CS3300: Introduction to Software Engineering

Design Document for Team Chat

Version 0.0: 6/4/2015

Team: Web Client

Contents

1 Introduction

1.1 Overview (Executive Summary)

Team Chat will use an MVC architecture, and it will be backed by several different resources ideal for MVC applications. We specifically chose AngularJS, which is a javascript framework that enhances dynamic web applications built over MVC. The web client will have views designated for user authentication, group management, chat organization, and the actual chat rooms. In this document, the Web Client team provides descriptions for our design considerations, diagrams representing our class structure, and some quick mockups of our user interface.

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

1.3

- **general debugging - www.stackoverflow.com**
- grunt - http://gruntjs.com/api/grunt
- angularJS - www.angularjs.org
- npm - www.npmjs.com
- bower - www.bower.io
- sass - http://sass-lang.com/documentation/file.SASS_REFERENCE.html
- express - http://expressjs.com/

2 Design Considerations

2.1 Assumptions

As the Web Client team, we are assuming the REST API will be there when we enter the integration phase for the project. As of right now, we are designing our views based on the given requirements. They directly affect what functionality we will offer and how our different components will interact using our MVC architecture.

2.2 Constraints

- Responsive
- Cross Browser Compatibility
- Server Availability
- REST API Integration
- Understandable/useable

2.3 System Environment

Team Chat is designed to be accessible through Firefox, Chrome, Safari, and IE9+. It will not matter whether the user is using Windows, Mac, Linux, iOS, or Android. Team Chat will be available for desktop, laptop, tablet, and mobile use.

3 Architectural (High-Level) Design

3.1 Overview

The architecture for our system is Model View Controller (MVC). Our web application will communicate through a REST API with the back end. The back end and the REST API will be handled by the server team. Using their API, we will exchange commands to get, post, push, and delete data. Whatever changes are made in the back end through our web client will be reflected in the web client. Any changes in the back end will eventually be reflected in the view.

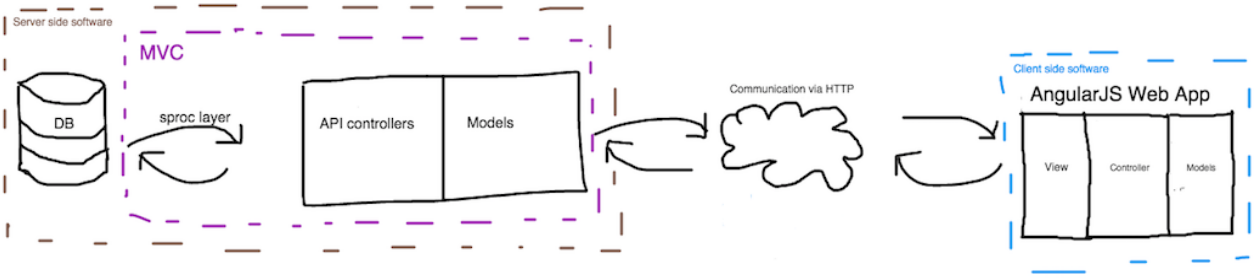
Because we are using AngularJS as our javascript framework, our MVC is a one off of the traditional MVC architecture. Because the Model and the View are tightly coupled in AngularJS, any changes in the view are reflected in the model, and any changes in the model are reflected in the view. The Controllers only handle business logic needed by the view.

3.2 Rationale

Within the Team Chat application, we, the Web App team, have chosen to use an MVC architecture. This is because it is a commonly used architecture in web application design. This means that there are many resources, tools, and languages tailored toward MVC. Specifically, AngularJS, the

framework we've chosen, is a structural framework for dynamic web applications that utilizes a version of MVC, albeit a tightly coupled view and model. We think that this architecture will not only enable our application to be simple and responsive, but will make our software simpler and more maintainable.

3.3 Conceptual (or Logical) View

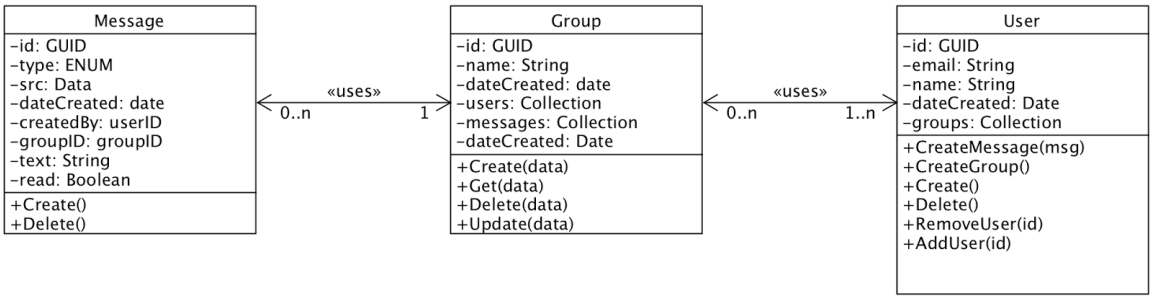


4 Low Level Design

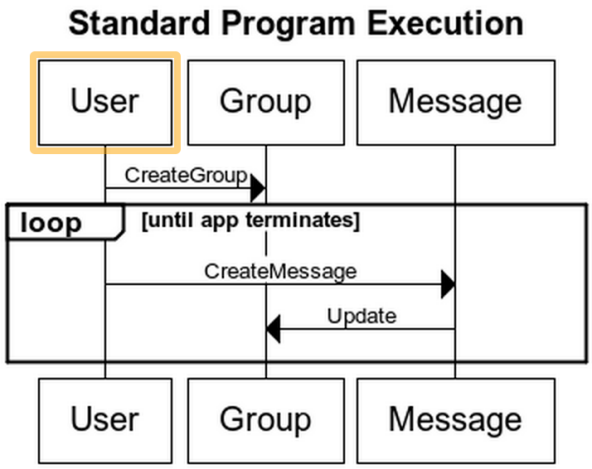
4.1 Overview

Due to the nature of AngularJS, certain properties of our design are difficult to model such as visibility of data fields. For example in Angular, the private and public properties are abstracted away from the programmer. For our purposes in UML modeling, we simply set all data fields to private and all methods to public in our class diagr

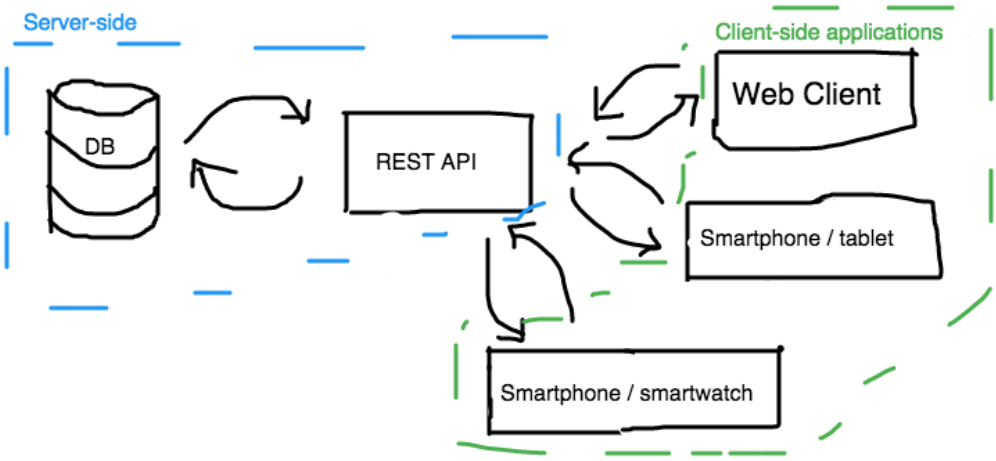
4.2 Class Diagram



4.3 Sequence Diagram



4.4 Component Diagram



5 User Interface Design