Project Reflection

DE BOER, Lucas (k2023556)

Kingston University

1. Introduction

The year-long project has come to an end. The reality of what I was able to produce now faces me. I have learnt a lot along the way, and with the curse of hindsight, I would have done many things differently. This reflection document will be precisely that: a reflection of the project's success and failures.

This document will review the artefacts generated throughout the project and express the things I have learnt with the help of stories collected along the way. All of which will have hopefully made me a better software engineer

2. Review of Project Artefacts

2.1. Introduction

The project utilises a combination of agile and Scrum methodologies. Scrum was used for its time-limited iterations called sprints and brief daily meetings. However, the meetings for this project only occurred once a week. The sprints lasted for around four weeks. By the end of each sprint, an artefact or product, such as a document or new system functionality, was completed.

This methodology created many artefacts throughout the project's lifetime, from documentation to code. This section will review these artefacts.

2.2. Status of Functional Requirements

This sub-section will review the functional requirements implemented into the system. Due to the project's time constraints, not all functional requirements were met. Instead, the most critical requirements must be prioritised first, using the MoSCoW technique.

MoSCoW prioritisation puts the functional requirements into four categories: must-haves, should-haves, could-haves, and won't-haves. The must-haves are the requirements that are fundamental to the system's functionality and, thus, pivotal to the success of the project. The project's client then validated and agreed upon this list.

However, even after focusing the development sprints on the must-have requirements, the project still struggled and failed to implement them. This could result from the time constraint or simply because the technology used throughout the project was new and thus unfamiliar.

2.3. Digital Artefacts

This sub-section will review the implementation of the project's frontend, backend, and database. Furthermore, a review of how well the project could follow best practices, patterns, and guidelines during its development will also be conducted.

Firstly, an overview of the project will be established. The project is a modern web application. The application uses a RESTful API to create a backend communicating with a MySQL database. The SPA aspect of the application essentially means that a full-page refresh will never occur. Instead, a SPA interacts with the user by dynamically rewriting the current web page or index file; only one index.html file is present. The data needed for the dynamic rewriting of the page is fetched from a RESTful API built using the Express JS framework.

An API or Application Programming Interface enables the "client" REACT-based web application to access resources within the MySQL database "server". REST or Representational State Transfer imposes a set of guidelines or conditions on how an API should work. APIs which follow the REST architecture style are called RESTful APIs. The REST architecture style includes principles such as statelessness. Statelessness implies the server will not store anything about the client's latest HTTP request; It will execute every client request independently of all previous requests. Thus, the client can request data at any time in any order, and the server will be able to fulfil these requests.

Frontend Development

The client-facing web application is written in REACT JS. To create a robust development environment, the project was made using the Vite build tool instead of the standard "Create React App." Both have a local host server, which allows the app to be viewed in Real Time during development. However, Vite has superior hot reloading speed, with it instantaneously and the CRA taking around five seconds longer.

REACT allowed me to use a new tool kit for developing web applications. With hooks and the capability to make many reusable components, REACT allowed the project to take a well-structured and easy-to-understand form. Reusable components were used throughout the project. Development was streamlined by eliminating the need for redundant code, and resources can be utilised more effectively. While reusable components were used, I did not get to make many of them as they still require extensive refactoring to ensure they are stable.

Moreover, Identifying and creating reusable components was essential to maintaining a consistent layout throughout the system. Learning and recognising new designs and elements can be time-consuming for many users. Therefore, using the same UI elements in the same places for every screen is better. This will hopefully help new users learn and use the system more efficiently.

The system was planned to be designed and developed so people with disabilities could use it. Everyone could perceive, understand, navigate, and interact with the system. A sans-serif font size of at least 16px was supposed to ensure that most people could easily read the content. Additionally, the length of the lines was planned to be optimised by ensuring they were not too long or too short, with the ideal range being between 45 and 75 characters per line. These features were not implemented into the prototype because of time constraints and low prioritisation.

However, the project has eliminated the exclusive use of colour to convey information. For example, using red to indicate an error occurrence in a form field is insufficient. To address this issue, the developers have incorporated the red colour into a text label that explicitly states the nature of the error and its cause so that the error can still be perceived.

A navigation diagram was made to represent the website structure and hierarchy visually. This diagram helped ensure the web application's crucial functionalities were easy to navigate. Furthermore, the three-click rule was kept in mind throughout the project to ensure that the system never required the user to make unnecessary clicks.

Lastly, the web application is responsive to different screen sizes with SASS styling rules.

Backend Development

The Express JS framework was used to develop a RESTful API. This backend acts as a middle layer between the REACT-based front end and the MySQL database, which holds all the information the front end needs.

A series of endpoints were created for the front end to communicate with the backend. These essentially act as a way for the front end to specify what information the backend needs to query the database for. All endpoints have adhered to some best design pattern practices to ensure consistency and ease of understanding. All URLs include nouns and do not include verbs. These nouns are always plural for consistency throughout the system. Furthermore, while not considered best practice, all endpoint syntaxes followed a pattern of defining the object type that needs to be returned as the first noun in the endpoint following /API. Thus, an endpoint to return all boats would look like "/api/boats". The endpoint syntaxes for listing, adding, viewing single objects, editing, filtering, searching, pagination, and sorting were all specified in the "Endpoint Specification" document.

The front end needs to query for many database entities such as boats, employees, booking and equipment. While a single file could handle all these requests, it would be impractical and complicated to code. The backend uses a series of routers and routes to resolve this issue. A file containing only the endpoints specific to a database entity, such as employees, is called a router. A router was made to handle the endpoint for boats, employees, item reservations, and bookings.

The create, read, update, and delete methods were generalised and held in a single " Accessor " class. This class was thus responsible for querying the database, and many different SQL generation methods were used to make the SQL required to query the database for the requested information. These SQL queries were all made into prepared statements to help prevent SQL injection.

A “Controller” class used these accessor methods. A “get” method in the controller class utilised the “read” method of the accessor class. The “post” is a “create”. The “put” is an “update”, and the delete is as expected.

The controller class ensures that the post and put requests passed their respective validation checks, which were handled and set up using the “express validator” NPM package. Furthermore, it sets the response object with the database query result. The front end will then use this response object to populate the screen with the relevant information.

Database

Installing PhpMyAdmin was very straightforward with the help of XAMPP. XAMPP provided an open-source, cross-platform web server called Apache and the MySQL database. These ran as local host servers on different ports.

The database was void of any data required for the web application to function; thus, inserting dummy data was a necessity and required a great deal of patience. ChatGPT was used to help generate this data as it could tailor a CSV to match my request; at least, in practice, it should do this. ChatGPT often gave out entirely bogus data, frequently requiring me to change the generated data.

To connect to the database, the express backend was given the name of the database, the port number to listen to, the host and the user. With this and the help of the MySQL package “mysql2/promise” a connection was created.

Guidelines, Patterns, and Best Practices

As stated previously, the application is both responsive and follows design guidelines. Furthermore, the endpoint syntaxes have been discussed above, emphasising that all endpoints have adhered to some best design pattern practices to ensure consistency and ease of understanding.

A folder structure pattern was adhered to ensure the project was easy for developers to navigate and maintain. Folders separated the project files into logical parts based on their purpose and use. The names of the files were constructed in a way that should hint to the developer about their functionality.

2.4. Technical Documentation

Technical documentation was generated throughout the project’s lifetime, with each document summarising what was accomplished during a specific phase of the project development life cycle. Each document was written with the perspective that a new developer would use it to familiarise themselves with the system’s architecture and layout. Each document was formally written and structured, which was a challenge every time.

The user requirements specification has become a handy reference when planning sprints. This document contains a list of all MoSCoW prioritised requirements with the reasoning and client validation behind them. It summarises the processes undertaken during the requirement extraction phase of the project. A comprehensive list of user stories was created for each user type, along with use case texts.

The user interface specification document outlines the design process and evaluates how the system will meet its accessibility and user interface design goals. It details low—and high-fidelity wireframes, UML activity, and sequence diagrams to visualise the system’s workflows.

The data model specification document aimed to help the reader understand the relationships between the different entities in the system. It specified how an entity relational diagram was constructed for each user story, allowing developers to easily comprehend the requirements that must be implemented during a sprint. Smaller constellations of the system’s ER diagram were used to validate them. SQL queries were generated for each constellation to verify that the proposed relationships were feasible.

An endpoint specification document was also created, which specified that all endpoints adhered to best design pattern practices. This ensured they were consistent and easy to understand. Syntax formatting examples were given for each endpoint type, such as listing, adding, viewing single objects, editing, filtering, searching, pagination, and sorting. This document acts as a manual for developers to reference and understand how to implement endpoints in a way that follows the rest of the system.

Lastly, I wrote a literature review of database security and protection measures. This document was the first formal writing I had done for the project and, thus, was the hardest to write. The report required extensive research and planning and knowledge of evaluating the usefulness and validity of information sources.

2.5. Conclusion

I have created many artefacts that fulfil specific project requirements during the year. These include helpful technical documentation for future reference and code that addresses the prioritised user requirements. Additionally, the code demonstrates my learning journey as I have acquired knowledge of the REACT and Express frameworks.

This section has evaluated the project artefacts. It has reviewed the status of the functional requirements and addressed the time constraints that have meant that not everything was implemented. Furthermore, it evaluated how the front, backend, and database were implemented and how these followed best practices.

The review has provided valuable insights into the project's strengths. It has identified areas that require further attention and improvement and aspects that have been successfully implemented. The section's findings will guide the development process in the future.

3. Impact on Personal Growth

This section will first revisit the project’s success criteria. Later, the evaluation of the project's success will be discussed, using the numerous stories accumulated to justify it.

Before delving further, it should be stated that personal growth can take many forms depending on who you are and where specifically you look. For this document, personal growth will be defined as the development of one’s capabilities and potential as a professional to facilitate employability and the realisation of career aspirations

3.1. Success Criteria

A final-year project is usually too ambitious in scope to be fully implemented. Therefore, the project's success can be more usefully measured against various criteria. These criteria were stated at the beginning of the project and will now be revisited.

Learning new technologies

New technologies will be exposed throughout the project and must be learned to a level necessary for the project's progression. These technologies include:

a. JavaScript (SPA and RESTful APIs)
b. React framework.
c. Express JS
d. XAMPP

Coding

Best practices will be adhered to throughout the project as best as possible. This could include looking for code smells, such as repeating code and refactoring it as and when it is found. Furthermore, debugging practices such as console.log and postcode will be used.

Version Control and Code Management

Along with having a code backup, branching and forking will be utilised for experimental purposes while learning new technologies.

- This will be evidenced by the project's GitHub repository, updated at the end of every code session.

Get a better understanding of the Software Development lifecycle.

Client centrality will be practised as the project continues. Client communication will take place at least once a month.

Agile development

As mentioned in the "Project Management Approach" section, this project will be completed over weekly sprints, each with its deliverables. Graeme's deliverable document will evidence meeting these weekly deadlines.

Employability

The project can provide numerous attributes which are sought after by employers, including:

a. Evidence of technical knowledge. Which will be presented through a technical report on "Database Security and Data Protection".
b. Recognizing the value of agile software development within the industry.

Cultivating the interpersonal skills and qualities that employers look for in graduates. Evidence of the skill being acquired through

3.2. Reflective Stories

3.2.1. Refactoring Code

I struggled to maintain a router file in my backend. This router had too many responsibilities, making it complex to manage. It contained numerous functions, each with a different goal and purpose, including the PUT, DELETE, POST, GET controllers, and query builders. Due to its complexity, refactoring was needed.

I took the initiative to refactor the router code to make it more organised and easier to manage. As a result, I separated each PUT, DELETE, POST, GET controller, and query builder into their respective files and folders. This approach made the router file more readable. However, I later discovered that it also made maintaining and debugging the project's file structure more challenging.

At the time, I thought this micro separation of concerns was good. However, I was blind to how bloated my code file structure was becoming, and at times, I found it challenging to find the specific file that needed debugging. It was slowly becoming unmanageable as more endpoints and entities, such as “employees and bookings”, were required from the server. Each new entity required new bespoke files for PUT, DELETE, POST, and GET controllers, as well as additional query files containing the SQL necessary to support these controllers. I must create at least eight new files to implement employee-focused user stories. If I had implemented this with my approach, it would have significantly increased my project’s file count and made it even worse to search through and maintain. I needed a better solution to make my backend code easier to maintain and scale.

Thankfully, many resources deal with backend refactoring, which addresses the issue of separating concerns and making the server more manageable to maintain, scale, and, most importantly, read and understand. The best practice approach from the onset differed from what I had done and, in hindsight, should have been the obvious way of going about it. After identifying that the GET, POST, PUT, and DELETE controllers were identical for any database entity, he made each controller reusable. After this step, I immediately realised that the issue with my code wasn’t that I separated everything; I had made nothing reusable, and thus, everything had to be bespoke.

Further along the tutorial, a single “Controller” class held the basic CRUDL methods. This baffled me, as I had genuinely forgotten that I could create classes in JavaScript. This simple step of creating a single class meant I could create one file containing all four instead of four separate controller files. The approach drastically reduced the project's file size and made debugging and coding much more manageable.

The process of completely tearing apart my backend to implement a more scalable and future-proof solution felt daunting. I knew it was something that I had to do because it would make my code easier to maintain and read. Throughout the refactoring process, I couldn’t test anything, which made picking apart and rearranging the entirety of my server that much more stressful. Furthermore, the fact that my project didn’t match the tutorial meant I had to truly understand what was being done to translate it to suit my code. This interpreting step turned a one-hour coding session into a four-hour one, and honestly, at times, it made me not want to continue with it.

Furthermore, due to the destructive nature of this refactoring, I made sure to make a Git branch for it. This made the entire session a little less nerve-wracking as I knew that the worst-case scenario would only be that I would have to roll back my code to my original, less effective approach.

This experience, if nothing else, has immediately made me aware of my apparent tunnel vision or lack of critical analytical skills when planning to refactor. As a developer, I have learnt the importance of analysing my code and identifying similar patterns between seemingly bespoke functions or components. This should help make my project less bloated and my code reusable and maintainable.

Furthermore, I learned the importance of working on a branch instead of the central repository. While I didn’t need to revert my code in the end, I still found comfort in knowing I had a working project during my refactoring.

Success Criteria	**This story meets the marked criteria.**
Learning New Technologies	Yes

Coding Best Practices	Yes
Version Control and Code Management	Yes
Better Understanding of Software Development Lifecycle	No
Agile Development	No
Employability	No

3.2.2. Client Centrality

My final year project has a real-world client. Having the client at the centre of development allowed the project to better fulfil the client's needs, preferences, and expectations throughout the entire software development cycle.

The client's living abroad generated many challenges for the project. One occurred during the usability testing phase because the project prototype was hosted on a local server with a local database. This made it difficult to share the project's current build with the client for feedback.

Two potential approaches were considered to test the application with the client. The first option was to allow the client to control my PC with the prototype running remotely. However, this required the client to download additional software. The second option involved transforming the pre-existing wireframes into interactive, high-fidelity, clickable flows that the client could access and run in a browser. I decided to go with the wireframes for ease of use and because it would allow for most workflows to be tested.

During the requirements analysis phase, I encountered another problem. Extracting the client's requirements was challenging as he didn't always have a clear idea of what he wanted. Additionally, some suggested requirements were beyond the scope of the project. These unclear ideas made it difficult to ensure that the project would meet his needs and expectations.

To help mitigate the suggestion of out-of-scope ideas and keep the client’s expectations reasonable, I reminded him throughout the process that this project was only university coursework and that I would likely not extend it onto the project after submission.

While having a friend as a client was lovely, it still gave me insight into how to develop around the client's expectations and requirements.

Success Criteria	**This story meets the marked criteria.**
Learning New Technologies	No
Coding Best Practices	No
Version Control and Code Management	No
Better Understanding of Software Development Lifecycle	Yes
Agile Development	No
Employability	Yes

3.2.3. Code Management Best Practices

The final-year project spans the entire academic year. Thus, it is advisable that I at least attempt to manage my code correctly. Furthermore, I must use a version control system such as GitHub for marking purposes.

As I was new to GitHub, I lacked the knowledge of its proper usage. Additionally, handling a project of such magnitude was uncharted territory for me. Therefore, I deemed it necessary to adhere to coding management best practices to avoid the risk of it becoming unmanageable and exceedingly challenging.

To manage my work effectively, I took the initiative to establish and utilise the Git Version Control System in conjunction with GitHub. I discovered that GitHub Desktop was a valuable tool for facilitating this connection since it provided a user-friendly interface for utilising Git's capabilities without requiring familiarity with command-line instructions.

Having set up Git, I can now commit my project's code to its repository. The repository became more valuable the more committed messages I sent. To ensure that the repository has a comprehensive history, I followed the suggestion that every small code change be committed frequently. Each commit should represent a single change, making it easier to track and revert the code if necessary.

As I kept committing frequently, my repository began to accumulate a long history, making it difficult for me to keep track of the purpose of each commit without looking at the code changes. To address this issue, I decided to start writing meaningful commit messages instead of entering random text to meet the requirement of having a message.

I decided to use Git's branching capabilities at an early stage. I created a new branch for each implementation and refactoring sprint and merged it back to the main branch once the sprint ended. I had some difficulty using branches initially, but with trial and error and a few tutorials, I better understood how to use them.

After following code management best practices, I have become less stressed about potentially breaking my code after a sprint. I now know I have many options to help solve this, including reverting my project to a previous state or simply not merging the sprint to my already stable main build. Version control has allowed me to take risks and has helped facilitate a constant safe learning environment as I explore the unfamiliar functionalities of REACT and Express.

Furthermore, it is paramount that I familiarise myself with these best practices for my future professional career. I will be working in a team and must adhere to established procedures to ensure consistency and comprehension of the shared repository.

Success Criteria	**This story meets the marked criteria.**
Learning New Technologies	Yes
Coding Best Practices	No
Version Control and Code Management	Yes
Better Understanding of Software Development Lifecycle	No

Agile Development	No
Employability	Yes

3.2.4. Debugging Code

It's important to remember that no programmer is perfect, and mistakes are bound to happen. Every coding project is likely to encounter its fair share of errors. In particular, my lack of knowledge of REACT made this final-year project highly susceptible to errors.

Debugging is unavoidable in the development process; resolving an error can take anywhere from a moment to never. This makes my debugging process crucial in identifying and fixing mistakes. However, its time-consuming nature is a problem, as I always work within a limited timeframe. As a result, any delay in debugging can cause implementations or refactoring to spill over into the next sprint, which is not ideal.

Testing the code as I program is a great help. With each minor modification, I ensure that the program still functions properly. This approach can make it easier to locate errors in my code. If the program stops working after implementing a change, the error is most likely found in the code that was just added.

Console logging is a tool that I find extremely useful when debugging. It has helped me tremendously in understanding what the code is doing, unlike what I had initially thought. A simple console log is often enough to help me fix an error. However, if console logging proves insufficient, I would appreciate an error message being thrown. This would enable me to investigate the error more specifically.

Lastly, if nothing worked, the best solution was to walk away from the project and let my mind wander off. By doing this, not only do I let my frustration subside, but I also help decrease the chance of getting tunnel vision. While it is cliché, it is notably hilarious how frequently I magically found a solution to my error mid-taking a shower.

Debugging has proven to be an excellent learning opportunity. It necessitated that I thoroughly understood my code and solidly comprehend REACT. Debugging requires a high level of proficiency in programming languages and an in-depth understanding of the underlying concepts. Through debugging, I enhanced my coding skills and deepened my knowledge of REACT, which has been invaluable in my programming endeavours.

Although debugging provides valuable learning experiences, it was one of the project's most frustrating and enjoyable aspects. The stress of meeting a deadline can make it easy to become impatient and angry; however, the feeling of fixing an error is unmatched.

Success Criteria	**This story meets the marked criteria.**
Learning New Technologies	Yes
Coding Best Practices	Yes
Version Control and Code Management	No
Better Understanding of Software Development Lifecycle	No
Agile Development	No

Employability	Yes

3.2.5. Code Best Practices

The more I am exposed to different technologies I can achieve, the better. Especially during my early years of development as a software engineer. Getting a taste of other technologies and becoming an all-around sound engineer is good. Before starting my final year project, I only knew how to make an interactive website using HTML, JavaScript, jQuery, Ajax, and PHP. These technologies provided me with a great foundation as they allowed me to understand the basics of web development, which could be applied anywhere.

However, I have always wanted to grow as a developer, so I created a single-page application using REACT and Express JS for my final year project. I chose this library and framework because it can support the project's needs and because they are highly sought after by employers.

I had to ensure that I correctly learned these two technologies to confirm that I could apply them in real-world scenarios at a job. I currently do not know precisely what will be expected of me in a job role I don't now have. However, I should learn both REACT and Express to a degree that will enable me to adapt and fill the role, whatever it is.

I must adhere to best practices for each to ensure that I am using REACT and Express JS correctly and in a universally approved manner. One such best practice that is expected from both is maintaining a clear and organised folder structure.

A well-organized folder structure proved extremely helpful throughout the project, especially as it grew each week. However, there were times during development when I had to refactor the folder structure because I found myself getting lost and disoriented more often than I would have liked.

In addition, to maintain consistency in my code formatting, I utilised a tool called Prettier. This linter tool ensured that all my files adhered to the same formatting patterns, which made it easier to comprehend and debug them as my eyes would become used to what looked right or wrong.

Learning about good coding practices has significantly improved the quality of my project. It has also equipped me with the valuable skill of producing code easily understood by other developers in the team. This is crucial for the success of the project. Employers seek candidates who possess a good understanding of coding practices. However, it is worth noting that some companies may have practices that must be followed. I can demonstrate my ability to adhere to these practices by referring to my final year project that was developed using good practices.

Success Criteria	**This story meets the marked criteria.**
Learning New Technologies	Yes
Coding Best Practices	Yes
Version Control and Code Management	No
Better Understanding of Software Development Lifecycle	No

Agile Development	No
Employability	Yes

3.2.6. Technical Writing

I wrote a technical report on database security as part of my final-year project. This technical report was intended to be a self-contained document that would allow me to highlight my technical writing proficiencies to prospective employers.

Although I have experience writing technical reports, I find it challenging to achieve its primary objective, which is to present complex information in a way that is easy for the targeted audience to comprehend. Additionally, technical writing requires brevity and a well-structured format, which includes headings, subheadings, references, and appendices. I struggle with this since I lack proficiency in formal writing, especially during the beginning days of my final year project.

To explain a complex idea simply and understandably, it is crucial to have a deep understanding of the topic. To achieve this, I conducted extensive research using various academic resources. I relied on trusted sources such as ICAT and IEEE, which contain high-quality papers written by respected and verified authors.

I managed to find and save many reports related to the topic. I took the time to read each of them, and after careful consideration, I kept only the most valuable reports. For about three weeks, I dedicated myself to reading as much as possible about the topic, and to keep track of the ideas I encountered, I used a mind map. After each week of reading, I would iterate over the mind map to ensure I captured all the essential concepts. Furthermore, I had to use the Harvard referencing style whenever relevant.

I found the mind map beneficial in organising my thoughts. Visually, it allowed me to see which areas I had delved into in greater detail as they were more populated. When it came time to write the report, I transcribed the mind map to extract around ten to thirteen paragraphs, which helped guide the report's structure and ensure that the topics flowed smoothly from one to the next.

To ensure that my technical report was easy to understand, I sought the help of someone else to proofread my work. This turned out to be extremely useful as there were several instances where I had assumed that I had explained a concept clearly, but in reality, I did not.

Overall, the experience of creating a technical report, from research to the final write-up, was invaluable. I learned how to evaluate sources of information based on the criteria of a report’s currency, relevance, authority, accuracy, and purpose. Using these criteria to judge, I ensured that all the sources I referenced in my technical write-up were reliable.

Through this project, I demonstrated my ability to conduct detailed research, analyse complex information, and present findings concisely and professionally, especially regarding the language used throughout the document, its structure and use of Harvard referencing. Employers consider these attributes valuable, increasing my chances of employment.

Success Criteria	**This story meets the marked criteria.**
Learning New Technologies	Yes

Coding Best Practices	No
Version Control and Code Management	No
Better Understanding of Software Development Lifecycle	No
Agile Development	No
Employability	Yes

3.2.7. Coding Interview

During my final year project, I had to apply for jobs throughout the year. Most software development jobs have multiple application stages to reduce the number of applicants and help the employer decide who to hire. One of these stages is the coding interview, often seen as a daunting challenge.

The coding interview process may differ depending on the employer; however, specific standard questions can be used to prepare. This can include coding or algorithmic problems.

The main problem with coding interviews is that I have no experience doing them. I am used to coding alone with nobody watching, making it stress-free and easy. However, the thought of having somebody watch me as I code and expect me to articulate my thought process gives me anxiety every time I think about it. Furthermore, the concepts usually asked about can be complex if I am not fully prepared.

Concepts such as knowing how to implement depth-first and breadth-first searching algorithms are not complex per se. Still, they can be easily unpractised, making it challenging to showcase knowledge in the stressful environment of a coding interview.

To familiarise myself with the concept of a coding interview and get some practice, I had to partake in a mock code interview. As part of this mock interview, I would be asked a series of questions picked from a disclosed list of possible topics. Having this topic list available before the interview allowed me to prepare for it. Furthermore, I knew I would have to implement a depth-first and breadth-first algorithm to search through a tree.

I researched and practised these topics in the context of the JavaScript language to learn how to implement the selected algorithms. I set up a VS code project where I could freely practice them and, more importantly, analyse how each algorithm works by experimenting with the code and ensuring I knew the purpose of each line.

The mock interview helped me prepare for an actual coding interview, hopefully making it less stressful. During the process, I learned about several JavaScript topics, such as iterators, higher-order functions, and the differences between var, let, and const. The last topic has already significantly impacted my work because I used the var keyword excessively throughout my final year project without realising the eventual consequences of such actions. Furthermore, the interview reminded me of my difficulty expressing my thoughts clearly. I acknowledge that improving this skill will be essential not only for my career but also for my daily life.

By engaging in this mock interview, I have grown as a developer, and hopefully, this will allow me to get the upper hand in the fiercely competitive job-hunting game. By being well-prepared, I increase my chances of successfully impressing the interviewer, thus increasing my chances of getting hired.

Success Criteria	This story meets the marked criteria.
Learning New Technologies	No
Coding Best Practices	Yes
Version Control and Code Management	No
Better Understanding of Software Development Lifecycle	No
Agile Development	No
Employability	Yes

3.2.8. Incremental Wireframe Fidelity

Although incremental fidelity in design is not exclusive to Agile methodologies, it can be aligned with Agile principles and practices. This is particularly evident in iterative development processes like Scrum, which this project followed.

Creating the wireframes for all user stories by hand became one of the most tedious tasks of the entire project. This tediousness became more apparent when I had to make medium-fidelity wireframes in Figma based on the previous sketches. However, I did not know how to use Figma, and now the design had to follow good design practices.

Learning Figma was relatively straightforward. To simplify the task, I used Figma's component-based system. This allowed me to create all reusable components and their variations just once. Once created, I could duplicate the parent component and place its children on any wireframe. Any changes to the parent component would reflect in all child components, thus saving time during design revisions.

Conducting research throughout the design process was essential for learning good design practices. This involved thoroughly investigating the principles and techniques that contribute to effective design. It was also crucial to consider the needs of the project's client.

One technique I used was maintaining visual consistency in all the wireframes. This involved keeping the layouts, colour scheme, and font usage consistent. I also identified the most essential content users will interact with and prioritised these elements in the wireframe by putting the most important content at the top of the page. This ensured critical features were prominently displayed and easily accessible to the users.

To make navigating between wireframes as easy as possible, I focused on reducing the number of clicks necessary to complete a task. While the "three-click rule" is a standard guideline in design, I realised that it couldn't always be applied to every situation. Some tasks are naturally more complex and require more interactions. Nonetheless, I kept the three-click rule in mind throughout the design process to ensure that each task could be completed with as few clicks as possible.

Overall, I found the design phase quite bittersweet. I invested a lot of effort into creating visually appealing and effective wireframes. However, I realised I could not fully implement all the designs into the final prototype due to time constraints.

My career goal is to become a software engineer, not a UX designer. However, going through the design phase will help me become a better professional overall. A software engineer must understand the responsibilities of each team member to ensure better teamwork.

Success Criteria	This story meets the marked criteria.
Learning New Technologies	Yes
Coding Best Practices	No
Version Control and Code Management	No
Better Understanding of Software Development Lifecycle	Yes
Agile Development	Yes
Employability	Yes

3.2.9. Learning Express

I developed a web application with an interactive user experience for my final year project. Although I had prior experience building such websites using PHP and JavaScript, I wanted to expand my knowledge and expertise. To that end, I used the Express JavaScript framework to make my backend.

Express posed a significant challenge as my prior knowledge of using this framework was non-existent. This hindered the development process as I had to familiarise myself with Express's functionalities. Consequently, early backend sprints had fewer user story implementations; however, they took the most time due to endless errors resulting from logical misunderstandings.

One of the first things I had to get acquainted with was how to define different endpoint syntaxes. These determine how the REACT front end could interact with my backend Express server.

Endpoint definitions are personal preferences and can be done however a developer wants. However, understanding said endpoint would be difficult if future developers were to be added to the team later. I defined my endpoints according to best practices, ensuring that most of the team's future developers could understand them at a glance.

After setting up the first endpoint, I tested the most basic version of my backend. I encountered an error that brought my attention to a previously unknown concept. When attempting to fetch objects from my web application, I received a CORS (Cross-Origin Resource Sharing) error.

To address my lack of understanding, I searched for relevant online documentation to identify the gap in my knowledge and potentially find a solution. This step often results in moments of personal triumph and exhilaration as I uncover the solution to my problem buried within a text cluster. The feeling of finding it is satisfying. However, after conducting initial research, if I still couldn't find a solution, I would follow a tutorial until I comprehended where I had gone wrong and, more importantly, why my code was incorrect.

I discovered that CORS is responsible for defining how my user-facing REACT web application communicates with my resource-acquiring back-end Express. The error I encountered was a simple fix involving the setup of a middleware. However, I had to research to figure out how to implement it. The entire process was time-consuming, requiring me to learn many new things.

Learning Express was arduous, marked by moments of distress and vexation when nothing worked, and the console was a sea of red text. This was particularly the case for me, as the project's success hinged on my ability to familiarise myself with the new framework promptly. It required patience, perseverance, and a willingness to overcome obstacles. Despite being challenging, those moments have offered the most learning opportunities. I had to develop an effective debugging process, which I now consider a valuable skill. It has enabled me to analyse various sources more efficiently to formulate solutions to complex problems.

I gained valuable experience with modern web technologies, which deepened my understanding of web development. This experience has undoubtedly improved my qualifications and will help impress prospective employers.

Success Criteria	**This story meets the marked criteria.**
Learning New Technologies	Yes
Coding Best Practices	No
Version Control and Code Management	No
Better Understanding of Software Development Lifecycle	No
Agile Development	No
Employability	Yes

3.2.10. Learning REACT

I developed a web application with an interactive user experience for my final year project. Although I had prior experience building such websites using PHP and JavaScript, I wanted to expand my knowledge and expertise. To that end, I used the REACT JavaScript library to build the application’s front end.

Employing REACT posed a significant challenge as my prior knowledge of using this library was non-existent. I had to rethink how much of the project I could realistically finish as I had to consider this lack of familiarisation when planning implementation sprint tasks. This hindered the development process as a considerable amount of time was required to familiarise myself with its strange characteristics. Notably, REACT's unique component structure necessitated a substantial shift in my approach to web development. Consequently, early implementation sprints had fewer user stories; however, they took the most time due to endless errors resulting from logical misunderstandings.

To overcome my lack of understanding, I just had to use REACT. I believe the best way to learn is by making and fixing mistakes. Although it sounds simple, it is the only practical approach that works for me. However, I had to follow strict guidelines when solving an error for this approach to work. Whenever I encountered an issue, I searched for relevant online documentation to identify my knowledge gap and potentially find a solution. Usually, the problem arose because of a wrong assumption of how something behaved or because I misused a hook. After conducting initial research, if I still couldn't understand how to implement a concept, I would follow a tutorial until I comprehended where I had gone wrong and, more importantly, why my code was incorrect. Furthermore, I learned some REACT best practices along the way, such as using hooks and setting up components.

Learning REACT, like any skill, was a formidable undertaking. The process was arduous, marked by moments of distress and vexation when nothing worked, and the console was a sea of red text. This was particularly the case for me, as the project's success hinged on my ability to familiarise myself with the new library promptly. It required patience, perseverance, and a willingness to overcome obstacles. Despite being challenging, those moments offered the most learning opportunities. I had to develop an effective debugging process, which I now consider a valuable skill. It has enabled me to analyse various sources more efficiently to formulate solutions to complex problems. The work paid off with moments of triumph as errors slowly vanished and everything started making sense, especially the component nature of REACT.

Using the REACT library allowed me to create a dynamic and responsive user interface. I developed a deeper understanding of web development and gained valuable experience working with modern web technologies.

Success Criteria	**This story meets the marked criteria.**
Learning New Technologies	Yes
Coding Best Practices	Yes
Version Control and Code Management	No
Better Understanding of Software Development Lifecycle	No
Agile Development	No
Employability	Yes

3.2.11. Requirement Sprint Planning

My final-year project has a long list of requirements; however, despite its intended purpose of simulating real-world developmental lifecycles, the final-year project is subject to a significantly shorter timeline for completion. Therefore, it is imperative that I efficiently manage my time to ensure the successful delivery of the project.

During the requirements phase of my project, I verified the "Must-Have" requirements. However, there were still too many requirements to implement due to a lack of time. I need to prioritise the requirements and implement unique functionality that can be showcased during my Viva presentation. Otherwise, the prototype will only have basic functionalities like creation, reading, updating, and deleting, which may not be unique but were necessary to implement first.

During the initial stages of the project, I had limited knowledge of REACT and Express JS. Therefore, I could only incorporate a single user story per sprint, which would have still taken significant time due to my unfamiliarity with the frameworks. However, as the project progressed, I became more comfortable with REACT and Express, and I could implement multiple user stories in a single sprint. Additionally, I now had a list of pre-made reusable components, which made the frontend implementation process faster and more efficient.

I am saddened that the project could not reach its full potential as I had envisioned. However, I am grateful for the learning experience of making critical decisions regarding requirement implementation. This experience has improved my ability to make tough decisions when prioritising requirements, based not only on client feedback but also on the need to deliver a presentable project by the end of the term. The skill of critical evaluation is a valuable tool to have for my future professional career and is sought after by many employers.

Success Criteria	**This story meets the marked criteria.**
Learning New Technologies	No
Coding Best Practices	No
Version Control and Code Management	No
Better Understanding of Software Development Lifecycle	Yes
Agile Development	Yes
Employability	Yes

3.2.12. Software Development Lifecycle

This project used agile methodologies throughout the development cycle, with an iterative approach to complete each part of the software development cycle. As its name suggests, the agile method is fast-paced, allowing the project to respond rapidly to ever-evolving requirements—however, this agile response and the iterative approach cause some problems.

Due to a lack of time, I started coding the prototype before completing the planning. Instead, the project’s documentation was done in parallel with the implemented prototype. This led to instances where the prototype and the documentation became out of sync, potentially resulting in the prototype implementing requirements not stated in the documentation or the requirements changing drastically because of being implemented.

Inconsistency between documentation and the prototype can create problems within the team, particularly when new developers join. It can be difficult for them to understand the project without explicitly referencing how and why specific requirements were implemented. This situation can be compared to building furniture without instructions. While it may be possible to complete the task, it will likely take significantly longer and be more challenging.

Each document was reviewed a couple of weeks before final submissions to address this issue. It was compared with the prototype, and any updates were incorporated into the document. This usually involved updating diagrams and user requirement specifications.

Reviewing each document made me reflect on my writing and allowed me to assess the quality of my previous work. This exercise helped me update my documents and notice how much I have refined my formal writing style over time.

This project allowed me to use agile methodologies for the first time while going through the software development lifecycle. This experience has given me a better understanding of agile's advantages and pitfalls, which I believe will be helpful for me in my future professional career.

Success Criteria	This story meets the marked criteria.
Learning New Technologies	No
Coding Best Practices	No
Version Control and Code Management	No
Better Understanding of Software Development Lifecycle	Yes
Agile Development	Yes
Employability	Yes

3.2.13. SPAs, MVC, RESTful APIs

My final year project is a REACT single-page application supported by an Express-based RESTful API. The project follows a Model-View-Controller (MVC) implementation.

I have experience developing an MCV website using JavaScript and PHP and implementing an MVC program using Java. However, I did not know how to implement the MVC structure in REACT, nor was I familiar with terms such as Spa, single-page application, or RESTful APIs.

To help set up my React environment, I am using Vite. Vite is a local development server that allows me to view my REACT app in a browser as I develop it. To familiarise myself with how the MVC model is implemented within REACT, I spent some time analysing the template website provided by Vite. This template website was very simple, which allowed me to appreciate how the REACT app is structured.

Furthermore, the structure of the React app clarified how my project will be a single-page application, as only one index.html file is present for the entire application. This HTML file references a JSX file containing the JavaScript that will build the dynamic HTML. From this simple observation of the file structure of REACT, I gained an additional understanding of what a SPA is. To progress my knowledge, I did some further research and found that a single-page application is an app that doesn't need to reload the page during its use and works within a browser. I also discovered that Facebook, Gmail, and Twitter are all SPAs.

I researched RESTful APIs and discovered that an API or Application Programming Interface enables an application or service to access a resource within another application or service. Meanwhile, REST or Representational State Transfer imposes a set of guidelines or conditions on how an API should work. APIs which follow the REST architecture style are called RESTful APIs. The REST architecture style includes principles such as statelessness and uniform interfaces. I am developing a RESTful API using the Express.js framework for this project. This framework enables the creation of a backend that communicates with a MySQL database. The API supports standard functions such as GET, POST, PUT, and DELETE to retrieve resources from the database.

This experience has improved my understanding of web terminologies such as the MVC model and single-page applications, which will be valuable for future employment opportunities. Furthermore, I have improved my research skills.

Success Criteria	This story meets the marked criteria.

Learning New Technologies	Yes
Coding Best Practices	No
Version Control and Code Management	No
Better Understanding of Software Development Lifecycle	No
Agile Development	No
Employability	Yes

3.2.14. Time Management

Every week was meticulously planned, each with about four tasks that needed to be completed. These deliverables ranged from documentation artefacts to code implementation sprints. Essentially, these generated resources would slowly build up to what was necessary for the project's final submission.

Each deliverable was crucial; falling behind on one week and missing a deliverable meant it had to be rescheduled to the following week at the cost of working more hours to catch up. As a result of having more tasks to do in a week, another task may fail to get done and be rescheduled to the next, and so on, and the cycle of pushing back tasks begins. The main problem arose because of the fear of getting caught in this endless domino effect.

Although I understand that this document only represents my final-year project and does not reflect my life, I find it challenging to separate the two due to the significant amount of time I have devoted to this project. As a result, my fear of falling behind and my desire for perfection ironically led me to make some unnecessary and foolish decisions.

Working late every night and thinking about my project from the moment I got out of bed was starting to get to me, and sacrificing every aspect of my life to meet deadlines created an imbalance that quickly burned me out. These decisions compromised the overall quality of my work and harmed my health.

Thankfully, this is not the first time I have dealt with burning out and having excessive self-expectations, so I could identify the signs early on and knew that I had to change something. Over Christmas, I had a lot of time to think introspectively about why I let this project consume me, and I made a list of necessary actions I needed to take to get me through to the end of my degree.

The list was simple; it is a list that most would call obvious and just the basics of living: getting seven hours of sleep, eating regularly, staying hydrated, and exercising regularly. The crucial aspect of this list was that I decided that everything would take little priority over it. If I must postpone a task, so be it; if I must hand it in late, so be it.

Since adhering to the list, I have significantly improved in all aspects of my life. Interestingly, I have achieved increased productivity by avoiding excessive focus on work. I can concentrate more effectively on the task instead of having a persistent sense of fatigue and exhaustion.

In the context of my professional career, it is essential to know how I should manage my time correctly. I aim to submit project tasks by their respective deadlines. However, I will only do so at the expense of the four basic instructions I listed previously because, in the end, if I'm healthy, everything will get done, regardless of any deadlines.

Success Criteria	This story meets the marked criteria.
Learning New Technologies	No
Coding Best Practices	No
Version Control and Code Management	No
Better Understanding of Software Development Lifecycle	No
Agile Development	Yes
Employability	Yes

3.2.15. Version Control

I used GitHub throughout the project development to help manage my code base and ensure it was always safely saved and accessible from any machine if necessary. GitHub also allowed me to keep a version history containing every committed change I made throughout the project's life. The ability to revert to a historical version of my code came in handy on the day I needed to present my prototype.

As part of the final year project, I was asked to present the current state of my prototype to a handful of individuals. However, this presentation was not my only task for that given week. I also had a UI implementation sprint. This meant that while I needed to have a stable build to showcase my prototype, I also needed to work on adding features, which would undoubtedly make the application vulnerable to sudden failures.

In this situation, creating a separate development branch would be necessary to ensure that the main branch remained stable and ready to be presented. This particular branch can then start the implementation sprint without delay. Unfortunately, I forgot to create this crucial branch and only realised this after I had made several days' worth of commits to the main branch. To make matters worse, this realisation occurred on the day of the presentation.

I was frantically searching for a solution when Google came to my rescue. At that time, I was unaware that I could revert to previous commits and believed that users could only revert to previous branch merges. However, to my delight, I discovered that it was possible and straightforward to do so. I was able to restore my code to a stable version successfully. Additionally, I took my progress on the new UI implementation and created a separate branch for it, which I should have done from the beginning.

I cannot express enough gratitude to the creators of Git; I quite literally owe them about four per cent of my final year project for this situation alone. While short-lived, I felt tremendous anxiety when going through my prototype before the presentation and seeing an onslaught of errors. This experience taught me the importance of using version control, and I learned that I should never forget to make a branch again.

This endeavour has made me more familiar with the Git version control system, a vital skill for aspiring developers like me. Most employers highly value a working knowledge of Git.

Success Criteria	This story meets the marked criteria.

Learning New Technologies	Yes
Coding Best Practices	No
Version Control and Code Management	Yes
Better Understanding of Software Development Lifecycle	No
Agile Development	No
Employability	Yes

3.3. Evaluating Project Success

This section will evaluate the projects' success based on previously stated success criteria and the stories outlined above. Each success criterion will be evaluated separately.

Learning New Technologies

As evidenced by the stories 3.2.1, 3.2.4, 3.2.5, 3.2.9, and 3.2.10, the opportunity to learn new technologies was successfully taken and applied throughout the project. The REACT framework was learned to a level necessary for the project’s progression. Furthermore, the Express framework was also understood to an acceptable level. The Express framework received the most attention regarding refactoring sprint focus.

Acquiring knowledge of the latest technologies required considerable patience and effort. The only way to obtain such knowledge was to employ them, encounter errors, and try to resolve them. There were no shortcuts to this personal learning process; only the code and I were present. By experimenting with REACT concepts and fixing the inevitable errors, I began to understand the framework. This understanding has helped me in other areas of my studies, particularly with my “Mobile App Development” module, where I currently use REACT Native. While the two slightly differ in syntax, the concepts of hooks, state, and reusable components are the same, and thus, the knowledge I gained through this final-year project is transferable.

Overall, acquiring new coding knowledge and familiarising new technologies has become one of the project's most successful aspects.

Coding Best Practices

This criterion focused on ensuring that I learnt the new technologies to a standard that ensured I could apply them in a universally accepted way. Doing so made my code more readable, and the project became more manageable as it scaled up throughout the year. The learning of “coding best practices” is evidenced by the stories 3.2.1, 3.2.4, 3.2.5, 3.2.7 and 3.2.10. As showcased by these stories, I learnt best practices in many ways.

At the beginning of the project, I did not follow best practices, as I was primarily focused on getting something to work first. This led to stories such as 3.2.1 and 3.2.10 unfolding. These two stories focus on how I learnt best through refactoring and solving errors. Refactoring became a time when I primarily focused on making my code follow best practices, as it usually involved decreasing file sizes and reducing repeated code.

Whenever I faced a challenge implementing certain functionalities and the documentation didn't help, I found watching video tutorials helpful. These tutorials incidentally also demonstrated best practices. Following the tutorial steps enabled me to learn and implement best practices.

Furthermore, simply researching coding best practices proved extremely useful, as evidenced by story 3.2.5. Having a list laid out to follow made it easy to understand the best practices for a specific technology, such as REACT. Applying them, however, can be time-consuming, especially if it involves refactoring existing code or file structures. Due to this, the project does not fully implement best practices throughout and instead has sporadic examples of them.

Version Control and Code Management

This criterion was one of the first areas I had to set up to ensure my project had a stable foundation to grow. As evidenced by story 3.2.3, Git and GitHub were set up immediately to ensure the project had a version control system before any development. This was important to ensure I could capture the project's history from the beginning. Furthermore, by setting up the system, I ensure that I can always revert my code to a previous, possibly more stable version.

Better Understanding of Software Development Lifecycle

One of the most significant shortcomings of this project was the lack of frequent communication with the client. Although the client was involved in critical stages of development, such as requirements gathering and usability testing, I failed to provide regular updates on the project's progress every month. As evidenced by story 3.2.2. However, this is only one aspect of the development lifecycle.

This project has given me a comprehensive understanding of each life cycle stage. From the initial planning and requirement-gathering phase to the design, implementation, and usability testing phases, I gained valuable insights into the project management process. This experience has refined my project planning, design thinking, and quality assurance skills. This has enabled me to develop a more holistic approach to project management, which I believe will be highly valuable in my professional career. This can be evidenced by stories such as 3.2.11, 3.2.12, and 3.2.14.

Agile Development

The project utilised a combination of agile and Scrum methodologies. Scrum was used for its time-limited iterations called sprints and weekly meetings. Every week, a sprint consisting of four deliverables had to be completed. The agile aspect of the methodology comes from the fact that different phases of the project were run in parallel. Having to follow agile methods for a year has undoubtedly made me more understanding of how it functions.

As evidenced by story 3.2.14, I struggled with time management because my weeks were filled with consistently time-consuming deliverables. I could not let this derail the entire project, so I had to improve project management to cope. I believe that this criterion was successfully carried out, as documentation has evidenced each sprint, and the project was able to deliver artefacts covering the entire marking scheme.

Employability

Finally, the potential increase in employability that this project has provided will be evaluated. This section looks at the sought-after skills that I have acquired. Increasing employability is an essential aspect of this project, especially when considering a future in the industry as a software engineer.

Every success criterion mentioned above was a learning experience. The project necessitated the learning of time management, which has made me more effective when completing a set of tasks, as evidenced in story 3.2.14. I had to learn new technologies to successfully implement the idea, as evidenced by stories 3.2.1, 3.2.4, 3.2.5, 3.2.9, and 3.2.10. I was put through and experienced the software development lifecycle, which has taught me a lot about the importance of each phase, evidenced by stories 3.2.11, 3.2.12, and 3.2.14. Furthermore, the project has taught me about version control (story 3.2.3), coding best practices (stories 3.2.1, 3.2.4, 3.2.5, 3.2.7 and 3.2.10) and how to deal with clients(story 3.2.2) during development. All of this has been a learning experience that has given me valuable skills I can use in the future.

4. Conclusion

While I am proud that I have managed to finish the largest project of my life and have undoubtedly never written this many documents, at the end of it all, the project was nothing more than just a learning experience. If I think about it like that, instead of a measure of what I could implement or not, this project was an immense success. It has changed my outlook on the software engineering career pathway and the software development industry. The project has given me a reality check concerning my immature belief that coding can be done without documentation. While I still do not like writing them, I now know how valuable they can be, as I had to reference them continuously throughout the project. I had forgotten certain aspects and requirements during the year. Furthermore, these technical documents have allowed me to practice my professional writing skills, which will undoubtedly be helpful in any professional career.

The project has changed how I will approach programming in the future. I made many mistakes that I would never like to repeat if I could avoid them. I cannot help but feel like I let myself down a bit when it came to the final code I was able to produce. While I had to learn to REACT and Express along the way, I still feel I did not do enough. Annoyingly, I became conservative with the time I spent coding. I needed to prioritise the more time-consuming documentation if I wanted to have any chance of getting a weekly sprint completed. Furthermore, I found myself not doing more than was necessary, which allowed the project to keep on track, but it often meant stopping in the middle of a coding flow after realising I was going off track with what I needed to do for that week.

All in all, this project has been a valuable learning experience. It has equipped me with the skills or at least the thoughts necessary to have a better chance of having a successful future professional career. The artefacts produced will be valuable in showcasing and evidencing this learning journey.