Runtime Terror

Final Project Report

Kameron Jordan, Luke Williams, Taylor Hall

Capstone Project Fall, 2020

The University of West Florida

December 4, 2020

CIS 4595: Capstone Project

Dr. Bernd Owsnicki-Klewe

# **Abstract**

One of the simple joys in life is to head down to one’s local coffee shop and enjoy a nice conversation with friends, coworkers, or even a friendly stranger. However, during these pressing times of social distancing and self-quarantining, these moments are slowly slipping away from our lives. The purpose of this project is to create a web application that allows users to recreate these moments from the comfort of their home. Users will be able to meet up and chat online with the people they know, as with any chat application, or be able to hop in group chats with strangers and find new friends. Ultimately, we plan to create an easier way for people to keep and form connections with others in what may otherwise be difficult nowadays.

# **Table of Content**s

[**Abstract**](#_30j0zll) **2**

[**Table of Contents**](#_3znysh7) **3**

[**List of Figures**](#_bfsq5nlh5t12) **5**

[**List of Tables**](#_1t3h5sf) **5**

[**1 Executive Summary**](#_2s8eyo1) **6**

[**2 Comparison of Requirements**](#_3rdcrjn) **7**

[2.1 Use Cases](#_lnxbz9) 7

[2.1.1 Creating an Account](#_35nkun2) 7

[2.1.2 Signing into an Account](#_44sinio) 7

[2.1.3 Handling chats with friends](#_2jxsxqh) 8

[2.1.4 Handling chats with strangers](#_z337ya) 8

[2.1.5 Handling Friends](#_3j2qqm3) 8

[2.2 System Components](#_1y810tw) 9

[2.2.1 The Client](#_2xcytpi) 9

[2.2.2 The Server](#_1ci93xb) 9

[2.2.3 The Database](#_2bn6wsx) 9

[2.3 New Requirements](#_qsh70q) 9

[2.3.1 One-on-One Chat](#_3as4poj) 9

[2.3.2 Session Management](#_1pxezwc) 10

[2.3.3 Custom APIs](#_49x2ik5) 10

[**3 Timeline Comparison**](#_2p2csry) **10**

[**4 Results Vs. Expectations**](#_23ckvvd) **11**

[4.1 Chat Rooms](#_ihv636) 11

[4.2 Database Management](#_1hmsyys) 11

[**5 Process Review**](#_41mghml) **12**

[5.1 Feature Driven Development (FDD)](#_2grqrue) 12

[5.2 Online Only Collaboration](#_vx1227) 12

[**6 Work to be Done**](#_3fwokq0) **12**

[**References**](#_2u6wntf) **13**

# 

# **List of Figures**

This section should begin on a new page. Microsoft Word® has an automated feature under the "References" menu called "Insert Table of Figures." For this feature to work, all figures must have a caption. This should be one of the last pages to be completed. Some examples follow:

1. Test matrix 5

2. Design prototype in wind tunnel 6

# **List of Tables**

This section should begin on a new page. Microsoft Word® has an automated feature under "Insert Table of Figures." Change the caption label from “figure” to “table.” For this feature to work, all tables must have a caption. This should be one of the last pages to be completed. Some examples follow:

1. Model parameters 3

2. Fuel consumption data under nominal conditions 10

# **1** Executive Summary

For this project, our group created a web-based chat application that would allow users to join up and chat with friends or new people. After a few alterations from our initial project idea, we were able to successfully meet the specified technical requirements we designed at the beginning. The changes had to be made due to time constraints and from running into errors that took longer than expected to overcome.

# **2** Comparison of Requirements

The requirements we specified in the Project Plan were mainly technical requirements such as how the user interacts with the server and database. Each of the requirements were met and behaved roughly as expected with some minor adjustments. Essentially, the final requirements were the same as the initial, expected requirements.

## **2.1** Use Cases

Comparison of initial and final use cases are as follows:

### 2.1.1 Creating an Account

The initial use case for creating an account specified that the user will click on a button to create an account, enter the information necessary for an account such as an email and username, and finally have that information validated by the database. This follows exactly how this use case behaves in the final project, so this case was met and passed without any issue.

### 2.1.2 Signing into an Account

The initial case for this event was that the user will click on a button on the home page to login to their account with their credentials. The behavior of this event on the back-end was that the information the user entered was to be sent to the server for authentication against the database. The back-end operations for this event remained the same, but the front-end was modified slightly: instead of clicking on a button on the home page to login, the user will automatically be redirected to the sign in page upon entering the website and won’t be able to head to another page until they either login or create an account. This is to prevent unexpected errors when the user tries to access content they’re only supposed to access with a validated account.

### 2.1.3 Handling chats with friends

This one's a little different than how it was expected to go down. Originally, users in the chat room would have been able to invite others to the room after its creation so that they can make friends through their other friends. However, as of now, users can only invite others, regardless of their status as a friend or not, when the room is being created as a private chat room; there is not an option to add more users to a private room after its creation. This was due to running short on time and not fully integrating this feature.

### 2.1.4 Handling chats with strangers

This use case is vastly different from the one initially expected for it. At first, users will enter a chat room of other randomly selected users and be placed into a waitlist until a minimum number of users are in the room. Then, they will be able to chat with each other with a random conversation starter at the top of the room to help get the ball rolling for chats. We had to change this one due it being slightly over ambitious with the skill level we had at the time with Javascript and React, and decided to focus on having public, group chat rooms that anyone can join and chat with at any time without the conversation starter/topic.

### 2.1.5 Handling Friends

One of the specified requirements for adding a friend to a user’s friend list was they would be able to add them from a group chat room by clicking on a button and picking from a list current users in the room to add. This feature was scraped for the sake of time, and now the only way to add a friend is by searching for them via the search page and sending a friend request that way.

## **2.2 S**ystem Components

The components of our system are composed of the client, the server, and the database.

### **2.2.1** The Client

The client behaves exactly like it was initially intended to: it sends requests to the server to handle actions the user initiated.

### **2.2.2** The Server

The server behaves as expected: it processes the requests sent to it and sends responses back to the client such as a list of users when requested or an error message when the user tries to login with the wrong information.

### 2.2.3 The Database

The database acts as it was originally intended to but with an addition: the database now holds information about chat rooms as well as users instead of just users. This addition was to fix an issue with persistence with chat rooms that was not originally anticipated. Also, the database was originally going to be a MySQL database but after some technical issues, the group decided to switch to MongoDB which reacted with Javascript’s React library much more smoothly.

## 2.3 New Requirements

These are requirements that were not originally planned for but were deemed necessary in order for our system to function more intuitively for the user or to adjust for changes in our project plan.

### 2.3.1 One-on-One Chat

The ability for users to chat with a single, other user was not planned for in detail. The basic idea at first was to have users create as many private rooms as they wish where they can invite others to any room at any time they want. We decided that a much simpler option was to just create a chat room with just the two users in a way similar to how a “direct message” would be in other chat applications. As such, users can chat privately with another user by searching for them on the search page and clicking on the chat icon to start a conversation with that user.

### 2.3.2 Session Management

The idea of maintaining a session didn’t cross our minds until we actually started building the application. We threw around ideas for storing user information locally to their machine so that they may stay logged in for a period of time instead of just immediately logging out after they connect to another page. We decided to go with a session that will maintain persistence until the user either logs out or closes the web page entirely.

### 2.3.3 Custom APIs

As stated above, our original database option was MySQL but we switched to MongoDB after some technical problems with MySQL. We choose MongoDB after doing some research and finding that it works quite well with JavaScript’s React library specifically with the application framework for Node.js, Express. Upon further research and practice, we created application programming interfaces or APIs using Express to define URL routes for our database.

# **3** Timeline Comparison

The timeline created in the project plan matches the way we tackled the project when it came to the order of how we completed each feature. Since we followed a Feature Driven Development agile development process, we placed an emphasis on completing features and using those as goals for our sprints.

Aside from setting up test pages and the database which were only one week apart, each sprint was in two week cycles. For the first couple of sprints, we were either on track or slightly ahead of schedule, which can be seen in our commit history on GitHub. However, once we began implementing the chat rooms and friend related functionalities, we started to fall behind due to unexpected problems we came across. The difficulty of these features were far greater than what we expected and required a lot more time than the two weeks we anticipated. Another contributing factor during this time frame was other engagements outside of this class and project such as other exams or projects in other classes or jobs that cut into time for this project.

# 4 Results Vs. Expectations

While we were able to meet most of our requirements to varying degrees, it was at a cost to some of the more unique, “nice to have” features.

## 4.1 Chat Rooms

One of the main initial selling points for this application was having conversation starters or topics appear somewhere in the chat room to help users chat with each other by having something to springboard off of to talk about. We were unable to implement this feature due to time constraints and choosing to focus more on functional requirements like sending and receiving chats themselves.

Another feature for the chat rooms that was cut was the ability to add friends or atleast send friends requests from that page. This again was another time issue and could have been implemented with more time on our hands.

## 4.2 Database Management

Originally, MySQL was selected as our database management system but we ran into some technical issues trying to get it to work with our application. There was a connection issue that caused any attempt to connect to just timeout. After researching ways to resolve this, the group decided to switch to MongoDB after learning that it along with the web framework Express.js worked much more efficiently with React.

# 5 Process Review

The process we followed was an agile development process called Feature Driven Development, which proved to be quite effective for how we handled the project. Another aspect to note how the entire collaboration for this project was done online and how despite the challenge that would seemingly impose, we were able to adapt and overcome it.

## 5.1 Feature Driven Development (FDD)

FDD is focused on the idea of designing, building, and implementing our system by features one at a time. One of the reasons why we choose this agile approach was because it’s usually recommended for smaller projects like the one we choose to work on. After experiencing it first hand, we can see why splitting the sprints of our development cycle by goals, where each goal is a feature, was quite effective until a point. That point was when we began implementing some of the more complex features such as chat and friend related functions. However, this is most likely due to these requirements we specified being more broad and encompassing than the previous ones, and had we split these down into more manageable goals per sprint, FDD would have been successful for us throughout the timeline.

## 5.2 Online Only Collaboration

Another aspect that is noteworthy in our development process is that all of communication and collaboration was done entirely online. While it was outside of our control due to the pandemic, it was a unique experience for each of us and would have been strenuous if not for the tools we used had to ease the burden.

# 6 Work to be Done

As mentioned above, there were some requirements that were not met due to time constraints. The biggest one was the selling point of our chat application that would have given users in group chats random conversation starters to play around with. Another that was discussed was having some sort of notification system that would have notified users of certain events when they happened such as receiving a message from a room different than the one the user is currently in or when a friend request has been accepted or denied. Some smaller features that would have fallen into the “nice to have” category would have been profile pictures and having users be able to customize their rooms with certain styles/themes.

# **References**

[[1] ReactJS](https://reactjs.org/docs/getting-started.html)

[[2] Node Package Manager](https://docs.npmjs.com/)

[[3] NodeJS](https://nodejs.org/en/docs/)

[[4] MongoDB](https://docs.mongodb.com/)

# 