Techbook - Social Media for the Tech-Savvy

Kevin Barry

Cian Gannon

B.Sc.(Hons) in Software Development

April 26, 2019

Final Year Project

Advised by: Dr John Healy , Dr Martin Kenirons

Department of Computer Science and Applied Physics

Galway-Mayo Institute of Technology (GMIT)

Contents

List of Figures

About this project

Abstract Social media today plays an expanding significant role in society, the information technology industry and the field of computer science. The use of social media is a hot topic for many organizations, with the aim to identify approaches in which companies can use applications to increase profits and grow product awareness. On a day-to-day basis, users from across the globe are becoming increasingly frustrated, wasting valuable time, scrolling through irrelevant content while companies are wasting money advertising to users outside their market. In order to achieve the optimal benefits from social media, for both users and businesses, the development of these technologies require approaches that focus on specific human interests and values.

This project aims to deliver a solution by developing a platform with the goal of delivering a social experience that targets a specific user base. As the authors are in the field of computer science the focus of the content will be to appeal to the tech-savvy user. The proposed solution will be a web application that will offer a unique online community to users and businesses interested in technology.

Authors This project was developed as a 15 credit project by Kevin Barry and Cian Gannon, final year students of Software Development at Galway Mayo Institute of Technology.

Acknowledgements The authors of this project would like to acknowledge and thank the project supervisors Martin Keniron and Dr.John Healy of GMIT for offering their time and advice throughout this project.

Chapter 1

Introduction

During our first three years in GMIT (Galway Mayo Institute of Technology) we were taught a broad range of topics from hardware to software. This was done to get us ready for whatever facet we chose. A major part of this project is to show what we have learned and put it into practice.

In late 2018, in the first semester of our fourth year, we were told we would be doing a group project over the two semesters. So in October 2018 Kevin Barry and Cian Gannon decided to form a group and started brainstorming ideas in order to start work early. Bringing those ideas to our project supervisors and getting their feedback on our idea we were able to form the idea for our final year project. With input from supervisors and using our own ideas, we were able to move forward with the technologies that we would use as our foundation. This was an extremely important decision for us as we wanted to pick relevant technologies and keep the scope within the scope of the level 8 course.

In our first week of the first year, our student union representatives created a group page on Facebook. This was something which kept everyone in the course in contact with one another outside of college hours. Social media has rapidly expanded in the last ten years, where it's now quite hard to find someone who doesn't have some sort of social media account on one of the numerous platforms. Our world has become ever increasingly dependent on social media and the features it brings.

Looking at social media and how reliant our world has become on it, we wanted to understand it better and have a stronger grasp on the underlying technologies.

1.1 Project Objectives

As mentioned above, our main goals for this project were to increase our understanding of how social media sites operate, the technologies used to make them and by changing how social media is designed by focusing at a specific target group.

This project is divided into two parts, the research-based dissertation, and the applied project. We will be discussing the project in terms of the research behind it and the technologies used to build it. For this reason, we split the objectives into dissertation or applied project based. We defined the objectives at a high level allowing us to break each objective into smaller tasks for development.

The objectives set out for the dissertation

Dissertation

- ***Introduce the concept of the project***. We will provide the reader with an introduction that describes the concept of the project, detailing its inspiration and goals.
- ***Provide an understanding of social media***. Through extensive research, we will examine and outline the concept of social media. We will investigate a vast variety of topics ranging from how social media began, how it became what it is today and what we must do to stay relevant in the modern world.
- ***Provide an understanding of web technologies***. To date, the vast number of technologies available for web development is expanding at a rapid rate. Exploring a range of different technologies we will discuss the pros and cons and give an insight into why we felt our final decision was most suitable for our project.
- ***Describe the development of the applied project***. This dissertation aims to give the reader a comprehensive guide to the development process from the first steps of initial research to the end product. We will examine the approach of the team to the applied project, including methodologies and technologies used along with the design and evaluation of the system. To conclude we will evaluate the project and discuss any issues that occurred, how they were solved and what we would do differently in future development.

Applied Project

- ***Produce a simple, easy to use web application***. The project will implement numerous complex algorithms with a sophisticated API server and database at the back end. The final product must hide all the complexities from the user with a functional and appealing front end. Regardless of the user's technical abilities, the web application will be simple to use and navigate.
- ***Deliver a social platform for tech-savvy people that differs from the norm***. Changing the way social media applications are generally developed, we will take a different approach and aim to deliver a solution by developing a platform with the end goal of delivering a social experience that targets a specific user base. The final application will provide an immerse online community where like-minded people can connect, share posts and interact with each other by commenting and giving feedback.
- ***Dive into new web technologies***. For the purpose of expanding the team's knowledge portfolio and skill-set, the development will be conducted using new technologies that we previously have not explored. Maximizing the learning outcomes of the module by implementing modern practices to result in a streamlined up-to-date application.
- ***Complete the project collaborating as a team using an efficient and effective approach***. The project will be developed as a collaborative effort. Using appropriate tools, methodologies and industry standards available to us, the project will be developed with the aim of simulating real industry experience. With the intention of maximizing efficiency and effectiveness, we will work closely swapping ideas, solving issues and reviewing performance to produce the optimal outcome.

1.2 Metrics for success or failure

In order to create the application in a controlled manner and allow us to easily manage progress in the early stages of this project it was crucial that we outlined the metrics for success or failure. Doing so, enabled us to keep on track with the core focus of what we were attempting to achieve. The metrics we defined related closely to the objectives outlined in Section 1.1. The simplified list of metrics for the success of both a dissertation and web application perspective is as follow:

- *A concise, yet comprehensive dissertation which can be understood by anyone regardless of the initial knowledge of the technologies implemented.* To measure this, as each section of the dissertation was roughly drafted up, we released sections to friends and fellow students to read. We would then process their feedback and make alterations were required.

- *A Simple, easy to use, functional web application.* To measure this we followed the same steps as above, releasing beta versions to friends of different technological abilities, allowing us to receive feedback both from a technical and non-technical perspective.

- *A social platform that actually develops an online community.* To measure the effectiveness of the application from a social point of view. We kept a log of the user activities while monitoring the site. Thus enabling us to at certain time points be able to see users following more users and interacting with a larger reach.

- *Teamwork Collaboration* We felt from a project management perspective and by researching the importance of a good team in industry, that we would measure the success of the team. With each weekly team meeting, we would take time to reflect on how we resolved issues, while looking at the strengths and weaknesses of our collaborative efforts.

1.3 Outline of each chapter

This paper has been organized into different chapters. Each chapter contains different details regarding various aspects of the project. The following subsections will briefly outline each chapter.

1.3.1 Context

In *Chapter 2* we will investigate how social media has expanded to play a significant role in society today. We will research a wide variety of topics relating to the topic. Starting with *Early Electronic Communication* we investigate how the internet was formed, giving us the base to develop online social platforms. Examining social media from its earliest days we discuss the *Rise of Social Media*. Finally, this chapter concludes by discussing *Modern Social Media* and how to *Stay Relevant in our Ever-changing World*.

1.3.2 Methodology

Chapter 3 will explore the approaches followed to plan, organize, manage and develop the project. We will discuss the methodologies that were adapted and combined to complete the research and development of the project along with why they were implemented. This section aims to give insight to the reader how the project transformed from research to final software while collaborating as a team.

1.3.3 Technology Review

In *Chapter 4* we will cover the technical side of our project, looking back on the technologies that made up the final revision of the project. We will explain the different technologies we added and how they were implemented through the project. We will look over the web stack we used and the technologies we added in order to create a more robust and useful social media site. We will go over why we used the given technologies and the benefits we saw in them over others.

1.3.4 System Design

In this chapter, we will discuss the architecture and design of the **TechBook** system. Presenting code snippets and visual diagrams to help portray a basic understanding of the application design. The contents of this chapter will begin with a brief overview of the flow of the architecture followed by a more in-depth portrayal separated into the Data Tier, Logic Tier and Presentation Tier.

1.3.5 System Evaluation

This chapter will evaluate the software developed in the project. We will evaluate the system in the areas of robustness, testing, and scalability. We will measure the results of the system against the objectives specified in the introduction. It will also highlight the limitations of the software and analyze where there are opportunities to improve the approach and technologies used,

1.3.6 Conclusion

To conclude we will briefly review the overall rationale and goals of the project. Highlighting our findings from Chapter 6 *System Evaluation.* Giving a final analysis, we review our discoveries gained from research and the new

skills acquired as a result. Finally, we will finish on a positive note with a brief discussion of the team's experience of the project.

1.4 Requirement Specification

Below is a list of the requirements of the project:

User Requirements

- Must be able to log in and log out.
- Log in will be persistent throughout a session.
- Must be able to register an account.
- Must be able to edit account details.
- Must be able to upload and edit profile follow.
- Allow user to follow other users profiles.
- Allow user to be followed by other profiles
- User can upload a post.
- User can comment on posts.
- Users credentials must be stored safely and encrypted on the database.

Chapter 2

Context

People have always wanted to stay in contact with one another. Since the introduction of the internet it has never been so easy, at first we used primarily email for communication with one another over the internet, but this was a technology adapted from letters. Early social media examples can be found in early instant messengers like MSN Messenger [1] which allowed the user to instantly communicate with one another across the world. But instant messenger platforms would be rapidly be replaced by other more robust sites such as Facebook(2004), Flickr(2004), Bebo(2005) and MySpace(2005) [1] to name a few. These sites revolutionized how we communicate today by giving us an online presence that others can use to find and communicate with us.

2.1 Early Electronic Communication

Early on, when the internet was just starting off it was used to connect universities and colleges together and was known as the ARPANET [2]. ARPANET allowed 3rd level institutions to communicate with one another through this new means of communication by linking their computers together to create the first usable computer network over a long distance. This evolved quickly, adding more universities, colleges, and research centers. This early form of communication was used to share research data between these institutions, which was seen as a great tool for the scientific method of sharing research and reading others in order to come to a conclusion. [3]

ARPANET quickly evolved into the internet we know today. As the internet progressed from a relatively small network between 3rd level institutions and research centers, to slowly entering peoples houses where everyone could connect to this one network. Early internet used telecommunications lines and shared bandwidth with phones. Although this was a great way of getting

the internet into peoples houses quickly by using the infrastructure already in place, this created a problem where you could either use the phone or the internet as you had to use the phone line to dial up the internet. This created a problem where the speed was too slow and any interruption such as using the phone to call someone. This stage of internet infrastructure was called dial-up. Dial-up was extremely slow with a maximum speed of 56Kbit/s, this slow speed and shared cable with a phone line made instant messaging slow and not very viable. For example, a 2GB file that today is a relatively small file would take 3 and a half days to download without interruption at the maximum speed and the maximum speed was rarely met and retained for most users. [3]

During this era emails were the main method of communication, it was text only and didn't require the user to be on their PC when the user sends the email. Email during this time was easy for users to understand during the early stage of home computers as it was just an electronic letter or e-mail. This allowed people of all ages to quickly grasp this new technology. And although emails would last well into the present their preference to communicate with friends and family would be replaced, and emails would be used in a more professional environment such as 3rd level institutions and the workplace. [4]

2.2 Rise of Social Media

Social media started in the late '90s but was a very early stage and was missing features that we consider standard today. These social media applications were centred around messaging applications such as AOL Instant Messenger(1997), MSN Messenger(1999) and Yahoo! Messenger(1999). These applications were extremely popular among the younger population replacing email for the most part for this younger generation. These applications created a standard among social media sites to come by offering an instant messenger to entice on their application to move onto the new competitors.

In early 2000 is the year social media sites we know today started out and came into their own. Sites such as Facebook(2004), Reddit(2005) and Twitter(2006) all launched in the early '00s and are still very active today. These sites quickly gained an audience from people who wanted a more involved experience. These sites offered profile pages the user could customize and have users interact with one another. Facebook, for example, is the most popular social media site today (2019) expanding year over year in terms of users. [1]

Facebook has remained relevant through expanding it's market and lan-

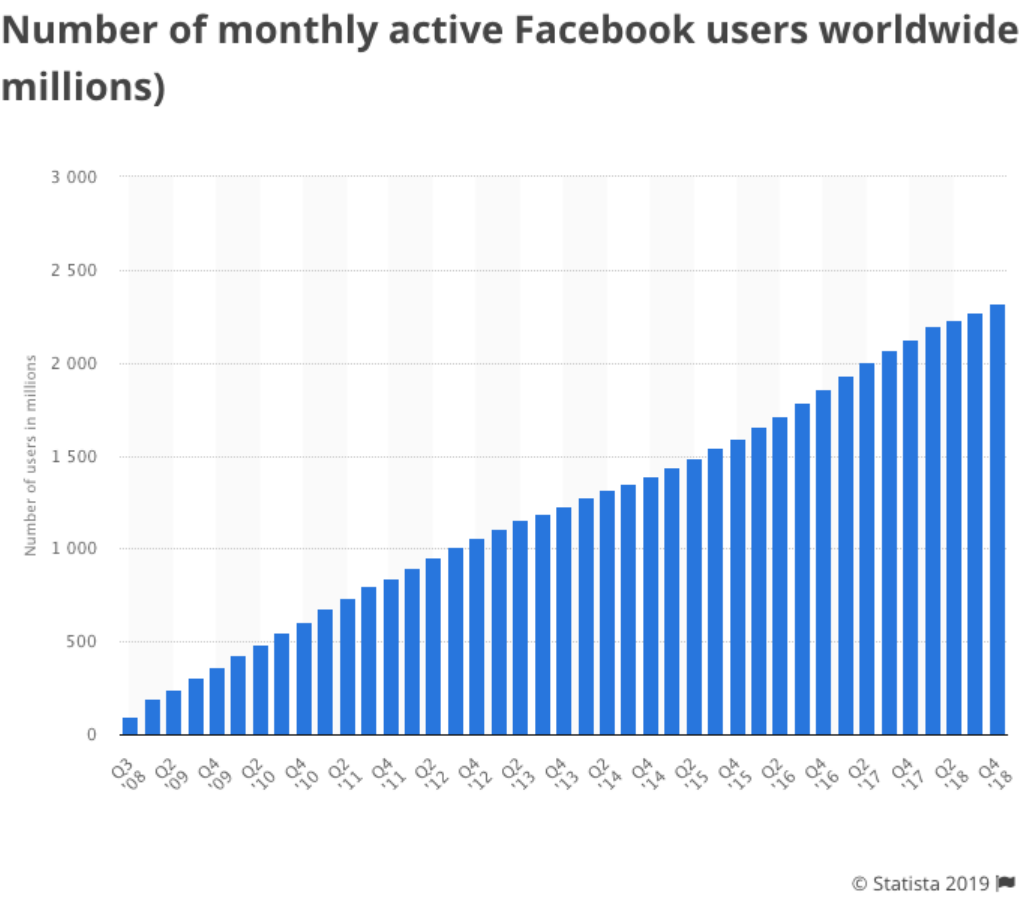

Figure 2.1: Facebook Growth

guage options to make it more inclusive for people outside the western hemisphere where it first took off. which allowed it to incrementally increase its user-base year over year.

Sites early on like MySpace(2003), Flickr(2004) and Bebo(2005) all gained traction but rapidly fell out of use once more competition came into the market such as Facebook and Twitter. These early social media sites were generally popular with the younger generation but as more robust social media sites came into the light they fell out of use. Facebook broke into the market by bringing features such as social games that brought a much younger audience onto their platform. Once this younger audience brought their friends which started a domino effect, and slowly these older sites started being abandoned as their style felt older, and they didn't differ enough to survive into the modern age. [5]

2.3 Modern Social Media

Modern social media is constantly changing and is a market of ”innovate or die”. Social media sites are always trying to innovate in order to keep the audiences they’ve built over the years. An example is when Snapchat came into the market Facebook tries to create its own version of Snapchat stories in order to keep users on their platform.

Modern social media sites have their own perks that keep people using a different variety of social media sites and applications. Facebook is more of a personal social media between people you know or have met. Twitter is used to keep up with people you would like to know about such as celebrities. Reddit is a post aggregator that is more of a classic forum from the early days of the internet. Instagram is a picture sharing application where you can follow friends and celebrities alike. All these social media sites are different enough and offer a unique approach that they have been able to survive together.

New social media sites that try to break into the social media sphere, need to be new and interesting. Social media apps such as Snapchat broke into the market by offering a unique take on when comparing them to their competition. They offered a completely different take with their ’stories’. Snapchat stories are short videos or pictures that only last 24 hours before they disappear, individual ’snaps’ also have a time limit where they can be viewed before they disappear.

2.4 Staying Relevant in our Ever-changing World

”Innovate or die” is a common industry phrase which year over years rings true. Social media site are always innovating in order to retain its user-base and expand it. Facebook, for example, is constantly adding to its feature list such as adding a story list which mirrors Snapchat’s story function where the user can post an image or short video which will disappear after 24 hours. Facebook is also constantly expanding its language base in order to keep expanding and making it easier for more and more users to the user its platform.

Facebook is a good example of a social media site which has stayed relevant due partly because it is constantly innovating and acquiring companies which make its platform more enticing to users. Facebook acquired ”Face.com” in 2012 for $100 million which gave Facebook access to their facial recognition software so Facebook could auto tag users if they knew the user was in the photo. Facebook also acquired Instagram for $1 Billion

in 2012 which is a picture sharing social media site which Facebook saw as an upcoming competitor. Facebook has even expanded outside of acquiring companies to help increase Facebook's user-base expand by acquiring companies such as Oculus VR for $2 Billion which is a VR headset development company responsible for the very popular Oculus Rift. Even though the Oculus Rift has nothing to do primarily with social media, Facebook has seen the technology and its future potential so it acquired the company to remain relevant well into the future. [6]

Chapter 3

Methodology

In this chapter, we will explore the approaches followed to plan, organize, manage and develop the project. We will discuss the methodologies that were adapted and combined to complete the research and development of the project along with why they were implemented. This section aims to give insight to the reader how the project transformed from research to final software while collaborating as a team.

Throughout our four years at GMIT a major emphasis was always placed on the importance of software development methodologies and the importance of choosing the most suitable methodology for a given project. There are numerous methodologies that have all been extensively explored such as Waterfall, RAD (Rapid Application Development) and Extreme Programming to name a few. For this project, we chose the main modern leader also based on the methodologies used by our employers, an approach known as Agile programming.

3.1 Version Control

In regards to version control with a focus on collaboration we decided to use the online GitHub Source Control Software. GitHub is a development platform allowing users to host and review code while also functioning as a simple yet powerful project management tool. Allowing the team to collaborate and support every aspect of the project management in one place it was an easy decision.

Figure 3.1: The Agile Cycle

GitHub enabled us to set up a team organization that could store both our dissertation and our applied project repository's. We took full advantage of the many features that Github has to offer. The main function of Github allowed us to commit our work to the repositories and merge changes into the existing project. Another effective feature of Github is the project section, which is used to optimize project management. With a built-in Kanban board, we were able to track our progress while also logging any bugs or issues that arise *(more details on this are given in Section 3.2.1)*. In respect to Software Development, another impressive feature of Github is the ability to create branches. Essentially a branch is a pointer to a unique set of modifications to the original "Master" branch and is given its own branch name. Creating separate branches allows us to work on features and experiment with different functions without having an effect on the main project. When each feature is tested and complete, Github simple allows us to *merge* our new changes back into the main *master* branch.

3.2 Agile development approach

Agile Software Development is a concept that allows a product be delivered to the customer in incremental releases while also being highly flexible, allowing requirements to change and the scope of the project to increase without major consequences on the design or current task[7].

We chose Agile as it stood out and had many benefits to our own project:

- Develop small, incremental releases
- Active user involvement
- Evolving requirements
- High level requirements

Figure 3.2: The Agile Cycle

Firstly as we were exploring all new languages, frameworks and libraries Agile allowed us to easily continuously integrate new features as our knowledge portfolio grew. Another major advantage was that this approach allowed us to have a working model every week rather than releasing the project in one release as would be in the Waterfall methodology. This alone gave us the opportunity to review each week's progress while also receive continuous feedback from our mentors.

From the outset of the planning phase, the main requirements and features of the software were identified. Subsequently, we broke these features down into subtasks which enabled complex tasks to be completed with simplicity. In the following subsections, we will explain how using two prominent features of Agile, namely Kanban and Sprints we could track our development and monitor our progress.

3.2.1 KanBan

The Kanban Method is a means to design, manage, and improve flow systems for knowledge work. The method also allows organizations to start with their existing workflow and drive evolutionary change. They can do this by visualizing their flow of work, limit work in progress and stop starting and start finishing [8].

The Kanban Method gets its name from the use of kanban - visual signaling mechanisms to control work in progress for intangible work products. In simplicity, Kanban is based around a Kanban board featuring Kanban cards (sticky notes). Thus allowing the full team to visualize crucial project information such as what tasks need doing, what tasks are in progress (and by whom) and what tasks are complete.

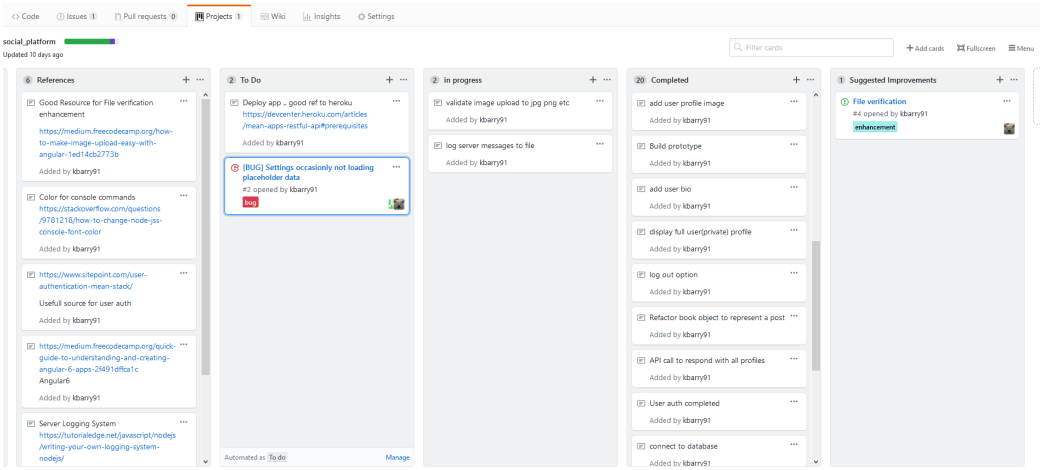

Figure 3.3: Github Kanban

Figure 3.3 above contains a snapshot of the projects Kanban board during development using the online project utility available on Github. This allowed us, as a team, to see at any point in time, what tasks needing doing, which were in progress or completed, while also tracking bugs and a section to propose improvements.

3.2.2 Sprints

The term Sprint is mainly used in the Scrum framework of the Agile methodology and for our project worked hand-in-hand with Kanban. A Sprint is a timed iteration of the continuous development cycle with a scrum being a meeting style discussion used to adapt feedback. During development, our default sprint duration was one week starting directly after each weekly review with our supervisor and ending at the following review where we would conduct a sprint plan. The aim of the sprint plan, is to as a team, reach two key decisions, **Sprint goal** what can be achieved in the sprint and **Sprint Backlog** the tasks to be completed in order to achieve the goal. During each sprint plan we would analyze the Kanban and discuss the following topics:

1. What did we complete in our preceding sprint?
2. What issues did we face?
3. Is anything preventing us from progressing with a feature?
4. Do any features need improvement?
5. What backlog items to be implemented in the next sprint?

6. What backlog items can we implement in the next sprint?

3.3 Selection Criteria

During the initial planning phases of the project, we evaluated numerous varying frameworks and technology stacks. From Java based Spring framework to Python's flask. As a team sat down to decide on the choice of technology and frameworks we would use. To do this we evaluated our project as a whole and began designing a criteria that we would base our final decision on.

The criteria for selecting an architecture was based on :

1. The type of web application we are trying to build.
2. Implementation of 3 tier architecture.
3. Server side must be easy to write and maintainable.
4. Languages that need to be improved or learned to use the framework.
5. Use of database alongside server.

Overall, based on the above criteria, we decided that the best solution for our project was to use the **MEAN** stack framework supported by Angular CLI. The Mean Stack provides multiple benefits which simplify the development of single-page web applications with 3-tier architecture.

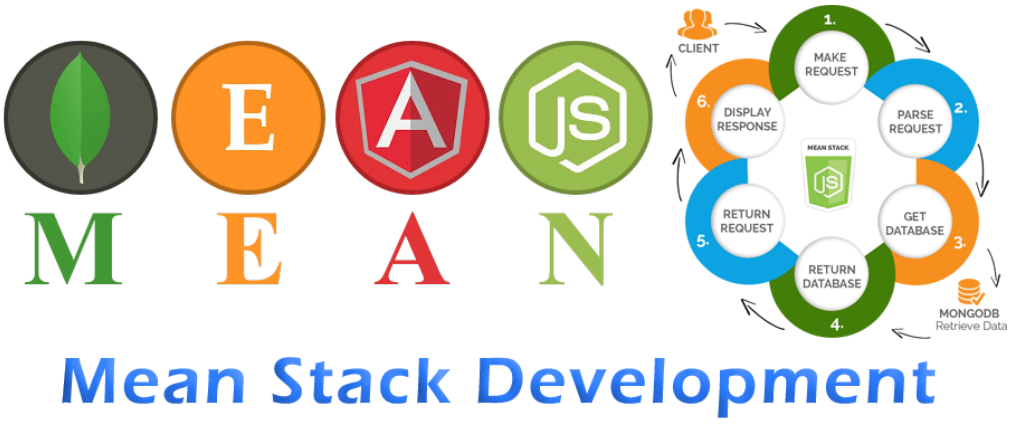

Figure 3.4: Mean Stack Development

A major benefit of the MEAN stack is that it would enable us to develop everything from the Angular client, to the Node server API all using JavaScript libraries. This allowed us to gain vast knowledge and increase our skill set in all areas of full-stack JavaScript applications. Another benefit of using MEAN stack is the increasing number of JavaScript libraries and Node modules available to aid development such as *Mongoose* for adapting *MongoDB* and *Passport.Js* for server-side authentication. In Section 4.1 we delve into more detail, giving a much broader explanation of the MEAN stack and the features that made it the go-to choice for Techbook.

3.4 Testing , Success and Failure Metrics

As defined briefly in Section 1.2 `Metrics for success or failure` our conditions were mainly focused and evaluated by user interaction. With the intention of monitoring success we decided on tests to evaluate each metric:

1. ***A concise, yet comprehensive dissertation which can be understood by anyone regardless of their initial knowledge of the technologies implemented.*** To be able to test the simplicity and understanding gathered from the reader of the dissertation we decided to review ten people. We decided to choose five people each of different technological knowledge to review sections of the dissertation. We split the groups into technical and not technical minded people getting the technical group to review Chapters *4, 5 and 6* whilst the non-technical group would read Chapters *1, 3 and 7*. Following this asking each tester to summarize briefly in around 5 sentences their findings and assess their understanding of the project on a scale of 1-10. We could then make adjustments based on the feedback received and repeat this step.

2. ***Simple, easy to use, functional web application***. As testing this ourselves we felt it would have too much bias as we would know how the system worked so we felt outsourcing testers again was the best solution. To test this we would host prototyped versions of the build throughout the development process. Only provided the testers with a link and no previous explanation on how the site worked. Subsequently, we would again have the testers report back to us with a rating based on the usability of the site from 1-10 and any pros cons they noted while using the application.

3. ***A social platform that actually develops an online community***. We have to be able to evaluate if the site actually functions as a social media platform. To confirm with the *Data Protection Act 2018* [9] we decided it would be best not to monitor users direct activity. Although it is hard to decide a concrete fixed value for what would be a success as the number of users subscribed to the service would be growing, the best way to test this is logging when users follow another profile and when they post or comment on articles. Integrating a server logging system into the project that does not store confidential information would give us a rough idea of the result.

4. ***Teamwork Collaboration***. We felt from a project management perspective and by researching the importance of a good team in industry, that we would measure the success of the team. In general, teamwork effort is something not easily measured in a definitive way. With each weekly team meeting, we would take time to reflect on how we resolved issues and while looking at the strengths and weaknesses of our collaborative efforts.

The graphs in Figure 3.5 briefly illustrate the iterative process taken for the purpose of pushing the above metrics to success.

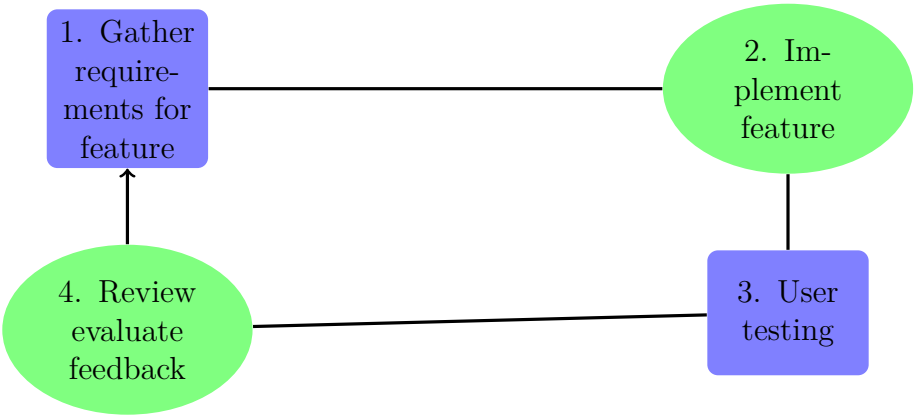

Figure 3.5: Success / Failure Test cycle

Chapter 4

Technology Review

This chapter will cover the technical side of our project, by looking back on the technologies that made up the final revision of the project. We will explain the different technologies we added and how they were implemented through the project. We will look over the web stack we used and the technologies we added in order to create a more robust and useful social media site. We will go over why we used the given technologies and the benefits we saw in them over others.

4.1 MEAN STACK/Framework

The MEAN stack is intended to provide a simple and fun starting point for cloud native full-stack JavaScript applications. MEAN is a set of Open Source components that together, provide an end-to-end framework for building dynamic web applications; starting from the top (code running in the browser) to the bottom (database).

The stack is made up of:

- MongoDB
- ExpressJS
- Angular
- Node.js

Early in development when we were brainstorming, we decided to do a full stack development. The next step after this decision was to figure out how to carry out this task. We looked into full development stacks such as the

MERN stack (MongoDB, ExpressJS, React, Node.js), MEAN stack (MongoDB, ExpressJS, Angular, Node.js) or doing a Java-based development stack (Spring boot, Angular, MongoDB). After lots of discussion and input from supervisors, we decided to not go with the Java-based development stack. We then took a deep dive into the difference between the MEAN/MERN stack to decide what route we wanted to take and start development as soon as possible. After looking into React JSX vs Angular HTML templating and experimenting with the two stacks we finally decided that the MEAN stack was more inline in what we wanted to learn.

Once we knew what stack we were using and how to use it in a basic form from our prototyping stage, we moved onto breaking the stack down to each component and getting to know each component better, so when full development starts we would have a better understand how the entire stack operates and interacts with each component. This gave us a greater understanding of how each component operated and allowed us to plan out how each component would operate from top to bottom. [10]

4.1.1 MongoDB

You may know what MySQL and other relational databases are. MySQL is analogous to a spreadsheet, it has name columns and rows of data. Every database "tool" shows the data as a spreadsheet (Microsoft Access). Data can be linked through special functions such as a "JOIN" mechanism to allow the user to built meta spreadsheets. [11]

MongoDB is a database and not a database tool (Mongo Compass), instead of a spreadsheet like MySQL it is more like a folder of documents. The contents of the "folders" are not checked to see if the new file differs from the current file format. This sounds like an inefficient system but all these folders are called collections in MongoDB. The collections are usually grouped by how the user would group MySQL tables. While MongoDB can contain keys like MySQL there is no built-in functionality in order to join documents together in order to retrieve related data in another collection.

Databases are classified into two categories of SQL and NoSQL. MySQL is an example of an SQL database, and MongoDB is an example of a NoSQL database. MongoDB is very reliable as its very hands off and gives the user the responsibility for how data is saved and read. If a user adds another entry into the collection that doesn't match the other entries MongoDB doesn't inform the user that the data is not in the correct format as collections do not have definitions and leave that responsibility to the user. The same goes for joining two collections together. The joining is left up to the user on how they carry that out, usually, the user would create an ID which is present in

both collections in order to 'join' them together.

MongoDB is so hands off about how users read and write to the database it stops the frequently used attack on SQL databases called SQL injection. An SQL injection occurs when the user directly interacts with the database which allows them to manipulate their input to test the database. MongoDB doesn't have this problem as its hands off the reading data to the user to the point where a search for that data will include the statement but nor process the statement as such. [11]

MongoDB uses the JSON standard to store data. The JSON data is then classified into a collection as we discussed above.

Example of how a JSON entry in a collection could look like:

```
{
    "username": "Smithy",
    "password": "smith123",
    "name": {
        "first_name" : "John",
        "second_name" : "Smith",
    }
}
```

4.1.2 ExpressJS

ExpressJS is a minimalistic web framework built for Node.js that waits for the browser to connect so that the server can send, process the request and respond with the relevant form (JSON, HTML, Raw Text, ETC...). Without ExpressJS the user must handle creating a server, handling routing all manually.

ExpressJS relies on APIs made available by Node.js. ExpressJS cannot exist without Node.js. ExpressJS is a thin layer over Node.js which makes developing servers and routes much easier. ExpressJS's main feature that makes it enticing to developers is how it serves dynamic content (Content that changes based on user requests).

ExpressJS also has the capability to server components from frameworks such as Angular, Ember and React and process requests made by those components to update their content. ExpressJS makes Node.js development easier, especially when creating APIs that front-end apps use. Express makes it easier to unify the front-end apps and the API.

ExpressJS was an easy pick for the project as it easily integrates Node.js and rapidly decreased the time it would take to get the server side API up and running. [12]

4.1.3 Angular

Angular is a front-end web framework built on top of JavaScript, it is used to develop single-page web applications. Single-page web frameworks (Angular, React, Ember) are websites which have all the functionality of a multiple page website without having the need to refresh the browser when moving from page to page. Single-page web development frameworks have become increasingly popular in the last few years due to how reactive they are. Pages react very fast and fluidly making user interaction with the website more positive than refreshing when moving to another page on the same website. [13]

Angular has been around since 2010 with the release of AngularJS. Later version came out and greatly improved upon the idea with Angular 2+ or Angular v2. This version would be improved upon and maintained by Google with the latest release being Angular 7 in late 2018. Despite angular being around for nearly 10 years, it has only been in the last few years where it has come into its own and seen real competition from other single-page frameworks such as React and Ember.

We chose Angular because of its integration with Node.js/ExpressJS which allowed us to get prototypes for testing the MEAN stack up and running very quickly and let us play around with the different functions it offered as part of its component-based setup. Within a week we had the entire MEAN stack in a functional state where Angular was served with Node.js/ExpressJS and then send and received data from the API served by the Node.js/ExpressJS server.

4.1.4 Node.js

Node.js is a cross-platform JavaScript run-time environment that allows developers to run JavaScript outside the browser. This framework is a huge deal for JavaScript. Node.js allows developers to perform actions with JavaScript on the developer's local machine like they would with other programming languages. What makes Node.js so beloved by developers is the built-in functionality of Node.js. Node.js has built-in HTTP function to allow Node.js to run HTTP actions all within the environment. This built-in HTTP functionality has made Node.js extremely popular and given the rise to isomorphic web applications and due to the fact that Node.js can run as a server back-end using JavaScript and have JavaScript on the front-end meaning there is the opportunity for the two to share code between them reducing testing and maintenance. [12]

Node.js is a great framework due to when paired with a front-end frame-

work such as Angular, and a database such as MongoDB. When using a setup like this the entire web stack is JavaScript-based and allows the user to easily manipulate data from top to bottom.

The reason we picked Node.js is because of its ease of use and how rapidly we could get a working HTTP server up and running. Node.js also made a lot of sense when paired with Angular which was the front-end framework we picked. Node.js makes developing HTTP servers easy, which allowed us to spend more time on setting up other features such as user authentication and the Reddit API.

Here is an example of all the code needed to create a simple Node.js server. This code will display ”Hello World!” to the user in the browser on port ”8080”.

```
var http = require('http');

//create a server object:
http.createServer(function (req, res) {
  res.write('Hello World!');
  res.end();
}).listen(8080);
```

Node Modules

Node.js allows developers to develop and use packages which provide the developer with more functionality.

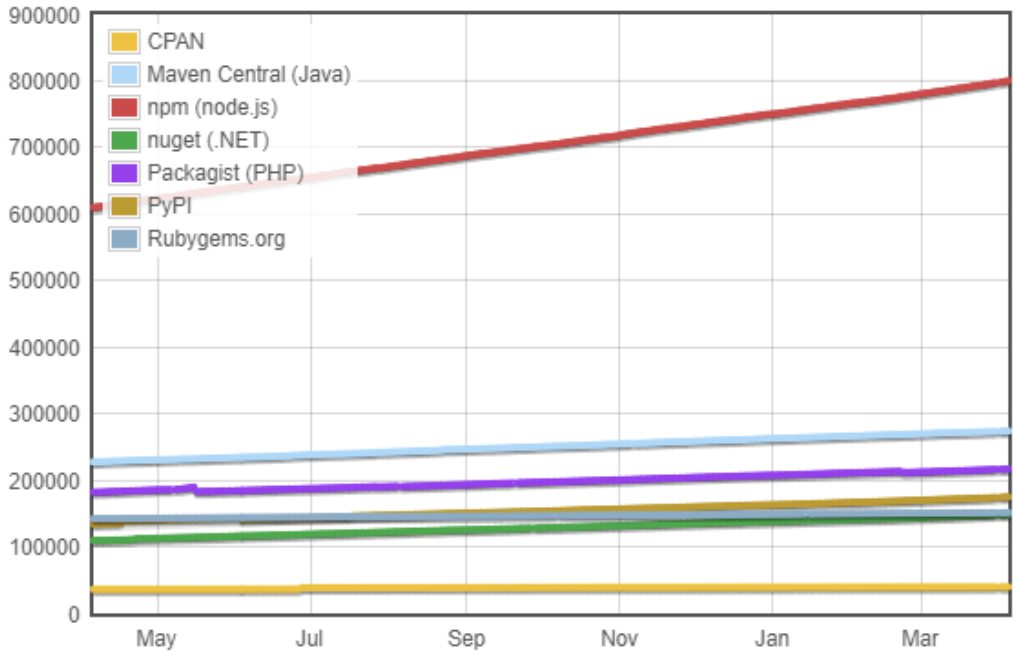

Figure 4.1: NPM Package Count

NPM is the world's largest software registry. NPM allows developers access to lots of packages for all sorts of development such as packages for Node.js, Angular, MongoDB and even developing Alexa skills. NPM allows developers to upload custom packages that they have developed and give access to other developers to use it in their system.

Node modules is a powerful resource with lots of packages that adds great functionality to the developer's project such as user authentication with Passport.js.

4.2 Deployment

For code deployments we looked at numerous cloud service provides to figure out what would be the best service to take use of for our project. We looked at Google Cloud, Heroku and AWS as the cloud service providers. Although these are only a fraction of the cloud service providers online, we found they were received very well by our supervisors as a way of deployment.

We ruled out Google cloud after lots of debate on how to move forward with deployment. The reason to remove Google Cloud first was we felt that we wanted to get a greater understanding of a different cloud service provider as we already had experience with Google Cloud. After ruling our Google

Cloud we were left with Heroku and AWS. After researching more into both options and figuring out what would better with our project we decided to go with AWS because of AWS's Elastic Beanstalk. AWS Elastic Beanstalk is a system provided by AWS for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers. Because of AWS's Elastic Beanstalk and us using the MEAN stack, it was an easy way to deploy our project, and an easier way to expand the project if we wanted to go commercial with the website. [14]

4.3 Standards

4.3.1 JSON

JavaScript Object-Notation is a specification for Serialization a data object. Serialization is a is the process that turns a price of code into a string of characters in order to transmit it. [15]

The basic idea is that if you want to transmit a piece of data that has attributes such as name, address, age, etc and turning it into a string.

An example of converting an object to JSON specification would be for example this object of the user.

```
class user{
    string username;
    string password;
    int age;
}
```

Let's say the user created an instance of this object and set the values to "Smithy", "smith123" and "25". To send this instance over to another machine in a standard the other machine we can't just send it in the raw text using HTTP so we need to serialize the data before transfer using the JSON specification.

The JSON string would look like this when the class is serialized:

```
{
    "username": "Smithy",
    "password": "smith123",
    "age": 25
}
```

Once the class is correctly serialized, the object can be added to the HTTP request to be sent over to the other machine. The JSON object would be added to the body of the HTTP request for the machine on the other end to retrieve and do with as it needs.

An example of it in the body would be:

```
{ "user": {"username": "Smithy","password": "smith123","age": 25} }
```

With this machine-readable string, I can now retrieve the data by deserializing the string back into an object. This all works because of the JSON specification which creates a standard for how to serialize and deserialize the JSON object. [15]

4.3.2 REST

Representational State Transfer or REST is a guideline for sending information on the Internet. It is an architectural system that has gained in popularity over the years. RESTful web services are a common way of accessing an online API. Users can make multiple different requests through one URL with certain parameters. The REST architecture design has an emphasis on nouns to tell what the resource the user is trying to access is and how the resource is going to be affected by HTTP modules. [16]

Let's say a developer is making a system that accesses a database that requires read, write, update and delete functions. A REST URL could be for example "/api/account". Using this URL we can create a RESTful resource that the developer has access to with all these functions.

Using the REST specification we can create these four functions in the one URL route or resource by using the different HTTP methods (GET, POST, PUT, DELETE) for each function. All four HTTP requests to do with a user account would use the "/api/account" route to read, write, update and delete the user's data.

The REST specification streamlines the development of a server API and makes managing the API on the server as well as the application using the API much easier.

4.4 REST API

The REST API we setup is intended to give access to all aspects of the project in a public format. The REST API is used by Angular which gives the user a visual representation of the data being sent and received by the

REST API. Some routes are protected by Passport.js which encodes user data used to access certain routes such as update profile picture or password.

A big benefit of the API is all functionality of the application is available from the REST API. This allows for us to if we decided to, add a mobile application or create a monitoring tool to see the data and analyze it. [16]

4.4.1 Security

Routes altering data (POST, PUT, DELETE) requires the user to be authenticated to prevent data being altered by another user. We authenticated certain routes such as create a post to check who is creating this post, and to link to their profile but also so only they can post in their name. Authenticating routes such as comment, post, settings, and following was a big part of the planning stage as we needed to figure out a way to stop intrusive behavior from outside sources abusing the API.

To add a layer of security to user requests and ensure no malicious requests are made by another user trying to impersonate another user we added a Node.js import called PassportJS. PassportJS is authentication middle-ware for Node.js. PassportJS allows

4.4.2 Swagger

Swagger is an open source software framework sponsored by SmartBear. Swagger is used to create, document and consume RESTful Web services. Swagger offers an easy way to document RESTful Web services by using Swagger documentation definitions and Swagger UI to document an API and give an example of how the API works. Swagger works by having the developer add comments similar to java-docs in order to document a route telling swagger the components of the route and the model it uses. Giving Swagger this information Swagger will use the data to create a JSON object with all the data about the API in a format that can be used by the developer for other means or to be used in conjunction with swagger UI, which we will discuss later.

Swagger.json

As we discussed above Swagger is used to document a route. But how does it do that? Swagger creates a JSON object which it hosts at a given route in order to be accessible by developers for their own purposes, or by using Swagger UI.

The JSON part of Swagger is where the swagger documentation is rendered into a machine-readable format.

```
{
    info: {
        title: "Node Swagger API",
        version: "1.0.0",
        description: "Api file for swagger"
    },
    host: "34.243.30.50:3000",
    basePath: "/",
    swagger: "2.0",
    paths: {},
    definitions: {},
    responses: { },
    parameters: { },
    securityDefinitions: { }
}
```

Above is an example from the hosted project on AWS. As we can see this is a very basic model for without any route or definitions. Swagger includes basic information about the API such as the host and general Swagger information such as version and title. Swagger includes the host and base-path used by the API, with a list of definitions and paths about each API route.

```
paths: {
    /api: {
        get: {
        tags: [
            "books"
            ],
            description: "Returns all books",
            produces: [
            "application/json"
            ],
            responses: {
                200: {
                    description: "An array of books",
                    schema: {
                        \$ref: "#/definitions/book"
                    }
```

```
                }
            }
        },
    },
},
```

An example of one of the paths contained in the paths object array. In paths we have a list of paths documented using Swagger, these paths contain information about the path such as the title in which the path could be classified under, the example above is 'Books'. With the title we can classify different paths that may handle something to do with 'books' and thus we may want to organize the paths under 'books'. The path also contains the schema used which is very important in order to understand how the API route operates. Does the path return JSON?, XML? raw text?

```
definitions: {
    book: {
        properties: {
            isbn: {
                type: "string"
            },
            title: {
                type: "string"
            },
            author: {
                type: "string"
            },
            description: {
                type: "string"
            },
            published_year: {
                type: "string"
            },
            publisher: {
                type: "string"
            }
        }
    }
},
```

Using the definition in conjunction with the path we can get a better picture of how to API route works and what data is needed for it to operate

correctly. The definitions contain definitions for each route and each definition inside definitions contains properties which describe the routes data and what type of data each property is.

Swagger UI

Swagger UI is a visual representation of the API documented using Swagger. Swagger UI is a great as it enables developers to save a lot of time in documenting an API, as it uses the Swagger JSON to visualize the API.

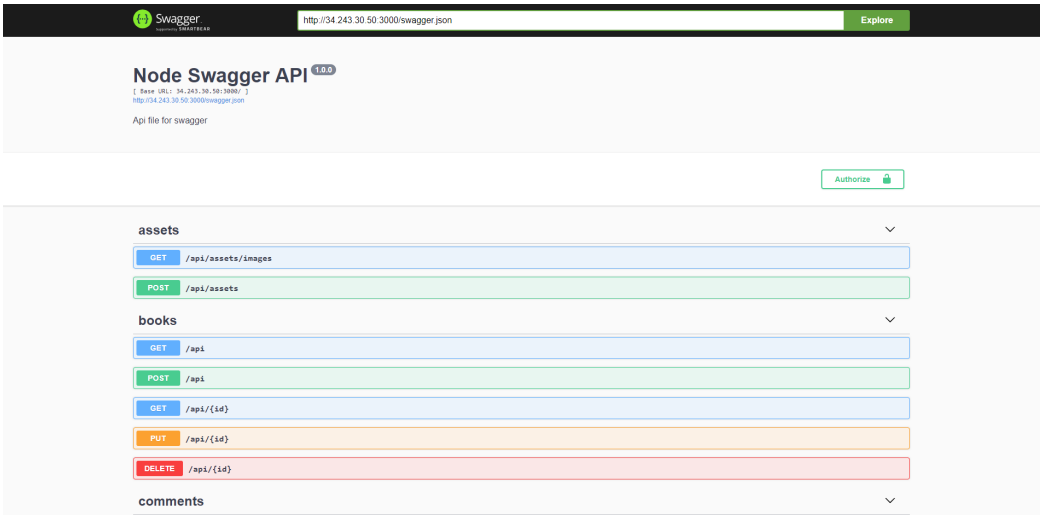

Figure 4.2: Swagger UI

Using Swagger UI we can use the hosted JSON from the Swagger Documentation to display the available routes in the API. Each route is categorized and sorted using the Swagger JSON. Each option under the category will use the path and definition of the Swagger JSON to create a visualized representation of the API and allow testing of the API.

4.5 Languages

4.5.1 JavaScript

JavaScript is the backbone of the web. JavaScript is responsible for front-end web development processes. Without JavaScript a web page would have no functionality to communicate with the server-side functions in order to save or read dynamic data. JavaScript runs the web, both front-end with JavaScript or TypeScript.

A very basic way to think of JavaScript is that HTML is your bone structure. It holds everything together with a solid foundation. CSS is your skin, it's what everyone sees and interacts with. So JavaScript is your brain, heart and other organs that make you, you. JavaScript is what makes everything function in an effective way.

Database

JavaScript is found in the database with MongoDB, which stores data using JSON (JavaScript Object Notation). MongoDB saves database entries using JSON to store elements of that collection. As we explained earlier what JSON is and how it stores classes into objects.

Server-Side

On the server-side we used Node.js. Node.js, as we discussed above, is what allows JavaScript to run outside the browser. JavaScript was used as the server-side processing which handled routing for the API and for the Angular front-end.

Front-End

The front end section of the web application uses TypeScript which is a super-set of JavaScript which we will talk about in greater detail below.

In the server-side API, handling routing for the web application.

4.5.2 TypeScript

TypeScript which is a super-set of JavaScript. JavaScript code is valid in a TypeScript environment, with some exceptions. A huge benefit over JavaScript is that TypeScript made it harder to create bugs and made JavaScript more manageable in larger systems.

JavaScript:

```
let name = 'TechBook',
users = 500,
isThreeTier = true;
```

TypeScript:

```
let name: string = 'TechBook',
users: number = 500,
isThreeTier: boolean = true;
```

The biggest change and most liked feature of TypeScript is variables are declared by using a variable type such as 'string', 'number' and 'boolean'. This is a big change over JavaScript where the variable type is interpreted as to how the developer manages the variable.

4.5.3 HTML

HTML (Hypertext Markup Language) is a markup language for creating web pages. HTML uses tags to define what sections of the web page do what.

HTML Example:

```
<h1> Title </h1>
<p> Paragraph</p>
<a href= "github.com"> Link </a>
```

HTML Rendered:
Title
Paragraph
Link

Every web-page on the internet is a hypertext document. A hypertext document differs from a raw text document by allowing images, formatting, and links. HTML is what holds all the components of a web-page together.

4.5.4 LaTeX

LaTeX is a text processor that is built around a markup language. Like HTML LaTeX uses tags to define a command or action to display text in a certain way. These tags work very similar to HTML where these tags are only seen on the source of the document and are hidden when the document is compiled into a format such as a PDF.

LaTeX Example:

```
\chapter{Technology Review}
\section{Languages}
\subsection{LaTeX}
Some text about LaTeX
```

LaTeX would then compile this text into a document, like this one, where it renders the chapter the section and subsection of that chapter and would compile the table of contents to include the new chapter and sections.

4.6 Styling and UI

In this section we will talk about the different technologies we used to develop the front-end user interface for our website.

User interfaces are an extremely important aspect of website design. As more people have access to the internet using different hardware and software to access a website, it has become harder to develop a streamlined user interface that works across all devices.

Things that can affect user interface development:

- Aspect Ratio
- Resolution
- Browser
- Mobile/Desktop

Creating an intuitive user interface that works across most devices is difficult because of the above reasons. Using different technologies we will discuss below often these technologies take care of many of these issues. As long as the developer keeps them in mind and tries to use the functionality they provide to make the website accessible to as many people as possible.

Mobile has become a major part of accessing the internet as smart-phones have become a bigger part in peoples lives and can be used anywhere. Developing a user interface that works on both desktop PCs and mobile smart-phones can be challenging. Developing a website that works on both devices was very important for us to keep up with modern trends.

4.6.1 CSS

As we discussed briefly in the previous section, CSS is the skin of the project. It's what most sites use to make their appearance more professional and not just a wall of text and buttons. CSS stands for 'Cascading Style Sheets. It's used by nearly all websites to improve their appearance and makes a website stand out from each other.

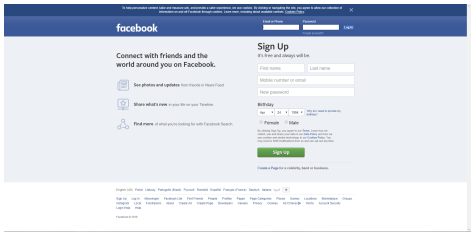

Figure 4.3: CSS Facebook

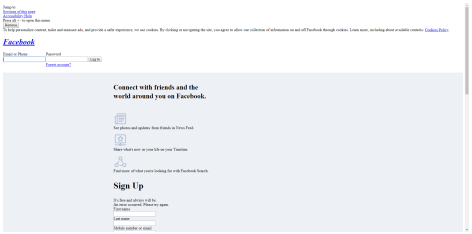

Figure 4.4: No CSS Facebook

As we can see above is an example of what the web would look like without CSS. Facebook would go from a modern looking website to a very basic text website with a few images spread around.

CSS is an extremely easy tool to grasp but takes time to master, especially with the different screen sizes, aspect ratios, and resolutions.

4.6.2 Bootstrap

Bootstrap is a pre-written library of CSS files that enables developers to rapidly develop the UI for their website. Bootstrap also allows developers to make their own CSS files and build on top of Bootstrap to refine a website design.

Bootstrap is great for quickly developing prototypes for websites as it takes a lot of the grunt work and allows the developer to concentrate on getting the functionality right. CSS offers a great way to style a website, but often it can be hard to get a certain function to work correctly. Bootstrap takes care of a lot of the functionality the developer may need such as a sticky navigation-bar. Instead of the developer making their own sticky navigation-bar using CSS and JavaScript, they can off-load that work to Bootstrap which will do all that using the specified Bootstrap class.

4.6.3 Angular Material

Angular Material is developed by Google and was developed to meet the demand for their new design spec 'Google Now'. Google Now laid out a grid-based website with animations and responsive cards, which was used until the service was discontinued.

> "Unlike real paper, our digital material can expand and reform intelligently. Material has physical surfaces and edges. Seams and shadows provide meaning about what you can touch" - Matias Duarte, VP of Design at Google

Angular Material was birthed from the now defunct service. Angular Material was made to create a Bootstrap-like alternative. Although Angular Material is an alternative to Bootstrap to develop front-end user interfaces, Bootstrap is often used in conjunction with Angular Material.

4.7 Other Technologies/Development environment

In this section we will go over the technologies we used to develop and test the web application.

4.7.1 Visual Studio Code

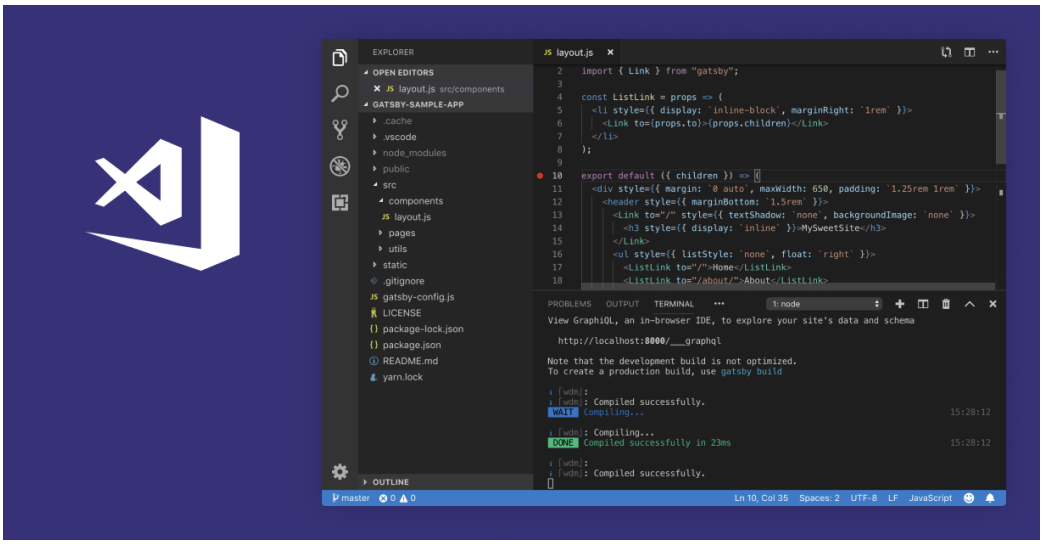

Figure 4.5: Visual Studio Code View

Use used Microsoft's Visual Studio Code as an integrated development environment as it was perfect for what we needed. We needed an integrated development environment that didn't just support one language. We needed one that supported, JavaScript, TypeScript, HTML, CSS, and LaTeX. Visual Studio Code offers specialized extensions that make developing with these technologies more initiative.

Extensions

Extensions for Visual Studio Code are add-ons developed by others to make the development environment in Visual Studio Code better for a specific task.

HTML CSS Support HTML CSS Support extension is just what it sounds like, it adds better support for HTML and CSS in Visual Studio Code. Although you can easily write code in Visual Studio Code for HTML and CSS this extension adds features that make the development of HTML and CSS like developing Java using Eclipse would.

- Class attribute completion
- Id attribute completion.
- Supports Zen Coding completion for class and id attributes.
- Scans workspace folder for CSS and SCSS files.
- Supports remote CSS files.

It's small features like these that make developing HTML and CSS a little easier, especially when developing a full stack website. HTML CSS Support extensions made developing with HTML and CSS feel more natural in Visual Studio Code.

JavaScript (ES6) Code Snippets JavaScript (ES6) Code Snippets is an extension which adds shortcuts for common JavaScript and Typescript functionality such as for loop, console log and class constructors.

Supports:

- JavaScript (.js)
- TypeScript (.ts)
- JavaScript React (.jsx)
- TypeScript React (.tsx)
- Html (.html)
- Vue (.vue)

The extension supporting both JavaScript and TypeScript was what sold us on the extension as those two languages made up the majority of our code-base.

Using the extension is very easy once you know a few of the shortcuts and how to use them. An example of use would be typing 'clg' and pressing TAB which would convert 'clg' to 'console.log();'. The extension made getting something setup easy such as 'fof' and pressing TAB becoming 'for(const item of object) '. Although very basic, it made writing small code snippets easier and more in line with IDEs such as Eclipse with 'sysout' and then pressing CTRL-SPACE which produced 'System.out.println'.

LaTeX Workshop LaTeX Workshop was used early on in development. The extension provided live PDF view to see how changes looked and has came with quality of life features such as shortcuts.

LaTeX Workshop made developing LaTeX in Visual Studio Code easier with the features it provided.

4.7.2 Browsers

We used two primary and popular browsers during development. We didn't want to just use one as a feature might work for one but not the other. This wasn't a big problem as Angular handled much of the primary features with each browser, the bigger problem with trying to get cross-browser support was CSS settings and how each browser would interpret them. We used chrome, Internet Explorer/Edge and Firefox as they had the largest market share.

Chapter 5

System Design

In this chapter, we will discuss the architecture and design of the **TechBook** system. In the following sections, we will present code snippets and visual diagrams to help portray a basic understanding of the application design. The architecture is modeled on what is known as the MEAN stack which resembles a three-tier architecture. MEAN is a free open-source software stack for building dynamic websites and supports the MVC (Model View Controller) architecture. The contents of this chapter will begin with a brief overview of the flow of the MEAN stack followed by a more in-depth portrayal separated into the Data Tier, Logic Tier and Presentation Tier.

5.1 Overview

The Presentation tier uses Angularjs, being a client-side JavaScript language it is the first one to process the request made by the client. The request is then sent to the Logic-tier which is a Nodejs server side JavaScipt language. Then the request enters the third phase using ExpressJS to make a request to the Data tier.

From that point, the data is retrieved from the MongoDB and the response is returned to Expressjs. Finally, NodeJS takes the data back from ExpressJS and the data is returned to the Presentation tier to display the result.

- Database **MongoDB**
- Server **Node.js/express**
- Client **Angular.js**

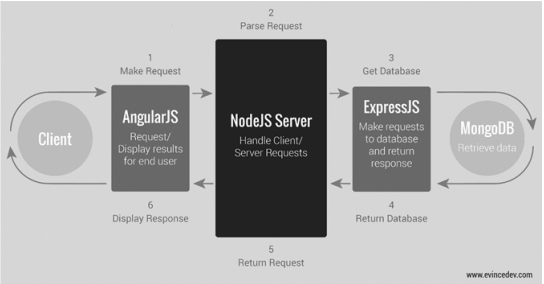

Figure 5.1: MEAN Stack

5.2 Data Tier

For the database adapting the MEAN architecture we used MongoDB.

5.2.1 Mongoose

Harnessing the true power of the Mean stack we used the Mongoose object modeling package for Node. Mongoose enabled us to have access to the full suite of MongoDB commands to perform CRUD (Create, Read, Update, Delete) operations.

To use mongoose we must first connect the logic tier to the data tier. We used the following command in the project directory to add it to our Node project:

```
npm install mongoose --save
```

Once the package was installed we have to access it in our project :

```
1 var mongoose = require('mongoose');
```

Finally to connect to our MongoDB:

```
1 // Connect to the mongodb using settings from the config file
    .
2 mongoose.connect(config.database, { promiseLibrary: require('
    bluebird') })
3   .then(() =>  console.log('\x1b[32m%s\x1b[0m', 'INFO:
    Connection to database succesfull'))
4   .catch((err) => console.error(err));
```

5.2.2 Mongoose Schema

Prior to performing CRUD operations, we had to define our mongooses models. These represent documents which can be saved, read and retrieved from our database. The Mongoose Schema is how we define attributes to these documents. To enhance security and the SRP(Single Responsibility Principle) numerous different models were defined and using keys enabled the ability to map relationships to other models.

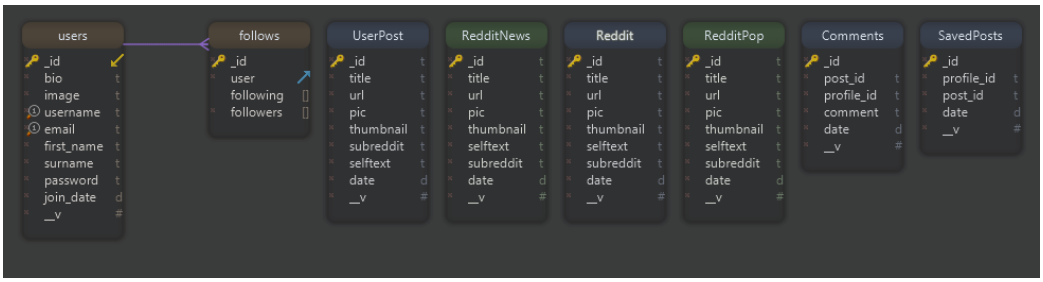

Figure 5.2: MongoDB models

Figure 5.11 below shows how the UserSchema was defined in user.js. The schema was designed to add maximum functionality while also protecting the system from having invalid objects added to the database. There are a few things to note about this particular schema.

- **Uniqueness** : The *username* and *email* fields are set to unique. This ensures a user can only have one account per email and that the username will not cause issues when a search query is performed.
- **Required** : The *username*, *firstname*,*surname* and *password* fields are required. These represent the bare minimum that must be initialized to have a valid account.
- **Default** : The *joindate*,*bio* and *image* are set to default. This allows a faster user registration with a default image used, the join date set to the current date and a generic bio that can all be updated using the settings link.

```
1 // Schema used to 'filter' data to be stored in the '
    UserSchema' collection in mongo
2 var UserSchema = new mongoose.Schema({
3   username: {
4     type: String,
5     unique: true,
```

```
6       required: true
7     },
8     email: {
9       type: String,
10      unique: true,
11    },
12    first_name: {
13      type: String,
14      required: true
15    },
16    surname: {
17      type: String,
18      required: true
19    },
20    join_date: {
21      type: Date,
22      default: Date.now
23    },
24    bio: {
25      type: String,
26      default: 'Tell me about yourself'
27    },
28    image:{
29      type: String,
30      default: 'profile.jpg'
31    },
32    password: {
33      type: String,
34      required: true
35    }
36  });
```

Listing 5.1: Defining User Schema

5.2.3 Password hashing

To ensure the highest standard of security any confidential user information should never directly be stored in a database. Mongoose allows us to define functions using 'pre(function,functionToExecute)' that executes prior to saving a document. Bcrypt a node-module that simplifies hashing can be used to both encrypt and decrypt sensitive data. Combining the use of these technologies when a user account is created or modified, the password is hashed before saving the document and in turn, the hashed value of the password is saved in the database thus protecting the original password being exposed in the event of a malicious attack.

```
1 // Define pre hook for document.
```

```
2  UserSchema.pre('save', function (next) {
3    var user = this;
4    // If password new or edited.
5    if (this.isModified('password') || this.isNew) {
6      // generate a salt and process data for 10 rounds
7      bcrypt.genSalt(10, function (err, salt) {
8        if (err) {
9          return next(err);
10       }
11       // Generate a hash of password.
12       bcrypt.hash(user.password, salt, null, function (err,
     hash) {
13         if (err) {
14           return next(err);
15         }
16         user.password = hash;
17         next();
18       });
19     });
20   } else {
21     return next();
22   }
23 });
```

Listing 5.2: Password Hashing

Ensuring the password is hashed is of major importance but equally as crucial is the ability to be able to compare the hashed value with a password entered by a user attempting to log in. To achieve this we were able to attach a comparePassword function to the UserSchema which executes the compare method from the Bcrypt module and checks for a match between the password entered and the hashed password stored in the database as shown in Figure 5.3.

```
1 // Compare password for login.
2 UserSchema.methods.comparePassword = function (passw, cb) {
3   bcrypt.compare(passw, this.password, function (err, isMatch
    ) {
4     if (err) {
5       return cb(err);
6     }
7     cb(null, isMatch);
8   });
9 };
```

Listing 5.3: Password Comparision

5.3 Logic Tier

The Logic tier is where the Web API service acts as the doorway to all business logic and data storage. Here we perform more complicated computations and sequencing of events allowing us to manipulate data by creating, reading, updating or deleting. Some events are user-driven such as editing a profile and some run periodically such as generating posts from the Reddit API. Due to NodeJS's non-blocking IO calls and light-weight implementation, it enables the application to scale handling thousands of concurrent connections. Being written in the NodeJS JavaScript it is perfect for connecting our Data-tier to our Presentation tier. The Logic tier also provides authentication to ensure no unauthorized access is given to the database.

5.3.1 Authentication

With the purpose of security and the recent rise in data leaks making social media the biggest data breach threat [17] the application was designed to secure data when a user is calling protected API routes.

At a high level the components of the authentication flow are as follows:

- The user data is stored in MongoDB, with the passwords hashed prior to saving.
- CRUD functions are built in an Express API such as login, register, get profile and update.
- The Angular application calls the API and deals with the responses.
- The Express API generates a JSON Web Token upon registration and passes this to the Angular application when a user successfully logs in.
- The Angular application stores the JWT in local storage to maintain the user's session.
- The Angular application checks the validity of the JWT when displaying protected views.
- The Angular application passes the JWT back to Express for validation when calling protected API routes.

5.3.2 JSON Web Tokens

For the purpose of security, we used JSON Web tokens to validate that a legitimate user is logged into the system and prevent unauthorized access. *JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties*[18]. A unique JWT token is created in accordance with the official JWT standard. A users token is signed with the username, password and an expiration date that renders the token invalid after a specified time has passed as seen below. To add to the security, the Token is also signed with a secret key defined in the *database.Js* class so that a user cannot simply try to recreate their own version of a token.

```
1  // Method to generate JWT token.
2  UserSchema.methods.generateJWT = function () {
3    var today = new Date();
4    var exp = new Date(today);
5    exp.setDate(today.getDate() + 60);
6
7    return jwt.sign({
8      id: this._id,
9      username: this.username,
10     exp: parseInt(exp.getTime() / 1000),
11   }, config.secret);
12 };
```

This token can be used for many types of verification. On the client or presentation tier, the JWT is returned from the server after a user has successfully logged in and is stored in local storage. This allows the system to check the validity of the user further allowing the restriction/ restriction of components. For more on how the *Json Web Tokens* are used on the presentation tier see Section 5.4.3.

5.3.3 Server Logging

For development and operation purposes we created a Node module log system on the server. A server logging system boasts many advantages for both the debugging and monitoring the API of the application. By setting up an exportable logger in the *config* folder and importing the logger into the API routes, we can easily debug and monitor information and errors on the server.

The logs required for the system were information logs, error logs and debug logs. The logs are located in *logs* directory on the server and separated into *debug.txt*, *info.txt* and *error.txt*. When a call to log a message is executed

the appropriate file is appended with a time stamp and the message. Figure 5.4 displays a shortened version of the logging implementation.

```
1  // Make logger exportable so any file can reference it.
2  var Logger = (exports.Logger = {});
3
4  // Creates folder for logs if one doesn't exsist.
5  if (!fs.existsSync('./logs')){
6      fs.mkdirSync('./logs');
7  }
8
9  // Create 3 write streams to allow to append info, error and
       debug logs to different streams.
10 var errorStream = fs.createWriteStream("./logs/error.txt");
11
12 // Append error message to log file along with the current
       date.
13 Logger.error = function(msg) {
14   var message = new Date().toISOString() + " : " + msg + "\n"
       ;
15   errorStream.write(message);
16 };
```

Listing 5.4: Server Logging

To use the logger we simply call the error/debug/info function as shown in Figure 5.5 when an error occurs following a failed registration.

```
1  // Import logger to handle server logging.
2  var logger = require("../config/serverlogger").Logger;
3
4  if (!req.body.username || !req.body.password) {
5      logger.error("[Register] : user attempted signup with no
       username or password");
6      res.json({
7        success: false,
8        msg: 'Please enter username and password.'
9      });
10   }
```

Listing 5.5: Server Log call from API

The message is then appended to the log file allowing us to easily view the activity on the server. Below is an example of the *error.txt* after an error has been logged.

```
1  2019-04-10T18:30:49.616Z : [Register] : user attempted
       register with duplicate username
2  2019-04-10T18:30:49.616Z : [Register] : user attempted
       register with duplicate email
```

```
3 2019-04-10T18:30:49.616Z : [Login] : user attempted login
      with incorrect password
4 2019-04-10T18:30:49.616Z : [Register] : user attempted signup
      with no username or password
```

Listing 5.6: error.txt log file

5.3.4 API

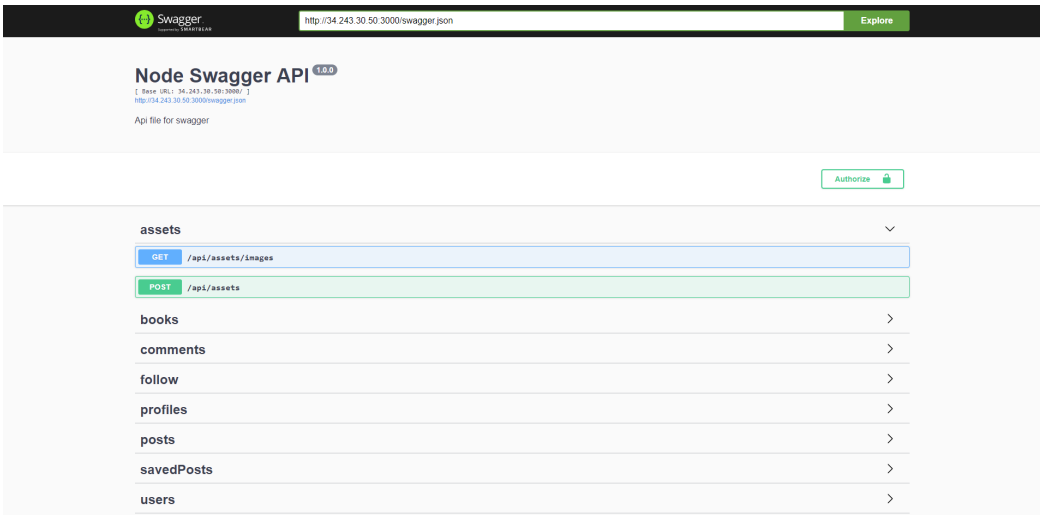

Figure 5.3: Swagger Documentation

We documented our API using Swagger we we talked about in *Section 4.4.2* of the technology review.

Our API is a RESTful web service which allows the user full access to the website by using the Angular client. The RESTful web API can be used to further expanded TechBook by making a mobile application that interacts with the API. To view the swagger documentation of TechBook, see *Section 8 Appendices*.

5.4 Application Tier

The presentation tier is based on the Angular front-end web framework. In contrast to traditional multiple page web applications the Client is presented as a single page application in which components are injected into. In this section we will look at the Angular app structure, the services which allow us to interact with our node js server and the finished view of the pages.

5.4.1 TypeDoc

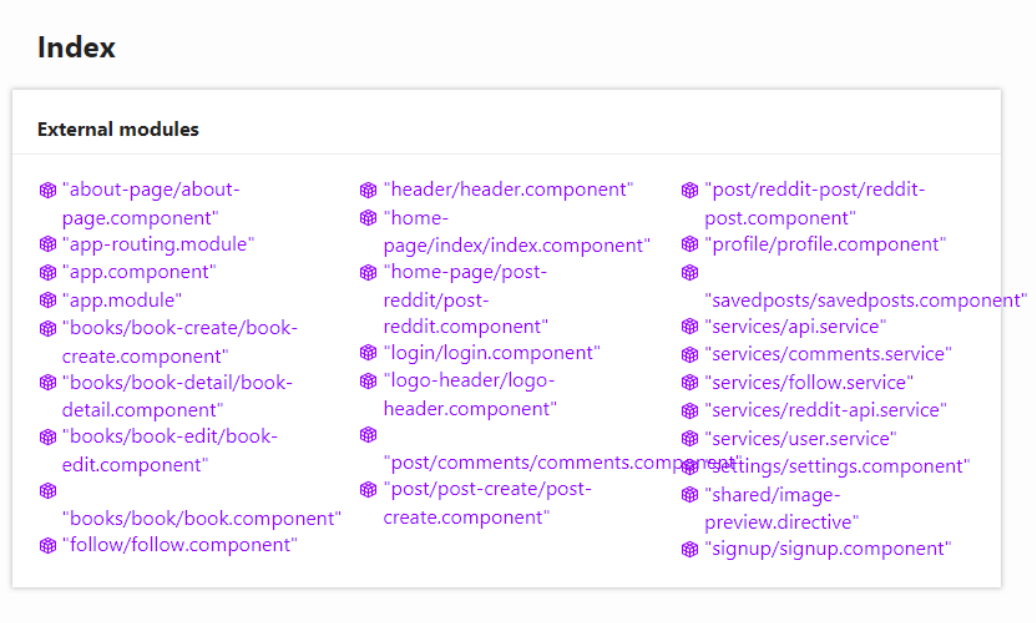

Figure 5.4: TypeDoc Documentation

```
1       /**
2         * Set title.
3         *
4         * @param {string} newTitle- new title.
5         * @memberof LoginComponent
6         */
7        public setTitle(newTitle: string) {
8          this.titleService.setTitle(newTitle);
9        }
```

Listing 5.7: TypeDoc Annotation

5.4.2 Angular Services

The Angular services allow our application to interact and access data from our database via the node server. Thus keeping data retrieval separate to page functions. In the services, we provide the logic to process HTTP requests to the API to perform CRUD operations on the data served by the Node.js/ExpressJS server.

5.4.3 Page views

In this section, we will look at some screenshots of the web application from the view of a PC web browser and on a Samsung Galaxy S6 mobile device. For each page we give a brief overview of the features and out summary of the functionality of the page will be provided along with any input validation that was used. As there is a lot of pages we will only describe the most important ones. A few screenshots will be provided of the finished site to show the design of the UI.

Navigation Pane

Overview

- Fully responsive.
- Positioned under header but sticks to the top of the page when scrolling down.
- When not logged in contains input boxes for username and password and option to register an account.
- Links to *Home, Make Post and About Page.*
- Displays the user avatar and username.
- Drop down menu contains a link to *Profile, Account, Friends, Saved Posts* and an option to log out.

In Depth Review

The header component is injected into the main index.js view. This allows us to render the navigation pane at all times while the main content view is adapted to the selected page view. The header component contains a header logo displaying the image that was designed for the application and a navigation bar with numerous features. When designing the navbar we decided it must meet the following conditions: appealing, easy to use and fully functional, all whilst being fully responsive to changing screen sizes and different devices.

To add the functionality when a user enters the site and is not currently logged in the navbar contains a login form that enables a user to log in. As long as a user is not logged in all other functionality from the navbar is disabled as seen in Figure 5.5 . To add to the pleasing aesthetic the navbar

is also made sticky. Figure 5.6 shows that when the page is scrolled down the header logo will scroll out of view and the navbar will stay located at the top of the page.

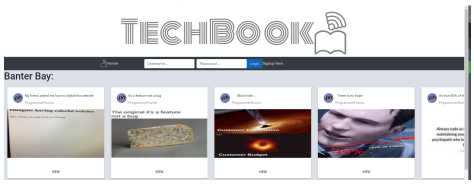

Figure 5.5: Header

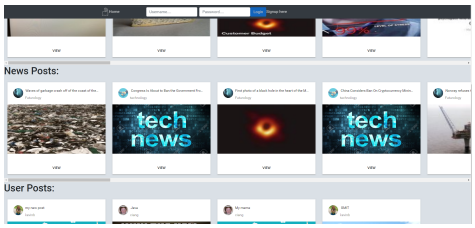

Figure 5.6: Sticky Navbar

However, when the user is logged in, even more functionality is active. Shown in Figure 5.7 relating general site functions three buttons with corresponding icons are added to redirect the user to the Home, Make Post and About Us page. For functions related to the current user, a drop-down menu was added along with the logged in users thumbnail photo and username. Selecting the arrow icon displayed in Figure5.8 displays a drop-down menu. This menu has options to render the Profile, Account Settings, Friends and Saved post pages. The Logout button in the drop-down menu logs the user out, deletes the JWT token from session storage and redirects to the homepage.

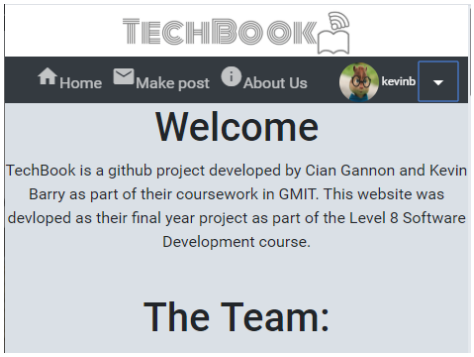

Figure 5.7: Navbar

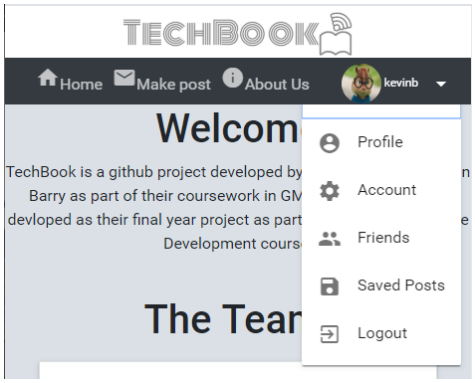

Figure 5.8: Navbar Dropdown active

Log in Page

Overview

- Fully responsive.
- Username and password input box.
- Log in authenticated by server.
- Returns *Log in failed. User not found* message if user not found in database.
- Returns *Incorrect password* message if password is incorrect.
- Hides password entered.
- Link to *Register Page*.
- Saves a JWT token retrieved from server after successful log in.

In Depth Review

The login page consists of a username and password input box. In order to submit a login attempt, both values must be supplied. The values are then authenticated on the server and if successful the server returns a JWT token that's stored in session storage and the user is redirected to the homepage view. If the username is invalid the server will return a "Login failed. User not found." error message to the view or in the case of an invalid password an "Incorrect password" message is returned.

Figure 5.9: Login Web View

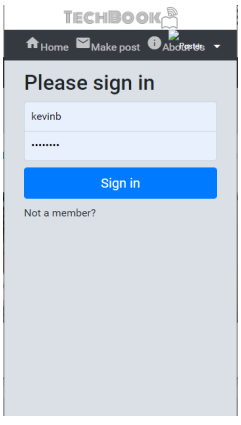

Figure 5.10: Login Mobile view

Register Page

Overview

- Fully responsive and validated.
- Checks database if username or email already exists returning error message *Username/email already in use please choose another* .
- Password and Confirm Password to validates passwords match, returns *passwords don't match* if not.
- Password must be 8 characters long, returns *passwords don't match*.
- Hides password entered.
- All fields are required, returns error message *X is required* .
- Email verified must have '@' and '.', returns *Email required* if not.
- Successful register save users details to database with password encrypted.

In Depth Review

The register page allows a user to enter credentials for a new user profile. To submit a register attempt all fields are required. This page view contains numerous different aspects that must be validated:

- Username : Required, Must be unique to the server.
- firstname : Required.
- surname : Required.
- email : Required, Must contain '@' followed by '.' symbol. Must be unique to the server.
- Password : Must be 8 characters and must match confirm password value.

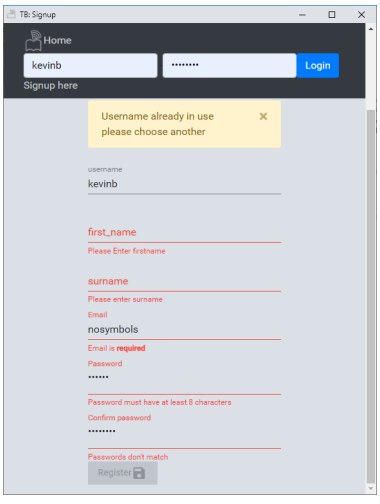

Figure 5.11: Register Validation

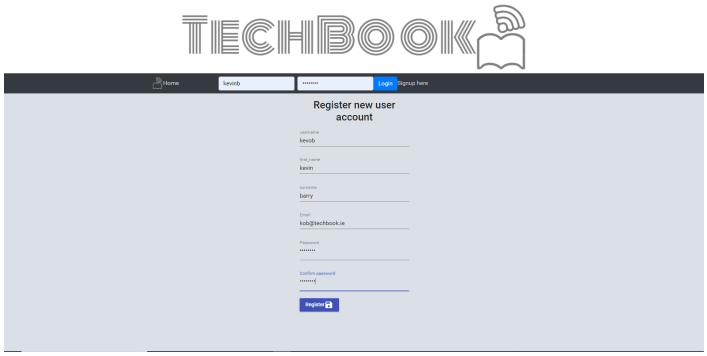

Figure 5.12: Register Web View

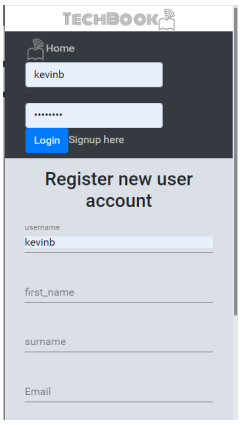

Figure 5.13: Register Mobile view

Profile Page

Overview

- Fully responsive grid layout, sectioned into Profile section, recent posts and recent comments.
- Profile section displays user avatar, profile image, and profile information.
- Server checks if viewing someone another user profile and displays a follow/unfollow toggle button if it is.
- Selecting follow/unfollow adds/removes an entry to the following table in the database using foreign keys to set a relationship between objects.
- Link to subscribed posts and friends page.
- Recent posts section displays a fully functional clickable list of angular material cards posts.
- Recent comments section displays a fully functional clickable list of angular material cards with latest comments.

In Depth Review

The profile page is used to view a users details (profile image, avatar, name, username email and when they registered) and activity (created posts and comments on other posts). By using **Angular Material Components**

the page has been split into a grid to display different sections as seen in the Figure 5.8 snippet. The grids are assigned to profile info, recent posts, and recent comments. Harnessing the functionality of **Angular Material Components** enables the page to be fully responsive. As you can see in Figure 5.14 the layout is set to have profile info to the left, while recent comments and posts are stacked on top each other to the right. However as the size of the screen scales, so does the view the grid components that will shrink as the screen size decreases and in the case of a mobile device as seen in Figure 5.15 the components are stacked vertically on top of one another.

```
<div class="mdc-layout-grid">
  <div class="mdc-layout-grid__inner">

    <div class="mdc-layout-grid__cell mdc-layout-grid__cell--span-4">
      <mat-card class="profile-card">
      .......
      </mat-card>
    </div>

    <div class="mdc-layout-grid__cell mdc-layout-grid__cell--span-8">
      <div class="mdc-layout-grid__inner ">
        <div class="mdc-layout-grid__cell mdc-layout-grid__cell--span-12">


          <!-- If the user has made any posts -->
          <div *ngIf="postsUser.length > 0">
            <h1>Recent Posts:</h1>
          ........
          </div>

          <!-- If the user hasn't made any post -->
          <div *ngIf="!postsUser.length > 0">
            <h1>No posts to show</h1>
            <p>Tell {{profile.username}} to get posting!</p>
          </div>
        </div>

        <div class="mdc-layout-grid__cell mdc-layout-grid__cell--span-12">

          <!-- If the user has made any comments -->
          <div *ngIf="commentsUser.length > 0">
            <h1>Recent Comments:</h1>
            <!-- Comment card -->
            <mat-card class="comment-card" *ngFor="let
```

```
    commentUser of commentsUser">
35                .....
36              </mat-card>
37            </div>
38
39            <!-- If the user hasn't made any comments -->
40              <h1>No comments to show</h1>
41
42            </div>
43          </div>
44        </div>
45      </div>
46    </div><div *ngIf="!commentsUser.length > 0">
47
48  </div>
```

Listing 5.8: Angular Material Component Grid

The profile page is used for both viewing a users own profile and viewing another users profile and is changed dynamically. When viewing another users account a follow/unfollow option appears. Once a user follows another user the button changes to unfollow and vice-versa. Viewing your own profile disables these options and allows the user to view pages they are subscribed to. This is achieved by a method in the *userService* shown in Figure 5.9 that checks if there is a valid JWT or *Json Web Token* for that user and adjust the options accordingly Figure 5.10.

```
1   // Check if a user is logged in.
2   isLoggedIn(): boolean {
3     var currentToken = this.getJwtToken();
4     if (currentToken) {
5       return true;
6     } else {
7       return false;
8     }
9   }
10
```

Listing 5.9: Validate JWT

```
1         <mat-card-actions>
2           <!-- If user is logged in -->
3           <div *ngIf="userAPI.isLoggedIn()">
4             <button mat-button [routerLink]="['/saved/',
    profile.username]">SUBSCRIBED POSTS</button>
5             <button mat-button [routerLink]="['/follow/',
    profile.username]">FRIENDS</button>
6             <button *ngIf="!isFollowing && !isUser" mat-
    button (click)="follow(profile._id)">FOLLOW</button>
```

```
7             <button *ngIf="isFollowing && !isUser" mat-button
    (click)="unFollow(profile._id)">UNFOLLOW</button>
8           </div>
9           <!-- If user is not logged in -->
10          <div class="user-buttons" *ngIf="!userAPI.
    isLoggedIn()">
11            <button mat-button [routerLink]="['/saved/',
    profile.username]">SUBSCRIBED POSTS</button>
12            <button mat-button [routerLink]="['/follow/',
    profile.username]">FOLLOW STUFF</button>
13          </div>
14        </mat-card-actions>
```

Listing 5.10: Follow / Unfollow option

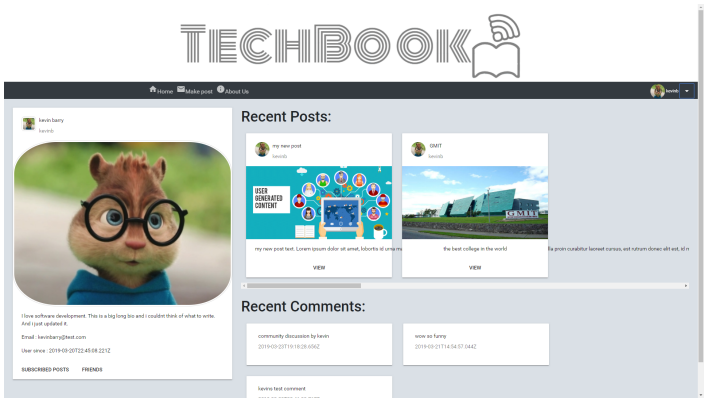

Figure 5.14: Profile Web View

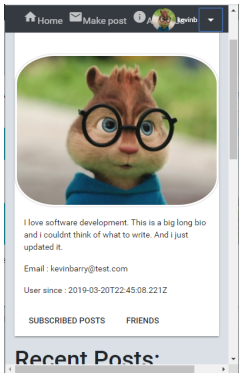

Figure 5.15: Profile Mobile view

Settings Page

Overview

- Fully responsive.
- Allows user to update email address, first name, surname, and bio in the database.
- Only values changed will be sent to the database to update.
- Server validates to ensure the email is unique and not already in database returning error message *Email already exists please choose another* if validation fails.

- Allows user to upload a profile image to the server.
- Image upload can be previewed and validation checks image is of image type.

In Depth Review

The settings page view allows a user to edit their current credentials and is accessed by clicking the settings tab on the navbar drop-down menu. The form fields are pre-filled from the user's data retrieved by the server.

```
/**
 * Set the form data in SettingsForm to the details of the
   current user.
 *
 * @param id - The current users id.
 */
setForm(id) {
  this.userService.getProfile(id)
    .subscribe(profile => {
      // Must extract profile data from response.
      this.profileinfo = profile[0];
      // Form object
      this.settingsForm.setValue({
        email: this.profileinfo.email,
        first_name: this.profileinfo.first_name,
        surname: this.profileinfo.surname,
        bio: this.profileinfo.bio
      });
    });
}
```

Listing 5.11: Auto fill form data

This page also allows the user to update or change their current profile image. Selecting the 'choose file' button pops up a file explorer window and allows the user to select an image. The chosen image is then shown in a preview tab so the user can have a peek prior to selecting the "Upload new profile image" button. This was achieved by importing the **ng2-file-upload** module and applying custom functions. The uploaded image is then saved in an image folder on the server and the user's account is updated with the location to access this image. The uploader is also validated to only accept image files.

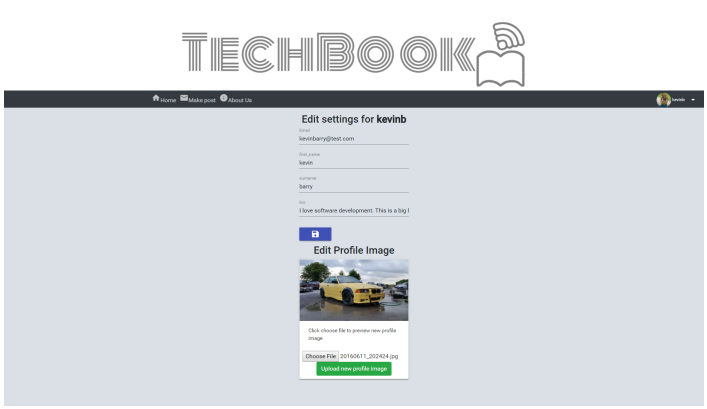

Figure 5.16: Settings Web View

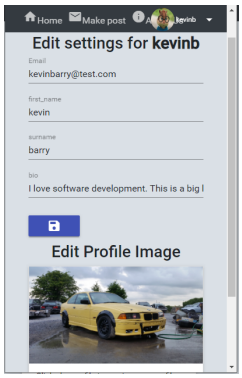

Figure 5.17: Settings Mobile view

Friends Page

Overview

- Fully responsive.
- Retrieves who a user is following and who is following them from the database.
- Toggle button to display either followers or following.
- Following/ followers are displayed as card lists with profile avatar, the username that acts as a link to their profile and the user's bio.

In Depth Review

The friend's page is a view that allows a user to toggle a view between followers and following. This page can be accessed in two ways which render different views. If a user wants to view a list of their own followers and who they are following they can select the "Friends" button in the navbar dropdown menu. If a user would like to view other users followers and following list a button "Follows" can be selected on the preferred user's profile page. The lists contain cards that show an image avatar, username, and bio for each user. To view a users profile from this view simply click on the username and the system redirects you to that user's profile page.

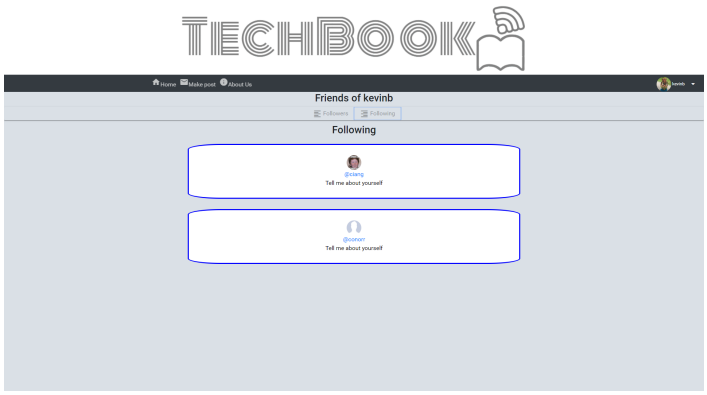

Figure 5.18: Friends Web View

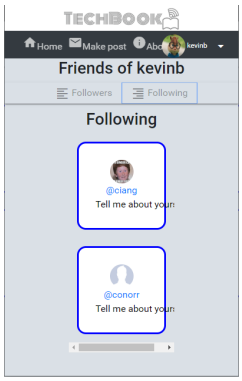

Figure 5.19: Friends Mobile view

Home Page

Overview

- Fully responsive.
- Divided into three sections displaying lists of cards for news posts, banter posts, and user posts.
- All content is retrieved from the servers API.
- The news posts and banter posts are generated from the Reddit API.
- Each card contains an avatar, title, the source, and a picture.
- A user can comment on a post by clicking into it.

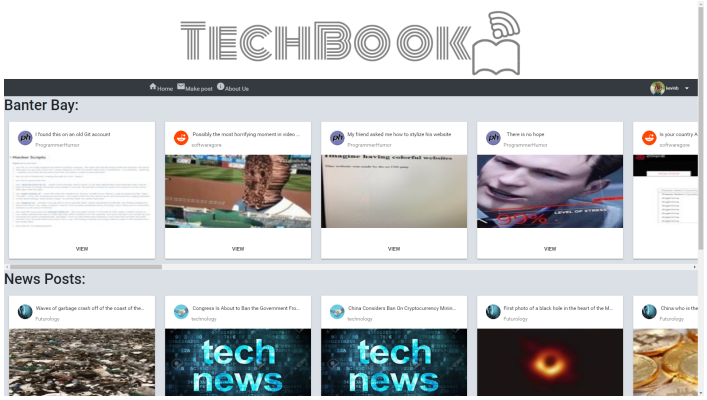

Figure 5.20: Home Page Web View

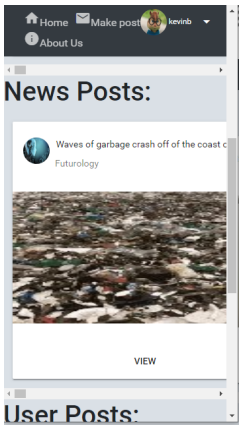

Figure 5.21: Home Page Web View

In Depth Review

About Page

Overview

- Fully responsive.
- Displays cards for the team members, documentation and the project components.
- Each card has an avatar, heading, brief summary and a link to the resource.

In Depth Review

The purpose of the About Page view is to give the user an insight into the project. The view is separated into three sections with each item being displayed on its own card. Each card consists of an image, a link to the resource, a summary of the resource link and also some brief information on that item. The Team section is an overview of the developers of the web application with a clickable link to view our Github profiles. The Documentation section contains links to the project source code, dissertation and the Swagger documentation for the API. The third section is used to provide links to resources and give information on some of the components used in the project such as MEAN Stack, MongoDB, Express.js, Angular, Node.JS, Reddit API, Passport.JS, HTML and CSS.

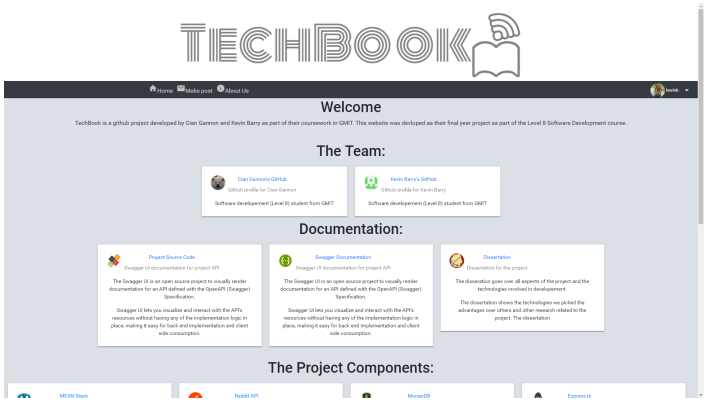

Figure 5.22: About Web View

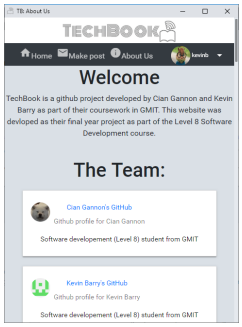

Figure 5.23: About Mobile view

Chapter 6

System Evaluation

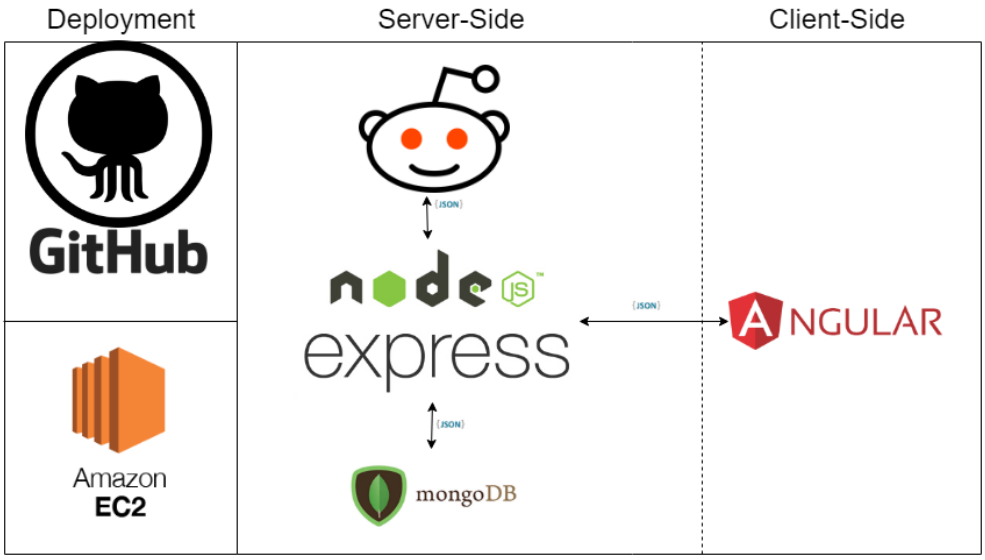

Figure 6.1: System Architecture

As discussed in section *1.1: Project Objectives*, we will now expand on the system implemented in this project.

Dissertation

- Introduce the concept of the project.
- Provide an understanding of social media.
- Provide an understanding of web technologies.
- Describe the development of the applied project

Applied Project

- Produce a simple easy to use web application.
- Deliver a social platform for tech-savvy people that differs from the norm.
- Dive into new web technologies.
- Complete the project collaborating as a team using an efficient and effective approach.

6.1 Testing

It was an important part of the project for us to plan out the project, and test each component of the full stack web development. We started prototyping the MEAN stack as well as developing the project early to test the functionality early.

6.1.1 Prototyping

Early on we built multiple applications to test the functionality of the MEAN stack. We built a Node.js application, then added ExpressJS. Getting a basic grasp of each technology before advancing was very import as to understanding how the system as a whole will work from top to bottom.

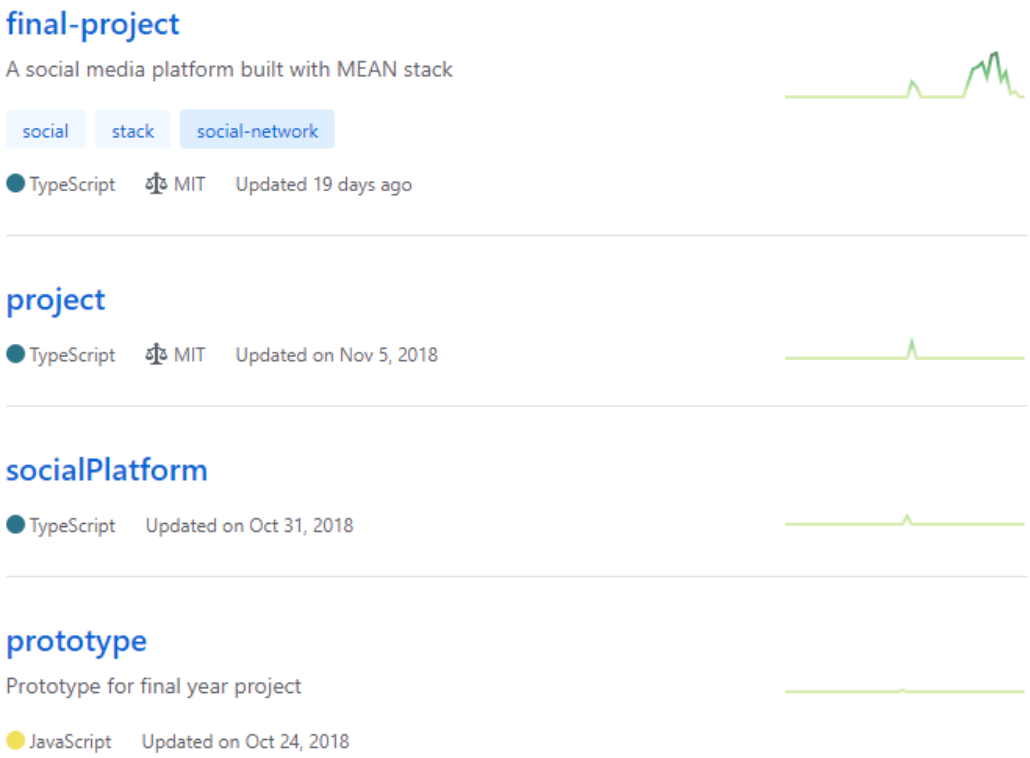

Figure 6.2: GitHub Prototypes

As you can see above there are four GitHub projects. The first one at the top is the final project. The rest are prototypes we developed to test the MEAN stack functionality and understand each component of the MEAN stack (MongoDB, ExpressJS, Node.js, and Angular).

6.1.2 Deployment Testing

We tested the project using a three-stage deployment. Initial, medial and final. Each stage covered certain criteria for the deployment process. Initially used for unit testing, medial for early deployment testing and final for the public release of the application.

Initial

Initial testing involved testing the application on the local-host and test each function to see how the application would react to our inputs. This was the primary method of testing individual components and ensuring the application would work on the local-host port 3000.

Medial

After the initial testing stage, we had a medial stage for deployment. We built a home server to test the application over a network. This involved buying and building the home server. Once setup and built we connected the PC to the home network and gave the PC a static IP (Internet Protocol). This was extremely important, without giving the server a static IP the local address such as 192.168.0.10 could be moved to 192.168.0.50. If the local IP of the server changed, the port forwarding address would need to change each time the servers did, which would be very tedious.

Once the IP of the server was setup we port forwarded the port 3000 to the server PC's static IP address. The port 3000 had to be port forwarded because of the application using that port. With this setup, we could access the application from another PC in another location. This was used to demo the application to supervisors in meetings.

Final

With a grasp on how deployment works from the stage above we had the knowledge on how and what needed to be done. We deployed our project on AWS (Amazon Web Services). We went with AWS because of their support for the MEAN stack made it a prime contender. Once deployed onto AWS we knew we could access it from inside the instance, but not outside. So knew from above we need to open the port and direct it to our AWS instance. Although in this case, AWS handled the local IP and all we had to do in deployment in this stage was open the port using AWS console, which acted as a router for our AWS instance.

6.1.3 Unit Testing

Unit testing was a testing method we used at every stage in development. Each function block of code was tested for the desired output such as a GET method to the API and then testing to see if that function was retrieving the JSON data before continuing onto the next stage.

6.1.4 System Testing

We used system testing on each stage of deployment to ensure all components worked properly in a deployed environment. This testing method involved using the application and all its components and features. This stage was done far less frequently than unit testing but was just as important to ensure our application was of high quality.

6.2 Performance

The application preforms remarkably well. The biggest 'bottleneck' that stifles performance will be the users connection to the program or the service it is hosted on.

The API performs very well, and because of MongoDB using JSON to store data there is no conversion needed, so the computing needed to access the API is very low.

We also used the Reddit API to make news posts and funny posts for users. This API at first was requested by the user when they accessed the home page but quickly realized that when we wanted to add comments and to save posts we would need to store the data in some way. We reformatted how users get the data and realized having the user access an API route that gets the data from the MongoDB was a little bit faster than when we originally set up the application.

The Angular client is extremely responsive and due to its single page, nature felt far more modern than a multiple page website. Angular being a single page web framework allowed our application to have a more modern look and feel to it. As the user changes the view to different components they never see a white screen when changing pages. The navigation bar, for example, is always present when changing views.

6.3 Evaluation of Objectives

In this section we will expand on the objectives we set out for the project. The project was built with these objectives in mind:

Produce a simple easy to use web application: We did produce a simple and easy to use web application. Using Node.js to host the application by delivering the Angular application to the user and host the API. Using Angular were were able to compartmentalize each component and work on each individually. Using Angular we were able to provide an easy to navigate and responsive single page web application.

Deliver a social platform for tech-savvy people that differs from the norm: We successfully made a social media platform developed around users who are tech-savvy. The site uses the Reddit API to make sure there is always news posts and funny light-hearted posts based around technology. We also allowed the user to make their own posts to share images and news articles.

We made a comment section to allows a community to built up. Allowing comments was extremely important to allow small developers to put up a post to ask for help and have other users help out. Or a user post a link to their GitHub to ask for their opinion. A comment section had to be nailed down in order to build a community.

To expand on community relations we made user profiles. User profiles allow other users to follow them, see their comments, posts, followed users and subscribed posts. User profiles allow other users to built up a rapport with one another.

We tried to differ from other technology-based social media by adding in funny light-hearted posts from the Reddit API. We felt the only social media available for the tech-savvy were from a much more professional standpoint.

Dive into new web technologies: Diving into a new web technology for us was very important. We had experience with Spring, GO and Ruby on Rails. But we wanted to learn a completely new architecture we had no experience with. After deciding on the MEAN stack, we quickly realized how different web development is with a framework like Angular. Angular as we discussed earlier is a single page web application framework. This was different from the traditional sending an HTML document and having the user interact with that.

We also had no experience with Node.js. Node.js uses JavaScript which we had quite a lot of experience with, but never outside the browser. Node.js is very powerful and allows developers to do quite a lot very quickly. Node.js includes the Node Package Manager (NPM), which allows developers to access the largest package library. NPM allowed us to considerate on creating function features by using packages such as Passport.js.

Complete the project collaborating as a team using an efficient and effective approach: As a group project and at a scale we've never done before, we wanted to ensure quality and consistency would be kept. We met up weekly to plan out the next stage of development. We also met up with our supervisors weekly to get feedback on our plan and progress made. Using this meeting was extremely important to ensure we were both on the same page.

We also made sure to keep in persistent contact with one another using application such as WhatsApp and Facebook, which allowed us to maintain in contact outside the college. These applications allowed us to share screenshots of errors we both had, and share our thoughts on a feature.

Using GitHub allowed us to both collaborate on a single project and

see the changes we were making to the project. Using GitHub branches we could test a feature before committing it to the master branch. GitHub was a great way for us to collaborate on the project together and get a feel for how collaborative projects work.

6.4 Limitations and Improvements

As the project developed and we got more and more experience with the MEAN architecture, we came up with better methods of developing features.

6.4.1 Server

As became more experienced with Node.js and its available packages, we realized better methods of using developing the API and retrieving data from the Reddit API. One example of an improvement we would do if we developed another API service is to use swagger from the beginning. As we developed an API service we created routes that would return a specific amount of posts for example. And a route that would return all posts. What we should have done is create a route for posts and send a number of posts I want. This could also be said for comments, such as sending how many I want and from what commenter I want them, rather than create a set route for sending a username that would return a set amount.

6.4.2 Client

As we got better at developing with Angular we improved elements of how data is displayed from the server. Early on we used tables to display the posts and later moved to Angular Material. One thing we would have liked to introduce to the social media site is user tags. We would have liked to add a tag to themselves to show the position they hold in the industry so other users can gauge their ability. Such as 'Cian¡Software Student 4 years¿', so other users can see how experienced they are and take their feedback into account.

6.5 Overall Evaluation

Overall the system works as indented and functions as an effective social media tool. It met the requirements set out in section 1.1 Project Objectives and went beyond what we originally set. Based on the tests we carried

out throughout the project with prototypes and unit tests we were able to build an effective application and find and fix any bugs throughout development. Prototyping was an important phase which helped up understand the MEAN stack on an individual component level. The MEAN stack architecture proved very reliable and easy to develop for. Although we could have improved and added more components to the application we were overall very happy with the progress we made.

Chapter 7

Conclusion

7.1 Overview

Concluding this dissertation, we will consider our aims and achievements of both the theoretical and applied components of our project while also complying and fulfilling all user requirements.

Our main goal of the project was to create a unique social media platform that provides an online community for tech-savvy users. We have succeeded in doing this, by developing an easy to use, responsive, web application that utilizes modern full-stack web development tools to create a 3 tier architecture. As presented throughout this paper our final product is a fully functional platform that allows users to register, log in, create posts, follow users and so much more.

In *Section 1.1* we proposed the following objectives:

- Introduce the concept of the project.
- Provide an understanding of social media.
- Provide an understanding of web technologies.
- Describe the development of the applied project.
- Produce a simple, easy to use web application.
- Deliver a social platform for tech-savvy people that differs from the norm.
- Dive into new web technologies.
- Complete the project collaborating as a team using an efficient and effective approach.

The evidence from our findings discussed in *Section 6 System evaluation* clearly demonstrates that the dissertation has achieved all its original objectives. We can conclude that we have provided the reader with an extensive yet clear insight into the world of social media. Our thorough investigation into all aspects of social media completed in *Context 2* demonstrate the major importance of social media in modern society. The strength of this importance contributes to the overall rationale and decision choice of developing a social platform.

Concluding the theoretical aspects of the dissertation, we dug deep into the research of web technologies and documenting the full development of the applied project. We have shown how the combination of modern technologies, advanced methodologies and best practices can be combined to the produce a fully functional modern web application.

Looking back at the applied objectives we are more than happy, achieving far more than we could have expected. **TechBook** offers a unique online community that can be navigated by all user regardless of their technological abilities. Harnessing the power of the MEAN stacks 3-tier architecture, we have produced a scalable, sophisticated API that handles the applications data load, while portraying a simplistic front end to the user hiding all of the complexities.

From a collaborative perspective, the project was completed with an efficient and effective approach. The result of our efforts, as a team, prove the massive significance of the importance of not only being technically skilled but being capable to work alongside others in order to achieve the optimum result. We functioned very well as a team, keeping communications professional while maintaining a friendly relationship. Solving all issues and differences of opinions as professionals with the projects end goals in mind.

In terms of the requirements stated in Section *1.4 User Requirements* the end product complies with all the initial user requirements specified. Allowing a vast variety of functionality to the user. The project hits all check boxes for what a social media platform requires to become a community by allowing users to register, make friends, follow other users and posts, share and give feedback on other users content.

7.2 Learning Outcomes

Throughout the entire scope of the project from start to finish, we have gathered a substantial amount of knowledge on a vast variety of subjects and technologies. From initial research and design to implementing a MEAN stack 3-tier application with a scale-able API. Pushing our abilities and im-

mersing ourselves into new technologies to achieve the learning outcomes of the module. The skills obtained and experiences earned will carry into our careers, being applied to industry practice. Our research gave us the awareness and ability to explore the present state of the art computing areas and evaluate the literature base. We have seen first hand the importance of critically evaluating the work and research required to complete a project.

From a team perspective the skills learned are priceless. It is clear that everyone works differently and no two developers are the same. Utilizing the strengths and weaknesses of different members of a team from a management and personal level is something so important. This is something that is not easily thought but a crucial skill we acquired through experience. It became evident that as the project grew in scope, the better the communication the better the result. The module has educated us on the importance of communication is vital in software development to ensure work can be completed quicker and deadlines can be met.

7.3 Final Thoughts

As we come to the end of the project and our final weeks in *GMIT*, we hand up this project with a smile on our faces, satisfied that we have achieved our goals, not only with this project but also in our academic career over the past four years. This project was challenging from the start but reaching the end goal has been such a rewarding experience.

7.4 A Word From The Developers

> *"Where to start..? This whole project has been an amazing experience from every perspective. Working alongside Cian has been an absolute pleasure. To think that 5 years ago we met in GTI as strangers doing a level 5 I.T. course, we could have never imagined working together. From our early days of learning about computers to recently both signing contracts with massive organizations in recent weeks. Shout out to Dr.John Healy and Martin Kenirons for all their inspiration and motivation with the project. To wrap this up, to whoever is reading this, I hope you enjoyed our project as much as we did."*
>
> — Kevin Barry

"A great way to end my years as a student. I began studying programming 5 years ago in GTI, in the same year as Kevin. I still remember learning the basics (Strings, arrays, for loops) and thinking 'How would I ever be able to build an application?'. This project is a milestone I've reached in my development as a software developer, that I didn't think I'd reach when I started. This project is a great note to end on in my years of studying software development. With help from our supervisors, Dr.John Healy and Martin Kenirons who were extremely supportive during the development process."

— Cian Gannon

Chapter 8

Appendices

Github Organisation:
https://github.com/Final-Year-Project-Cian-Kevin

Github Project Source Code:
https://github.com/Final-Year-Project-Cian-Kevin/final-project

Deployed Application Link:
http://34.243.30.50:3000

Swagger:
http://34.243.30.50:3000/api-docs/

TypeDoc:
http://34.243.30.50:3000/typedoc/

Video Presentation:
https://www.youtube.com/watch?v=mqdhf0jrF9I

Bibliography

[1] J. Van Dijck, *The culture of connectivity: A critical history of social media*. Oxford University Press, 2013.

[2] B. Winston, *Media, technology and society: A history: From the telegraph to the Internet*. Routledge, 2002.

[3] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff, "A brief history of the internet," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 22–31, 2009.

[4] A. S. Taylor and J. Vincent, "An sms history," in *Mobile world*, pp. 75–91, Springer, 2005.

[5] S. Asur and B. A. Huberman, "Predicting the future with social media," in *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pp. 492–499, IEEE Computer Society, 2010.

[6] A. Nadkarni and S. G. Hofmann, "Why do people use facebook?," *Personality and individual differences*, vol. 52, no. 3, pp. 243–249, 2012.

[7] "10 key principles of agile software development," 2008. `https://project-management.com/10-key-principles-of-agile-software-development/`.

[8] A. Alliance, "What is kanban." `https://www.agilealliance.org/glossary/kanban/`.

[9] I. S. Book, "Data protection act." `http://www.irishstatutebook.ie/eli/2018/act/7/enacted/en/html`.

[10] S. Aggarwal and J. Verma, "Comparative analysis of mean stack and mern stack.," *International Journal of Recent Research Aspects*, vol. 5, no. 1, pp. 133 – 137, 2018.

[11] D. McCreary and A. Kelly, *Making sense of NoSQL : a guide for managers and the rest of us.* Manning, 2013.

[12] S. Pasquali, *Mastering Node.js : Expert Techniques for Building Fast Servers and Scalable, Real-time Network Applications with Minimal Effort.* Packt Publishing, 2013.

[13] N. Kaufman and T. Templier, *Angular 2 Components.* Packt Publishing, 2016.

[14] M. Lang, M. Wiesche, and H. Krcmar, "Criteria for selecting cloud service providers: A delphi study of quality-of-service attributes.," *Information Management*, vol. 55, no. 6, pp. 746 – 758, 2018.

[15] S. S. Sriparasa, *JavaScript and JSON Essentials : Successfully Build Advanced JSON-fueled Web Applications with This Practical, Hands-on Guide.* Community Experience Distilled, Packt Publishing, 2013.

[16] V. Bojinov, *RESTful Web API Design with Node.js - Second Edition.*, vol. Second edition of *Community Experience Distilled.* Packt Publishing, 2016.

[17] P. Gilbert, "Social media becomes biggest data breach threat." `https://www.itweb.co.za/content/G98YdqLxZZNqX2PD`.

[18] O. J. standard, "Jason web tokendata protection act." `http://www.jwt.io`.