SE 3XA3: Software Requirements Specification Node Messenger

Team #24, Node Messenger
Tasin Ahmed - ahmedm31
Shardool Patel - pates25
Omar Elemary - elemaryo

December 5, 2018

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Revision**	**Notes**
2018-10-03	0	Initial Revision
2018-11-25	1	Final Copy

1 Project Drivers

1.1 The Purpose of the Project

As the world becomes increasingly more connected through the internet, many common internet users are looking for an easy way to communicate with each other or reach out to distant loved ones. The market for messengers has become saturated with products that put the goal of earning maximum revenue over the needs of the consumer. Our team's focus is to capture consumer interest by implementing a free and accessible web application messenger that allows them to chat with other users in a simple, clean and non-intrusive way. Node messenger will become a haven for users searching for a consumer-friendly product with great functionality.

1.2 The Stakeholders

1.2.1 The Client

The clients for Node Messenger are Dr. Asghar Bokhari and the teaching assistants for Software Engineering 3XA3. As the commissioners of this project, Dr. Bokhari and his teaching assistants will oversee the development of this software and guide our team to ensure that the final product produced meets required project standards and is appropriately documented.

1.2.2 The Customers

Node messenger will serve as a general purpose chat application for any user with access to the internet and a distinct email account. These customers of Node messenger will be users looking to connect with friends and family or start group chats for various discussions or collaborations.

1.2.3 Other Stakeholders

Other stakeholders included the development team and future developers who might want to build upon this open source project.

1.3 Mandated Constraints

The following is a list of constraints placed upon the development of this product

- The product is to be developed at a monetary cost of $0.
- The product is to be a free open source software that can be improved by future developers or the team.
- The product will remain free to use for its consumers and customers.
- The product will protect user information from privacy invasions.
- The final product will be complete by December 5th, 2018 - The client imposed completion date.
- The final product will operate on both Windows and Mac OS.
- The final product will operate on current conventional machines and will not require any specific hardware upgrades.
- The final product must be a re-implementation of Tinode Messenger, an open-source web app messenger. Therefore, it will need to share similar functionality and design to its inspiration Tinode Messenger.

1.4 Naming Conventions and Terminology

Not Applicable

1.5 Relevant Facts and Assumptions

- Users will comply with policies of use and will not try to intentionally ruin the experience for other users.
- One server can handle all the users.
- Users will follow established online interaction etiquette.
- Only one user is allowed per browsing session.
- Users is familiar with using online web messengers.

- Users are of age and are responsible for their own experience during web chat sessions.
- Users and developers have access to the internet.
- Users and developers have access to input and output devices (i.e keyboard, mouse, and screen)

2 Functional Requirements

2.1 The Scope of the Work and the Product

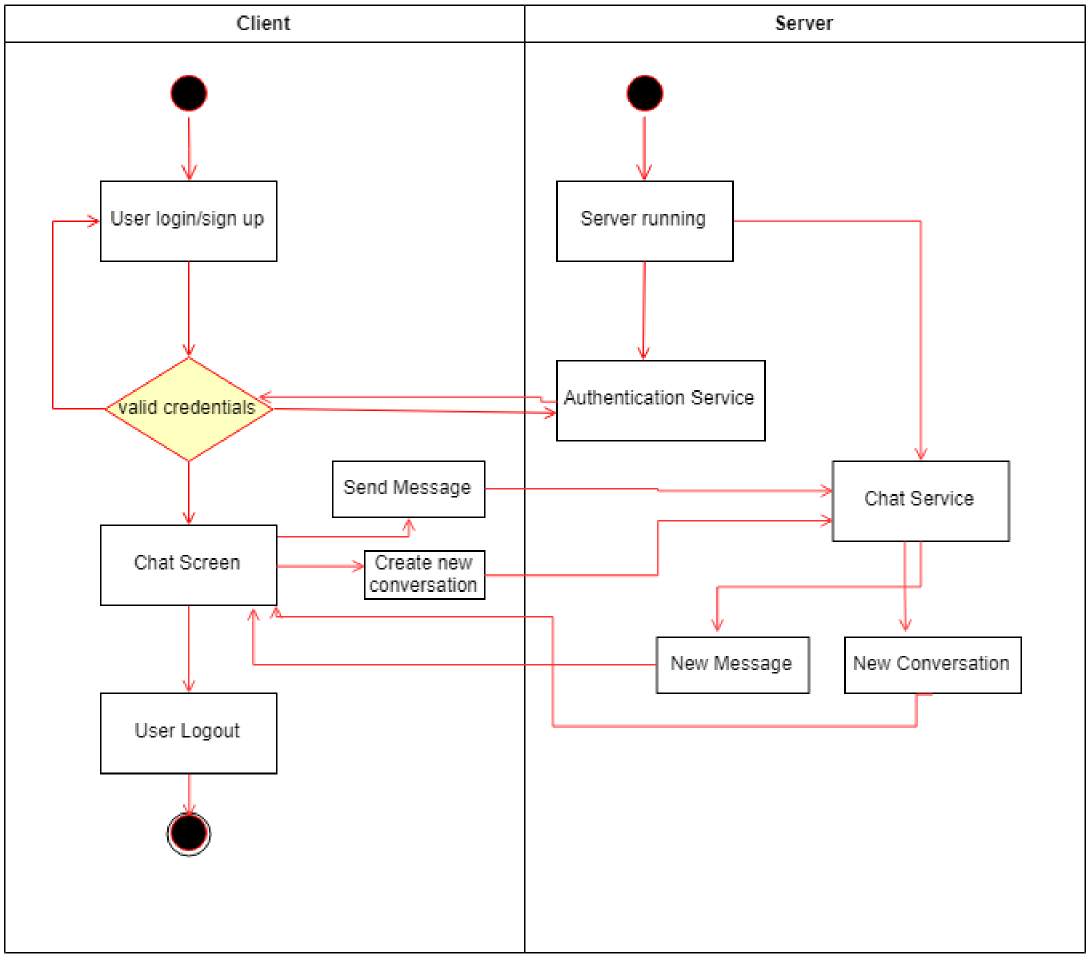

Figure 1: Activity Diagram

2.1.1 The Context of the Work

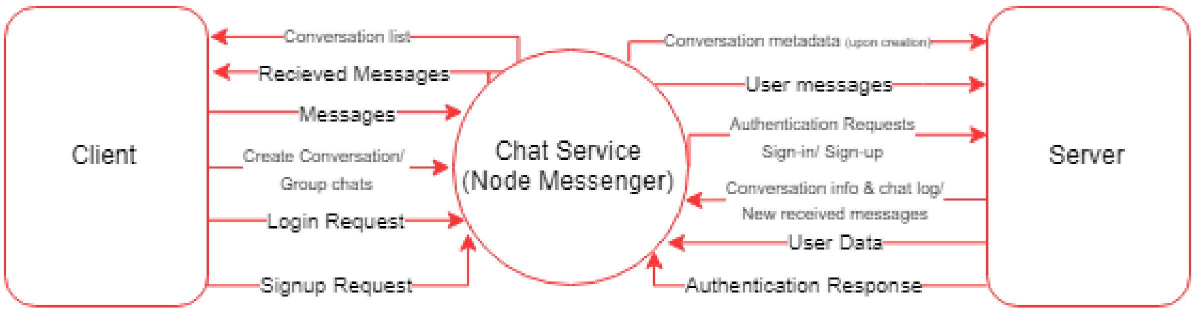
Client
Conversation list
Recieved Messages
Messages
Create Conversation/ Group chats
Login Request
Signup Request
Chat Service (Node Messenger)
Conversation metadata (upon creation)
User messages
Authentication Requests Sign-in/ Sign-up
Conversation info & chat log/ New received messages
User Data
Authentication Response
Server

Figure 2: Work Context Diagram

2.1.2 Work Partitioning

Event Name	Input and Output	Summary of BUC
Authentication Request	User Credentials (in)	Check the authentication credentials for the user trying to sign in or sign up
Sending a message	message data (in)	Log the message into appropriately conversation and broadcast it to other users in that conversation
Create conversation	Member emails (in)	Create a conversation reference in database with passed in members
Receive messages	Message data (out)	Send client the message data
Added to conversation	conversation info (out)	send client the conversation meta-data when they are added to a conversation

Table 2: Event List

2.1.3 Individual Product Use Cases

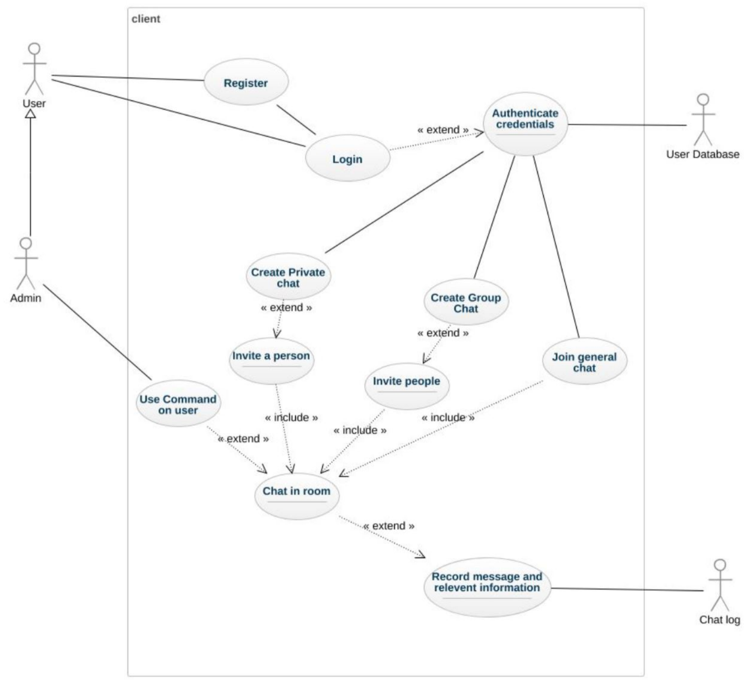

Figure 3: Example Use Case

2.2 Functional Requirements

- FR1: The software shall be free to access by all users through the web app ~~or cross platform app~~.
- FR2: The software shall allow user to log-in to web app messenger using ~~user name or~~ email address and password.
- FR3: The software shall allow user to remember their log-in information on the web app for easier and faster access.
- FR4: The software shall allow users to log-out and log-in to the system at will.

- FR5: The software shall allow user to create an account with Node Messenger by providing display name, email address and password ~~and profile picture~~.

- FR6: The software shall allow communication through one-on-one messaging between other Node Messenger users.

- FR7: The software shall allow the user to create conversations with other users through attached email ~~or phone numbers~~.

- FR8: The software shall allow the user to create group conversation with 10 other Node Messenger users independent of friendship status.

- FR9: The software shall support persistent message storing of multiple chats with paginated message history.

- FR10: The software shall distinguish between user sent and received messages using a different color scheme for each situation.

- FR11: The software shall convey the display name and send date attached to individual messages.

THE FOLLOWING ARE OLD FUNCTIONAL REQUIREMENTS NO LONGER APPLICABLE TO OUR CURRENT PRODUCT DUE TO APPLICATION SCALE

- FR12: ~~The software shall validate new accounts with the use of randomly generated verification codes sent to the user's email address.~~

- FR13: ~~The software shall allow the user to modify group conversation settings by changing its name and picture, add and remove members or leave group based on administrative status.~~

- FR14: ~~The software shall give the user control over individual conversations with the ability to delete chat and mute or block other user.~~

- FR15: ~~The software shall display the online and offline status of other Node Messenger users using green circle for online and grey circle for offline status.~~

- FR16: ~~The software shall allow the user to access other user's profile information via the info tab.~~
- FR17: ~~The software shall display an indicator for the number of unread messages besides chat menu and beside the browser tab name.~~
- FR18: ~~The software shall allow user to transmit documents, photos and videos of many file formats to other users.~~
- FR19: ~~The software shall send server-generated notification alerts and message sounds to the user whenever a message is received.~~
- FR20: ~~The software shall display message status notification indicating message delivery to server, received and read notifications and typing notifications.~~
- FR21: ~~The software shall allow the user to mute notification alerts and message sounds from Node Messenger.~~
- FR22: ~~The software shall allow user to modify their profile by changing their profile picture, name and password.~~
- FR23: ~~The software shall support font modifications such as bolding and italics.~~
- FR24: ~~The software shall support different languages and emojis.~~

3 Non-functional Requirements

3.1 Look and Feel Requirements

NFR1: Description: Messages and contacts should display properly
Rationale: Make contacts easy to add, and messages easy to send.
Fit Criterion: The messenger must be able to retrieve the correct contacts, and messages. It must be able to display messages in the proper order, from past to present.

NFR2: Description: The overall theme of our messenger should be visually appealing.
Rationale: The colours should capture the user's attention to keep their interest.
Fit Criterion: The colors used by our messenger must abide by the Color Theory, to ensure important areas of the messenger stands out.

3.2 Usability and Humanity Requirements

NFR3: Description: The structure of the messenger should be very self intuitive.
Rationale: The user must be able to use the messenger without requiring previous knowledge. Must be similar to other messengers to make the transfer to our messenger very easy.
Fit Criterion: Simple structure for the messenger, with contacts to the left, and message box taking up most of the screen. The user can start messaging other users with the click of a button.

NFR4: Description: Node Messenger should be easily accessible anywhere.
Rationale: Node Messenger will be accessible using a short and simple URL
Fit Criterion: The messenger must be able to perform well on all machines that have a browser supporting the latest versions of HTML, CSS, and Javascript.

NFR5: Description: The servers must be able to handle large numbers.
Rationale: Node messenger will be available to anyone with a electronic device, so it must be able to allow a large number of users to use our messenger at the same time.
Fit Criterion: Test different loads to see if the servers can handle a large number of users at once.

3.3 Performance Requirements

NFR6: Description: Display and send messages quickly.
Rationale: Store and retrieve messages quickly from the database.
Fit Criterion: The messenger must be able to retrieve past messages from the database fast, while being able to send messages to other users quickly.

NFR7: Description: Load the messenger quickly and effortlessly.
Rationale: Ensure the messenger is clutter free so it loads fast.
Fit Criterion: Ensure that the messenger keeps the machine's memory usage to a minimum throughout it's total usage time.

3.4 Operational and Environmental Requirements

NFR8: Description: The messenger can run on windows and other operating systems.
Rationale: The messenger will work on any operating system, given that a proper browser is used.
Fit Criterion: Node Messenger must be able to run on all browser that support the latest versions of HTML, CSS, and Javascript.

3.5 Maintainability and Portability Requirements

NFR9: Description: Maintenance must be kept to a minimum.
Rationale: Maintenance of the messenger shall be kept to a minimum as to not interfere with the user.
Fit Criterion: We will ensure that the messenger is atleast 90% error free prior to release, to ensure a positive user experience.

NFR10: Description: Node Messenger must be cross-platform
Rationale: Node Messenger must be usable from a variety of different platforms
Fit Criterion: We will test to make sure Node Messenger works on atleast 80% of devices

3.6 Security Requirements

NFR11: Description: Our messenger must keep personal user information safe.
Rationale: Node Messenger must not under any circumference share private user data.
Fit Criterion: The messenger must ensure 100% privacy of all user information.

3.7 Cultural Requirements

NFR12: Description: Node Messenger should be respectful of all users of different cultures and backgrounds.
Rationale: Node Messenger will make sure not to offend users of other culture and background.
Fit Criterion: The messenger should be available and respectful to every person worldwide.

3.8 Legal Requirements

NFR13: Description: Node Messenger must obey the law.
Rationale: Node Messenger must not disobey any laws.
Fit Criterion: Node Messenger must abide by all the rules.

3.9 Health and Safety Requirements

NFR14: Description: The colors Node Messenger uses must not put too much pressure on the eyes.
Rationale: Staring at a screen for a long period of time can cause eye strain, so we must make sure to use appropriate colors.
Fit Criterion: We will be using light colors that are easy on the

eyes, so they do not have a significant effect with long usage of the messenger.

4 Project Issues

4.1 Open Issues

The following information outlines issues that have been raised, but have yet to be resolved:

iss.1: The back-end infrastructure that the team plans to use (firebase) does not allow many custom data handling features such as encryption, complex queries etc.
This is not a important issue since the users' who might share sensitive information might be the only customers that will be affected in the unlikely case that the server would be accessed without appropriate authorization. Express server with web-socket communication will be considered and evaluated if the issues severely impacts security or any other concerns.

iss.2 The team also plans to use firebase authentication to handle sign in and sign ups. This is convenient but limits certain profile customization features like changing name, adding profile picture etc. This issue should not affect the usability/functionality of the application. Non-functional requirements such as customization might be affected. The team will research other services that allows greater user customization.

4.2 Off-the-Shelf Solutions

Currently, there are many solutions that occupy the market of online web communication.

1. Tinode Messenger: This product is a free to use open source web-app for chatting that acts as a model for our current implementation of Node Messenger. It features a minimal user interface with basic one-on-one messaging and group chat conversations. Users create a custom profile with an uploaded profile picture and set tags for discovery.

2. Facebook Messenger: This product is created to compliment the Facebook by connecting all their users and allowing them to communicate. The product implements many features such as file transmission, group

chat creation and providing the user a lot of control over their experience. The app can be accessed through the Facebook domain and is cross platform.

3. WhatsApp Messenger: This product has a web-app element that functions off the mobile app. However, it is a well received application praised for its simplicity in group chat creation and management along with connecting a plethora of users through telephone numbers.

4.3 New Problems

Node Messenger will occupy just one of the many web addresses being hosted. It should not create any problems, unless the website cannot hold a large amount of people at a time. The product will not have any effect on the installed systems, as nothing is required to be downloaded to be used. Node Messenger might cause problems for the users if used for a prolonged period of time, such as hand pain, or eye soreness.

4.4 Tasks

Task	**Completer's Role**	**Timeline**
Model Implementation	Software Engineer	Oct 10th
Model Revision	Client	Oct 14th
HTML and CSS implementation	Software Engineers	Oct 25th
Javascript backend	Software Engineers	Nov 1st
Revision	Client	Nov 3rd
Host Node Messenger	Software Engineers	Nov 24th
Maintenance	Software Engineers	Yearly

Task 1: Create a model of the Node Messenger website to visualize how everything will be positioned. This will help to understand how everything in our website will interact with each other.
Task 2: Create the front-end of the website using HTML and CSS. Ensure the website is clean and visually appealing.
Task 3: Create the back-end of the website using Javascript. Ensure all components of the messenger is fully functioning and creates the desired output. Task 4: Host the messaging app on a free web hosting site.

4.5 Migration to the New Product

Not Applicable.

4.6 Risks

Risks are inherent in the development of any product. The following information highlights possible risks for this project:

- Server Overload
- Privacy Invasion
- Inaccurate Data Rendering
- Bad Programming Practices
- Complex Project Scope
- Over-estimated Project Size
- Loss of Resources
- Incomplete Documentation
- Testing Difficulties

4.7 Costs

Monetary costs for this project have been constrained to $0 which is achievable with free web-hosting. High labour costs are evoked from the considerable amount of work hours required for the software's development. Team members must be able and willing to commit approximately 7 hours per week to complete their designated tasks. Resource costs also remain low with the assumption all development tools and software used such as **React Semantics** assets and **Firebase** remain free of charge.

4.8 User Documentation and Training

With the assumption that the consumer of this software is familiar with using web-messaging applications, the team plans to minimize user documentation as the product is required to be simple and intuitive to use. However, it is understood that with future additions and updates, the product may become more complex. Should the need arise, the team will compile a user documentation intended to assist all clients and consumers with effectively operating our software.

User document contents:

- User Manual
- Help Section

4.9 Waiting Room

The following are features that the team wishes to implement in the future development of this project:

- Notifications: An important component of any online messaging system is inform the user of updates and events centered around the application. Whether it is alerts to new messages or an indicator for total unread conversations, this feature would greatly improve the usability of our product. This feature would be developed as an update to the final release of the product.
- Profile Customization: The ability for the user to customize their profile helps gives them a distinct identity within our application and invest them in using it. Allowing the user to modify their profile information such as display name, phone number or profile picture would provide more search tag options for discovering users and therefore expand our user base. This feature would likely be incorporated in a Version 2 product update.
- Group Controls: This feature would allow the user to administer group chats by adding or removing members, modifying group name and picture and even leave the group if desired. This is essential in group

conversation programs that gives our product greater flexibility and functionality. Such a feature would be implemented in a Version 2 product update.

4.10 Ideas for Solutions

The following is a purposed solution to some of the outlined requirements that would be implemented through the back-end portion of the program.

Firebase Database: Utilizing Firebase for our back-end instead of socket.io can give us greater flexibility with our web-app as it allows us to properly structure the database and choose what information to store. The database can be built to store user information for authentication, messages and conversations with the use of specific fields and keys for better data communication while still retaining real time connection of socket.io.

5 Appendix

This section has been added to the Volere template. This is where you can place additional information.

5.1 Symbolic Parameters

The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.