CS3300: Introduction to Software Engineering

Design Document for CS 3300 Messaging App

Version 0.0: 06-04-2015

Team: API Team

Contents

IMPORTANT:

The paragraphs written in the "Comment" style are for the benefit of the person writing the document and should be removed before the document is finalized.

Also, don't forget to update the Table of contents before submission. (Right Click -> Update Field)

1

Introduction

The introduction section provides context for the project

1.1 Overview (Executive Summary)

This document outlines the requirements necessary to host a RESTful API for a real-time, messaging application supported by four clients. If the reader is unfamiliar with web servers, refer to section 1.2 for acronym definitions. Our service will provide the back-end support for the web, Android, and smart-watch clients. The API being developed will give clients access to a web service that can store or transfer their information and uploads. In addition, it will support the integration of four interactive bots. The messaging application will rely on the API for managing the information it needs in order to run. The architectural properties of a RESTful API will provide clients with an easy to use system for naming their resources. No server-specific methods will be needed to refer to the server's resources. This assures that any client that uses HTTP can communicate with our service.

1.2 Definitions and Acronyms

HTTP: Hypertext transfer Protocol
RESTful: Service that accepts http requests (Representational State Transfer)
GET: Requests a resource from the server. (Web page or image)
POST: Similar to get but places additional info is sent in the body of the request.
PUT: Upload a resource to the server
DELETE: Delete a resource from the server
Spring: Java Developmental Framework
IntelliJ: Interactive development environment (Will be used with Spring)
Maven: Software project management tool
Jackson https://github.com/FasterXML/jackson : Java JSON parsing library
URI: Uniform Resource Identifier
status codes (Ex: 404, 500) - responses to network requests
JSON: Javascript Object Notation.
API: Application programming interface. Specifies how software components should interact
Slack: The communication method set by the team. This application will help us organize team tasks and documents.
GroupMe: The communication method set by the team. GroupMe is an application available on mobile and web platforms.

1.3 References

List any references or related materials here. For instance, if the design required interfacing with an X10 hardware devices, then you would want a reference the X10 specification.
In this section:

(1) *Provide a complete list of all documents referenced elsewhere in the Design Document*
(2) *Identify the document by title, report number (if applicable), date, and publishing organization*
(3) *Specify the sources from which the references can be obtained*

This information can be provided by reference to an appendix or to another document

2 Design Considerations

These subsections describe issues that need to be addressed or resolved prior to or while completing the design as well as issues that may influence the design process.

1
2

2.1 Assumptions

Describe any assumption, background, or dependencies of the software, its use, the operational environment, or significant project issues. These are things that you are assuming to be true and that directly affect the design

The system was designed with the assumption that users will be able to send and receive JSON requests and responses. The use of JSON format to transfer resources will require a class diagram to be provided to the clients. This will define the classes, and inform the clients of class attributes. The class attributes will define the resource representation to be used by its JSON format. If a team cannot send/receive with the JSON format we will need to provide a different data format option such as CSV.

2.2 Constraints

Describe any constraints on the system that have a significant impact on the design of the system. (e.g., technology constraints, performance requirements, end user characteristics) These are things the customer has told you that directly influence the design (e.g., the DB must be an open-source, freely available DBMS).

- System must always be running
- The system must be able to accept and fulfill requests a from web client, an android phone, and android tablet, and an android watch.
- The system must able to accept and fulfill requests from several different sources at the same time

2.3 System Environment

Describe the hardware and software that the system must operate in and interact with.

3 Architectural (High-Level) Design

The architecture provides the top-level design view of a system and provides a basis for more detailed design work. These subsections describe the top-level components of the system you are building and their relationships. For an OO implementation in Java, for example, our components could become packages (or set of packages, depending on the level of granularity considered and the size of the system).
In defining your architectural design, you can follow one of the organizational styles seen in class (shared data repository, shared services and servers, and abstract machine/layered) or pick a different one if none of those is appropriate for your system.

1 REST
2
3

3.1 Overview

This section provides a high level overview of the structural and functional decomposition of the system. The section should list the different components and concisely discuss the major responsibilities and roles such components must play.

The system must utilize multiple software components in order to host a RESTful API. The main framework will be provided by Spring. The use of Spring's libraries will be used in the project's source code. Spring will be used with Maven in order to build the project and deploy it to the server. The server will be hosted by Amazon web service and by using the RESTful API, clients can access resources through our database. MySQL will be used to provide the database component of the system. By using HTTP protocol, clients will be able to manipulate resources hosted in the MySQL database.

3.2 Rationale

This section discusses why you are using the architecture you have chosen.

Architectural design must be considered in order to build a server that can handle its client's needs. One feature that defines REST is its uniform interface between clients. Or more simply, a uniform way for clients to find the URIs that identify resources. These resources are referenced by URL and HTTP protocol is used to manipulate them. Resources can be represented by different formats. Since our client developers are familiar with JSON, our service will use this as its format. JSON will provide a simple way to read, write and parse the resources. Through this implementation, the system as whole is simplified.

3.3 Conceptual (or Logical) View

This section should provide and describe a diagram that shows the various components and how they are connected. The conceptual view shows the logical/functional components of the system, where each component represents a cluster of related functionality. For UML, this would typically be a component diagram or a package diagram.

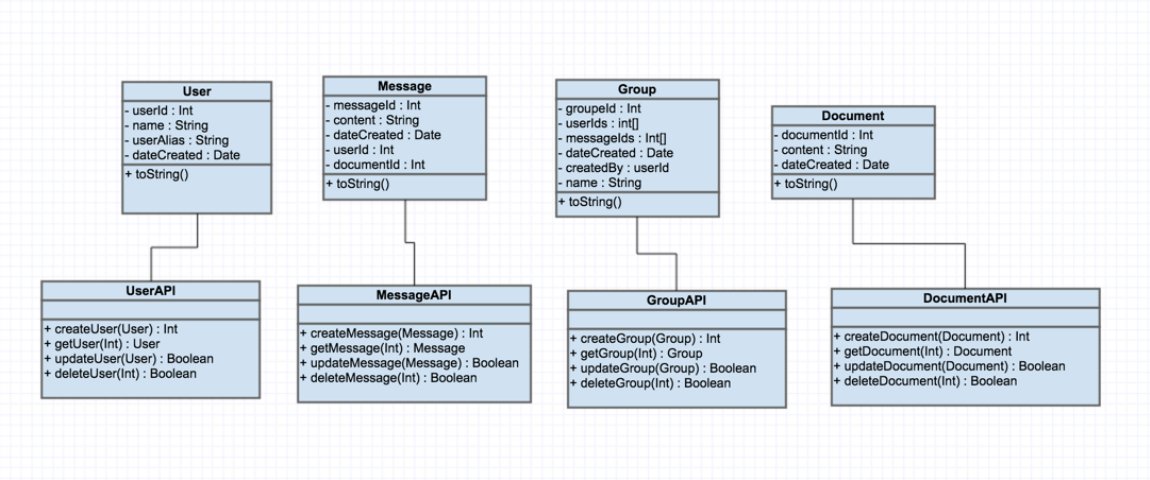
4 Low Level Design

This section provides the low-level design for each of the system components identified in the previous section. For your component, you should provide details in the following UML diagrams to show its internal structure.

1
2
3
4

4.1 Class Diagram

In the case of an OO design, the internal structure of a software component would typically be expressed as an UML class diagram that represents the static class structure for the component. Along with the system classes, you need to show their attributes and relationship between the classes (Association, Aggregation, Generalization, Dependency, Multiplicity etc.)



4.2 Sequence Diagram

This diagram shows details of how objects in your software component interact with each other and in what order.

4.3 Component Diagram

In this diagram you would show the other components your software component interacts with. For each external component, list the interfaces that you share with it.

If needed, you can show additional diagrams like activity & state diagrams.

5 User Interface Design

User Interface is the API Documentation located in our README.md
URL : https://github.gatech.edu/cs3300/p1server/blob/master/README.md