Billy Catchpole

14188056

Recall

Project Proposal

Level 6

2019-20

Table of Contents

**Introduction**

1. Problem statement3
2. Requirements3-4
3. Objectives4-5

**Research methodology**

1. Secondary research5-6
2. Primary research6

**Resources and skills**

1. Existing6
2. Required 6

**Project plan**

1. About the literature review7
2. Creating the task list and Gant chart7
3. Arguing time estimates7-8

**Risks**

1. Risk management8-9

**LSEP**

1. UH Ethics approval9
2. Multiple category issues9
3. Legal issues10
4. Social issues10-11
5. Ethical issues11
6. Professional issues11-12

**Summary**

1. Proposal summary12

**References**

1. References13

**Appendix**

1. Data to be collected on human subjects……………………………………………………..………………..14
2. Full description of resources and skills including project methodology14-17
3. A list of each phase17
4. Work ethic18
5. Task list18-20
6. Gantt chart21

**Introduction**

**Problem statement**

Learning a new programming language or concept can be a daunting task for both experienced professionals and novices alike. With new syntax, design patterns and algorithms to master, a plethora of new information must be understood and remembered.

A common solution involves taking notes on said concepts via a range of methods including word documents, notepad software and code comments.

While useful, this solution presents another problem, notes become unstructured, the syntax hard to distinguish and spread over a range of locations in computer folders. Finding the right document, then searching for the right entry can become cumbersome.

**Aim of this project**

Therefore, this project will provide a modern web-based solution where users can manage their notes and programming concepts in an orderly, structured fashion.

It will include :

* Predefined structure to the notes
* Centralised access to the notes (search function)
* Easily distinguishable syntax from the note body
* Ability to share notes with peers

**Core functional requirements**

* 1. Users can create a unique account
  2. Users can reset their password via email
  3. Users can create, modify, delete and view their notes
  4. Users can only CRUD their data
  5. Users can search for a specific note by its title
  6. Users can view their notes by subject area
  7. Users can copy their syntax with 1 click

**Advanced functional requirements**

* 1. Users can group multiple notes into a process
  2. Users can invite their friends to give them access to their notes
  3. A Tutorial showing the user how to use the various features of the app which may be replayed
  4. Users can search for a specific note via its title, search tags, subject or syntax, one search string will be run against all criteria using an algorithmic flow

**Core non-functional requirements**

* 1. Usable with minimal training
  2. Readable, maintainable, modular architecture across the application

**Advanced non-functional requirements**

* 1. JsonWebTokens to protect against CSRF attacks
  2. Encryption of all sensitive data including email and passwords
  3. Dos protection
  4. Data sanitization to protect against noSQL injection (MongoDb) and Cross-Site scripting attacks
  5. Brute force attack protection
  6. Passwords must be at least 8 characters and contain an uppercase letter
  7. Asynchronous processing of all requests to the API
  8. Responsive on mobile, tablet and desktop

**Objectives**

1. Using NodeJS, MongoDb, and express, design and implement all core functionality of the REST API and database, to provide the foundations of the application, by 15th June including the report writeup for this phase.
2. Through secondary research (via a literature review), further investigate, draw conclusions on what is required, then implement protection for web security threats to the API, to be equipped to deal with any potential attackers, by the 29th June including the report writeup for this phase.
3. Using testing frameworks such as mocha and chai, write automated tests for every endpoint of the API, to ensure it performs as expected and to be able to sign the backend off as complete, by the 4th July including the report writeup for this phase.
4. Focusing on mobile-first, design, connect to the backend, and implement the user interface (including all front-end business logic), to bring the front and back end together to create a working application, by the 1st august including the report writeup for this phase.
5. Using media queries, complete the interface on tablet and desktop devices, to capture all potential users regardless of their device and sign of the design as complete, by the 10th august including the report writeup for this phase.
6. In addition to using primary research by watching friends and family use the application, manually perform e2e tests for the application, to try and uncover any flaws/defects in the application and enhance usability and ensure it works correctly, by the 18th august including the report writeup for this phase.
7. Using my experiences learned along the way, write the project conclusion, to grow and take away lessons from this project, by the 19th august.
8. Bringing together the sections which were written during each stage of the project, finalise the project report checking it for academic standards, that it is engaging and reflects the development experience accurately, to create a finished project report which complements the artefact, by the 28st august
9. Submit the report, and artefact on or before 28th august.

**Research methodology**

**Secondary research**

Since this application will be marketed to software engineers and students at large, the main point of research will be around web application security and preventing attackers exploiting vulnerabilities which they could use to damage legitimate users.

**Article 1 –** *Prevention Of Cross-Site Scripting Attacks (XSS) On Web Applications In The Client Side* (SHALINI and USHA, 2011)[1]

This article provides an overview of the different types, methods used, detection and prevention methods of cross site scripting attacks.

**Article 2 –** *Mongodb Nosql Injection Analysis And Detection* (Hou et al., 2016)[2]

Exploration of how attackers inject database queries into a website to covertly gain access to a database, and how to prevent them from doing so.

**Article 3 –** *BRUTE FORCE ATTACK – BLOCKING TECHNIQUE* (G. Sowmya, 2013)[3]

A look at how attackers brute force crack passwords and the various ways of preventing them such as locking, CAPTCHA, unique IP addresses and more.

**Article 4 –** How to protect your server against dictionary attacks (James, 2012)[4]

Similar type of attack to brute force, however a different password cracking method is used.

**Article 5 –** *Authentication And Authorization Of End User In Microservice Architecture* (He and Yang, 2017)[5]

Methods of authenticating microservice API’s including Json web tokens

**Primary research**

The primary research will involve observing human participants use the application to identify any usability issues. The most common and serious issues will be fixed, thus providing the ability to continuously improve the usability of the application. **UH ethics approval will be required.** *For an explanation of data to collected, see appendix 1a.*

**Resources and skills**

**Existing**

The following resources and skills are currently owned or familiar.

1. React.js – for the UX
2. Node.js and express – for the API
3. MongoDb and Mongoose – for the Database
4. Visual Studio Code
5. Firebase and Heroku
6. FlatIcon and other sources of free images – **Attribution required**
7. BrowserStack – **Premium service**
8. Chrome developer tools
9. Desktop computer including internet access
10. GitHub
11. Postman
12. Bcrypt
13. Helmet 14. Report writing and referencing skills

**Required**

The following resources and skills are currently not owned or unfamiliar.

1. Web application security measures
2. Physical devices for testing
3. UH ethics approval
4. Integration and end to end testing
5. Interactive tutorial building 19.Project management skills

All software used will be free or open source, including the hosting providers, therefore this project will not have any associated costs. The trade-off is however, free hosting providers provide a limited service and there may be some delay in response times and no built in Dos protection. GitHub student provides free access to many paid services such as Browserstack.

*For a full description of every resource and skill,* ***in addition to the project methodology****, please visit the appendices and refer to Appendix 1b – Full description of resources and skills.*

**Project Plan**

**Please note: the literature review has not been included in the timescale, as it will be completed before the start date of the project, during the project planning module.**

Upon reviewing the objectives, conclusions were drawn that this project has 5 phases, divided up into a total of 11 sub-phases.

*For a list of each phase and sub-phase, please see refer to the appendices, Appendix 2 – a list of each phase.*

**Creating the task list and Gantt chart**

Each sub-phase was used to analyse what needed to be done and helped to produce a list of tasks. Each task had a start, finish and time allocated attribute which meant that creating the Gantt chart was as simple as transferring the list into excel format.

One issue however was that it initially had 42 tasks. Upon advice from Gani, the task list was further analysed and trimmed down to 29 tasks by eliminating some granularity and merging various <1day tasks into one more substantial task.

The chart also prompted that frontend testing was completely missed which meant that an already optimistic schedule had to be further adjusted to squeeze in another very important sub-phase.

**Arguing time estimates**

The allocated time of 10 days for report writing will be enough. This is because another 17 days have been indirectly allocated in other phases, they are split between 2- and 3-day periods at the end of every major sub-phase. This gives a chance to write the relevant report sections while the experience is still fresh.

The Gantt chart shows that after every phase has been completed, an initial draft of that phase for the report will be written and when the end of the project comes, an additional 10 days will be available to polish off the report and write the missing sections.

An optimistic estimate of 14 dayshas been allocated for the mobile interface and business logic part of the project. This was a trade-off from the original planned time of 23 days to provide extra time for automated e2e testing, which is very important, especially so in the case of business logic being changed to enhance usability later during the project.

Notice no time for requirements gathering has been allocated, this is due to the project being aimed at software engineers at large, rather than a specific client. This allows for all features of the application to be planned during the design phases without relying on clients, but my own vision for the project.

Another optimistic estimate can be seen in the backend development phase of the project. With only 3 days to implement controller logic, the quality of the endpoint and database planning will be essential to staying on track during this task. The next task also includes extra development time if any bugs are found during manual testing.

One pessimistic estimate was the one given for automated testing. This is an area of little experience and additional leeway has been planned in-case of any implementation difficulties.

One last consideration is the Gantt chart assumes that work will be carried out 7 days per week for the duration of the project. This has been taken into consideration and will be the case, as demonstrated by the screenshot in the appendix working 7 days per week for months is not new to me.

*Please see appendix 3 – work ethic*

*Please see appendix 4 – Task list*

*Please see appendix 5 – Gantt chart screenshot*

**Risks**

All risks lead to a potential consequence of the project not being delivered on time or an unfinished, buggy product being delivered.

However, some risks are more likely to happen than others, with each being categorised and a mitigation strategy being put in place, the chance of failure is lowered.

|  |  |  |  |
| --- | --- | --- | --- |
| **Software/hardware risks** | **Chance** | **Consequence** | **Mitigation** |
| Loosing unsaved work | Low | Late delivery | Using GitHub to backup work |
| Computer malfunction/breaking | Low | Inability to develop the product | Insurance to fix the computer, borrow family members laptop |
| **Skillset risks** | **Chance** |  | **Mitigation** |
| Inexperienced concepts such as automated testing | Medium | Falling behind schedule | Pessimistic time estimates for testing to allow leeway. |
| Unknown concepts such as web security | High | Falling behind schedule | Starting the literature review ASAP and learning about web security now before the project starts |
| **Planning/schedule risks** | **Chance** |  | **Mitigation** |
| Insufficiently detailed designs such as the endpoints, database and wireframes | Medium | Falling behind schedule | **Unknown** |
| Poor productivity | Medium | Falling behind schedule | Software design phase before development to give direction |
| Scope creep | High | Not completing in-scope tasks | Gantt chart with a full task list to know what is in and out of scope |
| Incorrect time estimates | Very high | Falling behind schedule | Agile and scrum development methodology for flexibility |
| **Product risks** | **Chance** |  | **Mitigation** |
| Competitors | Unknown | Loss of potential revenue | Limiting usability testing to a trusted set of people. |
| Non user friendly final product | Medium | Abandonment of the application by users | Usability testing on real humans |
| Insufficient testing | High | Buggy, unusable final product | Pessimistic time estimates for testing to allow enough time. |
| **External risks** | **Chance** |  | **Mitigation** |
| Becoming too ill to work | Very low | Falling behind schedule | Ask for an extension |
| Failure to obtain UH ethics approval | Very low | Inability to carry out primary research on usability testing | Applying in good time or running usability tests only on myself. |
| Dependency on open source | Medium | Buggy software | Minimising package usage |

**LSEP Issues**

**UH Ethics approval - Very important**

No primary research data can be collected until the approval has been acquired from the university. Regardless of who the person is or how many people will be studied, approval is needed.

The process will begin once a supervisor has been assigned to the project and study plans have been discussed. After the relevant approval forms have been filled in, the supervisor will submit them to the HSET ECDA who will then conduct a risk assessment.

After the risk assessment, approval will be granted or denied.

**Multiple category issues**

**Privacy: - Very important**

Perhaps one of the most important issues, privacy of data falls under the legal and social category of issues. There are many laws in place which govern which data is to be collected and how it is to be handled. Furthermore, it is a social and ethical responsibility to keep people’s data safe.

Although this application will not hold sensitive data for example, credit card info, the laws must be adhered too and only data essential to the function of the application will be collected. Furthermore, it will be secured via the various planned web security features.

**Legal issues**

**GDPR: - Very important**

This European law, to be become the UK data protection bill, describes how data should be processed and stored and sets out consumer rights.

For this project it means data must be:

* Relevant to the application
* Stored for a minimum time period
* Up to date
* Encrypted
* Erasable

Furthermore, a data protection officer must be appointed. (GDPR in a Nutshell - What You Should Know About the Changes Regarding European Data Privacy, 2019)[9]

**Copyright:**

This law was put in place to prevent people from stealing each other’s ideas and work. It applies to this project because other code-snippet managers exist.

However, “copyright law is about protecting a particular *expression* of an idea, not the idea itself, and not functional elements of a work” (Bond, 2018)[10]. The plan is to build this project completely from scratch with an exception of open source packages, where copyright is not applicable.

**Social issues**

**Quality, usability and testing of the finished product:**

This product is primarily aimed at software engineers at large who have the choice whether they want to use it or not. Furthermore, the secondary target, general students at large, also have the choice whether to use it or not.

Therefore, the product must be thoroughly tested for bugs and usability to ensure that it is not abandoned completely.

Also, a reliable product lowers the risk of any social backlash if they become reliant on it for their source of revision.

**Availability**

Especially so in 2020, the public consume software on an ever-expanding range of devices. This application needs to be available on many devices to include as many customers as possible.

**Social backlash**

This project has a low risk for social backlash because it will provide a useful service to anyone that uses it. Nobody will be directly impacted in a negative way from this application.

However, one small risk is that people may lose all their notes in the case of a database failure. This would cause trust issues and social backlash; therefore the database must be secure and reliable.

**Ethical issues**

**Attribution**

This application will most likely use media such a images and icons which were created by other people. One solution would be to pay for media, however, since this is a university project, free media will be used.

This brings rise to another issue of attribution. Many sources of free media come with a mandatory attribution to the author if they are used in a project, therefore this project will have to list attributions somewhere.

**Plagiarism**

Apart from the use of open source technologies such as React, node and various packages, all work in the project will be original. All business logic and algorithms will be written from scratch, the open source will provide the framework to build the application on.

**Professional issues**

The British computer society a professional body which guides software engineers how to create software responsibly, has a code of conduct which has 4 main areas.

**Public interest** :

This project has already considered public interest where applicable with various security and accessibility features. People’s data will be secure, and the application will not discriminate against anyone.

**Professional competence and integrity :**

The professional competence does not apply to this project as services are not being sold to anybody and the legislation has been considered and will be complied with. Respecting others goes without saying and will not be an issue for this project.

**Duty to relevant authority :**

The UH ethics approval procedure will be followed, and work will not be plagiarised.

**Duty to the profession :**

The BCS code of conduct will be adhered to along with various laws to ensure that everything is done correctly, and software engineering remains a well-respected profession.

**Proposal summary**

This application will be developed to a high standard with many measures for security in place. The literature review will provide in-depth knowledge on what needs to be implemented.

In addition to this, the existing skills and experience will allow the API, interface and business logic to be implemented to a professional standard. Furthermore, additional leeway has been allocated for areas which may need them such as design and testing.

There are many risks, some major and some minor which have been accounted for and a mitigation plan has been put in place to cope with them. Lastly the LSEP issues of the project have been reviewed and will be considered during the development of the project.

After writing this project proposal, the next step would be to undertake the literature review to get an idea on the current state of web security exploits and protection methods.

**References**

[1] - SHALINI, S. and USHA, S., 2011. *Prevention Of Cross-Site Scripting Attacks (XSS) On Web Applications In The Client Side*. [online] Research Gate. Available at: <https://www.researchgate.net/publication/267918987\_Prevention\_Of\_Cross-Site\_Scripting\_Attacks\_XSS\_On\_Web\_Applications\_In\_The\_Client\_Side> [Accessed 14 March 2020].

[2] - Hou, B., Qian, K., Li, L., Shi, Y., Tao, L. and Liu, J., 2016. *Mongodb Nosql Injection Analysis And Detection*. [online] Research Gate. Available at: <https://www.researchgate.net/publication/306300665\_MongoDB\_NoSQL\_Injection\_Analysis\_and\_Detection> [Accessed 14 March 2020].

[3] - G. Sowmya, 2013. *BRUTE FORCE ATTACK – BLOCKING TECHNIQUE*. [online] Ijecs.in. Available at: <http://www.ijecs.in/index.php/ijecs/article/view/1810> [Accessed 14 March 2020].

[4] - James, A., 2012. *How To Protect Your Server Against Dictionary Attacks – Linux Academy*. [online] Linux Academy. Available at: <https://linuxacademy.com/blog/linux/how-to-protect-your-server-against-dictionary-attacks/> [Accessed 19 March 2020].

[5] - He, X. and Yang, X., 2017. *Authentication And Authorization Of End User In Microservice Architecture*. [online] Iop science. Available at: <https://iopscience.iop.org/article/10.1088/1742-6596/910/1/012060/meta> [Accessed 14 March 2020].

[6] - Aggarwal, S., 2018. *Modern Web-Development Using Reactjs*. [online] Ijrra.net. Available at: <http://ijrra.net/Vol5issue1/IJRRA-05-01-27.pdf> [Accessed 14 March 2020].

[7] - Crawford, T. and Hussain, T., 2016. *A Comparison Of Server Side Scripting Technologies*. [online] Csce.ucmss.com. Available at: <https://csce.ucmss.com/cr/books/2017/LFS/CSREA2017/SER3291.pdf> [Accessed 14 March 2020].

[8] - Gyorodi, C., Percherle, G., Gyorodi, R. and Olah, A., 2015. *A Comparative Study: Mongodb Vs. Mysql*. [online] Research gate. Available at: <https://www.researchgate.net/publication/278302676\_A\_Comparative\_Study\_MongoDB\_vs\_MySQL> [Accessed 19 March 2020].

[9] – SoftwareONE. 2019. *GDPR In A Nutshell - What You Should Know About The Changes Regarding European Data Privacy*. [online] Available at: <https://www.softwareone.com/en-gb/blog/articles/2017/04/11/gdpr-in-a-nutshell> [Accessed 23 March 2020].

[10] - Bond, K., 2018. *Copyright Law Basics For UK Software Developers — Smashing Magazine*. [online] Smashing Magazine. Available at: <https://www.smashingmagazine.com/2018/03/copyright-law-basics-for-uk-software-developers/> [Accessed 23 March 2020].

[11] -Wax, D., n.d. *Scrum For One*. [online] Lifehack. Available at: <https://www.lifehack.org/articles/featured/scrum-for-one.html> [Accessed 31 March 2020].

**Appendices**

**Appendix 1 – Data to be collected on human subjects**

The following data will be collected :

* Age
* Gender
* Usability data/any other feedback about the application

Aside from the age and gender, no personal information will be collected and each subject will remain anonymous.

**Appendix 2 – Full description of resources and skills**

**React.js – for the UX**

An extremely powerful interface building framework which can be used on mobile, web and desktop applications, React was chosen because of the familiarity, re-usability, speed. “*The reason for highly efficient performance of the framework is essentially is the virtual DOM feature of the framework.”* (Aggarwal, 2018)[6]

**Node.js and express – for the API**

A fast, modern, non-blocking runtime, Node.js facilitates the use of JavaScript as a server-side programming language. Express is a node.js framework for building servers makes building web servers very easy. Node and express were chosen due to their speed and familiarity. “*Node.js is a very fast technology that allows for event driven, non-blocking I/O giving it an extra boost compared to the other technologies*” (Crawford and Hussain, 2016)[7]

**MongoDb and Mongoose – for the Database**

A flexible, modern, non-relational database, MongoDb is an alternative to traditional databases such as SQL. “*MongoDB provided lower execution times than MySQL*” and “*a more flexible structure“* (Gyorodi, Percherle, Gyorodi and Olah, 2015)[8], are just a few reasons why MongoDb was chosen. Mongoose is a framework for SQL for MongoDb and Node.js which makes interacting with the database a breeze.

**Visual Studio Code**

This ide provides many features out of the box including autocompletion, an integrated terminal and syntax error detection. It is a very powerful web development suite used by millions of developers over the world.

**Firebase and Heroku**

Google firebase and Heroku are free hosting services which will be used to host the front and backend of the web app respectively.

**FlatIcon and other sources of free images**

Royalty free images will be used for the iconography and images in this project, in many cases the author will have to be credited somewhere on the web site.

**BrowserStack**

To test the responsiveness of the application, BrowserStack, a paid online platform which provides real devices, operating systems and browsers to test web applications on will be used. The extensive collection of devices and browsers range from iPhone 5 to iPad Air to desktop Firefox and everything in between.

**Chrome developer tools**

Chrome dev tools will be used to debug the application. It has a full suite of tools including a console, network information, local storage and much more.

**Desktop computer including internet access**

The application will be developed on my personal computer which has all the necessary software installed.

**Github**

Github is a version control system which is very familiar. In addition to providing a platform for team collaboration, it can be used to securely store code and revisit previous code in case of an error.

**Postman**

An open source library used to test API endpoints without a real front-end sending the requests.

**Bcrypt**

A library used for encrypting passwords so they are not exposed in plain text in case of a database breach.

**Helmet**

A library used for web security. It adds various headers, filters requests for basic XSS protection and removes some vulnerable headers.

**Report writing and referencing skills**

A dissertation style report will need to be written alongside the development of the project which will require report writing and referencing skills. The report will need to be both professional and engaging to the reader.

**Web application security measures**

Security features such as XSS, CSRF and brute force protection are currently unexplored and will need to be researched before their implementation. This research will be carried out during the literature review and secondary research phase of the project.

**Physical devices**

Various physical devices will be required to test the application on, such as mobile phones, tablets and laptops. These devices will be borrowed from family and friends with their permission.

**UH ethics approval**

Ethics approval will be required from the university to carry out usability primary research on human subjects.

**Integration and end to end testing**

This application will need extensive testing, the methods chosen will be automated integration and end to end testing via Cypress. This type of testing is mostly unfamiliar and will need to be studied more.

**Interactive tutorial**

A tutorial to help new users and reduce the risk of abandonment will be implemented. It will show the users how to use the application by providing interface overlays which will rely on logic-heavy algorithms and have multiple stages of the tutorial. Each stage will show how to use a certain feature, e.g. how to upload a piece of syntax.

**Project management skills**

Various small projects have been undertaken previously, however, they did not involve this much preparation and they were not to this scale or seriousness.

**Project methodology**

“Scrum for one”, (Wax, n.d.) [11]is an agile method designed for 1 person. It involves constant reflection of ones work in order to improve and adapt continuously.

In the context of this project, each piece of functionality will be planned, developed and tested to create a working prototype. Rather than shipping a piece of the product for a customer to review, it will be given to humans for usability testing for feedback.

**Appendix 2 – a list of each phase.**

**Setup**

* Setting up the environment, GitHub repositories database and report

**Backend**

* Design and analysis
* Development
* Security
* Automated testing

**Frontend**

* Design and analysis
* Development
* Automated testing
* Interface responsiveness

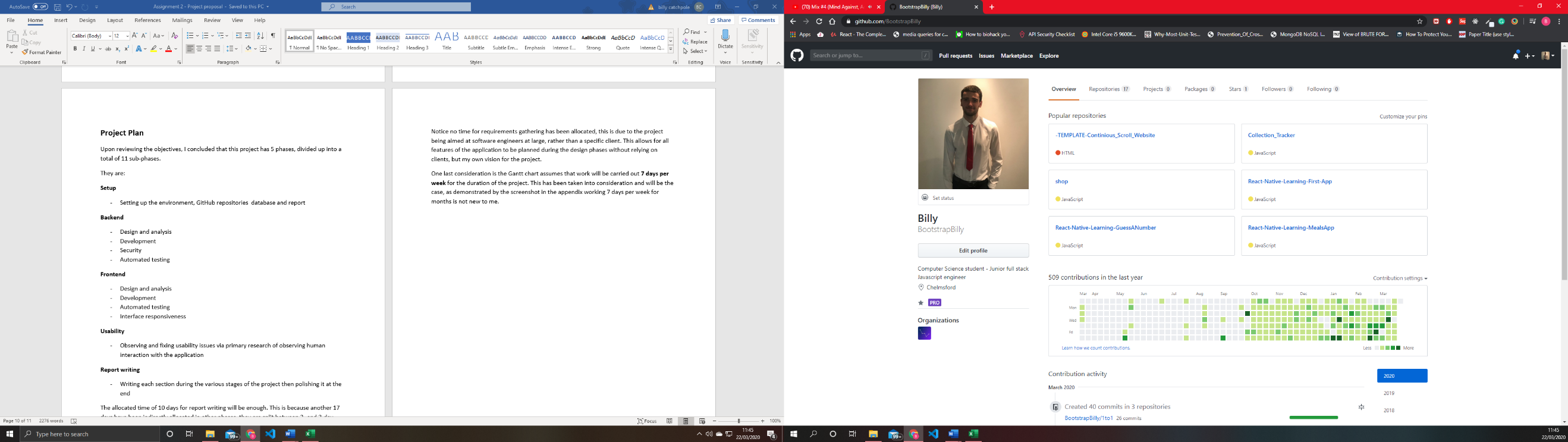
**Usability**

* Observing and fixing usability issues via primary research of observing human interaction with the application

**Report writing**

* Writing each section during the various stages of the project then polishing it at the end

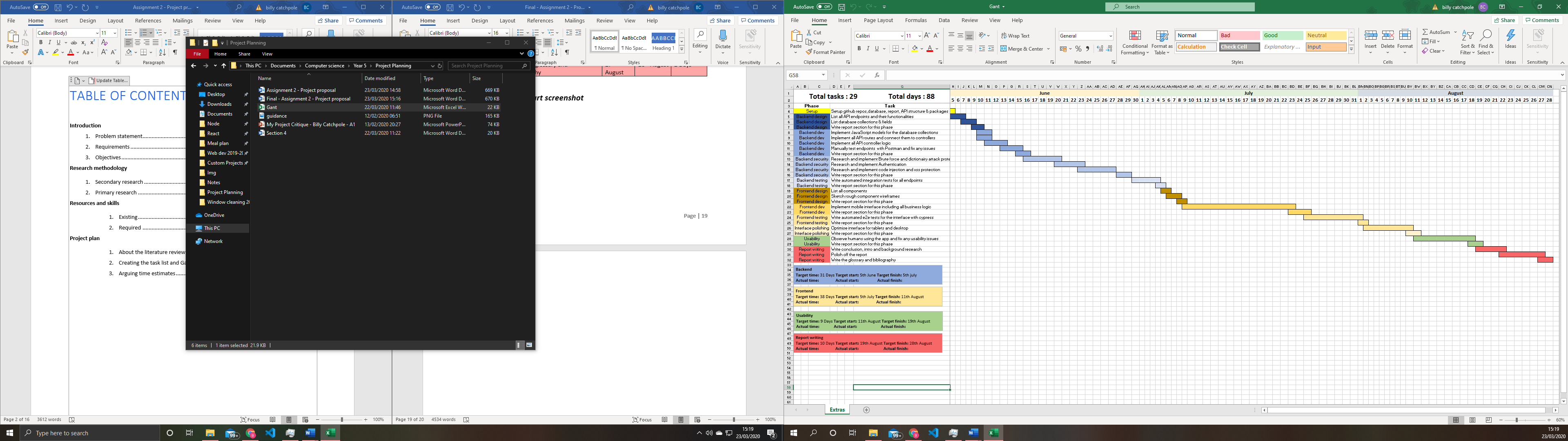
**Appendix *3 – work ethic***

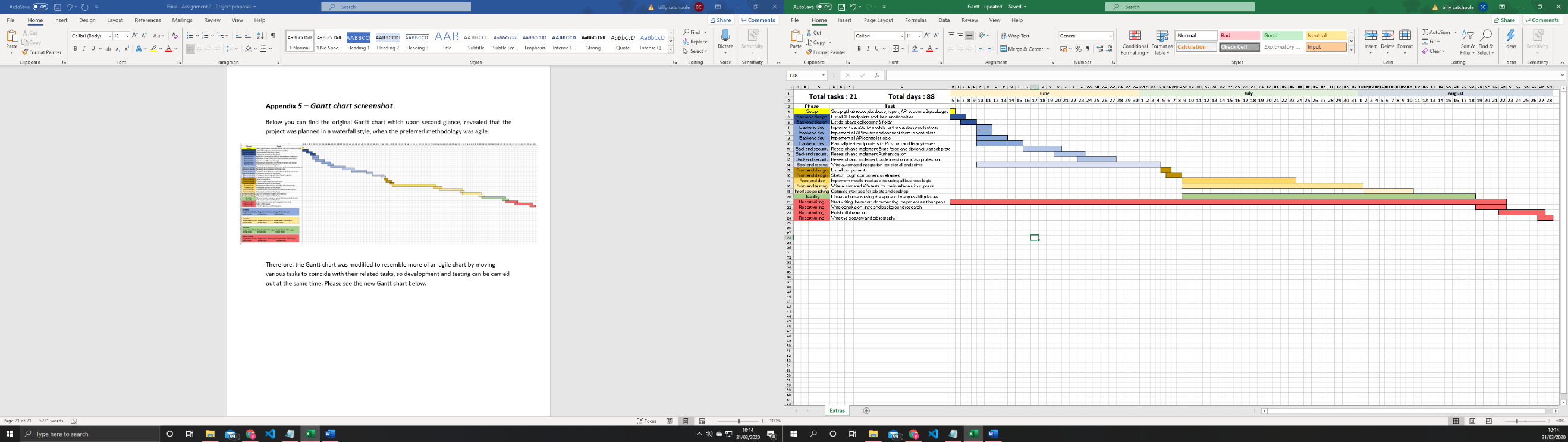
****Evidence of development activity on GitHub over the past 6 months. As you can see it averages around 6.7 days per week(excluding the Christmas period), which means undertaking this project on a 7 day per week schedule should not be a problem.

**Appendix *4 – Task list***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Project phase | Task description | Start Date | Deadline – end of the day | Duration |
| Setup | Create a project folder and **GitHub repository** for the **API and interface**, then connect them via the Git CLI  Create the **database** and the **report document**  Setup the **API structure** including **DB connection** and packages such as **helmet** | 5th June | 5th June | <1 Day |
| Backend analysis and design | Compose a **list of** all required API **endpoint functionalities** and include them in the live documentation | 5th June | 7th June | 3 Days |
| Backend analysis and design | Compose a **list of** all required database **collections**, along with a list of their **fields,** theninclude them in the live documentation | 7th June | 9th June | 3 Days |
| Backend analysis and design | Write the **initial draft** of the final report for the **backend** **analysis and design** phase. | 9th June | 10th June | 2 Days |
| Backend development | Using the list of collections and fields and Mongoose, **implement the JavaScript models** for each collection. | 10th June | 11th June | 2 Days |
| Backend development | **Implement all** **routes** for the endpoints in the API and connect them to their controllers | 10th June | 11th June | 2 Days |
| Backend development | **Implement all** **controllers** including their logic to complete every endpoint | 11th June | 13th June | 3 Days |
| Backend development | Manually **test** each endpoint using **postman** and fix any bugs found | 13th June | 15th June | 3 Days |
| Backend development | Write the **initial draft** of the final report for the **backend development** phase. | 15th June | 16th June | 2 Days |
| Backend security | **Research**  **and implement brute force** attack and **dictionary attack** protection. | 16th June | 20th June | 5 Days |
| Backend security | **Research and implement authentication and authorisation** | 20th June | 23rd June | 4 Days |
| Backend security | **Research and implement xss and code injection protection** | 23rd June | 27th June | 5 Days |
| Backend security | Write the **initial draft** of the final report for the **backend security** phase. | 28th June | 29th June | 2 Days |
| Backend testing | Write **automated integration tests** for each API endpoint, fixing any bugs discovered on the way | 30th June | 4th July | 5 Days |
| Backend testing | Write the **initial draft** of the final report for the **backend testing** phase. | 4th July | 5th July | 2 Days |
| Frontend analysis and design | Compose a **list of** **all components** including pages, stateful components and dumb components | 5th July | 6th July | 2 Days |
| Frontend analysis and design | Sketch a rough **wireframe** of every component identified in the previous step | 6th July | 8th July | 3 Days |
| Frontend analysis and design | Write the **initial draft** of the final report for the **frontend** **analysis and design** phase. | 8th July | 9th July | 2 Days |
| Mobile UX design and business logic implementation/backend integration | One component at a time, **implement** the **mobile interface** including all **business logic** (state management, API calls, utility methods, data manipulation) | 9th July | 23rd July | 15 Days |
| Mobile UX design and business logic implementation/backend integration | Write the **initial draft** of the final report for the **frontend development** phase. | 23rd July | 25th July | 3 Days |
| Frontend testing | Write automated e2e tests for the interface with cypress | 25th July | 1st August | 8 Days |
| Frontend testing | Write the **initial draft** of the final report for the **frontend** **testing** phase. | 1st August July | 2nd August | 2 Days |
| Interface polishing | **Optimise** the interface for a range of popular **tablets** including iPad, Google Nexus and Samsung Note and **desktop** devices. | 2nd August | 10th August | 9 Days |
| Interface polishing | Write the **initial draft** of the final report for the **interface polishing** phase. | 10th August | 11th August | 2 Days |
| Usability enhancement | **Observe humans** using the application to uncover and **fix** **usability** issues | 11th August | 18th August | 8 Days |
| Usability enhancement | Write the **initial draft** of the final report for the **usability enhancement** phase. | 18th August | 19th August | 2 Days |
| Report writing | Write the **initial draft** of the **project conclusion, introduction and background research** | 19th August | 22nd August | 4 Days |
| Report writing | Polish off the report including enhancing the grammar, writing style and clarity and proofreading every section | 22nd August | 27th August | 6 Days |
| Report writing | Write the glossary and bibliography | 27th August | 28th August | 2 Days |

**Appendix *5 – Gantt chart screenshot***

Below you can find the original Gantt chart which upon second glance, revealed that the project was planned in a waterfall style, when the preferred methodology was agile.

Therefore, the Gantt chart was modified to resemble more of an agile chart by moving various tasks to coincide with their related tasks, so development and testing can be carried out at the same time. Please see the new Gantt chart below.

29 tasks were trimmed down to 21 tasks, by merging all time allocated for writing the report section at the end of each phase into 1 task. This task now runs from the beginning to the end of the project, allowing a more flexible, agile approach.

Usability testing changed to start from the beginning of the frontend development task, as it is dependant on an interface for the subjects to interact with.

Front and backend testing tasks were moved to the start of frontend and backend tasks respectively. This is because they are not dependant on any other tasks.

Lastly, the backend phase of the project was modified so that all development and manual testing starts from the same date, as they are all dependant on the backend planning phase. However, their deadline remains the same.

One caveat of this agile thinking is that the frontend is still dependant on the backend being finished. This is not very agile, however, in a real project there would likely be different teams working on each end simultaneously.