MUNSN Software Requirements Specifications

Revision 1

February 5th 2017

Group F

REVISION NO.	REVISED BY	DATE	REVISION DESC.
0	Andrew	01/16	Created document
0.1	Andrew	01/18	Added use cases and use case diagram
0.2.0	Kyle	01/19	Added Non-Functional Requirements
0.2.1	Kyle	01/30	Uploaded a sample SRS to Reference material. Did some minor edits to match, perhaps change further (in ref. to use cases and functional requirements)
0.2.2	Kyle	01/30	Began work on External Interfaces. Editing and more detail may be required. There are a large amount of repetition between sections.
0.3	Andrew	01/02	Filled out introduction section
0.4	Andrew	01/05	Modified functional requirements
1.0	Andrew	01/05	Revision 0 finalized
1.1	Andrew	02/24	Added extra feature (instant messenger) Added use case for extra feature and modify account
1.2	Andrew	02/26	Added up-to-date architecture section

Introduction

Purpose

The purpose of this document is to provide a detailed description of the MUN Student Social Network. It will contain explanations of purpose, design and functionality of the features, interfaces and system along with any constraints under which they must be operated.

Intended Audience

This document is intended to be used by interested users of the web application, professional or hobbyist web developers, and all stakeholders looking for detailed description of the design and any other party interested in this design.

Scope

The software product described in this document is a web application by the name of MUNSN (Memorial University of Newfoundland Social Network). The product will do the following:

- allow users that attend MUN to register an account
- create a profile that displays personal information
- allow users to communicate with other users by creating posts on profiles or groups
- connect users with other users in the form of virtual friendships
- Collaborate on course work in groups and polls
- Facilitate a lost and found system for physical possessions
- Modify privacy settings on profile information and content

The product will **not** do the following:

- Allow unregistered users to create posts

- Allow users to register an account with an email not associated with MUN
- Allow users to answer a poll of a user they are not friends with
- Allow users access to the phone number of a user, who posted a lost and found item, with whom they are not friends
- Allow user access to a user's resume or schedule if they are not friends with that user

The application of this product is to act as a social networking platform from which students of MUN can connect, form friendships, collaborate in study groups, and share personal information. The benefits of using this software product is enhancement of the interconnectedness of students at MUN which improves social interaction, ease of information sharing and work collaboration. Examples of software usage that provides these benefits includes:

- the formation of virtual study groups that allow students to collaborate on assignments and studying
- Creating posts in groups or profiles to initiate discussion and networking opportunities
- Creating and answering polls to attain data on predictions regarding possible exam questions

Definitions

- HTTP: an application protocol for distributed, collaborative, hypermedia information systems
- CSS: a style sheet language used for describing the presentation of a document written in a markup language
- MongoDB: an open-source, document database designed for ease of development and scaling
- Node.js: an event-driven I/O server-side JavaScript environment based on V8
- HTML: the standard markup language for creating web pages and web applications
- MUN: Memorial University of Newfoundland, the university from which the MUNSN website originates and contains members from
- MUNSN: Memorial University of Newfoundland Social Network, the name of the web application described by this document
- FRxxx - Labeling convention for functional requirements. FRxxx = Functional Requirement # xxx

References

- https://www.tutorialspoint.com/dbms/er_diagram_representation.htm
- https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model
- https://en.wikipedia.org/wiki/Non-functional_requirement
- http://www.requirements.com/glossary/nonfunctionalrequirements/tabid/91/default.aspx

Overview

The remainder of this document contains information on the description of the system being designed in regards to functional and nonfunctional requirements, use cases that describe the features of the system, and external interface requirements.

The specific requirements section details the functional requirements for the completed software product. It describes the anticipated inputs, processing of those inputs, and expected outputs for each feature. There is also a short description for each feature that indicates the priority level of implementing that particular feature.

The external interface requirements section follows the functional requirements section and describes how the product will interface with users, other software, and hardware. An example web page wireframe diagram is included for visualization of the user interface.

The non-functional requirements section describes the performance, security, and structure of databases for which an entity diagram can be found.

This document concludes with a list of use cases that expand on how the functional requirements will be used in terms of user interaction, inputs, error handling, and outputs.

Specific Requirements

Functional Requirements

FR001: Account Registration

Description and Priority

High priority - The system shall allow a user to create an account that uses a MUN email address. The user provides required personal and account information and upon completing the registration form an email is sent to the provided email address. In the email is a link that must be clicked by the user, and upon doing so the account will be successfully created and stored in the database. The user can then login and modify their account, and access the website's services.

Inputs

- Username
- Password
- Name
- Birthday
- Address
- Sex
- Profile picture
- MUN email address

Processing

- System shall accept user-provided MUN email address for registration (ex: dfcm15@mun.ca)
- Username shall be the MUN username (ex: dfcm15)
- System should reject the password if it does not contain at least the following:
    - One uppercase character
    - One number character
    - Minimum 8 characters

- System shall accept additional information that may be supplied by the user:
    - Name
    - Birthday
    - Address
    - Sex
- System shall be capable of accepting a profile picture from the user if they so choose to upload one upon registration

Outputs

- System shall send user a confirmation email after registration.
- Accounts shall be stored in a MongoDB database
- Accounts shall be persistent (accessible from different locations without losing functionality/data)
- Profile information associated with the account shall be updatable upon request of the user associated with the profile
- Profile is successfully created and activated when email link is clicked on
- Users start with no friends

FR002: Login

Description and Priority

High Priority - The website allows users to login to their registered accounts by providing correct credentials. Once logged in, the user can access a variety of services made available to registered accounts.

Inputs

- Username
- Password

Processing

- User credentials are entered into text fields
- User clicks login button
- System shall check credentials to find if a matching pair exists in database

Outputs

- User is logged in if the credential pair is found
- User is told to try new credentials if pair is not found

FR003: Profile and Group Content

Description and Priority

High Priority - For each account, there will be an associated profile that contains content such as polls, posts which can contain comments, pictures, and contact information. This content can be retrieved and modified after specific requests. Groups will exist in ways that are very similar to profiles and are therefore included in this feature description.

Inputs

- Request to view pictures
- Request to view comments

- Request to view polls
- Request to view profile/group information
- Request to upload/delete pictures
- Request to modify/create posts
- Request to modify/create comments
- Request to modify/create polls
- Request to modify/create profile/group information
- Request to view resume
- Request to modify/upload resume
- Request to view friends list
- Request to modify privacy options
- Request to join group/befriend

Processing

- Upon request to view content, the content will be searched for in the database using several pieces of information which include:
    - the id of the user attempting to access the data
    - the id of the user/group that owns the data being accessed.
    - the id of the content to be viewed
- Upon request to modify content, the content will be searched for in the database using several pieces of information which include:
    - the id of the user attempting to modify the data
    - the id of the user/group that owns the data being modified
    - The id of the content to be modified
- Upon request to create content, the content (either a post, poll, picture, etc) will be created and placed in the container (such as group or profile). Prior to doing so, the id of the user attempting to create and upload the data will be used to check to ensure they have rights to upload this content to the group or user profile.

Outputs

- If the user has rights to access the data, it will be delivered. Else, the data will not be shown.
    - Content will be displayed at the top of the profile/group from newest to oldest
- If the user has rights to modify the data, it will be modified. Else, the data will not be modifiable.
- If the user has rights to create and upload the data to a particular container, it will be possible to upload data to the container.

FR004: Group Management

Description and Priority

Medium Priority - Registered users can create study groups and have new members join either by invite or by setting the group to be open. Groups function in similar ways as the profiles in terms of content (please see Profile and Group Content). Users can join groups.

Inputs

- Request to create group
- Request to join group

- Request to modify group settings
- Group information
- Data for posts

Processing

- Upon request to create a group, the system shall ask for group information such as name and description and privacy settings (such as setting the group to private or public)
    - When the group is created, the system shall set the creating user as the group owner which implies they have administrative rights over the group and its members
- Upon request to join a group, the system shall check if the group is public or private.
    - If it is public, the user will join the group
    - If it is private, a request will be sent to the group owner for approval
    - If the owner or a member of the group (with rights to do so) has requested a user to join, the user will join the group upon accepting the request
- Upon request to modify the group settings, the system will obtain the user ID of the user attempting to modify the group settings and determine if the user is permitted to do so
    - If the user has access, they will be able to modify who can send invites, modify the group information, and modify the group members list
- Users can make posts in the group's timeline

Outputs

- Groups are created
- Users can join
- Invitation is sent to user
- Posts are created in the group

FR005: Friends

Description and Priority

Medium Priority - Users can befriend other users by sending requests to a desired user. The recipient user can then choose to deny or accept that request.

Inputs

- Friend request
- Reject friend request
- Accept friend request
- View profile
- Request to view friends on profile

Processing

- Users can search for friends (search function or Suggested Friends feature)
- The system shall permit users to request friendship from other user accounts
- The system shall notify the recipient user of a friend request when created
- The system shall permit the recipient user to accept or deny a friend request
- The system shall display user's friends on their profile

Outputs

- The system shall form a friendship between the two users
- Friends can view each other's timeline and information

FR006: Guests

Description and Priority

Medium Priority - Guests will be allowed to enter the site unregistered, but will have their privileges correspondingly reduced. Guests will only be allowed to view public groups and profiles.

Inputs

- Request to enter site as guest

Processing

- Shall system shall log in user through a guest account

Outputs

Users can view the site with limited privileges

FR007: Posts

Description and Priority

Medium Priority - Users can post in groups and other user's profiles

Inputs

- Text
- Hyperlinks
- Images
- Request to modify privacy options

Processing

- Temporarily hold text data in text field for user to edit
- When submit button is clicked, store post in database
- Privacy option modified if request made. Options include:
    - Only the poster can see the post
    - Everyone (including guests) can see the post
    - Only friends of the poster can view the post
    - Only a specified list of friends can see the published posts

Outputs

- Post data stored in database and linked to the containing user profile or group
- Posts viewable on group or profile
- Posts displayed on timeline from newest-top to oldest-bottom
    - A complete history of a user's post should be available on their timeline.
- User and applicable people (based on privacy options) can comment:
    - Comments can be erased
    - Editable
    - History of edits can be displayed
- User can disable comments on a post

FR008: Suggested Friends

Description and Priority

Low Priority - The system shall suggest potential new friends to the user based on several criteria.

Inputs

- User enters main control page

Processing

- System shall search for the user in the database to determine their list of courses, study groups, programs, and friends of friends
- Based on this criteria, the system shall create a small list of friends to add

Outputs

- The system will present this list to the user
- Suggested friends are displayed from most relevant-top, to least relevant-bottom
- The user can remove other people from appearing in their suggested friends feed. This will prevent that particular person from reappearing in the current user's suggested friends.

FR009: Lost & Found

Description and Priority

Low priority - Users will be able to post lost or found items to the page in a specific format. Users can post lost or found items in a specified area. The format of the posts will include photos, description, location and some method for other users to contact that user.

Inputs

- Data describing lost item
- Data describing found item

Processing

- The system shall post the notices of lost or found items in the lost and found using data given by user describing items

Outputs

- If the poster is a friend, they will have access to their phone number
- If not, the available information would be the poster's email
- The location of a posted "lost" or "found" item will be placed on a mini map of the area
- The posts will be posted to a timeline within the lost & found page which can be viewed by any user.
- The system can match a lost item post with a found item post using the information posted in each form.

FR010: Schedule/Calendar

Description and Priority

Low Priority - Users have their own schedule/calendar which is displayed on their profile. The schedule/calendar will contain information on the times of their classes and important dates such as exam dates.

Inputs

- Dates and hours of classes
- Dates of exams

- Any other event they wish to display

Processing

- System shall read in dates and hours and enter them into the appropriate cells in a table
- System shall update the schedule on the user's profile

Outputs

- Schedule is displayed as a calendar
- Events can make posts in timelines
- Friends can view schedules/calendars

FR011: Polls

Description and Priority

Low Priority - Polls will be used to obtain feedback from users regarding any question of the poster's choosing. Polls will contain

Inputs

- Questions for polls
- Answers for polls

Processing

- System shall store questions for the poll
- System shall store the poll and link it with the appropriate container (whether a user or a group)
- System shall store any answers to the poll and store what user created those answers

Outputs

- Poll created with questions available for users to answer
- Only friends of the creating user may answer the poll

FR012: Resumes

Description and Priority

Low Priority - Resumes can be contained in each user's profile should they choose to upload one. The resume will be viewable in-browser if the user accessing the resume is friends with the user who owns it.

Inputs

- Resume file/data
- Request to view resume

Processing

- The file will be uploaded to the user profile either in the form of a pdf or as an HTML file
- The system shall check whether the requesting user is friends with the owning user

Outputs

- The file will be stored in the database for retrieval once uploaded
- The file will be shown to the requesting user if they are friends with the owning user

FR013: Instant Messenger

Description and Priority

Low Priority - (Extra feature) Allow users to instantly message their friends in a chat box

Inputs

- Text message
- Expand/Minimize Chat box
- User to chat with

Processing

- The UI will expand or minimize the chat box every time it is clicked
- Text entered in a text field and submitted will be sent to the content server and stored, and then transmitted to the recipient user
- The UI will refresh the chat box with the conversation of the selected user

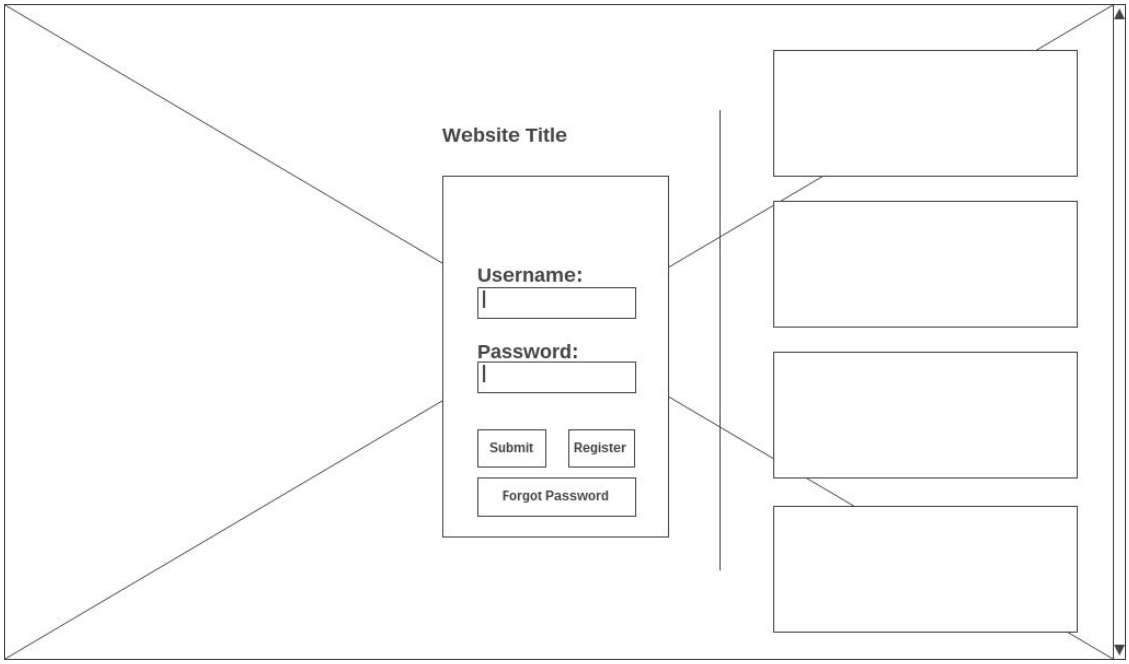
Outputs

- Conversation stored in the database for each user pair
- Conversation in chat box updated in real time with latest message

External Interface Requirements

User

All user interaction with the web page will be done through the HTML served to the users for each page. These pages will contain a set of standard input buttons and fields, which will be used for any basic functionality that is available on each page (e.g. Home button, Search field). Other, non-ubiquitous input fields will be made available as needed on each page. All output from these interactions will be displayed within the page itself.



Login Page Wireframe

Hardware

This software will be accessible using any device which has access to a web browser. Therefore, the web pages and style sheets shall be created to work properly in each possible case. Communication will use the standard HTTP protocols.

Software

This software will use MongoDB (v 3.2.11) for all database requirements. Node.js (v. 4.6.1) will be used to serve the web pages.

Communication

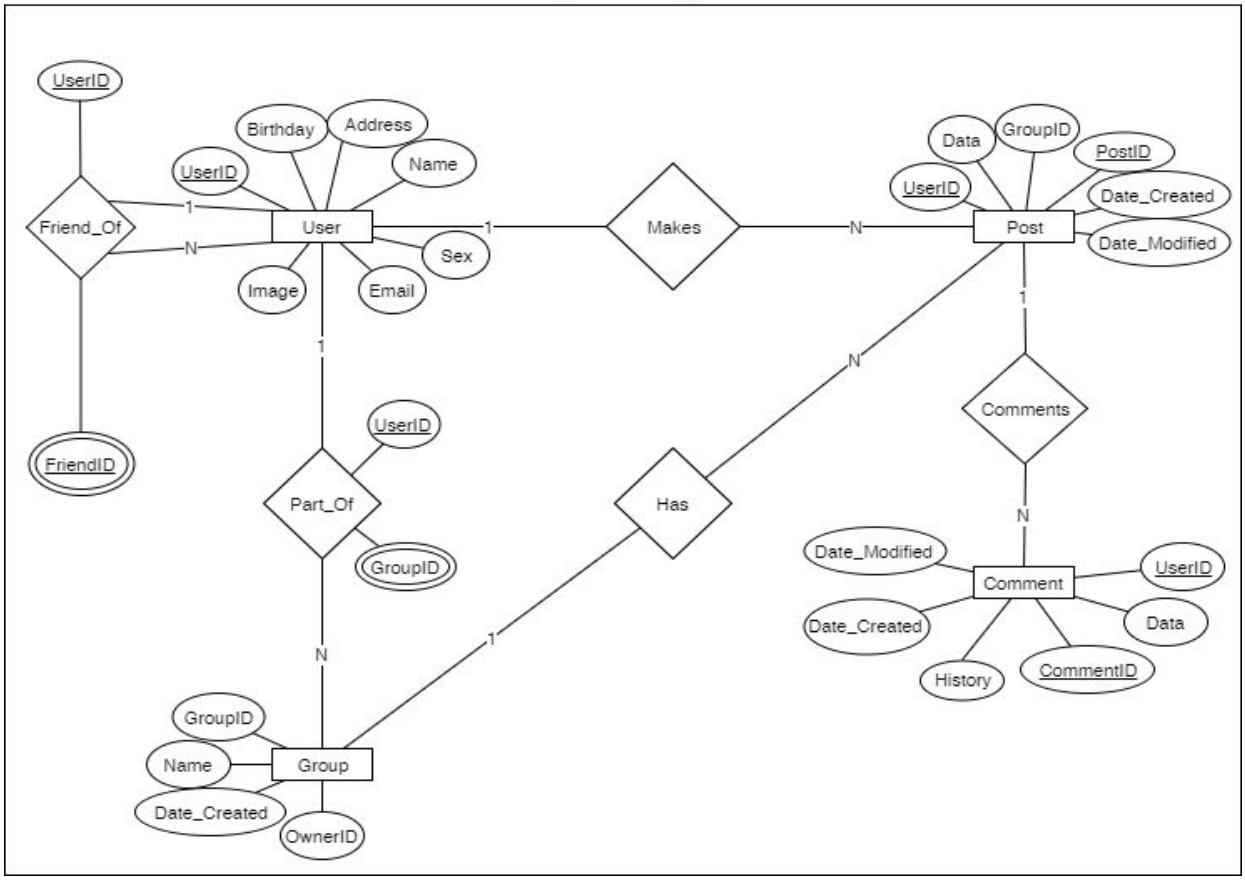
This software will use email servers to communicate important information to users (e.g. registration confirmation). Any web browser should be accepted, and all HTML, CSS, etc. should be formatted to follow this. In all cases, the software should use the HTTP protocol.

Non-functional Requirements

Data Structure

- Using MongoDB, content is stored in the database where it is accessed as a 3rd party server. It will store:
    - Lists of users
    - Lists of users with active sessions
    - Lists of groups
    - Pictures
    - Posts
    - Polls
    - User's friends lists
    - User account information
    - Group information

Entity Diagram



Entity Diagram

- HoganJS, a templating tool, is used to populate webpages in conjunction with the database to easily create web pages and display relevant information on them.

Performance

- The system will send out confirmation emails to users within quickly after account registration (no more than 5 minutes). This insures that users can use their account quickly after registering. This allows for less downtime.
- User activity will be displayed within 5 seconds in all relevant locations on the site, unless stated otherwise.
- Caching images and/or pages may be used to save the servers from bandwidth drain along with other possible performance savings.

Security

- All sensitive account information, (e.g. passwords or personal information) will be stored using encryption to insure they are protected.
- A secure socket layer connection will be established so that users are not suffering from man in the middle attacks and packet sniffers.
- New accounts created cannot be fully activated or permanently created until a confirmation email is sent and instructions are followed to activate an account.

- Due to the nature of the mun.ca domain protection against bots and account spam is unnecessary. The reasoning for this is that it is not possible to make multiple accounts through normal means.

Software Quality Attributes

- The system will automatically append the mail server to the username. It is required that all usernames match users mun.ca account.
- Lists of users will be truncated to a set number of users best matching the description. A larger list will be generated on user request.
- All functionality will be available at all times, unless otherwise stated. There should be no limitations to the time/location when some functionality may be used.

System Use Cases

Use Case UC0: Create Account

Scope: Website Homepage
Level: user-goal
Primary Actor: Guest
Stakeholders and Interests:
User - needs an account to access the system
Administrator - Authentication is required so that other users cannot abuse the system
Preconditions: User has a mun email with no prior account attached
Postconditions: User account is created
Trigger: Site visitor clicks on registration button
Main Success Scenario:
1. User accesses site
2. User to start a new account
3. User enters personal information
4. User types in their MUN email ID
5 User enters a password.
6. System sends a confirmation email to confirm successful creation and validation
Extensions:

2a. User uploads profile picture

Use Case UC1: Modify Profile

Scope: Website Userpage
Level: user-goal
Primary Actor: User
Stakeholders and Interests:
- User wants to modify account information
Preconditions: User has an account and is currently signed in.
Postconditions: Any changes are saved
Main Success Scenario:
1. User navigates to their profile page
2. User initiates modification
3. User makes any changes they want to make within the scope of the system
4. User confirms changes to the system

Use Case UC2: View Profile

Scope: Website Profile
Level: user-goal
Primary Actor: User
Stakeholders and Interest:
User - wants to view a profile
Preconditions: The profile exists
Postconditions: The profile still exists and is in good condition
Main Success Scenario:
1. The user navigates to a profile
2. System displays the profile to the user.
3. The user can view the profile

Extensions
1a. Via clicking on link from user search results
1b. Via clicking on link from suggested friends
1c. Via clicking on link from friends list

Use Case UC3: Add Friend

Scope: Website Profile
Level: user-goal
Primary Actor: User
Stakeholders and Interests:
- User wants to add a friend
- User has been asked to add a friend

Preconditions: Both users are part of the system
Postconditions: A request is sent to the requested user and any result is saved

Main Success Scenario:

1. User is on the profile page of a potential friend or other similar method
2. User sends a friend request via system interfaces
3. System sends request to the other user
4. Some result is received from the other user

Extensions:

4a. Recipient user rejects friend request

4b. Recipient user accepts friend request

4b1. User friends lists in database are updated to reflect connection

Use Case UC4: Suggest Friend

Scope: Website

Level: user-goal

Primary Actor: User

Stakeholders and Interests:

- Make connections with users

Preconditions: Both users exist and are related arbitrarily

Postconditions: Any results are saved

Main Success Scenario:

1. User gets a notification for a suggested friend

Extensions

1a. User adds suggested friend (see Add Friend Use case)

1b. User ignores friend suggestion

1c. User deletes friend suggestion

Use Case UC5: Search For User

Scope: Website Profile

Level: user-goal

Primary Actor: User

Stakeholders and Interests:

- Searching for known users

Preconditions: User may or may not exist

Postconditions: The existence of a user and results are given to the searcher

Main Success Scenario:

1. User has navigated to their profile page
2. User initiates search via search feature
3. User types the name of the other user to be searched
4. Results are displayed to the user

Use Case UC6: Logout

Scope: Website

Level: user-goal

Primary Actor: User
Stakeholders and interests:
- Logging out from the site for various reasons

Preconditions: User is logged in
Postconditions: User is logged out
Main Success Scenario:
1. User is presently viewing the site
2. User hits a logout button

Extensions
1a. User connection times out
    1a1. User is logged out

Use Case UC7: Create Group

Scope: Website User View
Level: user-goal
Primary Actor: User
Stakeholders and Interests:
- User wants to create a group

Preconditions: The user is registered
Postconditions: Group created
Main Success Scenario:
1. User accesses their user view page
2. User initiates group creation
3. User enters the appropriate information to create a group
4. User finishes entering information

Extensions
4a. Information insufficient
    4a1. User cancels group creation
4b. Information sufficient
    4b1. Group created

Use Case UC8: Create Resume

Scope: Website User View
Level: user-goal
Primary Actor: User
Stakeholders and Interests:
- User wants to create a resume for view by others
- Other Users want to review qualifications of the User

Preconditions: User exists, is registered, has no resume prior
Postconditions: User has created a resume that is saved
Main Success Scenario
1. User is on their profile control page
2. User initiates resume creation process

3. User fills out generic form
4. User submits form and a resume is created

Use Case UC9: Create Schedule

Scope: Website Userview
Level: user-goal
Primary Actor: User
Stakeholders and Interests:

- User wants to create a schedule for view by others
- Other Users may want to know the schedule of the user

Preconditions: User exists, is registered, has no prior schedule
Postconditions: Users schedule is saved
Main Success Scenario:

1. User is on their control page
2. User initiates a create schedule instance
3. User fills out a generic form
4. User submits the form and a schedule is created

Use Case UC10: Modify Schedule

Scope: Website Userview
Level: user-goal
Primary Actor: User
Stakeholders and Interests:

- User wants to modify an existing schedule
- Current schedule Information was incorrect

Preconditions: User exists, is registered, has a schedule to modify
Postconditions: Users schedule is modified and saved
Main Success Scenario

1. User is on their control page
2. User initiates a modification of an existing schedule
3. User changes information on the repopulated generic form
4. User submits the form and the schedule is updated

Use Case UC11: Modify Resume

Scope: Website User view
Level: user-goal
Primary Actor: User
Stakeholders and Interests:

- User wants to upload a new resume
- Information or qualifications changes to be imposed upon the resume

Preconditions: User exists, is registered

Postconditions: User's new resume is displayed

1. User is on their profile control page
2. User initiates an upload of a new resume
3. User submits the new document

Use Case UC12: Modify Group

Scope: Website Group view
Level: user-goal
Primary Actor: Group Admin
Stakeholders and Interests:

- Group Admin wants to modify a group

Preconditions: Group exists and user has control over group
Postconditions: Group still exists and changes are saved
Main Success Scenario:

1. Group admin accesses group view
2. Group admin starts modifying the group
3. Group admin makes changes to the group

Extensions
3a. Group admin saves changes to the group
3b. Group admin rejects changes to the group
    3b1. Group reverts to unchanged state

Use Case UC13: View Group

Scope: Website Profile
Level: user-goal
Primary Actor: User
Stakeholders and Interests:

- User wants to view a group

Preconditions: Group exists and is visible to the user
Postconditions: Group still exists
Main Success Scenario:

1. User is viewing their profile
2. User looks through their groups
3. User selects a group to view
4. The group displays to the user

Use Case UC14: Make Post

Scope: Website
Level: user-goal
Primary Actor: User
Stakeholders and Interests:

- User wants to make a new post.

Preconditions: User has permission to make a post in the group/profile.
Postconditions: Users post is saved to the relevant group/profile.
Main Success Scenario:

1. User accesses a page with commentable objects
2. User selects an object to make a comment on
3. User enters in a body/description for the comment
4. User submits the comment or cancels the comment.

Use Case UC15 Make Poll

Scope: Website Profile
Level: user-goal
Primary Actor: User
Stakeholders and Interests:

- User wants to make a poll for a course

Preconditions: User exists, is registered and has a course to make a poll about
Postconditions: Poll is created and relevantly visible
Main Success Scenario:

1. User wants to create poll
2. User initiates a generic poll creation form
3. User fills out the form
4. User submits the form
5. Form is created and displayed relevantly

Extensions
1a. User goes to group to create poll
1b. User goes to own profile to create poll

Use Case UC16: Use Poll

Scope: Website Profile
Level: user-goal
Primary Actor: User
Stakeholders and Interests:

- User wants to use a poll relevant to them

Preconditions: User exists, is registered and the poll is visible to the user
Postconditions: User input to the poll is updated and saved
Main Success Scenario:

1. User accesses an existing poll
2. User selects an option pertaining to the poll
3. User submits their decision
4. Poll is updated and results are displayed

Use Case UC17: Login

Scope: Website
Level: user-goal
Primary Actor: User
Stakeholders and Interests:

- User wants to begin session and access services available to their registered account

Preconditions: User exists and has registered account
Postconditions: User session activated and can access services available to their registered account
Main Success Scenario:

1. User enters log in page
2. User types username into username field
3. User types password into password field
4. User presses login button

Extensions
4a. Username and password match not found
4a1. User is notified and prompted to re-enter credentials
4b. Username and password match found
4b1. User logged in and account session active

Use Case UC18: Modify Account

Scope: Website Profile
Level: user-goal
Primary Actor: User
Stakeholders and Interests:

- User

Preconditions: User is logged in
Postconditions: User data is updated in the database
Main Success Scenario:

1. User enters account page
2. User enters password into prompt
3. User Modifies account settings
4. User saves changes

Extensions
2. Password is incorrect.
2a1. User is prompted to re-enter password

Use Case UC19: Instant Messenger

Scope: Website Profile
Level: user-goal

Primary Actor: User

Stakeholders and Interests:

- User pair

Preconditions: Users are friends

Postconditions: User pair conversation updated in content database

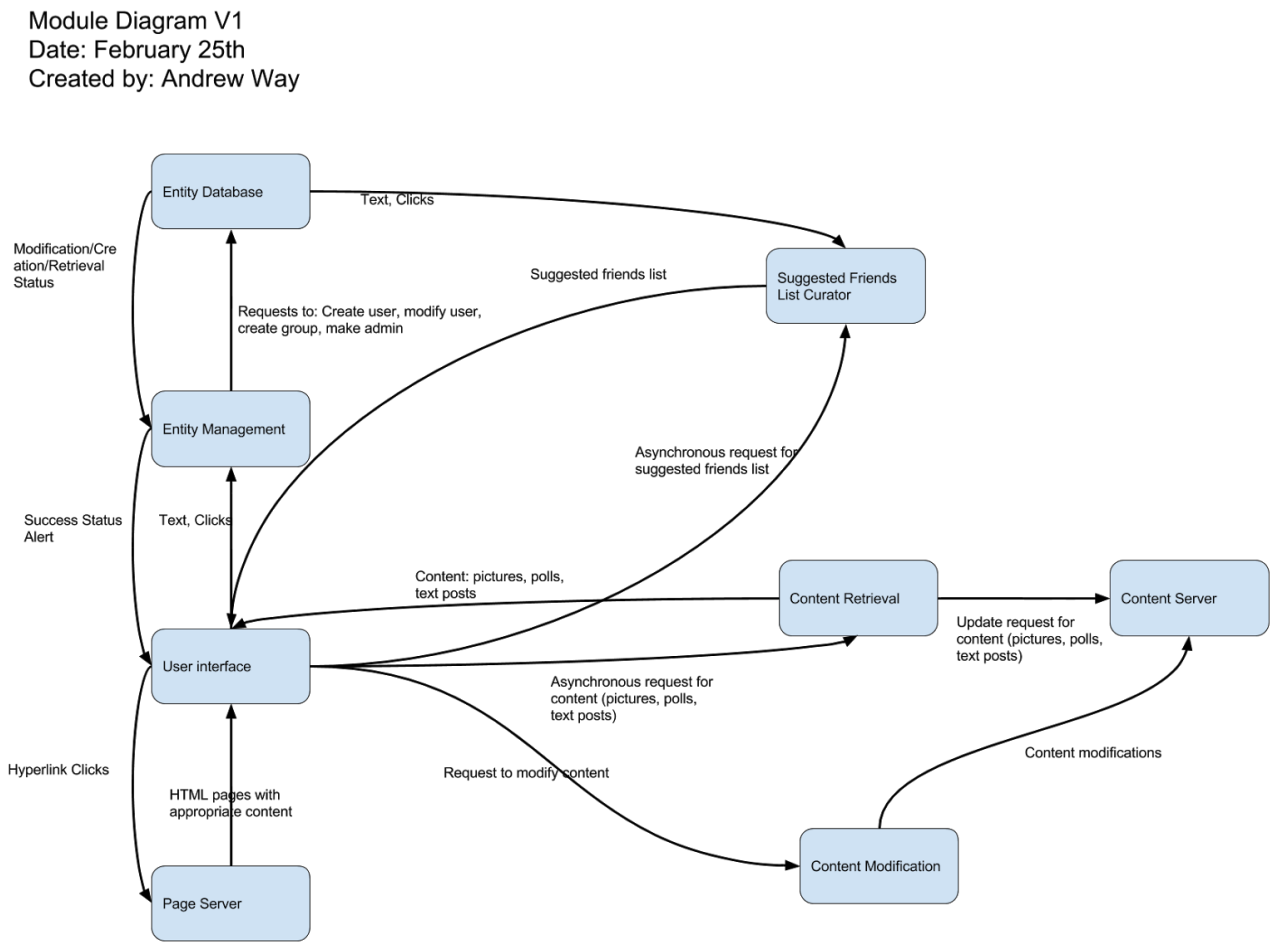
Main Success Scenario:

1. User opens chat box
2. User selects recipient user from option list/search bar
3. User types message into text field
4. User clicks “Send”
5. Text is saved into database

Extensions

5. Recipient user is logged in
    - 5a1. Recipient user receives chat notification
    - 5a2. Recipient user opens chat box
    - 5a3. Recipient user reads message

Architecture



Entity Management (Devin)

Description:

(Client-side JavaScript) Account creation, management, authentication, and modification associated with guests, registered users, group admins, and group members.

Relevant Classes

-

Inputs

- Username

- Password
- Email address
- Generic personal information
- Responses from user database

Processing

- Database requests to create account based on supplied information
- Database requests to modify account based on supplied information
- Database requests to create group admin based on supplied information
- Database requests to modify group admin based on supplied information

Outputs

- Update status

Module-Module Interfaces

- User Database
- User interface

Entity Database (Devin)

Description:

(MongoDB, Node.js) Storage, updating, retrieval, and back-to-front transfer of user data which includes lists of registered users and their associated profiles, lists of group admins and their associated groups, lists of members belonging to groups, lists of friends belonging to a user, and the personal data associated with each user

Relevant Classes

Inputs

- Request for friends list
- Request for user profile information
- Request for user account information
- Request for group information
- Request for group administrator information
- Request for group member information
- Request to form friendship
- Request to modify user profile information
- Request to modify user account information
- Request to modify group information
- Request to modify group administrator information
- Request to modify group member information
- Request to create account

- Request to create group

Processing

Outputs

1. User profile/user account/group/group administrator/group member/friend information
2. Modification success status
3. New user account created
    a. Verification e-mail
4. New group

Module-Module Interfaces

- User Access Management

User Interface (Kyle)

Description:

HTML and CSS files that comprise the collection of user-interactable input sources (buttons, text fields, hyperlinks)

Relevant Classes

Inputs

- Text
- Mouse clicks

Processing

- Modify interface according to client-side scripts following user interaction

Outputs

- Modified user interface

Module-Module Interfaces

- User Access
- Content Access

Page Server (John)

Description:

Respond to requests made from front end for HTML pages and their associated content

Relevant Classes

Inputs

- Clicks on hyperlinks

Processing

- HTML page and CSS referenced by click is retrieved
- Backend content supplied where necessary

Outputs

- Client browser is updated with retrieved HTML and CSS

Module-Module Interfaces

User Interface

Content Retrieval (Karl)

Description:

Handle client-side requests for new content such as pictures, polls, posts, comments, suggest friends lists by modifying the user interface and using data retrieved from the user database and content server

Relevant Classes

Inputs

- Text
- Clicks

Processing

- Run scripts initiated by user actions

Outputs

- Modify user interface with or without data retrieved from user database and content server

Module-Module Interfaces

- User interface
- Content Server
- User Database

Content Server (John)

Description:

Store, maintain, modify, and update photos, posts, comments, polls

Relevant Classes

Inputs

- Request for content
- Request to create/modify/update content

Processing

- Update the content
- Upload or save modified content

Outputs

- Requested content

Module-Module Interfaces

- Content Management
- Content Modification

Content Modification (Karl)

Description:

Modify content upon requests from the user interface

Relevant Classes

Inputs

- Request to modify content
- Content modification

Processing

- Modify content in content server

Outputs

- Content server modified
- User interface updated

Module-Module Interfaces

- User interface
- Content server

Suggested Friends List Curator (Devin)

Description:

Detect active user and determine list of suggested friends from user database based on several criteria

Relevant Classes

Inputs

- Main portal open

Processing

- User’s friend list is searched

Outputs

- Most common friends of friends is listed

Module-Module Interfaces

- User Interface
- User Database

Class Diagram