CS3300: Introduction to Software Engineering

Requirements for Team Chat - Web Client

Version 0.1: 6/1/15

Team: Web Client

Table of Contents

1. Introduction

1.1 Overview (Executive Summary)

Team Chat is an application providing intra-team communication for projects. It is a client-server application, with a REST API for web, smartphone, tablet, and smartwatch client-side applications. The application will allow users to send messages and share documents. Our team, within the Team Chat application, is the Web App team. In this document, the Web App team provides a description of the user requirements, user characteristics, and system requirements. Highlights include our user interfaces, system interfaces, functional requirements, and nonfunctional requirements.

1.2 Definitions and Acronyms

1. Web App - Web Application
2. UI - User Interface
3. MVC - Model View Controller
4. DOM - Document Object Model
5. CSS - Cascading Style Sheets
6. HTML - Hyper Text Markup Language
7. OOCSS - Object Oriented Cascading Style Sheets
8. SASS - Syntactically Awesome Style Sheets
9. npm - Node Package Manage

2. User Requirements

2.1 Software Interfaces

```
"angular": "latest",

"angular-ui-router": "latest"

"express": "^4.9.5",

"grunt": "^0.4.5",

"grunt-angular-templates": "^0.5.7",

"grunt-concurrent": "^1.0.0",

"grunt-contrib-clean": "^0.6.0",

"grunt-contrib-compass": "^1.0.1",

"grunt-contrib-compress": "^0.13.0",

"grunt-contrib-concat": "^0.5.1",

"grunt-contrib-copy": "^0.6.0",

"grunt-contrib-cssmin": "^0.12.2",

"grunt-contrib-uglify": "^0.9.1",
"grunt-env": "^0.4.1",
"grunt-express": "^1.4.1",
"grunt-filerev": "^2.3.0",
"grunt-http-upload": "^0.1.8",
"grunt-ng-annotate": "^0.10.0",
"grunt-preprocess": "^4.0.0",
"grunt-usemin": "^3.0.0",
"matchdep": "^0.3.0"
"http_parser": "^0.1.8",
"node": "0.10.26",
"v8": "3.14.5.9",
"ares": "1.9.0-DEV",
"uv": "0.10.25",
"openssl": "1.0.1e"
"npm": "1.4.3,
```

2.2 User Interfaces

The web client will be accessed through any web browser, and there will be multiple views set up that correspond to their respective functionalities. A user can navigate to the views designed for message sending/receiving, managing his/her groups, user searching, document sharing, and possibly managing user profiles. To accommodate different screen sizes and resolutions, we are aiming to make a responsive UI.

2.3 Product Functions

1. As a user, I want to be able to log in
2. As a user, I want to be able to search for another user
3. As a user, I want to be able to select a user and send him/her a message
4. As a user, I want to be able to see previous messages and reply
5. As a user, I want to be able to create a group and edit its members
6. As a user, I want to be able to send messages to a group
7. As a user, I want to be able to upload/share documents in a chat
8. As a user, I want to be able to view documents from a chat
9. As a user, I want to be able to delete one of my groups
10. As a user, I want to be able to remove someone from my group
11. As a user, I want to be notified when I receive a message
12. As a user, I want to be notified when I receive a group invite
13. As a user, I want to be able to edit my profile

2.4 User Characteristics

As the designers, we do not expect the users to have any prior experience with chat-based apps. We plan on laying out each web page to be user-friendly for both new users and the more tech savvy users. However, we do expect the users' ages to be around the high school to college range. Since we are going for ease-of-use when planning our designs, accommodating all users will be pretty straightforward. We can avoid using overly technical jargon, and design each page to be simple, yet concise.

2.5 Assumptions and Dependencies

1. If the REST API decides to make major changes, our front-end would require major refactoring.
2. Dependent on the REST API for back-end processing. If we are blocked somehow, we may need to implement a temporary endpoint for testing data.
3. If the overall app's requirements change, our front-end would require major refactoring.
4. Development effort dependent on other priorities (course work, exams, projects, etc).

2.6 Apportioning of Requirements

Sprint 1

- Be able to log in
- Be able to search for another user
- Be able to select a user and send him/her a message
- Be able to see previous messages and reply
- Be able to create a group and edit its members
- Be able to send messages to a group

Sprint 2

- Be able to upload/share documents in a chat
- Be able to view documents from a chat
- Be able to delete one of my groups
- Be able to remove someone from my group

Sprint 3

- Be notified when I receive a message
- Be notified when I receive a group invite
- Be able to edit my profile

3. System Requirements

3.1 Functional Requirements

- The system must allow users to log into the system with a unique "username" or register a new account with a username.
- Each user account must have their legal name associated with a username. Username can be the user's full legal name.

- The system may allow users to log into the system with a password.
- The system must allow users, once logged in, to search for team members by legal name.
- The system must allow users to be able to send messages to another user.
- The system must allow users to see messages sent to them.
- The system must allow users to create a group of users.
- The system must allow users to add users to a group.
- The system must allow users to see the groups they are registered in.
- The system must allow users to remove members from a group.
- The system must allow users to delete groups.
- The system must allow users to be able to send messages to a group they belong to.
- The system must allow users to be able to see messages sent to them from other users in any group they are a member of.
- The system must allow users to send files to other users or any group they belong to.
- The system must allow users to view any files sent in a message.
- The system must allow users to download any files sent in a message.
- The system must notify the user of any messages sent to them while the page is in focus with a audible or visual stimulus.
- The system must notify the user of any messages sent to them while the page is not in focus with an audible stimulus.
- The user may be allowed to create a profile and change profile settings.

3.2 Non-Functional Requirements

- The system must integrate with a REST API.
- Users must be able to create groups and send messages without a formal course within 1 minute of logging into the application for the first time.
- Team Chat web client will be reachable via any browser.
- The system will communicate with the clients via HTTP.
- The system will respond with appropriate status codes at each endpoint.

3.2.1 Software Quality Attributes

Compatibility	The Web Client must be compatible with the REST API and, therefore, the other clients. Otherwise, the system will be useless.
Dependability	The Web Client must be available to users and reliable for users so that whenever they want to communicate with their team, they can.
Effective	The Web Client must perform the tasks the user expects, otherwise it will be useless.
Responsive	The Web Client should be seamless and fluid for users so that is enjoyable to use.
Simplicity	The Web Client should be simple to use and have a simplicity of design to make the application enjoyable and aesthetic.
Understandability	The services that the Web Client offers should be easily understood. This way, the Web Client team will not have to train users and the application will be self evident.

Usability	Similar to Understandability, the application should be easily used, requiring no training.