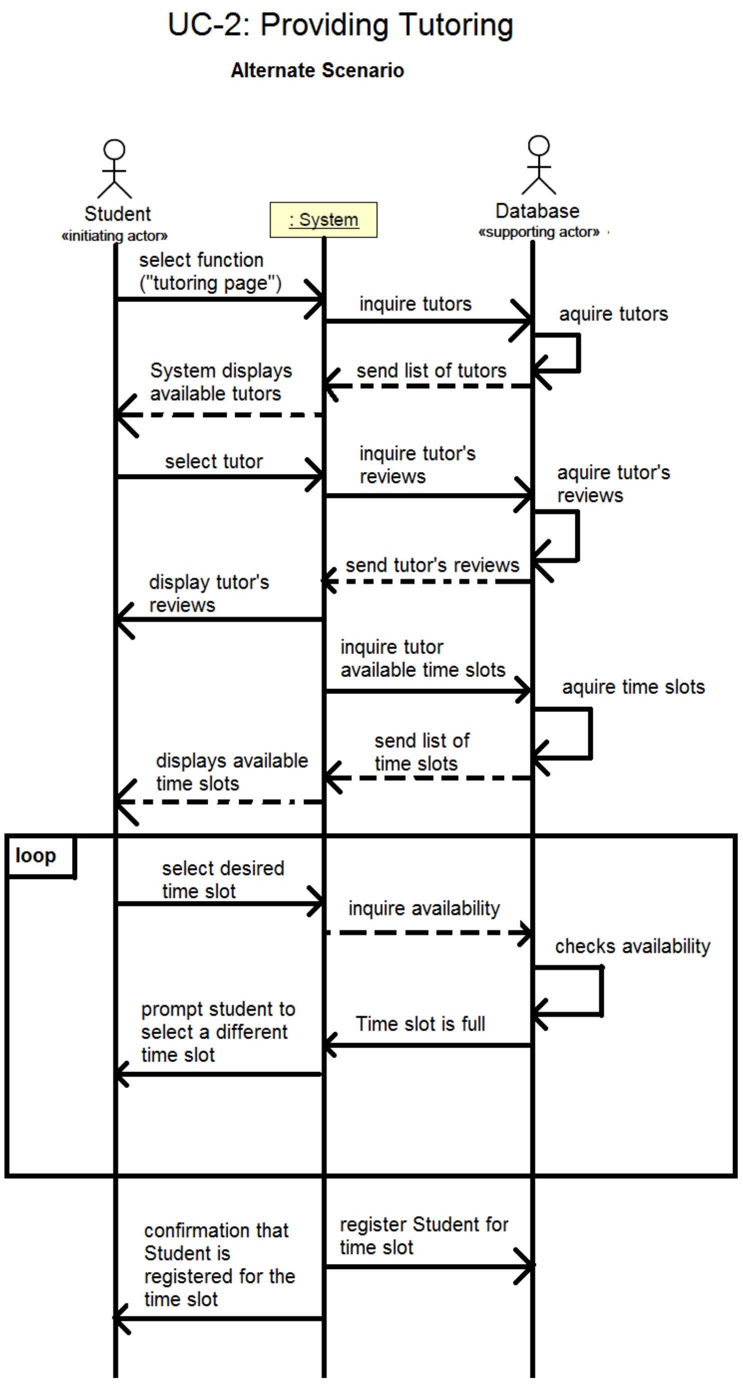


Figure 4.4.6 - UC2: Providing Tutoring (Alternate Scenario - full time slot)

Valid users will access the Tutor page. The system will collect the available tutors and displays the list of tutors from the database onto the Tutor page. User will select a tutor to display the tutor's review once the system acquire the tutor's review information from the database. The system will also obtain the available time slots from the database and displays the availability to the user. If system verifies the selected time slot is full, an error message will be prompted telling the user to select another time slot.

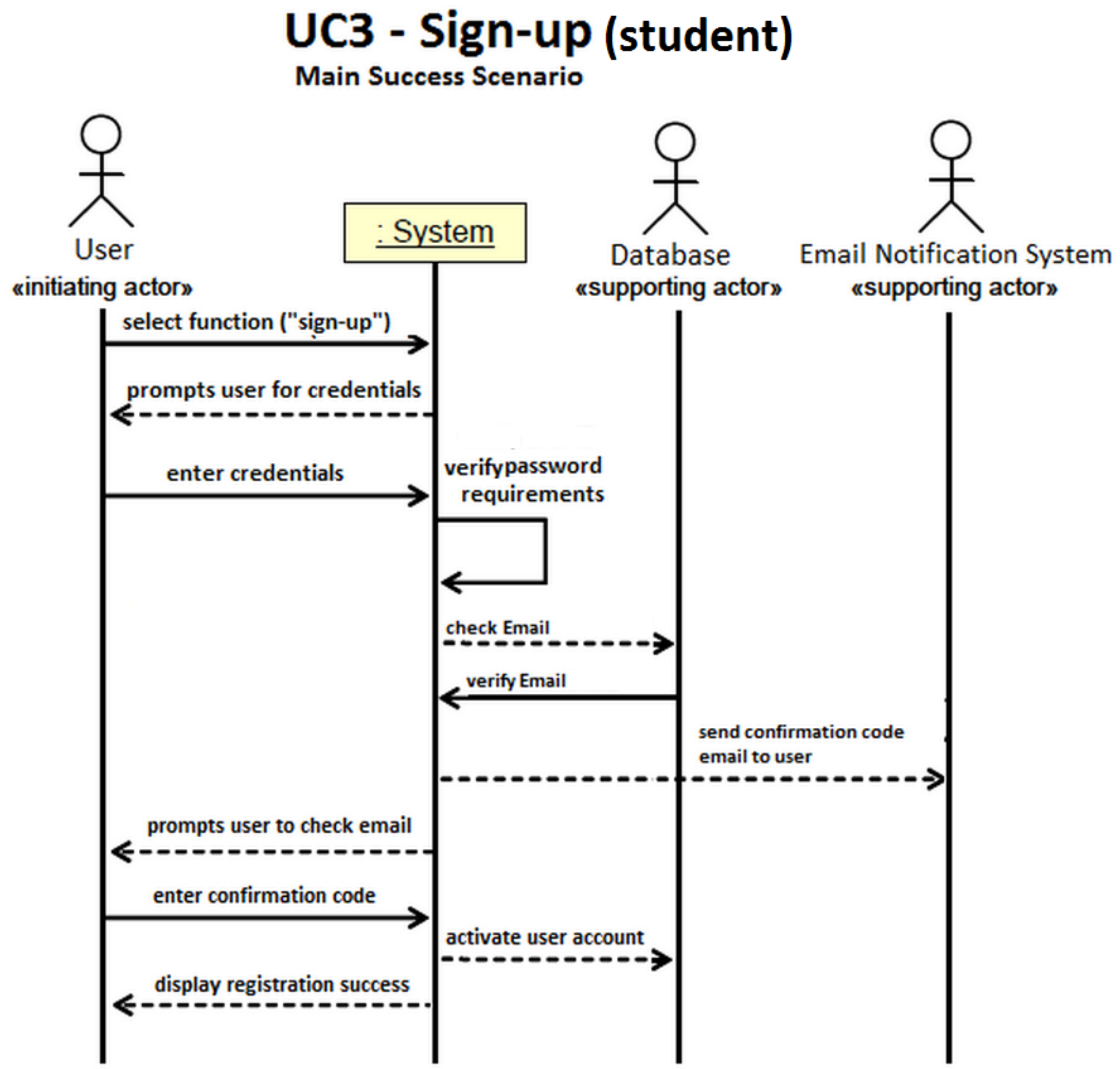


Figure 4.4.7 - UC3: Sign Up (Main Success Scenario)

User will click on the sign-up link and the system will redirect them to a form. User will enter the required credentials and submit the form. System will verify the information and check with the database to avoid duplicate accounts. Once verified by the database, the system will send a confirmation code email to the user through the Email Notification System. User will have to confirm the email using the confirmation code so system can activate the account.

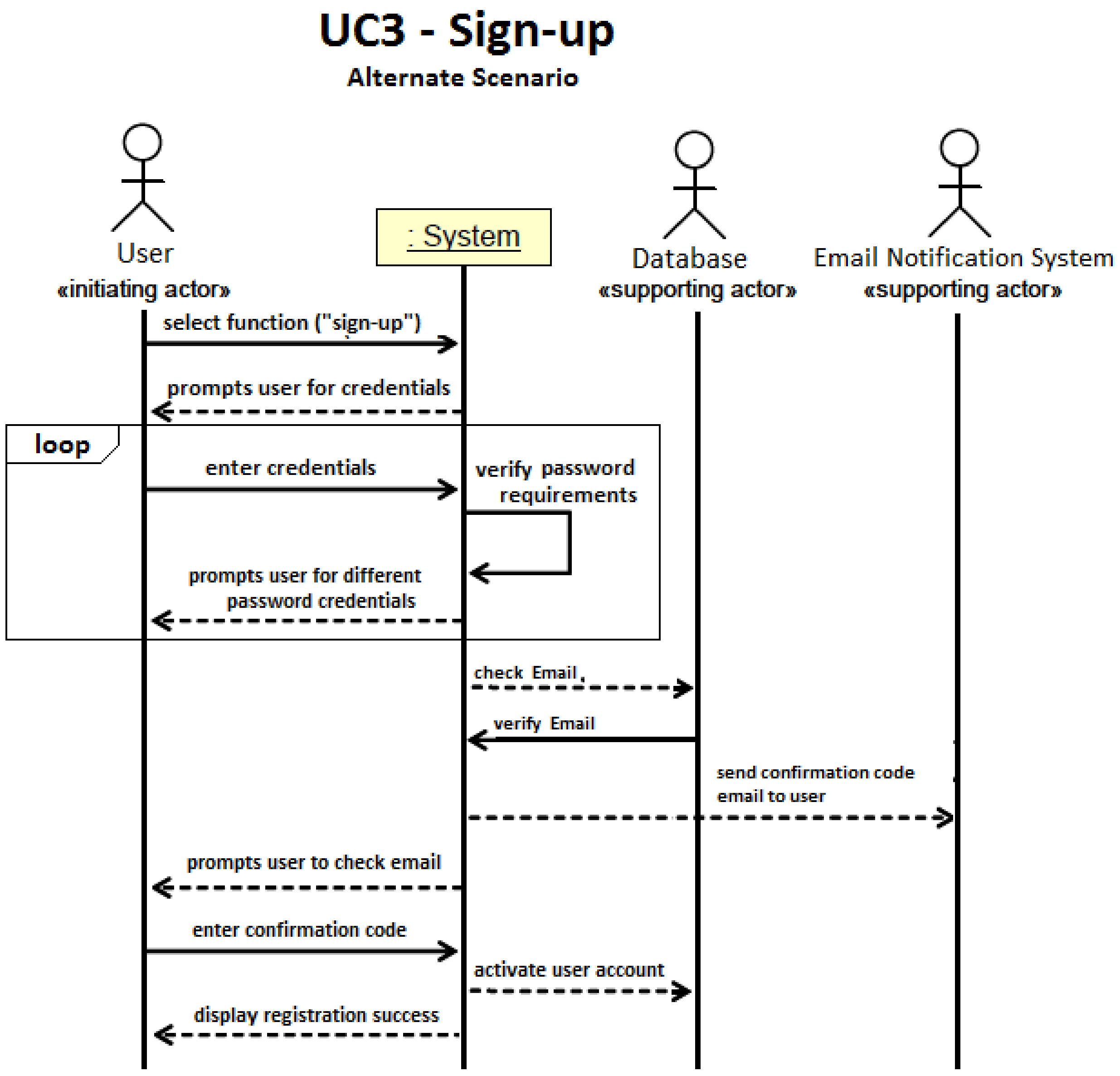

UC3 - Sign-up
Alternate Scenario
User
«initiating actor»
: System
Database
«supporting actor»
Email Notification System
«supporting actor»
select function ("sign-up")
prompts user for credentials
loop
enter credentials
verify password requirements
prompts user for different password credentials
check Email
verify Email
send confirmation code email to user
prompts user to check email
enter confirmation code
activate user account
display registration success

Figure 4.4.8 - UC3: Sign Up (Alternate Scenario 1 - Failed Password Requirement)

Similar to the main scenario of Sign Up as seen in Figure 4.4.7, the system will redirect User to a form. User will enter the required credentials and submit the form. System will verify the information and check with the database to avoid duplicate accounts. In this case, the password did not fulfill the security requirements, so the system will request user to create a different password. Once the requirement is fulfilled, the system will continue to check and verify the credentials with the database and send a confirmation code to activate the account.

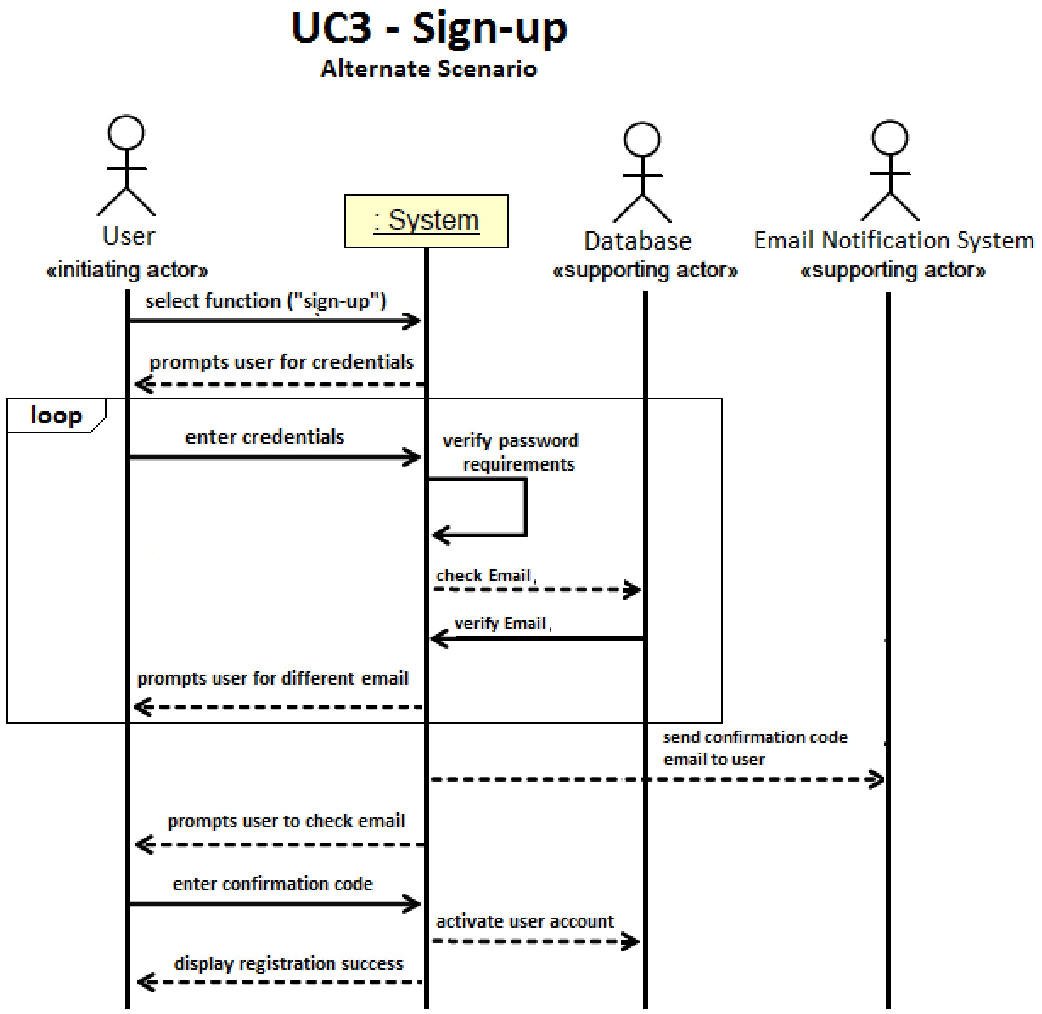

UC3 - Sign-up
Alternate Scenario
User
«initiating actor»
: System
Database
«supporting actor»
Email Notification System
«supporting actor»
select function ("sign-up")
prompts user for credentials
loop
enter credentials
verify password requirements
check Email,
verify Email,
prompts user for different email
send confirmation code email to user
prompts user to check email
enter confirmation code
activate user account
display registration success

Figure 4.4.9 - UC3: Sign Up (Alternate Scenario 2 - invalid email)

Similar to the alternate scenario of Sign Up as seen in Figure 4.4.8, the system will redirect User to a form, and User will submit the completed form. System will verify the information and check with the database to avoid duplicate accounts. In this case, the email is invalid (account was previously created or is not .edu account). System will request user to use a different email. Once the email is valid, the system will continue to check and verify the credentials with the database and send a confirmation code to activate the account.

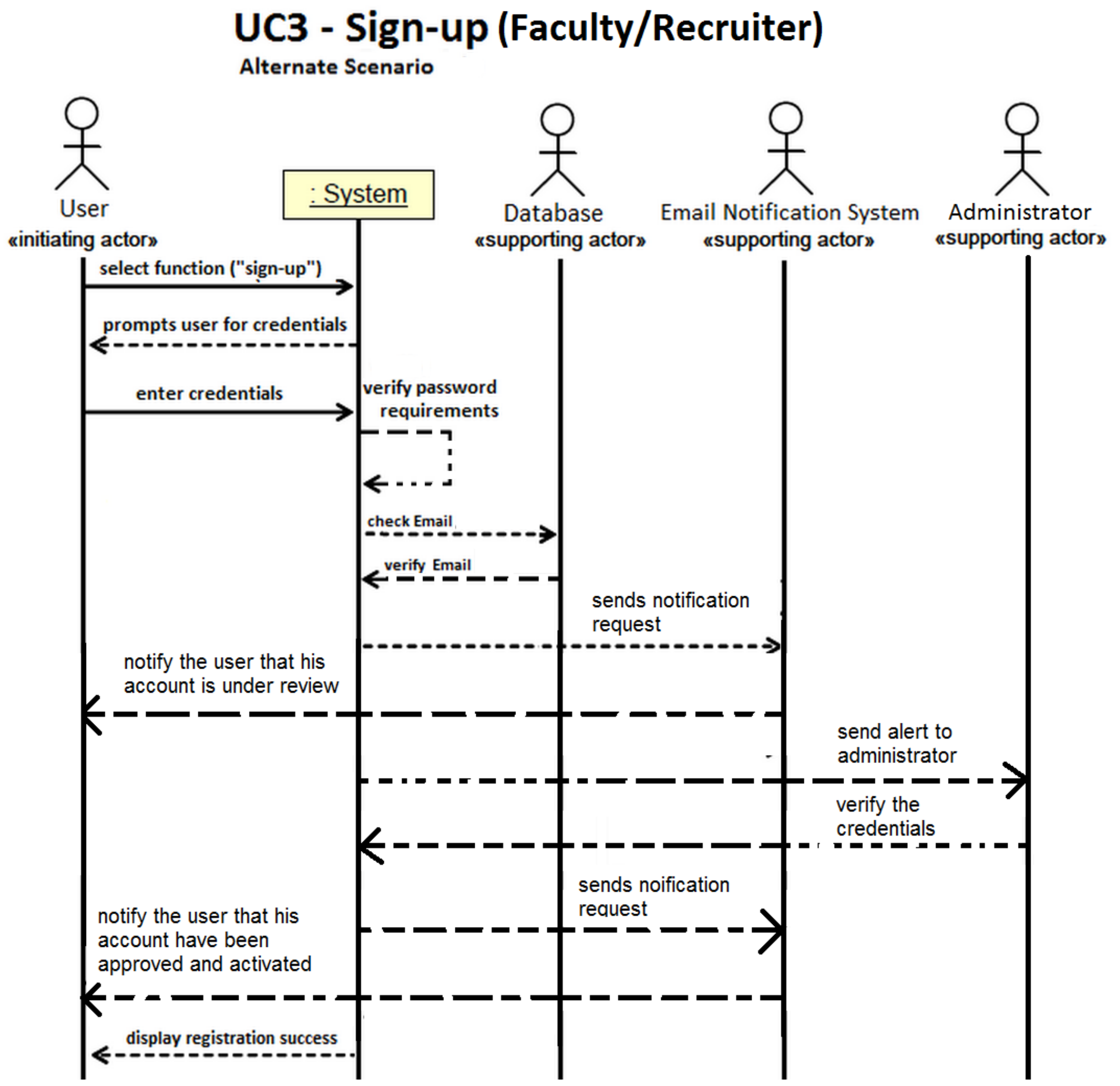

UC3 - Sign-up (Faculty/Recruiter)
Alternate Scenario
User
«initiating actor»
: System
Database
«supporting actor»
Email Notification System
«supporting actor»
Administrator
«supporting actor»
select function ("sign-up")
prompts user for credentials
enter credentials
verify password requirements
check Email
verify Email
sends notification request
notify the user that his account is under review
send alert to administrator
verify the credentials
sends noification request
notify the user that his account have been approved and activated
display registration success

Figure 4.4.10 - UC3: Sign Up (Alternate Scenario 3 - Faculty/Recruiter Signup)
*Note: For faculty/recruiter to sign up in Creddit, it will take 2-3days to review and verify.

Similar to the main success scenario of Sign Up as seen in Figure 4.4.7, System will redirect User to a form. If signing up for Faculty account, user must supply proof of employment. If signing up for Recruiter account, company email is required. User will submit the completed form. System will verify the information and check with the database to avoid duplicate accounts. Once verified by the database, the system will notify the User that the account is under review and alert the administrator to confirm the account. Once the administrator verifies the credentials, they will send a notification to the system to activate the account and notify the user.

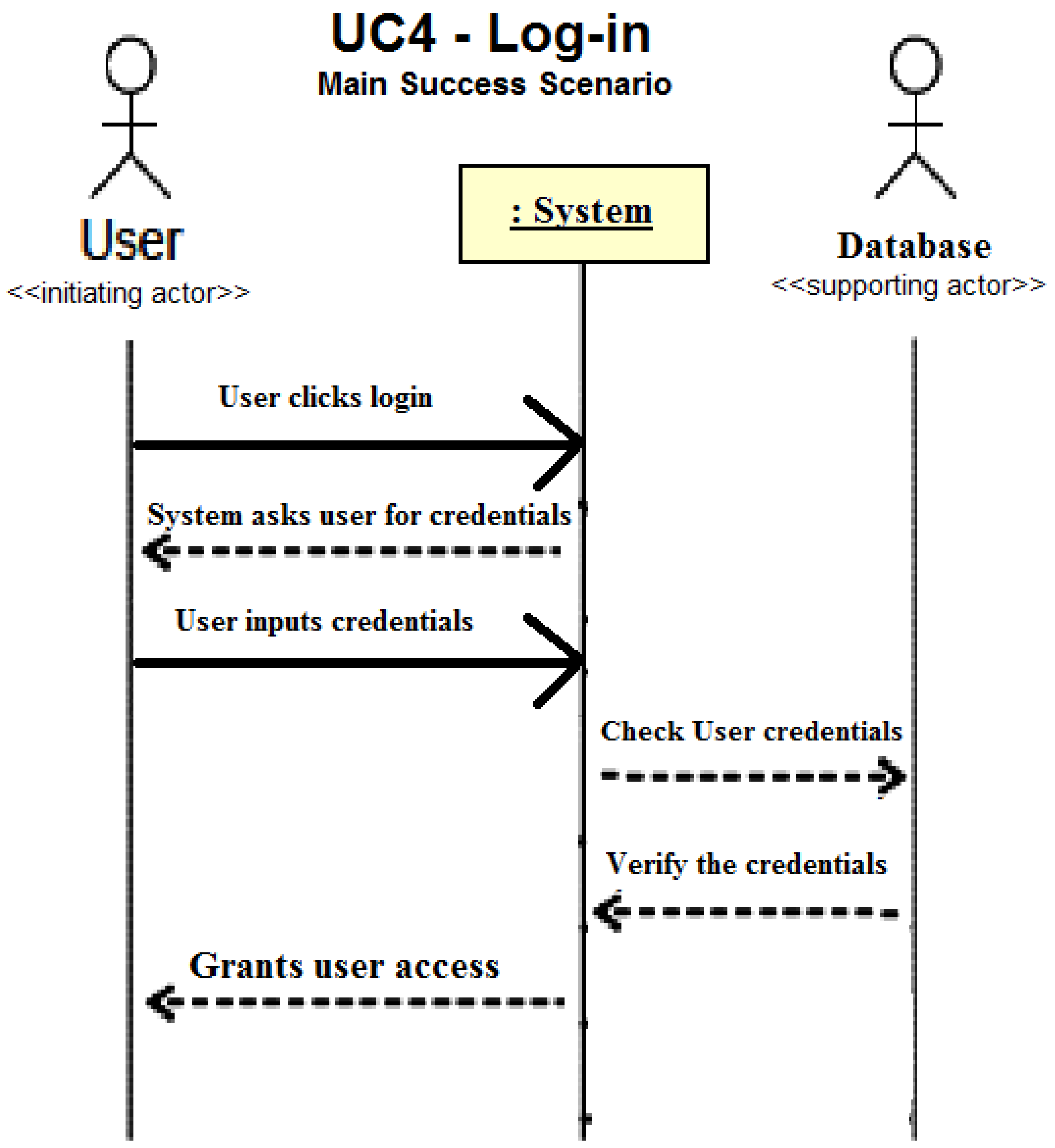


Figure 4.4.11 - UC4: Login (Main Success Scenario)

User will click on the login link and the system will redirect them to a login form. User will enter the required credentials and submit the form. System will verify the information and check with the database to confirm user information. Once verified by the database, user is granted access to account and Creddit.

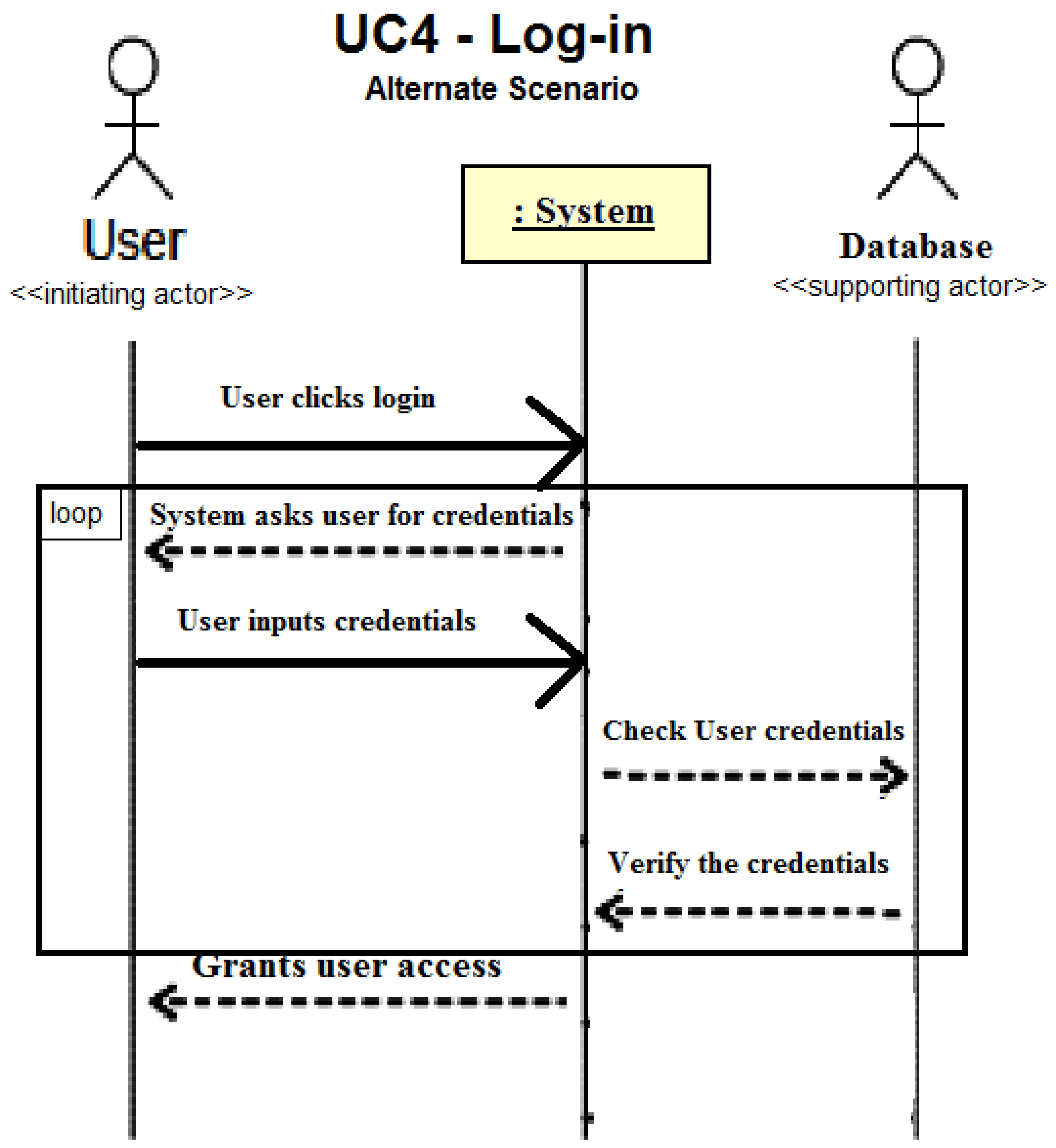


Figure 4.4.12 - UC4: Login (Alternate Scenario 1 - incorrect credential)

Similar to the Main Success Scenario in Figure 4.4.11, System will redirect user to a login form. User will enter the required credentials and submit the form. System will verify the information and check with the database to confirm user information. If the information is not valid, access will be denied and will as user to re-enter login information. Once verified by the database, user is granted access to Creddit account.

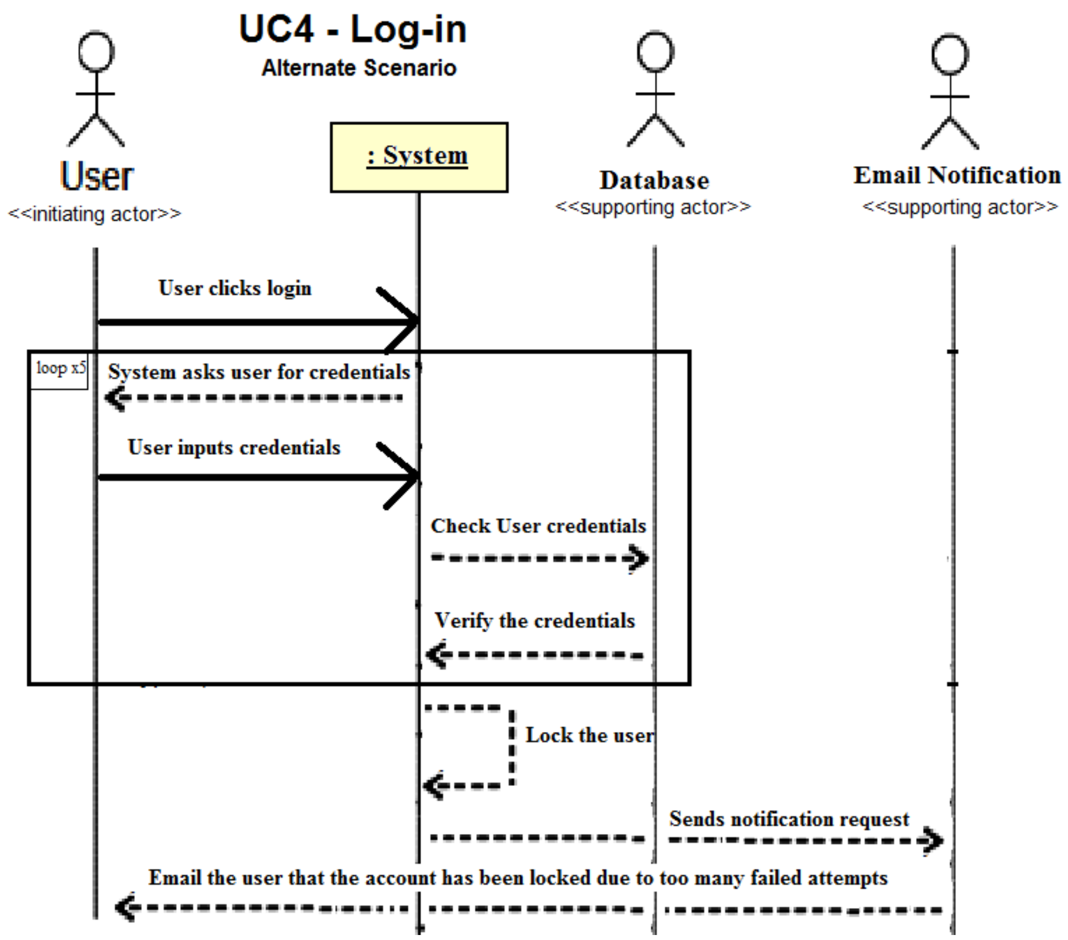


Figure 4.4.13 - UC4: Login (Alternate Scenario 2 - account lock)

Similar to the Alternate Scenario in Figure 4.4.12, System will redirect user to a login form. User will enter the required credentials and submit the form. System will verify the information and check with the database to confirm user information. If the information is not valid for more than 5 times, access will be denied and the account will be locked. System will request the Email Notification to send an email informing the owner of the UserID/email that the account is locked due to multiple failed attempts.

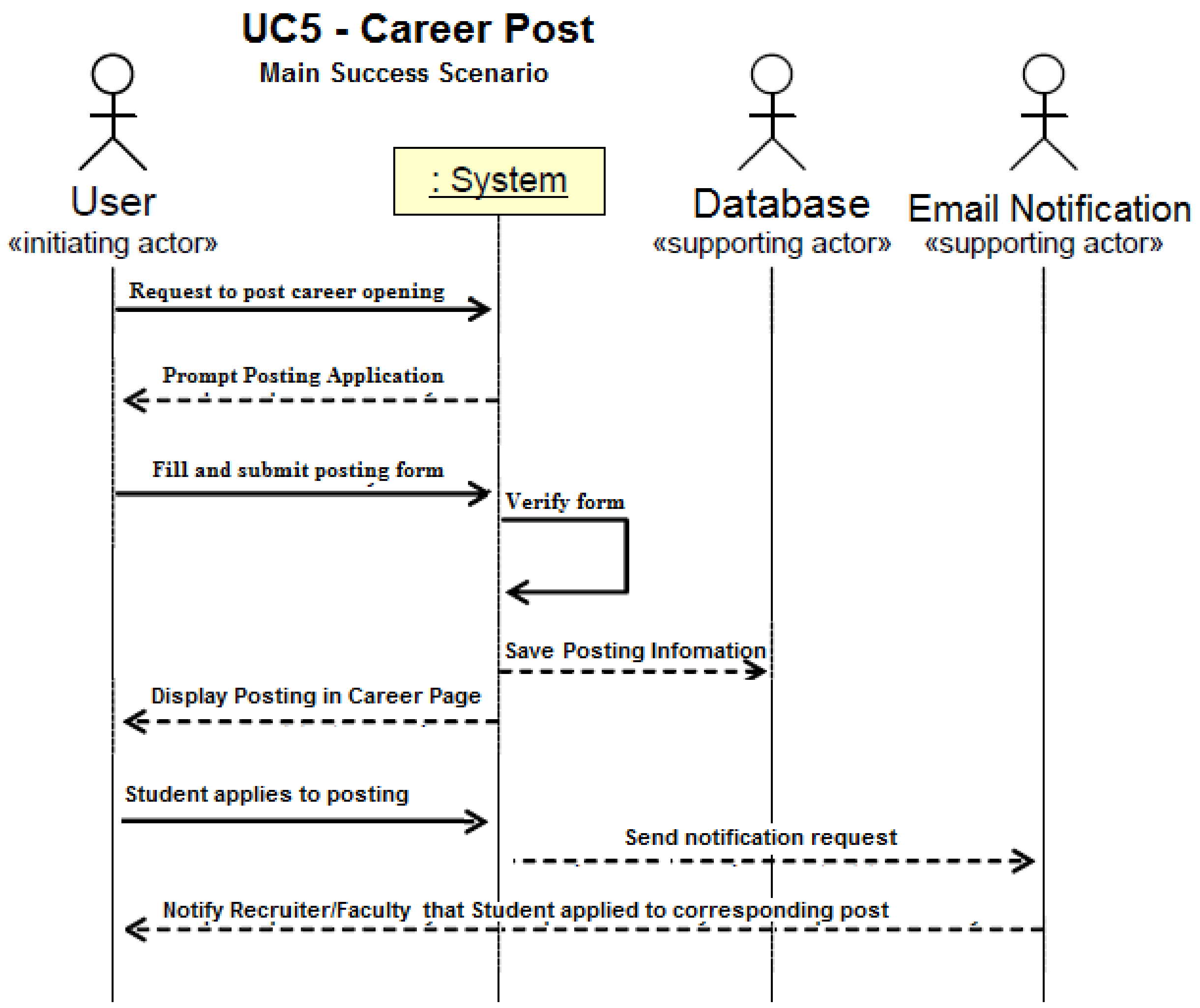

UC5 - Career Post
Main Success Scenario
User
«initiating actor»
: System
Database
«supporting actor»
Email Notification
«supporting actor»
Request to post career opening
Prompt Posting Application
Fill and submit posting form
Verify form
Save Posting Infomation
Display Posting in Career Page
Student applies to posting
Send notification request
Notify Recruiter/Faculty that Student applied to corresponding post

Figure 4.4.14 - UC5: Career Post (Main Success Scenario)

Valid users will go to the Career Page and the system will load and display the career postings. Users request to create a new career opening. System will prompt a posting form, and Users will fill and submit the form. System will verify the form and will store the post into the database. System will also display the new post in the Career page. If Student User applies to the post, the system will request the Email Notification to notify the owner of the post.

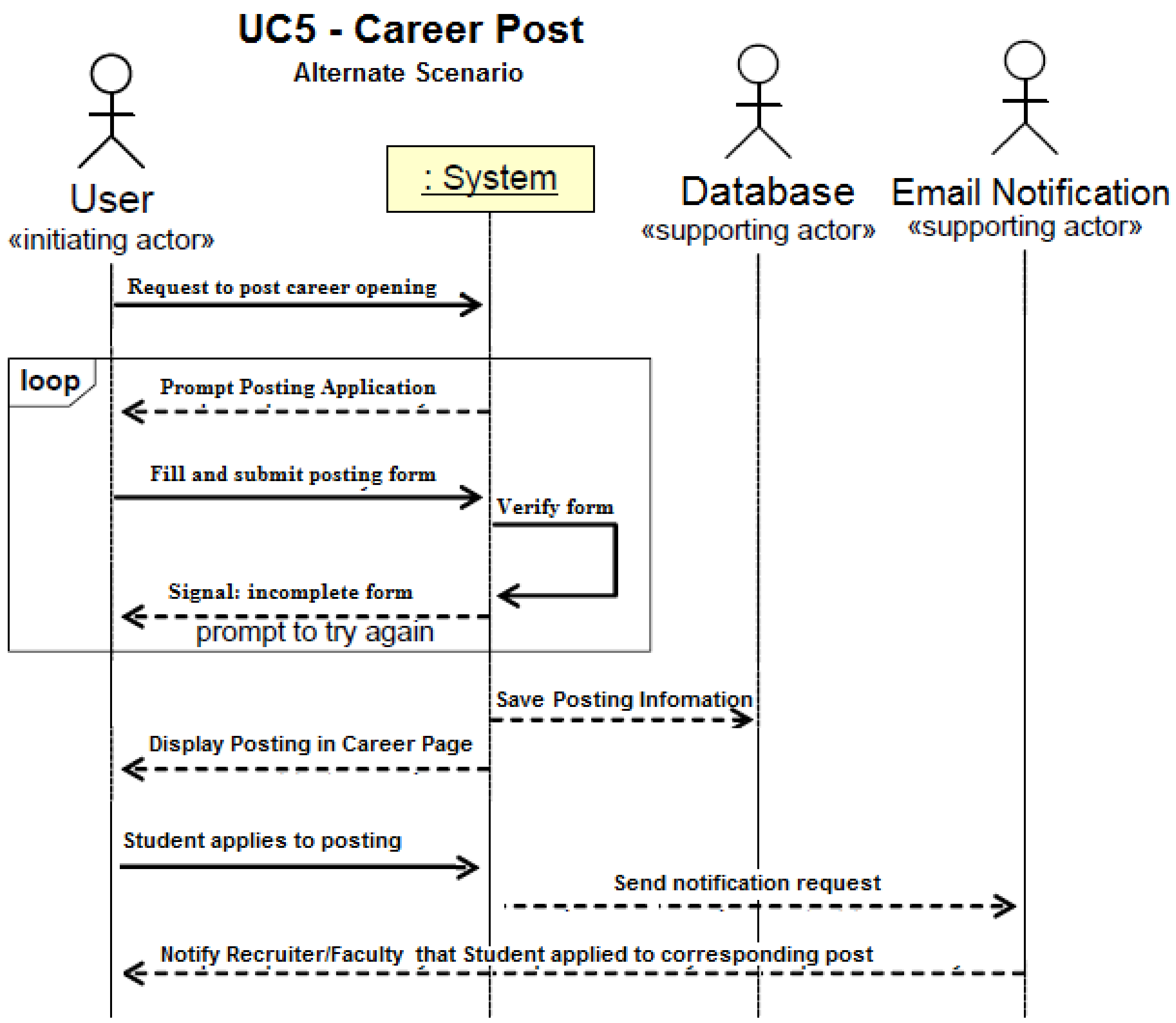

UC5 - Career Post
Alternate Scenario
User
«initiating actor»
: System
Database
«supporting actor»
Email Notification
«supporting actor»
Request to post career opening
loop
Prompt Posting Application
Fill and submit posting form
Verify form
Signal: incomplete form
prompt to try again
Save Posting Infomation
Display Posting in Career Page
Student applies to posting
Send notification request
Notify Recruiter/Faculty that Student applied to corresponding post

Figure 4.4.15 - UC5: Career Post (Alternate Scenario 1 - incomplete form)

Similar to the Main Success Scenario in Figure 4.4.14, Valid Users request to create a new career opening. System will prompt a posting form, and Users will fill and submit the form. System will verify the form and if the form is incomplete, the system will notify the user to complete the required credentials. Once completed, the system will store the post into the database and display the new post in the Career page. If Student User applies to the post, the Email Notification will notify the owner of the post.

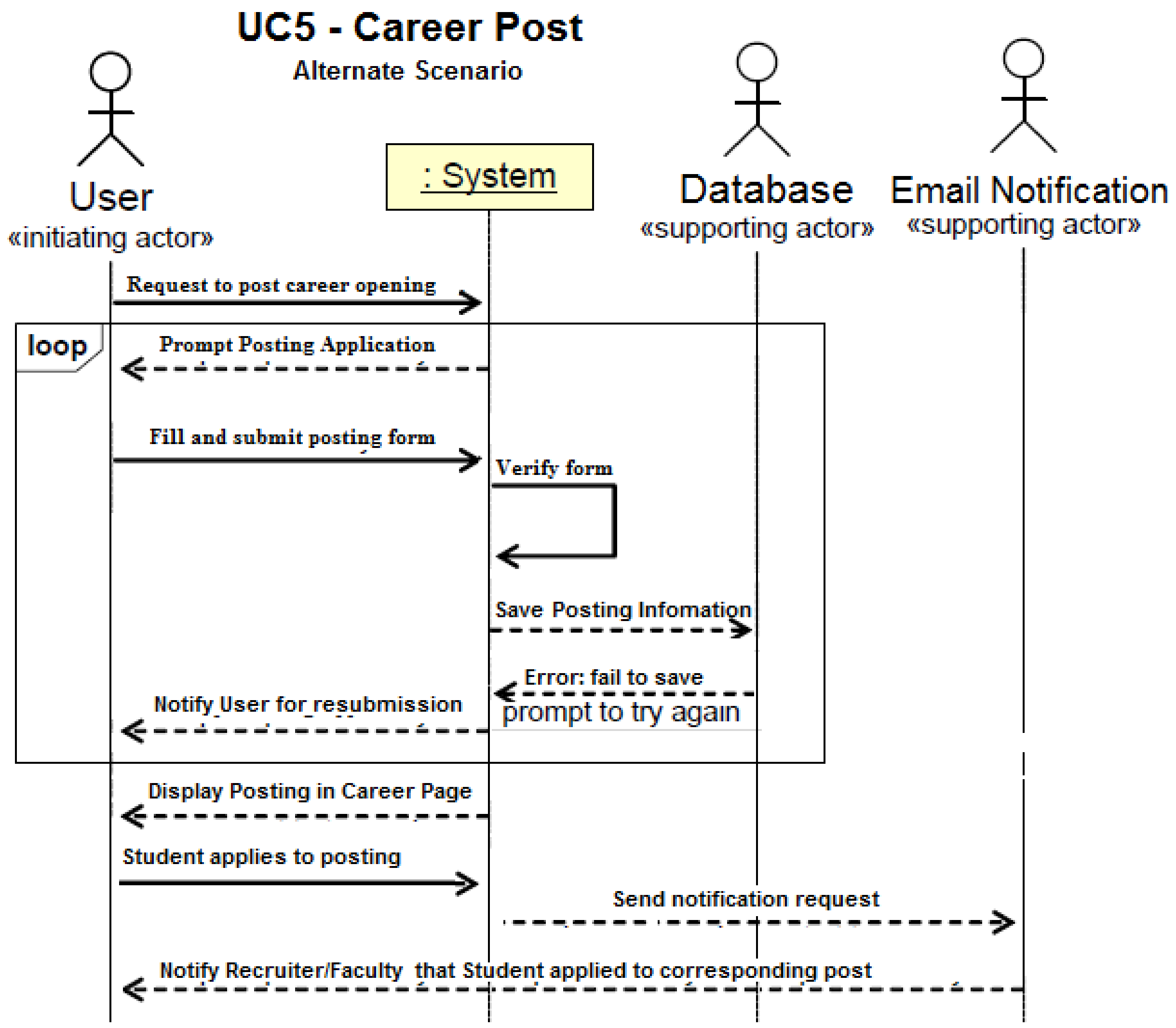

UC5 - Career Post
Alternate Scenario
User
«initiating actor»
: System
Database
«supporting actor»
Email Notification
«supporting actor»
Request to post career opening
loop
Prompt Posting Application
Fill and submit posting form
Verify form
Save Posting Infomation
Error: fail to save
prompt to try again
Notify User for resubmission
Display Posting in Career Page
Student applies to posting
Send notification request
Notify Recruiter/Faculty that Student applied to corresponding post

Figure 4.4.16 - UC5: Career Post (Alternate Scenario - saving failure)

Similar to the Alternate Scenario in Figure 4.4.15, Valid Users request to create a new career opening. System will prompt a posting form, and Users will fill and submit the form. System will verify the form and if the form is incomplete, the system will notify the user to complete the required credentials. Once the required fields are filled, the system will attempt to store the information into database. However, in this scenario, the system fails to store, and an error message will display and will ask for a resubmission.

4.5 Acceptance Tests for Use Cases

Acceptance Tests for Use Cases are similar to Acceptance Test Cases in Chapter 4.5. These tests are employed so the users can decide whether to accept the system or return it for further development. It specifies the step-by-step of how the user interacts with the system and what is the system expected to do. The following test cases below are selected from the use cases described above in Chapter 5.3.1. These test cases are selected with reasons as stated from above in Chapter 5.3.4. Acceptance tests are not limited to the ones described below.

Test-Case Identifier: TC-1.01 **Use Case Tested:** UC-1: Posting in Forums - main success scenario **Pass/Fail Criteria:** The test passes if system saves user post on to forum and that post can be accessed/viewed/replied by other student and faculty users. **Input Data:** Post	
Test Procedure:	**Expected Results:**
Setup: User must be Student, Faculty, or Admin	
Step 2. Submit blank post to post field.	Website displays “Please fill out the post field.” and doesn’t not process the request.
Step 3. Submit incomplete post, not all fields are filled.	Website displays “Please complete all fields.” and doesn’t not process the request.
Step 4. Submit a valid post.	Website redirects user to the site with contained post.

Figure 4.5.1
Test Case diagram for the main success scenario of posting in the forums.

Test-Case Identifier: TC-1.02 **Use Case Tested:** UC-1: Posting in Forums - alternate scenario **Pass/Fail Criteria:** The test passes if system saves user comment on to forum and that comment can be accessed/viewed/replied by other student and faculty users. **Input Data:** Post	
Test Procedure:	**Expected Results:**
Setup: User must be Student, Faculty, or Admin	
Step 1. Submit blank comment to comment field.	Website displays “Please fill out the comment field.” and doesn’t not process the request.

Step 2. Submit a valid post.	Website redirects user to the site with contained post.

Figure 4.5.2
Test Case diagram for the alternate scenario of commenting in the forum.

Test-Case Identifier: TC-2.01 **Use Case Tested:** UC-2: Signing up for Tutoring - main success scenario **Pass/Fail Criteria:** The test passes if the user selects an available time and date from a chosen tutor. **Input Data:** None	
Test Procedure:	**Expected Results:**
Setup: User must be a Student.	
Step 1: User selects subject that they want tutoring in.	Website displays subcategories within that subject.
Step 2: User selects subcategory that they require help in.	Website displays available tutors for that subcategory.
Step 3: User selects available tutor.	Website displays the page of the selected tutor, including subject, rating, reviews, and schedule.
Step 4: User selects available date from schedule listed.	Website displays available times for that date.
Step 5: User selects available time from the times listed.	Website signs student up for that time and displays a confirmation message.

Figure 4.5.3
Test Case diagram for the main success scenario of signing up for a tutor.

Test-Case Identifier: TC-2.02 **Use Case Tested:** UC-2: Signing up for Tutoring - alternate scenario **Pass/Fail Criteria:** The test fails if the user attempts to select an unavailable date or time. **Input Data:** Email address, Password	
Test Procedure:	**Expected Results:**

Setup: User must be a Student. Step 1: User selects subject that they want tutoring in. Step 2: User selects subcategory that they require help in. Step 3: User selects available tutor. Step 4: User selects a date that is booked. Step 5: User selects available date from schedule listed. Step 6: User selects a time that is booked. Step 5: User selects available time from the times listed.	Website displays subcategories within that subject. Website displays available tutors for that subcategory. Website displays the page of the selected tutor, including subject, rating, reviews, and schedule. Website displays an error message, asking for the user to select an available date. Website displays available times for that date. Website displays an error message, asking the user to select an a Website signs student up for that time and displays a confirmation message.

Figure 4.5.4
Test Case diagram for the alternate scenario of signing up for a tutor.

Test-Case Identifier: TC-3.01
Use Case Tested: UC-3: Sign Up - main success scenario
Pass/Fail Criteria: The test passes if the email address is not previously taken and the password meets the criteria.
Input Data: name, college/university, email address, date of graduation, major, classes currently taking, resume, and password.

Test Procedure:	**Expected Results:**
Step 1: User clicks on join	Website redirects the user to the signup form.
Step 2: User submits the blank fields of the signup form.	Website redirects the user to a confirmation page that says that a confirmation email has been sent to the respective email address.

Figure 4.5.5
Test Case diagram for the main success scenario for signing up for an account on Creddit.

Test-Case Identifier: TC-3.02 **Use Case Tested:** UC-3: Sign Up - alternate scenario **Pass/Fail Criteria:** The test fails if the password does not meet the criteria. **Input Data:** name, college/university, email address, date of graduation, major, classes currently taking, resume, and password.	
Test Procedure:	**Expected Results:**
Step 1: User clicks on join	Website redirects the user to the signup form.
Step 2: User submits the signup form with mismatching password or password that does not meet the criteria.	Website displays "Incorrect password provided".
Step 3: System leaves the password field blank.	Website displays "Missing [corresponding] field."
Step 2: User re-submits the blank fields of the signup form, correctly.	Website redirects the user to a confirmation page that says that a confirmation email has been sent to the respective email address.

Figure 4.5.6
Test Case diagram for an alternate scenario for signing up for an account on Creddit.

Test-Case Identifier: TC-3.03 **Use Case Tested:** UC-3: Sign Up - alternate scenario **Pass/Fail Criteria:** The test fails if the email address is previously taken or is not .edu. **Input Data:** name, college/university, email address, date of graduation, major, classes currently taking, resume, and password.	
Test Procedure:	**Expected Results:**
Step 1: User clicks on join	Website redirects the user to the signup form.
Step 2: User submits the signup form with an email address that has previously been taken or is not .edu.	Website displays "This email is already taken or is not an educational email".
Step 3: System leaves the email field blank.	Website displays "Missing [corresponding] field."

Step 2: User re-submits the blank fields of the signup form, correctly.	Website redirects the user to a confirmation page that says that a confirmation email has been sent to the respective email address.

Figure 4.5.7
Test Case diagram for an alternate scenario for signing up for an account on Creddit.

Test-Case Identifier: TC-4.01
Use Case Tested: UC-4: Login - main success scenario
Pass/Fail Criteria: The test passes if the user inputs an email address and password that exists in the database.
Input Data: Email address, password

Test Procedure:	**Expected Results:**
Step 1: User clicks on login.	Website redirects the user to the login page.
Step 2: User submits the login form with his/her email address and password.	Website redirects the user to the main page with their corresponding permission.

Figure 4.5.8
Test Case diagram the main success scenario for logging into Creddit.

Test-Case Identifier: TC-4.02
Use Case Tested: UC-4: Login - alternate scenario
Pass/Fail Criteria: The test fails if the email and password do not match.
Input Data: Email address, password

Test Procedure:	**Expected Results:**
Step 1: User clicks on login.	Website redirects the user to the login page.
Step 2: User fills out the login form with a mismatching email and password.	Website displays “Your email and password do not match!”
Step 3: System leaves the email and password fields blank.	Website displays “Incorrect user credentials”.
Step 4: User re-submits the login form with his/her email address and password correctly.	Website redirects the user to the main page with their corresponding permission.

Figure 4.5.9
Test Case diagram for an alternate scenario for logging into Creddit.

Test-Case Identifier: TC-4.03 **Use Case Tested:** UC-4: Login - alternate scenario **Pass/Fail Criteria:** The test fails if the email and password do not match 5 times. **Input Data:** Email address, password	
Test Procedure:	**Expected Results:**
Step 1: User clicks on login. Step 2: User fills out the login form with a mismatching email and password. Step 3: System leaves the email and password fields blank. Step 4: Step 2 and Step 3 happen 5 times.	Website redirects the user to the login page. Website displays "Your email and password do not match!" Website displays "Incorrect user credentials". Website redirects the user to a page saying "You have incorrect credentials 5 times. Your account will be locked until you confirm your email" and locks the user account. The system will send an email to the corresponding email address.

Figure 4.5.10
Test Case diagram for an alternate scenario for logging into Creddit.

Test-Case Identifier: TC-5.01 **Use Case Tested:** UC-5: Career Post - main success scenario **Pass/Fail Criteria:** The test passes if the post is uploaded successfully in the Career Page after filling out all the requirements in the posting form. **Input Data:** Career descriptions	
Test Procedure:	**Expected Results:**
Setup: The user must be a Recruiter or Faculty. Step 1. Click on link to Career Page from User Dashboard.	Redirects user to a posting form with blank fields. Redirects user to Career Page and displays the new post in the list.

Step 2. Fill in the required blank fields and submit the form.	

Figure 4.5.11
Test Case diagram for the main success scenario for posting a career post.

Test-Case Identifier: TC-5.02 **Use Case Tested:** UC-5: Career Post - alternate scenario (2a) **Pass/Fail Criteria:** The test fails if the posting form does not reload and if the post is not uploaded in the Career Page. **Input Data:** Career descriptions	
Test Procedure:	**Expected Results:**
Setup: The user must be a Recruiter or Faculty.	
Step 1. Click on link to Career Page from User Dashboard.	Redirects user to a posting form with blank fields.
Step 2. Submit a blank form or unfinished blank fields.	Prompt user to fill in the form and refresh the posting form.
Step 3. Fill in the required blank fields and submit the form.	Redirects user to Career Page and displays the new post in the list.

Figure 4.5.12
Test Case diagram for the alternate scenario for posting a career post.

5. Effort Estimation

5.1 User Effort Estimation

Use Case	Action	Description	Total Clicks/Keystrokes
UC-1	Posting in Forum	Click on forums + click on topic/subject + click on the specific forum + click on comment box + click submit	5
UC-2	Signing up for Tutoring	Click on tutors + click on subject + click on topic + click on tutor + click on tutoring date + click join	6
UC-3	Sign up	Click sign up + click password + click name + click school + click school email + click password + click gender + click submit	8
UC-4	Login	Click login + click enter school email + click enter password + click login	4
UC-5	Career Post	Click post jobs + click enter information + click what types of majors sought after + click submit	4
UC-6	Using the Search Bar	Click on search bar + click search	2
UC-7	Creating a Chat	Click chat + click search person + enter to find person + click on correct person	5
UC-8	Using the Email system	Click on email + click on compose + click on receivers + click on topic + click on dialog box + click send	6
UC-9	Upvoting / Downvoting	Click upvote or downvote	2
UC-10	Becoming a tutor	Click sign in to be a tutor	1
UC-11	Point decay	None	0

UC-12	Point reset	None	0
UC-13	View Forum	Click on forums	1
UC-14	Saving Draft Forum Post	Click save	1
UC-15	Autosave Comments	None	0
UC-16	Update Schedule	Click update schedule + click date + click start time + click end time + click confirm change	5
UC-17	Check Tutoring Signups	Click check who signed up + click section	2

5.2 Developer Effort Estimation

User Case Points method provides the ability to estimate the person-hours a software project requires based on its use cases. **(UCP) = UUCP x TCF x ECF**

UUCP:

Unadjusted Use Case Points (UUCP) measures the complexity of the functional requirements.

UAW

The Unadjusted Actor Weight (UAW), based on the combined complexity of all the actors in all the use cases.

Actor Name	Description of relevant characteristics	Complexity	Weight
Guest	Guest interacts with our system via a graphical user interface.	Complex	3
Recruiter	Recruiter interacts with our system via a graphical user interface.	Complex	3
Faculty	Faculty interacts with our system via a graphical user interface.	Complex	3
Student	Student interacts with our system via a graphical user interface.	Complex	3

Admin	Admin interacts with our system via a graphical user interface.	Complex	3
Database	Database interacts with our system through a network communication protocol.	Average	2

UAW = (5 x complex) + (1 x average) = 5(3) + 1(2) = 17

UUCW

The Unadjusted Use Case Weight (UUCW), based on the total number of activities (or steps) contained in all the use case scenarios.

Use Case	Action	Description	Category	Weight
UC-1	Posting in Forum	Simple user interface.	Average	10
UC-2	Signing up for Tutoring	Complex user interface.	Average	10
UC-3	Sign up	Simple user interface.	Complex	15
UC-4	Login	Simple user interface.	Average	10
UC-5	Career Post	Simple user interface.	Average	10
UC-6	Using the Search Bar	Complex user interface.	Simple	5
UC-7	Creating a Chat	Simple user interface.	Average	10
UC-8	Using the Email system	Simple user interface.	Average	10
UC-9	Upvoting / Downvoting	Simple user interface.	Simple	5
UC-10	Becoming a tutor	Average user interface.	Simple	5
UC-11	Point decay	Simple user interface.	N/A	0
UC-12	Point reset	Simple user interface	N/A	0

UC-13	View Forum	The user accesses the forums	Simple	5
UC-14	Saving Draft Forum Post	The user have the option to save their posting as a draft, which will be saved in a listbox for later access.	Simple	5
UC-15	Autosave Comments	The system will automatically save the drafts of the comments incase of any system downs or window exits.	N/A	0
UC-16	Update Schedule	The Tutor updates their schedule based on available times for teaching.	Average	10
UC-17	Check Tutoring Signups	The Tutor views the list of Students signed up for a specific tutoring session.	Simple	5

UUCW = (1 x complex) + (7 x average) + (6 x simple) = (1 x 15) + (7 x 10) + (6 x 5) = 115
UUCP = UAW + UUCW = 17 + 115 = 132

TCF:
Technical Complexity Factor (TCF)—Nonfunctional Requirements

Technical Factor	**Description**	**Weight**	**Perceived Complexity**	**Calculated Factor (WeightxPerceived Complexity)**
T1	Distributed web-based system	2	3	2 x 3 = 6
T2	User should expect good performance but nothing exceptional	1	3	1 x 3 = 3
T3	End-user should expect efficiency but nothing exceptional	1	3	1 x 3 = 3
T4	Internal processing is relatively simple	1	1	1 x 1 = 1
T5	No requirement for reusability	1	0	1 x 0 = 0
T6	No need to install	0.5	0	0.5 x 0 = 0

T7	Should be easy to use	0.5	5	0.5 x 5 = 2.5
T8	No portability concerns	2	0	2 x 0 = 0
T9	Easy to change is minimally required	1	1	1 x 1 = 1
T10	Concurrent use is required	1	4	1 x 4 = 4
T11	User security is important	1	5	1 x 5 = 5
T12	No direct access for third parties	1	0	1 x 0 = 0
T13	No user training required	1	0	1 x 0 = 0
Total				25.5

Constant-1 = 0.6
Constant-2 = 0.01
TCF = Constant-1 + Constant-2 x Technical Factor Total = 0.6 + (0.01 x 25.5) = 0.855

ECF:
Environmental complexity factors (ECF) measure the experience level of the people on the project and the stability of the project.

Environmental Factor	**Description**	**Weight**	**Perceived Impact**	**Calculated Factor (WeightxPerceived Complexity)**
E1	Beginner familiarity with UML-based development	1.5	1	1.5 x 1 = 1.5
E2	Beginner familiarity with application problem	0.5	1	0.5 x 1 = 0.5
E3	Some knowledge of object-oriented approach	1	2	1 x 2 = 2

E4	Beginner lead analyst	0.5	1	0.5 x 1 = 0.5
E5	Highly motivated, but some team members occasionally slack	1	4	1 x 4 = 4
E6	Stable requirements expected	2	5	2 x 5 = 10
E7	No part-time staff will be involved	-1	0	-1 x 0 = 0
E8	Programming language of average difficulty will be used	-1	3	-1 x 3 = -3
Total				15.5

Constant-1 = 1.4
Constant-2 = 0.03

ECF = Constant-1 + Constant-2 x Environmental Factor Total = 1.4 + (0.03 x 15.5) = 1.865

UCP = UUCP x TCF x ECF = 132 x 0.855 x 1.865 = 210.4839

If we assume we have a productivity factor (PF) of 30:
Effort estimation (units are person-hours)= 210.4839 x 30 = 6314.517 person-hours

6. Domain Analysis

Domain Model

6.1 Concept Definitions

Concept Name	Type	Concept Definition
	D	Prepare a database query that retrieves the user's requests and best matches the user's search criteria and retrieve the records from the database.
Email Notifier	D	Notifies the user when another user responds to the post via email, and when user is promoted to be a Tutor. Also, sends a confirmation email to a new user so he may start his adventure on Creddit
Private Chat Notifier	D	Notifies the user when he/she receives a private chat via Creddit.
Forum Notifier	D	Notifies the user when another user responds to the post via Creddit.
Login Information	K	Container for user's authentication data, such as password and username.
Login Checker	D	Verifies whether or not LogIn data entered by the user is valid.
Interface Page	D	Creates/ displays a page that shows the user the current context, what actions can be done, and outcomes of the previous actions.
Postprocessor	D	Filter the retrieved records to match the actor's search criteria.
Archiver	D	Assigns a request (i.e., log in credentials, posts, messages, documents) a tracking number and sends it to the database.
Investigation Request	K	List of "interesting" records for further investigation, complaint description, and the tracking number.
Sign Up Checker	D	Verifies the user's ID and the email has not been used.
Advance Search	K	Form specifying the search parameters for database log retrieval
Vote Maintainer	D	Increases or decreases the student's or tutor's points in the database.
Promoter	D	Promotes a student to tutor if the student passes the interview

Login Locker	D	Locks the user account after 5 failed attempts of login.
Account type Notifier	D	The account type notifier notifies an administrator to verify the student/faculty/recruiter account
Point Collector	K	Container for all the points in the database
Tutor Calendar Availability Investigator	D	Verifies if the tutor has a spot for students available for a given time slot on a given day
Post Deleter	D	Deletes a post if the user decides to delete his own post
Report Collector	K	Container for all the reports.
Password Resetter	D	Resets the password when the user clicks “Forgot Password”
Tutor Time Register	D	Register a student for a tutor’s time slot if it is available
Post Creator	D	Creates a new page with a post if the user decides to create a post.
Post View Counter	K	Container that has the number of points the post has received.
Signout	D	Logs the user out of his account
Tutor Arrival	D	Verifies that the tutor shows up to tutor at his designated times.
Post Comment Counter	K	Container of the number of comments a post has received
Student Arrival	D	Verifies that the student has showed up to the tutoring session
Student Remover	D	Remove student from the remaining tutoring session if the student fails to show up to the tutoring session three times.
Tutor Remover	D	Remove tutor status from student user If tutor fails to show up to two tutoring sessions without a valid reason
Tutor Schedule Editor	D	Allows the tutor to cancel / change the time of certain tutor sessions to allow for flexibility in the event that an unforeseen circumstance should arise
Student Schedule Conflict Notifier	D	Allows the student to notify the tutor that he will not be able to attend a session.
Tutor Rating System	D	Allows students to rate a tutor that he has been taught by.

Page Deleter	D	Deletes a page
Comment Deleter	D	Allows the User to delete his own comments

Types **"D"** and **"K"** denote *doing vs. knowing* responsibilities, respectively.

6.2 Association Definitions

Concept Pair	**Association Description**	**Association Name**
Interface Page ←→ Database Connection	Database Connection passes the retrieved data to Interface Page to render data for user display.	provide data
Archiver ←→ Investigation Request	Archiver creates/generates Investigation Request	generates
Archiver ←→ Database Connection	Archiver requests Database Connection to store investigation requests into the database	requests save
Database Connection ←→ Login Checker	Log in checker requests Database Connection to verify the user credentials.	requests log in
Archiver ←→ Email Notifier	Archiver requests Email Notifier to send an email to user about confirmations, requests, and replies.	requests notify
Archiver ←→ Private Chat Notifier	Archiver requests Private Chat Notifier to notify user a private chat message.	requests notify
Archiver ←→ Forum Notifier	Archiver requests Forum Notifier to notify user of an update in their forum post.	request notify
Advance Search ←→ Database Connection	Advance search requests for records from the database.	searches
Login Information←→ Login Checker	Login searcher validates user input and sends it to login Checker for validation.	requests login
Sign Up ←→ Database Connection	Sign up verifies user information and sends it to database to process/add user.	requests sign up
Post Comment Counter ←→ Archiver	Records the comment counters of posts to the archiver to database.	stores comment counter
Post View Counter ←→ Database Connection	Records the view counters of posts to the archiver to database.	stores view counter

Tutor Rating System ←→ Database Connection	Tutor Rating system request Database Connection to store new tutor ratings in the database	requests save
Post Creator ←→ Archiver	Notifies the Archiver of a new post to be assigned a tracking number.	requests notify
Student Arrival ←→Student Remover	Student arrival sends a notification to the remover to validate that the student should be removed from the tutoring session	requests notify
Tutor Arrival ←→ Tutor Remover	Tutor checking sends a notification to the Tutor Remover to validate that the tutor should have his status as a tutor removed	requests notify
Archiver ← → Vote Maintainer	Archiver requests vote maintainer to update the user's points accordingly	requests save
Login Checker←→ Login Locker	Login Checker notifies the Login Locker after the user has entered the wrong password 5 times	requests notify
New User Verifier ←→ Email Notifier	Verifier sends a request for notifier to send a confirmation email to new user.	requests notify
Vote Maintainer ← → Point collector	Vote Maintainer updates the user's points accordingly in point collector.	requests save
Student Schedule Conflict Notifier ←→ Database Connection	The Student Schedule Conflict Notifier notifies the Database connection that a student has a conflict with his tutoring schedule	requests notify
Tutor Calendar Availability Investigator ←→ Database Connection	Verifies with the database connection if there is a spot available at a certain time and date for the student	requests notify
Sign out ←→ Database Connection	Sign out notifies database the user is logging out	requests logout
Report Collector ←→ Database Connection	Report collector request Database Connection to store all the reports in the database	requests save
Email Notifier ←→ Database Connection	Email Notifier verifies with the Database Connection if a student has accumulated enough point to be eligible to apply for tutoring	requests verification
Promoter ←→ Database Connection	Promoter verifies with the Database Connection that the student is now a tutor	requests verification

Tutor Schedule Editor ←→ Database Connection	Allows the tutor to change his/her schedule in the Database.	requests change in database
New User Verifier ←→ Email Notifier	New User Verifier notifies the Email notifier to send a confirmation email to the user	requests notify
Post Deleter ←→ Page Deleter	Post Deleter notifies Page Deleter that the post has been deleted and to delete the page	requests notify
Tutor Time Register ←→ Tutor Calendar Availability Investigator	Tutor Time Register request from the Tutor Calendar Availability Investigator if the time slot on that day is available	request notify
Password resetter ←→ Login Information	Password resetter requests login information even though the user is already signed in to verify whether or not the user truly desires to reset the password	requests login
Interface Page ←→Comment deleter	Interface page prepares the option that allows you to delete your comment	prepares
Archiver ←→ New User Verifier	When new user is being verified archiver requests for the new user's information that has been entered	requests verification
Advance Search ←→ Postprocessor	Advance search notifies the postprocessor to filter the data that closely matches what the user wants.	requests database
Tutor time register ←→ Tutor calendar availability investigator	Student registers for tutoring and tutor calendar availability investigator checks if there is an available spot	requests verification
Comment Deleter ←→ Database Connection	Comment Deleter notifies the Database Connection that the comment has been deleted	request notify
Sign Up Checker←→ New User Verifier	The sign up checker notifies the new user verifier so that the system can in fact create a new user and add him/her to the database	request database

6.3 Attribute Definitions

Concept	Attribute	Attribute Definition
Notifier (general - Forum, Email, Private Chat)	Tracking Number	Assigns a tracking number to an event (private chat, comment, etc.) to store information to the database and allows tracking on popular/trending/critical investigation statuses, posts and activities.
Archiver		
Investigation Request		
Interface Page	User Interface	Displays interface to deal with interaction with user and data.
Login Information	Data Logging	Data logging has to with the storage or retrieval of logged data or the logging of data.
Login Checker		
Login Locker		
Signout		
Password Resetter		
SearchRequest	Search Parameters	Form specifying the search parameters for database log retrieval and filter the retrieved records to match search criteria.
Postprocessor		
Database Connection	Data Storage	Data storage deals with the storage of the data.
Sign Up		
Vote Maintainer		
Post Creator		
Tutor Rating System		
Account Type Notifier	Data Verification	

New User Verifier		Temporary data stored and will not be stored in Main database until verified (either by other concept or admin).
Tutor Calendar Availability Investigator		
Notifier		
Student Schedule Conflict Notifier		
Student Arrival		
Promoter		
Post View Counter	Data Analyzing	A counter on specific data.
Tutor Arrival		
Point Collector		
Report Collector		
Post Comment Counter		
Student Remover	Data Remover	Data remover deals with the removal of the data.
Tutor Remover		
Password Resetter		
Page Deleter		
Comment Deleter		
Post Deleter		

6.4 Traceability Matrix

		Domain Concepts												
Use Case	Priority Weight	Database Connection	Page Maker	Email Notifier	Private Chat Notfier	Forum Notifier	Login	Log In Checker	Interface Page	Postprocessor	Archiver	Investigation Request	Sign Up Checker	Search Request
UC-1	39	X		X		X			X		X			
UC-2	27	X		X					X					
UC-3	25	X	X	X			X	X	X				X	
UC-4	29	X		X			X	X	X					
UC-5	29	X	X	X					X		X			
UC-6	26	X							X	X	X			X
UC-7	27	X			X				X					
UC-8	25	X				X			X					
UC-9	28	X							X		X			
UC-10	31	X		X					X					
UC-11	32	X		X										
UC-12	31	X	X	X					X					
UC-13	43	X							X		X			
UC-14	32	X							X		X			

UC-15	27	X							X		X			
UC-16	27	X							X					
UC-17	21	X	X						X					

		Domain Concepts												
Use Case	Priority Weight	Vote Request	Promoter	Tutor Promotion Notifier	New User Verifier	Login Locker	Faculty Account Notifier	Recruiter Account Notifier	Point Collector	Tutor Calendar Availability Investigator	Post Deleter	Report Collector	Password Resetter	Tutor Time Register
UC-1	39											X		
UC-2	27									X				X
UC-3	25				X		X	X						
UC-4	29					X							X	
UC-5	29													
UC-6	26													
UC-7	27													
UC-8	25		X	X	X	X	X	X					X	X
UC-9	28	X		X					X					

UC-10	31		X	X										
UC-11	32								X					
UC-12	31								X					
UC-13	43									X	X			X
UC-14	32													
UC-15	27													
UC-16	27									X				X
UC-17	21									X				

		Domain Concepts											
Use Case	Priority Weight	Post Creator	Post View Counter	Post Comment Counter	Signout	Tutor Checker - Tutoring	Student Checker - Tutoring	Student Remover-Tutoring	Tutor Remover - Tutoring	Tutor Schedule Editor	Student Schedule Conflict Notifier	Tutor Rating System	Comment Deleter
UC-1	39	X											
UC-2	27									X		X	
UC-3	25												
UC-4	29												

UC-5	29	X											
UC-6	26												
UC-7	27												
UC-8	25												
UC-9	28	X				X							X
UC-10	31					X	X	X	X	X	X	X	
UC-11	32				X								
UC-12	31												
UC-13	43		X	X									X
UC-14	32	X											
UC-15	27	X											X
UC-16	27					X	X			X	X		
UC-17	21				X						X	X	

7. Interaction Diagrams

Interaction diagrams show the behavior of objects and data in the system. It emphasize how data is moved and how objects communicate with each other.
The following interaction diagrams below are for use cases described in the Fully Dressed Descriptions (Section 5.3.4).

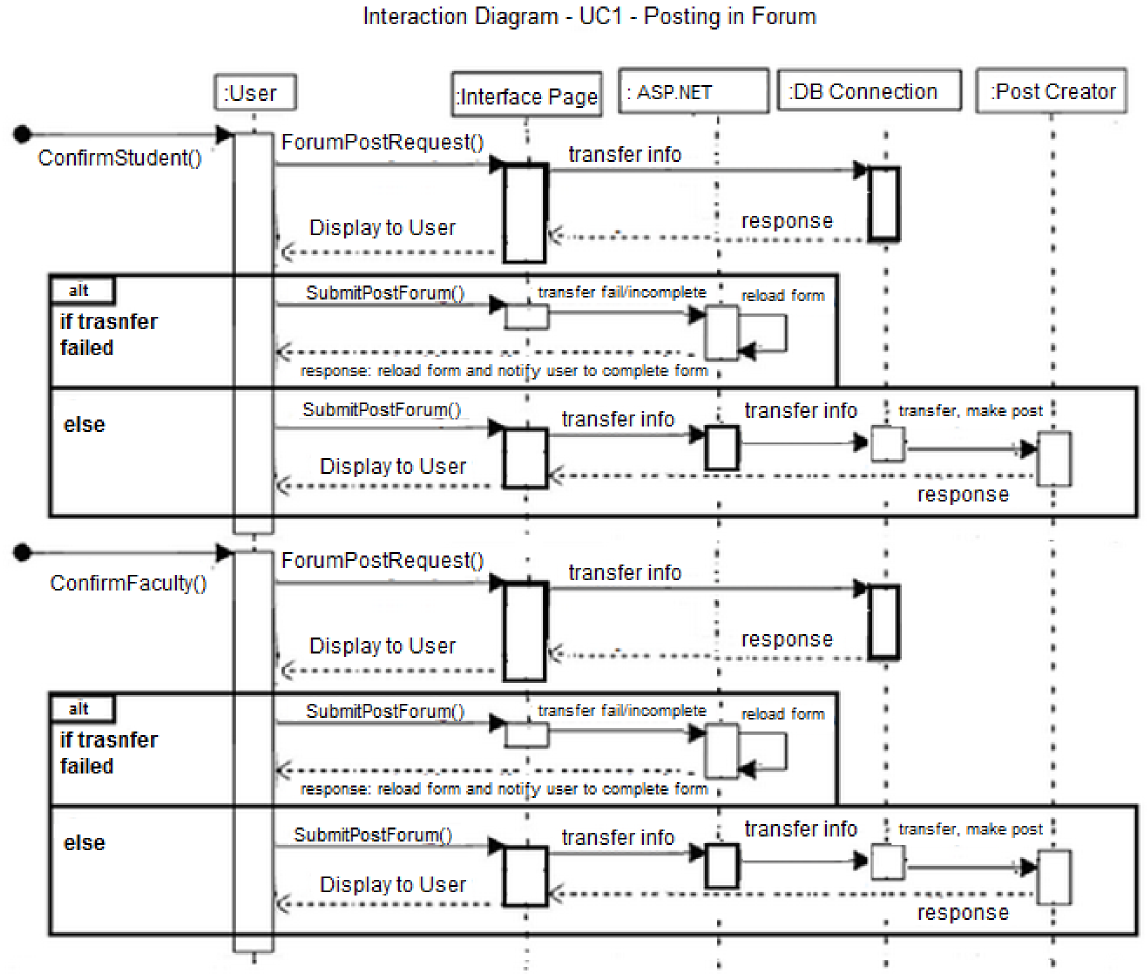


Figure 7.1 - Interaction Diagram for UC1 Posting in Forum

This sequence diagram shows the steps on how a user will post into the forum and how the objects will interact. First the forum posting sequence requires to confirm if a user is a student or faculty user to post as other users don't have the ability to post. After confirming, the user will interact with the interface page, which will send the database the information through ASP.NET. The interface page will get the response, then display it to the user with the data form post required for the user to fill out. The user will then fill out the form. If the form submitted is incomplete, the interface page will reload the page and require the user to adequately fill it out. This loop will continue until the form is filled out correctly. When the form is confirmed as completed, the info will be transferred to the archiver for a reference number, then created by the Post Creator.

The design pattern for Use Case 1 is Model View Controller (MVC) architectural pattern. The user submits a form to the controller, which will manipulate the data in the model server with the new data. Then the interface page will be updated with the new data that can be seen by the user. The framework is divided as a client server relationship.

However, since this was not discussed in class, the design pattern that is similarly to the use case is Publish-Subscribe messaging behavioral pattern. Both the receiver and sender do not know how many users can read their messages or who they are. Since this is a forum, everyone can read it. The poster is the publisher that will submit postings in the forum, and the users, who are the subscribers, can view the post once the submission is published in the forum page.

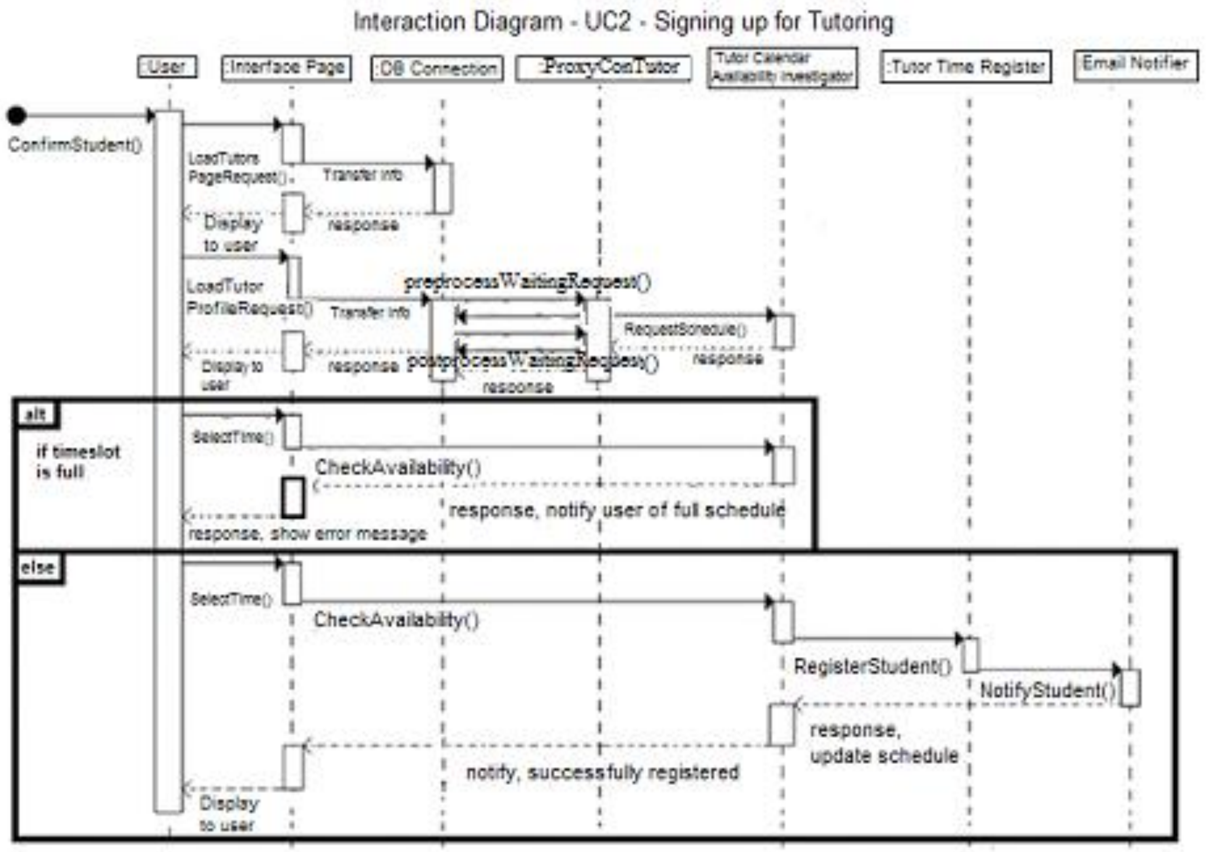


Figure 7.2 - Interaction Diagram for UC2 Signing up for Tutoring

The sequence diagram above was used to clearly illustrate the relations between the different objects of the system. Initially the system checks to see that the user is a student, since faculty and recruiters cannot sign up for tutoring. The Database is requested to load the list of tutors, which is returned as a response from the Interface Page. Once the student selects a tutor, this sends a request to the Database to load that tutor's profile. The user's schedule and rating are retrieved from the Tutor Rating System and the Tutor Calendar Availability Investigator, which are returned as a response through the Interface Page. On the tutor's profile, the student selects a time to register for. The database checks for availability through the Tutor Calendar Availability Investigator, and if the time is unavailable, the user is notified to select an available time. If the time is available, the student gets registered for that time slot through the Tutor Time Register, which in turn tells the Email Notifier to send an email to the student confirming the sign up. The student is also notified via a response through the Interface Page.

In this use case, we have revised the design of signing up to become a tutor to be a safer, more efficient and correct pattern so that it will now be able to intercept and preprocess requests so that all access to the real subject is optimized. This is why we have used the proxy pattern, which was one of the designs discussed in class, allowing our system to move forward in terms of design.

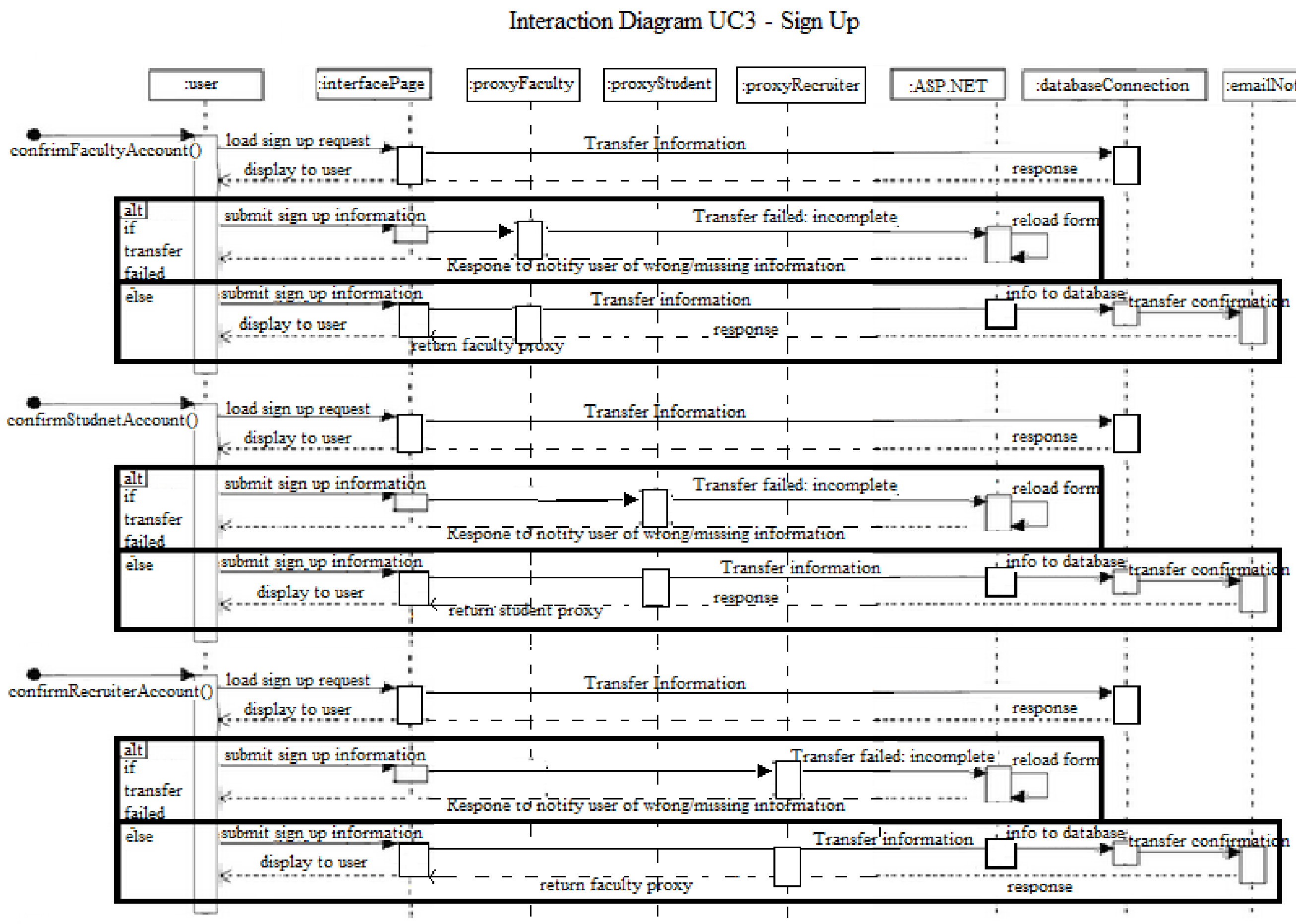


Figure 7.3 - Interaction Diagram for UC3 Signing Up

This sequence diagram shows the steps on how a user will sign up for Creddit. First the signing up sequence diagram requires to know which type of user you are (faculty, student, or recruiter). It will then load the signUpRequest and transfer the information to the interface page which will be returned as a response. The user that is signing up will then submit his or her information. If the transfer is a fail, it means it is incomplete or there is incorrect information, it will reload the page and tell the user what is wrong. If the sign up information is correct, it will transfer the information to the archiver. The archiver will then decide on which action to take and send an email for confirmation. Once the email has been confirmed there will be a notification of successfully signing up to the interface page which will be shown to the new user. We use the proxy design pattern in this interaction diagram for security reasons. This is to ensure that the three types of users (faculty, student, and recruiter) receive only the access that is permitted.

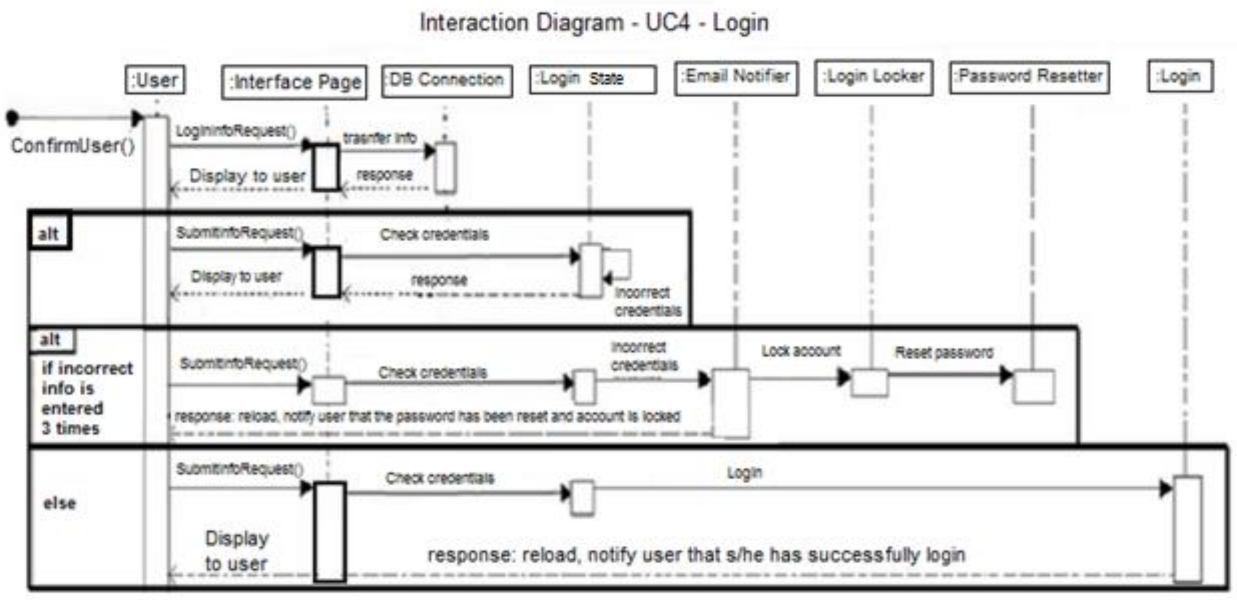


Figure 7.4 - Interaction Diagram for UC4 Login

The sequence diagram above shows the steps behind the scene of how a user will log into Creddit. The user is prompted to enter his information via LoginRequest() which the database connection transfers the information to the interface page. The Login Checker checks to verify If the user has entered in the incorrect password or not. If the incorrect information has been entered, the system responses back to the user asking him to resubmit the correct info. This will loop until the User has entered in the correct information or the User has entered in the incorrect information 3 times. If the user has entered the incorrect information 3 times, the system will notify the user that his account has been locked and the password needs to be reset. The Login Locker will lock the account that the user was trying to login to, and the Password Resetter will reset the password of the account. The Email notifier while send the a email asking to reset the password to the email of the locked account. If the User has entered in the correct information, the User will be login.

In this use case, Login Checker has been updated to Login state because this design essentially already entails the state pattern approach. Since the login state requires to be informed of the state and then respond whether or not the credentials that have been entered are correct. The state of the user would then be changed which is why the state pattern is implemented here and makes the design of the system safer.

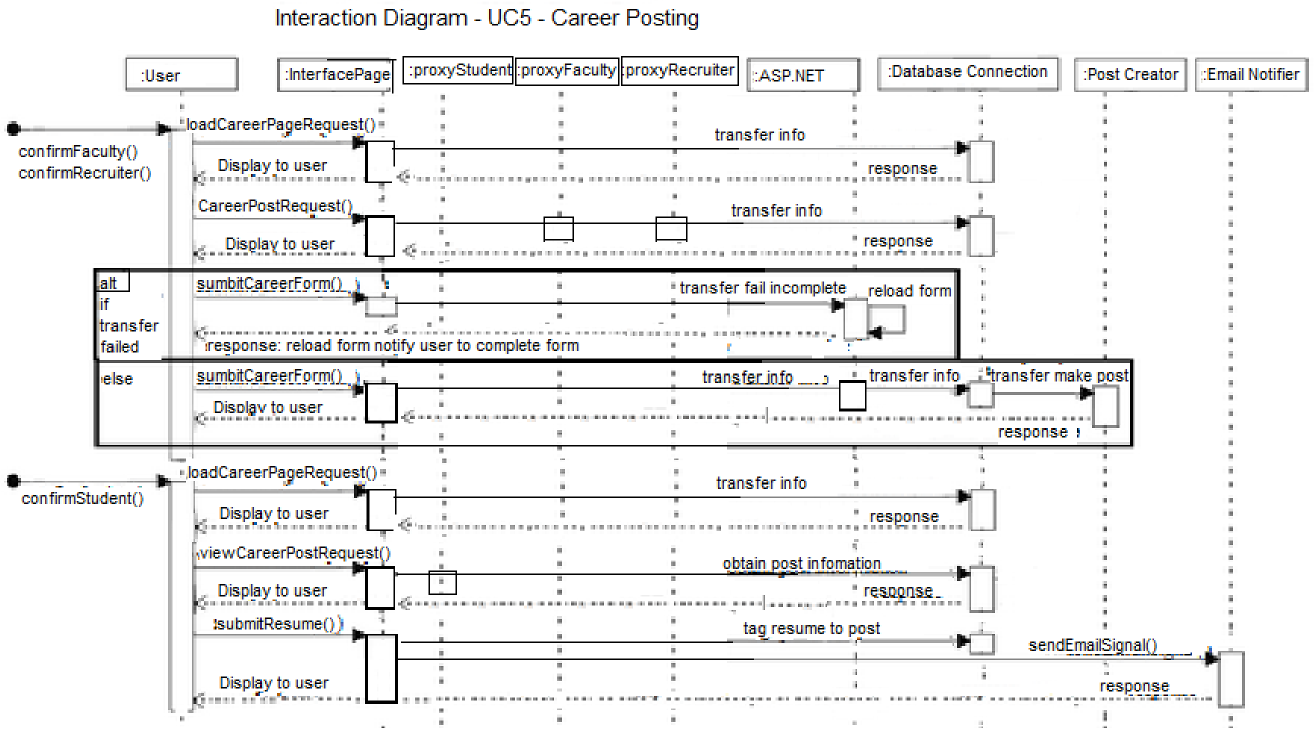


Figure 7.5 - Interaction Diagram for UC5 Career Posting

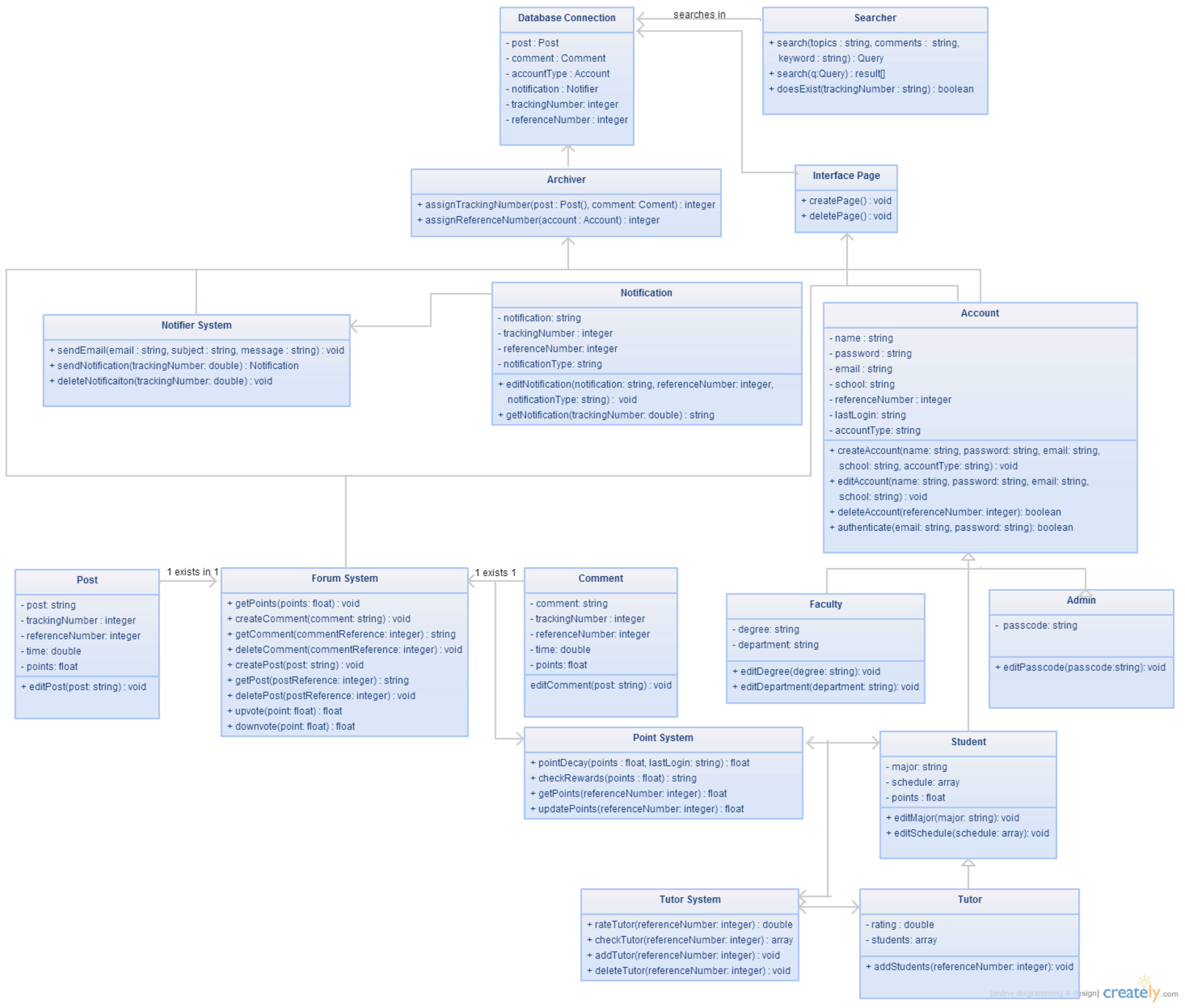
*Note: This feature is currently not implemented due to time constraints. Therefore, this feature will not be updated until further notice.

A sequence diagram is used for this use case because it clearly shows how the objects interact with each other. It illustrates the communication between the functions of the system and the user directly. Faculty or Recruiter users are allowed to access the Tutor page and post a new Career option. It sends a posting request to the database and it will sent back the posting form through the interface page. The user submits the completed form back to the database and the new data will be archived with a tracking number. This new data is now sent to the Post Creator to publish the new posting into the Career Page. It is important to note that a user cannot comment on these posts. They are strictly for viewing purposes and if a student is interested, then he/she would user would use the contact information given to find out more about the opportunity listed. If the user submitted an incomplete form, then the database will notify the user to re-submit the form. This loop will continue until the form is filled completely. The system would then ask whether or not the faculty user is satisfied with the given information he/she provided by a confirmation asking yes/no. Student users may access the Career Page with the database sending the posting information to the user through the interface page. This allows users to select a post to view, after the database obtains the information through the archiver. If Student submit resume to specific posting, database and archive will tag it to the corresponding post and notify Email notifier to send an email to the post owner. In the case that the position is a scam or not recommended, it could be flagged as inappropriate and that given post would be reviewed by the admin.

This use case is based under Proxy Design Pattern. Proxy is used for security reasons. It is to check if the user have access to certain privileges in the Career Page. The proxy is used to check if the user is a faulty or recruiter. If they are, then they have the privileges to post a career opportunity. If the user is a student, they have the access to submit resumes and view career posts. Otherwise non-Creddit users cannot access the page.

8. Class Diagrams and Interface Specification

8.1 Class Diagram



Database Connection
- post : Post
- comment : Comment
- accountType : Account
- notification : Notifier
- trackingNumber: integer
- referenceNumber : integer
searches in
Searcher
+ search(topics : string, comments : string, keyword : string) : Query
+ search(q:Query) : result[]
+ doesExist(trackingNumber : string) : boolean
Archiver
+ assignTrackingNumber(post : Post(), comment: Coment) : integer
+ assignReferenceNumber(account : Account) : integer
Interface Page
+ createPage() : void
+ deletePage() : void
Notifier System
+ sendEmail(email : string, subject : string, message : string) : void
+ sendNotification(trackingNumber: double) : Notification
+ deleteNotificaiton(trackingNumber: double) : void
Notification
- notification: string
- trackingNumber : integer
- referenceNumber: integer
- notificationType: string
+ editNotification(notification: string, referenceNumber: integer, notificationType: string) : void
+ getNotification(trackingNumber: double) : string
Account
- name : string
- password : string
- email : string
- school: string
- referenceNumber : integer
- lastLogin: string
- accountType: string
+ createAccount(name: string, password: string, email: string, school: string, accountType: string) : void
+ editAccount(name: string, password: string, email: string, school: string) : void
+ deleteAccount(referenceNumber: integer): boolean
+ authenticate(email: string, password: string): boolean
Post
- post: string
- trackingNumber : integer
- referenceNumber: integer
- time: double
- points: float
+ editPost(post: string) : void
1 exists in 1
Forum System
+ getPoints(points: float) : void
+ createComment(comment: string) : void
+ getComment(commentReference: integer) : string
+ deleteComment(commentReference: integer) : void
+ createPost(post: string) : void
+ getPost(postReference: integer) : string
+ deletePost(postReference: integer) : void
+ upvote(point: float) : float
+ downvote(point: float) : float
1 exists 1
Comment
- comment: string
- trackingNumber : integer
- referenceNumber: integer
- time: double
- points: float
editComment(post: string) : void
Faculty
- degree: string
- department: string
+ editDegree(degree: string): void
+ editDepartment(department: string): void
Admin
- passcode: string
+ editPasscode(passcode:string): void
Point System
+ pointDecay(points : float, lastLogin: string) : float
+ checkRewards(points : float) : string
+ getPoints(referenceNumber: integer) : float
+ updatePoints(referenceNumber: integer) : float
Student
- major: string
- schedule: array
- points : float
+ editMajor(major: string): void
+ editSchedule(schedule: array): void
Tutor System
+ rateTutor(referenceNumber: integer) : double
+ checkTutor(referenceNumber: integer) : array
+ addTutor(referenceNumber: integer) : void
+ deleteTutor(referenceNumber: integer) : void
Tutor
- rating : double
- students: array
+ addStudents(referenceNumber: integer): void
[online diagramming & design] creately.com

8.2 Data Types and Operation Signatures

DB: Connection

Class that stores the essential variables for the forum to the database class.

- post: Post

  A single instance of a post.

- comment: Comment

  A single instance of a comment.

- accountType: Account
  A single instance of an account.

- notification: Notifier
  A single instance of a notification.

- trackingNumber: integer
  A tracking number for an instance of a post, comment, account, or notification.

- referenceNumber: integer
  A reference number that links a post, comment, notification, and account with each other.

Searcher
Class that allows user to search website using specific keywords.

- createQuery(topics: string, comments: string, keyword: string) : Query
  Requests for information from the database that matches the user's search query.

- search(q:Query): result[]
  Displays the result of the search query.

- doesExist(trackingNumber: string) : boolean
  Checks if the search query matches the data from the database and returns a boolean value.

Archiver
Assigns and retrieves posts/comments/accounts using tracking numbers and reference numbers through PHP or mySQL script connecting to database.

- assignTrackingNumber(post: Post(), comment: Comment) : integer
  Assigns a tracking number to everything that entails posts, accounts, comments, and notifications

- assignReferenceNumber(account: Account) : integer
  Assigns a reference number to posts, accounts, comments, and notifications to make clear connections.

Notifier System
Class that keeps track and creates of notifications to send.

- sendEmail(email: string, subject: string, message: string) : void
  Sends an email to the user whenever another user replies to a post or comment.
  Sends an email to the user whenever it is the first time signing up for a new account (a confirmation email)
  Sends an email to the user whenever the user is eligible to become a tutor.

- sendNotification(name: string, notification: string) : void
  Sends a notification request to a user account.

- deleteNotification(trackingNumber: double) : void
  Delete a notification when user sees it.

Point System
Class that defines the point system and monitors the points and rewards of each user.

- pointDecay(points: float, LastLogin: string) : float
  Function which decays points if a user hasn't logged into their account in a long time. Also decays the value of a user's votes if they upvote/downvote too frequently.

- checkRewards(points: float) : string
  Function which tracks a user's current points and returns the tier of that user when they reach a reward tier.

- getPoints(referenceNumber: integer) : float
  Function which returns the current points that the user has.

- updatePoints(referenceNumber: integer) : float
  Function updates the points of the user to the most recent point value.

Notification
Class that keeps track of every notification with its content and tracking number.

- notification: string
  The content of the notification to be sent.

- trackingNumber: integer
  A tracking number assigned for an instance of a notification.

- referenceNumber: integer
  A reference number that links a post, comment, notification, and account with each other.

- notificationType: string
  A string that holds the type of notification.

- editNotification(notification: string, rreferenceNumber: integer, notificationType: string) : void
  Function that edits the notification's properties (reference number, notification type, and

- getNotification(trackingNumber: double) : string

Forum System
Class that defines the forums, including creating and deleting comments and posts, accessing the points attached to those comments and posts.

- getPoints(points: float) : integer
  Return the points the post/comment received.

- createComment(comment: string) : void
  Create a comment in the forum to put into database through archiver.

- getComment(commentReference: integer) : string
  Return the comment with the inputted comment reference number.

- deleteComment(commentReference: integer) : void
  Delete the comment with the inputted comment reference number.

- createPost(post: string) : void
  Create a post in the forum to put into database through archiver.

- getPost(commentReference: integer) : string
  Return the comment with the inputted comment reference number.

- deletePost(postReference: string) : void
  Delete the post with the inputted comment reference number.

- upvote(points: float) : float
  Add one point to the account's points and return the new points.

- downvote(points: float) : float
  Subtract one point from the account's points and return the new points.

Post
Class that keeps track of every post with its content and tracking number.

- content: string
  The content of a post, defined every time a post is created.

- trackingNumber: integer
  A tracking number assigned for an instance of a post.

- reference number: integer
  A reference number for the account that created the comment.

- time: double
  Holds the time the post was posted.

- points: float
  Holds the number of upvote/downvote points this post has received.

- editComment(comment: string) : void
  Function which changes the post string to what user inputs.

Comment
Class that keeps track of every post with its content and tracking number.

- content : string
  The content of a comment, defined every time a comment is created.

- trackingNumber: integer
  A tracking number assigned for an instance of a comment.

- reference number: integer

A reference number for the account that created the comment.

- time: double
  Holds the time the comment was posted.

- points: float
  Holds the number of upvote/downvote points this comment has received.

- editComment(comment: string) : void
  Function which changes the comment string to what user inputs.

Interface Page
Class that creates and deletes a new page for new posts, signup, and etc.

- createPage() : void
  Create a new page with the request.

- deletePage() : void
  Delete an existing page.

Account
Class that contains and handles a user account's information.

- name: string
  Contains the first and last name of the user account.

- password: string
  Contains the password of the user account.

- email: string
  Contains the email of the user account.

- school: string
  Contains the school/university of the user account.

- reference number: integer

An individual reference number for the account to differentiate users.

- createAccount(name: string, password: string, email: string, school: string, accountType: string, degree: string, major: string, company: string) : void
  Creates an account with the given input data, assigning it a reference number to track the account.

- editAccount(name: string, password: string, email: string, school: string, accountType: string, degree: string, major: string, company: string) : void
  Function used to change any data above the user wishes to change about their account.

- deleteAccount(referenceNumber: integer): boolean
  Function which deletes the account and returns the result.

- authenticate(email: string, password: string): boolean
  Function which authenticates the user sign in and returns the result.

Faculty
A faculty account for professors and other faculty users.

- degree: string
  Contains the graduated degree of user.

- department: string
  Holds the department the faculty is affiliated with.

- editDegree(degree: string) : void
  Allows faculty to edit/update their degree.

- editDepartment(department: string) : void
  Allows faculty to edit/update the department they're affiliated with.

Admin
A admin account which has special editing/moderator privileges in the site.

- passcode: string
  Contains a authorization code to confirm admin user is genuine to access moderating privileges in features.

- editPasscode(passcode: string) : void

  Allow change/update in passcode for admin.

- confirmPasscode(passcode: string) : boolean

  Return authorization confirmation when user input matches passcode.

Student
A student account for currently studying students in college.

- major: string

  Contains the major the student is currently pursuing.

- schedule: array

  Holds an array of all the classes the student is currently taking.

- points: float

  Holds the compiled upvotes/downvotes the student has compiled.

- editMajor(major: string) : void

  Function that allows student to update and edit their pursued major.

- editSchedule(schedule: array) : void

  Function that allows the student to update their schedule.

Tutor
A tutor account that inherits student account.

- rating: double

  Contains the current rating the tutor has.

- students: array

  Holds the array of all the students the tutor teaches.

- addStudents(referenceNumber: integer) : void

  Adds a student under the tutor to the students array.

Tutor System
Handles the methods of the tutor page.

- rateTutor(referenceNumber: integer) : double

Rates tutor based on the accumulated ratings from the other users.

- checkTutor(referenceNumber: integer) : array
  Checks whether or not the tutor is available or not to tutor.
  Checks whether or not the given user is in fact a tutor or not.

- addTutor(referenceNumber: integer) : void
  Admin adds a tutor to the database when the user is eligible and verified to become a tutor.

- deleteTutor() : void
  Tutor or Admin deletes a tutor from the database when the tutor decides to resign or loses privileges

8.3 Traceability Matrix

	Classes													
Domain Concepts	Database Connection	Post	Comment	Notification	Notifier	Archiver	Page Maker	Account	Acount Handler	Searcher	Logger	Point System	Forum	Tutor
Database Connection	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Email Notifier	X				X							X		
Forum Notifier	X				X									
Login Information	X								X					
Login Checker	X								X					
Interface Page	X	X												
Postprocessor	X					X				X				
Archiver	X	X												
Investigation Request	X	X	X			X								
Sign Up Checker	X							X						

Advance Search	X					X				X				
Vote Maintainer	X											X		
Promoter	X											X		
Login Locker					X				X					
Point Collector	X											X		
Tutor Calendar Availability Investigator	X													X
Post Deleter	X						X							
Report Collector	X					X								
Password Resetter	X							X						
Tutor Time Register	X													X
Post Creator	X	X					X							
Post View Counter														
Signout														
Tutor Arrival														
Post Comment Counter														
Student Arrival														
Tutor Remover	X											X		X
Tutor Schedule Editor	X													X
Student Schedule Conflict Notifier														
Tutor Rating System	X													X
Page Deleter	X	X												
Comment Deleter	X		X											

8.4 Design Patterns

Although our system can be refactored to match any of the design patterns we learned in lecture, we believe that the proxy and the pub-sub patterns most accurately describes our system. The proxy pattern is as simple as it seems: requests are intercepted and preprocessed to enable safe, efficient, and correct access to the real subject. The pub-sub pattern is also quite simple as it is focused on detecting events instead of making decisions and issuing orders to Doers/Subscribers. Because our project is a web application, the user input occurs primarily through the mouse or the keyboard. Our UI is responsible for receiving these inputs and then calling the appropriate methods to fulfill these calls. As a result, there is a focus on the other objects' work: deciding when it is appropriate to call each one. However, our system does not currently support reversible actions, which would be implemented in the future as it would make certain aspects of our website optimal for the user. For instance, a reversible action would be useful if the user would want to delete an account that was accidently created.

Our system also makes use of the proxy pattern. As we learned in lecture, a proxy object pre-processes requests before forwarding them to subjects when appropriate. In order for the users to have access to certain features that we have in our website, like having the privilege to become a tutor, it is safer to have a proxy pattern to ensure that these processes are handled correctly. In our project, it is definitely possible to implement the majority of the patterns due to the fact that our system is quite complex and can be designed in various ways. Unfortunately, due to time constraints, we have implemented a few of patterns that we have gone over. If we are to continue our project in the future, we would definitely include more of the design patterns that we have gone over in class so that our system is moving in the right direction.

8.5 Object Constraint Language (OCL)

Class	Invariant	Precondition	Postcondition
DatabaseConnection	Database is always available for access	Information is sent to the database	Information has been retrieved from the database
Searcher	Database is always available for access	The information being searched must already exist	The items that match the search is returned or no results found is returned
Archiver	Database is always available for access	Obtain location of new data	New data has been tagged with a tracking number
InterfacePage	Database is always available for access	The necessary information must be retrieved from the database	The page loads the necessary information onto the page
Notification	Database is always available for access	Event that sends out a notification	User will see notification through email
Account	Database is always available for access	Require inputted information will be stored into database.	User account will be created and verified.
Faculty	Database is always available for access	Require degree and department information	Store information into database and allow access to certain features
Admin	Database is always available for access	Require password input	Confirm password to access information
Student	Database is always	Require to be a student that is part of a university	Store information into database and allow access to certain features

	available for access		
Tutor	Database is always available for access	Require tutor name, student name	Confirm and add student name under tutor time session
TutorSystem	Database is always available for access	Tutor must be available in database and ratings must be checked	Ratings and reviews are displayed for students to view before choosing tutor sessions
PointSystem	Database is always available for access	There must exist a comment or post that reflects the current value of points	The upvote or downvote should register in the system and the new value will show
Comment	Database is always available for access	The inputted statement will be taken into database	The statement will be displayed and stored into database
Forum System	Database is always available for access	A post is created or already created	New comments and posts are shown
Post	Database is always available for access	The inputted statement will be taken into database	The statement will be displayed and stored into database
Notifier System	Database is always available for access	A notification is made to send to specified email	Email is sent to user

9. System Architecture and System Design

9.1 Architectural Styles

Client/Server Architecture is a model that helps simplifies the development of the software by dividing the process into client and server. This model is our main architectural design, because this architecture aims at allowing access for one or more clients to several resources or functionalities in a central server; it separates our system requirements into two easily programmable systems. First, the client, which acts as the User Interface, requests data from the server, and waits for the server's response. Secondly, the server, which authorizes what privileges each user is granted based on their account type.

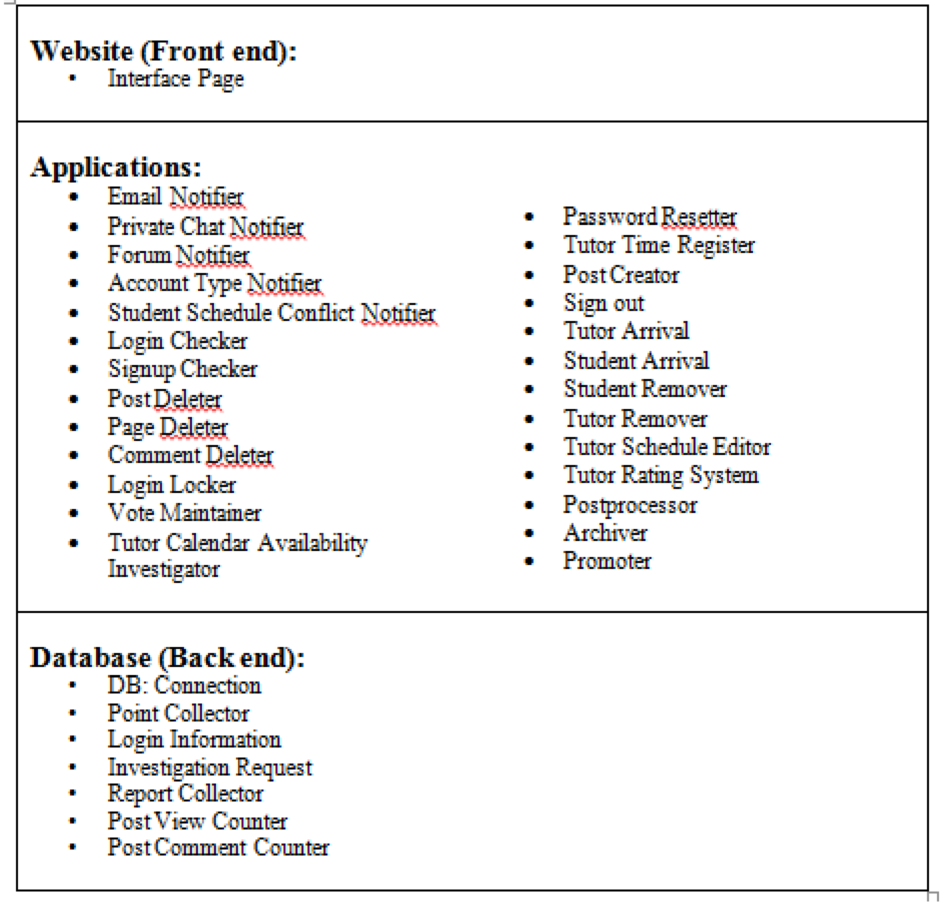
In the Client/Server model, there are many types of tiers available to describe the system architecture. These tiers are meant to physically separate applications into different concentrations that can be distributed between client and server. Typically, the tiers are categorized by application, presentation, and data processing. Usually the tiers are separated as presentation, middle, and data. Presentation tier is where users interact with the applications. MIddle tier is the communication medium for both presentation and data tier. Data tier is where the data is located, saved, and modified.

Creddit will have a 3-tier Client-Server architectural style as we have a presentation tier, logic tier, and data tier. Starting from the lowest tier, Creddit has a data tier to store and retrieve user information, emails, chat information, forums, etc. The middle tier, which is the logic tier, coordinates and moves information back and forth between the data and the presentation tier. The logic tier also evaluates, decides, and processes commands. The top most tier is the the presentation tier. The presentation tier translates tasks and results into something a user can understand. It will translate it to a webpage in which the user can easily navigate. For example, the user will enter information and hit "login" on the presentation tier, the presentation tier will send it to the logic tier and the logic tier will decide what to look for. After it decides what it wants, it sends it to the data tier and will retrieve the user information from the data tier. The logic tier will then decide if the information sent from the presentation page matches with the data, and return a result of successful or unsuccessful login.

9.2 Identifying Subsystems

Three subsystems are created to help achieve the architectural style.

1. Website (Front End): This subsystem deals with user interactions, where the clients may interact with the server through the Creddit website. It maintains the display and connection between the database and user commands to retrieve and send data.
2. Application: This subsystem is the connection between the users and the data with the technical operations. It consists of the functions that is needed to connect the front end and back end.
3. Database (Back End): The database subsystem handles data management and notification alerts. It maintains the flow of data processes.



Website (Front end):

- Interface Page

Applications:

- Email Notifier
- Private Chat Notifier
- Forum Notifier
- Account Type Notifier
- Student Schedule Conflict Notifier
- Login Checker
- Signup Checker
- Post Deleter
- Page Deleter
- Comment Deleter
- Login Locker
- Vote Maintainer
- Tutor Calendar Availability Investigator
- Password Resetter
- Tutor Time Register
- Post Creator
- Sign out
- Tutor Arrival
- Student Arrival
- Student Remover
- Tutor Remover
- Tutor Schedule Editor
- Tutor Rating System
- Postprocessor
- Archiver
- Promoter

Database (Back end):

- DB: Connection
- Point Collector
- Login Information
- Investigation Request
- Report Collector
- Post View Counter
- Post Comment Counter

Figure 9.1 - Subsystems

9.3 Mapping Subsystems to Hardware

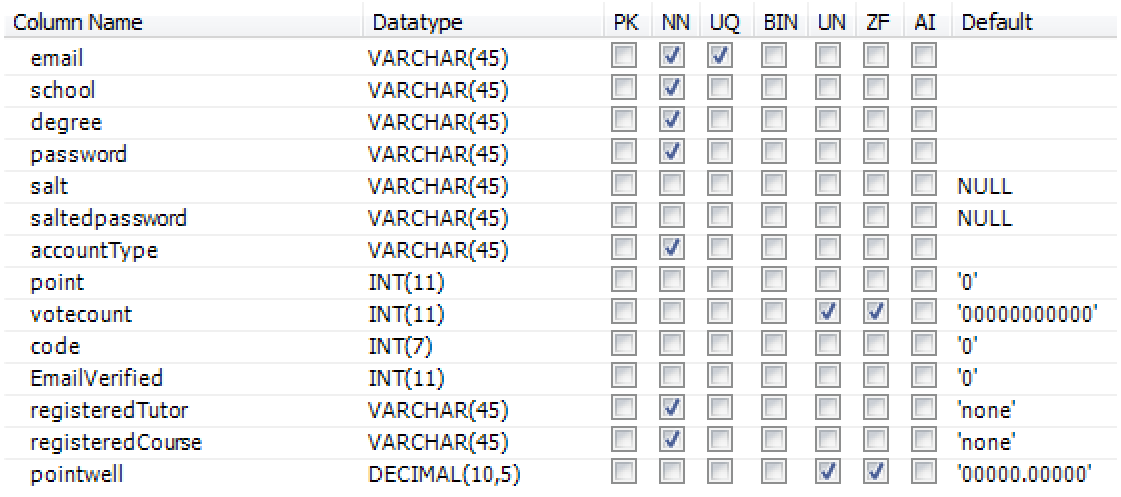
Creddit will be accessible anywhere that web access exists. This means the client in our client/server relationship will be a web client. A web browser is an example of a web client, and can be remotely access from the server via HTTP. This web client/server model will need to be run across multiple computers, or subsystems. A web browser will be used to request the various data from our server, as well as the creation of post. The database will be saved on hard-drives, which will be located in the server computer.

9.4 Persistent Data Storage

Creddit will require data, such as post and information to outlive requests and sessions. Data will be stored in a MySQL database as a relational database management system. Because there is potential for the database to grow very quickly over time, it will be saved on hard drives. The MySQL database will consist of multiple tables. The main tables will be a tutor table, forum table and user table. The tutor table will used for the tutoring feature. The user table will contain all the information about the users such as their account type and personal information. The forum table will contain the information related to posting or commenting on the forums. There are many more tables included below. However, they do not represent all of the tables needed for this system

Account Table:

The Account Table contains all the user information that is filled when the user creates an account on Creddit.



Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
email	VARCHAR(45)		✓	✓					
school	VARCHAR(45)		✓						
degree	VARCHAR(45)		✓						
password	VARCHAR(45)		✓						
salt	VARCHAR(45)								NULL
saltedpassword	VARCHAR(45)								NULL
accountType	VARCHAR(45)		✓						
point	INT(11)								'0'
votecount	INT(11)					✓	✓		'00000000000'
code	INT(7)								'0'
EmailVerified	INT(11)								'0'
registeredTutor	VARCHAR(45)		✓						'none'
registeredCourse	VARCHAR(45)		✓						'none'
pointwell	DECIMAL(10,5)					✓	✓		'00000.00000'

Figure 9.2

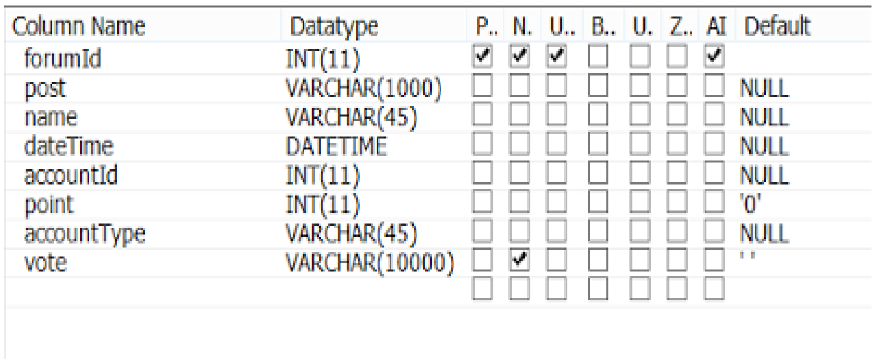
registeredTutor: Records that student has registered under a tutor for a tutoring session when they sign up.

registeredCourse: Records that student has registered under s specific class when they sign up.

The purpose of having these two types is the make sure that the student has taken this course under a specific tutor before they are allowed to post a review. When a student wishes to post a review on a tutor, the system will check that they have signed up for the course as well as the tutor. Both are necessary because one tutor may teach more than one class.

Forum Table:

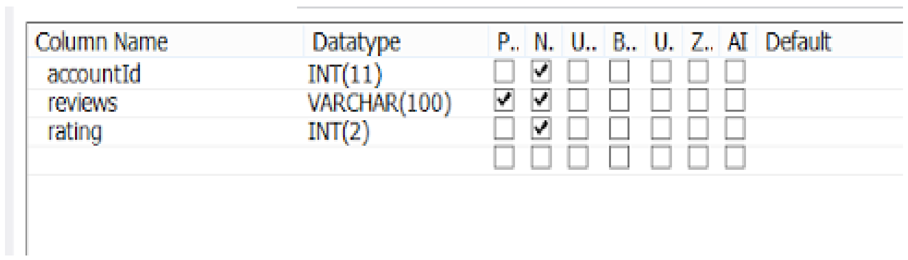
The Forum Table contains the information on each discussion post. The forumId is needed to track the post.



Column Name	Datatype	P..	N.	U..	B..	U.	Z..	AI	Default
forumId	INT(11)	✓	✓	✓				✓	
post	VARCHAR(1000)								NULL
name	VARCHAR(45)								NULL
dateTime	DATETIME								NULL
accountId	INT(11)								NULL
point	INT(11)								'0'
accountType	VARCHAR(45)								NULL
vote	VARCHAR(10000)		✓						' '

Figure 9.3

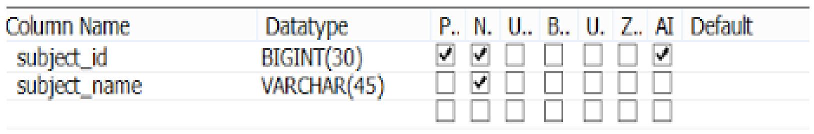
Reviews Table:
The Reviews Table holds the rating and reviews of the tutor.



Column Name	Datatype	P..	N.	U..	B..	U.	Z..	AI	Default
accountId	INT(11)		✓						
reviews	VARCHAR(100)	✓	✓						
rating	INT(2)		✓						

Figure 9.4

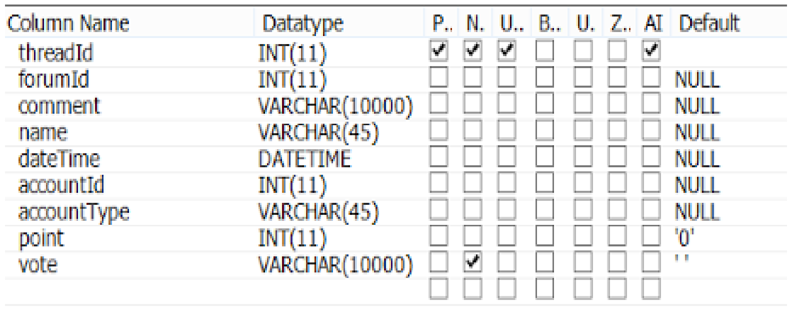
Subject Table:
The Subject Table contains information on subject categories.



Column Name	Datatype	P..	N.	U..	B..	U.	Z..	AI	Default
subject_id	BIGINT(30)	✓	✓					✓	
subject_name	VARCHAR(45)		✓						

Figure 9.5

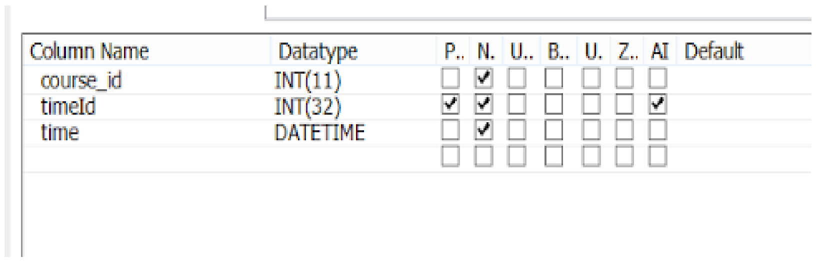
<u>Thread Table:</u>
The Thread Table contains the information on the comments. Each comment can be referenced back with the threadId.



Column Name	Datatype	P..	N.	U..	B..	U.	Z..	AI	Default
threadId	INT(11)	✓	✓	✓	☐	☐	☐	✓	
forumId	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
comment	VARCHAR(10000)	☐	☐	☐	☐	☐	☐	☐	NULL
name	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL
dateTime	DATETIME	☐	☐	☐	☐	☐	☐	☐	NULL
accountId	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
accountType	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL
point	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
vote	VARCHAR(10000)	☐	✓	☐	☐	☐	☐	☐	''
		☐	☐	☐	☐	☐	☐	☐	

Figure 9.6

<u>Time Table:</u>
The Time Table contains information on the available times the tutor offers.



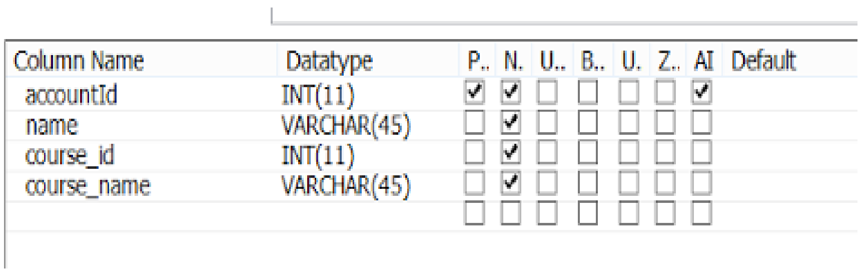
Column Name	Datatype	P..	N.	U..	B..	U.	Z..	AI	Default
course_id	INT(11)	☐	✓	☐	☐	☐	☐	☐	
timeId	INT(32)	✓	✓	☐	☐	☐	☐	✓	
time	DATETIME	☐	✓	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	

Figure 9.7

timeId: holds the identification number for the time of a specific course and tutor. The purpose of the identification code is so that there can be multiple times for a class.

time: holds the actual time and date for that course under a specific tutor.

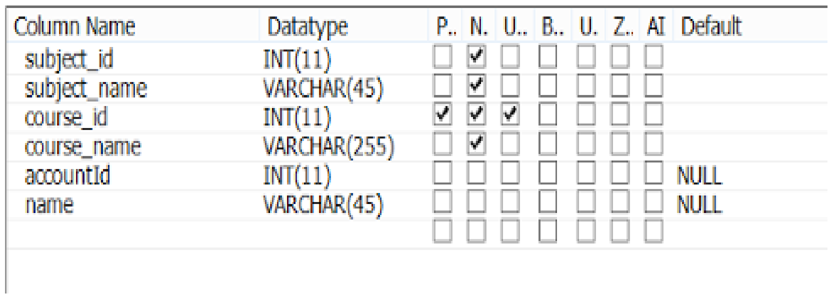
<u>Tutor Table:</u>
The Tutor Table contains information about the tutor.



Column Name	Datatype	P..	N.	U..	B..	U.	Z..	AI	Default
accountId	INT(11)	✓	✓	☐	☐	☐	☐	✓	
name	VARCHAR(45)	☐	✓	☐	☐	☐	☐	☐	
course_id	INT(11)	☐	✓	☐	☐	☐	☐	☐	
course_name	VARCHAR(45)	☐	✓	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	

Figure 9.8

Tutorcourse Table:
The Tutorcourse Table contains information about the available courses the tutor offers.



Column Name	Datatype	P..	N.	U..	B..	U.	Z..	AI	Default
subject_id	INT(11)	☐	✓	☐	☐	☐	☐	☐	
subject_name	VARCHAR(45)	☐	✓	☐	☐	☐	☐	☐	
course_id	INT(11)	✓	✓	✓	☐	☐	☐	☐	
course_name	VARCHAR(255)	☐	✓	☐	☐	☐	☐	☐	
accountId	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
name	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	

Figure 9.9

9.5 Network Protocol

Since the only form of communication between the server and the client is the website, we decided to use PHP because it is standard in creating dynamic web pages Furthermore, it works well with relational database management systems. We chose SQL because it is the standard RDMS used to manage and manipulate large amounts of data. Lastly, we use HTML because, with the release of HTML 5, HTML has become one of the most powerful and simple markup languages for developing web pages.

9.6 Global Control Flow

Execution Orderliness:
The system is defined as event-driven because the processes are determined by the flow of events depending on user inputs. The user will interact with the system. The server will wait for the user to make an action before any data is processed. The user can perform any actions in any order as long as they have already logged into Creddit.

Time dependency:
The point system is a periodic real-time system, where the time constraint is 7 days.
The newsfeed is an real-time system, where the most popular forum are updated daily.
These real-time systems will update the database at exact defined times.

Concurrency:
As a website, multithreading must be supported. It is expected that there will be concurrent users accessing the website or the database. Multithreading is also needed in chatrooms, tutorings, and other sharing features between users.

9.7 Hardware Requirements

User-end:
The user end will need an up-to-date web browser. All information is stored on a database, so the user will only need minimal temporary disk storage.

Back-end:
The back-end system will need will require a database with to store all the information, such as, user data, forum posts, chats, emails, and etc. The back-end system will also be handling a large amount of traffic, so high bandwidth network connection will be required.

10. Algorithms and Data Structures

10.1 Algorithms

Creddit point system requires an algorithm that will allow the upvote/downvote system to be relevant but will also guard against potential threats such as users who might try to abuse the upvote/downvote system. The algorithm Creddit has developed resolves both these issues. Our algorithm uses an exponential function that determines how many points a student will get from an upvote or downvote. If a user upvotes/downvotes frequently, the conversion of his upvotes/downvote to the amount of points given for that upvote/downvote will decrease at an exponential rate. In other words, if the system realizes that the user upvotes/downvotes very frequently, this would mean that the user is spamming the value of an upvote/downvote. Therefore, the system would lessen the value of the corresponding user's upvotes/downvotes simply due to the fact that the user would not be giving a clear distinction of what he/she likes/dislikes or is spamming which could lead to inaccuracies. The more the user spams, the smaller the value of each point becomes. This method stops users from abusing the upvotes/downvotes system by lessening the impact of their votes. These values assigned to the upvote/downvotes are reset every week.

~~upvote/downvote point value conversion~~

~~Point value=1-(1/(1+e-(x-10)))~~

~~x is the amount of upvote/downvotes the user has already submitted.~~

UPDATE: The algorithm has been updated to be based on a vote count system that will determine how many points are given per vote. For example, if the vote count is less than ten, the point worth is equal to one. If the vote count is between the numbers 10 and 55, then the point worth is equal to e^-(voteCount/100). If the vote count is greater than 55, each vote is worth 0.15. These points are collected by the point well which will determine when a user is eligible to become a tutor. The point well is invisible to the users and is different than the point amount that is shown when the user logs in. The point value attributed to each account and has its own column on the table within the database.

For the search page, the algorithm that is implemented is by using the Levenshtein distance algorithm along with some other functions. First, the string that is input into the search bar is parsed and tokenized. Then, we also parse and tokenize all the comments, posts, tutors, and tutor courses within the database. We then compare the input string tokens to all of the database tokens and see how similar they are using Levenshtein's distance algorithm. If they are similar, we then display the results.

10.2 Data Structures

The main data structure of our system is going to be a relational database executed through

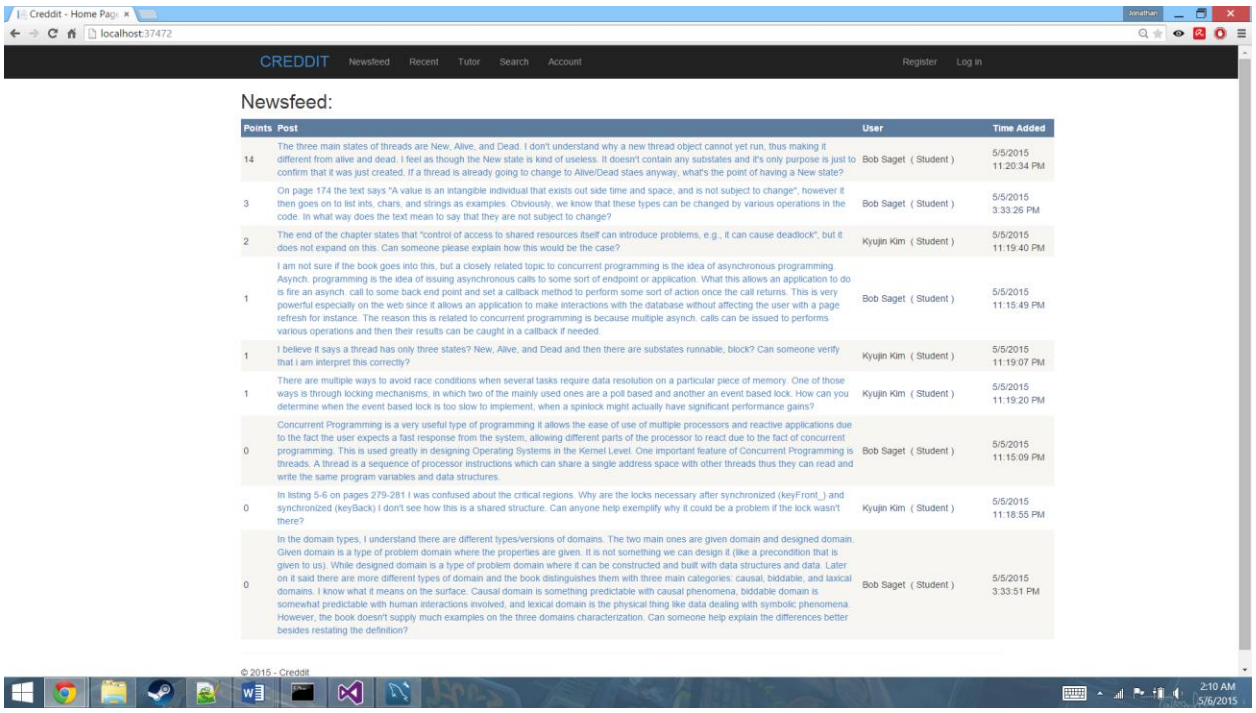
MySQL, a relational database management system. The benefits of this database is its use of tables and the ability to quickly search through these tables based on “keys”, specific shared information, that connects the tables together. A relational database is great because it accepts a large amount of data like we expect from such a large-scale website and it also supports queries, which will be made frequently through use of the search bar. Everything from account information to schedules will be stored within the SQL database, linking pertinent information together through use of reference numbers to link to accounts and tracking numbers to link to posts. We chose to use a MySQL database for it’s efficient handling of large amounts of data and it’s ability to store this data in an organized way.

11. User Interface Design and Implementation

Preface

The original screen mock-ups that we created in report 1 part 3 where to get a general idea of how the website would look like and how it would be navigated. It is much easier to focus Creddit in the direction we want it to go once there are visuals of it. We have cleaned the pictures up and removed some of the features and account types that will not be implemented in Creddit. We have removed the following features: Documents, Resume Builder, Jobs/Research/Recruiters, Chat, and the account type Recruiter. Also, we have improved the appearance of the hand drawn pictures. The user effort has the same. Creddit has been designed to look sleek and intuitive thus it is very easy to navigate.

1. **User Interface Design**



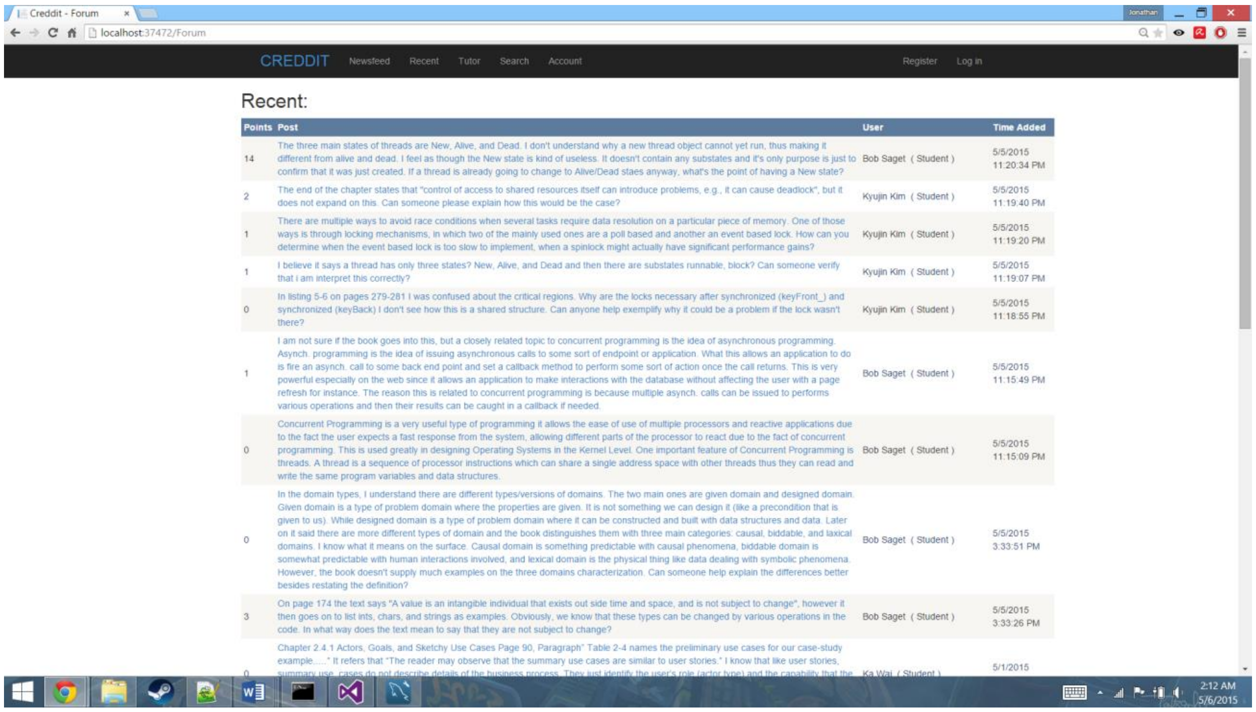
Newsfeed:

Points	Post	User	Time Added
14	The three main states of threads are New, Alive, and Dead. I don't understand why a new thread object cannot yet run, thus making it different from alive and dead. I feel as though the New state is kind of useless. It doesn't contain any substates and it's only purpose is just to confirm that it was just created. If a thread is already going to change to Alive/Dead staes anyway, what's the point of having a New state?	Bob Saget (Student)	5/5/2015 11:20:34 PM
3	On page 174 the text says "A value is an intangible individual that exists out side time and space, and is not subject to change", however it then goes on to list ints, chars, and strings as examples. Obviously, we know that these types can be changed by various operations in the code. In what way does the text mean to say that they are not subject to change?	Bob Saget (Student)	5/5/2015 3:33:26 PM
2	The end of the chapter states that "control of access to shared resources itself can introduce problems, e.g., it can cause deadlock", but it does not expand on this. Can someone please explain how this would be the case?	Kyujin Kim (Student)	5/5/2015 11:19:40 PM
1	I am not sure if the book goes into this, but a closely related topic to concurrent programming is the idea of asynchronous programming. Asynch. programming is the idea of issuing asynchronous calls to some sort of endpoint or application. What this allows an application to do is fire an asynch. call to some back end point and set a callback method to perform some sort of action once the call returns. This is very powerful especially on the web since it allows an application to make interactions with the database without affecting the user with a page refresh for instance. The reason this is related to concurrent programming is because multiple asynch. calls can be issued to performs various operations and then their results can be caught in a callback if needed.	Bob Saget (Student)	5/5/2015 11:15:49 PM
1	I believe it says a thread has only three states? New, Alive, and Dead and then there are substates runnable, block? Can someone verify that i am interpret this correctly?	Kyujin Kim (Student)	5/5/2015 11:19:07 PM
1	There are multiple ways to avoid race conditions when several tasks require data resolution on a particular piece of memory. One of those ways is through locking mechanisms, in which two of the mainly used ones are a poll based and another an event based lock. How can you determine when the event based lock is too slow to implement, when a spinlock might actually have significant performance gains?	Kyujin Kim (Student)	5/5/2015 11:19:20 PM
0	Concurrent Programming is a very useful type of programming it allows the ease of use of multiple processors and reactive applications due to the fact the user expects a fast response from the system, allowing different parts of the processor to react due to the fact of concurrent programming. This is used greatly in designing Operating Systems in the Kernel Level. One important feature of Concurrent Programming is threads. A thread is a sequence of processor instructions which can share a single address space with other threads thus they can read and write the same program variables and data structures.	Bob Saget (Student)	5/5/2015 11:15:09 PM
0	In listing 5-6 on pages 279-281 I was confused about the critical regions. Why are the locks necessary after synchronized (keyFront_) and synchronized (keyBack) I don't see how this is a shared structure. Can anyone help exemplify why it could be a problem if the lock wasn't there?	Kyujin Kim (Student)	5/5/2015 11:18:55 PM
0	In the domain types, I understand there are different types/versions of domains. The two main ones are given domain and designed domain. Given domain is a type of problem domain where the properties are given. It is not something we can design it (like a precondition that is given to us). While designed domain is a type of problem domain where it can be constructed and built with data structures and data. Later on it said there are more different types of domain and the book distinguishes them with three main categories: causal, biddable, and laxical domains. I know what it means on the surface. Causal domain is something predictable with causal phenomena, biddable domain is somewhat predictable with human interactions involved, and lexical domain is the physical thing like data dealing with symbolic phenomena. However, the book doesn't supply much examples on the three domains characterization. Can someone help explain the differences better besides restating the definition?	Bob Saget (Student)	5/5/2015 3:33:51 PM

© 2015 - Creddit

Figure 12.1.1a

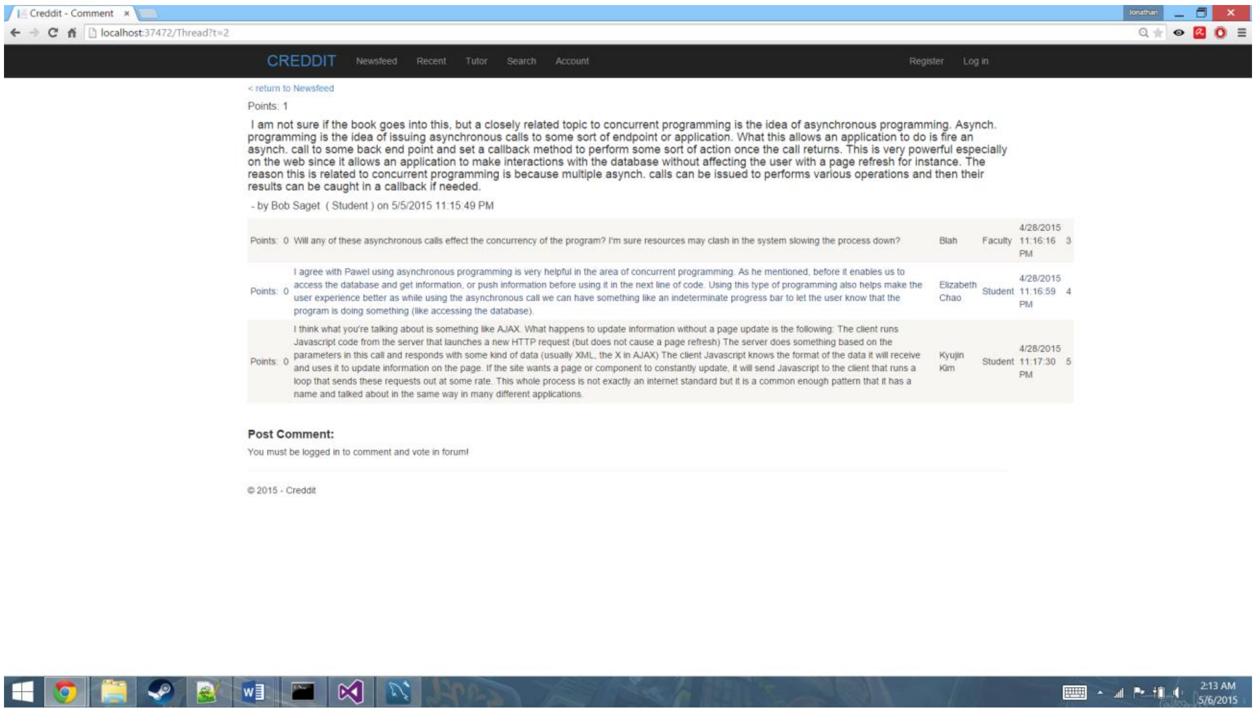
2. The Forums (UC-1)



Recent:

Points	Post	User	Time Added
14	The three main states of threads are New, Alive, and Dead. I don't understand why a new thread object cannot yet run, thus making it different from alive and dead. I feel as though the New state is kind of useless. It doesn't contain any substates and it's only purpose is just to confirm that it was just created. If a thread is already going to change to Alive/Dead staes anyway, what's the point of having a New state?	Bob Saget (Student)	5/5/2015 11:20:34 PM
2	The end of the chapter states that "control of access to shared resources itself can introduce problems, e.g., it can cause deadlock", but it does not expand on this. Can someone please explain how this would be the case?	Kyujin Kim (Student)	5/5/2015 11:19:40 PM
1	There are multiple ways to avoid race conditions when several tasks require data resolution on a particular piece of memory. One of those ways is through locking mechanisms, in which two of the mainly used ones are a poll based and another an event based lock. How can you determine when the event based lock is too slow to implement, when a spinlock might actually have significant performance gains?	Kyujin Kim (Student)	5/5/2015 11:19:20 PM
1	I believe it says a thread has only three states? New, Alive, and Dead and then there are substates runnable, block? Can someone verify that i am interpret this correctly?	Kyujin Kim (Student)	5/5/2015 11:19:07 PM
0	In listing 5-6 on pages 279-281 I was confused about the critical regions. Why are the locks necessary after synchronized (keyFront_) and synchronized (keyBack) I don't see how this is a shared structure. Can anyone help exemplify why it could be a problem if the lock wasn't there?	Kyujin Kim (Student)	5/5/2015 11:18:55 PM
1	I am not sure if the book goes into this, but a closely related topic to concurrent programming is the idea of asynchronous programming. Asynch. programming is the idea of issuing asynchronous calls to some sort of endpoint or application. What this allows an application to do is fire an asynch. call to some back end point and set a callback method to perform some sort of action once the call returns. This is very powerful especially on the web since it allows an application to make interactions with the database without affecting the user with a page refresh for instance. The reason this is related to concurrent programming is because multiple asynch. calls can be issued to performs various operations and then their results can be caught in a callback if needed.	Bob Saget (Student)	5/5/2015 11:15:49 PM
0	Concurrent Programming is a very useful type of programming it allows the ease of use of multiple processors and reactive applications due to the fact the user expects a fast response from the system, allowing different parts of the processor to react due to the fact of concurrent programming. This is used greatly in designing Operating Systems in the Kernel Level. One important feature of Concurrent Programming is threads. A thread is a sequence of processor instructions which can share a single address space with other threads thus they can read and write the same program variables and data structures.	Bob Saget (Student)	5/5/2015 11:15:09 PM
0	In the domain types, I understand there are different types/versions of domains. The two main ones are given domain and designed domain. Given domain is a type of problem domain where the properties are given. It is not something we can design it (like a precondition that is given to us). While designed domain is a type of problem domain where it can be constructed and built with data structures and data. Later on it said there are more different types of domain and the book distinguishes them with three main categories: causal, biddable, and laxical domains. I know what it means on the surface. Causal domain is something predictable with causal phenomena, biddable domain is somewhat predictable with human interactions involved, and lexical domain is the physical thing like data dealing with symbolic phenomena. However, the book doesn't supply much examples on the three domains characterization. Can someone help explain the differences better besides restating the definition?	Bob Saget (Student)	5/5/2015 3:33:51 PM
3	On page 174 the text says "A value is an intangible individual that exists out side time and space, and is not subject to change", however it then goes on to list ints, chars, and strings as examples. Obviously, we know that these types can be changed by various operations in the code. In what way does the text mean to say that they are not subject to change?	Bob Saget (Student)	5/5/2015 3:33:26 PM
0	Chapter 2.4.1 Actors, Goals, and Sketchy Use Cases Page 90, Paragraph" Table 2-4 names the preliminary use cases for our case-study example....." It refers that "The reader may observe that the summary use cases are similar to user stories." I know that like user stories, summary use cases do not describe details of the business process. They just identify the user's role (actor type) and the capability that the	Ka Wai (Student)	5/1/2015

Figure 12.2.1a



< return to Newsfeed

Points: 1

I am not sure if the book goes into this, but a closely related topic to concurrent programming is the idea of asynchronous programming. Asynch. programming is the idea of issuing asynchronous calls to some sort of endpoint or application. What this allows an application to do is fire an asynch. call to some back end point and set a callback method to perform some sort of action once the call returns. This is very powerful especially on the web since it allows an application to make interactions with the database without affecting the user with a page refresh for instance. The reason this is related to concurrent programming is because multiple asynch. calls can be issued to performs various operations and then their results can be caught in a callback if needed.

- by Bob Saget (Student) on 5/5/2015 11:15:49 PM

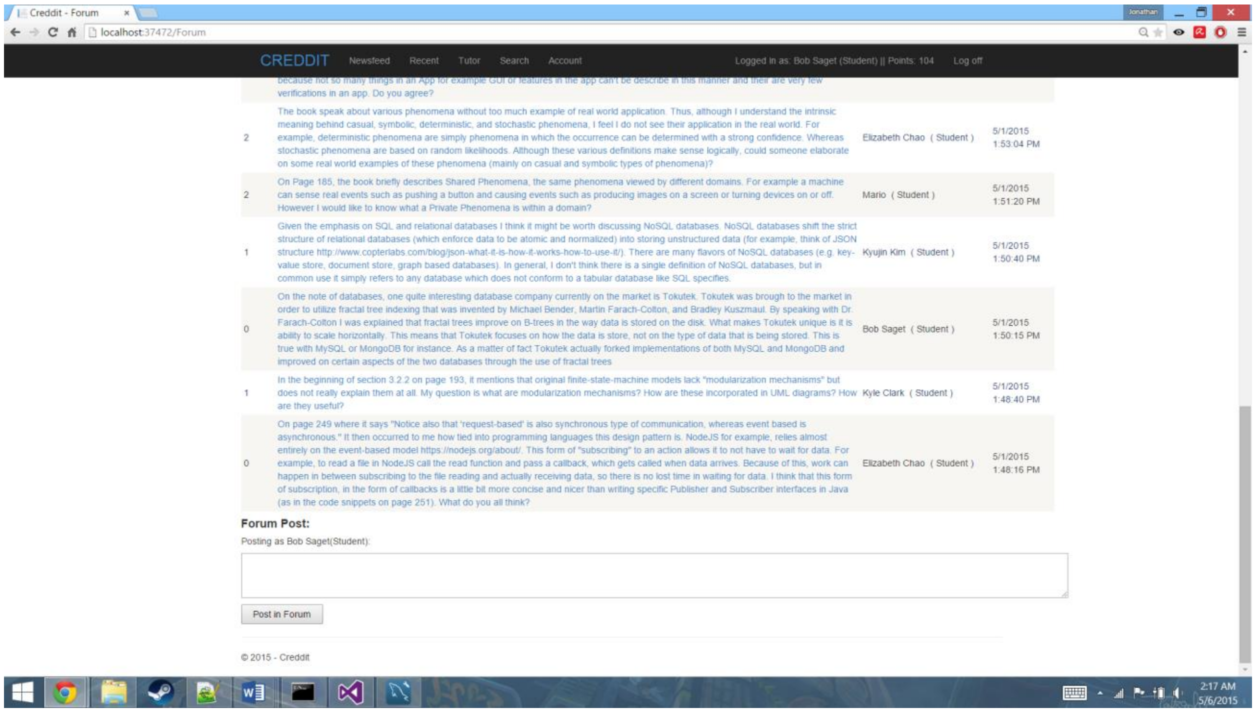
Points: 0	Will any of these asynchronous calls effect the concurrency of the program? I'm sure resources may clash in the system slowing the process down?	Blah	Faculty	4/28/2015 11:16:16 PM	3
Points: 0	I agree with Pawel using asynchronous programming is very helpful in the area of concurrent programming. As he mentioned, before it enables us to access the database and get information, or push information before using it in the next line of code. Using this type of programming also helps make the user experience better as while using the asynchronous call we can have something like an indeterminate progress bar to let the user know that the program is doing something (like accessing the database).	Elizabeth Chao	Student	4/28/2015 11:16:59 PM	4
Points: 0	I think what you're talking about is something like AJAX. What happens to update information without a page update is the following: The client runs Javascript code from the server that launches a new HTTP request (but does not cause a page refresh) The server does something based on the parameters in this call and responds with some kind of data (usually XML, the X in AJAX) The client Javascript knows the format of the data it will receive and uses it to update information on the page. If the site wants a page or component to constantly update, it will send Javascript to the client that runs a loop that sends these requests out at some rate. This whole process is not exactly an internet standard but it is a common enough pattern that it has a name and talked about in the same way in many different applications.	Kyujin Kim	Student	4/28/2015 11:17:30 PM	5

Post Comment:

You must be logged in to comment and vote in forum!

© 2015 - Creddit

Figure 12.2.1b



	because not so many things in an App for example GUI or features in the app can't be describe in this manner and their are very few verifications in an app. Do you agree?		
2	The book speak about various phenomena without too much example of real world application. Thus, although I understand the intrinsic meaning behind casual, symbolic, deterministic, and stochastic phenomena, I feel I do not see their application in the real world. For example, deterministic phenomena are simply phenomena in which the occurrence can be determined with a strong confidence. Whereas stochastic phenomena are based on random likelihoods. Although these various definitions make sense logically, could someone elaborate on some real world examples of these phenomena (mainly on casual and symbolic types of phenomena)?	Elizabeth Chao (Student)	5/1/2015 1:53:04 PM
2	On Page 185, the book briefly describes Shared Phenomena, the same phenomena viewed by different domains. For example a machine can sense real events such as pushing a button and causing events such as producing images on a screen or turning devices on or off. However I would like to know what a Private Phenomena is within a domain?	Mario (Student)	5/1/2015 1:51:20 PM
1	Given the emphasis on SQL and relational databases I think it might be worth discussing NoSQL databases. NoSQL databases shift the strict structure of relational databases (which enforce data to be atomic and normalized) into storing unstructured data (for example, think of JSON structure http://www.copterlabs.com/blog/json-what-it-is-how-it-works-how-to-use-it/). There are many flavors of NoSQL databases (e.g. key-value store, document store, graph based databases). In general, I don't think there is a single definition of NoSQL databases, but in common use it simply refers to any database which does not conform to a tabular database like SQL specifies.	Kyujin Kim (Student)	5/1/2015 1:50:40 PM
0	On the note of databases, one quite interesting database company currently on the market is Tokutek. Tokutek was brough to the market in order to utilize fractal tree indexing that was invented by Michael Bender, Martin Farach-Colton, and Bradley Kuszmaul. By speaking with Dr. Farach-Colton I was explained that fractal trees improve on B-trees in the way data is stored on the disk. What makes Tokutek unique is it is ability to scale horizontally. This means that Tokutek focuses on how the data is store, not on the type of data that is being stored. This is true with MySQL or MongoDB for instance. As a matter of fact Tokutek actually forked implementations of both MySQL and MongoDB and improved on certain aspects of the two databases through the use of fractal trees	Bob Saget (Student)	5/1/2015 1:50:15 PM
1	In the beginning of section 3.2.2 on page 193, it mentions that original finite-state-machine models lack "modularization mechanisms" but does not really explain them at all. My question is what are modularization mechanisms? How are these incorporated in UML diagrams? How are they useful?	Kyle Clark (Student)	5/1/2015 1:48:40 PM
0	On page 249 where it says "Notice also that 'request-based' is also synchronous type of communication, whereas event based is asynchronous." It then occurred to me how tied into programming languages this design pattern is. NodeJS for example, relies almost entirely on the event-based model https://nodejs.org/about/. This form of "subscribing" to an action allows it to not have to wait for data. For example, to read a file in NodeJS call the read function and pass a callback, which gets called when data arrives. Because of this, work can happen in between subscribing to the file reading and actually receiving data, so there is no lost time in waiting for data. I think that this form of subscription, in the form of callbacks is a little bit more concise and nicer than writing specific Publisher and Subscriber interfaces in Java (as in the code snippets on page 251). What do you all think?	Elizabeth Chao (Student)	5/1/2015 1:48:16 PM

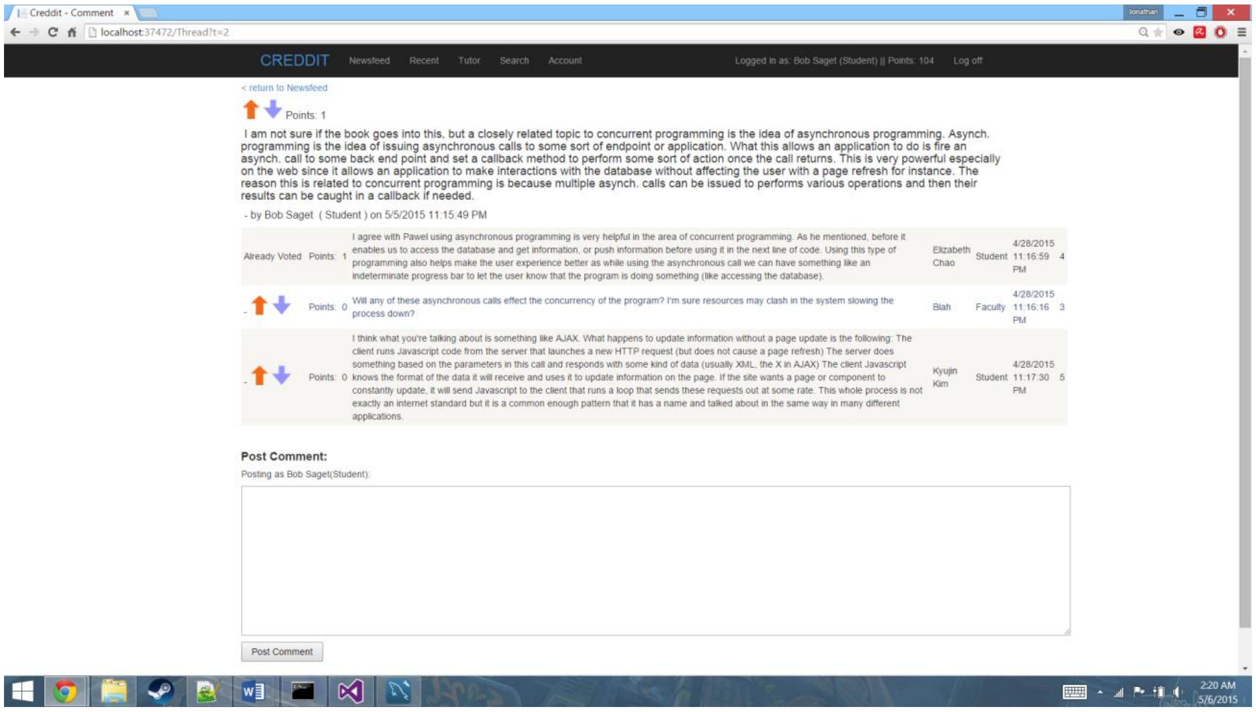
Forum Post:

Posting as Bob Saget(Student):

Post in Forum

© 2015 - Creddit

Figure 12.2.2a



Points: 1

I am not sure if the book goes into this, but a closely related topic to concurrent programming is the idea of asynchronous programming. Asynch. programming is the idea of issuing asynchronous calls to some sort of endpoint or application. What this allows an application to do is fire an asynch. call to some back end point and set a callback method to perform some sort of action once the call returns. This is very powerful especially on the web since it allows an application to make interactions with the database without affecting the user with a page refresh for instance. The reason this is related to concurrent programming is because multiple asynch. calls can be issued to performs various operations and then their results can be caught in a callback if needed.

- by Bob Saget (Student) on 5/5/2015 11:15:49 PM

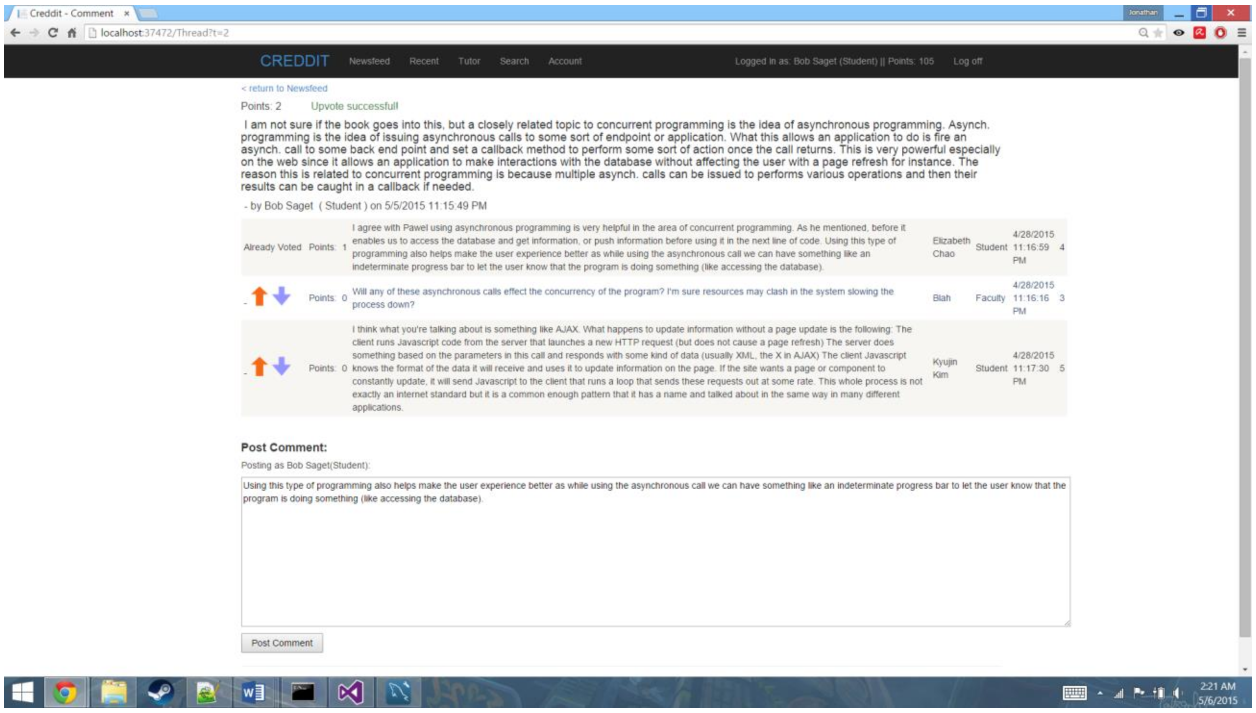
Already Voted	Points: 1	I agree with Pawel using asynchronous programming is very helpful in the area of concurrent programming. As he mentioned, before it enables us to access the database and get information, or push information before using it in the next line of code. Using this type of programming also helps make the user experience better as while using the asynchronous call we can have something like an indeterminate progress bar to let the user know that the program is doing something (like accessing the database).	Elizabeth Chao	Student	4/28/2015 11:16:59 PM	4
-	Points: 0	Will any of these asynchronous calls effect the concurrency of the program? I'm sure resources may clash in the system slowing the process down?	Blah	Faculty	4/28/2015 11:16:16 PM	3
-	Points: 0	I think what you're talking about is something like AJAX. What happens to update information without a page update is the following: The client runs Javascript code from the server that launches a new HTTP request (but does not cause a page refresh) The server does something based on the parameters in this call and responds with some kind of data (usually XML, the X in AJAX) The client Javascript knows the format of the data it will receive and uses it to update information on the page. If the site wants a page or component to constantly update, it will send Javascript to the client that runs a loop that sends these requests out at some rate. This whole process is not exactly an internet standard but it is a common enough pattern that it has a name and talked about in the same way in many different applications.	Kyujin Kim	Student	4/28/2015 11:17:30 PM	5

Post Comment:

Posting as Bob Saget(Student):

Post Comment

Figure 12.2.2b



Creddit - Comment

localhost:37472/Thread?t=2

CREDDIT Newsfeed Recent Tutor Search Account Logged in as: Bob Saget (Student) || Points: 105 Log off

< return to Newsfeed

Points: 2 Upvote successfull

I am not sure if the book goes into this, but a closely related topic to concurrent programming is the idea of asynchronous programming. Asynch. programming is the idea of issuing asynchronous calls to some sort of endpoint or application. What this allows an application to do is fire an asynch. call to some back end point and set a callback method to perform some sort of action once the call returns. This is very powerful especially on the web since it allows an application to make interactions with the database without affecting the user with a page refresh for instance. The reason this is related to concurrent programming is because multiple asynch. calls can be issued to performs various operations and then their results can be caught in a callback if needed.

- by Bob Saget (Student) on 5/5/2015 11:15:49 PM

Already Voted	Points: 1	I agree with Pawel using asynchronous programming is very helpful in the area of concurrent programming. As he mentioned, before it enables us to access the database and get information, or push information before using it in the next line of code. Using this type of programming also helps make the user experience better as while using the asynchronous call we can have something like an indeterminate progress bar to let the user know that the program is doing something (like accessing the database).	Elizabeth Chao	Student	4/28/2015 11:16:59 PM	4
-	Points: 0	Will any of these asynchronous calls effect the concurrency of the program? I'm sure resources may clash in the system slowing the process down?	Blah	Faculty	4/28/2015 11:16:16 PM	3
-	Points: 0	I think what you're talking about is something like AJAX. What happens to update information without a page update is the following: The client runs Javascript code from the server that launches a new HTTP request (but does not cause a page refresh) The server does something based on the parameters in this call and responds with some kind of data (usually XML, the X in AJAX) The client Javascript knows the format of the data it will receive and uses it to update information on the page. If the site wants a page or component to constantly update, it will send Javascript to the client that runs a loop that sends these requests out at some rate. This whole process is not exactly an internet standard but it is a common enough pattern that it has a name and talked about in the same way in many different applications.	Kyujin Kim	Student	4/28/2015 11:17:30 PM	5

Post Comment:

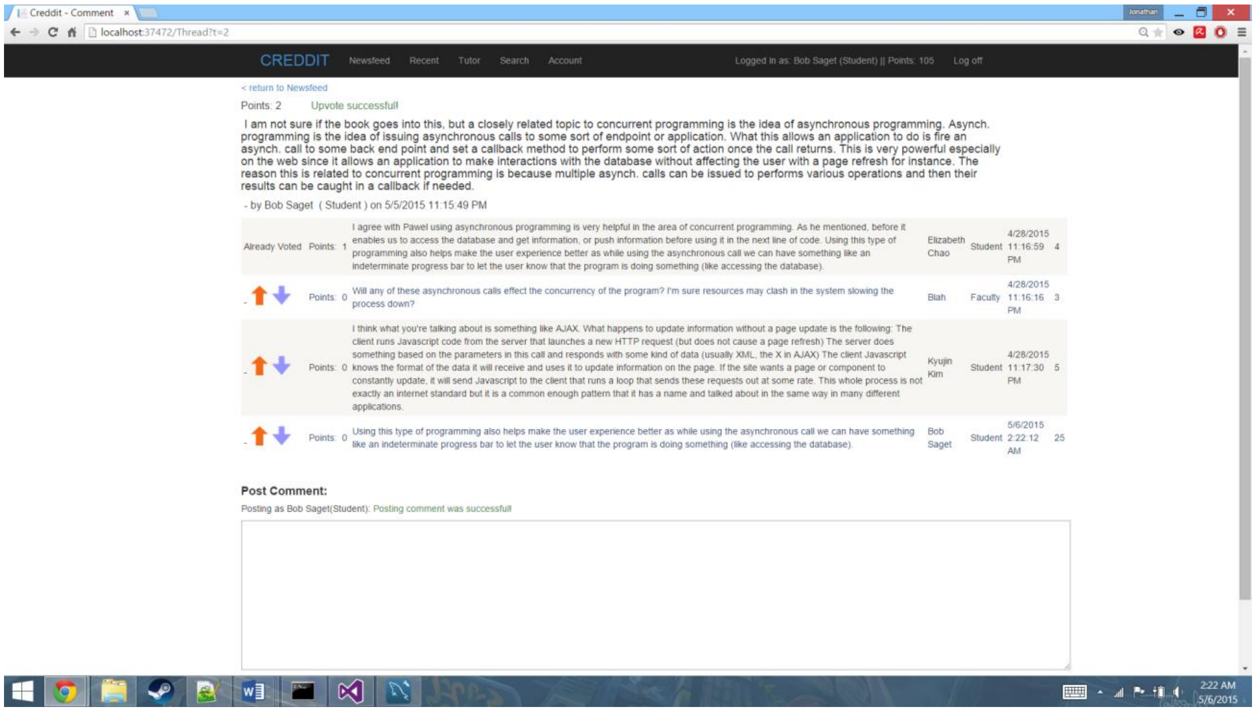
Posting as Bob Saget(Student):

Using this type of programming also helps make the user experience better as while using the asynchronous call we can have something like an indeterminate progress bar to let the user know that the program is doing something (like accessing the database).

Post Comment

2:21 AM 5/6/2015

Figure 12.2.3a



Creddit - Comment

localhost:37472/Thread?t=2

CREDDIT Newsfeed Recent Tutor Search Account Logged in as: Bob Saget (Student) || Points: 105 Log off

< return to Newsfeed

Points: 2 Upvote successfull

I am not sure if the book goes into this, but a closely related topic to concurrent programming is the idea of asynchronous programming. Asynch. programming is the idea of issuing asynchronous calls to some sort of endpoint or application. What this allows an application to do is fire an asynch. call to some back end point and set a callback method to perform some sort of action once the call returns. This is very powerful especially on the web since it allows an application to make interactions with the database without affecting the user with a page refresh for instance. The reason this is related to concurrent programming is because multiple asynch. calls can be issued to performs various operations and then their results can be caught in a callback if needed.

- by Bob Saget (Student) on 5/5/2015 11:15:49 PM

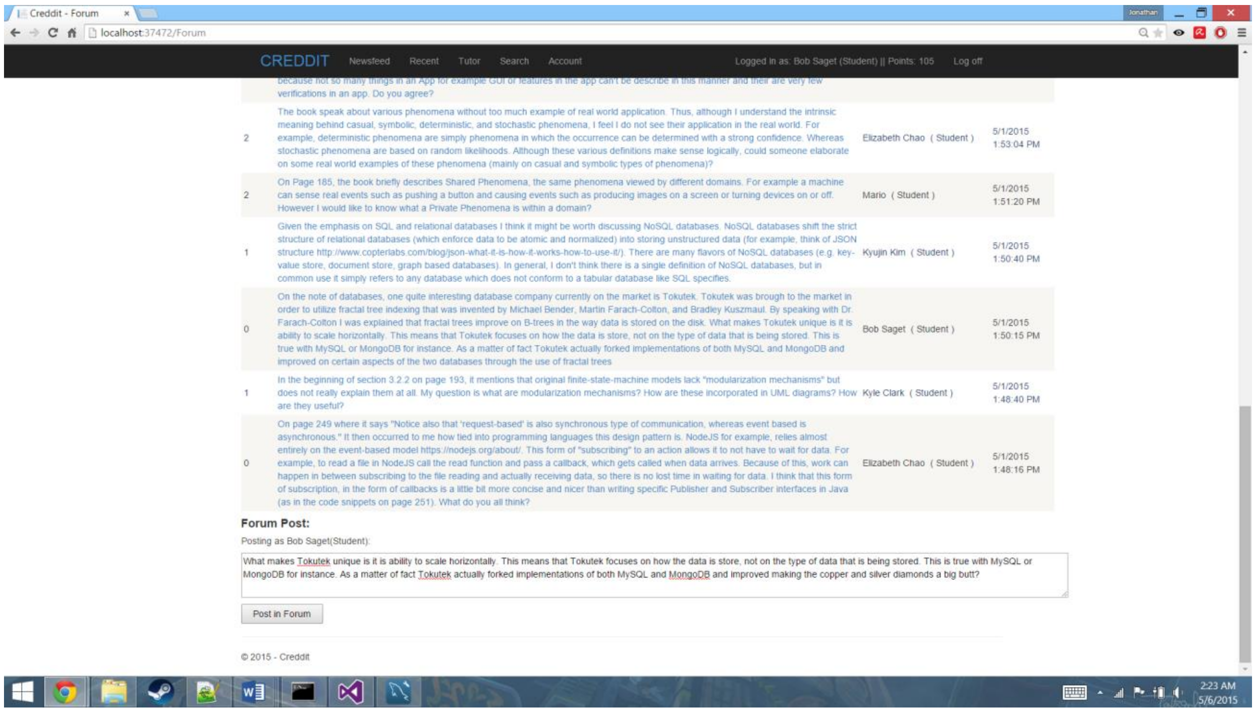
Already Voted	Points: 1	I agree with Pawel using asynchronous programming is very helpful in the area of concurrent programming. As he mentioned, before it enables us to access the database and get information, or push information before using it in the next line of code. Using this type of programming also helps make the user experience better as while using the asynchronous call we can have something like an indeterminate progress bar to let the user know that the program is doing something (like accessing the database).	Elizabeth Chao	Student	4/28/2015 11:16:59 PM	4
-	Points: 0	Will any of these asynchronous calls effect the concurrency of the program? I'm sure resources may clash in the system slowing the process down?	Blah	Faculty	4/28/2015 11:16:16 PM	3
-	Points: 0	I think what you're talking about is something like AJAX. What happens to update information without a page update is the following: The client runs Javascript code from the server that launches a new HTTP request (but does not cause a page refresh) The server does something based on the parameters in this call and responds with some kind of data (usually XML, the X in AJAX) The client Javascript knows the format of the data it will receive and uses it to update information on the page. If the site wants a page or component to constantly update, it will send Javascript to the client that runs a loop that sends these requests out at some rate. This whole process is not exactly an internet standard but it is a common enough pattern that it has a name and talked about in the same way in many different applications.	Kyujin Kim	Student	4/28/2015 11:17:30 PM	5
-	Points: 0	Using this type of programming also helps make the user experience better as while using the asynchronous call we can have something like an indeterminate progress bar to let the user know that the program is doing something (like accessing the database).	Bob Saget	Student	5/6/2015 2:22:12 AM	25

Post Comment:

Posting as Bob Saget(Student): Posting comment was successfull

2:22 AM 5/6/2015

Figure 12.2.3b



	because not so many things in an App for example GUI or features in the app can't be describe in this manner and their are very few verifications in an app. Do you agree?		
2	The book speak about various phenomena without too much example of real world application. Thus, although I understand the intrinsic meaning behind casual, symbolic, deterministic, and stochastic phenomena, I feel I do not see their application in the real world. For example, deterministic phenomena are simply phenomena in which the occurrence can be determined with a strong confidence. Whereas stochastic phenomena are based on random likelihoods. Although these various definitions make sense logically, could someone elaborate on some real world examples of these phenomena (mainly on casual and symbolic types of phenomena)?	Elizabeth Chao (Student)	5/1/2015 1:53:04 PM
2	On Page 185, the book briefly describes Shared Phenomena, the same phenomena viewed by different domains. For example a machine can sense real events such as pushing a button and causing events such as producing images on a screen or turning devices on or off. However I would like to know what a Private Phenomena is within a domain?	Mario (Student)	5/1/2015 1:51:20 PM
1	Given the emphasis on SQL and relational databases I think it might be worth discussing NoSQL databases. NoSQL databases shift the strict structure of relational databases (which enforce data to be atomic and normalized) into storing unstructured data (for example, think of JSON structure http://www.copterlabs.com/blog/json-what-it-is-how-it-works-how-to-use-it/). There are many flavors of NoSQL databases (e.g. key-value store, document store, graph based databases). In general, I don't think there is a single definition of NoSQL databases, but in common use it simply refers to any database which does not conform to a tabular database like SQL specifies.	Kyujin Kim (Student)	5/1/2015 1:50:40 PM
0	On the note of databases, one quite interesting database company currently on the market is Tokutek. Tokutek was brough to the market in order to utilize fractal tree indexing that was invented by Michael Bender, Martin Farach-Colton, and Bradley Kuszmaul. By speaking with Dr. Farach-Colton I was explained that fractal trees improve on B-trees in the way data is stored on the disk. What makes Tokutek unique is it is ability to scale horizontally. This means that Tokutek focuses on how the data is store, not on the type of data that is being stored. This is true with MySQL or MongoDB for instance. As a matter of fact Tokutek actually forked implementations of both MySQL and MongoDB and improved on certain aspects of the two databases through the use of fractal trees	Bob Saget (Student)	5/1/2015 1:50:15 PM
1	In the beginning of section 3.2.2 on page 193, it mentions that original finite-state-machine models lack "modularization mechanisms" but does not really explain them at all. My question is what are modularization mechanisms? How are these incorporated in UML diagrams? How are they useful?	Kyle Clark (Student)	5/1/2015 1:48:40 PM
0	On page 249 where it says "Notice also that 'request-based' is also synchronous type of communication, whereas event based is asynchronous." It then occurred to me how tied into programming languages this design pattern is. NodeJS for example, relies almost entirely on the event-based model https://nodejs.org/about/. This form of "subscribing" to an action allows it to not have to wait for data. For example, to read a file in NodeJS call the read function and pass a callback, which gets called when data arrives. Because of this, work can happen in between subscribing to the file reading and actually receiving data, so there is no lost time in waiting for data. I think that this form of subscription, in the form of callbacks is a little bit more concise and nicer than writing specific Publisher and Subscriber interfaces in Java (as in the code snippets on page 251). What do you all think?	Elizabeth Chao (Student)	5/1/2015 1:48:16 PM

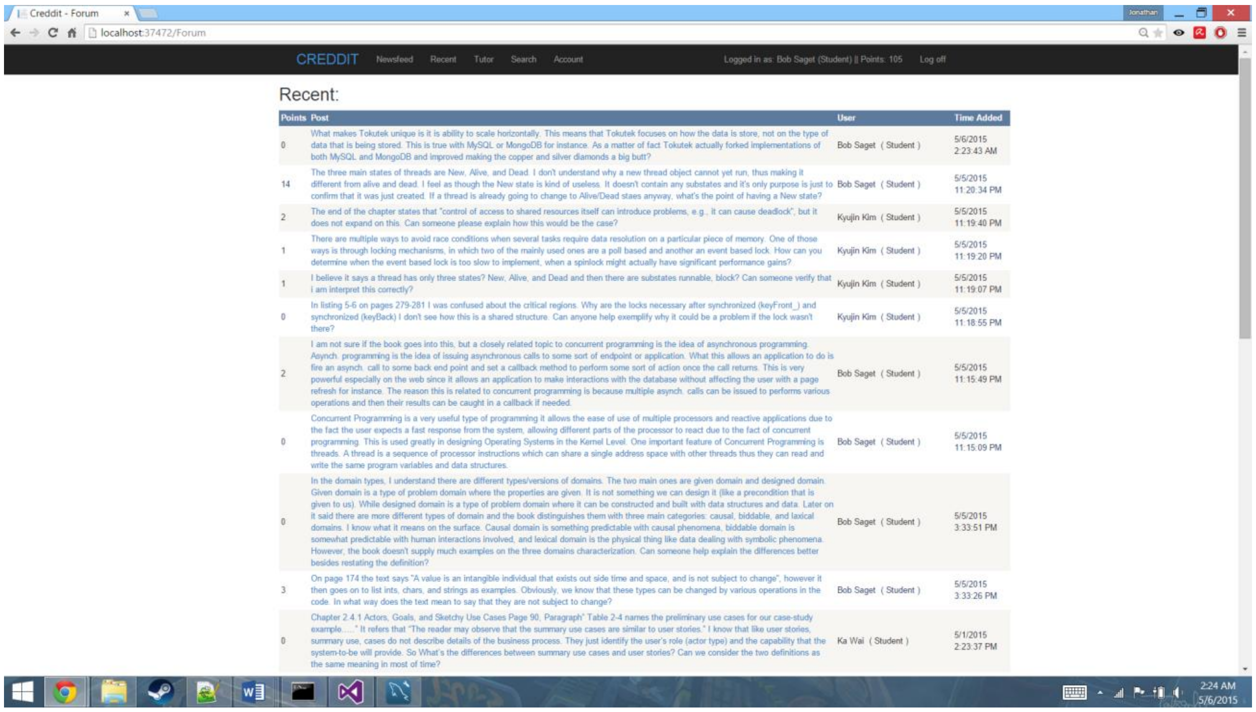
Forum Post:

Posting as Bob Saget(Student):

What makes Tokutek unique is it is ability to scale horizontally. This means that Tokutek focuses on how the data is store, not on the type of data that is being stored. This is true with MySQL or MongoDB for instance. As a matter of fact Tokutek actually forked implementations of both MySQL and MongoDB and improved making the copper and silver diamonds a big butt?

© 2015 - Creddit

Figure 12.2.4a

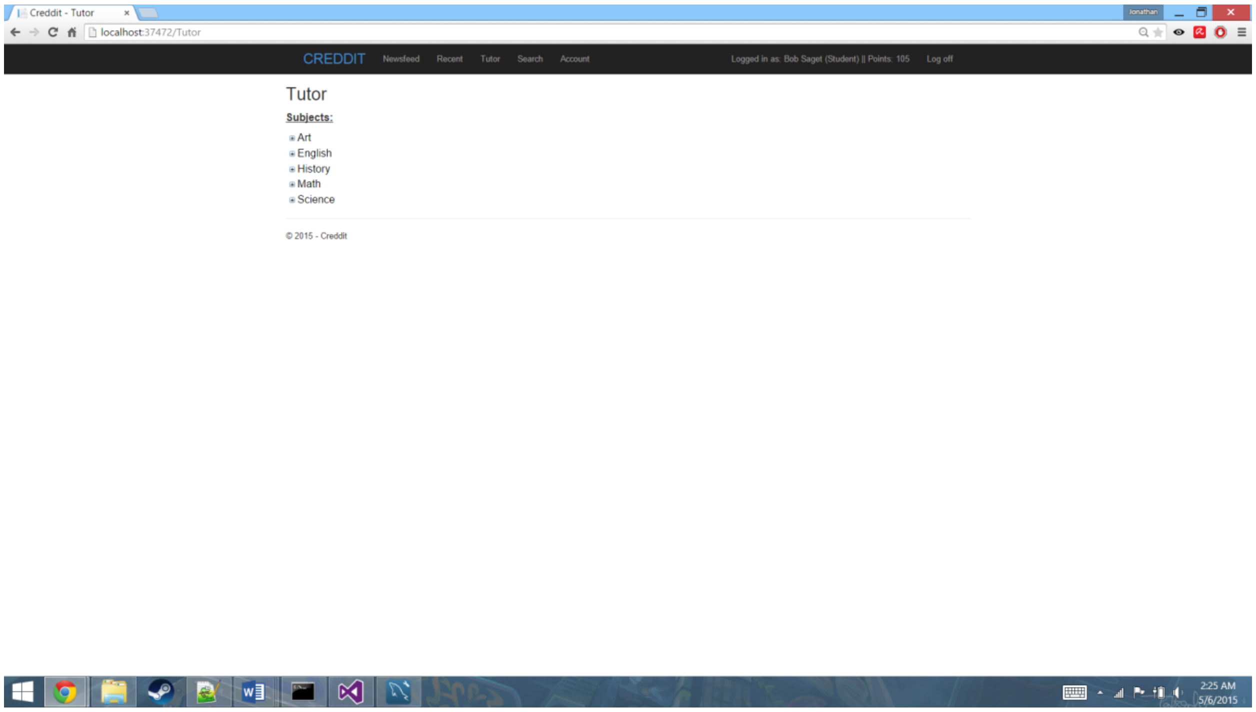


Recent:

Points	Post	User	Time Added
0	What makes Tokutek unique is it is ability to scale horizontally. This means that Tokutek focuses on how the data is store, not on the type of data that is being stored. This is true with MySQL or MongoDB for instance. As a matter of fact Tokutek actually forked implementations of both MySQL and MongoDB and improved making the copper and silver diamonds a big butt?	Bob Saget (Student)	5/6/2015 2:23:43 AM
14	The three main states of threads are New, Alive, and Dead. I don't understand why a new thread object cannot yet run, thus making it different from alive and dead. I feel as though the New state is kind of useless. It doesn't contain any substates and it's only purpose is just to confirm that it was just created. If a thread is already going to change to Alive/Dead staes anyway, what's the point of having a New state?	Bob Saget (Student)	5/5/2015 11:20:34 PM
2	The end of the chapter states that "control of access to shared resources itself can introduce problems, e.g., it can cause deadlock", but it does not expand on this. Can someone please explain how this would be the case?	Kyujin Kim (Student)	5/5/2015 11:19:40 PM
1	There are multiple ways to avoid race conditions when several tasks require data resolution on a particular piece of memory. One of those ways is through locking mechanisms, in which two of the mainly used ones are a poll based and another an event based lock. How can you determine when the event based lock is too slow to implement, when a spinlock might actually have significant performance gains?	Kyujin Kim (Student)	5/5/2015 11:19:20 PM
1	I believe it says a thread has only three states? New, Alive, and Dead and then there are substates runnable, block? Can someone verify that i am interpret this correctly?	Kyujin Kim (Student)	5/5/2015 11:19:07 PM
0	In listing 5-6 on pages 279-281 I was confused about the critical regions. Why are the locks necessary after synchronized (keyFront_) and synchronized (keyBack) I don't see how this is a shared structure. Can anyone help exemplify why it could be a problem if the lock wasn't there?	Kyujin Kim (Student)	5/5/2015 11:18:55 PM
2	I am not sure if the book goes into this, but a closely related topic to concurrent programming is the idea of asynchronous programming. Asynch. programming is the idea of issuing asynchronous calls to some sort of endpoint or application. What this allows an application to do is fire an asynch. call to some back end point and set a callback method to perform some sort of action once the call returns. This is very powerful especially on the web since it allows an application to make interactions with the database without affecting the user with a page refresh for instance. The reason this is related to concurrent programming is because multiple asynch. calls can be issued to performs various operations and then their results can be caught in a callback if needed.	Bob Saget (Student)	5/5/2015 11:15:49 PM
0	Concurrent Programming is a very useful type of programming it allows the ease of use of multiple processors and reactive applications due to the fact the user expects a fast response from the system, allowing different parts of the processor to react due to the fact of concurrent programming. This is used greatly in designing Operating Systems in the Kernel Level. One important feature of Concurrent Programming is threads. A thread is a sequence of processor instructions which can share a single address space with other threads thus they can read and write the same program variables and data structures.	Bob Saget (Student)	5/5/2015 11:15:09 PM
0	In the domain types, I understand there are different types/versions of domains. The two main ones are given domain and designed domain. Given domain is a type of problem domain where the properties are given. It is not something we can design it (like a precondition that is given to us). While designed domain is a type of problem domain where it can be constructed and built with data structures and data. Later on it said there are more different types of domain and the book distinguishes them with three main categories: causal, biddable, and laxical domains. I know what it means on the surface. Causal domain is something predictable with causal phenomena, biddable domain is somewhat predictable with human interactions involved, and lexical domain is the physical thing like data dealing with symbolic phenomena. However, the book doesn't supply much examples on the three domains characterization. Can someone help explain the differences better besides restating the definition?	Bob Saget (Student)	5/5/2015 3:33:51 PM
3	On page 174 the text says "A value is an intangible individual that exists out side time and space, and is not subject to change", however it then goes on to list ints, chars, and strings as examples. Obviously, we know that these types can be changed by various operations in the code. In what way does the text mean to say that they are not subject to change?	Bob Saget (Student)	5/5/2015 3:33:26 PM
0	Chapter 2.4.1 Actors, Goals, and Sketchy Use Cases Page 90, Paragraph" Table 2-4 names the preliminary use cases for our case-study example….." It refers that "The reader may observe that the summary use cases are similar to user stories." I know that like user stories, summary use, cases do not describe details of the business process. They just identify the user's role (actor type) and the capability that the system-to-be will provide. So What's the differences between summary use cases and user stories? Can we consider the two definitions as the same meaning in most of time?	Ka Wai (Student)	5/1/2015 2:23:37 PM

Figure 12.2.4b

3. Providing Tutoring (UC-2)



Creddit - Tutor

localhost:37472/Tutor

CREDDIT Newsfeed Recent Tutor Search Account Logged in as: Bob Saget (Student) || Points: 105 Log off

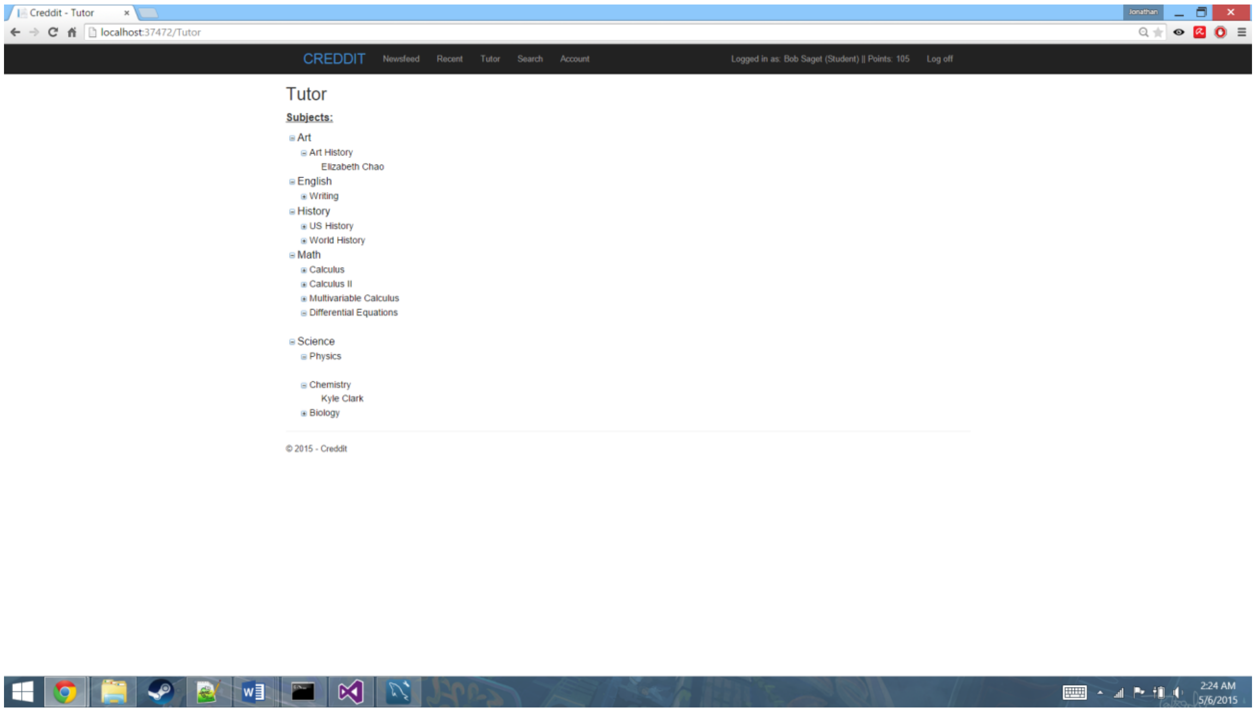
Tutor

Subjects:

- ⊞ Art
- ⊞ English
- ⊞ History
- ⊞ Math
- ⊞ Science

© 2015 - Creddit

Figure 12.3.1a



Creddit - Tutor

localhost:37472/Tutor

CREDDIT Newsfeed Recent Tutor Search Account Logged in as: Bob Saget (Student) || Points: 105 Log off

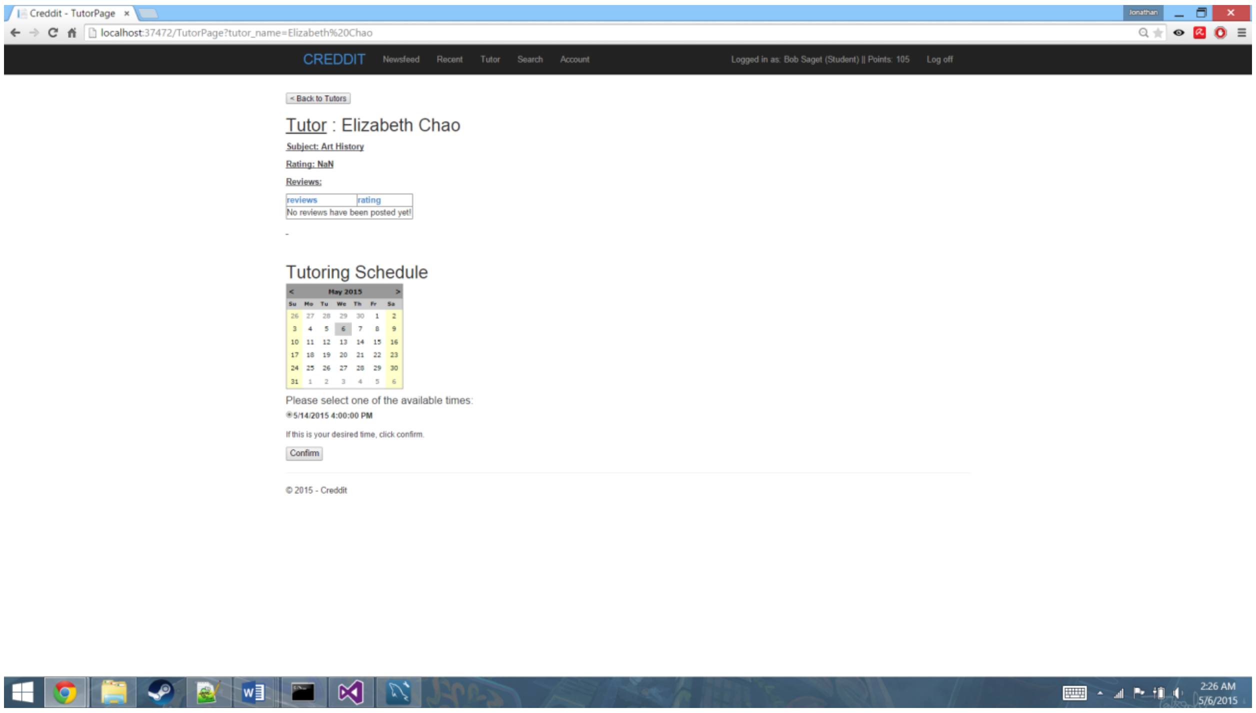
Tutor

Subjects:

- ⊟ Art
  - ⊟ Art History
    - Elizabeth Chao
- ⊟ English
  - ⊞ Writing
- ⊟ History
  - ⊞ US History
  - ⊞ World History
- ⊟ Math
  - ⊞ Calculus
  - ⊞ Calculus II
  - ⊞ Multivariable Calculus
  - ⊟ Differential Equations
- ⊟ Science
  - ⊟ Physics
  - ⊟ Chemistry
    - Kyle Clark
  - ⊞ Biology

© 2015 - Creddit

Figure 12.3.1b



Tutor : Elizabeth Chao

Subject: Art History

Rating: NaN

Reviews:

reviews	rating
No reviews have been posted yet!	

Tutoring Schedule

<			May 2015			>
Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

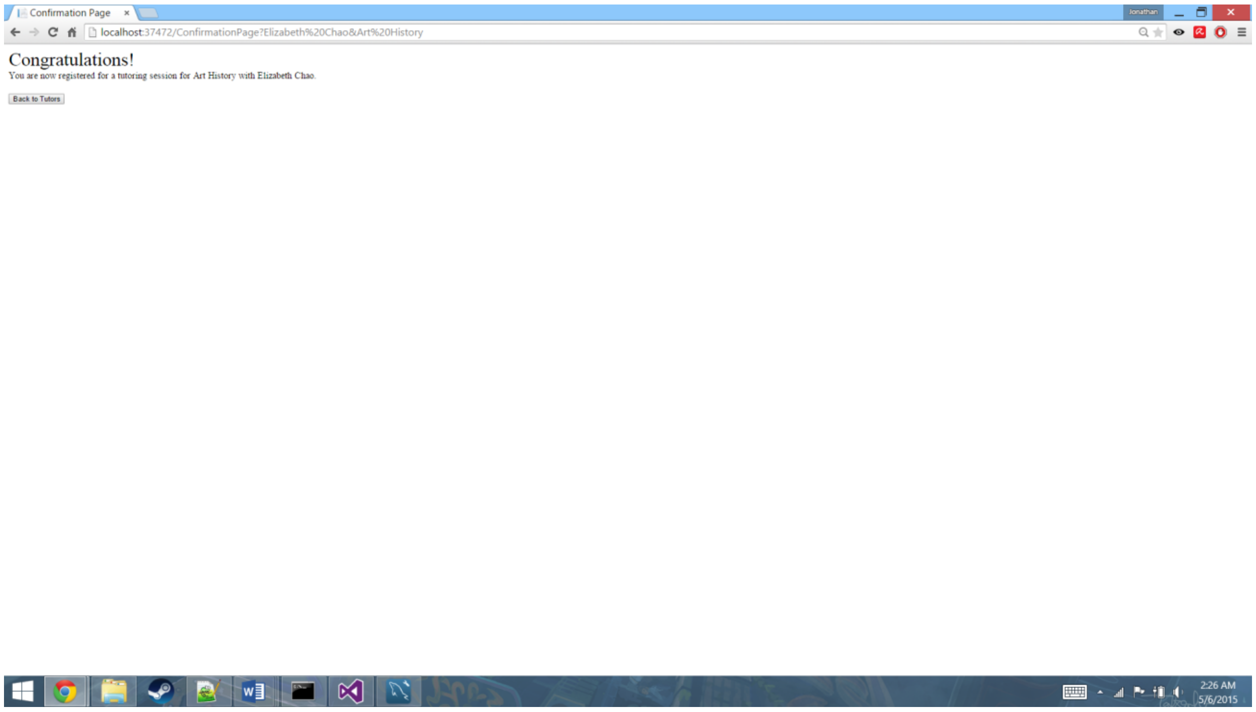
Please select one of the available times:

● 5/14/2015 4:00:00 PM

If this is your desired time, click confirm.

© 2015 - Creddit

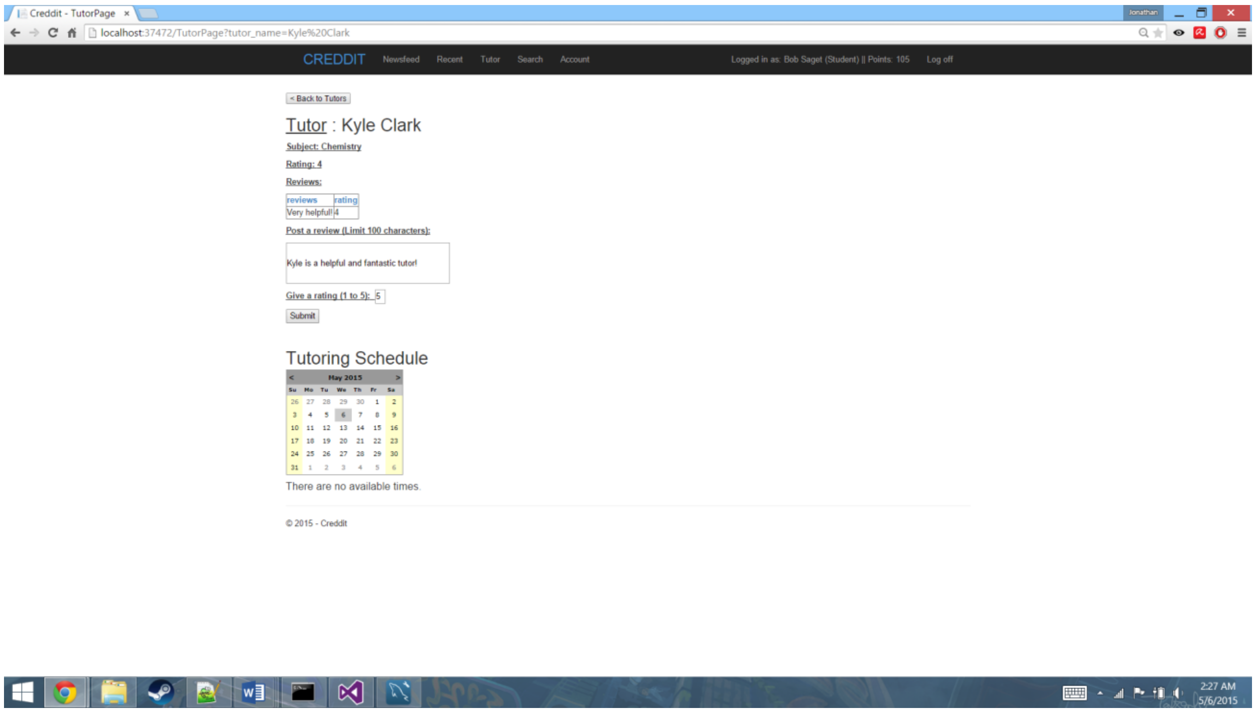
Figure 12.3.2a


Confirmation Page
localhost:37472/ConfirmationPage?Elizabeth%20Chao&Art%20History
Back to Tutors
2:26 AM
5/6/2015

Congratulations!

You are now registered for a tutoring session for Art History with Elizabeth Chao.

Figure 12.3.2b



localhost:37472/TutorPage?tutor_name=Kyle%20Clark

Logged in as: Bob Saget (Student) || Points: 105

Tutor : Kyle Clark

Subject: Chemistry

Rating: 4

Reviews:

reviews	rating
Very helpful!	4

Post a review (Limit 100 characters):

Kyle is a helpful and fantastic tutor!

Give a rating (1 to 5): 5

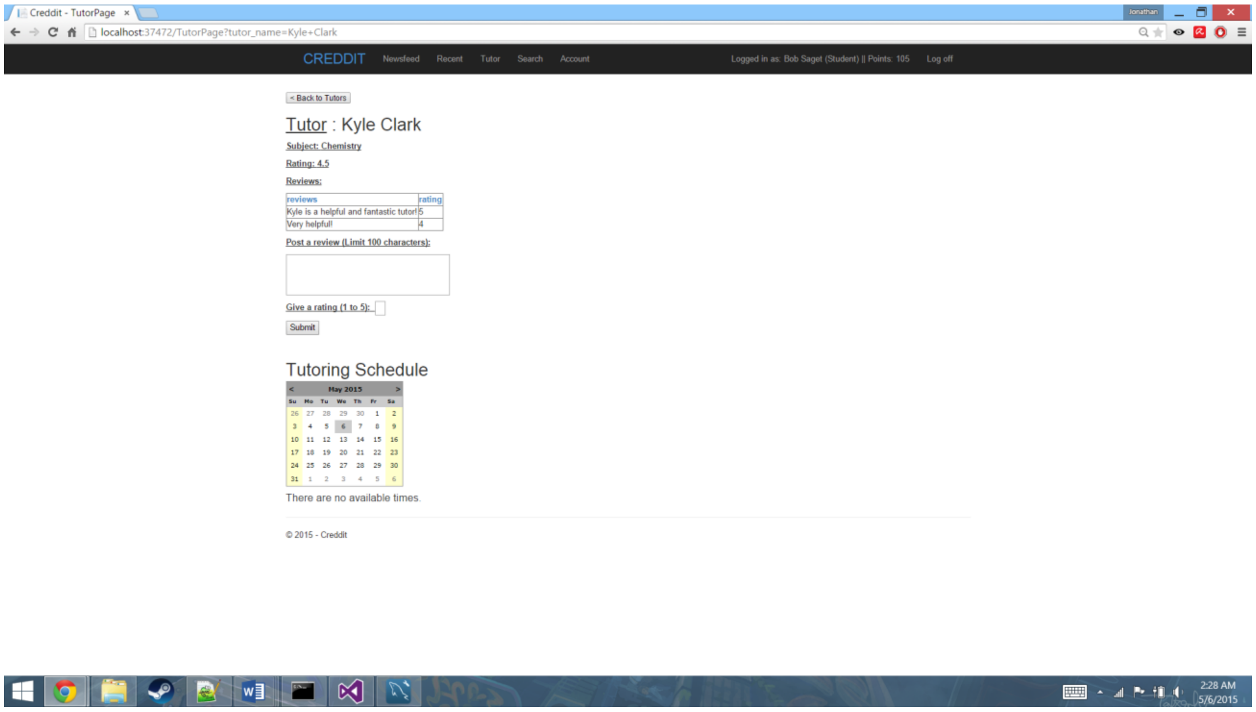
Tutoring Schedule

Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

There are no available times.

© 2015 - Creddit

Figure 12.3.3a



localhost:37472/TutorPage?tutor_name=Kyle+Clark

Logged in as: Bob Saget (Student) || Points: 105

Tutor : Kyle Clark

Subject: Chemistry

Rating: 4.5

Reviews:

reviews	rating
Kyle is a helpful and fantastic tutor!	5
Very helpful!	4

Post a review (Limit 100 characters):

Give a rating (1 to 5):

Tutoring Schedule

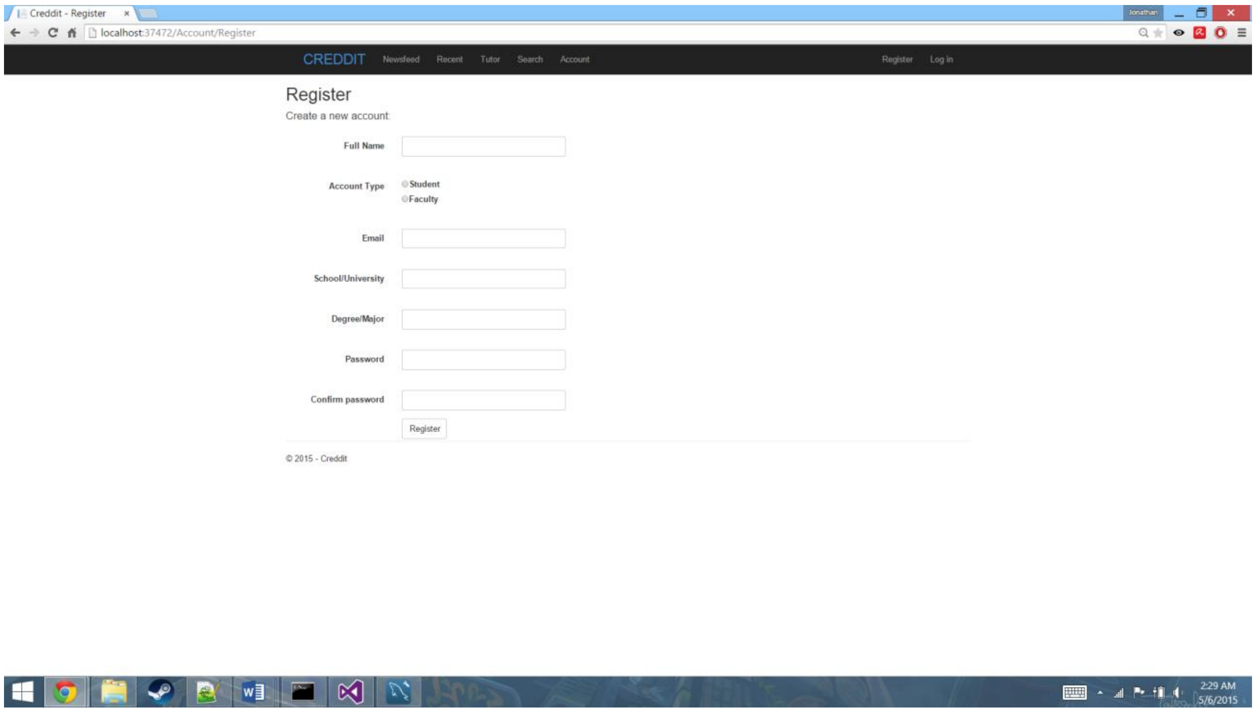
Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

There are no available times.

© 2015 - Creddit

Figure 12.3.3b

4. Sign-up (UC-3)



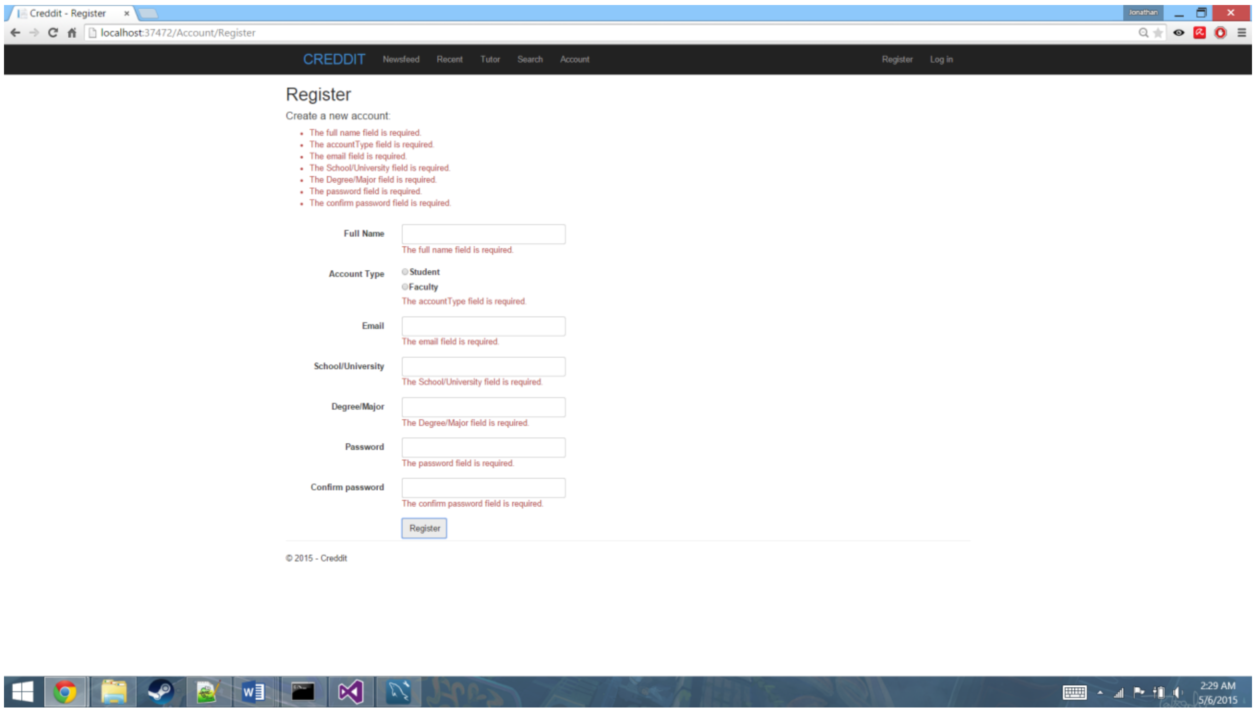
Register

Create a new account:

Full Name	
Account Type	Student Faculty
Email	
School/University	
Degree/Major	
Password	
Confirm password	

© 2015 - Creddit

Figure 12.4.1a



Register

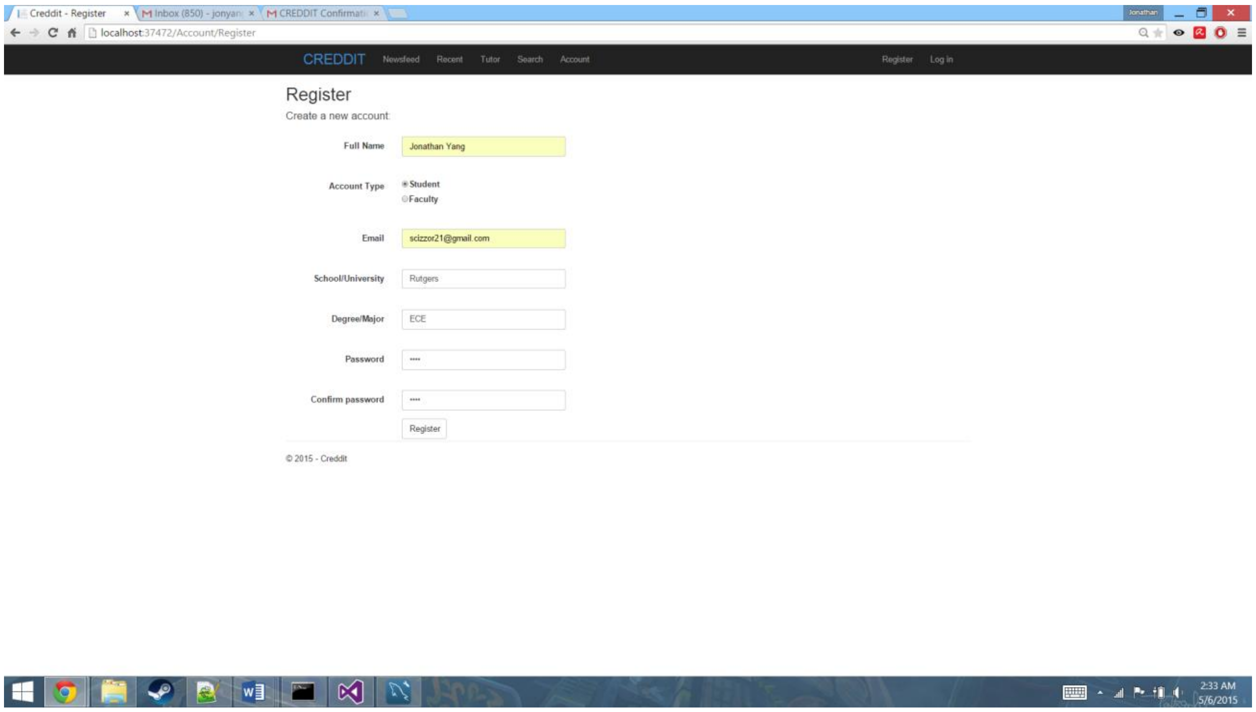
Create a new account:

- The full name field is required.
- The accountType field is required.
- The email field is required.
- The School/University field is required.
- The Degree/Major field is required.
- The password field is required.
- The confirm password field is required.

Full Name	The full name field is required.
Account Type	Student Faculty The accountType field is required.
Email	The email field is required.
School/University	The School/University field is required.
Degree/Major	The Degree/Major field is required.
Password	The password field is required.
Confirm password	The confirm password field is required.

© 2015 - Creddit

Figure 12.4.1b



Register

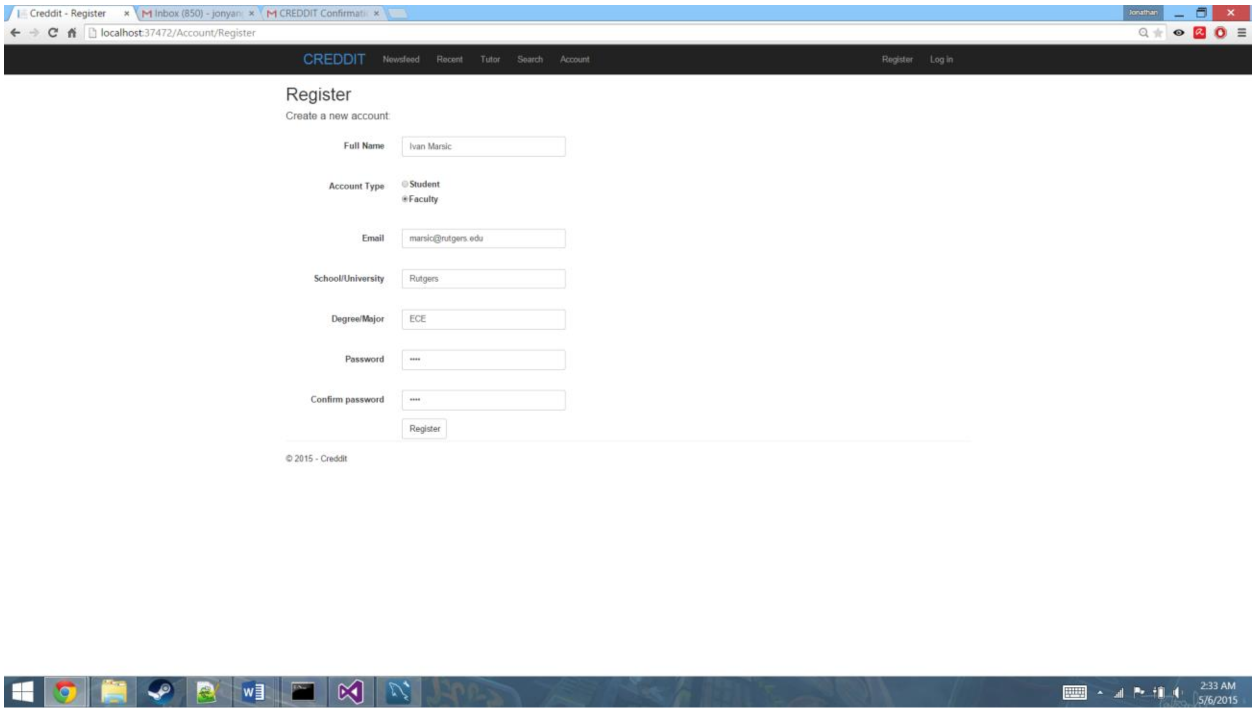
Create a new account.

Full Name	Jonathan Yang
Account Type	[✓ Student] Faculty
Email	scizzor21@gmail.com
School/University	Rutgers
Degree/Major	ECE
Password	••••
Confirm password	••••

Register

© 2015 - Creddit

Figure 12.4.2a



Register

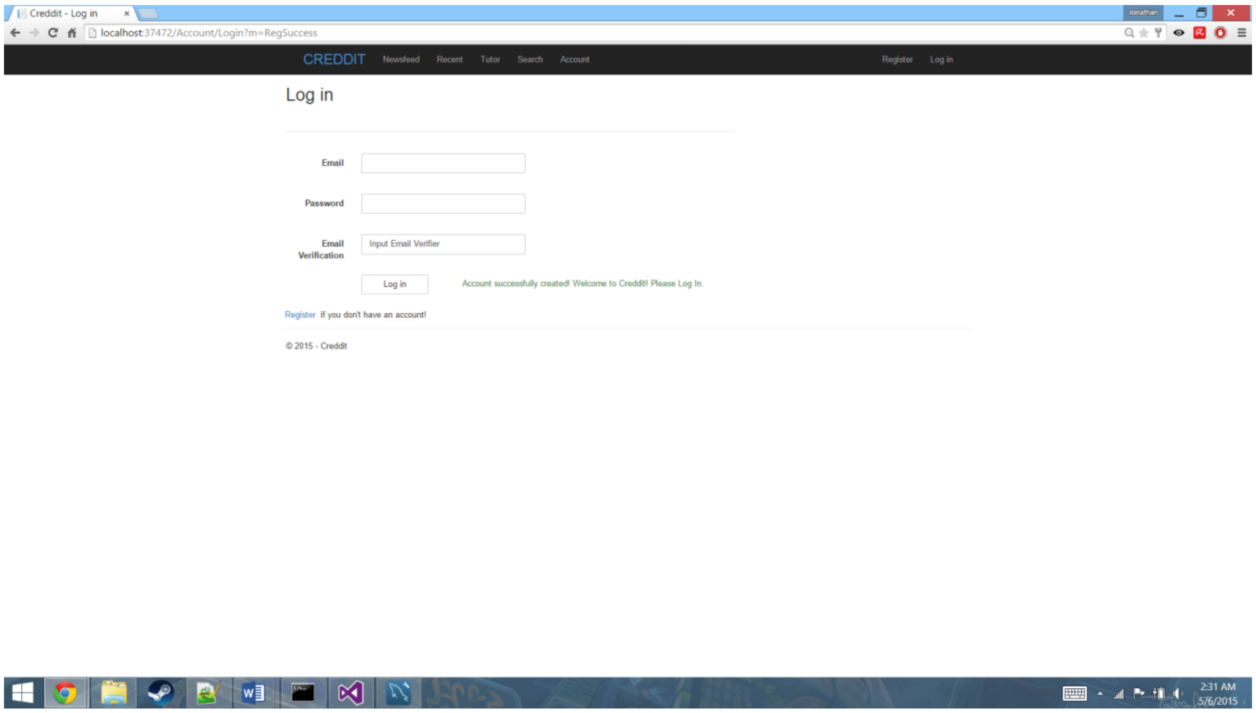
Create a new account.

Full Name	Ivan Marsic
Account Type	Student [✓ Faculty]
Email	marsic@rutgers.edu
School/University	Rutgers
Degree/Major	ECE
Password	••••
Confirm password	••••

Register

© 2015 - Creddit

Figure 12.4.2b



Log in

Email

Password

Email Verification

Input Email Verifier

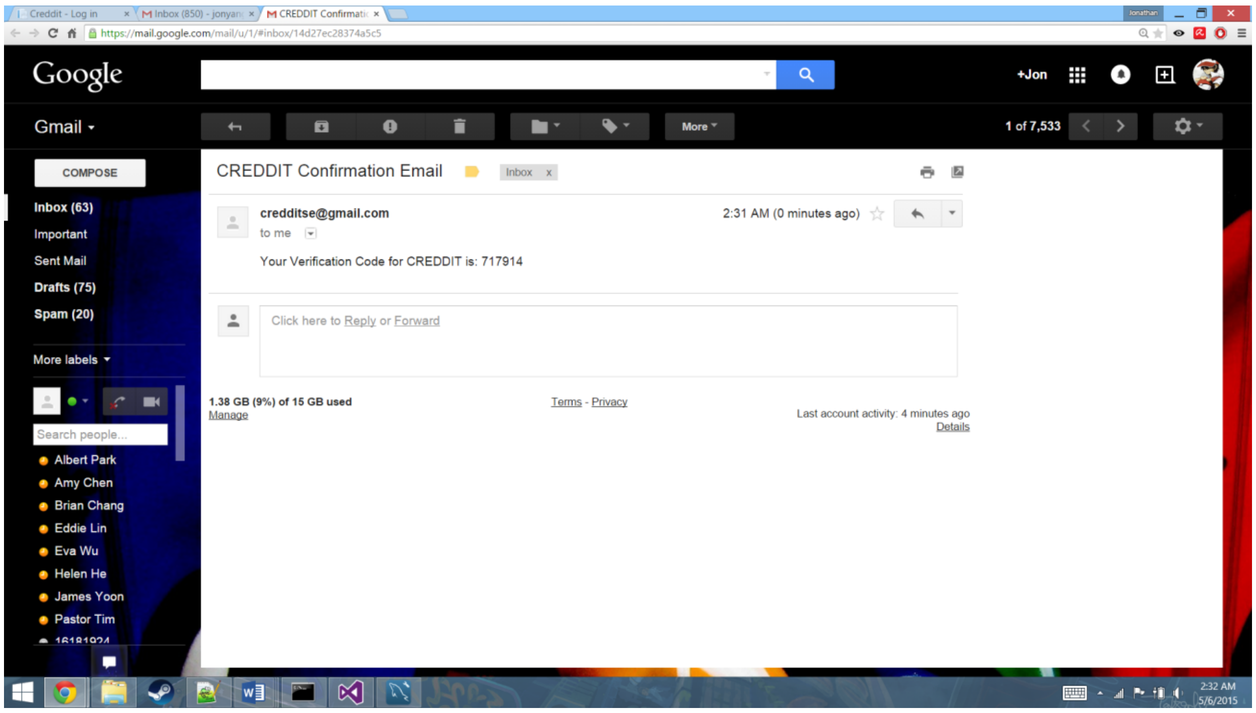
Log in

Account successfully created! Welcome to Creddit! Please Log In.

Register if you don't have an account!

© 2015 - Creddit

Figure 12.4.3a



CREDDIT Confirmation Email Inbox x

credditse@gmail.com 2:31 AM (0 minutes ago)

to me

Your Verification Code for CREDDIT is: 717914

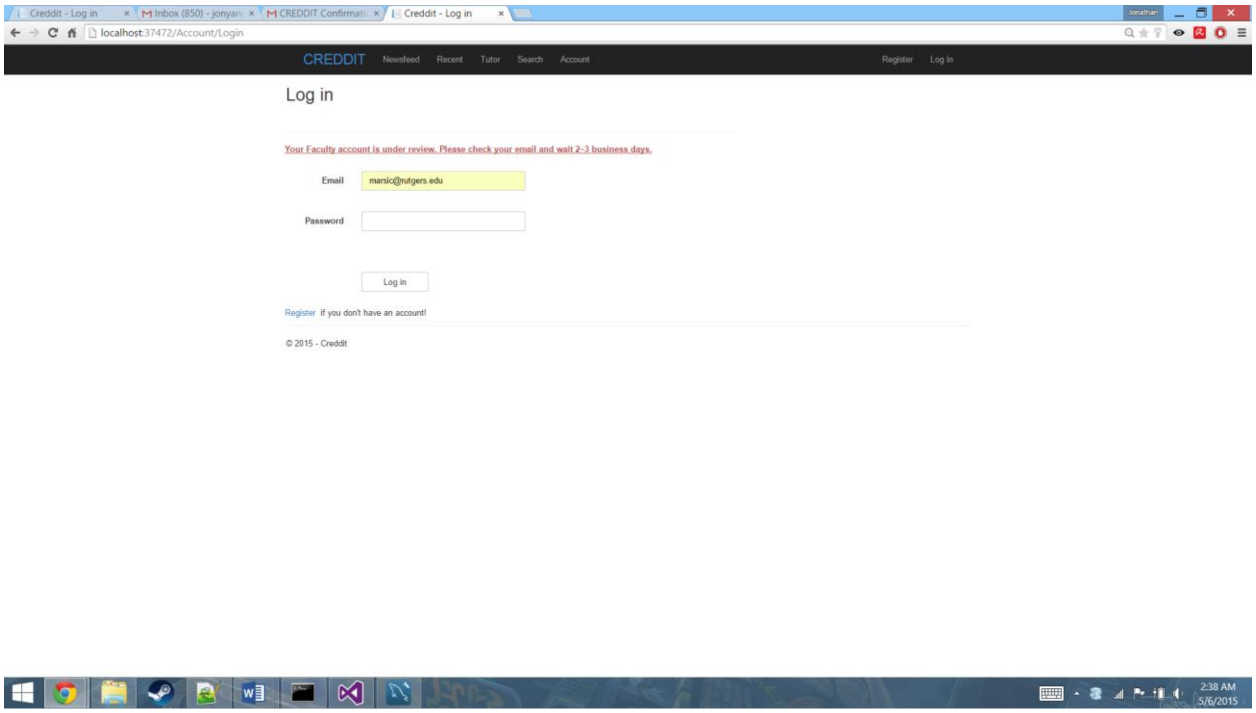
Click here to Reply or Forward

1.38 GB (9%) of 15 GB used
Manage

Terms - Privacy

Last account activity: 4 minutes ago
Details

Figure 12.4.3b



Log in

Your Faculty account is under review. Please check your email and wait 2-3 business days.

Email: marsic@rutgers.edu

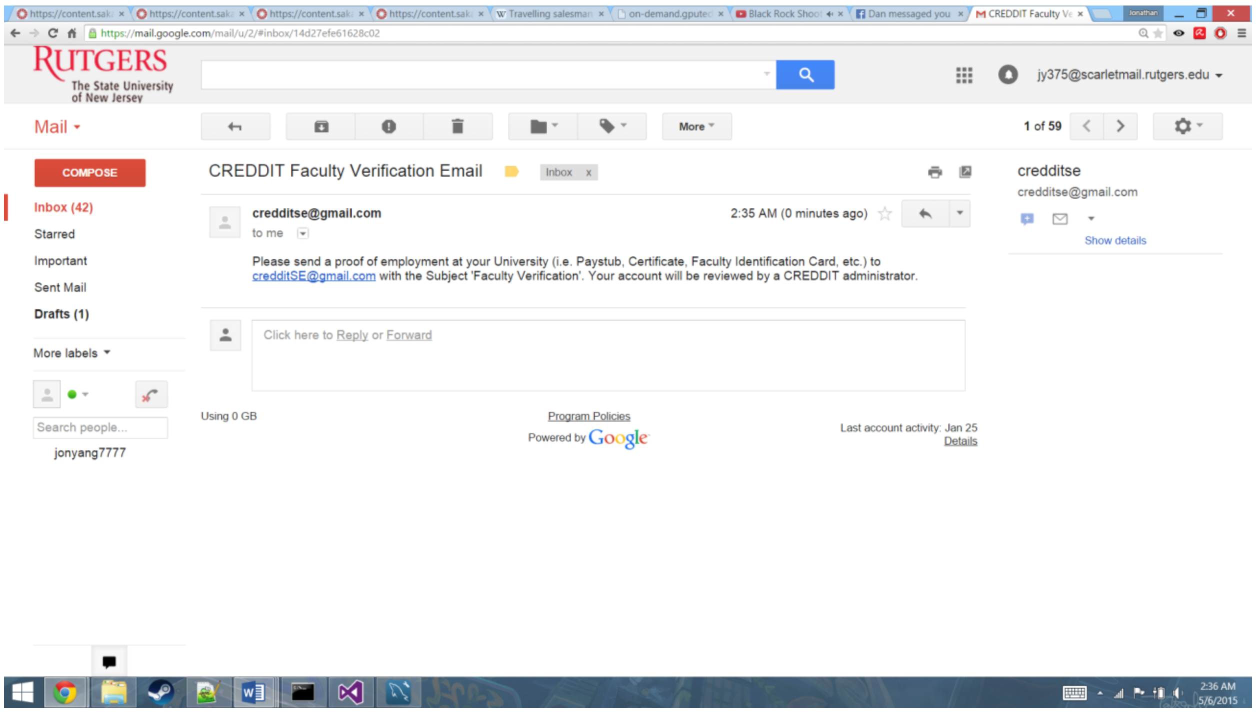
Password:

Log in

Register if you don't have an account!

© 2015 - Creddit

Figure 12.4.4a


https://mail.google.com/mail/u/2/#inbox/14d27efe61628c02
jy375@scarletmail.rutgers.edu
Mail
COMPOSE
Inbox (42)
Starred
Important
Sent Mail
Drafts (1)
More labels
More
1 of 59

CREDDIT Faculty Verification Email

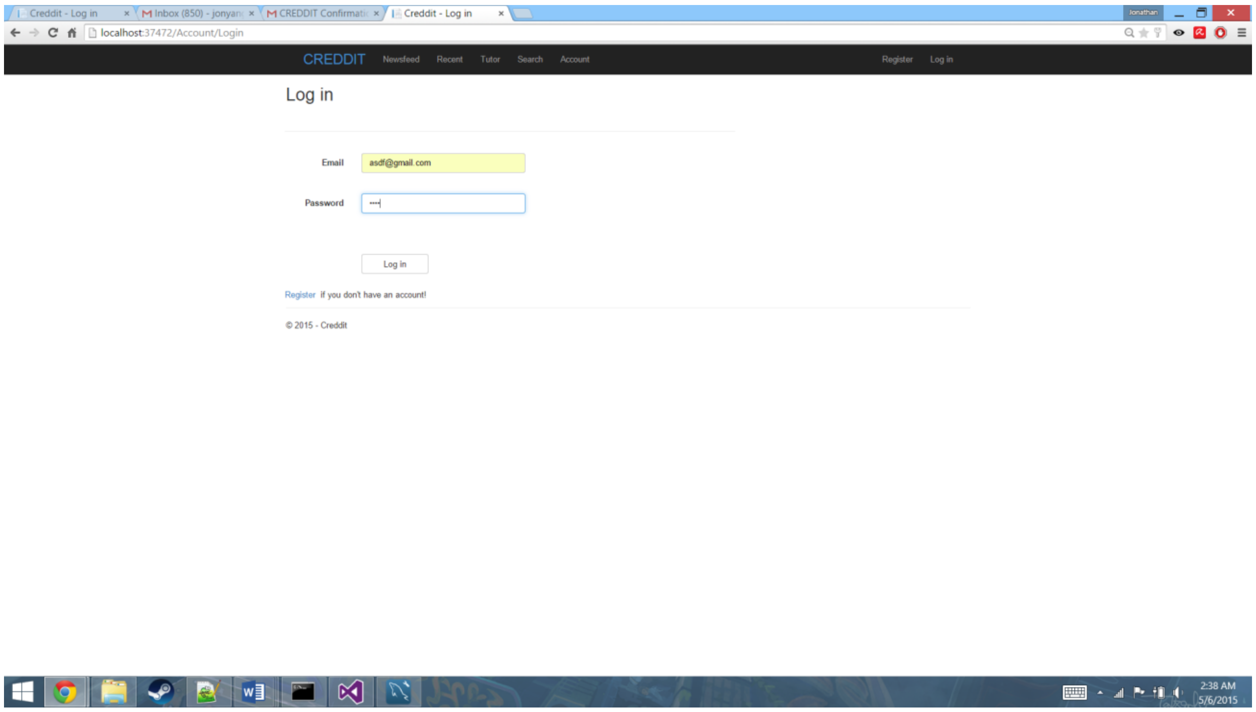
creddítse@gmail.com 2:35 AM (0 minutes ago)
to me

Please send a proof of employment at your University (i.e. Paystub, Certificate, Faculty Identification Card, etc.) to credditSE@gmail.com with the Subject 'Faculty Verification'. Your account will be reviewed by a CREDDIT administrator.

Click here to Reply or Forward

Figure 12.4.4b

5. Login (UC-4)



localhost:37472/Account/Login

Log in

Email asdf@gmail.com

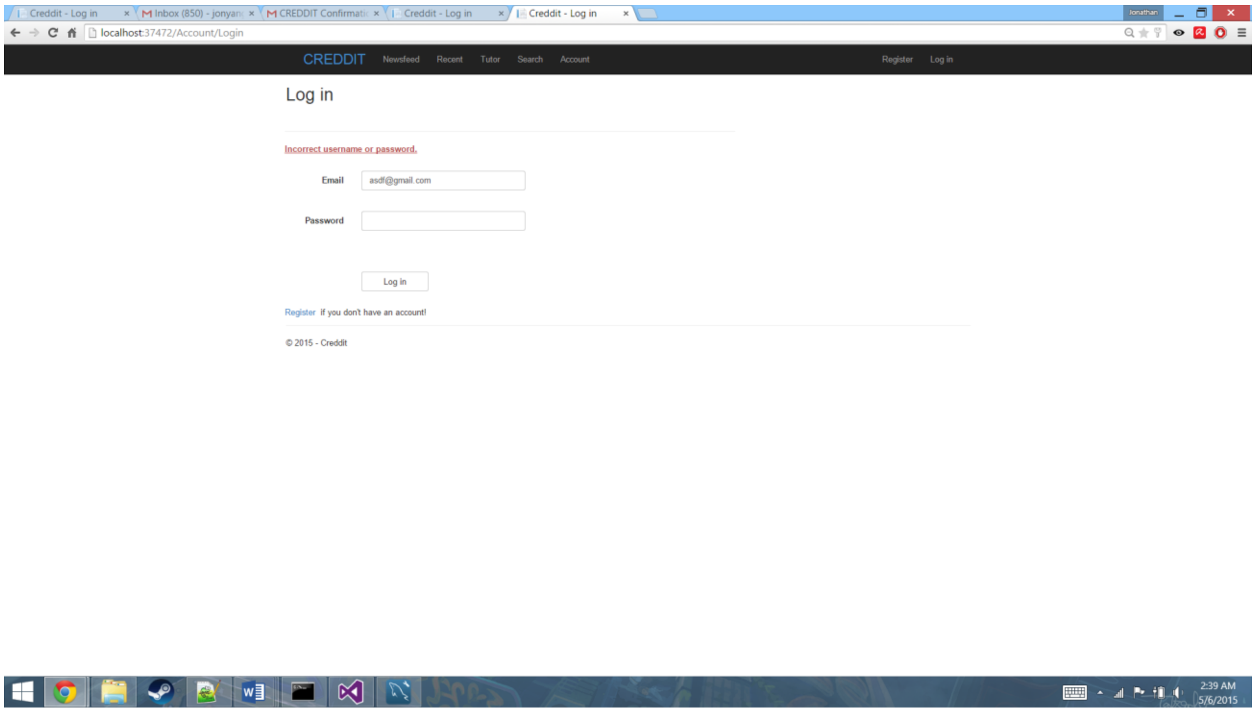
Password ····

Log in

Register if you don't have an account!

© 2015 - Creddit

Figure 12.5.1a



localhost:37472/Account/Login

Log in

Incorrect username or password.

Email asdf@gmail.com

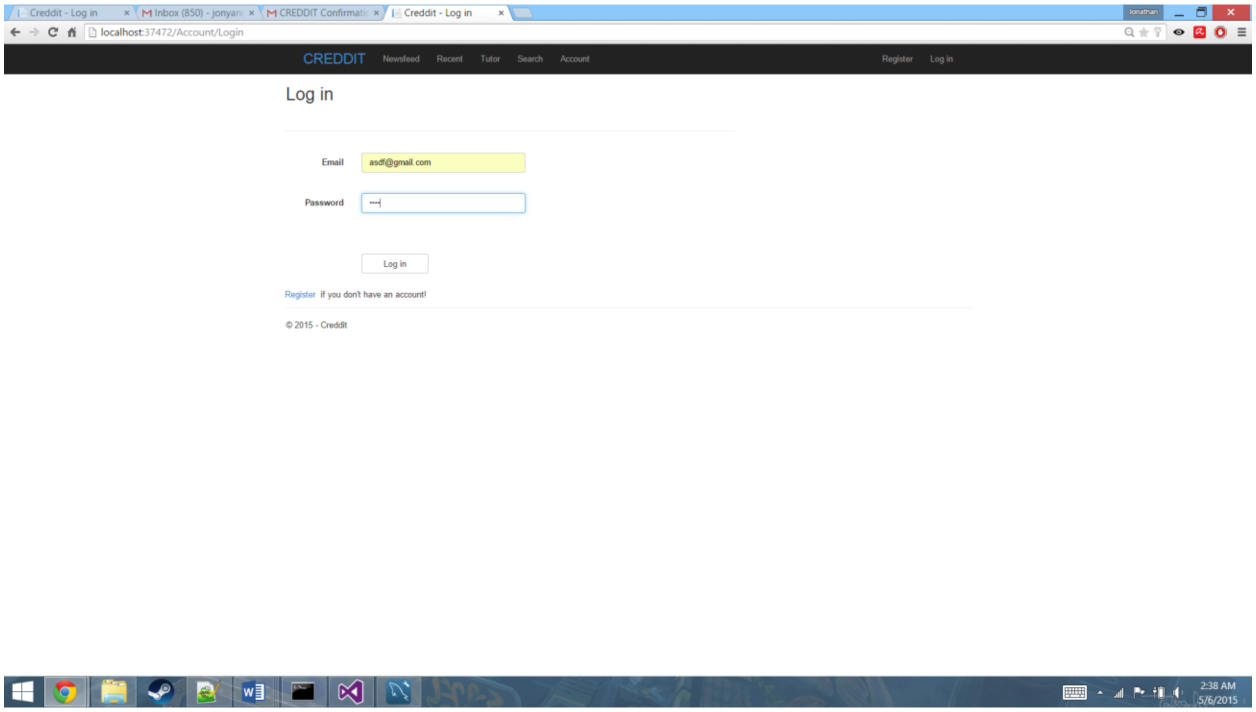
Password

Log in

Register if you don't have an account!

© 2015 - Creddit

Figure 12.5.1b



Log in

Email: asdf@gmail.com

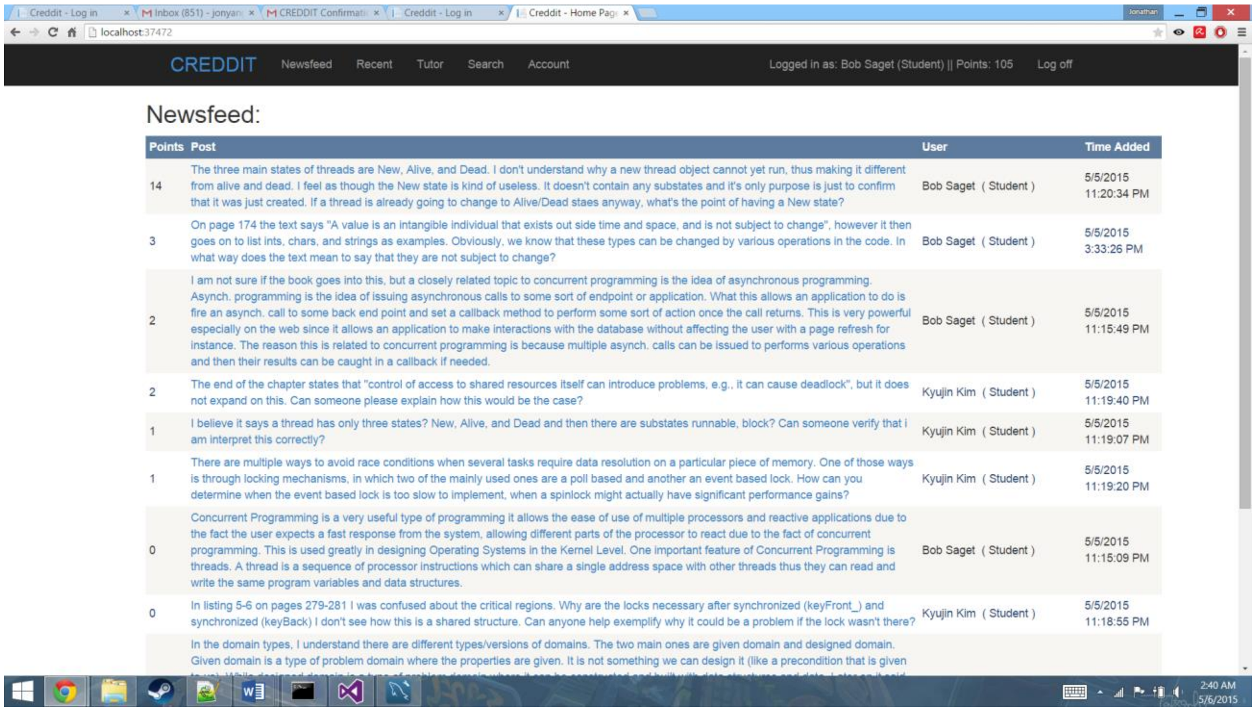
Password: ••••

Log in

Register if you don't have an account!

© 2015 - Creddit

Figure 12.5.2a



Newsfeed:

Points	Post	User	Time Added
14	The three main states of threads are New, Alive, and Dead. I don't understand why a new thread object cannot yet run, thus making it different from alive and dead. I feel as though the New state is kind of useless. It doesn't contain any substates and it's only purpose is just to confirm that it was just created. If a thread is already going to change to Alive/Dead staes anyway, what's the point of having a New state?	Bob Saget (Student)	5/5/2015 11:20:34 PM
3	On page 174 the text says "A value is an intangible individual that exists out side time and space, and is not subject to change", however it then goes on to list ints, chars, and strings as examples. Obviously, we know that these types can be changed by various operations in the code. In what way does the text mean to say that they are not subject to change?	Bob Saget (Student)	5/5/2015 3:33:26 PM
2	I am not sure if the book goes into this, but a closely related topic to concurrent programming is the idea of asynchronous programming. Asynch. programming is the idea of issuing asynchronous calls to some sort of endpoint or application. What this allows an application to do is fire an asynch. call to some back end point and set a callback method to perform some sort of action once the call returns. This is very powerful especially on the web since it allows an application to make interactions with the database without affecting the user with a page refresh for instance. The reason this is related to concurrent programming is because multiple asynch. calls can be issued to performs various operations and then their results can be caught in a callback if needed.	Bob Saget (Student)	5/5/2015 11:15:49 PM
2	The end of the chapter states that "control of access to shared resources itself can introduce problems, e.g., it can cause deadlock", but it does not expand on this. Can someone please explain how this would be the case?	Kyujin Kim (Student)	5/5/2015 11:19:40 PM
1	I believe it says a thread has only three states? New, Alive, and Dead and then there are substates runnable, block? Can someone verify that i am interpret this correctly?	Kyujin Kim (Student)	5/5/2015 11:19:07 PM
1	There are multiple ways to avoid race conditions when several tasks require data resolution on a particular piece of memory. One of those ways is through locking mechanisms, in which two of the mainly used ones are a poll based and another an event based lock. How can you determine when the event based lock is too slow to implement, when a spinlock might actually have significant performance gains?	Kyujin Kim (Student)	5/5/2015 11:19:20 PM
0	Concurrent Programming is a very useful type of programming it allows the ease of use of multiple processors and reactive applications due to the fact the user expects a fast response from the system, allowing different parts of the processor to react due to the fact of concurrent programming. This is used greatly in designing Operating Systems in the Kernel Level. One important feature of Concurrent Programming is threads. A thread is a sequence of processor instructions which can share a single address space with other threads thus they can read and write the same program variables and data structures.	Bob Saget (Student)	5/5/2015 11:15:09 PM
0	In listing 5-6 on pages 279-281 I was confused about the critical regions. Why are the locks necessary after synchronized (keyFront_) and synchronized (keyBack) I don't see how this is a shared structure. Can anyone help exemplify why it could be a problem if the lock wasn't there?	Kyujin Kim (Student)	5/5/2015 11:18:55 PM
	In the domain types, I understand there are different types/versions of domains. The two main ones are given domain and designed domain. Given domain is a type of problem domain where the properties are given. It is not something we can design it (like a precondition that is given		

Figure 12.5.2b

12. Design of Tests

The following test cases will be programmed and used for unit testing: Login, Sign up, Posting, Tutor Session, and Advance Search. This section will only describe how the test cases are designed. For further details, the algorithms and user interface requirements will be tested in the demo.

Integration Test links all to most unit tests into one testing component of the system. For this system, the Integration Testing strategy that will be used is Vertical Integration. This method is the better approach to develop user stories in parallel. Each cases starts with the user working on the acceptance tests, which will test a particular user case. It seems reasonable to write the unit tests based on acceptance tests and use cases because there are necessary code that is relevant to it.

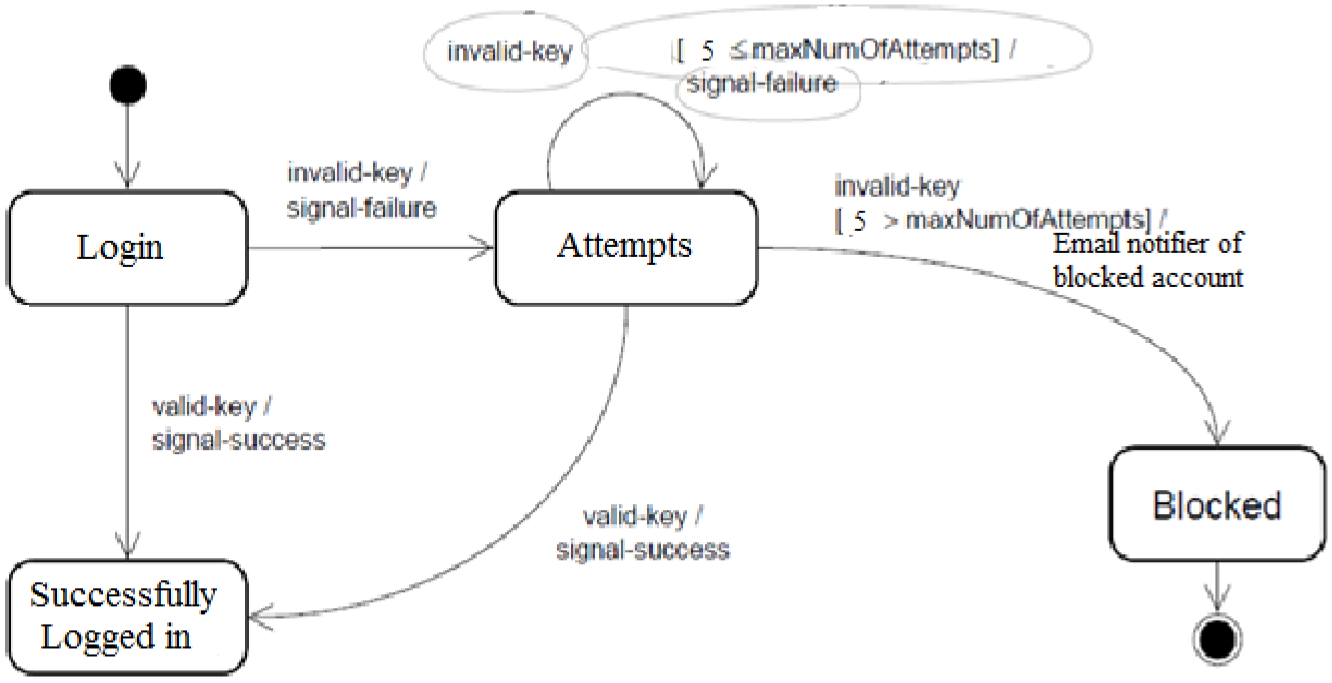


Figure 13.1 - State Diagram of Account Authenticator (Login)

Account Authenticator State Diagram:

- States {Blocked, Attempts, Login, Successfully Login}
- Events {valid-key, invalid-key}
- Valid Transitions {Login → Successfully login, Login→ Attempts, Attempts→ Attempts, Attempts→ Successfully login, Attempts→ blocked}

In this test case, the Account Authenticator will be tested to verify if users can access their respective account successfully. The test will cover the different attempts of successes and failures in logging into the system.

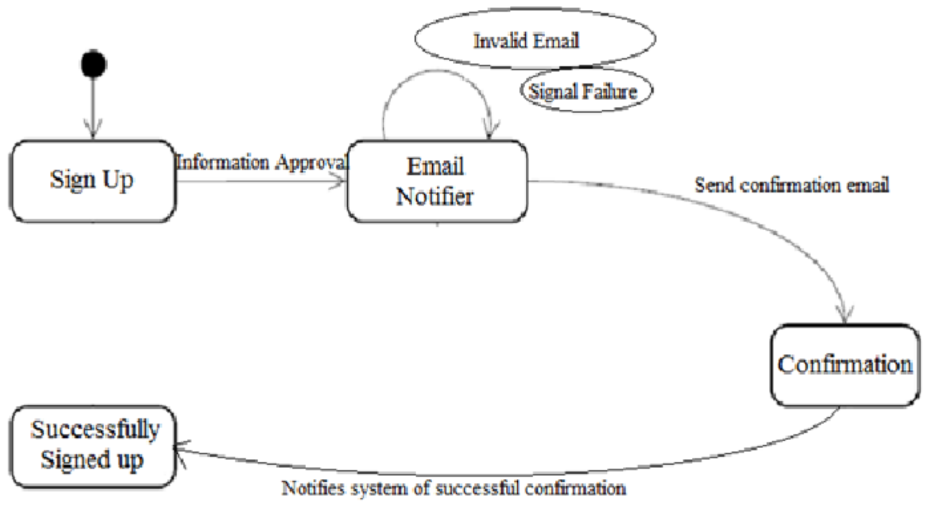


Figure 13.2 - State Diagram of Account Creator (Signup)

Account Creator State Diagram:

- States {Confirmation, Email Notifier, Sign up, Successfully Signed up}
- Events {Information Approval, Invalid Email/SIgnal Failure, notifies system, sends confirmation}
- Valid Transitions {Sign up → Email Notifier, Email Notifier → Email Notifier, Email Notifier → Confirmation → Successfully Signed up}

In this test case, the Account Creator will be tested to verify if user can create a new account successfully. The test will cover the success scenario of creating an account and will test the Email Notifier as well when the creation of an account fails.

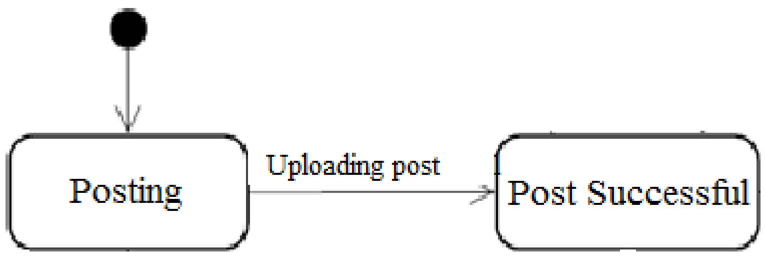


Figure 13.3 - State Diagram of Post Creator (Posting)

Post Creator State Diagram:

- States {Posting, Post Successful}

- Events {upload}
- Valid Transitions {Posing → Successfully Signed up}

In this test case, the Post Creator will be tested to verify if user can post or upload a new discussion successfully. The test will cover the success scenario of creating a post since the rate of failure is at the minimum to zero.

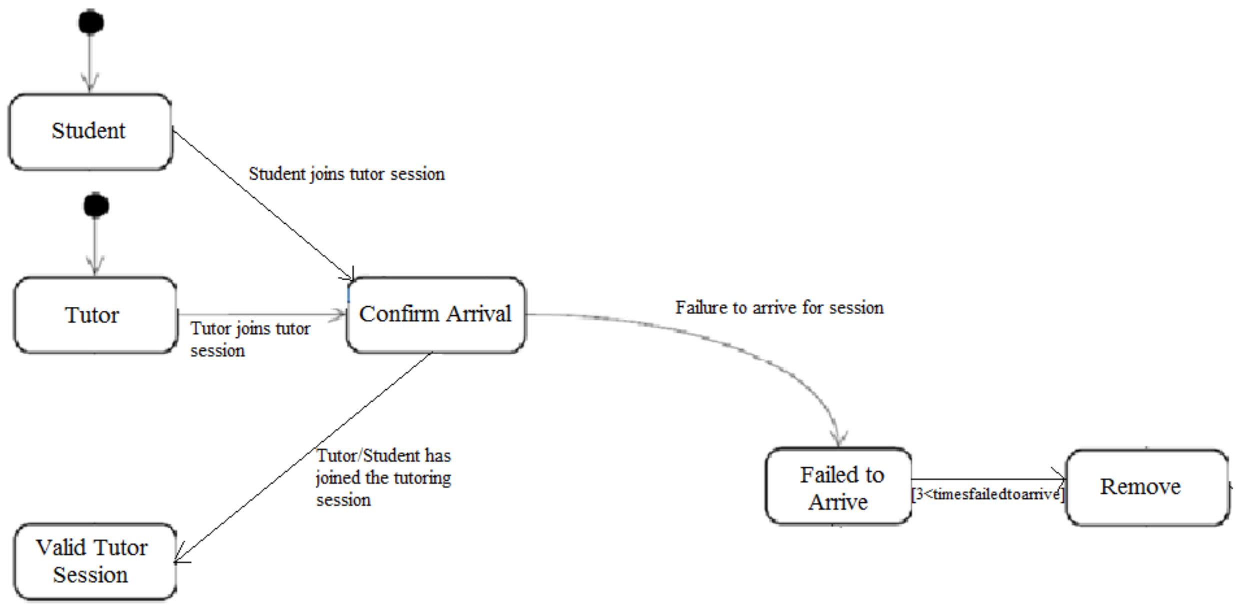

Student
Student joins tutor session
Tutor
Tutor joins tutor session
Confirm Arrival
Failure to arrive for session
Tutor/Student has joined the tutoring session
Valid Tutor Session
Failed to Arrive
[3<timesfailedtoarrive]
Remove

Figure 13.4 - State Diagram of Session Validator (Tutor Session)

Session Validator State Diagram:

- States {Student, Tutor, Confirm Arrival, Valid Tutor Session, Failed to Arrive, Remove}
- Events {joining, fail to arrive}
- Valid Transitions {Student/Tutor → Confirm Arrival, Confirm Arrival → Valid Tutor Session, Confirm Arrival → Failed to Arrive, Failed to Arrive → Remove}

In this test case, the Session Validator will be tested to verify if the approved users joined the specified tutor session successfully. The test will cover the success scenario of starting a tutor session and the fail scenario of creating a tutor session.

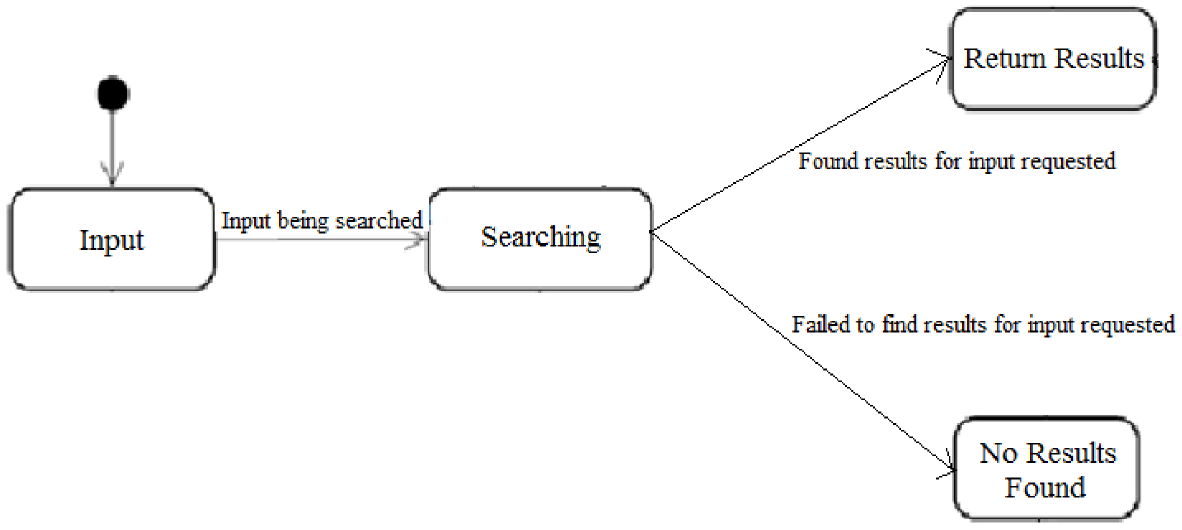

Input
Input being searched
Searching
Found results for input requested
Return Results
Failed to find results for input requested
No Results Found

Figure 13.5 - State Diagram of Searcher (Advance Search)

Searcher State Diagram:

- States {Input, Searching, Return Results, No Results Found}
- Events {input searching, failed to find results, found results}
- Valid Transitions {Input → Searching, Searching → Return Results, Searching → No Results Found}

In this test case, the Searcher will be tested to verify if the users can search keywords successfully. The test will cover the scenarios of finding search results and displays them to users.

.

13. History of Work, Current Status, and Future Work

Our group has decided that the role of project management will be split equally among our members, however Nathan del Carmen will be tasked with the responsibility to ensure that the members are completing their work on time as well as making sure that all the deadlines are met. We have decided to meet every Wednesday morning at 10:00 am at the Reading room library at SERC. There we review the work that is to be done that week, review professor and TA's emails, split the work, brainstorm about what needs to be done and how we will accomplish it and ensure that everyone is on the same page with regards to where our project is heading and the direction we want to take it. Nathan del Carmen will also be in charge of configuration management. If we believe that it is necessary to meet again, we meet on Thursday or Friday. Our group keeps in touch using the group messenger app that Facebook provides.

13.1 Merging the Contributions from Individual Team Members

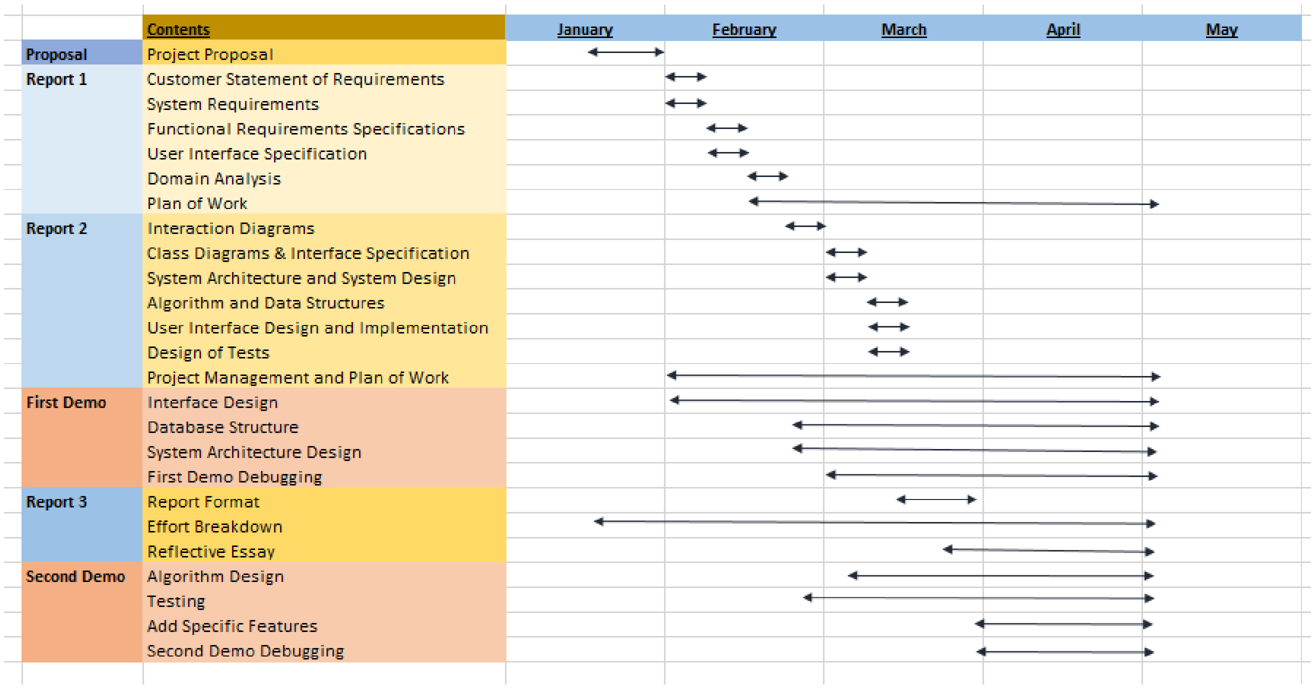
Luckily every member of our team had a time slot in which we could all meet at the same time. This allowed for excellent communication throughout the group and everyone was onboard with any changes that happened. We had decided early on that the report would be done in Times New Roman. Main topics would be done in 20 font bolded, subtopics would be 12 font bolded, and everything else is 12 font un-bolded. Our work was conducted through Google Docs so all members were able to work on the project at the same time and ensure everyone was consistent. After passing the 100 page mark on Report 1, Google Docs became unusable due to the slowness of the application. For Report 2, we created another Google Doc to store the report. Google Docs does not allow you to convert it into a PDF, thus we needed to move the report into Microsoft Word in order to convert it to a PDF. However when we copied the Google Doc into Microsoft Word, the alignment of pictures, numbers of pages, and other format issues arose. Jon Yang was in changes of correcting all these issues and held the master copy of each iteration of the report.

13.2 Project Coordination and Progress Report

Our classes and functions implemented above with the multiple systems in place will be split up between our members. Some cases which are already functional on our site are the basic forum functionalities, a simple point system, account type handling, sign up and log in. These functionalities are still bare bones, thus are implemented but not yet fully functional or finalized to a degree of user acceptance. More testing must still be done to make sure all the current acceptance tests passes to prevent any bugs/glitches from occurring. We currently plan on expanding our website by adding the basic functionalities of the tutoring section while continuing to clean up our current functionalities before the demo presentation. Our plan for the presentation is to show a clean functional basic update of all the working functionalities we currently have in place, including every features' code, database, SQL scripts, and system functions. In terms of project management activities, our responsibilities for each feature/use cases are broken down below in our "Breakdown of Responsibilities" section.

Update: The cases that are currently functional on our site are the search bar, tutor page, basic forum functionalities, a more complex point system, account type handling, sign-up and login. These functionalities are starting to become more and more like the design that we intended for it to look like. At this point, it would probably still need more testing to be done to make sure that all the previous and the current acceptance tests pass. We plan on expanding our website hopefully implementing the chat system or the documents feature while continuing to clean up our current functionalities. Our plan for the future is to show a clean functional basic update of all of the existing functionalities as well as the search bar. In terms of project management activities, our responsibilities for each feature/use cases are shown below in our "Breakdown of Responsibilities" section.

13.3 Plan of Work

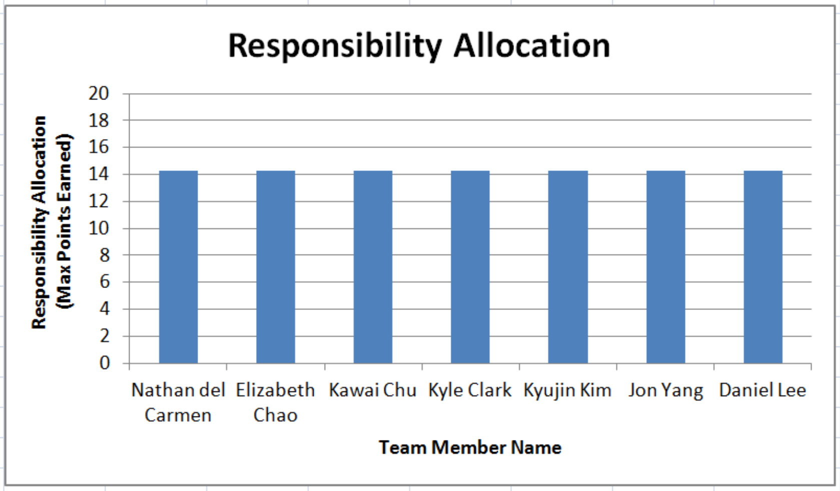


	Contents
Proposal	Project Proposal
Report 1	Customer Statement of Requirements
	System Requirements
	Functional Requirements Specifications
	User Interface Specification
	Domain Analysis
	Plan of Work
Report 2	Interaction Diagrams
	Class Diagrams & Interface Specification
	System Architecture and System Design
	Algorithm and Data Structures
	User Interface Design and Implementation
	Design of Tests
	Project Management and Plan of Work
First Demo	Interface Design
	Database Structure
	System Architecture Design
	First Demo Debugging
Report 3	Report Format
	Effort Breakdown
	Reflective Essay
Second Demo	Algorithm Design
	Testing
	Add Specific Features
	Second Demo Debugging

a. Breakdown of Responsibilities

All team members contributed equally.

Report Distrubution	Nathan del Carmen	Elizabeth Chao	Kawai Chu	Kyle Clark	Kyujin Kim	Jon Yang	Daniel Lee	Percentage
Project Management (16 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.1: Interaction Diagrams (30 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.2: Class Diagram and Interface Specification (10 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.3: System Architecture and System Design (15 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.4: Algorithms and Data Structure (4 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.5: User Interface Design and Implemention (11 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.6: Design of Tests (12 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.7: Plan of Work (2 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Sec.8: Refrences (-5 Points)	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	100%
Total Points that could be aquired				100				
Points Aquired	14.29	14.29	14.29	14.29	14.29	14.29	14.29	100.00



13.4 Breakdown of Responsibilities

Our team is divided into three sub-teams of groups two, two, and three. Each subteam is responsible for their own product (forum, tutor/point system), while all groups will collectively work on one more topic (search). Each subgroup will include necessary UML diagrams and charts. Each subgroup will meet at least one-to-two times a week, and the entire group will meet biweekly. More meetings will be scheduled if found necessary. The subgroups/entire group will meet in meetings that last 1-3 hours long. Each meeting will have a planned agenda specifying which topics to cover. General meeting discussions will start of with (1) filling in on what everyone is working on/did, then it will proceed with (2) particular issues and questions that need to be resolved, and last it will conclude with (3) future plans and what to do. Each week a different member will lead the meeting to ensure everyone is on track and everyone is doing their job.

The subgroups split into the following:

Forum: Team A (Kyujin Kim, Daniel Lee)

UC-1 - Posting in Forum → Kyujin Kim
UC-13 - View Forum → Daniel Lee
UC-14 - Saving Draft Forum Post → Daniel Lee
UC-15 - Autosave Comments → Kyujin Kim

Tutor/Point system: Team Alpha (Kawai Chu, Elizabeth Chao, Nathan DelCarmen, Jonathan Yang)

UC-2 - Signing up for Tutoring → Elizabeth Chao
UC-9 - Upvoting/Downvoting → Kawai Chu
UC-10 - Becoming a tutor → Nathan DelCarmen, Jonathan Yang
UC-11 - Point decay → Kawai Chu
UC-12 - Point reset → Elizabeth Chao
UC-16 - Update Schedule → Nathan DelCarmen
UC-17 - Check Tutoring Signups → Nathan DelCarmen

Login/Signup: Team 1 (Kyle Clark, Nathan DelCarmen)

UC-3 - Sign Up → Kyle Clark
UC-4 - Login → Kyle Clark

Search: Team A / Team Alpha / Team 1

UC-6 - Using the Search Bar → Jonathan, Nathan, Kyle

<u>Note</u>: We have decided that we will be focusing our efforts on these features, the forms, the tutor and point system, login and signup, and the search bar. We believe these are the features that are the ones that are the most vital to our website and key to its survival. Other features such as the career post, the chat, and the email were deemed unessential.

14. References:

1. *Randi Weingarten is President of The American Federation Of Teachers*
   http://blogs.edweek.org/teachers/classroom_qa_with_larry_ferlazzo/2013/02/response_how_peer_assistance_can_improve_teacher_practice.html

1. **How Bad Is the Job Market for the College Class of 2014?**
   http://www.slate.com/blogs/moneybox/2014/05/08/unemployment_and_the_class_of_2014_how_bad_is_the_job_market_for_new_college.html

1. We will use this website to find a search bar and use it in our project.
   http://www.dmoz.org/Computers/Open_Source/Software/Internet/Search_Engines/

1. We will use this website to obtain a grasp of how wikipedia works and extract ideas from it.
   http://en.wikipedia.org/wiki/List_of_wiki_software

1. This is the basic outline of what must be done in our project, and we have followed the descriptions precisely.
   http://www.ece.rutgers.edu/~marsic/Teaching/SE/report1.html