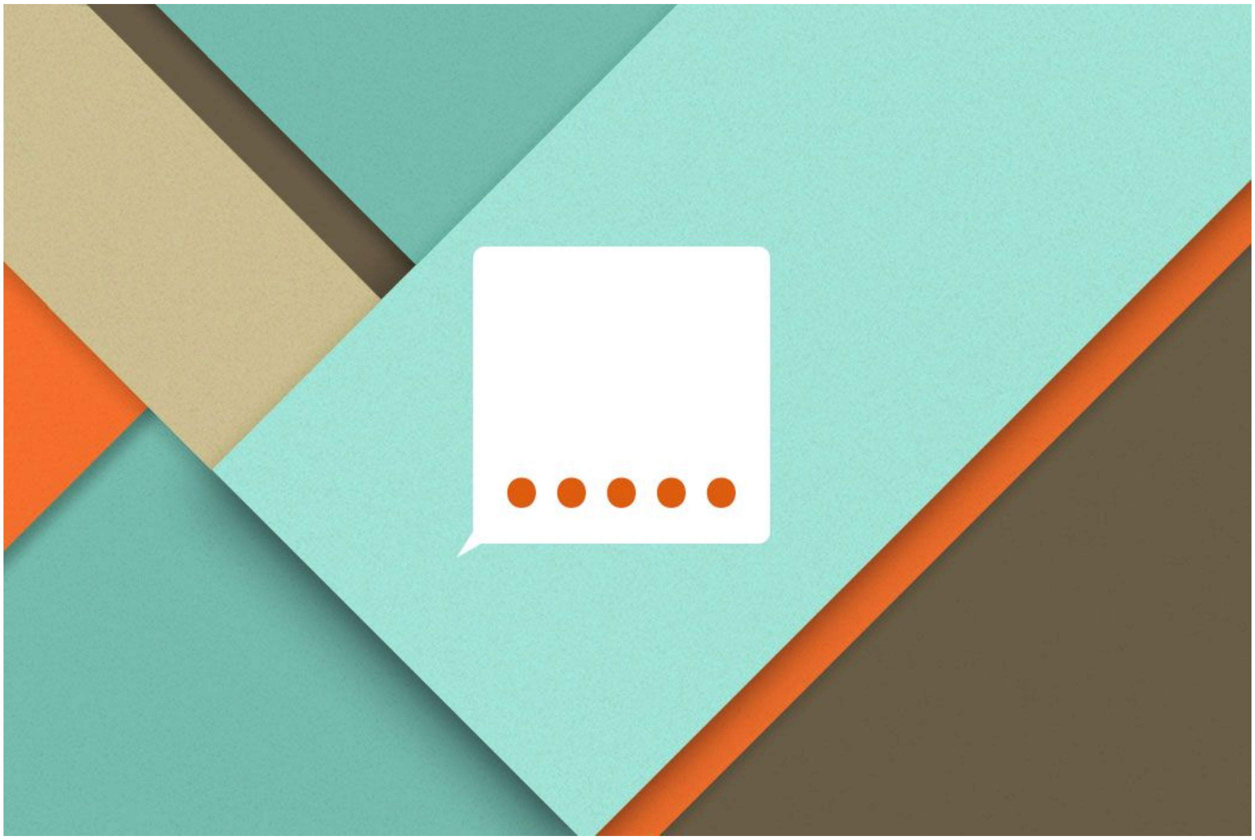
Group 7 Final Report

Web Chat Server



Project Group Team 7 Members

Nicholas McCormick

Joshua McBrayer

Caitlyn McClain

Table of Contents

1 Overview

1.1 Introduction

The goal of Team Seven was to create a lightweight, browser based chat application that can serve a large number of clients and allow fast, reliable, instant messaging between clients. The chat application will be hosted on a server and clients will be able to connect via a third-party browser such as Google Chrome or Firefox. Once logged on, clients are able to enter a username for themselves, and chat with other clients on the same server.

Project Goals Update

Team Sevens original goal was to build and deploy a real-time chat server with the ability to store messages logs to the internet. This entails designing/building the application, and finding a suitable place to host the application as well as the database. At this point in the design process we have only found a place to deploy the application to, but not with support for a database, unfortunately we will be unable to add a relational database to our real-time chat server due to the restrictions of the server we wish to deploy to. However, it will be possible to add a database into our application in the future should the option become available.

Our project document has been updated to reflect this change.

1.2 Team Member Roles

Roles	Project Lead, Front-end Programmer, Back-end Programmer, User Interface Designer, Project Document Writer
Member	Nicholas McCormick
Description	Nicholas was responsible for coding the application and doing any research that needed to be done. Including investigating where to host the application.

Roles	Code Review, Schedule, Quality Assurance Manager
Member	Joshua McBrayer
Description	Joshua was responsible for the schedule. He was also be responsible for reviewing the other members work.

Roles	Front-end Programmer, Back-end Programmer, Technical Writer
Member	Caitlyn McClain
Description	Caitlyn was be responsible for contributing to the coding of the application, and aiding with the technical writing.

1.3 Contribution History

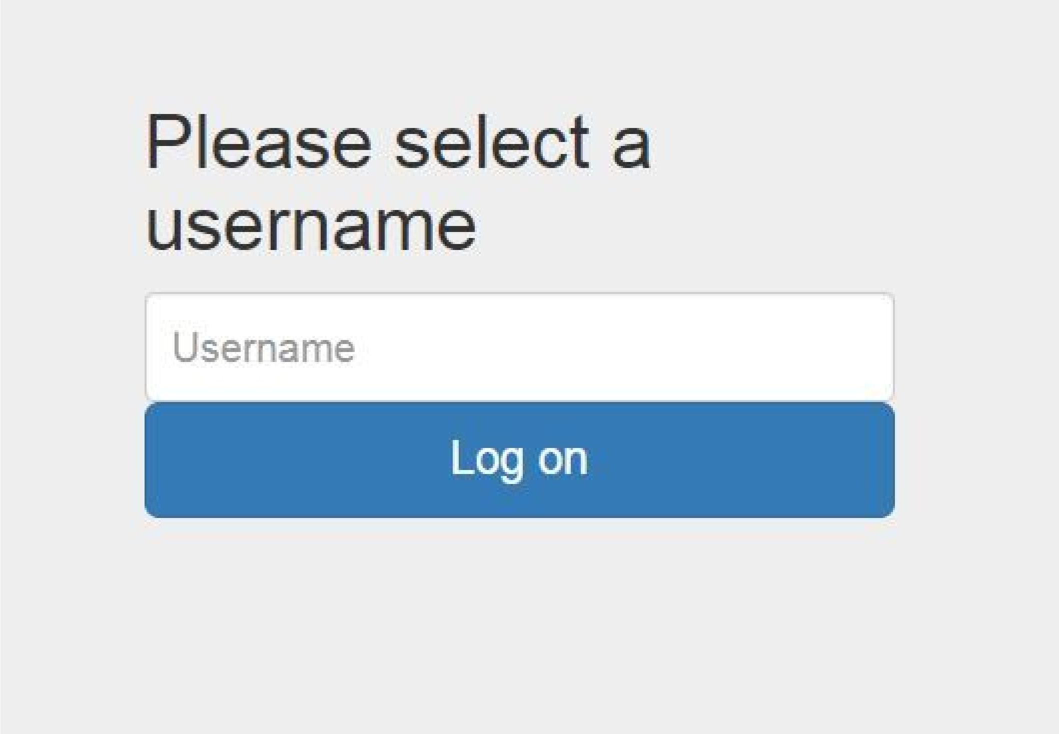
Team seven’s web chat server was only possible because of the efforts of Nick McCormick, Caitlyn McClain, and Joshua McBrayer. Team seven used the website ‘discord’ as its main communication hub alongside email. Google docs was used so the team could all share and work on the same document. Nick was responsible for creating the google document, as well as keeping it up-to-date during the application’s development cycle. During the first programming phase Caitlyn stepped in to document phase 1. Caitlyn also created the application’s logo in Photoshop. Josh was responsible for the schedule.

The programming phases began with the group creating a prototype chat application in Java. Joshua wrote and developed the program. During this time Nick found a place the team could host the chat application. Nick was responsible for developing, coding (front and back-end) and deploying the chat application to the Azure web server.

1.4 How to use the application

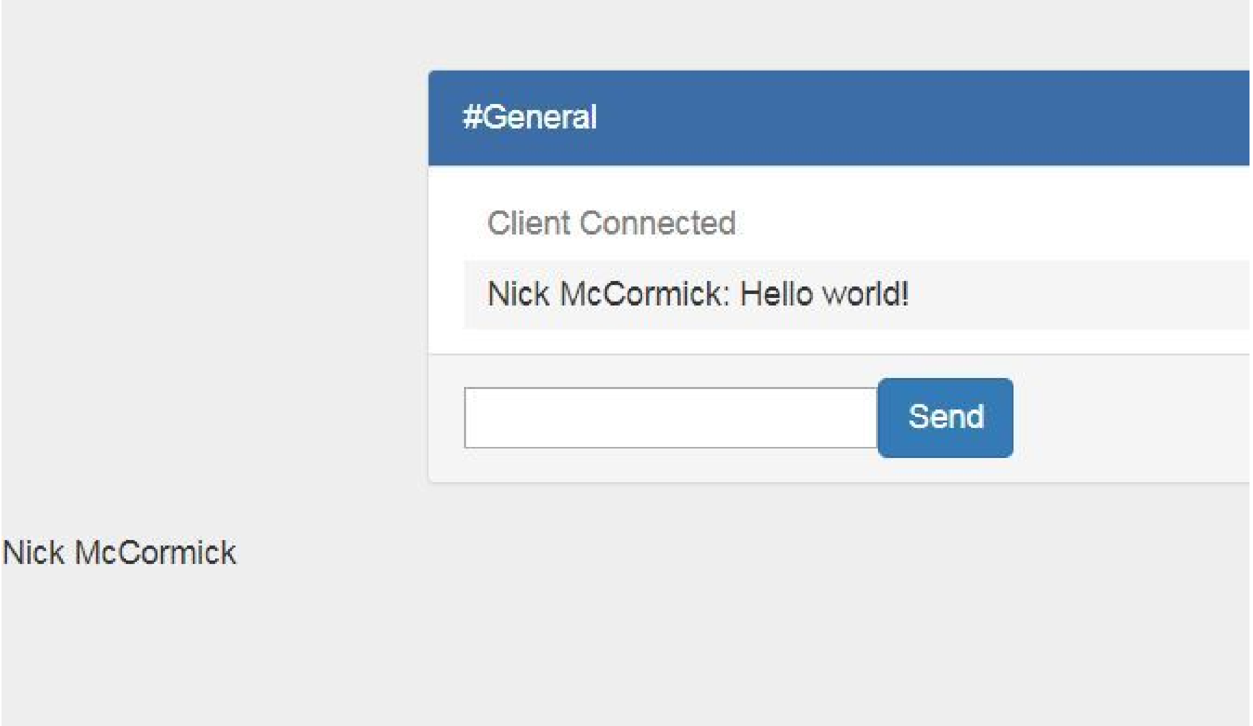
The chat application is accessible from any browser connected to the internet. The only requirements for the application is a device connected to the internet and a modern web browser. The user must enter a username before entering the general chat. Once the user enters chat their name is displayed below the chat area along with anyone else currently in the chat.

1.4.1 The index page of the application.



Please select a username

1.4.2 After entering a username.



Client Connected

Nick McCormick: Hello world!

Nick McCormick

2 Project Plan

2.1 System Specification

The user will be able to interact with the program by using third party web browsers such as google chrome or mozilla firefox over HTTP. HTML/CSS/Javascript make up the interface that users interact with while the back-end is written using Node.js Javascript.

The client needs:

1. A device connected to the internet.
2. A third-party web browser.

2.2 Risks

This section addresses the risks involved with the project. The purpose of this section is to identify, analyze, and communicate those risks among the team so those risks can be mitigated as much as possible. Team seven utilizes online communication tools such as email, instant messaging, and voice chat to communicate with each other. The hope is that by identifying the risks in the early stages of the project, the team will still be able to meet goals and deadlines even if there are problems.

2.2.1 Risks Final Report Update

Highlighted table sections were risks the group encountered and needed to adapt to. In group sevens case we were unable to both deploy the application to a web server while also implementing a SQL database to store chat logs. Group seven was able to deploy our application to the Microsoft cloud web server ‘Azure’ for free under a student licence, however hosting a database on said web server was not an option due to the price.

Risk	Probability	Impact	Plan
Unfamiliar with the Node.js library.	90%	Being unfamiliar with Javascript and the Node.js library would cause major delays and could cause the team to fall behind the schedule.	Become more familiar with Javascript and the Node.js library through self-learning using research.
Unable to deploy application to server.	30%	If the chat application is unable to be deployed onto a public server we will have to host the application locally and the team will not meet one of its goals.	Do research into possible hosting services and check compatibility between the application and the languages the server supports. Choose back-end language that is compatible with the hosting server.
Unfamiliar with client to server, server to client communications and websocket programming.	50%	The team could experience delays and miss deadlines.	Become more familiar with websocket programming and Socket.IO.
Unable to find suitable place to host relational database server.	50%	Certain features of the application will need to be cut. These features include logging the messages in chat, and displaying the last 10 - 20 messages left in the chat when a person	Early research should be done to find a suitable database and place to host the relational database as well as spending the necessary time to set up the database.

		joins. Time would also be wasted.	

3 User Guide

3.1 Overview of Chat Room Features

Web Interface

Our browser based web chat application will utilize a simple but effective graphic user interface to allow easy navigation and messaging within the chat room. Users will be able to enter a username, see when clients join, leave, and are currently typing.

You can access the web interface here:
http://cmsc495webchatproject.azurewebsites.net/chat.html

System Requirements

Our chat application will run entirely in the browser. The only thing required by the user is an internet connection and a modern web browser with JavaScript enabled.

Viewing and Posting Messages

The main page of the application is where you will be able to choose a chat handle, username color and begin posting messages to the group chat. The server will show who is currently typing, when someone joins and leaves the chat server. To post a message simply write out your message and click the “Send” button. The server will post and echo your message to all other users.

Exiting the Application

To exit the application you can simply close the browser window. Doing so will terminate the session and post a logout message to the other users in the group chat.

Future Updates

The purpose of this section is to list some possible improvements and updates Group Seven may implement into the chat application.

- Usernames ✔
- Friend Lists
- Multiple chat rooms
- A SQL Database
- Show who is online ✔
- Private messaging

4 Project Design

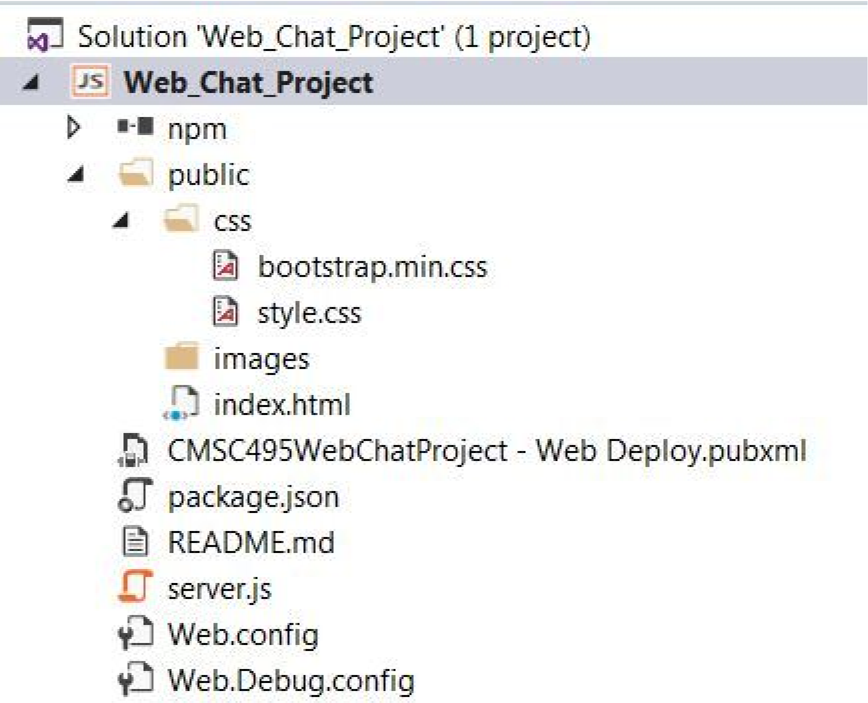
4.1 User Interface and Functionality

Team Seven created a user-interface for our chat application that allows clients to message other clients in the same chat room instantly. The chat application will be hosted on Microsoft's cloud based server called Azure and will be accessible via any third-party browser such as Google Chrome or Firefox. Clients will have the ability to enter a username and send messages to other clients in the chat. Additional functionality of the chat application will include:

- See when people join the chat.
- See when people leave the chat.
- See who is currently typing.
- Select color of the username.
- Enter a Unique Username.

4.2 Application Structure

The current structure of the application is as follows:



```
Solution 'Web_Chat_Project' (1 project)
  Web_Chat_Project
    npm
    public
      css
        bootstrap.min.css
        style.css
      images
      index.html
    CMSC495WebChatProject - Web Deploy.pubxml
    package.json
    README.md
    server.js
    Web.config
    Web.Debug.config
```

4.3 Alternate Designs

Early in the development process the team decided to use Javascript as the core language. By agreeing to use Javascript the team hoped to avoid issues late in the development cycle when the front-end and back-end code would have to work together. An alternate design however, could have used PHP as the back-end language.

5 Test Plan and Results

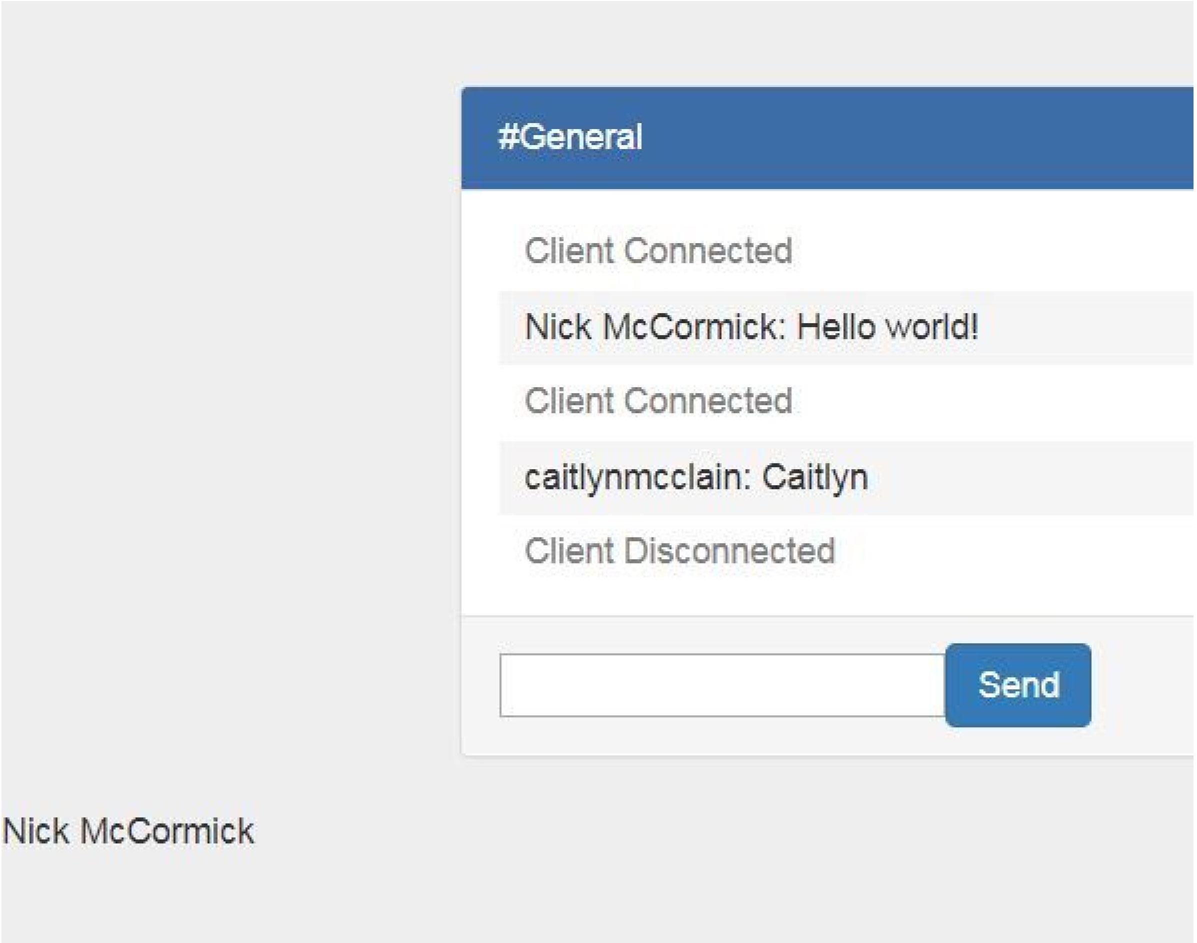
5.1 Test Plan

The goal of this initial test plan is to develop use cases in which the application might behave unexpectedly and potentially fail. This table represents the initial test plan prior to developing the project and will serve as a general guide when creating the phase one test plan.

As the project progresses the test plan will evolve to meet the needs of the project and will become a more comprehensive test plan.

Action	Date	Test Case	Expected Outcome	Results Found	Pass/Fail
Project Runs					
The project will run.	10/15	The code of the project runs.	Command prompt should say, “Project is running.”	Success	Pass
The server is created.	10/15	Create the chat server.	In the command prompt, output should read “The server was created!”	Success.	Pass
The server is listening for user input.	10/15	The server is running and waiting to accept input.	The chat server should echo the input of a client to every client connected to the server.	Success	Pass
Chat Server Features					
User connects.	10/14	The server should let a user connect.	({user}) has entered the chat.	Success	Pass
User enters username.	10/14	The server should accept a valid username.	The user enters the chat with the entered username.	Success	Pass
User enters a message and sends it.	10/8	The server should accept the message text and test it to make sure it’s safe, then echo it to other clients.	The message should appear in the chat. Other clients should be able to see it.	Success.	Pass
Client closes browser.		The session terminates.	Other clients in the chat should see that a user has disconnected.	Success	Pass

Picture of application during testing.



Client Connected

Nick McCormick: Hello world!

Client Connected

caitlynmcclain: Caitlyn

Client Disconnected

Nick McCormick

6 Development History

Chat Application Project			
MILESTONES			
TASKS	Done By	DESCRIPTION	DATE
Project Proposal			
Group 7 Created		Team members assigned.	8/24

Setup Group communication.	Nick	Created discord server.	8/27
Assess individual skills	Group 7	Assess team member's experience/skills	8/28
Assign tentative roles	Group 7	Assign temporary roles.	8/28
Project Plan			
Setup Project Plan Document	Nick	Create the google doc. Write the project plan.	9/1
Research support technologies	Nick/ Josh	Investigate potential references and library documentation.	9/4
Compose the Project Plan Paper	Group 7	Caitlyn document layout and look. Nick technical writing. Josh schedule.	9/4
Application Logo	Caitlyn	Design logo in Photoshop	9/4
Update look and layout of project document.	Caitlyn	Improved the look of the document.	9/4
Test Plan/User Guide			
Setup Test Plan & User Guide	Nick	Update document with sections and layout of test plan and user guide.	9/10
Compose the Test Plan/User Guide	Nick	Document the initial test plan and user guide.	9/10
Project Design			
Compose Project Design Report	Nick	Research and document the design of the project.	9/25
Phase 1			
Design front-end components	Nick	Create file structure of app and necessary HTML/CSS files.	9/27
Design back-end components	Nick	Create back-end Javascript file.	9/28
Find suitable web server to host app.	Nick	Create an account and set-up the account to host the app.	9/25
Develop prototype web chat	Josh	Write the code for a local chat server in	9/28

application in Java.		Java.	
Document the prototype app	Caitlyn	Updated the project document with phase 1.	9/28
Code review.	Group 7	Review the code and documentation for phase 1 before turning it in.	9/28
Phase 2			
Setting up the project file and installing the application dependencies.	Nick	Install Node.js and project dependencies (Socket.IO and Express)	10/2
Writing the core functionality of the application	Nick	Write the core functionality of the application.	10/2
Deploy the application to Microsoft's Azure web server	Nick	Deploy the chat application to the server.	10/2
Phase 3			
Develop Front-end	Nick	Continue to develop core functionality.	10/7
Develop Back-end	Nick	Continue to develop core functionality.	10/7
Create and improve User Interface	Nick	Create HTML/ CSS files. Add Bootstrap CSS template.	10/8
Update project document with phase 3 notes and milestones/documentation.	Nick	Write about phase 3 in the project document.	10/8
Final Report			
Outline final project document.	Nick	Begin to write and layout the final report.	10/11
Write the conclusion	Josh Caitlyn	Wrote the conclusion and lesson learned	10/15
Proofread final report	Josh Caitlyn	Proofread final report	10/15

7 Conclusions and Lessons Learned

7.1 Lessons Learned

Communication was one of the most important aspects to completing this group project. Individual progress also was necessary to complete the chat server. We all had to pull our own weight in this project, by supplying our ideas for areas we can do well in. Managing tasks and time for the necessary tasks to be completed was also a key initiative for us to do well in this project.

We used the Discord app to communicate with each other and share ideas, photos, code, and updates on the work completed. We seemed to be in constant contacted almost daily with our progress, we listed tasks we wanted to complete and took enormous effort to monitor the schedule for deadlines.

Proper assessment skills were necessary for our work to progress over the course of eight weeks when we agreed to complete our weekly tasks.

7.2 Design Strengths

Our main strength in the design is that any user may go online and have a conversation with another user. Due to the fact that there are no requirements for login, any user may create a username to chat with other users. The design is user-friendly, intuitive, and easy to use. The web chat server will attract a following since users can be specific about their username or become anonymous.

7.3 Limitations

The major limitation that we encountered with this project was a lack of time.

- Time constraints
- GUI design
- Troubleshooting

7.4 Suggested Improvements for the Future

Profiles for the user to create and photos to upload. The user may also accessorize their font to closely match their preferences. We believe these would make the web chat server more personal, so that users will engage in more time spent on the application.

Private and direct messages would also be a future improvement for the web chat server, as this will draw more attention from the user. When the web chat server does reach this stage, we could develop an SQL database to hold all account settings, password information, and personal information for each user and to organize the information.