SOFTWARE REQUIREMENTS SPECIFICATION

for

Yabber

Prepared by Aravind Vasudevan

April 22, 2017

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for the "Yabber" chat application. It will illustrate the complete declaration for the development of the system. This document will describe the requirements and use cases of the system.

1.2 Document Conventions

Term	Definition
Student / User	Primary user of the application who chats in the user site
Admin	User capable of adding, removing and modifying details of students in the database.
System	Refers to the chat application.
MVC	Model-View-Controller
emoji	icon used to express an idea or emotion in electronic communication.

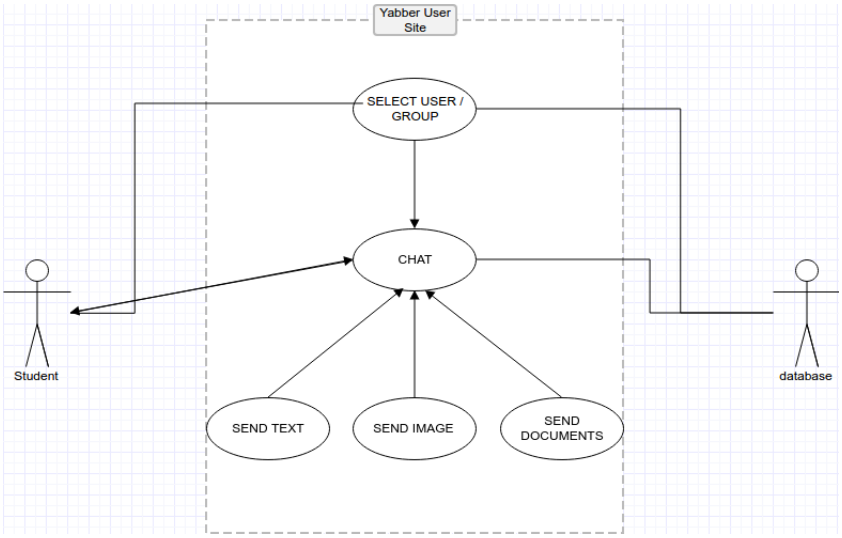
1.3 Project Scope

"Yabber" is a LAN based chat application that is meant to be used within a College. It allows students to log in with their register number and chat with their peers and staffs. They can also chat with emojis, share images, and other documents types within the application and also create groups with their peers and staffs. The college management can manage the student accounts using the admin panel. The panel has options to add, remove and modify student details.

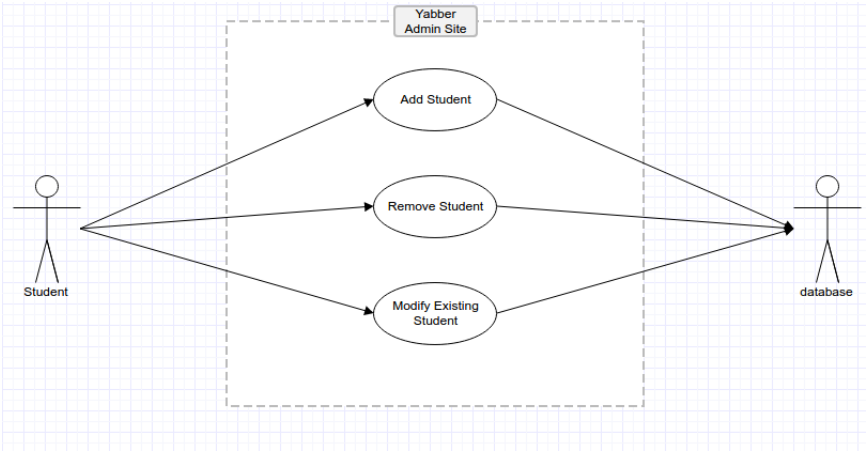
The application will be hosted in the college's local server and can be accessed throughout the college using a web browser in the connected systems.

2 Overall Description

The application consists of two parts: User Site and Admin Site.



The User Site is the part which the logged in user accesses. In this site, the user will be able to select another user or a group to communicate. The chat column will be updated with the user selected chat and the user can send and view messages, images and documents. The user will be provided with a emoji keyboard to share non-alphanumeric characters in the chat.



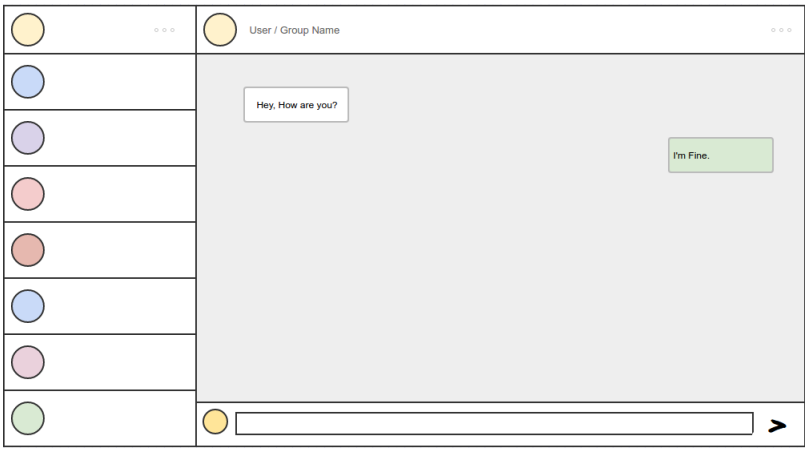
The Admin Site is the part which the admin user(College Management) accessses. In admin panel, the admin can add a new student to the entry, remove a student or modify an existing student's details.

3 External Interface Requirements

3.1 User Interfaces

When a user opens the client side with their browser, the login page appears. In this interface, the user can enter their register number and password to enter to the chat application.

The next interface on the client side would be the chat interface. Here, the user can create groups, select group / other user and chat with them. When the user clicks on the list item on the left side menu, the chat column on the right side is updated with that chat's controls. The user can send emojis using the smiley icon left of the input box and upload images and other document using either the hamburger menu on the top right or by using the right-click context menu.



The admin panel consists of an Admin login interface using which the college management can login into the admin panel.

The logged in admin user can add a new student using the green plus button on the top right side and remove a student using the red minus sign next to the student's name. The admin can modify details about the student by double clicking the student's listing.

3.2 Hardware Interfaces

All the computers that access the server should be connected to the local area network and the server side of the application should be run on the LAN's server. No other special hardware requirement is needed for the application to run.

3.3 Software Interfaces

The application uses node.js with express framework for the backend, uses Redis for caching, uses MongoDB to store the user information and chat history and uses gulp for build automation. The frontend is built using Angular 2, SASS and ng-bootstrap. Hence, Node.js, Redis, MongoDB should be installed on the server. Gulp and Angular 2 should be installed as global dependencies from npm.

3.4 Communications Interfaces

The application's client can be accessed in a browser using 'http' protocol. The messages are sent and received using web sockets and hence making the chat fast compared to timed refreshing or long polling.

4 Functional Requirements

- The system shall provide login page for the user.
- The system shall list all the users registered in the left column when logged in.
- The system shall load the chat history and controls when a user from left column is selected.
- The system shall provide a text area to type and send messages.
- The system shall provide a context menu and a hamburger menu which provides options to upload images and documents.
- The system shall play a notification sound when a new message arrives.
- The system shall trigger a notification using Notification API when a message arrives and the tab is out of focus.
- The system shall identify links in messages and convert them to anchor tag.
- The system shall make the uploaded images enlargeable.

- The system shall replace the Unicode emoji with the image link before saving.
- The system shall provide login interface for admin.
- The system shall provide the option to add users to the database.
- The system shall provide the option to remove users from the database.
- The system shall provide the option to modify users information in the database.

5 Nonfunctional Requirements

5.1 Performance Requirements

Since this is a chat application, we need the server to respond as quickly as possible and hence we need to host the server side in a high powered server connected to the LAN.

5.2 Security Requirements

Since this is a public application, the password must be encrypted and stored. The password can only be changed by the admin to protect it getting manipulated by malicious users.

5.3 Portability Requirements

Since this is a web application, it should be accessible throughout the network and the server-side application should be easily portable to another server.

6 Future Scope

The application will be built using MVC architectural paradigm and hence, is easily upgradable in the future. Most features can be added incrementally to the application as modules.