SOFTWARE REQUIREMENTS SPECIFICATION

For CS 4770 Group B

Prepared By:

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

February 5, 2017

Table of Contents

Introduction 3

1.1 Purpose 3

1.2 Scope 3

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

1.5 Overview 4

Overall Description 4

2.1 Product Perspective 4

2.2 Product Functions 5

2.3 User Characteristics 5

2.4 Design and Implementation Constraint 5

2.5 Assumptions and Dependencies 5

2.6 Operating Environment 6

Specific Requirements 6

3.1 User Interfaces 6

3.2 Hardware Interfaces 6

3.3 Communications Interfaces 6

3.4 Software Interfaces 6

Use Case Digram 8

# 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. 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 developing many server applications

MongoDB:

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

MUN: Memorial University of Newfoundland

HTML: Hyper text 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

## 1.4 References

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, dependancies. It also gives a general idea of the product functions and it’s use cases. The use cases shows the flow of control and it is portrayed using use case diagrams. Additional to it, 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 products 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

- Possibly talk about TCP/IP???

## 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.(This probably needs some better description)

## Use Case Digram