Software Requirements Specification

CS4770 Team A Project

Version 1.0 Approved

Prepared by Floris Bouman, Joshua Davis, and Bradley Gavan, Matthew Randell, and Curtis White

Team A

February 4, 2017

Table of Contents

1. Introduction **5**
1.1 Purpose 5
1.2 Document Conventions 5
1.3 Intended Audience and Reading Suggestions 6
1.4 Product Scope 6
1.5 References 7

2. Overall Description **8**
2.1 Product Perspective 8
2.2 Product Functions 8
2.3 User Classes and Characteristics 9
2.4 Operating Environment 9
2.5 Design and Implementation Constraints 9
2.6 User Documentation 10
2.7 Assumptions and Dependencies 10

3. External Interface Requirements **11**
3.1 User Interfaces 11
3.2 Hardware Interfaces 13
3.3 Software Interfaces 13
3.4 Communications Interfaces 14

4. Domain Model **14**

5. System Features (Use Cases) **14**
5.1 Account Creation 14
5.2 Login 15
5.3 Adding Friends 16
5.4 Suggested Friends 17
5.5 Posting Content 18
5.6 Content Visibility 19
5.7 Posting Permission 20
5.8 Commenting 21
5.9 Comment Replying 21
5.10 Group Creation 23
5.11 Group Joining 23
5.12 Accepting Joiners 24
5.13 Invite Permission 25

5.14 Found Items 26
5.15 Lost Items 26
5.16 Schedule 27
5.17 Résumé 28
5.18 Poll Creation 29
5.19 Answering a Poll 29
5.20 Inbox Messaging 30
5.21Logout 31

Other Nonfunctional Requirements 31
Performance Requirements 31
Safety Requirements 31
Security Requirements 31
Software Quality Attributes 32

Other Requirements 32
Appendix A: Glossary 32
CSS 32
Friend 32
Git 33
Group 33
HTML 33
Interface 33
MongoDB 33
Node.js 33
Repository 33
User 34
Appendix B: Analysis Models 34

Revision History

Name	Date	Reason For Changes	Version
Version 1.0	04/02/2017	Initial Specifications Release	1.0

1. Introduction

1.1 Purpose

Our product is a social network platform named Memorial Student Social Network, abbreviated as MUNSSN. This Software Requirements Specification (SRS) describes the full system, although some requirements and features may change as the project develops. The purpose of the MUNSNN is to provide a social media platform that will be designed for students at Memorial University. This will provide a student-oriented social platform for members to share relevant school information with each other. Students will be able to use their MUN email to create an account and become members of the platform. Each member will have a personal homepage which will include courses that the member is enrolled in, their course schedule, MUNSSN groups that they are members of, their resume, a list of other members that they are friends with, and content that the user has uploaded. The platform will support a “friends” system where members who are “friends” can interact with each other so as to be able to see each others profiles, post content on each other’s homepage, send private messages, and comment on each other’s posted content. The platform will also support “Groups” for students belonging to the same courses to share content and communicate.

This product will connect members of the Memorial student body socially as well as academically. MUNSSN will help students collaborate on their assignments.

1.2 Document Conventions

This document is typed in 11pt Arial font with subheadings in bold 14pt Times New Roman font. Main headings are in bolded 18pt Times New Roman font. Each heading is numbered by section and subsection. For example, this section is 1.2, which is the second subsection of the first section.

Requirement statements are placed in subsections of the appropriate type. For example, requirements related to visual aspects such as HTML/CSS are placed under Software Interfaces rather than given their own heading.

1.3 Intended Audience and Reading Suggestions

This document is intended for developers and project managers. This document can also be beneficial for other readers who are interested in both technical and non-technical aspects of a project of this type. This Software Requirements Specifications document contains sections on the Overall Description, External Interface, Design Model, System Features, Nonfunctional features. Each section is numbered and contains sub-sections. The header and page numbers of each section and subsection is found in the table of contents. It is suggested for developers and project managers to read the entirety of this document, in the order that it is written. This is to ensure that all details of the project are fully understood. For a brief overview of the project and it’s scope and uses, for both technical and non-technical reader, section 1.4 the “Product Scope” and section 6 “

System Requirements" would be read.

Users should access the Documentation in the Help System in order to understand how to access and use the website. A user account is not intended to have access to internal structure of features, therefore this Specification is beyond a user's needs.

1.4 Product Scope

Our product is a Memorial Student Social Network (MUNSSN). It will be a social media platform designed specifically for students at Memorial University.

- Students will be able to use their MUN email to create an account.
- Each account will have a homepage with information about the member, as well as content they post.
- The platform will support a "friends" system where members who are "friends" can interact with each other by being able to see each other's profiles, post content on each other's profiles and comment on each other's posted content.
- The platform will also support "Groups" for students belonging to the same courses to share content and communicate with one another.
- This product will connect students all across Memorial University, but more specifically connect students who are members of the same programs and courses.
- The goal in this product is for students to work together to both boost academic performance and have an increased sense of community at the university. It will be easier than ever for students to be in contact and collaborate on group work, reach out to other members of their courses and to interact with each other.
- MUNSSN will support the creation of groups; open platforms for discussion and posting content for members of specific courses.

1.5 References

- The format for this document follows the System Requirements Specification template created by Karl E. Wiegers (Copyright © 1999).
- MongoDB is a NoSQL database that will be used for storage for MUNSSN. The version we will be using is MongoDB shell 3.2.11. Reference information can be found at: http://mongodb.github.io/node-mongodb-native/2.2/quick-start/?_ga=1.199032047.1936227906.1485803242
- JavaScript 1.8.5 will be used to manage the web content. References can be found at:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

- HTML5 will be used for creating the web content for MUNSSN. Reference information can be found at the HTML5 API page at: http://html5index.org/

- CSS3 will be used to manage the style of the web pages. Reference: https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

- Node.js v4.6.1 will be used to create and run the MUNSSN server and routing functions. Reference documents for Node.js can be found here: https://nodejs.org/en/docs/

- Jasmine JS Framework v2.5.2 will be used to perform unit tests for MUNSNN Javascript modules. Reference documents can found at: https://jasmine.github.io/2.5/introduction.

2. Overall Description

2.1 Product Perspective

The product being specified in this SRS is a new, fully contained product that is meant to create a social platform specific to Memorial University students. The context of this product is from the notion of creating a social media type design that will allow students to interact with other students and share relevant information within the university. This product is something that Memorial University currently does not have and can fill what could be an important gap within the student community.

2.2 Product Functions

- Guests with MUN emails can sign up to the website.
- Members will be able to upload a profile picture.
- Members will be able to become friends with other members by sending a friend invitation.
- Members will be able to see possible friends they can add based on certain information they have entered such as courses, friends, etc…
- Each member should have their own unique timeline in which content can be posted.
- Members can set their own permissions on who can see or post on their timeline.
- Members can put comments on others posts based on privacy settings that have been set.
- Members can create groups and set permissions within it for other members.
- The software will include a lost and found functionality that will allow users to post photos of found items.
- Members will be able to initiate their own course schedule and once complete a calendar will be displayed.
- Each member can upload their resume which can be accessed by their friends.
- Members will be able to create polls and other members that are also taking the specific course will be able to vote on it.
- Members will be able to send messages to other members.

2.3 User Classes and Characteristics

The majority user class that will be using this product will be student based. Most will be using laptops or mobile devices to access the online software, and will not be power user type. Some academics like professors or teaching assistants may also use this software to expand the teaching and communication tools they have at their disposal.

2.4 Operating Environment

This software is intended to be used on web browsers, specifically Google Chrome or Mozilla Firefox. Any hardware, browser, or operating system which supports Javascript will support the software. The software will require an internet connection to function. The software will be able to be viewed on mobile devices as well with an adaptive CSS structure for proper viewing on a variety of display devices.

2.5 Design and Implementation Constraints

There are several factors that may limit the design and implementation of the software. The number one constraint for this design will be the time constraint, as this software has only a couple month window to be designed and built. Other constraints include Memorial University rules and regulations. The software must fall within the guidelines provided by Memorial University and must not violate any policies. The software must be compatible with the software packages installed on MUN's server to run within the system. User access is also constrained to active students of Memorial University as a verified MUN email is required to use the software.

2.6 User Documentation

Users can access a help page on the website before login, which explains the main processes involved in signing up and completing a profile, as well as using functions such as messaging.

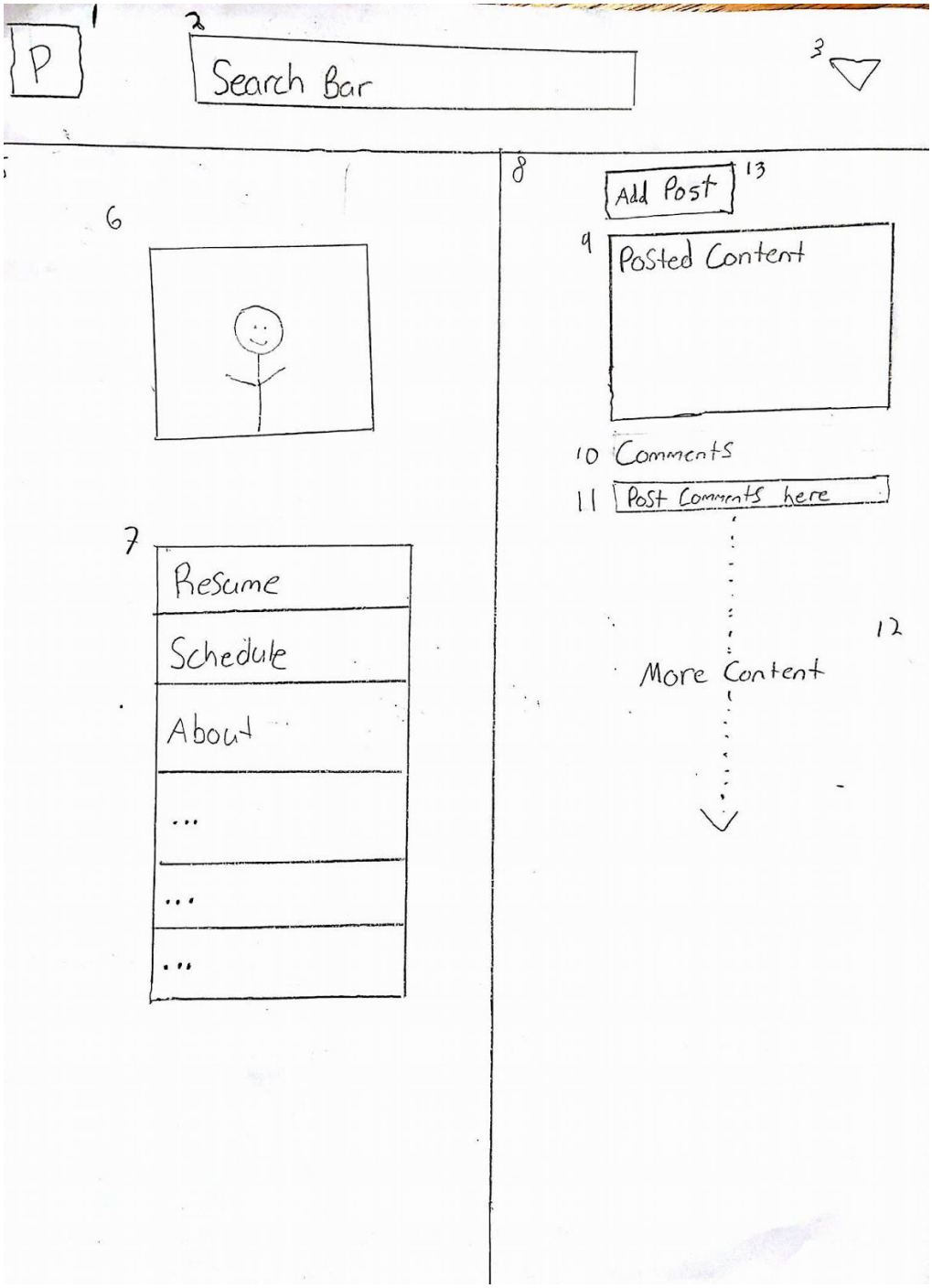
Upon a user's first login, notes will point out several features and give brief explanations of their purpose and how to use them. Once these notes are closed, they do not reappear. The user may consult the help page from that point on.

2.7 Assumptions and Dependencies

It is assumed that MUN will continue to support the given version of Node.js and MongoDB. It is also assumed that MUN will maintain its current mun.ca email format into the future. Modern browsers will be used to run this software, so it is assumed they will continue to support the use of current HTML, CSS and Javascript features.

3. External Interface Requirements

3.1 User Interfaces



1.) "Profile" button for returning to your profile from anywhere on the website.
2.) Search bar where the user can search for users/classes/groups
3.) "Options" drop down button that will drop down when clicked on and provide the user with options such as profile options/security/privacy and help.
4.) Static navigation bar, which appears on every page.1-3 are always within this bar.
5.) Left half of the page is static in sizing, and without scrolling (6-7 are always within this part).
6.) Photo for a group/individual
7.) Expanding menu containing modules (ex. Resume, Schedule, "About"). When one of these modules is pressed on it expands to take up most of the left side of the page (or as much as is needed of the left side).
8.) Right half of the page is static in position too, but it can scroll, if there is enough content to warrant scrolling (9-11 is included within this section).
9.) Each piece of content (not comments) posted will go here (another box will be created each time content is posted).
10.) Directly under each content box the comments are posted (if there are any).
11.) Under that is a box in which comments can be posted.
12.) Content will continue on down the page, from newest to oldest posted content.
13.) “Add Post” button takes user to a page where they can specify what they want to post to their wall.

3.2 Hardware Interfaces

The system takes receives data from the user through a mouse/touchpad/touchscreen, a keyboard/touchscreen and possibly a webcam too, if we decide implement a feature using that.

3.3 Software Interfaces

The social media website is implemented visually via HTML/CSS (v. 5.00 for HTML, recent to 2014, and CSS3, standardized as recently as 2014) in conjunction with Node.js (v. 7.4.0, recent to January 2017) for website structural manipulation and behaviour processing. These technologies integrate with MongoDB (v. 3.2.10, recent to September 2016) to create and store data schema for users to interact with the website.

3.4 Communications Interfaces

To connect to a network, Node.js creates and establishes a server connection which routes user actions through MongoDB for storing any data passed from HTML elements. The server will route through HTTP to a localhost for viewing in any web browser.

Version control will be handled via Github so that project files can be uploaded, downloaded, edited and viewed easily.

4. Domain Model

To be included in a future revision.

5. System Features (Use Cases)

5.1 Account Creation

5.1.1 **Name**: Guests can sign up to become a member of the social network

5.1.2 **Goal**: Guest creates an account

5.1.3 **Input**: MUN Email and password

5.1.4 **Output**: Confirmation email, new account

5.1.5 **Main Scenario**: A user who has not previously used the social network wants to create an account and join

5.1.6 **Precondition**: User must have a MUN email which hasn't previously been used for an account

5.1.7 **Steps**:

5.1.7.1 User visits homepage.

5.1.7.2 User clicks sign up button

5.1.7.3 User enters email.

5.1.7.4 User clicks next.

5.1.7.4.1 System confirms that the entered email is a valid MUN email

5.1.7.4.2 (Invalid email entered) System sends User back to previous page,and the User is prompted to enter a valid email.

5.1.7.5 User enters personal information (password twice, name, optional picture)

5.1.7.5.1 System confirms both passwords match (active while typing them)

5.1.7.6 System sends a confirmation email to the users inputted MUN email, that, will be used to activate account.

5.1.7.7 User clicks the confirmation link

5.1.7.8 (User is now a student) System prompts student(User) to select their courses.

5.1.7.9 Student(User) pressed the done button, and the signup process is complete.

5.1.8 **Postcondition**: User now has an account and can be an active member of the social

network

5.1.9 Exceptional Scenario 1

5.1.9.1 Example: You heard about this cool new social media platform, MUNSSN, for MUN students. You decide to make an account to check it out.

5.2 Login

5.2.1 Name: Login to system

5.2.2 Goal: User logs into their account

5.2.3 Input: User's email and password

5.2.4 Output: User is logged in to their account

5.2.5 Main Scenario: User enters their email and password ,if accepted user is logged in to their account

5.2.6 Precondition: User has already created an account

5.2.7 Steps:

5.2.7.1 User enters their @mun email into the username field

5.2.7.2 User enters their password in the password field

5.2.7.3 User pressed "Log In" button

5.2.7.3.1 If Username and/or password incorrect User is prompted to try again

5.2.8 Postcondition:

5.2.9 Exceptional Scenario 1

5.2.10 Example: Josh wants to check his study group for information so he logs into MUNSSN

5.3 Adding Friends

5.3.1 Name: Send "friend requests" to other members

5.3.2 Goal: To "add" friends (other members of the social network) to your "friends list"

5.3.3 Input: A member of the social network that you wish to add as a friend

5.3.4 Output: A request, which can be accepted or denied, sent to that member

5.3.5 Main Scenario: A user finds another user which they want to be friends with. They send them a friends request. The user either declines and the two

users will not become friends, or they accept and the two users are friends and can interact on the social network.

5.3.6 Precondition: There are two accounts on the social network which are not already friends

5.3.7 Steps:

5.3.7.1 User hovers over the name of another students they are not friends with.

5.3.7.1.1 A button called "Add as a Friend" drops down from their name

5.3.7.2 The User pressed that button

5.3.7.2.1 That button changes from "Add as a Friend" to "Added"

5.3.7.3 A message is sent through the system, which the added friend will receive, giving them the option to Accept/Decline/Ignore

5.3.7.4 That student presses accept, and now both students are friends.

Alternate Path:

5.3.7.4 Student receiving request presses decline

5.3.7.4.1 The student who sent the request will not get a message informing them of this, but will be able to send more friend requests.

Alternate Path:

5.3.7.4 Student receiving request presses ignore

5.3.7.4.1 The student who sent the request will not get a message informing them of this, and can't send any more friend requests.

5.3.8 Postcondition: If accepted, the two users are no friends and can interact with each other within the social network.

5.3.9 Exceptional Scenario 1

5.3.10 Example: Bradley sees Matt's profile on his friend's page. Bradley hovers Matt's name and clicks the "Add Friend" option that appears. Matt receives the friend request and hits "Add" and they are now friends. Yay

5.4 Suggested Friends

5.4.1 Name: Finding suggested friends

5.4.2 Goal: Have friend suggestions displayed so users can more easily connect

5.4.3 Input: Other members who have mutual friends, classes and interests

5.4.4 Output: A list of members who you may know irl

5.4.5 Main Scenario: You looks at the suggestion list and sees buddy there from class. Wonder if he gots the notes from class the other day, might as well send him a friend request.

5.4.6 Precondition: You have something in common with other members that are not friends

5.4.7 Steps:

5.4.7.1 Top friend suggestions are listed in a section in the side panel

5.4.7.2 User can select a “see more” option

5.4.7.3 User is taken to a page with a full list of suggested friends

5.4.8 Postcondition:

5.4.9 Exceptional Scenario 1

5.4.10 Example: You are using MUNSNN and you glance over at your the suggestions list. You notice someone from your class, you decide to add them as a friend.

5.5 Posting Content

5.5.1 Name: Posting content to a user’s own timeline

5.5.2 Goal: A user can post content to their own timeline that can be seen by others on their profile

5.5.3 Input: The content the user wishes to display

5.5.4 Output: The content on the user’s timeline for others to see

5.5.5 Main Scenario: A user has some content they’d like to share, they go to their timeline, add the content, hit submit. Other members can now view it.

5.5.6 Precondition: User has an account, as well as some content they’d like to share

5.5.7 Steps:

5.5.7.2 Users can see their personalized schedule

5.5.7.2.1 User can create/edit a personal schedule

5.5.7.3 Users can see a collapsed view of their friends list

5.5.7.3.1 Users can click a “see more” button to expand the list to completion

5.5.7.4 Users can see suggested friends in a collapsed view

5.5.7.4.1 Users can click a “see more” button to expand the list to completion

5.5.7.5 Users can scroll through all postings to the timeline, displayed in chronological order

5.5.7.5.1 Users can search for keywords in posts in their timeline

5.5.7.5.2 If no results are found return “no results” to the user

5.5.7.6 Users can post/see their resume

5.5.7.6.1 User can upload/create/edit a resume

5.5.8 Postcondition: The content is posted, others may now see it

5.5.9 Exceptional Scenario 1

5.5.10 Example: I took a picture of my dog taking a nap. The world should see it, so I post it to my timeline for all to enjoy.

5.6 Content Visibility

5.6.1 Name: Visibility of posted content

5.6.2 Goal: A user has control over who can see the content they post. Whether it be public, available to only their friends or a subset of their friends

5.6.3 Input: The setting for the visibility of a particular content

5.6.4 Output: The content is available to only those specified by the visibility setting

5.6.5 Main Scenario: A user posts some content which they don’t want certain people to see

5.6.6 Precondition: The user posts some content

5.6.7 Steps:

5.6.7.1 User clicks the “Settings” button

5.6.7.2 User is shown a list of settings which can be modified

5.6.7.3 User finds and clicks the “privacy” option

5.6.7.4 From the privacy settings, user select the “visibility” option

5.6.7.5 Given a list of options of who can see , user selects that only they can see their own posts

Alternate Path:

5.6.7.5 Given a list of options of who can see , user selects that everyone can see their published posts

Alternate Path:

5.6.7.5 Given a list of options of who can see , user selects that only their friends can see their published posts

Alternate Path:

5.6.7.5 Given a list of options of who can see , user selects that only a specified list of friends can see their published posts.

5.6.7.5.1 User keeps the existing specified list of friends that can see their published posts

5.6.7.5.1.1 User adds friends to the existing specified list of friends that can see their published posts

5.6.7.5.1.2 User removes friends from the existing specified list of friends that can see their published posts

5.6.7.5.1.3 User specifies a new subsection of their friends which can view this particular post

5.6.8 Postcondition: The content is not viewable by those specified

5.6.9 Exceptional Scenario 1

5.6.10 Example: You posted a picture of you drinking at the computer science mixer. You only want your close buddies to be able to see the picture so you limit the visibility of the post.

5.7 Posting Permission

5.7.1 Name: who can post on my timeline

5.7.2 Goal: Members should be able to define who can post on their timeline

5.7.3 Input: Names of people who cannot post on your timeline

5.7.4 Output: Users specified cannot post to your timeline

5.7.5 Main Scenario: You don't want certain people posting on your timeline

5.7.6 Precondition: You have friends, you want some of them posting to your timeline

5.7.7 Steps:

5.7.7.1 User navigates to their personal timeline.

5.7.7.2 User clicks the privacy setting option (drop down menu)

5.7.7.3 User picks the option they want.

5.7.7.3.1 Everyone that is a member can post on the timeline

Alternate Path:

5.7.7.3.1 Only friends have the ability to post on the timeline

Alternate Path:

5.7.7.3.1 Only the user will have the ability to post on their timeline.

5.7.8 Postcondition: Now the specified users can't post to your timeline

5.7.9 Exceptional Scenario 1

5.7.10 Example: Gary keeps posting spam on your profile. He won't stop so you block him from posting on your timeline.

5.8 Commenting

5.8.1 Name: Members can comment on each others' posts

5.8.2 Goal: Allow members to leave comments on each others' posts

5.8.3 Input: A comment written by a user to a specific post

5.8.4 Output: The submitted comment is now visible to other users below the post

5.8.5 Main Scenario: A user sees a post and wished to write a comment about the post.
The user writes on a comment in the comment field, submits it, and it is now viewable by other members

5.8.6 Precondition: There is an existing post that the user can see

5.8.7 Steps:

5.8.7.1 User is looking at a post

5.8.7.2 Clicking on the comments section post, the user sees the most recent comments

5.8.7.2.1 The user hits expand to see expanded comments section to display all of the comments

5.8.7.3 The user clicks the empty text area at the bottom of the page and can start to write a comment

5.8.7.4 The user writes a comment

5.8.7.4.1 The user no longer wishes to write a comment

5.8.7.5 The user hits the “post” button (or presses ‘enter’) in the text area to post the comment

5.8.7.5.1 The user deletes what they have written or either navigates away from the comment area and the comment is not posted.

5.8.7.6 Users that can see the original post can now see the comment added to the comment section of the post

5.8.8 Postcondition: The submitted comment is view by other users beneath the post

5.8.9 Exceptional Scenario 1

5.8.10 Example: Jimmy posts a dank Arthur meme and Bob thinks that it is clever in it’s satire of post-modernism in today’s society. Bob comments “lulz” for everyone to see, below the dank meme.

5.9 Comment Replying

5.9.1 Name: replying to a comment

5.9.2 Goal: respond to a comment so other users can see

5.9.3 Input: a reply written by the user

5.9.4 Output: the reply is now visible under the comment it is referring too

5.9.5 Main Scenario: A user sees a comment and wished to write a reply about the post. The user writes a reply in the comment field, submits it, and it is now viewable by other members

5.9.6 Precondition: There is an existing comment that the user can see

5.9.7 Steps:

5.9.7.1 User sees an existing comment on a post and wants to reply

5.9.7.2 User clicks the “reply” button beside the comment that they wish to respond to

5.9.7.3 A text box appears below and to the right of the original comment for the user to write their response (Note: A reply will be directly below the original comment and indented to the right. This applies recursively).

5.9.7.4 The user writes a comment

5.9.7.4.1 User changes their mind, they do not write a comment

5.9.7.5 User hits the “post” button (or presses ‘enter’) in the text area to post the reply comment

5.9.7.5.1 The user deletes what they have written or either navigates away from the comment area and the reply comment is not posted.

5.9.7.6 Users that can see the original post can now see the reply comment added to the comment section of the post

5.9.8 Postcondition: The submitted reply is view by other users beneath the post

5.9.9 Exceptional Scenario 1

5.9.10 Example: Brad is being abused by feminists for a post about his opinion on politics. Matt sees the comments and decided to back his friend up by replying and agreeing.

5.10 Comment Editing

5.10.1 Name: editing a comment the user has posted

5.10.2 Goal: edit a comment so other users can see the updated content

5.10.3 Input: updates to be made by the user

5.10.4 Output: the edited comment is now visible instead of the previous comment

5.10.5 Main Scenario: A user comments and wishes to edit that comment. The user makes the desired changes in the comment field, submits it, and it is now viewable by other members

5.10.6 Precondition: There is an existing comment that the user has posted

5.10.7 Steps:

5.10.7.1 The user sees a comment that has an “edited” tag next to it.

5.10.7.2 The user clicks on the edited tag

5.10.7.3 All previous edits of the comment are shown in the order that they were posted

5.10.7.4 User wants to edit a comment that they posted

5.10.7.5 User clicks the “edit” button on their comment

5.10.7.6 Their existing comment becomes editable

5.10.7.7 User edits their comment

5.10.7.7.1 The user does not edit their comment

5.10.7.7.2 No changes are made

5.10.8 Postcondition: The edited comment can now be viewed

5.10.9 Exceptional Scenario 1

5.10.10 Example: Brad is being abused by feminists for a post about his opinion on politics. Matt sees the comments and decided to back his friend up by replying and agreeing but makes a grammatical error. He edits his comment quickly to avoid the onslaught of corrections

5.11 Group Creation

5.11.1 Name: Group Creation

5.11.2 Goal: A user can create a Study Group, to which other users can be invited and share content

5.11.3 Input: A group name, description and picture

5.11.4 Output: A page, similar to a timeline, but for a specified topic instead of user.

5.11.5 Main Scenario: A user wants to create a group with members of their class to discuss the material, assignments, etc for that class

5.11.6 Precondition: There is a user to create the group, a group by the given name does not already exist.

5.11.7 Steps:

5.11.7.1 User goes to Groups page.

5.11.7.2 Selects “Create Group”

5.11.7.3 Form appears. User enters name and purpose of group.

5.11.7.4 User selects whether group is public or private:

5.11.7.4.1 If private, user may set a condition on who can try to join.

5.11.7.4.2 If public, any user may attempt to join(groups are public by default).

5.11.7.5 User may begin sending invites and wait for users to join.

5.11.8 Postcondition: The study group now exists. Members can navigate to the group page to view posts, add posts, comment on existing posts and invite new members to join.

5.11.9 Exceptional Scenario 1

5.11.10 Example: A team from CS4770 creates a study group page to discuss which members are in charge of tasks, ask each other questions, share progress.

5.12 Group Joining

5.12.1 Name: Join a group

5.12.2 Goal: To join an existing study group

5.12.3 Input: An existing study group

5.12.4 Output: Joining the group and being a member

5.12.5 Main Scenario: A study group exists for a class a user is taking, the user finds this group and wishes to join

5.12.6 Precondition: The group exists and the user is not already a member

5.12.7 Steps:

5.12.7.1 User goes to Groups Page.

5.12.7.2 User searches for group in search form by:

-Name

-Class

-Major

5.12.7.2.1 If successful, search results appear.

5.12.7.2.2 If required fields are missing, error message appears. User must try

again.

5.12.7.2.3 If no results can be found, error message appears and user must modify search.

5.12.7.2.4 Once results have appeared:

5.12.7.2.5 User may select a desired group

5.12.7.2.5.1 Goes to group page which states important information. User selects “Join Group”

5.12.7.2.5.2 If group is public, user is added.

Alternate Path:

5.12.7.3.1.2 If group is private, user must be accepted by creator or administrator of group.

5.12.7.2.6 User may leave page without completing join process.

5.12.8 Postcondition: The user is now a member and can now view and post content to the group page

5.12.9 Exceptional Scenario 1

5.12.10 Example: Francine is in CS1000 and has some questions about the assignment. She searches and find a study group for CS1000 already exists and joins it. She can now interact with members of her class to discuss the assignment.

5.13 Accepting Joiners

5.13.1 Name: Accepting Joining members

5.13.2 Goal: Users who requested to join a group are accepted or denied by the group Administrator

5.13.3 Input: Pending user requests to join a group

5.13.4 Output: pending requests are accepted or denied

5.13.5 Main Scenario: The system admin sees who has requested to join the group they control. if the request is accepted, the user is added to the group. If

the request is denied the pending request is deleted

5.13.6 Precondition: There must be at least one request to make a decision on

5.13.7 Steps:

5.13.7.1 User who created a group goes to group page and can see a list of applications to join a group for private groups.

5.13.7.2 User scrolls through the list and may select a user and either:

5.13.7.2.1 Accept user into group

5.13.7.2.2 Decline application

5.13.7.3 After each acceptance or denial, users are removed from list.

5.13.7.4 User may keep going until list is cleared.

5.13.8 Postcondition:

5.13.9 Exceptional Scenario 1

5.13.10 **Example:** Harry want to join the CS society page so he sends a join request to the group. Curtis, the page admin, knows Harry and accepts his request to join

5.14 Invite Permission

5.14.1 Name: Granting Invite Permission

5.14.2 Goal: A user is given permission to invite others to a group

5.14.3 Input: A user who is in a group without permission to invite

5.14.4 Output: a user who now has invite privilege

5.14.5 Main Scenario: The admin of a group wishes to allow a user in the group to invite other people to join

5.14.6 Precondition: At least one user in the group that does not have invite privilege

5.14.7 Steps:

5.14.7.1 Creator goes to group page and selects “Members.”

5.14.7.2 List of members appears and Creator selects a button next to member name that states “Grant Invite Privilege.”

5.14.7.3 Button changes to “Revoke Invite Privileges.”

5.14.7.4 User gains invite privileges and may now invite others into the group.

5.14.7.4.1 Creator may revoke these privileges by returning to list and selecting "Revoke Invite Privileges."

5.14.8 Postcondition:

5.14.9 Exceptional Scenario 1

5.14.10 **Example:** Floris has created a group for his class to share information but doesn't know everyone's name, he gives the few people he does know invite privilege so they can invite more people

5.15 Found Items

5.15.1 Name: Lost and Found post

5.15.2 Goal: To upload a photo and description of a lost item that a user found to the Lost and Found page, as well as its approximate location.

5.15.3 Input: A photo and a description

5.15.4 Output: The photo and description being posted to the top of the Lost and Found page timeline

5.15.5 Main Scenario: A user finds a lost item on campus. They want the owner to find it so they post a picture of the item along with a short description and the approximate location of where the item was found

5.15.6 Precondition: A user finds a lost item and wants to return it to its owner

5.15.7 Steps:

5.15.7.1 User uploads picture to lost and found time line

5.15.7.2 User adds description to picture

5.15.7.3 User adds an approximate location

5.15.7.3.1 Location is attached to a mini map which is attached to the main post

5.15.7.4 Post is uploaded to the found section of the page

5.15.8 Postcondition: The photo and description being posted to the top of the Lost and Found page timeline

5.15.9 Exceptional Scenario 1

5.15.10 **Example:** A user finds a phone on campus. They don't know who it belongs to and wants to find the owner so they post a picture of the phone, write a short description and post the approximate location it was found.

5.16 Lost Items

5.16.1 Name: Finding your lost stuff

5.16.2 Goal: Find your stuff on the lost and found page and get in contact with the user who posted it

5.16.3 Input: The lost item found on the Lost and Found page

5.16.4 Output: The contact info of the poster of the lost item (phone number if friend, email if not).

5.16.5 Main Scenario: A user loses an item, checks the lost and found page and finds it. Clicking on it, they get the posters contact information.

5.16.6 Precondition: A user finds an item of theirs on the Lost and Found page

5.16.7 Steps:

5.16.7.1 Lost items appear in a list sorted by newest post by default

5.16.7.1.1 Items can be sorted by location

5.16.7.2 Users can click on the item to see an enlarged picture as well as the posters contact information

5.16.7.2.1 If friends and has permission poster's phone number is shown

5.16.7.2.2 If not friends or permission denied display poster's email

5.16.8 Postcondition: The user now has the contact information for the person that posted their lost item.

5.16.9 Exceptional Scenario 1

5.16.10 **Example:** A user finds a phone on campus. They don't know who it belongs to and wants to find the owner so they post a picture of the phone,

write a short description and post the approximate location it was found.

5.17 Schedule

5.17.1 Name: Member's schedule section

5.17.2 Goal: To create a calendar, viewable by the user and their friends, of their current class schedule

5.17.3 Input: Times/Days that the user has class

5.17.4 Output: A calendar showing the user's class schedule

5.17.5 Main Scenario: The user has their class schedule and wants to upload it for them and their friends to see

5.17.6 Precondition: User has an account and the schedule for their class

5.17.7 Steps:

5.17.7.1 User selects the schedule option

5.17.7.2 User is taken to the create/edit page

5.17.7.2.1 if the user already has a schedule posted it is displayed,if not a blank table is displayed

5.17.7.3 User enters the dates of the semester, the course name, times when the class is held, and the campus it takes place at

5.17.7.4 User is prompted to save changes before exiting

5.17.7.5 Most recent saved Schedule is now available to Users friends

5.17.8 Postcondition: The user's schedule is posted on their timeline in a calendar view. The user's friend can see it.

5.17.9 Exceptional Scenario 1

5.17.10 Example: The user has class at 9, 10, 2 on Monday, Wednesday and Friday. The user enters that information and a calendar view of those dates/times is posted to the users profile.

5.18 Résumé

5.18.1 Name: Resume the upload

5.18.2 Goal: User's can upload their resume and have it be viewable by their friends

5.18.3 Input: The user's resume

5.18.4 Output: A section on the user's profile where other members can see their resume

5.18.5 Main Scenario: A user wants to upload their resume for their friends to see

5.18.6 Precondition: The user has an account and also has a resume

5.18.7 Steps:

5.18.7.1 User selects the resume option

5.18.7.2 User is taken to the option page and presented with the options

-create/edit resume

-upload resume

5.18.7.3 User selects the create/edit option

5.18.7.3.1 if the user already has a resume posted it is displayed,if not a blank document is displayed

5.18.7.4 User types desired changes

5.18.7.5 User is prompted to save changes before exiting

5.18.7.6 Most recent saved resume is now available to Users friends

Alternate Path:

5.18.7.3 User selects upload option

5.18.7.4 User's file manager is opened and is prompted to upload the resume file they wish to post

5.18.7.5 If upload is successful redirect user to create/edit page and open the uploaded file

5.18.7.5.1 if unsuccessful prompt user to try again or use a different file

5.18.7.6 User is prompted to save changes before exiting

5.18.7.7 Most recent saved resume is now available to Users friends

5.18.8 Postcondition: The user has a resume section on their profile where their friend's can

see it

5.18.9 Exceptional Scenario 1

5.18.10 Example: John's on EI gettin' top stamps the taxes on cases of Blue Star just went up. John uploads his resume to show his friends his qualifications so that he can look for some cash jobs.

5.19 Poll Creation

5.19.1 Name: Creating a poll

5.19.2 Goal: User's should be able to create a poll about a course. Only those friends who have the same course as the poll poster should be able to see the pool and respond to it.

5.19.3 Input: A poll with at least 2 options

5.19.4 Output: Member's who are in the same course as the poll creator can cast a vote for the poll

5.19.5 Main Scenario: A member is curious about the opinion of a other members in their class about something in the class. They create a poll to see what their classmates think.

5.19.6 Precondition: The member is in a course

5.19.7 Steps:

5.19.7.1 User clicks "Create Poll" button on group page.

5.19.7.2 Form appears. User enters name, question, and choices.

5.19.7.2.1.1 If User needs more choices, "add" button adds another field to enter. User may add up to 10 fields.

5.19.7.2.1.2 If 10 fields have been added, "add" button disappears.

5.19.7.3 User clicks "Done" button and poll is then saved.

5.19.7.4 User's poll count is incremented. If a User reaches a max number, they cannot create any more polls.

5.19.7.5 User who created poll can edit poll by clicking "Edit" button (Repeats A.2)

5.19.7.6 Poll disappears after set amount of time or after User selects "Delete Poll." Count is decremented.

5.19.8 Postcondition: The poll is created and other members of the course are able to cast their vote

5.19.9 Exceptional Scenario 1

5.19.10 Example: Jimmy is wondering if anyone else in his course has any idea of what is going on in class, so Jimmy makes a poll. No one else knows what's going on either.

5.20 Answering a Poll

5.20.1 Name: Voting on a poll

5.20.2 Goal: For a member to cast their vote on a poll

5.20.3 Input: A vote for one of the options in the poll

5.20.4 Output: The poll's vote count increases by 1 for the option the member voted for

5.20.5 Main Scenario: A member creates a poll for a class. Another member of the class sees the poll and casts their vote. The total scores of the options reflect the new vote.

5.20.6 Precondition: A poll has been created for a course that the member is a member of

5.20.7 Steps:

5.20.7.1 User who belongs to group or class may vote in an open poll which they can see on group page.

5.20.7.2 User selects radio button for choice.

5.20.7.3 User clicks "Vote" button and vote is then saved.

5.20.7.4 Poll becomes closed to User once vote is cast. They cannot vote again.

5.20.8 Postcondition: The total scores of the poll options reflect the new vote.

5.20.9 Exceptional Scenario 1

5.20.10 Example: Jimmy created a poll asking if anyone else in his course has any idea of what is going on in class. Peggy has no idea either so she votes "no".

5.21 Inbox Messaging

5.21.1 Name: Sending private messages

5.21.2 Goal: Members can send private messages to their friends mailbox

5.21.3 Input: A written message

5.21.4 Output: The message appears in the recipient's mailbox

5.21.5 Main Scenario: A member wants to send a message to a friend. It is a private message so they do not post it on their timeline, but instead send it to their mailbox

5.21.6 Precondition: The two users are friends

5.21.7 Steps:

5.21.7.1 User selects the mailbox tab

5.21.7.2 User is taken to their personal mailbox ,displaying all current messages

5.21.7.3 User selects a friend to send a message to

5.21.7.4 User types desired message

5.21.7.5 User clicks send button

5.21.7.5.1 User can press the draft button to save the message but not send it

5.21.7.6 User's selected friend receives the message in their mailbox

5.21.8 Postcondition: The recipient has the sent message in their mailbox

5.21.9 Exceptional Scenario 1

5.21.10 Example: Francine wants to ask Annie about the assignment in one of her courses. She sends her a private message. Annie reads it in her mailbox and sends a private message in response to Francine's mailbox.

5.22 Logout

5.22.1 Name: Logging out

5.22.2 Goal: User is logged out of the system

5.22.3 Input: logout button is pressed

5.22.4 Output: User is logged out

5.22.5 Main Scenario: User presses Logout and is logged out of the System

5.22.6 Precondition: User is logged in

5.22.7 Steps:

5.22.7.1 User presses “options” drop down bar

5.22.7.2 User Selects “Log Out”

5.22.7.3 User is logged out of the system

5.22.8 Postcondition:

5.22.9 Exceptional Scenario 1

5.22.10 **Example:** John is finished looking at MUNSSN so he logs out.

6. Other Nonfunctional Requirements

6.1 Performance Requirements

1. Dynamic Loading: Loads partial amounts of data to increase speed performance
2. Data Compartmentalization: data is displayed in smaller compartments which can be expanded.

6.2 Safety Requirements

1. Recoverability: System could be backed-up for easy recovery should a major issue arise.
2. Multiple Save States: System should have a “clean slate” back-up to completely reset the data as well as a constantly updating save state for daily use and fixes.

6.3 Security Requirements

1. Privacy: Outside users should not see information about other users which those users do not want others to see.
2. Data Integrity: The database should store all user data accurately and efficiently.
3. Security: Only access to individual account.
4. Liability: make sure that Users accept the terms and conditions before using this product.

6.4 Software Quality Attributes

1. Serviceability: Should be able to be easy debugged and patched.
2. Availability: Server should have minimal downtime.
3. Usability : Should be arranged in an intuitive way that is easy to understand for all users.
4. Accessibility: Users can easily personalize Timelines and other settings.
5. Extensibility: Functions should be largely independent (Low coupling) so that new features can be easily added without breaking the system.
6. Testability: Each feature should be easily tested for correct functionality.

7. Other Requirements

1. Age Permission: Make sure the person is of legal age in their country to sign up for this type of product.
2. Verified Access: The User MUST hold an @mun email account and have a student number.

Appendix A: Glossary

The following terms are used throughout this document and may have unclear meanings or meanings which are specific to this document. They are presented here in alphabetical order.

CSS

- Cascading Style Sheet. A CSS file provides the visual aspects such as colors, layout positioning and sizing constraints for a website.

Friend

- A user who is added to the friendslist of another user. Each user can keep track of their friends by viewing their profiles, adding posts to friends' profiles and and sending messages to their friends.

Git

- A system for version control. System files may be uploaded and downloaded from a Git repository to keep track of updates to product features.

Group

- A collection of users who are connected by a common interest or purpose. Groups have their own pages where users who belong to that group can view news and posts from other group members. Users can also post news and polls on a group page if they belong to that group.

HTML

- A language by which websites are structured. It provides a skeleton for the visual aspects of a website.

Interface

- The components of the product which allow modules to communicate with each other. For example, the login module allows the user to communicate with the database and gain access to their account.

MongoDB

- A database module which allows the creation and storage of user and system data to be accessed during account creation and login.

Node.js

- A framework of Javascript components which provides developers with access to black-box functionality for web applications.

Repository

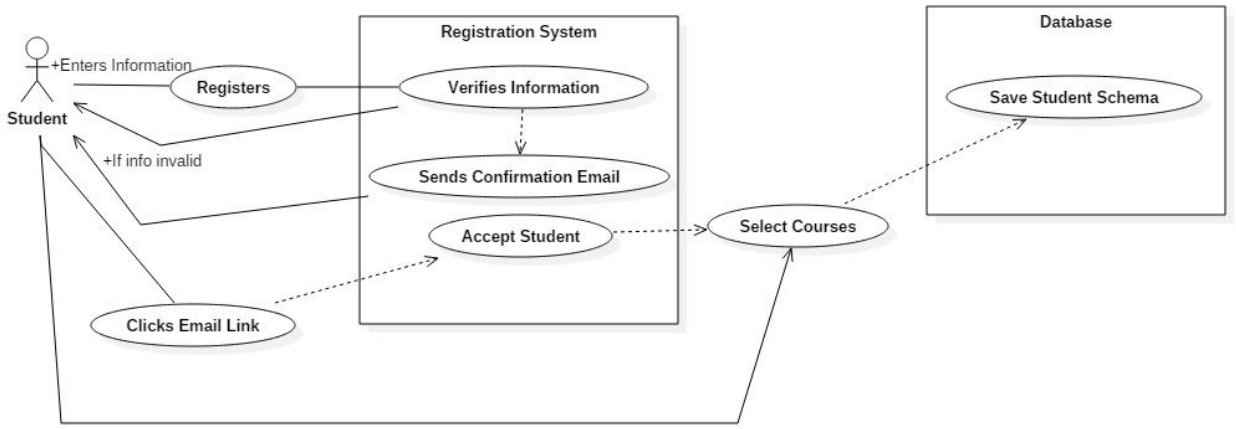
- A segment of Git where product files are stored. Each project may have it's own repository. Only users with permission to access a repository (typically developers) may access and edit the files.

User

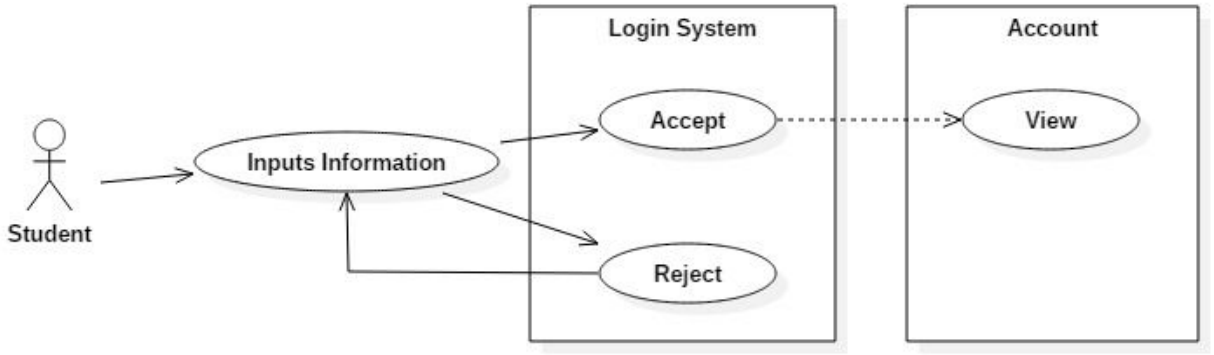
- A person who has access to the product through an account. The user is typically an end consumer who has no role in the development process.

Appendix B: Analysis Models

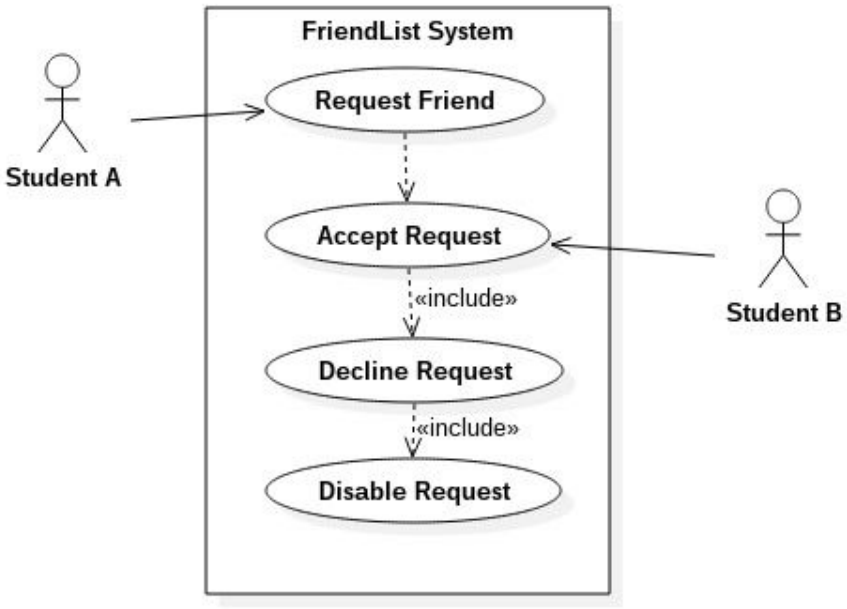
1. Account Creation (Use Case 1)



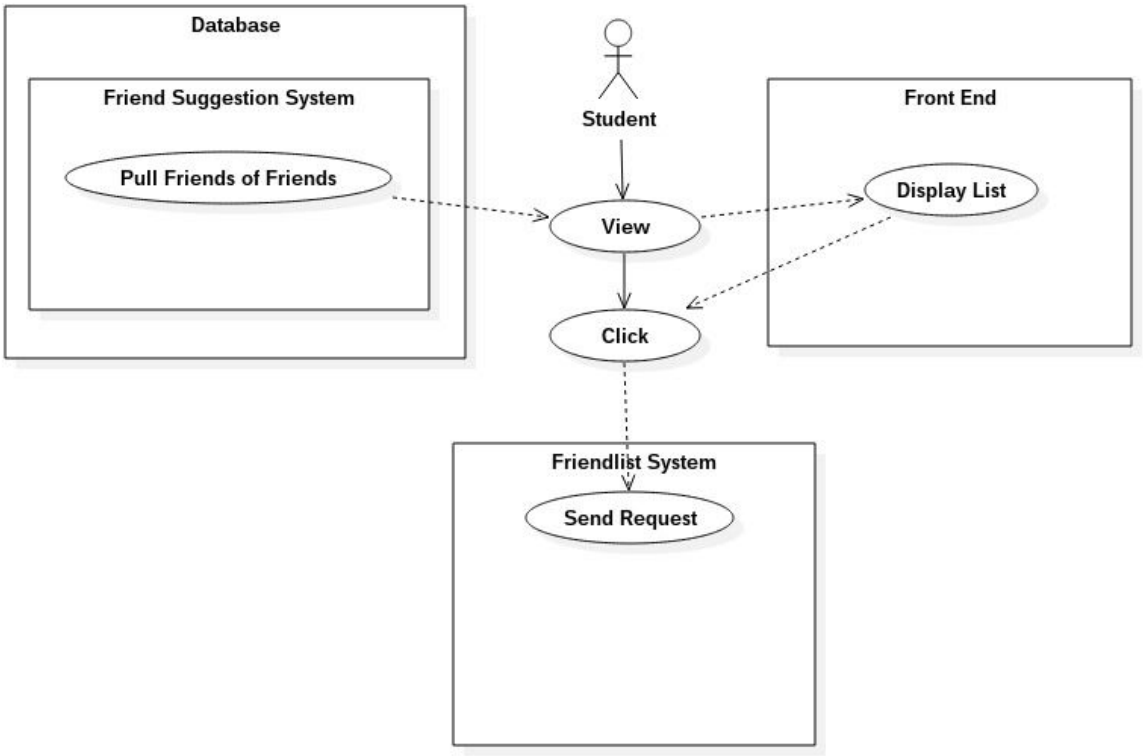
2. Log In (Use Case 2)



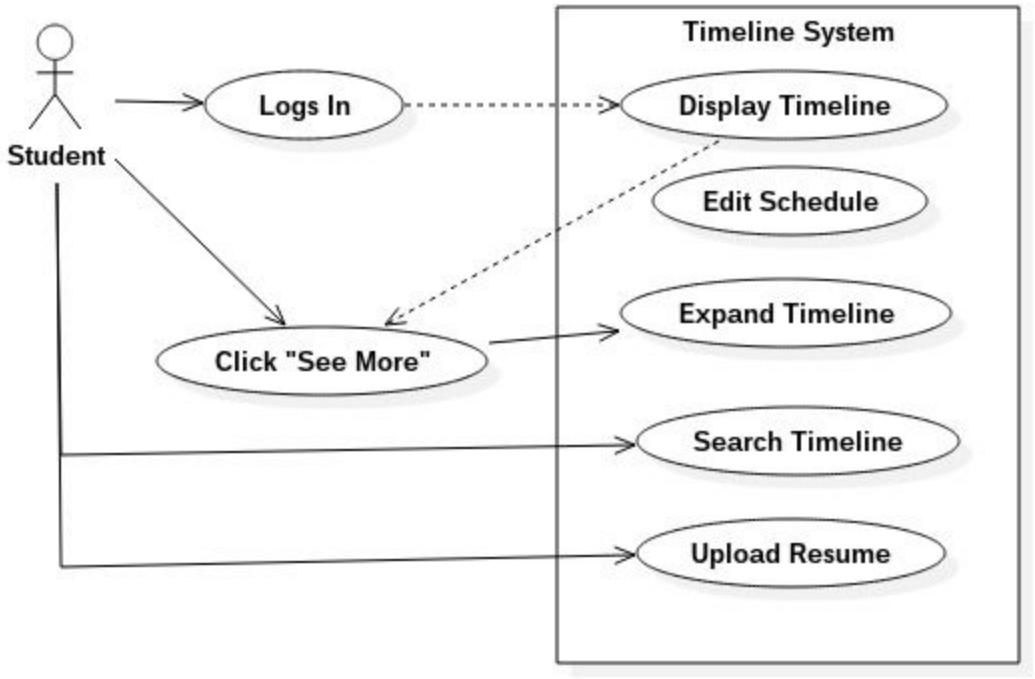
3. Friend List (Use Case 3)



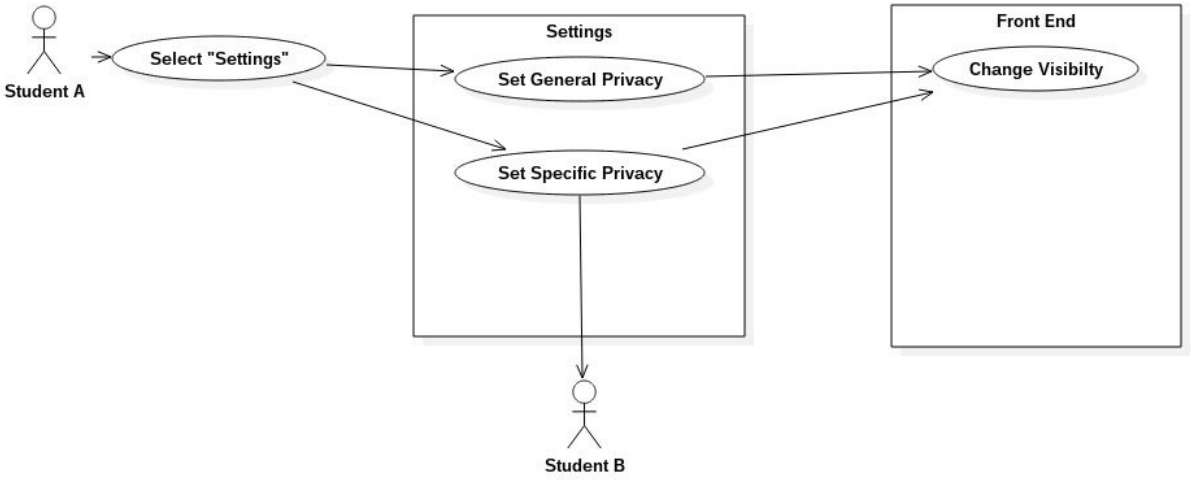
4. Suggested Friends (Use Case 4)



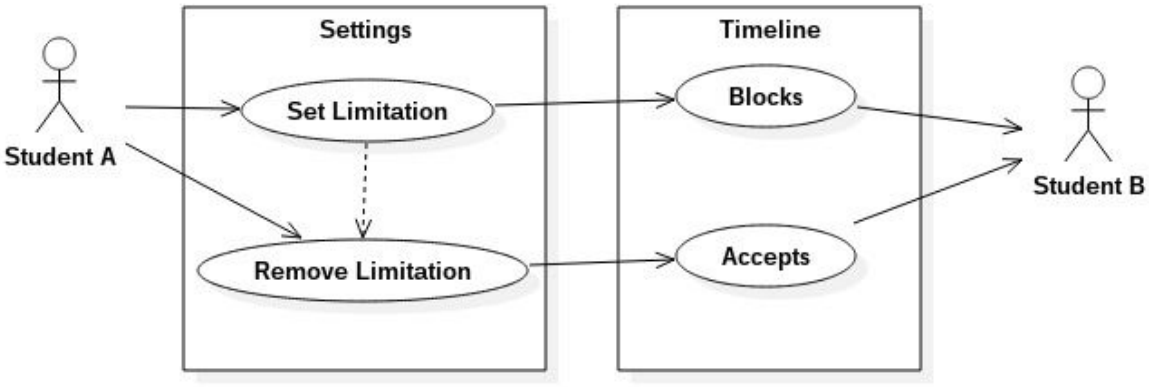
5. Timeline (Use Case 5)



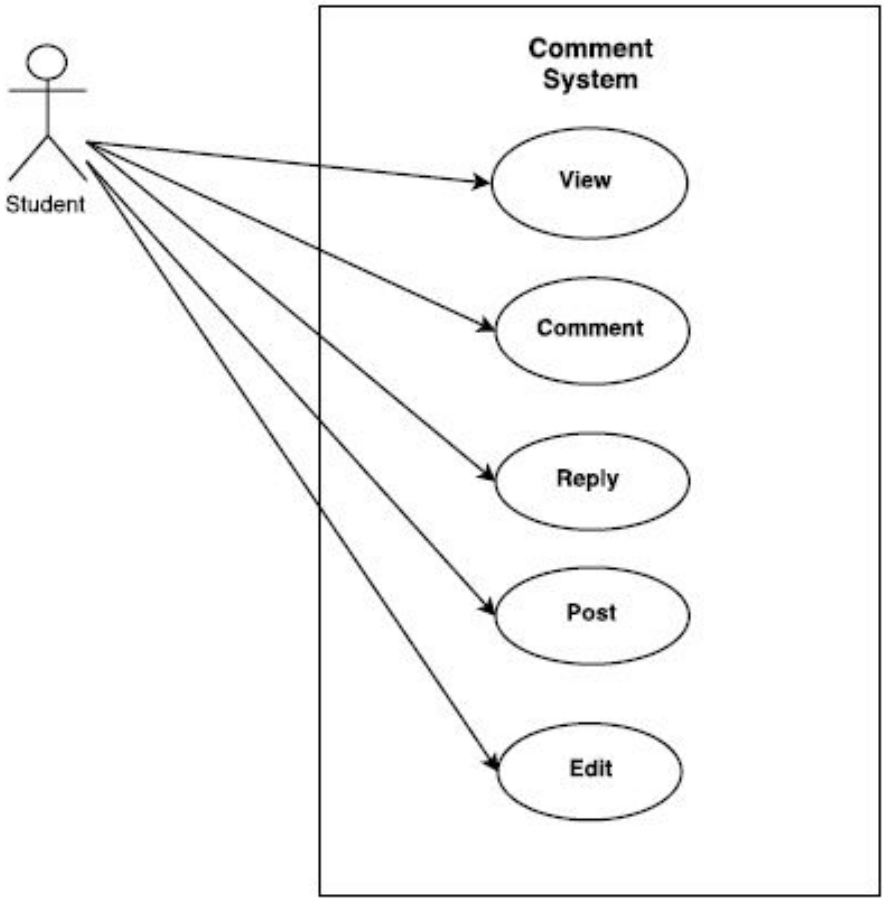
6. Privacy (Use Case 6)



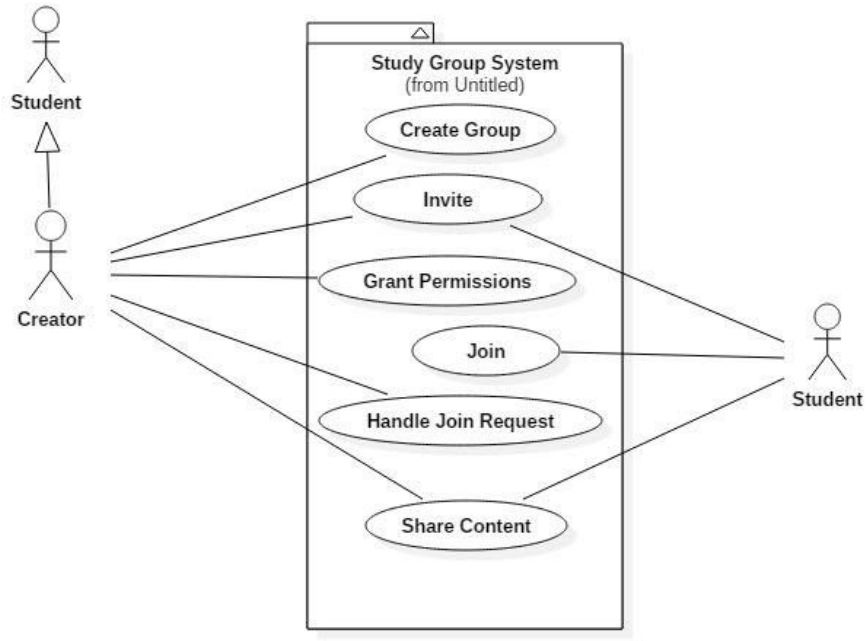
7. Post Restriction (Use Case 7)



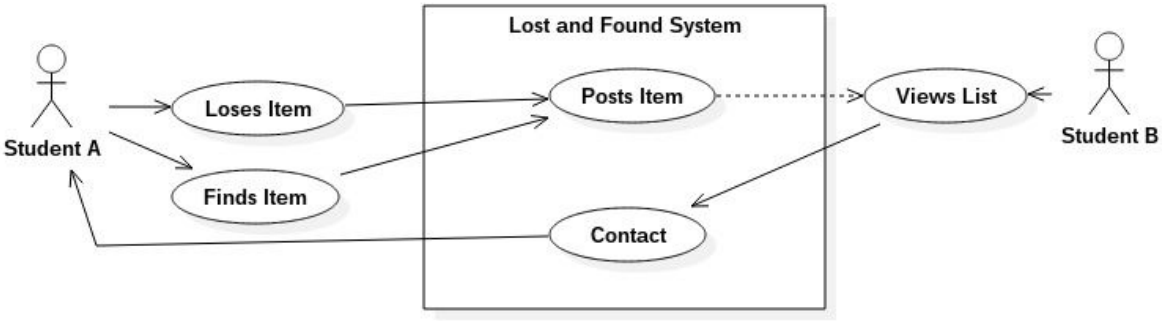
8. Comments (Use Cases 8, 9, 10)



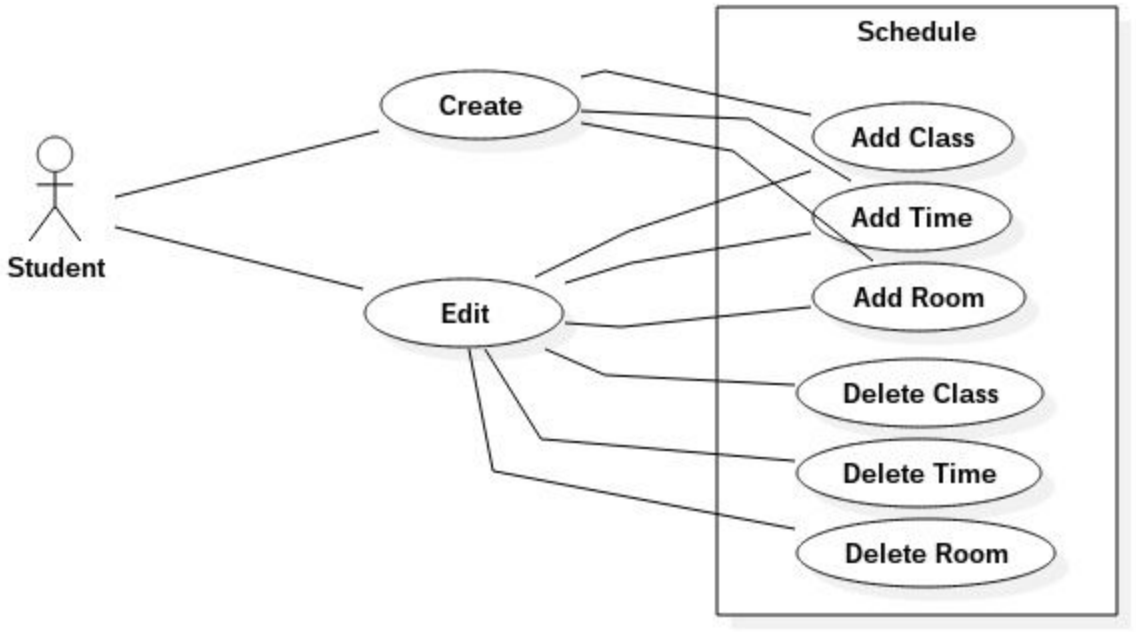
9. Study Groups (Use Cases 11, 12, 13, 14)



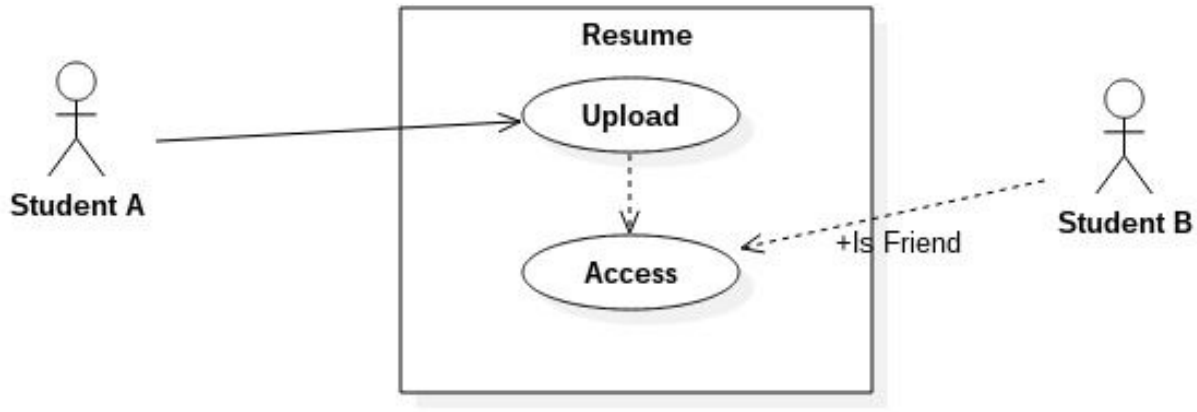
10. Lost and Found (Use Cases 15, 16)



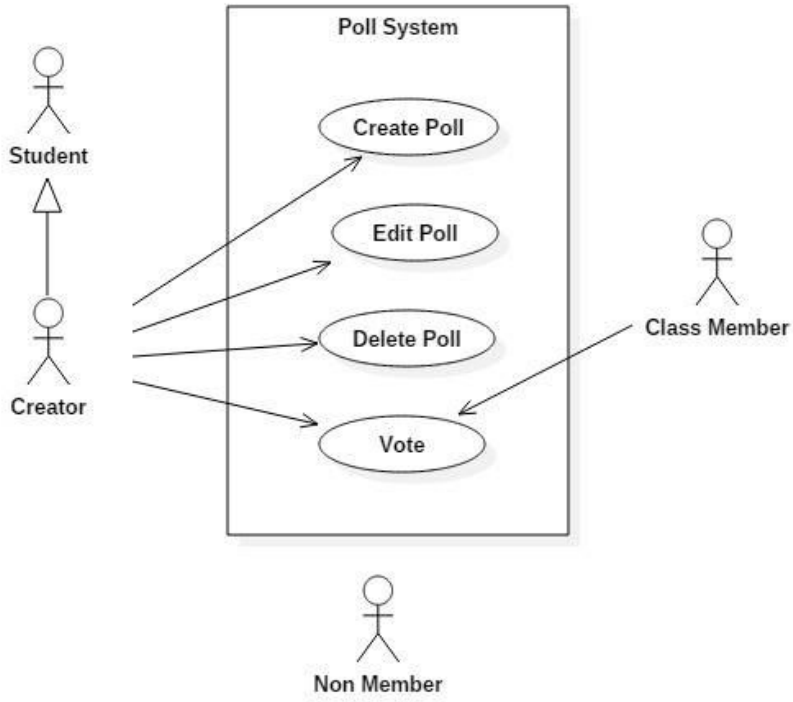
11. Schedule (Use Case 17)



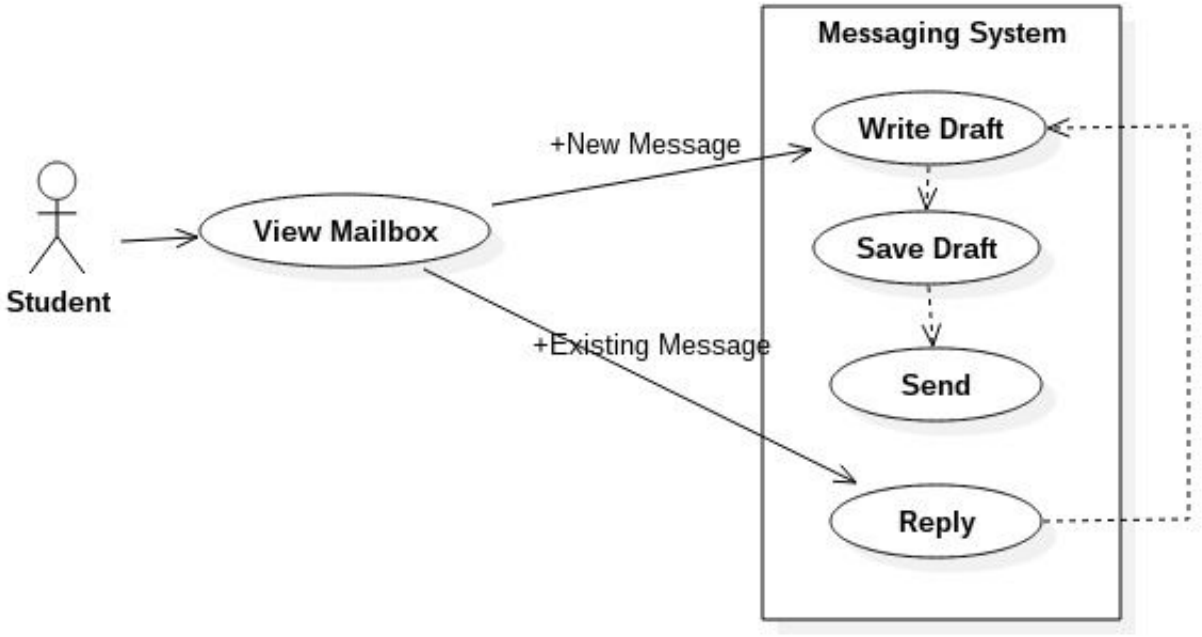
12. Résumé (Use Case 18)



13. Poll System (Use Cases 19, 20)



14. Messaging (Use Case 21)



15. Logout (Use Case 22)