William Trefethen

Talk

Requirements Document

Table of Contents

1. Introduction

This document outlines the requirements of Talk, a realtime chat application for desktop. Talk is written entirely in Java and is composed of a server-side application and a user friendly client-side application.

1.1. Purpose and Scope

The purpose of this document will be to outline in detail the scope and requirements of the Talk Project. This will originate from a business use case perspective and develop into fully detailed maps of requirements for each of those use cases.

1.2. Target Audience

The target audience of this document is any stakeholders, potential investors, and any developers working on the Talk Project. Because of this, the language will be slightly technical but geared toward the correct understanding of developers and stakeholders alike.

1.3. Terms and Definitions

- Talk; Talk Project; Talk Application - the name of the program being developed.
- GUI; UI; UX - Terms referring to the graphical user interface that will be featured on the client side of Talk.
- Server-Side / Client-Side - terms referring to which part of the application will perform a certain action
- User; Client - anyone using the client-side application; a user of the program.

2. Product Overview

Talk is a realtime chat application written in Java. Its main feature is a global chatroom between online users. It additionally features direct messaging between users, a basic user profile system, and server-side chat history. To support its main features it also requires a user friendly UI which allows users to login, logout, register, direct message, chat to the global message board, and receive login notifications regarding other users.

2.1. Users and Stakeholders

The purpose of this section is to expose the various stakeholders and users' roles in this project and to outline what work has been done thus far to describe the product's features and requirements. The two major parties involved are the client and the developer.

2.1.1. Client

The client is the party requesting that the chat application be built. The initial requirements of the application were first outlined by the client as shown below:

This chat application consists of a chat server and an indefinite number of chat clients (users). The server accepts connections from the clients and delivers all messages from each client to other clients or delivers messages from one client to a specific client if a user wants. This is a tool to communicate with other people over Internet in real time.
A client must communicate with the server and, simultaneously, it must be ready to read messages from the standard input to be sent and accept messages from other clients. Users must log in first before they can communicate with the server or other users. After they enter their accounts and passwords, the application will check the validity of the information. If it is not valid, users will get the error message that either the account doesn't exist or its password doesn't match. If a new user wants to use this application, he/she must register first. Each user's information, such as account and password, is

stored by server. In addition, the server should save all the records of chat for each user. If they want, users can access their own records.

The server informs all online users when someone logs in or logs out. And when a user logs in, he/she can see all online users with their names highlighted. If a user wants to communicate with someone else, he/she just need to double click the name. Otherwise, the message sent by the user can be seen by all other online users.

The user interface is to be user-friendly.

2.1.2. Developer

The developer, William Trefethen, is responsible for building the application. In an attempt to formalize the client's initial requirements, this document was produced by the developer. The initial formalization of the requirements is in the form of basic use cases for the application as outlined below.

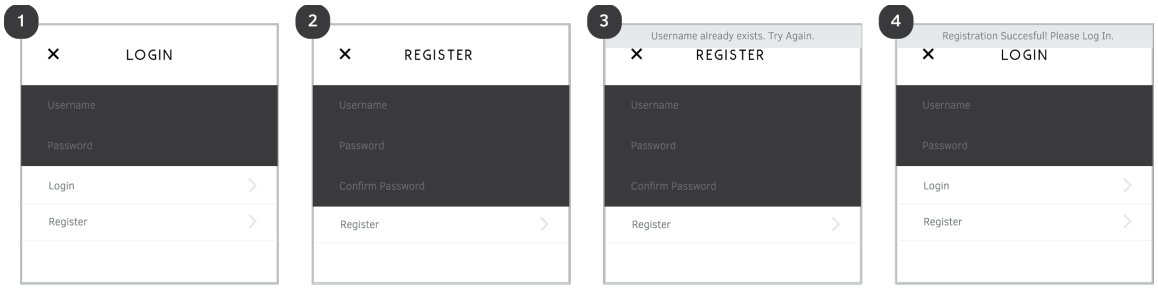
2.2. Use cases

There are three major use cases for the Talk application. First - when a new user uses the application, they must register. Second, when a returning user begins, they must login. Third, when a user is logged in, they must use the application to chat with others. The following outlines in detail these three major use cases.

2.2.1. Register

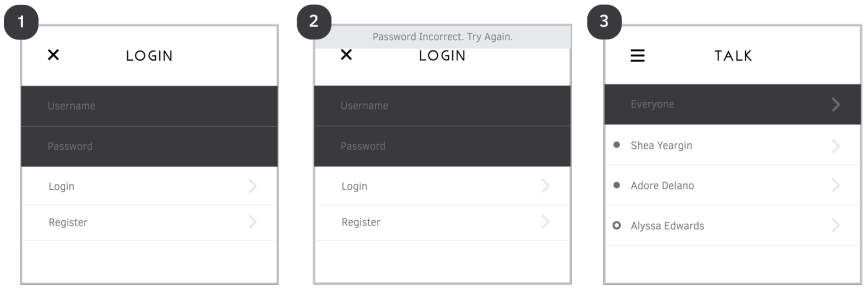
1. Upon opening the application for use, the user is first prompted to login at the **Login Screen**.
2. The user does not have an account, so selected the Register button and is taken to the **Register Screen**.
3. Upon successfully completing the register form with a matching password confirmation (validated client-side), the server is sent the registration request. If the username already exists, the server responds with a reject. This will notify the user that the username is taken.

4. Upon Successful registration the user is brought back to the **Login Screen** which is the starting point of the login use case (2.2.2)



2.2.2. Login

1. Upon opening the application for use, or after a successful registration, the user is first prompted to login at the **Login Screen**.
2. Login data is sent to the server, which either verifies the login credentials, or denies for either an unrecognized username or an incorrect password. If the login is not accepted, a notification appears to notify the user that the login has failed.
3. Upon logging in, the user is brought to the **Main Menu Screen** which is the starting point for any chat-related use cases. Simultaneously, the server relays to all other online clients that the user has logged in causing a notification.

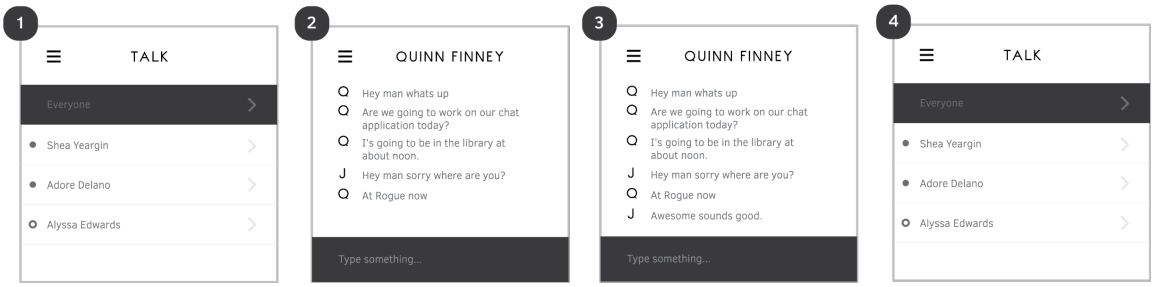


2.2.3. Chat

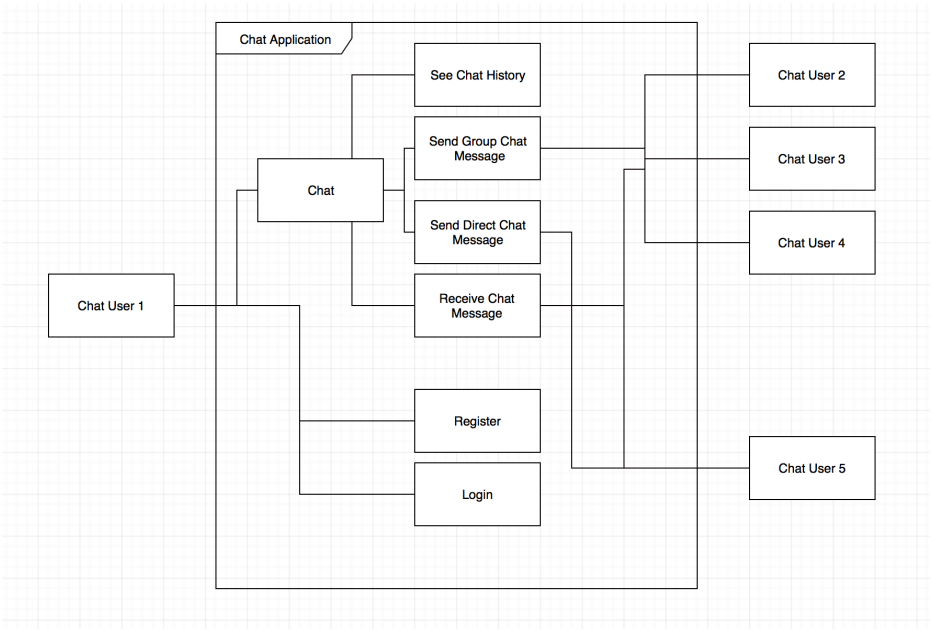
1. Upon successful login, the user begins at the **Main Menu Screen**. They have the option to click on “Everyone” or select a name from a list of all users (online or offline).
2. Upon selecting “Everyone” or another user’s name, the user is brought to the **Chat Screen** - a simple interface showing past and current messages in the selected chat

room and a text field in which to type messages. A packet is sent to the server to notify it that the user is now receiving messages from this chat room. The server the responds with the entire chat-room history with which to populate the chat screen.

3. Upon submitting a non-empty message (validated client-side) into the text field, the message is sent to the server which then relays the message to all the online clients in the chatroom, and saves the message to the chatroom's history on the server side.
4. Upon Exiting the chat room by pressing the menu button, the user is taken back to the **Main Menu Screen** and the server is notified that this user is no longer receiving messages from this chat room.



Below is a more detailed use case diagram outlining all the major and sub use cases of the application.



3. Functional Requirements

The functional requirements of the application outline mainly *what* the application must do rather than *how*. The following represents a **comprehensive** outline of the requirements initially outlined by the client.

3.1. User Account System

The user account system of Talk is a standard, basic user account system. In short, it requires a username and password account which can be created and logged in to.

3.1.1. User Registration

It is required that the application allows users to create anew account. This account will require a username and password. The newly created account must be saved server-side.

3.1.2. User Login

It is required that users with accounts can log into their accounts using the previously created username and password. Specifically it is required that this screen validate the login credentials server-side, and notify the user upon a failed login that either their password was incorrect or their username does not exist.

3.2. Chat System

Two major types of chat are required: global chat, and direct messaging. It is required that both of these types of chats can be initiated and that their history be accessible by the user.

3.2.1. Chat Types

It is required that users be able to initiate a chat with any other user by double clicking on that user's name in a list. Otherwise, the program's primary behavior should be to allow the user to message directly into a global chat room. The user should be able to switch between these various chat rooms.

3.2.2. Chat History

It is required that users are able to access the history of their chats in both the direct message and the global chat context. This requires the server save chat histories between users one to one as well as the global chat history, and that those histories can be sent to the client for viewing.

3.3. UI and Notifications System

Various notifications and UI features were also outlined in the client's requirements.

3.3.1. Login Notifications and Highlighting

It is required that when a remote user logs in, all other online users receive some form of notification that this event has taken place. Additionally, for all online users, their name must be highlighted in a list of users to denote this status.

3.3.2. Login Error Messages

It is explicitly required that the login screen have two distinct error messages - one for incorrect password and one for unrecognized username. These notification are to pop up as error messages upon a failed login attempt.

4. Nonfunctional Requirements

Nonfunctional requirements outline primarily *how* the application will achieve what is does rather than *what* it does.

4.1. Server-Client System

In order to connect many clients together across a network, a server-client system will be developed. There will be a single server instance which all clients will connect to and interact through.

4.1.1. Packet System

The main communication protocol of the server-client interface will be the packet system. This system will be able to write primitive data types to packages of data which can be transmitted from client to server and vice versa in order to communicate events and data. The packet system will run writers and readers on separate threads allowing for fast as possible communication. It will also provide a thread-safe queuing system for packets.

4.1.2. Server-Side JSON File Data Saving

The system will also require the server side of the application to save data. Explicitly, the server must store user account data and past chat room messages. In order to contain the entire application as a single JVM process, data will be read and written to local files on the server using the JSON protocol. JSON will provide a versatile system to go from objects to disk then back again. The JSON.simple library will be used.

4.2. GUI Library

In order to implement a graphical user interface effectively, Java's Swing library will be used to design a GUI which will be supported across all graphical platforms. Although there exist newer frameworks for UI, in order to target a more universal version of Java, swing will be used to develop GUI look and feel, and event handling.

4.3. Full Java Implementation

It is additionally required that the entire application be built in Java, both for the server and the client application. The client application is expected to be compatible with all common desktop operating system through the Java Virtual Machine. The server application will not feature a UI, but must be compatible with common server operating systems as well as desktop operating systems. All external libraries used should come packaged with the final Jar files.

5. Milestones and Deliverables

The milestones for this chat application will be described in terms of what a user will be able to do at the end of each milestone. This is to help create an objective requirement which stakeholders as well as developers can confirm. Design, Analysis, Implementation, and Testing will be carried out with respect to each of these major sprints. A Gantt chart for estimated timeline is displayed at the end of this section.

5.1. Milestone 1 : Backend System

Milestone 1 includes all the behind the scenes work required for the application to work correctly, excluding a polished user interface.

5.1.1. Mi1.1 : Server-Client System and User Accounts

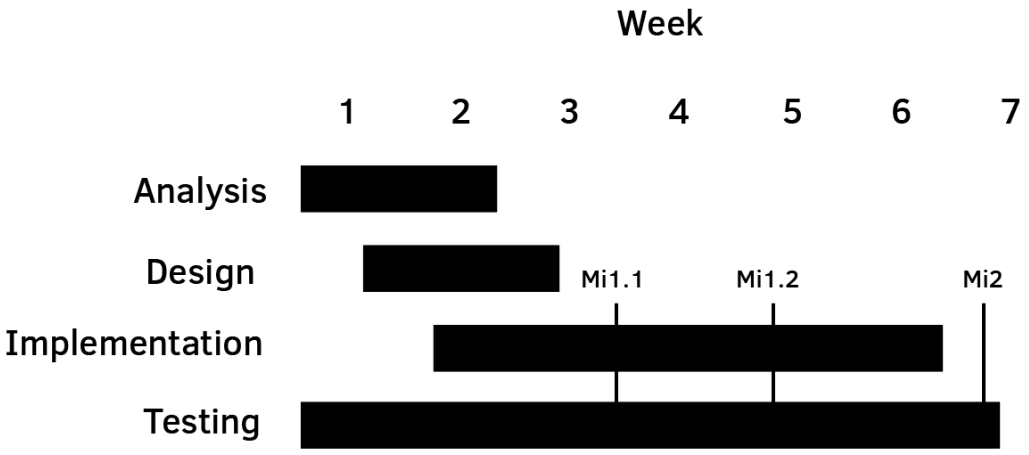
The first step of milestone 1 will be the server-client system basics (Socket and Packet system), as well as basic user account features: account creation, login, account information reading and writing via JSON. Users will be able to login and create accounts from the command line on a remote client upon completion of this stage.

5.1.2. Mi1.2 : Chat System

Stage 2 includes all the backend functionality necessary to start a chat with another user, join a chatroom, enter a chatroom, and communicate through that chatroom. Users will be able to login, register, direct message, message to the global chat room, and receive login notifications through the command line on a remote client after this phase.

5.2. Milestone 2 : GUI System

The second major milestone will be to take all of the previously developed backend systems and plug them into a graphical user interface that will server as a user-friendly outer shell for the application.


Week
1
2
3
4
5
6
7
Analysis
Design
Implementation
Testing
Mi1.1
Mi1.2
Mi2