SOFTWARE REQUIREMENTS SPECIFICATION

For CS 4770 Group B

Prepared by:

Osede Onodenalore, Samuel Ash, Jeff Conway, Maria Aloysius.

February 5, 2017

Contents

[Introduction 4](#_Toc474073365)

[1.1 Purpose 4](#_Toc474073366)

[1.2 Scope 4](#_Toc474073367)

[1.3 Definitions, Acronyms, and Abbreviations 5](#_Toc474073368)

[1.4 Reference 5](#_Toc474073369)

[1.5 Overview 5](#_Toc474073370)

[Overall Description 6](#_Toc474073371)

[2.1 Product Perspective 6](#_Toc474073372)

[2.2 Product Functions 6](#_Toc474073373)

[2.3 User Characteristics 6](#_Toc474073374)

[2.4 Design and Implementation Constraint 6](#_Toc474073375)

[2.5 Assumptions and Dependencies 7](#_Toc474073376)

[2.6 Operating Environment 7](#_Toc474073377)

[Specific Requirements 7](#_Toc474073378)

[3.1 User Interfaces 7](#_Toc474073379)

[3.2 Hardware Interfaces 7](#_Toc474073380)

[3.3 Communications Interfaces 7](#_Toc474073381)

[3.4 Software Interfaces 8](#_Toc474073382)

[Functional Requirements 8](#_Toc474073383)

[4.1 User Creates a Profile 8](#_Toc474073384)

[4.2 User Signs In 8](#_Toc474073385)

[4.3 User Accesses Another User’s Profile 9](#_Toc474073386)

[4.4 User Adds a Friend 9](#_Toc474073387)

[4.5 User Creates a Post 10](#_Toc474073388)

[4.6 User Edits the Visibility of a Post 10](#_Toc474073389)

[4.7 User Makes a Comment on a Post 11](#_Toc474073390)

[4.8 User Creates a Group 11](#_Toc474073391)

[4.9 User Edits a Group 12](#_Toc474073392)

[4.10 User Creates a Schedule 12](#_Toc474073393)

[4.11 User Uploads a Resumé 13](#_Toc474073394)

[4.12 User Adds Item to Lost and Found 13](#_Toc474073395)

[4.13 User Removes an Item from Lost and Found 14](#_Toc474073396)

[4.14 The User Creates a Poll 14](#_Toc474073397)

[4.15 The User Votes on a Poll 14](#_Toc474073398)

[Use Case Diagram 16](#_Toc474073399)

# Introduction

## 1.1 Purpose

The intended users for this software are students at Memorial University of Newfoundland. Its purpose is for students within the university to be able to interact with each other through the system and share opinions and content. Initially, this document is to be read by our professor/client Dr. Saeed Samet and his teaching assistant Navid Shekoufa. However, we recommend any individual who has knowledge of programming and web development may read this document.

## 1.2 Scope

This project will be written in Node.js and MongoDB as its database.

Objective:

Node.js is an event driven architecture that is scalable in Web Applications with many input/output operations. One advantage of using Node.js is that the functions are designed to be non-blocking. Commands do not have be sequential and wait for the preceding command to finish.

MongoDB is an open-source cross platform environment that uses JavaScript server side execution for queries. It can be used to store files over multiple machines and provides function to manipulate files.

Benefit:

Node.js works very well with the document-oriented database program MongoDB. This program is also written in JavaScript like Node.js.

Goals:

Since Node.js is primarily used to build network web programs such as web servers, it is a better environment for this software. And we will be using MongoDB to store any data relating to the user and the associated functionalities and privileges. When necessary we will retrieve any data to use for the output of the webpages.

## 

## 1.3 Definitions, Acronyms, and Abbreviations

**Node.js:** JavaScript runtime environment used to develop many server applications

**MongoDB:** It is an open-source, cross platform database program.

**MUN:** Memorial University of Newfoundland

**HTML:** Hypertext markup language

**GIT:** Open source environment used to handle projects in a fast and efficient way.

**D2L:** Desire2Learn is an educational technology used at the university by professors and students

**UC**: User Classes

**The System:** refers to the software product this document is being specified for. Since it is a network which is being developed, it is easier to refer to the product as **the system**.

**The User:** refers to the primary user of the system at any given time. In some situations, where privileges are important, for example when discussing editing a post or group, **the user** is used to refer to the creator of the post or the owner of the group.

**The Other User** - refers to another user of the system with whom the user is interacting.

## 1.4 Reference

Resources used to create this document have been used from the templates found on D2L for the CS 4770 course and an example of an SRS document found online from the URL <http://www.cise.ufl.edu/~dgoldste/se/SRS.doc> and <http://www.slideshare.net/indrisrozas/example-requirements-specification>

## 1.5 Overview

This document gives a description of the software, the product, the intended users, the software and hardware constraints, assumptions, dependencies. It also gives a general idea of the product functions and it’s use cases. The use cases show the flow of control and it is portrayed using use case diagrams. Additionally, the document describes the functional and the non-functional requirements for the software.

# Overall Description

## 2.1 Product Perspective

This product is a self-contained product. Within this system, a student is able to register as long as they have a student ID and an email just like the MUN self-service application. Like many social media applications, this product enables the user to add friends, communicate with them and much more. This product assumes that the student is a verified user at the university with an email suffix “@mun.ca”, making it indirectly dependent to the MUN database.

## 

## 2.2 Product Functions

The major functions the product must perform is:

i) create a user profile

ii) sign in a registered user

iii) display another user's profile

iv) send friend requests, add friends when accepted

v) create a post, allow user to edit own posts and comment on any post they can see.

vi) create a group, allow group owner to edit a group, allow users to join groups

vii) create a course schedule for any user

viii) allow users to upload a resume

ix) maintain lost and found section, allow users to create posts within it

x) create a poll, allow owner to delete polls, allow users to vote on polls

## 2.3 User Characteristics

The User characteristics of the intended users is novice - advanced education level, provided the user is a registered student at University. Any type of user will be able to access the software provided they are a registered student at MUN or have administrative privileges.

## 2.4 Design and Implementation Constraint

Any regular user will be able to perform any basic functions like signing in, creating a profile, adding friends, creating/editing a post, joining a group, creating and voting a poll. However, each user has to have a valid email id that ends with mun.ca, which is the way the system assumes that the user is a MUN user. Every User has privacy restrictions; the profile has limited visibility for non- friends/ the public. But for every friend in the profile have complete access to the user's profile and vice-versa.

## 2.5 Assumptions and Dependencies

Any Software related dependencies will be listed in a read me file on the GIT repository. We are assuming that whoever reads the documents has already viewed this file and has an understanding. Dependencies: Use Cases, Diagrams, Functional/Non-functional

## 

## 2.6 Operating Environment

The Project is web-based and will be hosted on the university server. It is intended to work on any browser in any operating system. It is currently assumed that it is only viewed on a personal computer and not on any mobile environment.

# Specific Requirements

## 3.1 User Interfaces

The first interface will be our login screen. In this interface the user will have to input their username and password to gain access to their account and profile.

We intend to have an interface for creating and post and all of the associated functionalities such as commenting, uploading images and publishing content

As well an interface for shared interaction between users such as the study groups and lost and found.

## 3.2 Hardware Interfaces

* The system must be connected to the internet.
* The system will be deployed on a MUN server that will connected using a port number.
* Network interaction will use https

## 3.3 Communications Interfaces

The system will interact with the database to retrieve any necessary data related to the user, friends, groups and content. This will be connected via internet.

## 

## 3.4 Software Interfaces

Along with an internet connection. The system will make indirect use of an internet browser. As mentioned before, this application is written in Node JS(v4.6.1) with use of MongoDB (v3.2.11) as the database. Other than that, the system does not tell any software what to do.

# Functional Requirements

For each of the following functional requirements, each requirement will be discussed in the following fashion:

1. Explanation: a brief description of the requirement
2. Inputs: the input required to achieve the expected output
3. Processing: what happens when the input is received by the system
4. Expected Output: an explanation of what is expected from the system for this requirement.

## 4.1 User Creates a Profile

4.1.1 The user should be able to create a profile, with an @mun.ca username required, and receive a confirmation email with a link. When the link is clicked, the user should return to the index page and be notified that their profile is ready.

4.1.2 The user must input a number of required fields, including their real name, a chosen username (which must end in @mun.ca), a password, etc.

4.1.3 The system must verify that these fields are appropriate, including ensuring that the input username includes @mun.ca. If any field is not as desired, the user will be told to change it. If all required fields are filled and meet the desired criteria, the system must create a new profile associated with the user in the database and send a confirmation email to the provided username/email.

4.1.4 Once the system has sent the confirmation email to the user, the user should be able to log in to the system at any time after the confirmation link has been accessed.

## 4.2 User Signs In

4.2.1 The user should click "sign in" and be presented with fields to enter their username and password, and then click "sign in" to be taken to their homepage.

4.2.2 The user must input their username and password correctly in order to access their profile.

4.2.3 The system should present the user with a username and (hidden characters) password field when they click "sign in”. The system must verify that the user attempting to log in exists within the database. If not, the system will inform the user that the username or password is incorrect. If so, the system must ensure that the password is correct for the associated username before logging in the user. If not, the system will inform the user that the username or password is incorrect. If so, the system should start a session and allow the user to use the system.

4.2.4 The system should present the user with their own personal homepage after logging them in.

## 4.3 User Accesses Another User’s Profile

4.3.1 The user should be able to access the profiles of other members a number of different ways, including via a search, through a received friend request, through a group, etc. The user should click that member's name to be taken to their profile.

4.3.2 The user is able to access another user’s profile a number of ways. The user must click the profile of the other user, which can be displayed to the user via a search, through a received friend request, or through a group.

4.3.3 The system must acknowledge the request of the user to view the other user’s profile page and redirect the user there. When a user navigates to another member's profile, the visibility of each post should be checked and only those posts that the user has permission to view should be displayed.

4.3.4 The user should find themselves on the profile page of the other user.

## 4.4 User Adds a Friend

4.4.1 The user should be able to send a friend request to another user by clicking a link to do so once they are on the other user’s profile page.

4.4.2 While on the profile page of the other user, the user should click “Add Friend” to begin the process of adding the other user as a friend.

4.4.3 When the user attempts to add the other user as a friend, the system must send a notification shortly thereafter to the other user. The other user may view this notification from their homepage and may choose, at their discretion, accept or decline. If the other user chooses to decline, the user will not be notified of this, but they will not be added to each other’s friends lists. If the other user chooses accept, the system must place a reference to each user in the other’s friends list, likely their username, so that it can refer to each user as being confirmed as friends with the other in the future.

4.4.4 As mentioned above, after the user attempts to add a friend, the notification of this request must be sent to the other user.

## 4.5 User Creates a Post

4.5.1 The user should be able to click "create post" while on their homepage or on the page of a group they belong to, and have the ability to post content.

4.5.2 The user must input at least once character into a text field and/or an image via a file upload field before clicking “post”.

4.5.3 The system must present the user with field(s) though which they can post content when requested. When a user creates a new post the system should update the database, display the content of the post in the appropriate place (on the user’s profile or in a group) within 3 seconds, and refresh the page.

4.5.4 The content of the post should be viewable shortly after it is posted in the appropriate location (on the user’s profile or in a group).

## 4.6 User Edits the Visibility of a Post

4.6.1 User should be able to edit the visibility of any post they have made by clicking on "edit post" and changing the visibility.

4.6.2 The user must request to edit their post, and must provide whatever changes desired (changes to text, a replacement photo, or removing the text or photo of a post so long as one of those remains), changing the visibility options for a post (public, friends only, selected list of friends or only me (the user)) or possibly deleting the post entirely.

4.6.3 The system should present the user with options for visibility, along with actual editing ability, when the user clicks "edit post”. When the user edits the visibility options of a particular post, the system should update the database with the new permission rules. The new permission rules and any changes should take effect within 3 seconds.

4.6.4 The post will be edited in any way the user desires, or removed if desired, and any changes should take effect within 3 seconds.

## 4.7 User Makes a Comment on a Post

4.7.1 The user should have the ability to click "reply" on any post they can see (on their dashboard or in a group) and should be able to post content underneath a post.

4.7.2 Once the user clicks “reply”, they must enter at least one character in the presented text field before being able to click “post”.

4.7.3 System should present the user with field(s) for posting content under a post when requested. When a user attempts to reply to a post, the system should check if that user has permission to reply based on the visibility rules of the post.

If they have permission, the system should prompt the user to enter a reply. Once submitted (and verified; ensure the text field is not empty, etc.) the database should be updated, the reply should be displayed within 3 seconds, and the user’s page should be refreshed. The system should send the owner of original post a notification that informs them someone has added a comment to their post.

If they do not have permission, the system should display a message explaining why the reply cannot be made.

4.7.4 The user’s comment should be displayed underneath the original post within 3 seconds.

## 4.8 User Creates a Group

4.8.1 The user should have access to a groups page, accessible from their homepage, on which they can create a group, or edit any owned group.

4.8.2 The user must fill out a number of fields (including group name, group purpose/description, etc.) and choose the group permission (public, private) when creating a group.

4.8.3 When the user requests to create a new group, the system should ensure the group details are acceptable (required fields are filled out, permission is set, etc.) Once accepted, the system should update the database to include the new group, add the user who created it to the group and set them as the owner. The system is then responsible for ensuring the visibility of the group (set by the owner at creation and can be changed at any time).

4.8.4 Once the user has created a group, it should become visible to all users who can see it (based upon the permission chosen by the user). The user then has control of the group to add or remove any members, create or delete any posts, and update the group details at any time.

## 4.9 User Edits a Group

4.9.1 The user should be able to edit any group they own to change the attributes of the group, add or remove members, remove posts, etc.

4.9.2 When the user requests to edit one of the groups they own, they must perform any changes they desire (add or remove members, change group name, group description, permission) before being able to click “confirm changes”.

4.9.3 The system must allow the user to edit some, if not all, attributes of the group, including possible deletion. The system is responsible for updating any edits in the database within 5 seconds.

When the user invites a user to the group, the system should send a notification to the other user asking them if they would like to join the group. If accepted, the system should update the group in the database to include the other user as a member.

When the user changes another group member's invite privileges, the system should update the other user’s privileges in the database and send a notification of their new privileges to the other user.

When the owner removes a group member from the group, the system should update the group in the database by removing the other user, and should send a notification to the other user informing them of their being removed from the group.

When the owner deletes the group, the system should update the database by removing the group, and should send a notification to all group members of its deletion.

4.9.4 After editing a group, the user should be informed their changes have been saved, and the page should be refreshed.

## 4.10 User Creates a Schedule

4.10.1 The user should be able to create a (one) schedule (from their profile page) of their courses for their friends to view.

4.10.2 When the user wishes to create a schedule of their courses, they must do so from their own profile page. The user must fill out necessary fields including course name/number, days and times of each class before they can publish their schedule.

4.10.3 The system must allow the user to input their course schedule, however, the user will be limited to a set number of courses (6) to put into their schedule. The system must store the data from the schedule and associate it with the user, and redirect the user back to their own profile page.

4.10.4 After creation of the user’s schedule, the schedule should be visible to any of the user’s friends from the user’s profile page.

## 4.11 User Uploads a Resumé

4.11.1 The user should be able to upload a (one) resumé (from their profile page) for their friends to view.

4.11.2 To display their resumé from their profile, the user must upload their resumé in an appropriate file format.

4.11.3 The system must ensure that the file format the user provides for upload can be displayed properly on the user’s profile. If it cannot be displayed properly, the user must be notified by the system to try to upload their resumé in an appropriate file format, and the system should provide a list of formats which work for the user to consider. The system is responsible for maintaining the file for the user’s resumé in the database and associating it with the user. The system should redirect the user back to their profile page

4.11.4 After the user uploads their resumé, it should be visible to any of the user’s friends from the user’s profile page.

## 4.12 User Adds Item to Lost and Found

4.12.1 The user should be able to create a post in the lost and found section (available from dashboard) about an item they have found.

4.12.2 To add an item to the lost and found, the user must include an approximate location and a short description of the item they have found, along with a method of contact (and possibly a photo).

4.12.3 The system should present user with fields to create a post about an item they have found, when requested. The user should be able to include a short description, location, and a photo of the item, so system must present fields appropriate for this.

When the user attempts to add an item to the lost and found, the system should first confirm that they are signed in. If not, give the user an error. If so, the user should be redirected to the appropriate page to enter information about the item.

When a user submits information about an item, the system should confirm the required fields are filled out (description, location found, etc.). If the fields are not filled, the system should give the user an error. If the fields are filled, the system should update the database to include the new item.

4.12.4 After adding an item to the lost and found, the item should appear near the top of the lost and found section (most recent adds to the lost and found should appear at the top of the page).

## 4.13 User Removes an Item from Lost and Found

4.13.1 Once the other user (in this case the owner of an item) has been in contact with the user (the person who found their lost item), the user should be able to remove the post for that item.

4.13.2 Upon being contacted by the other user outside of the system, the user should be able to return to the lost and found section and click “remove post” on the post that the user made corresponding to the item owned by the other user.

4.13.3 The system must ensure that any post in the lost and found section can be easily removed by the user who posted it. When attempting to remove an item from the lost and found, the user should be asked if they are sure of this action. Upon removing the item, the system should update the database to reflect this change and update the site within 3 seconds.

4.13.4 Upon removal of an item, the lost and found section should appear the same, with the exception of the item removed being no longer visible.

## 4.14 The User Creates a Poll

4.14.1 The user should be able to create a poll from within a group they are in, and be able to include the question/proposal as well as multiple choices for other users to answer, up to a limit of 7 possibilities.

4.14.2 To create a poll, the user must access the page of the group they wish to create a poll in, and must provide the question which they are asking, as well as up to 7 options for other users to choose as a response.

4.14.3 The system is responsible for storing the poll and all answers to it in the database, and make it visible to all members of the group it was created in. After a set amount of time (selected by the user), the poll should close and the answers to it should be visible for 48 hours.

4.14.4 Upon creation of the poll, it should be visible to all members of the group in which it was created.

## 4.15 The User Votes on a Poll

4.15.1 The user should be able to see polls in any group they're in, and answer any polls they wish to.

4.15.2 To vote on a poll, the user must simply select whichever answer they wish, and click “submit”.

4.15.3 The system must ensure that all members of a group can see and vote on a poll created within that group. The system must check to make sure that the user has not already answered the poll. If not, the system must give the other user the option to choose a response. If so, the other user should not be able to select a response.

4.15.4 After voting on a poll, the user should no longer be able to vote on that poll, and their answer should be stored in the database.

# Use Case Diagram