
Memorial University of Newfoundland Social Network(MUNSN)

Prepared By: Group D

Tyler Vey
Saahil Budhrani
Mark Hewitt
Allan Collins

Software Requirements Specification

Version: 1.0
Date: 02/05/2017

Table of Contents

| | |
|-----------------------------------|----------|
| 1.0 Introduction | 4 |
| 1.1 Purpose | 4 |
| 1.2 Product Scope | 4 |
| 1.3 Overview | 4 |
| 1.4 Definitions | 5 |
| 1.5 References | 6 |
| 2.0 General Description | 7 |
| 2.1 Product Perspective | 7 |
| 2.2 Product Functions | 7 |
| 2.3 User Characteristics | 8 |
| 2.4 Design Constraints | 8 |
| 2.5 Assumptions and Dependencies | 8 |
| 3.0 Specific Requirements | 8 |
| 3.1 User Interface | 9 |
| 3.2 Hardware Interfaces | 10 |
| 3.3 Software Interfaces | 11 |
| 3.3.1 Interfaces | 11 |
| 3.3.2 Software Used | 11 |
| 3.3.3 Frameworks and API's Used | 11 |
| 3.4 Communication Interfaces | 12 |
| 3.4.1 Communication Protocols | 12 |
| 3.5 Functional Requirements | 12 |
| 3.5.1 Use Case: Create an Account | 13 |
| 3.5.1.1 Preconditions: | 13 |
| 3.5.1.2 Main Flow: | 13 |
| 3.5.1.3 Subflows: | 13 |
| 3.5.1.4 Alternate Flows: | 13 |
| 3.5.2 Use Case: Add Friends | 14 |
| 3.5.2.1 Preconditions: | 14 |
| 3.5.2.2 Main Flow: | 14 |
| 3.5.2.3 Subflows: | 14 |
| 3.5.2.4 Alternate Flows: | 14 |
| 3.5.3 Use Case: Suggested Friends | 14 |
| 3.5.3.1 Preconditions: | 14 |
| 3.5.3.2 Main Flow: | 14 |
| 3.5.3.3 Subflows: | 14 |
| 3.5.3.4 Alternate Flows: | 14 |
| 3.5.4 Use Case: Timeline | 14 |

| | |
|---|----|
| 3.5.4.1 Preconditions: | 14 |
| 3.5.4.2 Main Flow: | 15 |
| 3.5.4.3 Subflows: | 15 |
| 3.5.4.4 Alternate Flows: | 15 |
| 3.5.5 Use Case: Timeline Visibility | 15 |
| 3.5.5.1 Preconditions: | 15 |
| 3.5.5.2 Main Flow: | 15 |
| 3.5.5.3 Subflows: | 16 |
| 3.5.5.4 Alternate Flows: | 16 |
| 3.5.6 Use Case: Individual Timeline Item Visibility | 16 |
| 3.5.6.1 Preconditions: | 16 |
| 3.5.6.2 Main Flow: | 16 |
| 3.5.6.3 Subflows: | 16 |
| 3.5.6.4 Alternate Flows: | 16 |
| 3.5.7 Use Case: Timeline Posting Rights | 16 |
| 3.5.7.1 Preconditions: | 16 |
| 3.5.7.2 Main Flow: | 16 |
| 3.5.7.3 Subflows: | 17 |
| 3.5.7.4 Alternate Flows: | 17 |
| 3.5.8 Use Case: Posting Comments | 17 |
| 3.5.8.1 Preconditions: | 17 |
| 3.5.8.2 Main Flow: | 17 |
| 3.5.8.3 Subflows: | 17 |
| 3.5.8.4 Alternate Flows: | 17 |
| 3.5.9 Use Case: Create Study Group | 17 |
| 3.5.9.1 Preconditions: | 17 |
| 3.5.9.2 Main Flow: | 17 |
| 3.5.9.3 Subflows: | 17 |
| 3.5.9.4 Alternate Flows: | 18 |
| 3.5.10 Use Case: Lost and Found | 18 |
| 3.5.10.1 Preconditions: | 18 |
| 3.5.10.2 Main Flow: | 18 |
| 3.5.10.3 Subflows: | 18 |
| 3.5.10.4 Alternate Flows: | 18 |
| 3.5.11 Use Case: Create Schedule | 18 |
| 3.5.11.1 Preconditions: | 18 |
| 3.5.11.2 Main Flow: | 18 |
| 3.5.11.3 Subflows: | 18 |
| 3.5.11.4 Alternate Flows: | 18 |
| 3.5.12 Use Case: Upload Resume | 19 |
| 3.5.12.1 Preconditions: | 19 |
| 3.5.12.2 Main Flow: | 19 |

| | |
|--|-----------|
| 3.5.12.3 Subflows: | 19 |
| 3.5.12.4 Alternate Flows: | 19 |
| 3.5.13 Use Case: Create A Poll | 19 |
| 3.5.13.1 Preconditions: | 19 |
| 3.5.13.2 Main Flow: | 19 |
| 3.5.13.3 Subflows: | 19 |
| 3.5.13.4 Alternate Flows: | 19 |
| 3.5.14 Use Case: Communicate With Chat | 20 |
| 3.5.14.1 Preconditions: | 20 |
| 3.5.14.2 Main Flow: | 20 |
| 3.5.14.3 Subflows: | 20 |
| 3.5.14.4 Alternate Flows: | 20 |
| 3.6 Nonfunctional Requirements | 20 |
| 3.6.1 Performance | 20 |
| 3.6.2 Usability | 20 |
| 3.6.3 Security | 20 |
| 3.6.4 Scalability | 21 |
| 3.6.5 Maintainability | 21 |
| 3.6.6 Portability | 22 |
| 3.6.4 Reliability | 22 |
| 3.7 Traceability Table | 22 |
| 4.0 Planning | 23 |
| 4.1 Scheduling and Tasks | 23 |
| Week 1(5-1-17 to 13-1-17): | 23 |
| Week 2(14-1-17 to 21-1-17): | 23 |
| Week 3(22-1-17 to 28-1-17): | 23 |
| Week 4(29-1-17 to 5-2-17): | 23 |
| Week 5(6-1-17 to 12-2-17): | 23 |
| Week 6(13-1-17 to 19-2-17): | 23 |
| Week 7(13-1-17 to 19-2-17): | 24 |
| Week 8(20-1-17 to 26-2-17): | 24 |
| Week 9(27-1-17 to 5-3-17): | 24 |
| Week 10(6-17 to 12-3-17): | 24 |
| Week 11(13-17 to 19-3-17): | 24 |
| Week 12(20-3-17 to 25-3-17): | 24 |
| Week 13(26-17 to 2-4-17): | 24 |

1.0 Introduction

This section provides an overview of the SRS Document and a description of the project's scope. In addition it describes the project's purpose, defines the terms used and provides the relevant references.

1.1 Purpose

The purpose of this SRS is to provide a detailed description of the requirements for Memorial University Social Network (MUNSN), a Social Network targeting all University students. It will outline the Functional and Nonfunctional requirements of the project as well as giving Use Cases to demonstrate its use. This document intended to formally define the requirements of the system. It can be used as a basis to judge whether a contract has been fulfilled or breached. It is also intended to be used as guide for developers in the development of the system and the creation of test cases.

1.2 Product Scope

We are building a University Social Network that can be deployed to any University. We are branding this version as MUNSN - Memorial University Social Network. Users access MUNSN via a Website. MUNSN provides the core functionality of a Social Network, communicating with other Users, creating groups, a timeline of User activity and more.

The Website confirms that the User is a Memorial University Student and suggests other Users to connect with. Users may create and invite other Users to form groups. Users can communicate through the creation of posts in groups or on User timelines. There will be a lost and found function allowing Users to report and claim lost items.

The Website will be built using AngularJS for client side MVC framework, and bootstrap for mobile friendly responsive design framework. The page will be served to clients using NodeJS-Express web server responding to standard POST, GET, PUT, DELETE requests. The web server will access a RESTful web service built using NodeJS-Express. The web service responds to URL requests from the web server. Typical requests involve CRUD operations of images, text, and videos. The web service accesses the MongoDB database using the MongoDB wire protocol which is quite fast since it uses no network handshake. It will use external API's such as the google maps API and angularJS API.

1.3 Overview

This document consists of four sections. This first section is this introduction and provides initial information required for understanding the next three sections.

The second section describes the general factors that affect the requirements. It outlines the context the product will operate in and provides a summary of its functions. The section also details the product's intended Users and other constraints affecting the system's requirements.

The third section is an in depth explanation of the functional and nonfunctional requirements of the product. It details the interfaces used by the system and includes the major use cases. In addition the third section includes a summary of function priority for the product.

The four and final section exists to express planning and task related information. It includes a rough summary of past and future tasks breakdown and schedule.

1.4 Definitions

| Term | Definition |
|-------------------|---|
| System | in this document, refers explicitly to the MUNSN website and all hidden functionality |
| User | someone who is registered and is using the website |
| Guest | a visitor to the site that is not logged in to an account. |
| Timeline | the default page a user will see when logged on; displays all comments, posts and other relevant activity |
| Visibility | the ability of a user to see certain aspects of a timeline. Setable by the owner of the timeline |
| Post | an item such as text, pictures or notifications a user can put on their timeline |
| Notification | a short message notifying a user of an event that appears on their timeline. Notification events include; <ol style="list-style-type: none"> 1. sending or receiving a friend request 2. accepting or ignoring a friend request 3. another user accepting or ignoring your friend request 4. A friend posts new content on their timeline |
| Friend | a User designated as a "Friend" by the current User and who has in turn designated the current User as a "Friend" as well. |
| Social Networking | the use of dedicated websites and applications to interact with other users, or to find people with similar interests to oneself. |
| Social Network | a dedicated website or other application that enables users to communicate with each other by posting information, comments, messages, images, etc. |

| | |
|---|---|
| Login | a registered user must enter their valid @mun email with matching password in order to access the full functionality of the website. |
| Password | user must have a password linked to their account. Valid passwords are subject to certain criteria including use of at least one capital letter and number and no symbols and be a minimum of 8 characters. |
| Database | a structure that stores the various information used in the system |
| Framework | an abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software |
| Node.js | a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. |
| MongoDB | an open source, document-oriented database. Non-relational. |
| API | application program interface, is a set of routines, protocols, and tools for building software applications. An API specifies how software components should interact and APIs are used when programming graphical user interface (GUI) components |
| Excalibur | Memorial University Computer Science department's private server |
| Use Case | a list of actions or event steps, typically defining the interactions between a role and a system, to achieve a goal |
| Representative State Transfer (Restful) Web Service | An architectural style capable of providing interoperability between systems and the internet. |
| AngularJS | A front-end web Javascript web frameweb that enables client side MVC framework. |

1.5 References

IEEE Software Engineering Standards Committee, "IEEE Std 830 - 1998, IEEE Recommended Practice for Software Requirements Specifications", October 20, 1998.

Wieggers, Karl E. Software Requirements, Microsoft Press, Redmond, WA, 1999.

2.0 General Description

This section will give the general factors affecting the product and its requirements. It will give context for the product and its functions along with descriptions of the expected users and the related features. Furthermore the section will detail the background informing said requirements.

2.1 Product Perspective

The system will consist of a single Social Networking website. Functionally the site will provide the basic feature set of Facebook but with access restrictions and functionality specific to the needs of Memorial University students.

The product will feature a lost and found function that will provide a map and location data for the lost items. To display this map and location data the product will connect to Google Maps through Google's API. The API will take a location provided by a User and area specifications given by the developers and return a site useable map with the location indicated.

As a Social Network the website will store extensive User, Group and Communications data. To store this data a MongoDB database will be used, the site will make full use of the read, write, update and delete functionality of the database. The database is accessed via Node.js in the HTML page on the user's browser.

2.2 Product Functions

The site will verify that Users are Memorial University Students by checking that the User has a valid Memorial University email upon account creation.

Users may find other Users whom they can designate as Friends through either a site wide search or a list of individually customised suggestions provided by the system. The suggestions take into account Courses, Friends and Program of the User.

Users may create Groups of Users and choose how other Users may join. The options for joining are open to anyone, entry upon application approval and invite only.

Users may communicate either through a built in Chat function or through Posts in Groups, on the User's own Timeline or on other User's Timelines. Furthermore Users may comment upon Posts they have access to. User's may set the permissions on their Posts to set who else can comment and see the Posts. Besides Posts a User may also create Polls for other Users to vote on with the same privacy settings as ordinary Posts. Users can upload a copy of their resume and add the courses they are taking to a calendar.

A Lost and Found will be provided allowing Users to report or look for lost or found items. This function included a generated map which indicates where the item was found.

Each User will have a Notification feed informing them of other User's activity. Activities giving notifications include Friend Requests, Posts, Polls and Comments.

2.3 User Characteristics

Only users with an active MUN email account are able to sign up for an account. Users without a MUN email address can only view the site. This restricts the set of users to;

1. Current Memorial University Students
2. Alumni
3. Professors
4. Staff

Since the use of the social network Facebook is so widespread the UI design of the MUNSN will be made intentionally similar which should facilitate learning and navigation of the UI more casual users.

2.4 Design Constraints

The product has been given several software constraints.

1. MongoDB must be used for the back-end database.
2. Node.js must be used for to access the mongodb backend from the user's front-end interface.
3. Google Map API will be used to access Google Maps for the lost and found feature.
4. Twitter Bootstrap web API will be used to for web layout / responsive design / mobile friendly site. (reword?)
5. Only users with an active MUN email address can sign up for an account.
6. Users without an account can view the webpage.
7. Product must be delivered by April 9th, 2017.
8. Project must be deployed to the MUNCS Server Excalibur.
9. The user front-end must run on all standard browsers (Internet Explorer, Firefox, Chrome, and Safari)
10. All passwords must be sent using an encrypted HTTPS connection.
11. Our system gmail account will not be associated with a visa.
12. Acceptable file formats for images include jpg and png.
13. Acceptable file formats for videos include mp4 and avi.
14. Acceptable formats for documents include .doc.

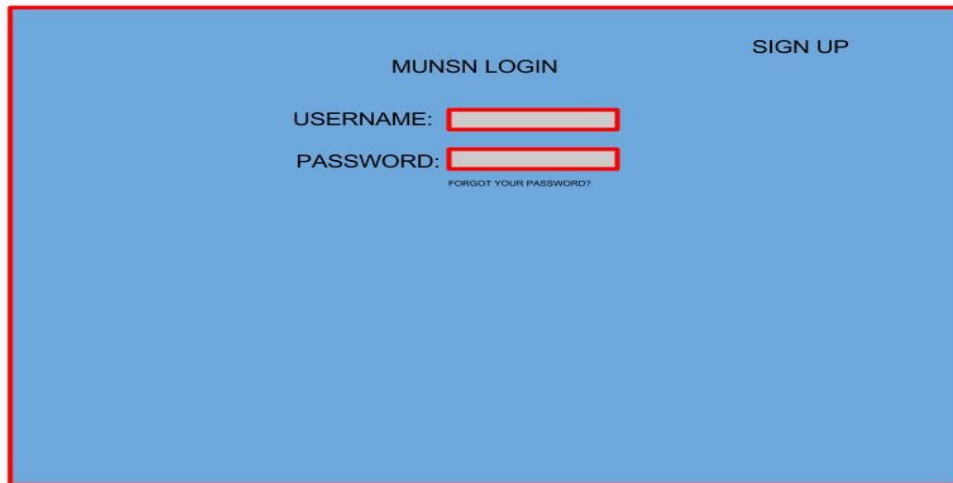
2.5 Assumptions and Dependencies

1. The user is using one of the major browsers, which includes;
 - a. Internet Explorer
 - b. Firefox
 - c. Chrome
 - d. Safrai
2. The users browser has the most recent updates installed.
3. Deployment Server must have Mongodb and Node.js with Express framework installed.

3.0 Specific Requirements

This section outlines all the specific requirements of the system including functional and nonfunctional requirements.

3.1 User Interface



The image shows a login and sign-up interface on a blue background. At the top left is the text "MUNSN LOGIN" and at the top right is "SIGN UP". Below "MUNSN LOGIN" are two input fields: "USERNAME:" followed by a red-outlined box, and "PASSWORD:" followed by a red-outlined box. Below the password field is a small link that says "FORGOT YOUR PASSWORD?".

Figure 1.0 Tentative Interface for Home Page



The image shows a tentative home page layout divided into three main sections: USER1, ACTIVITY BAR, and MUNSN. A horizontal red line separates the header from the content. Below the line, the layout is divided into two columns by a vertical red line. The left column (USER1) contains a list of posts: "POST 1", "POST 2", and "POST 3", each followed by a dashed line. Below the posts is "LOST AND FOUND ITEM 1" followed by a light blue box containing the text "---MAP AND PIN OF LOST ITEM". The right column (MUNSN) contains a list of links: "LOST AND FOUND", "USER GROUPS", "RECENT POLLS", and "ECT...".

Figure 1.1 Tentative Interface for Home Page

When a user navigates to the website they will see a login page, see figure 1.0. It will not matter at this point if the user has an existing account or not. A user with an account will be able to enter their credentials, email and password, in order to access the full website. New

users will be able to create a new account from this login screen by clicking on the link provided, see Use Case 1.

When the user has successfully logged in to their account they will be taken to their home page, see figure 1.1. On this page they will see their timeline which contains various posts including posts by the user, posts by the user's friends as well as relevant posts from courses the user may be taking. See Use Cases 4, 5, 8, and 13.

In order for the user to navigate to other pages on this site there will also be a sidebar with a host of various links that provide a shortcut to other features that are provided.

3.2 Hardware Interfaces

Supported device types include all devices cable of running an up to date version of the major browsers listed in section 2.5. Users are able to upload and retrieve images, video and text from the MUNSN system running on the Excalibur server. Users connect to the MUNSN Presentation Layer on Excalibur by accessing a Node.js-Express web server on port 80. For authentication an encrypted HTTPS connection on port 443 is used. A RESTful Web Service provides access to the mongodb database and the system's external email on Gmail's SMTP server. This is illustrated in diagram below.

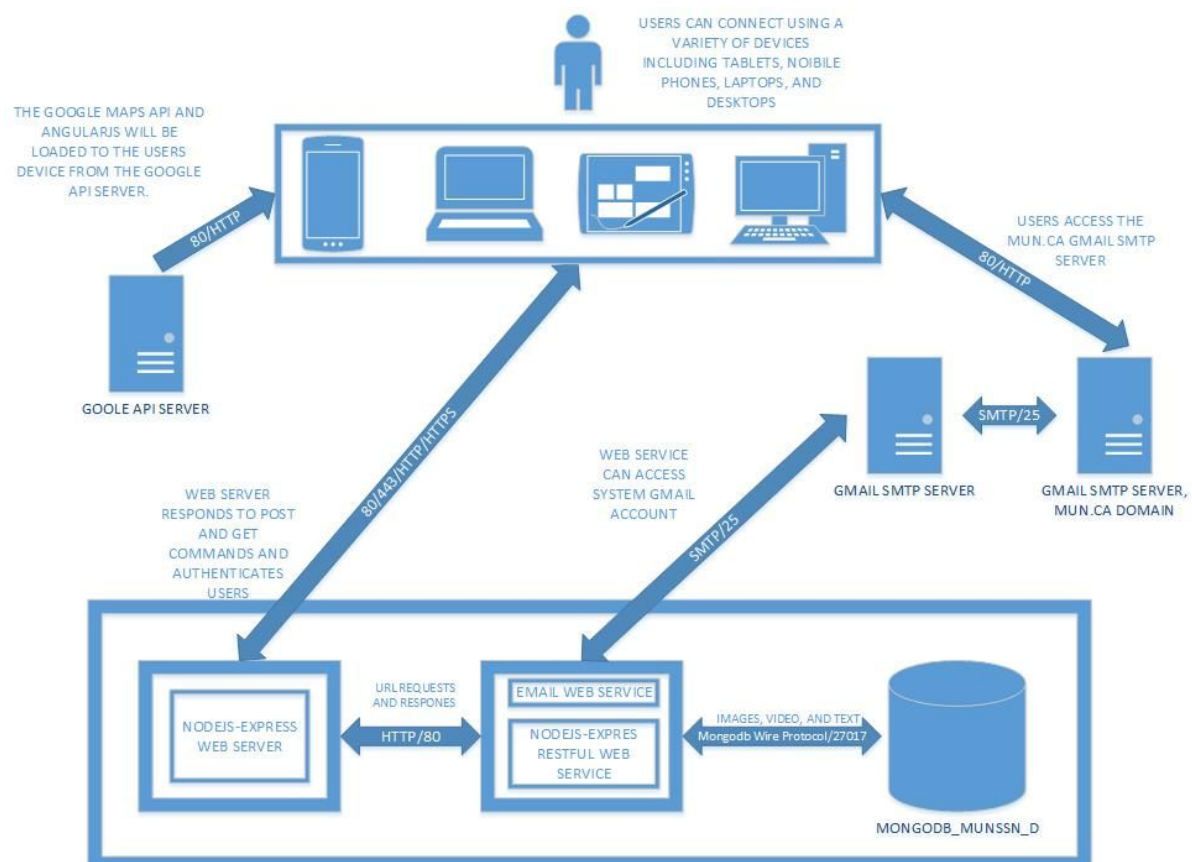


Figure 2.0

3.3 Software Interfaces

3.3.1 Interfaces

The system is broken down into three modules;

1. Presentation Module - this contains the a web server built using NodeJS with the Express framework. It responds to HTTP requests like GET, POST, etc. The user connects to the presentation module. The module itself can send requests to the service module in the form of URL's.
2. Service Module - this contains the Restful Web Service built using NodeJS with the frameworks Express and Mongoose. The Presentation Module will send the Restful Web Service a request in the form of a URL and the web service responds with the results to say a query of the mongodb database or the result of an authentication request using Mongoose.
3. Data Module - This is the actual Mongodb database. It has no direct connection to the end user and responds to requests by the web service on port 27017 using the Mongodb wire protocol. There will be requests to retrieve, store, and delete video, images, and text.

3.3.2 Software Used

1. Mongodb v3.2.11 - used as the back end database. It is an open-source, cross-platform document-oriented database program. It is classified as a NoSQL database since it does not use a relational schema. The communication between the database and the user will consist of operations the read write and modify data.
2. Node.js v4.6.1 - open-source, cross-platform, Javascript runtime environment.

3.3.3 Frameworks and API's Used

1. Google Maps Embedded API - used to get information about a user's location and display a map showing the location of lost and found items. A user gets this api by connecting to the google api server directly (https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap) . This actually exposes our API KEY to anyone who requests the web page. This could be avoided by installing the google map web service api on our web service module. Since this requires a visa to be associated with the account we choose to go with the embedded version.
2. Twitter Bootstrap Web Framework (version 3.3.7) - used to achieve responsive and mobile-friendly design. User connects to the bootstrap server directly (<https://maxcdn.com/bootstrap/3.3.7/css/bootstrap.min.css>) to get a copy of the framework on their machine.
3. AngularJS front-end web framework (version 1.6.1) - a web application framework that can build a client side MVC framework. User connects to the google api server directly (<https://ajax.googleapis.com/ajax/libs/angularjs/1.6.1/angular.min.js>) to download the library to their own machine.

4. Express (version 4) - A minimalist web framework for NodeJS. This has a web server to handle our requests. This must be installed server-side by using the terminal command 'npm install express --save'. Please refer to the expressjs.com page 'Getting started'.
5. Mongoose (version 4.8.1)- Object Modelling for NodeJS. We are going to use to handle authentication and possibly object modelling.

3.4 Communication Interfaces

3.4.1 Communication Protocols

1. Hyper Text Transfer Protocol (HTTP) - standard web based communication. Used to transfer HTML page content. Port 80 is the default port and this will be used.
2. Hyper Text Transfer Protocol Secure (HTTPS) - encrypted web based communication. This is used during login to send user and password information to the server securely. The default port is 443 which is what we will be using.
3. Simple Mail Transfer Protocol (SMTP) - used to send and receive email messages. The default port is port 25, which is what we will be using.
4. MongoDB Wire Protocol - a simple TCP/IP request/response style protocol. There is no network connection handshake. The default port is port 27017, which is what we will be using. We will not be giving users direct access the database but instead access will be through a web service.

The system requires access to an SMTP server. We plan on accessing the gmail smtp server externally through a account designated for the MUNSSN. The web service should be able to access the gmail SMTP server on port 25.

3.5 Functional Requirements

This section details the Functional Requirements of the system using Use Cases and a Use Case Diagram.

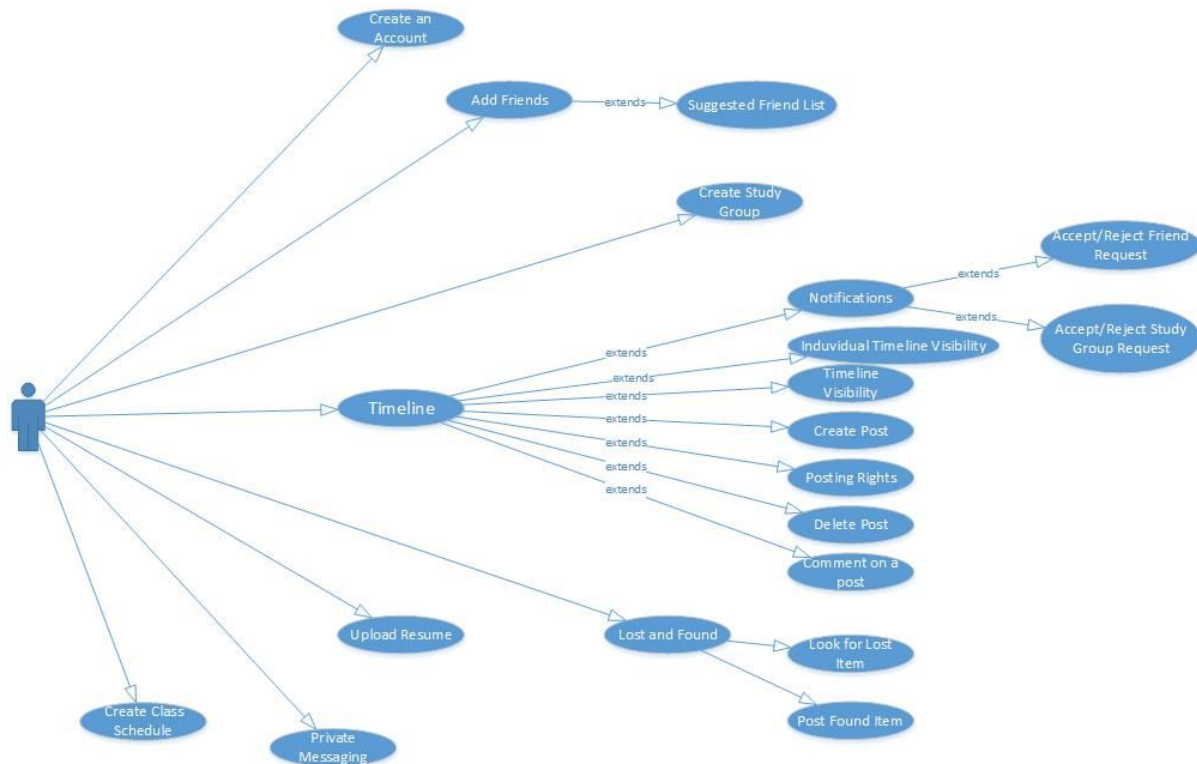


Figure 3.0

3.5.1 Use Case: *Create an Account*

3.5.1.1 Preconditions:

User's email must not be connected to an existing account.

3.5.1.2 Main Flow:

The System provides form fields for email and password as well as the option to upload a profile picture from the User's computer. When the User submits the form the System sends an email to the address provided by the User. The User reads this email and follows an included link back to a confirmation page on the Website. The User confirms their email by clicking the confirmation button on the page

3.5.1.3 Subflows:

None

3.5.1.4 Alternate Flows:

1. If the information entered into the forms does not meet the requirements the submit button will return to the signup page and display the relevant error
2. If the User attempts to upload a profile picture that fails to meet the profile picture requirements then the picture will not be uploaded and the User will be shown the relevant error.
3. System is unable to access the SMTP server. After 5 minutes the user is notified that the email server is unavailable.

3.5.2 Use Case: *Add Friends*

3.5.2.1 Preconditions:

1. User must be registered.
2. The User's friend list begins empty the first time this Use Case is run.

3.5.2.2 Main Flow:

User navigates to another User's page. The User sends a friend request to the target User using the "Send Friend Request" button [S#]. The receiving User has the request appear in their timeline and may accept or ignore [E#] the request. If accepted both Users are registered as Friends and the request is no longer shown.

3.5.2.3 Subflows:

1. After being used the "Send Request" button is disabled.

3.5.2.4 Alternate Flows:

1. If the Friend Request is ignored then it ceases to appear in the recipient's New Feed.

3.5.3 Use Case: *Suggested Friends*

3.5.3.1 Preconditions:

The user is registered

3.5.3.2 Main Flow:

The User navigates to their Friend List page. The System displays a list of 5 Users that are related to the User [E1]. The User may navigate to the Profile page of any of the Users shown in the list.

3.5.3.3 Subflows:

None

3.5.3.4 Alternate Flows:

1. If the User has no Friends, Courses, or Programs to match to other users, then the list is a random selection of Users.

3.5.4 Use Case: *Timeline*

3.5.4.1 Preconditions:

The user is registered and their own timeline is visible.

3.5.4.2 Main Flow:

The content within the timeline is sorted chronologically with the most recent at the top. If the timeline has no posts then it will be empty. Content consists of posts and *<Notifications>*. Posts contain images, messages, and video. Posts can be commented on by the user or other users that have rights to do so. *<Notifications>* are a short messages notifying a user of an event that appear on their timeline. Events include . Timeline options include Visibility, Posting Rights, and New Post. Options available on an individual post include Delete, Comment, and Visibility. When the *<User>* clicks the timeline option, 'New Post', the new post window opens up. It has an area for typing text and the following options available, Upload Video/Image, Publish, and Cancel. When the *<User>* types text, text appears. When the *<User>* Clicks Upload Video/Image a filepicker is shown. When the *<User>* selects a file from the filepicker and hits OK the file picker disappears and the selected media appears on the new post window. When the *<User>* hits publish the new post window disappears and the new post appears at the top of the user's timeline

3.5.4.3 Subflows:

1. *<User>* can respond to notifications by accepting or ignoring friend requests. Accepting or ignoring a friend request is a notification event and it will appear on the User's timeline.
2. *<User>* can reply to messages from other users on their timeline. (move to main flow or separate use case)x not sure if this is even a requirement.
3. When creating a new post the *<User>* hits the cancel button. New Post window disappears and *<User>* is returned to their timeline.

3.5.4.4 Alternate Flows:

1. *<User>* has attempted to upload an unsupported filetype or has exceeded max size limits. The system notifies the user that the file type is unsupported or that they have exceeded the max file size limit depending on the violation and they will then be given another to upload the file or to cancel the operation.
2. *<User>* has exceeded the max character limit for text in a post. User is unable to type anymore text and is alerted.

3.5.5 Use Case: *Timeline Visibility*

3.5.5.1 Preconditions:

1. There are 3 *<Users>* referred to as User A, User B, and User C and a *<guest>*.
2. User A is friends with User B and User C
3. User B has timeline visibility set to 'Everyone'
4. User C has timeline visibility set to 'Only Friends'
5. User A has timeline visibility set to 'Only User'

3.5.5.2 Main Flow:

The Timeline option Visibility is visible on the timeline menu bar. The available options are Only User, Everyone, Only Friends, List of Friends. User A makes a post on his own timeline. When User B, User C and *<Guest>* navigate to User A's homepage the new post is not visible. User B makes a post on his timeline. The post is visible to both User A, User C and *<Guest>*. User C makes a post on his timeline. The post is visible to User A, but not to

User B and <Guest>. User A sets timeline visibility to 'List of Friends' and chooses only User B. User A's posts are visible to User B, but not to User C and <Guest>.

3.5.5.3 Subflows:

None

3.5.5.4 Alternate Flows:

None

3.5.6 Use Case: *Individual Timeline Item Visibility*

3.5.6.1 Preconditions:

1. There are 3 <Users> referred to as User A, User B, and User C and a <guest>.
2. User A is friends with User B and User C
3. User B has a timeline post with individual visibility set to 'Everyone'
4. User C has a timeline post with individual visibility set to 'Only Friends'
5. User A has a timeline post with individual visibility set to 'Only User'

3.5.6.2 Main Flow:

The Individual Timeline option Visibility is visible on the each timeline post. The available options are Only User, Everyone, Only Friends, List of Friends. When User B, User C and <Guest> navigate to User A's homepage the post is not visible. User B's post is visible to both User A, User B and <Guest>. User C's post is visible to User A, but not to User B and <Guest>. User A sets timeline visibility for individual post to 'List of Friends' and chooses only User B. This post is visible to User B, but not to User C and <Guest>.

3.5.6.3 Subflows:

None

3.5.6.4 Alternate Flows:

None

3.5.7 Use Case: *Timeline Posting Rights*

3.5.7.1 Preconditions:

1. There are 3 <Users> User A, User B, and User C, and a <Guest>.
2. User A is friends with User B and User C.
3. User A has timeline posting rights set to 'Only User'
4. User B has timeline posting rights set to 'Everyone'
5. User C has timeline posting rights set to 'Only Friends'

3.5.7.2 Main Flow:

User A, User C and <Guest> can navigate to User B's timeline and create a post successfully. When User B, User C and <Guest> navigate to User A's timeline and click new post they all receive a message informing them that they do not have rights to post to this

timeline. User A can navigate to User C's timeline and create a post, but <Guest> and User B receive a message informing them that they do not have rights to post to this timeline.

3.5.7.3 Subflows:

None

3.5.7.4 Alternate Flows:

None

3.5.8 Use Case: *Posting Comments*

3.5.8.1 Preconditions:

User must have a valid account and a friend's status to comment on.

3.5.8.2 Main Flow:

A user can see a post from a friend or another user due to the visibility setting. User A posts a comment on User B's status. User A's comment can only be edited by them but the status can be deleted by User B due to hierarchy of the posting.

3.5.8.3 Subflows:

None

3.5.8.4 Alternate Flows:

1. The user changes the privacy of the status while the comment is being produced and then the comment won't be able to be posted if the privacy setting is changed before the comment is posted.

3.5.9 Use Case: *Create Study Group*

3.5.9.1 Preconditions:

The user must be a valid member and have friends in the network.

3.5.9.2 Main Flow:

Group Creator becomes the admin. The admin can invite other members to join the group and can set different levels of accessibilities for each invited member. A public group will allow any member to join instantaneously. A private group can only add members by the admin inviting them, accepting their invitation to join the group or by other members inviting them to join, if they are allowed to do so. Within the group content can be shared and posted by all the members.

3.5.9.3 Subflows:

None

3.5.9.4 Alternate Flows:

None

3.5.10 Use Case: *Lost and Found*

3.5.10.1 Preconditions:

User is registered must have found a lost object.

3.5.10.2 Main Flow:

A lost item can be reported by a user by posting a picture of it with a short description and a location tag in order to show the location of the item in the lost and found section. A user can contact a poster regarding a lost item by their email, if they are friends the user has access to the poster's phone number.

3.5.10.3 Subflows:

None

3.5.10.4 Alternate Flows:

None

3.5.11 Use Case: *Create Schedule*

3.5.11.1 Preconditions:

User is registered.

3.5.11.2 Main Flow:

The user initiates the process by navigating to the schedule section of their profile. They are asked if they want to create a new schedule for this semester. They will then be asked to create a new slot and then proceed to enter a date, name and time (fill box). When complete they will submit the new information which will then be saved and be put into the schedule.

The User will now have the option to enter a new slot and so on until all the desired slots have been entered and the user closes the program.

3.5.11.3 Subflows:

None

3.5.11.4 Alternate Flows:

None

3.5.12 Use Case: *Upload Resume*

3.5.12.1 Preconditions:

User is registered

3.5.12.2 Main Flow:

The user will navigate to the resume section of their profile where they will be prompted to upload a new resume. If they choose to do so they will be provided with the ability to choose a file from their computer from which they can choose their resume.

3.5.12.3 Subflows:

None

3.5.12.4 Alternate Flows:

1. If the user navigates to the resume section of their profile and a resume already exists they will have the option to delete/remove the existing resume. If they choose to do so they will follow the main flow.
2. If the user attempts to upload a file that is not supported or not the expected file type they will receive an error. All standard image files should be accepted.

3.5.13 Use Case: *Create A Poll*

3.5.13.1 Preconditions:

User is registered.

3.5.13.2 Main Flow:

The initial member will begin creating a poll, they must choose a course they are currently taking. These options will be limited to courses that the user has set in their schedule, see Use Case 3.5.11. They will fill in the information needed about the poll (name, question and answers) then the poll will be posted to that user's timeline and will be visible to other members taking that course. Other members will vote on the poll or respond to it with a comment. The original member who created the poll may choose to close the poll at anytime disabling other members from commenting or voting.

3.5.13.3 Subflows:

None

3.5.13.4 Alternate Flows:

If the user attempts to make a poll about a course they are not taking they will receive an error and be brought back to the beginning of the main flow.

3.5.14 Use Case: *Communicate With Chat*

3.5.14.1 Preconditions:

1. User is registered
2. User has at least one Friend

3.5.14.2 Main Flow:

User navigates to Chat Page. User selects a Friend and views their Conversation. User types in the desired message in the Chat Box and presses the Send Button. The Friend will receive a notification indicating they have received a message and may follow the same process as the original User to reply.

3.5.14.3 Subflows:

None

3.5.14.4 Alternate Flows:

None

3.6 Nonfunctional Requirements

3.6.1 Performance

1. A user's homepage should load in less than 5 seconds.
2. When opening an account a user should receive an email from the system within 5 minutes.
3. Any user action should have a response within 5 seconds. This includes, playing a video, creating a post, creating a study group, etc.
4. System is available 99.5% of the time.
5. Search results should be returned within 10 seconds.

3.6.2 Usability

1. System options are highly visible and easily discoverable to the user.
2. The system is easy to navigate.
3. Every feature should be able to be accessed by the user in 3 clicks or less.

3.6.3 Security

1. Communication/ User authentication - users are authenticated through encrypted communications - the HTTPS protocol. We may use the Mongoose module to handle authentication.
2. Email verification - only students with @mun email accounts will be able to register. An email will be sent on creation of a new account to verify the user.
3. Password protection - on creation of a new account a user will create a password for themselves linked to their email. Passwords must contain at least one capital, one number and no symbols and be a minimum of 8 characters long.

4. Privacy settings - users will be able to set the visibility of their post as outlined in use cases 3.5.6 and 3.5.7.
5. Database security - Users do not have direct access to the back-end database. Access is through a web service - a RESTful API. The actual client code contains no direct queries to the database. (This is close to what SQL calls a stored procedure)
6. Security is improved by employing a 3 Tier Architecture which includes the Data Layer, the Service Layer, and the Presentation Layer. This is possible by breaking the system into 3 modules as discussed in section 3.2. The Data Layer is the Excalibur server which has Mongodb and NodeJS installed - this will be our backend. The service layer runs the RESTful web service. It communicates with both the Presentation layer through port 80 and the Data Layer through port 27017. The presentation layer serves up the web pages to clients. It communicates to the client on port 80 and port 443 for authentication and the service layer on port 80. Users actually are connected to the presentation layer and have no direct to the back-end database.

3.6.4 Scalability

1. The current system should be able to potentially support all Memorial University students.
2. System should be able to handle double the number of users every 4 years. This is since Alumni are able to maintain or open an account and the average length of a degree is four years. In 2029 we should expect the system to be able to handle 8 times the current users.
3. Current scalability is limited by the constraint of only being able to deploy the system to one server, Excalibur.
4. Using a RESTful web service and a 3 tier architecture with Data, Service and Presentation Layers is excellent for scalability. It enables the project to be broken down into 3 modules that can each be deployed to a different server. You would have a server to serve web pages, a server to handle service requests such as authentication, smtp requests, and mongodb CRUD (Create, Remove, Update, Delete) operations and a server that hosts the mongodb database. A Database Administrator (DBA) can fine tune each server to its specific performance and security needs as demand grows. Also by separating the modules you could actually have an entirely different front-end access the systems back-end through the web service. If we did not break out into modules then a different front-end would mean a re-write of the entire system.
5. The overall demand on the system is reduced by having users connect to an external server directly and download a copy of a library rather than the system's own servers.
6. Using a RESTful web service is a good idea versus WSDL or SOAP since it is not restricted to return data in a specified format. JSON or XML, etc can be returned.
7. Node.js event-driven architecture is capable of asynchronous I/O which optimizes throughput and scalability. It is able to handle many concurrent connections without forcing developers to deal with handling threads. This is an advantage over servers like Apache or IIS.

3.6.5 Maintainability

1. All connection strings should be put into configuration files for easy maintenance.
2. All features that apply to Universities in general should be kept on the main branch.

4.0 Planning

4.1 Scheduling and Tasks

Week 1(5-1-17 to 13-1-17):

- **Tyler Vey:** Use Cases, Functional and Nonfunctional Requirements from project description 1-3.
- **Saahil Budhrani:**Use Cases, Functional and Nonfunctional Requirements from project description 7-9.
- **Mark Hewitt:**Use Cases, Functional and Nonfunctional Requirements from project description 9-12.
- **Allan Collins:**Use Cases, Functional and Nonfunctional Requirements from project description 4-6.

Week 2(14-1-17 to 21-1-17):

- **Tyler Vey:** General improvements on SRS Formatting and distribution of sections 1, 2 and 4.
- **Saahil Budhrani:** General improvements on SRS Formatting and distribution of sections. Worked on sections 3.5(Use Case diagram), 3.6, 3.7, 4 and powerpoint presentation.
- **Mark Hewitt:** General improvements on SRS Formatting and distribution of sections. Worked on definitions and section 3.0 through 3.5
- **Allan Collins:** General improvements on SRS Formatting and distribution of sections. Worked on sections 1.2, 2.3, 2.4, 3.2 - 3.4, 3.5.4, 3.5.5, 3.5.6, 3.5.7, 3.6.

Week 3(22-1-17 to 28-1-17):

- Finalization of SRS document as a group and minor formatting.

Week 4(29-1-17 to 5-2-17):

- Powerpoint presentation
- Milestone 1 presentation.
- Work commences on Architectural Document, system's decomposition into modules.

Week 5(6-1-17 to 12-2-17):

- Division of module responsibility between team members.
- Creation of UML and other visualisations.
- Progress on Architectural Document.

Week 6(13-1-17 to 19-2-17):

- Finalization of Architectural Document.
- Milestone 2 presentation.
- Work commences on Module Documents.

Week 7(13-1-17 to 19-2-17):

- Work on descriptions of the functionality and interface of each module.
- Work on descriptions of module testing plans and demonstration.

Week 8(20-1-17 to 26-2-17):

- Continued work on descriptions of the functionality and interface of each module.
- Continued work on descriptions of module testing plans and demonstration.

Week 9(27-1-17 to 5-3-17):

- Finalization of Module Documents.
- Milestone 3 presentation.
- Preliminary System Development.

Week 10(6-17 to 12-3-17):

- Creation of Modules based on Module Documents.

Week 11(13-17 to 19-3-17):

- Integration of all modules.
- Creation of System Testing Document.

Week 12(20-3-17 to 25-3-17):

- Deployment of Integrated System.
- Creation of Data for System Demonstration and testing.
- Improvement upon System Testing Document.

Week 13(26-17 to 2-4-17):

- Completion of Integrated System.
- Submission of System Testing Document.
- Milestone 4 presentation.