Final Year Project Title

Aaron Moran

Conor Shortt

Thomas Kenny

B.Sc.(Hons) in Software Development

April 30, 2021

Final Year Project

Advised by: Kevin O'Brien

Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

Contents

1 Introduction **4**

2 Context **5**

3 Methodology **6**

3.0.1 Testing 8

4 Experiments **10**

5 Technology Review **15**

6 System Design **16**

7 System Evaluation **18**

8 Conclusion **19**

9 References **20**

About this project

Abstract This project is an Social Media platform where you can create communities, post up discussions and follow your friends. . . . etc etc

Authors Aaron Moran, Conor Shortt, Thomas Kenny.

Chapter 1

Introduction

The final year project requires you to show initiative, consistency and the ability to showcase what you have learned over the course of four years. Three members consist of Aaron Moran, Conor Shortt and Thomas Kenny. We are like minded students that share an interest for learning new skills and developing our ability as computer programmers. We have different skills that when combined complement each other from creativity to problem solving.
With our interest in social media we decided early on that a social media website was a project that we felt we all could contribute too. A social media website requires multiple features, the in depth use of a database and an engaging UI. We were confident from the start with the team we have we could develop a website that could meet the standards of the websites out in the world today.
Throughout this development process we were were going to earn valuable experiences from testing to overcoming adversity.

Chapter 2

Context

The purpose of our project is to allow people to communicate through communities sharing ideas and knowledge whether it be news or comical. The objectives which we want to achieve from developing this project is to create and application where users can follow, react and comment and join communities as they wish without any restrictions.

All of the development processes will be discussed throughout the document as follows:

- Introduction - This will run down through the team members and the reason as to why we chose this project.
- Context. (Current)
- Methodology - This chapter will discuss the steps we took to develop the project and the testing which we carried out along the way.
- Experiments - Within this chapter we discuss the issues we faced and how we carried out experiments to overcome and solve the bugs.
- Technology Review - This is where we run down through the development stack that we chose and why we chose it.
- System Design - This chapter discusses the design decisions that we chose and the implementation of all our components.
- System Evaluation -
- Conclusion - This is where we discuss the final stages of development and what we learned while working as a team on a project.

Chapter 3

Methodology

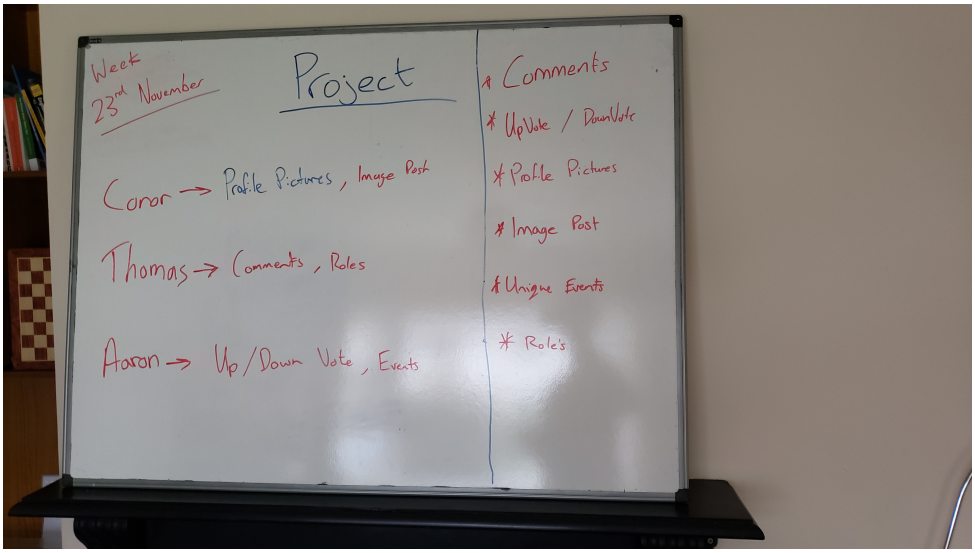
- For the development of this application, we decided the best approach to take was to use Agile. Agile is a methodology focused around iterative development. Using this method requires a team to work collaboratively and consistently. Before we started any development it was important to have a clear plan of what the features we wanted to incorporate and develop. Living in the same house proved useful in these times as we were able to have in-person meetings. Our first meeting we drew up a plan of how we were going to develop this application. Our first aim of was to agree on a development stack. After research into numerous development stacks such as the MEAN stack and MERN stack we decided on the MERN stack. We carried out extensive research into the factors that are common to all great websites, such as:

- **Functional** - Probably the most important aspect of every website. It must work quickly and as expected. Page loading times must be kept as low as possible, especially when working with large amounts of data - this was a big issue that we encountered when dealing with images.

- **Easy to Use** - User Experience (UX) plays a key role in helping visitors use, understand, and continue to use an application. If an application is difficult to use/understand, you can guarantee that a user won't stay longer than a few seconds. We made sure that the design we had created was user friendly and carried out blocks of black box testing to ensure that there was no confusion when using the website. Any feedback that we were given on positioning of components and the fluidity of the features we tweaked so that the design would relate to a lot of people and required no learning curve as navigating around the website was familiar.

- **Robustness** - The application should work when used on different devices, the page should automatically resize and adjust when used on different sized screens.

  With our choice of stack agreed upon, the next step was dividing the roles in the group. Conor and Thomas have an interest in back-end development and Aaron was keen on developing the Front-end. The first meeting was beneficial as we had a concrete foundation to build upon.
  We decided the best approach to development was to work on two - three features a week. We set up a whiteboard with a list of features to be developed, every week we would divide the features between the three of us. An example of our work process is shown below.



Each week we would have a meeting detailing the work done and evaluate the features developed. This was an important process as we got to discover what we could improve and what works well. If something needed improving we would put it back on the list for future meetings. If a feature was functional and proved to be efficient we would cross it off the board.
These meetings were also valuable for tracking bugs throughout the development process. If a team member encountered a bug or an issue we would write the it on the board and discuss it at the next meeting. Bug tracking was a process that requires good communication and teamwork as for an issue to be overcome we had to share solutions and

potential bug fixes.
Throughout the development of this project, we have used many collaboration tools including GitHub, Slack and Discord. While using these tools we were able to communicate new ideas and solutions to problems we may have been facing. GitHub helped us to track our progress and any mistakes that were made we could easily rollback commits. As we are all working on this project remotely, VOIP was incredibly important as we could communicate instantly, raising any issues we may be having and coming to brainstorming a solution.

3.0.1 Testing

Testing was a vital part of the development process. Using Agile required us to test after each feature was incorporated. We used many testing methods during the process of this application the main testing methods used were unit testing, smoke testing ,regression testing and stress testing.

- **Unit Testing** is type of software testing where individual components are tested during the development phase.
  We decided as a collective that it would be best if these tests were carried out at the end of the week. This was decided as features would be implemented at the end of week. If a component failed one of these tests we would work together to get the feature up and running. Carrying out unit tests provided our application with more efficient components and also saved us time.

- **Smoke Testing** is a type of software testing where tests are carried out to ensure the most important features work.
  We carried out smoke testing once a major feature was added. One of these tests carried out was to ensure our feed was working correctly. The feed was an important feature, this required us to test it quite frequently. As a result, the feed became more efficient and improved with every test.

- **Regression Testing** is used to ensure that no recent changes have altered previously working features.
  Regression testing was carried out by the entire group before we pushed to Github. If a member of the team was developing a feature, before this feature was to be implemented they would be required to carry

out a test to ensure this feature did not alter a preexisting component. Regression testing resulted in our application being more efficient.

- **Stress Testing** is used to determine the stability of an application by testing beyond normal operational capacity.
  Stress testing was vital in the development of our application to ensure our application did not slow down when put under stress. This was best displayed when loading large amounts of images. Without stress testing and experiments our application would not be as efficient as it is today.

- **User Acceptance Testing** is carried out on users to test the software to validate that it is performing according to the required real-life scenarios.
  We carried out this testing by allowing our friends to demo the website. This test was crucial to understand if our website was up to standard of other social media platforms. From this testing phase we learned more about how our website behaves under control of end users. In doing so we were able to adjust and tweak minor details to ensure our website met a high standard.

Chapter 4

Experiments

We tried out many of different experiments throughout the development of our project to improve its user experience and run-times. Between Front End and Back End we tried out many different design patterns and user interfaces that would suit our project and through trial and error found the best suitable solutions to suit our needs. We were able to utilize Googles Lighthouse performance testing tool, and also the Performance tab in Chrome DevTools to examine our load times.

We faced issues while trying to load a large amount of images on a users feed and decided to experiment with ways around this bug. We tried reducing the amount of imported packages and keep the project package small so that it would be less pressure on the website when running. This clean up didn't seem to make any improvements on our website speeds and load times, but was still a successful experiment as it tidied up our code and removed any unnecessary and unwanted external packages.

Initially, the database models were extremely inefficient as we had entire objects being stored within one another. For example, the society model was storing the entire list of users within that society as user objects which contained unnecessary information like images, names, date of birth, etc [4.1].

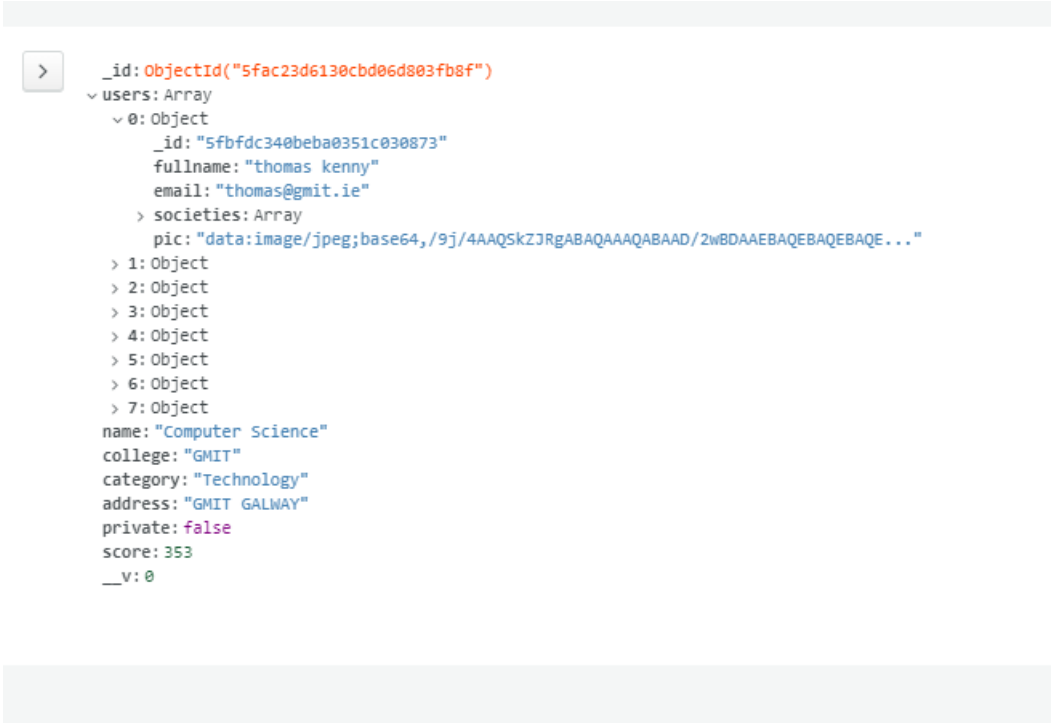


Figure 4.1: Inefficient Society Model

We then vastly reduced the amount of data shared information between collections within our MongoDB database, instead of passing all of the posting users information we only shared the ID keys which would give us full access to the details of the posting user and the posts information. Changing this structure within the communication between our collections saved us many resources and storage space within our database. This only marginally improved load times but was a success as it made our website perform at a higher speed than previously, and we learned a new solution to sharing information within our application which could be applied to many other features and functionalities.

However, even after optimizing the design of the database and making sure that no model contained unnecessary data, the website was still performing extremely poorly on Lighthouse scores [4.2]. As you can see from this figure everything except for performance was scored highly, and as the Chrome Devtools only analyze network requests and front-end components, we presumed the problem was within the structure of the database and as we had already restructured the models, we determined that the queries that were providing data to the get requests were the problem.

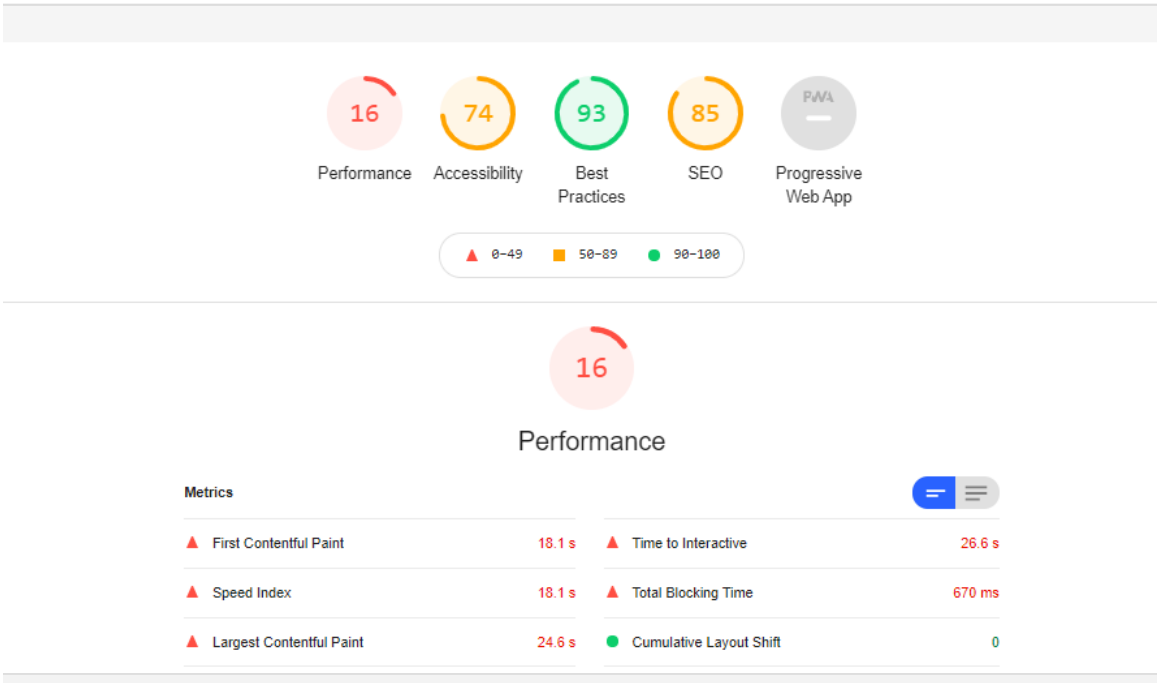


Figure 4.2: Poor Lighthouse scores

After analyzing the performance of the website using the Performance tab in Google Chrome DevTools tool, which allows you to view network requests, we evaluated that get requests for things such as the discussion feed, society list, or any request that required a large amount of data to be fetched was drastically slowing down load times [4.3]. From this figure we noted that the request for discussion feed data, which included all images, comments, title, etc, was taking significantly longer than other requests. After analyzing the queries, we noted that every field in the *Societies* collection were being returned in the request. And after some requirement analysis, we determined that only a handful of these fields were actually being displayed in the discussion feed component. We then parameterized the database queries, so that the requests could pass in a string with only the required fields for each request, and also a limit variable that limited the amount of documents returned from the query. This greatly increased the flexibility of the queries, as before we had a whole host of overloaded methods for getting different data from the database, and now we only needed a handful of queries to perform the same operations. This increased the performance of the website massively, as the discussion feed was now loading eight times faster, and all other requests were loading much faster too. This also greatly improved Lighthouse performance scores [4.4].

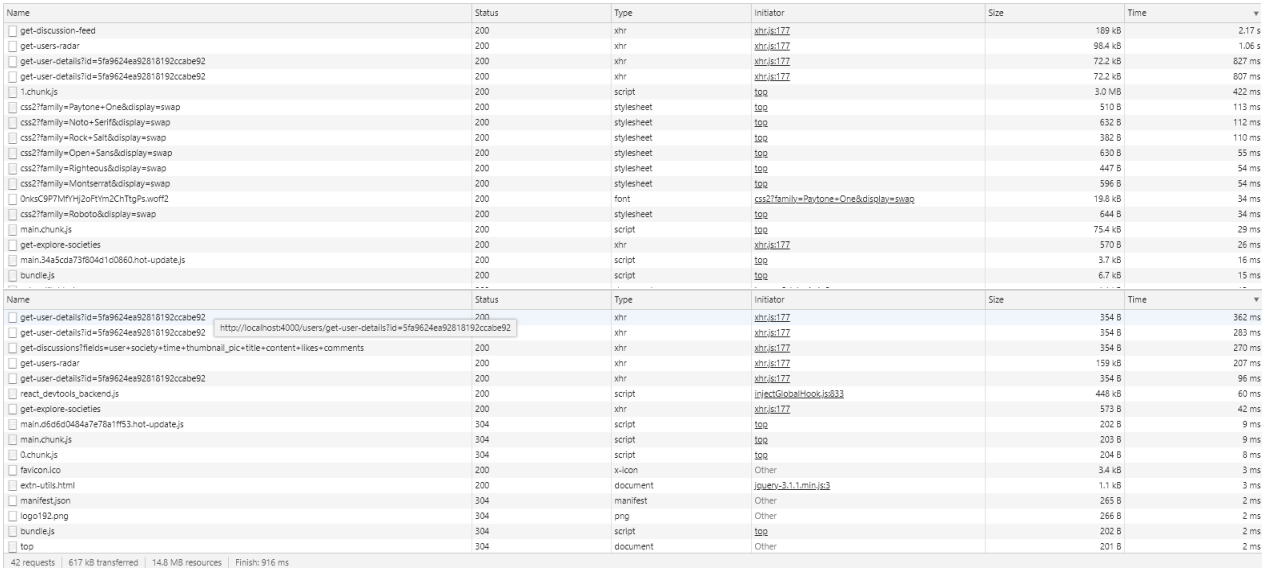


Figure 4.3: Slow Requests (Top) vs Improved Requests (Bottom)

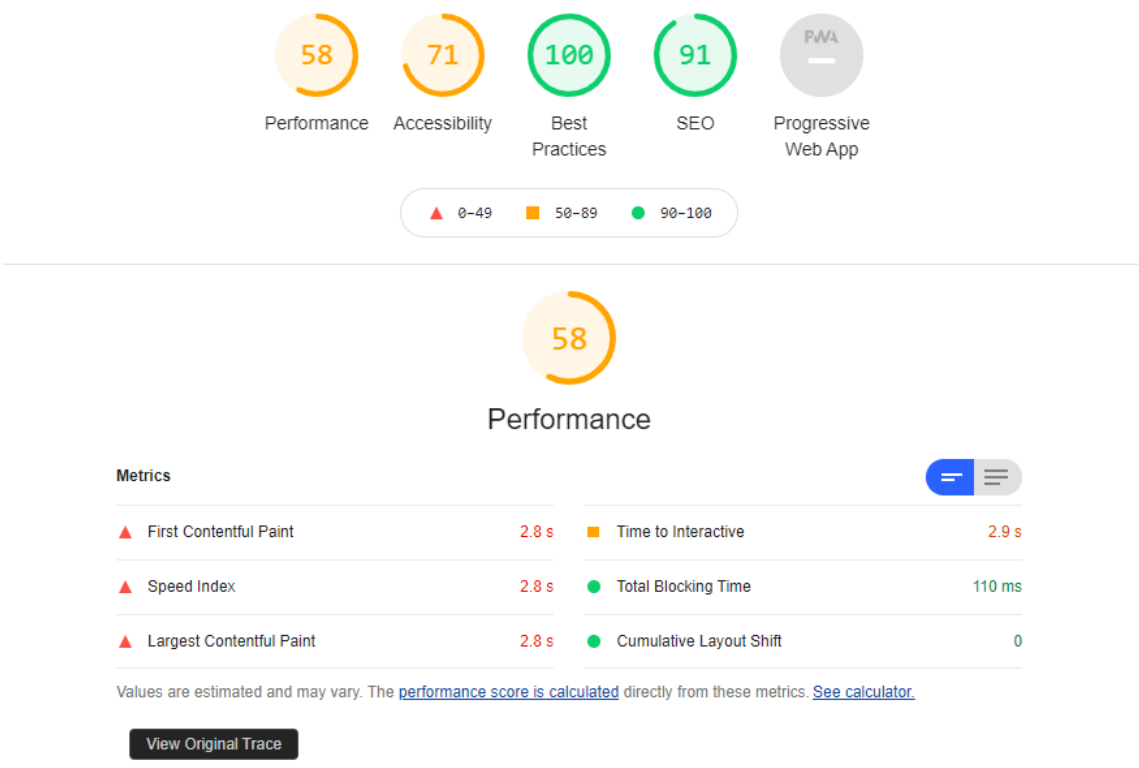


Figure 4.4: Improved Lighthouse Performance

However, this wasn't the entire issue. Although the performance was better than before, there was clearly still more problems at hand as it was still quite low comparative to other websites, such as Facebook which had a performance of 80 when analyzed with Lighthouse. Through much deliberation and many days of discussion, and after looking through many forums and articles for building fast React applications, we determined that the storage of large photos was slowing down the load times dramatically. We found that the best practices which are being used in industry is to store the large photos on an external server such as Amazon AWS and link them back to our application. This helped us to achieve satisfying results as our applications load times improved dramatically and our Google Lighthouse Performance was showing better results.

- Website Runtimes
- Responsiveness
- Speeds and Performance
- Loading in images via MongoDB/Amazon
- Front End
- Colorways
- UI
- FIGMA, WIREFRAME
- Design Decisions
- Icon imports, Boostrapping

Throughout the development of this project we tried out many different User Interfaces and Front-End designs to try and find which was the most suitable layout for our project. We used software such as Figma and Wire framing to design and test our designs which we could implement, once we had three official designs we came to a conclusion on which one would be most suitable for our project to match its features.

Chapter 5

Technology Review

The development stack that we chose for this project was the M.E.R.N stack. This development stack consists of MongoDB, Express.js, React,js and Node.js. We chose this stack as it was best way for us to spread the work load throughout the group as it touches all areas of website development such as front-end, back-end and databases. The M.E.R.N stack is a highly used development stack within industry and across many well respected applications we use today. React was the best choice JavaScript library for us to use as it is highly documented online and is backed by the world leading social media company Facebook. We were able to use many resources throughout the development cycle such as Reactjs Documentation and StackOverflow.

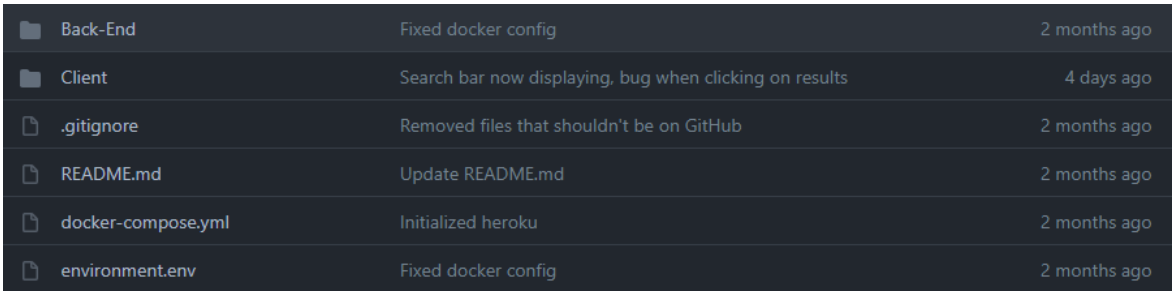
React has become a leading member of the JavaScript libraries which are extremely popular today. The main uses of React are to allow developers to created a large scale application in a fast, scalable and simple way. React is considered one of the easiest JavaScript libraries to learn as to start developing you only need basic knowledge of HTML and CSS, there are no limits when it comes to React as they are constantly updating their library and exploring new techniques for developers to utilise.

MongoDB is an extremely simple out of the box database which is also highly used today. The ability to simply store data within collections inside of a MongoDB database and to receive the data within your application allows developers to move quickly. MongoDB has a user friendly interactive database where you can easily examine the data which is being passed and stored inside of your collections.

Chapter 6

System Design

The structure which we have implemented into our project was by dividing up the Back-End and Client (containing Front-End components) into separate folders. This way we could easily distinguish which areas we were working on as both folders contain a large amount of scripts which relate to their desired areas. We decided to structure our project this way as from our research this seemed to be the most standard and mainstream structure that is being used within the industry. Inside of each folder the command 'npm install' must be used to install all the project dependencies, previously we had the Server and Client combined together within the same folder. This proved to be quite messy and meant the amount of dependencies being installed at the same time would slow down the process of development. By dividing them into separate folders it meant the members working on Back-End content did not need to install dependencies which were being used by the members working on the Front-End components.



Class Components

All of our JavaScript files have been implemented using class components. The reason as to why we chose Class components instead of Functional components was because we were dealing with many state changes within our

application and needed to utilise the support and logic that class components provide. The main differences between the two types of components are their syntax, Class components were the most familiar to us as it has a similar structure to Java.

Heroku Deployment

Currently our project is being hosted using Heroku. This is the best option for our project as Heroku allows you to host up to five applications free of charge...

Displaying Lists and Feeds The displaying of lists within our application allowed us to re-use a lot of components...

Chapter 7

System Evaluation

Chapter 8

Conclusion

Overall we have achieved our goal of developing an application where users could share opinions and join communities based off of their common interests.

Ultimately, working as a team to produce this project has helped us to develop and improve on our team working skills and problem solving skills ...

Chapter 9

References

Bibliography