



PRJ400 REPORT

David Fox (S00172018)

GitHub Repo: <https://github.com/S00172018/prj400>



APRIL 13, 2020

Contents

Chapter 1 Overview	4
Chapter 2 Goal and Scope.....	5
2.1 Goal.....	5
2.2 Scope.....	5
2.3 User Stories	5
Chapter 3 Areas of Research	7
Introduction	7
Existing Applications	7
Microsoft To Do	7
Todoist	8
Asana	9
OmniFocus	9
Observations.....	10
Notable Features.....	10
Potential Gaps.....	10
Progressive and Single Page Apps	11
Progressive Web Application (PWA)	11
Advantages of Progressive Web Applications (PWAs).....	11
Single Page Application (SPA).....	11
Progressive Single Page Application (PSPA)	11
Frontend Framework	12
Introduction	12
ReactJS	12
Vue.js	12
Angular	13
Comparison.....	13
Conclusion.....	14
Development Environment.....	14
Introduction	14



Visual Studio	14
Visual Studio Code	15
Eclipse	15
Comparison	15
Conclusion	16
Backend Framework	16
Firebase	16
Express	16
Laravel	17
Comparison	17
Conclusion	17
Database	17
MySQL	17
MongoDB	18
Microsoft SQL Server	18
Comparison	18
Conclusion	19
Testing Frameworks	19
Testing with Vue	19
Jest	19
Mocha	19
Comparison	19
Conclusion	19
Chapter 4 Design and User Interface	20
4.1 First Draft Wireframes	20
4.2 Application Architecture	22
Front End Technologies	22
Back End Technologies	22
4.3 Database Structure	23
Chapter 5 Time Management	24
5.1 Introduction	24
5.2 Semester 1	24



5.3 Semester 2	24
Chapter 6 Problems / Challenges Encountered	26
Dropped Project.....	26
Hardware Difficulties	26
Reliance on Virtual Machines.....	26
No Access to Resources.....	26
API Documentation.....	26
New Tools and Tech	26
Chapter 7 Learning and Outcomes.....	27
Final Application	27
New Technology	27
Applying Agile	27
What I Would Have Done Differently.....	28
Future Development	28
Conclusion	29
Bibliography.....	29



Chapter 1 Overview

The idea behind this project was to create a productivity application for task management. Unlike a number of task management applications, this one sought to bring together the task and the time and is inspired by the Emergent Task Planner by David Seah¹. Users are given the ability to create an event list and assign their events to specific dates on a calendar. The application also allows for users to interact with the tasks on the calendar via drag and drop functionality, assigning tasks to different placements and scaling their duration.

For the purpose of learning, I decided to create the project using the Vue framework. I was drawn to Vue as it is a relatively new framework for building web applications and one that has been growing in popularity for a number of years. It has also been praised by many developers for its flexibility, compartmentalisation and extensive documentation.

User and event data is stored in a Cloud Firestore database. Creating an event triggers a method that adds a document with the data to a task-list collection in the database, tied to the user's individual account. Interacting with an event on the calendar also causes the database to be updated. Firebase was implemented into the application for both authentication and hosting.

Over the course of development on the project, I applied methodologies that I have picked up during my work placement, namely SCRUM. Work on the project was carried out in biweekly sprints, although this was not always be possible due to commitments to other modules and assignments. I participated in meetings with my supervisor every week to give an update on my progress.

I also employed a number of project management tools. Trello was used in order to manage and keep track of my sprints and user stories, as well as demonstrate my progress to my supervisor between meetings. Updates on the project were pushed to a GitHub repository and I posted weekly updates to a WordPress blog in order to keep a record of the work that was carried out.

¹ <https://davidseah.com/node/the-emergent-task-planner/>



Chapter 2 Goal and Scope

2.1 Goal

The overall goal for this project is to build a task management application that bridges the gap between a to do list and a calendar. The end result should allow the user to easily add and remove tasks to and from a to do list and provide them with a clear visual representation of their tasks on a calendar view, so they not only know what they have to do, but also when to do it.

In terms of personal goals, I was determined to develop the application using a relatively new technology with which I have had no prior experience, hence why I have settled on the Vue.JS framework. Over the course of development on the application, I aim to gain a solid foundation in this framework, so that I may apply what I have learned to potential future projects.

I also hope to re-enforce my understanding of several of the underlying principles of web development that I have learned and experimented with over the course of my studies, as well as the Agile principals I have been introduced to as a part of my third year project and work experience.

This project is also significant in that it will mark the first occasion that I have developed a full-stack application on my own while following the Agile methodology.

2.2 Scope

Development on the project will follow the Agile methodology. This involves devising a backlog of user stories and assigning them to sprints of a typically varying length. Prior to full development commencing on the application, I will devise a number of user stories with the aid of the Trello web application. This will be in order to break down the development of the app into smaller chunks so that I will be able to carry out effective one-to-two-week sprints. These stories will then be assessed and prioritized in accordance with their importance using the MoSCoW prioritization technique, consisting of Must-have, Should-have, Could-have and Won't-have.

2.3 User Stories

The user stories devised for the development of the project are as follows:

- As a user I want to create a task so I can add to my task list.
- As a user I want to see a daily calendar so I can decide when to complete my tasks.
- As a user I want to assign tasks to dates on a calendar so I can keep track of my tasks.
- As a user I want to assign a start time to a task. So that I know what time that it is due to begin.
- As a user I want to assign an end time to a task, so that I know when it is due to be finished.
- As a user I want my data to persist to a database so I can save my data.
- As a user I want to delete a task so I can remove it from my list.
- As a user I want to edit a task so I can update its contents.
- As a user I want to assign a priority to a task so I can see my prioritized tasks more easily.
- As a user I want to log in so I can see my tasks.
- As a user I want to write a description for my tasks so that I can further expand upon it.



- As a user I want to see attractive styling so that using the application is easy, legible and consistent, allowing me to see and interact with my tasks more clearly.
- As a user I want a responsive web page so that I can use the application on my mobile devices.



Chapter 3 Areas of Research

Introduction

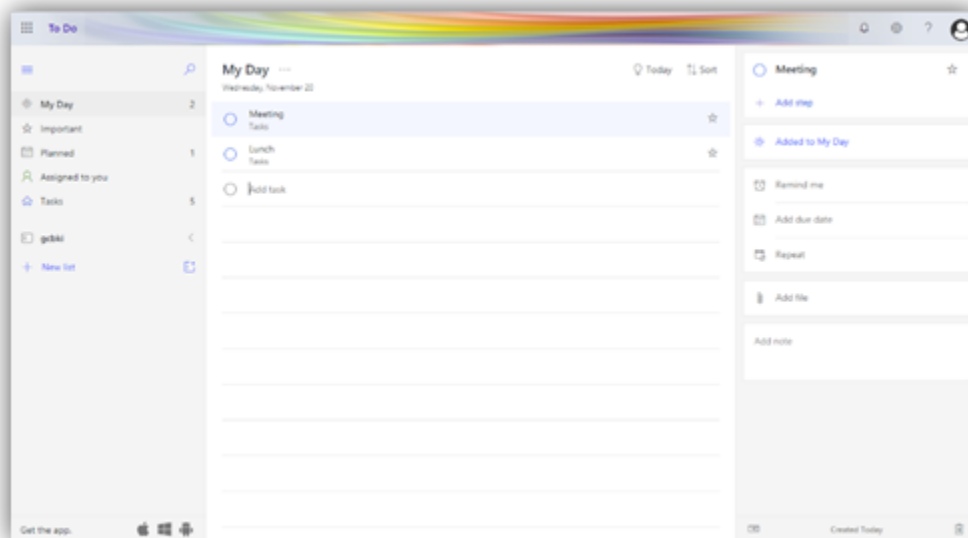
For my PRJ400 I plan to develop a productivity web application. It is in the early stages of development, but at this point I plan for it to consist of a task management component and a weekly calendar on the main page which allows the user to drag and drop specific tasks they have created on to the different sections of the calendar. The purpose of this chapter was to inform the development of my application through research of various technologies, articles, and applications. In the first section, researched some of the most popular application in order to provide me with a reference point for my app. In the following section, I researched the differences between progressive and single-page web applications. Finally, I compare various technologies such as frameworks and databases in order to select what I felt were the most suitable for me and my application.

Existing Applications

From my research, I have found that there are many existing productivity / task management applications on the market. This will simultaneously aid me in my research and present a challenge as I seek to learn from what has been attempted in the past, while also attempting to distinguish my application from what has already been realised.

For this section, my aim is to identify the essential components of popular productivity / task management applications while also looking for gaps that haven't been filled. I will be examining and comparing some of the most popular task management applications available in order to learn from them and build my knowledge base from what is expected and what is missing that I can build upon. I am hoping this will inform my decisions on features and their implementation.

Microsoft To Do

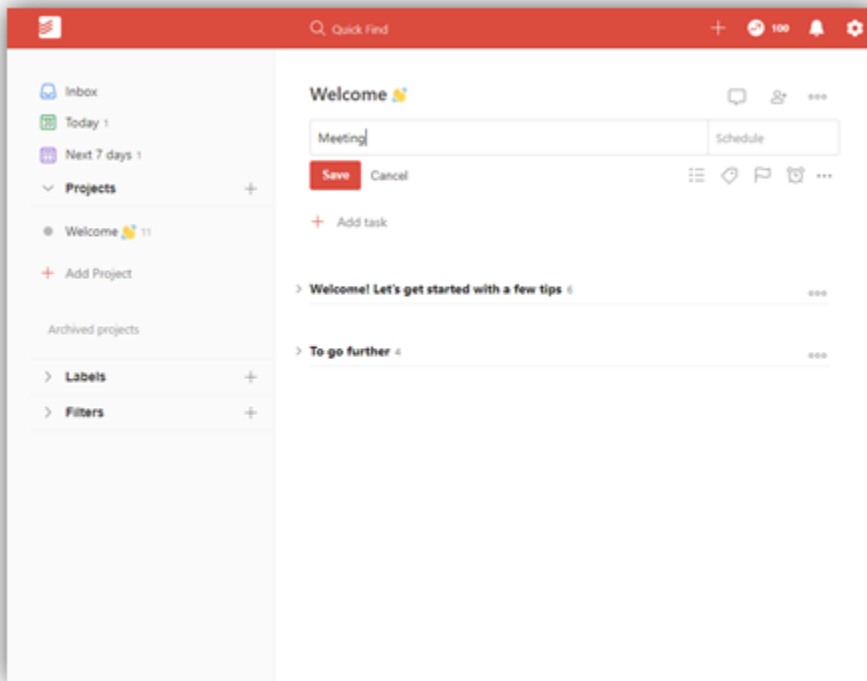


Microsoft To Do is one of the newest task management applications to hit the market. It was initially released as a preview in 2017 and was relaunched in 2019 with a redesigned UI and a new set of



features. The application was also integrated into other Microsoft applications such as Outlook. It was developed with the intention of being a successor to Wunderlist, which Microsoft acquired in 2015. From my own experience using the application, Microsoft To Do does not offer as much depth as some of the other applications I have looked into, but its clean, simple interface and integration with Outlook make it very appealing.

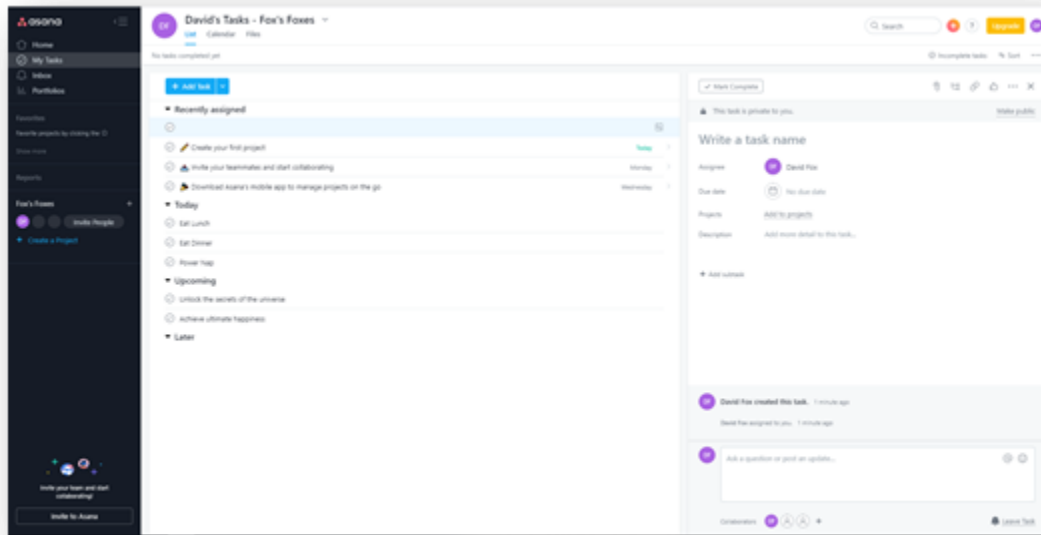
Todoist



Todoist is a flexible task management application that can be easily used by either individuals or teams. It is the biggest competitor of Microsoft To Do. It was developed by Doist Ltd. and was originally released in 2007. A distinguishing feature of Todoist that I really admired was the gamification of task completion via a points system. Collecting points allows the user to unlock in-app badges, promoting further productivity.

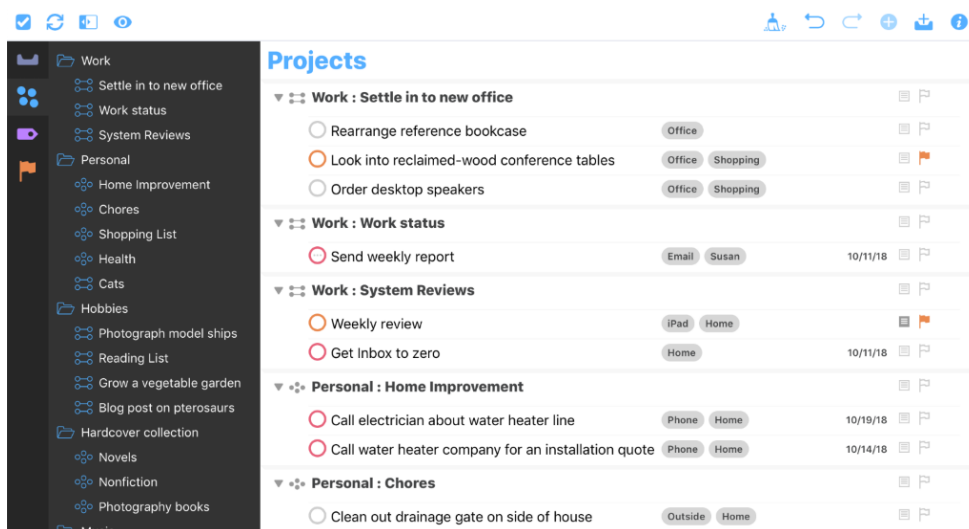


Asana



Asana is a team project-oriented task management application. It developed by Facebook co-founder Dustin Moskovitz and ex-engineer at both Facebook and Google Justin Rosenstein. The application was launched for free in 2011. A feature of Asana which I appreciated most of all was the ability to create team projects and assign members to different tasks. This was similar to a software I used during my work experience, albeit more user friendly. Unfortunately, many of Asana's features were blocked out for me as I was using the basic free package.

OmniFocus



OmniFocus is a personal task management application designed for MacOS and IOS. A version was later developed for web; however, it was designed as a companion app and requires the former in order to



take advantage of it. It released in 2008 and was developed by The Omni Group. I liked the depth of organisation provided by this application in comparison to the others I have researched, however, I was I found the sheer quantity of text on screen a little overwhelming and difficult to follow at times.

Observations

After scouting out the various productivity and task management applications available on the market, I have taken note of some recurring features / components.

Notable Features

- **Task List:** The most fundamental component of a productivity / task management application. All applications I have researched have implemented this feature as the basis.
- **Layout:** Most of the applications I have researched are single page apps with extremely similar layouts.
- **Favourite button:** Some applications have implemented a favourite button which allows the user to save / highlight specific tasks.
- **Points System:** Todoist includes a points-based system that allows users to earn points and unlock badges by completing tasks.
- **Organise:** Some of the applications I have researched allow the user to organize their tasks into different folders.
- **Team:** Some applications have a feature that allow for users create teams and assign people to specific tasks.

Potential Gaps

- **Interactive calendar:** For my application, I intend the main feature to be the integration of task manager and calendar. From my observations, few of the existing applications I have looked at have implemented this.
- **Colour-coding:** For my application, I intend to have an emphasis on colour-coding. This will vary depending on how the task's priority. From what I have researched, some applications distinguish certain tasks from others using more subtle indicators.
- **Time-boxed Tasks:** Another potential feature I am looking into for my application is the ability for the user to time-box specific tasks. Research has indicated that setting limits on the time it will take to complete a task can subconsciously trick our minds to become more efficient. *"Time boxing doesn't guarantee that you'll finish the work in the allotted time. However, it can definitely help with focus. And deciding in advance on the time investment helps Parkinson's Law, that work expands to fill the time allotted for it, to work to your advantage."* (Saunders, 2019)



Progressive and Single Page Apps

Progressive Web Application (PWA)

A Progressive Web Application could be described as a fusion of web and mobile applications. They were originally proposed by Google in order to improve the quality of life regarding web applications, particularly for users with poor internet connections.

“Progressive web applications are the new standard in the modern era of web development.” (Ivanova & Georgiev, 2019)

Some of the notable characteristics that make up PWAs include the ability to operate the application offline and push notifications, allowing for a faster and more user-friendly experience for consumers. Examples of some well know progressive web applications include Spotify, Pinterest and Uber.

Advantages of Progressive Web Applications (PWAs)

- Faster performance due to how website content is cached
- Greater freedom – Does not need to be submitted to application store
- Less limited by internet connection – self-contained ala mobile applications
- Can be accessed via URLs or indexed by search engines
- Designed like mobile apps – Greater user experience

Single Page Application (SPA)

A Single Page Application (SPA) is an application which consist of only one page. When the user interacts with components and navigates to different sections of the app, instead of loading a separate page, the relevant components on the page is altered. Most of the applications I have developed have been SPAs.

Progressive Single Page Application (PSPA)

Progressive Web Applications and Single Page Applications are not mutually exclusive and features of both can be implemented into the same application.

It is too early at this point in development to know for certain, but my application is likely to be a SPA. If there is time towards the end of development, I will look at trying to implement PWA characteristics.



Frontend Framework

Introduction

There are a wide variety of potential web development frameworks available for development on the application. In this section, I will be looking at a selection of frameworks based on their feature sets, popularity and release dates as well as my own personal experience.

For the development of the project, I am keen to incorporate recent technologies and technologies with which I have had no prior experience. I am also aware that developing the application with an unfamiliar framework is likely to slow down development to a certain degree and deciding on a framework with which I am more acquainted with may also prove to be more beneficial.

Below I have narrowed down my options to three web development frameworks, two of which I have worked with on past projects. I will give an overview of each of the frameworks in order to determine which is the best fit for the development of the application.

ReactJS

ReactJS is an open-source JavaScript library for building web interfaces and single page applications. It was developed by Facebook and initially released in 2013. Having developed in React for my Third-Year group project, it is my second choice of framework for the development of the application.

React uses virtual DOM as opposed to real DOM. The virtual DOM is an in-memory representation of the real DOM that seamlessly updates in response to alterations to the code therefore, increasing performance and allowing us to immediately see our changes take effect. This also makes it significantly faster to detect and locate bugs in the code.

React offers extensive documentation that is complete with solutions to many commonly encountered known issues users may encounter. As React is owned and maintained by Facebook, it has a strong backing compared to other frameworks such as Vue.js.

React is especially notable for having been used for the development of several popular social media applications such as Facebook and Twitter. With Analytics from both Stack Overflow and Google suggesting it as one of the most popular frameworks for web development, this makes a compelling case for the development of the project.

Vue.js

Vue.js is an open-source JavaScript library for developing single-page web applications and user interfaces. It is my first choice of framework as it is a relatively new technology that has gained a lot of traction since its release. It was initially released in 2014 by an ex-Google engineer, Evan You.

Vue.js has been described as the best of both worlds in relation to React and Angular. *"It has really great balance of features that come out of the box and It's incredibly easy to add more along the way like Routing and State Management."* (Vivek, 2018)

Like React, Vue also uses a virtual DOM which gives us an in-memory representation of the real DOM allowing for faster performance and increased efficiency as we can see our changes to the code immediately take effect as we interact with it.



As it was developed independently, Vue does not possess a strong backing from any major corporations such as Facebook or Google. Instead, it is maintained by developer Evan You and an assortment of volunteers within the Vue community.

Although Vue is not as strongly backed as other frameworks, it still possesses an extremely vibrant and still-growing community, as evidenced by its extensive documentation, online tutorials and demo projects.

I am drawn to Vue in particular as it is the youngest of the three frameworks that I have selected as well as the only one I have no experience with.

Angular

Angular is an open-source, TypeScript-based JavaScript framework. It was developed by Google and originally released in 2010. It was developed as a successor to AngularJS, though it differs significantly, and both can be considered entirely separate entities. Angular is widely considered to be one of the more complex frameworks with a steep learning curve. As Angular is owned and maintained by Google, it has a particularly strong backing compared to the majority of other frameworks.

“This framework of JavaScript uses real DOM. It is extremely hard to handle because if you lose the flow, you will have to go deep into the codes to actually find out the issue.” (Martin, 2019)

Angular includes features like dependency-injection, two-way data binding, RESTful API and AJAX handling. Angular 9 is set to be released in 2020.

According to analytics from Google and Stack Overflow, Angular is the least popular framework of the three by some distance. This could be attributed to the rate at which its documentation becomes outdated as Angular continues to update and evolve rapidly, leading to a certain amount of disruption experienced by some developers. This could also be linked to certain issues surrounding the launch of Angular 1 back in 2010.

Comparison

Vue	React	Angular
Developed independently by ex-Google engineer, Evan You, therefore it is not as strongly backed as React, but its popularity has resulted in a vibrant and growing community. Released in 2014.	React was developed and is maintained by Facebook, therefore has a strong backing, with a continuous stream of updates, improvements, features and bug fixes. It was released in 2013. Released in 2013.	Developed and maintained by Google, therefore has a strong backing, with a continuous stream of updates, improvements, features and bug fixes. Released in 2010.
Popular applications that were developed using Vue include TuneIn, 9Gag and GitLab.	Popular applications that were developed using React include Facebook, Twitter, Instagram, WhatsApp and Airbnb.	Popular applications that were developed using Angular include Netflix, Vevo, Lego, The Guardian and the console version of YouTube.



Analytics suggest that Vue is the 2 nd most popular of the three.	Analytics suggest that Vue is the most popular of the three.	Analytics suggest that Vue is the least popular of the three.
Virtual DOM. Gives us an in-memory representation of the real DOM allowing for faster performance and increased efficiency as we can see our changes to the code immediately take shape.	Virtual DOM. Gives us an in-memory representation of the real DOM allowing for faster performance and increased efficiency.	Angular uses real DOM which can make development a more time-consuming process and extra prone to attracting bugs. This also results in slower performance than either Vue or React.
I have had no prior experience developing with Vue. While this is likely to slow down development, I am also keen to work on something new.	I have had some experience developing a React application as part of my third-year group project.	I have had experience developing multiple Angular applications as part of my web development modules.

Conclusion

The frontend that I have decided to use for the project is Vue. While I have had varying degrees of experience with other programming languages such as Angular and React, for this project I have decided that I would attempt to implement as many unfamiliar technologies as I felt that I could manage.

The research that I have conducted shows that Vue both is fast and relatively easy to learn while encompassing some of the best of both Angular and React. I have also found myself drawn to its satisfying structure and well-defined architecture, as well as its thriving community and built-in features.

Development Environment

Introduction

As with the choice of web development frameworks, there is an abundance of viable options to consider when settling on a development environment. An Integrated Development Environment (IDE), provides the user with an interface for their code from which they can access a multitude of essential functions such as a code editor, compiler and more.

However, not every development environment is an IDE, as we can use a combination of a lightweight code editor and a vast array of plugins of our choice in order to carry out most of the same procedures and achieve the same results as we could do on a typical IDE such as Visual Studio.

Below I have narrowed down my choices to three options, two of which I have worked with before.

Visual Studio

Microsoft's Visual Studio is a C-based cross-platform IDE. Unlike Visual Studio Code, it includes features such as text auto-completion, database-integration and server configuration. It is mostly designed for users who are developing .NET applications, although it can also be used for other kinds of projects.

This is the primary IDE I have worked on throughout my studies and therefore is the development environment that I am most familiar with.



As Visual Studio is a full-blown IDE, its performance is not as light and efficient as that of Visual Studio Code. It is more suitable for developing more complex projects.

Visual Studio Code

Visual Studio Code is a lightweight source code editor developed by Microsoft. It was released in 2015 and it is one of the most popular code editors available. Unlike Visual Studio, it is not an IDE, but can be configured to execute a lot of the same features and functionality.

“Not quite an IDE (that’s actually [a separate product altogether](#)), VS Code can take on most of the tasks of an IDE with the right configuration and plugin library. The community for VS Code is incredibly passionate, and that works to everyone’s benefit.” (Keeton, 2019)

As Visual Studio Code is not an IDE, it offers faster performance and is more lightweight in comparison to the other options on this list, which means that it is more suitable for less complex projects.

Visual Studio Code has been the primary development environment I have used for web development modules and projects. It is also one of the most popular code editors for web development projects overall.

Eclipse

Eclipse is a Java-based cross-platform IDE developed and maintained by the Eclipse Foundation. It was initially released in 2001 and it is primarily used for the development of Java applications, but plugins can also be installed to develop applications in other programming languages.

A notable feature of Eclipse in comparison to the other development environments is perspectives. They allow the user to customize their views and editors as well as making the process of installing new languages and platforms smoother.

Unlike the other Visual Studio and Visual Studio Code, Eclipse is the only development environment on this list that I have not had any prior experience with. While I am keen to incorporate technologies with which I am unfamiliar with, as I have already decided on using Vue as my framework, this is no longer essential.

Again, performance on Eclipse is not as efficient on that of Visual Studio Code, although are not directly comparable as Eclipse is an IDE.

Comparison

Visual Studio	Visual Studio Code	Eclipse
Visual Studio is an IDE developed by Microsoft. It was initially released in 2015.	Visual Studio Code is a source code editor developed by Microsoft. It was initially released in 2015.	Eclipse is a Java-based IDE developed by The Eclipse Foundation. It was initially released in 2001.
Includes lots of built-in functionality including database integration, text auto-complete, and server configuration.	Much more lightweight as it is a text editor as opposed to an IDE, making it more suitable for less complex projects.	Similar feature set to Visual Studio. Also includes perspectives which allow the user to customize their views



		and editors and the process of installing new language and platforms is smoother.
Operates slower than Visual Studio Code, although performance is not truly comparable as that is not an IDE and doesn't offer the same depth of features.	It is fast and lightweight compared to Visual Studio and Eclipse, therefore it is better suited to smaller projects.	Operates slower than Visual Studio Code, although performance is not truly comparable as that is not an IDE and doesn't offer the same depth of features.
I have had lots of experience developing with the Visual Studio IDE.	I have had lots of experience developing applications with the Visual Studio Code editor.	I have had no prior experience developing with the Eclipse IDE.

Conclusion

I have decided to use Visual Code as it is faster and more lightweight than either Visual Studio or Eclipse and is better suited to the development of web applications. My research has also shown that it is the most widely used code editor regarding Vue.

Backend Framework

Firestore

Firestore is a web application development framework. It was founded in 2011 by Firestore, Inc., and later purchased by Google in 2014. Firestore offers cloud-hosted info, accessible via a simple API.

Firestore offers a vast array of easy-to-integrate services for authentication, databases, hosting and more. I have had some experience working with Firestore backend services for my third-year project and found it to be a reliable and satisfactory experience with a heavy emphasis on streamlining and ease of use.

Express

Express is a micro-framework for Node.js and the most widely used backend framework for applications developed with Vue. Like Vue, it uses JavaScript, therefore code can be shared between the server application and a Vue client.

"Express.js is by far the most popular choice for a Node-based framework. Developers like its minimalism, which make it easy to create with and really fast, and its flexibility, allowing you to choose your own database, ORM, authentication etc if you should need them." (Gore, 2018)

Express comes highly recommended for applications with an emphasis on the frontend that require only a simple server application and a simple API. Unlike Firestore, I have not used this backend framework before.



Laravel

Laravel is an open-source PHP framework. It was developed by Taylor Otwell as an alternative to the CodeIgniter framework and it was initially released in 2011. It is shipped with Vue.js by default. Laravel is one of the most popular test frameworks for Vue applications.

A key feature of the framework is its attempt to streamline mundane software development activities such as user authentication and routing in order to allow for a smoother development experience.

As with Express, I have had no previous experience with this framework and thus am unfamiliar with how it operates.

Comparison

Firestore	Express	Laravel
Firestore is a web application development framework founded in 2011 by Firebase, Inc., and then purchased in 2014 by Google.	Express is a micro-framework for Node.js. It was released in 2009.	Laravel is an open-source PHP framework. It was released in 2011.
BaaS that offers options for database management, authentication and hosting with minimal setup and configuration.	Flexible, minimalist and high-performing framework with a powerful feature set for developing web applications. Comes highly recommended for Vue.	Attempts to streamline mundane software activities such as authentication and routing, allowing for a smoother development experience.
I have had some experience developing with Firestore as part of my third-year group project.	I have had no prior experience using Express.	I have had no prior experience using Laravel.

Conclusion

I have decided to use Firestore as the backend framework for the project as my research points to it being the one of the most popular backend frameworks for developing with Vue. My familiarity with Firestore from using it with previous projects will also prove valuable to me as it will at least partially make up for some lost time in coming to grips with Vue.

Database

MySQL

MySQL is an open-source relational database management system based on C and C++ programming languages. It was originally developed by MySQL AB and released in 1995. As of 2010, MySQL is owned and maintained by Oracle Corporation.

MySQL is a relational database, which means it is table-based with the data being more conventionally structured. It also uses the structured query language (SQL) in order to access and interact with the



data. This would be an advantage to me as I have had lots of prior experience working with the SQL language.

MySQL offers high performance and scalability, although it is slower than MongoDB in certain aspects due to fundamental differences in the database structure and how data is queried.

MongoDB

MongoDB is a document-based database management system developed by MongoDB Inc. and initially released in 2009. It is non-relational which means that data is not stored in the conventional form of rows and columns. Instead, data is stored in JSON format, thus allowing for a flexible structure and more scalability than that of relational databases. MongoDB outperforms MySQL in certain contexts due to differences in the database structure.

I have had no prior experience working with MongoDB.

Microsoft SQL Server

Microsoft SQL Server is a relational database management system. As SQL Server is a Microsoft product, it is only compatible with Windows PCs. As SQL Server is backed by Microsoft, it can be noted as one of the most secure and trusted DMSs on this list and offers excellent all-round security features. Unlike the MySQL and MongoDB, it is not open-source and therefore must be paid for. SQL Server is also the database that I have had the most experience with.

Comparison

MySQL	MongoDB	Microsoft SQL Server
MySQL is a relational database. It was initially released in 1995 by MySQL AB and is currently maintained by Oracle Corporation.	MongoDB is a NoSQL database. It was developed by MongoDB Inc. and initially released in 2009.	SQL Server is a RDMS. It was developed by Microsoft and initially released in 1989.
Well known organisations that use MySQL databases include Netflix, YouTube, Twitter, Spotify and NASA.	Well known organisations that use MongoDB databases include Sony, Citrix, Twitter and SurveyMonkey.	Not as popular as MySQL or MongoDB. However, not directly comparable as it is not open source.
High performance and scalability. Slower than MongoDB in some contexts due to fundamental differences in the database structure and how data is queried.	High performance and scalability. Outperforms MySQL in certain contexts due to differences in database structure.	Is the clear leader in performance and power, but a higher level of skill is required to configure and maintain.
I have had some prior experience using MySQL databases.	I have had no prior experience using MongoDB.	I have had lots of experience with SQL Server.



Conclusion

I have decided to use a MySQL database for the project as I have had prior experience using the SQL language and I have had less experience with it than SQL Server. It is also a relational database which I think will be more suitable for the application as the data will always be consistent. It will also be a learning opportunity as I have not had a lot of experience using it.

Testing Frameworks

Testing with Vue

The official Vue documentation recommends the Jest or Mocha testing frameworks, both of which have built-in options with the Vue CLI that work out of the box.

Jest

Jest is a JavaScript testing framework developed and maintained by Facebook. Jest runs tests synchronously, allowing for extra speed and efficiency. My research has shown that it prioritizes ease of use over deep configuration.

Mocha

Mocha is a JavaScript test framework that runs on Node.js. It is developed and maintained by a group of volunteers and was released in 2011. Research indicates that it emphasizes configuration over ease of use, possibly pointing towards it being more suited to large commercial apps.

Comparison

Jest	Mocha
Jest is a JavaScript testing framework developed and maintained by Jest.	Has Mocha is a JavaScript testing framework developed and maintained by a team of volunteers.
Jest is fast and easy to set up and has shorter assertion syntax, making it appropriate for smaller apps.	Mocha is slower to set up than Jest and requires more configuration, pointing to it being better suited to large-scale apps.
Jest is a relatively newer framework compared to Mocha but has since become quite popular.	Mocha has been around longer than Jest, and thus offers more complete documentation.

Conclusion

I have decided to use Jest as my testing framework my research as shown that the Jest framework is better suited to projects that are smaller in scale, whereas Mocha is more suited to larger, business-oriented projects.

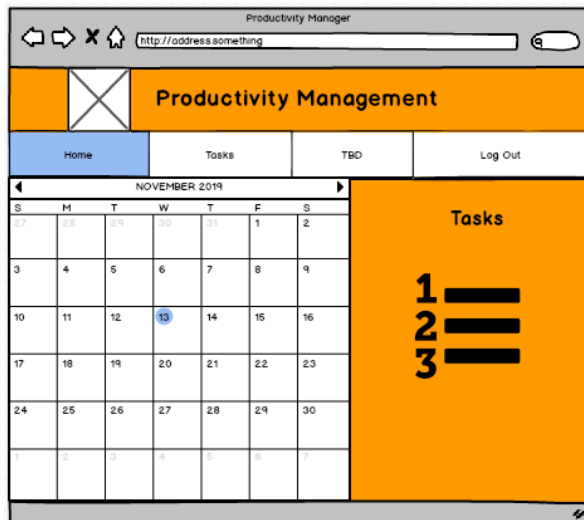
"If you have a large project with the need for flexibility and customization then Mocha is probably the choice for you. If you have a smaller project and don't need the extra setup and configuration up front, Jest is probably the better option." (Wheeler, 2018)



Chapter 4 Design and User Interface

4.1 First Draft Wireframes

Figure 1 Home



This screen depicts the main component of the application. The user is presented with the calendar API on one side of the screen, a list of the tasks they had created on the right side of the screen. From here, users will have the ability to click on a task and assign it to a particular date and time via drag and drop functionality.

Figure 2 Tasks



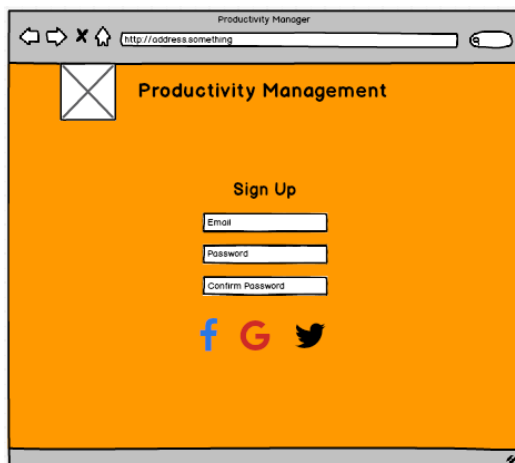
This screen depicts the second most important component of the application. From here, users are presented with a list of the tasks they have created and can freely add or remove tasks at their leisure. Task attributes include title, start time, end time, description, module and priority and can be sorted / filtered by such.



Figure 3 Login

The screenshot shows a web browser window titled "Productivity Manager" with the address bar displaying "http://address.something". The page has an orange background and a white "X" icon in the top left corner. The title "Productivity Management" is at the top. Below it, the word "Login" is centered. There are two input fields: "Username" and "Password". At the bottom, there are three social media icons: Facebook (f), Google (G), and Twitter (bird).

This screen depicts the login form. Users also have the option of logging in via Facebook, Twitter or Google. This will allow the user to access the main components as well as see and interact with their own tasks. Users will only be able to access to their own tasks.

Figure 4 Sign Up

The screenshot shows a web browser window titled "Productivity Manager" with the address bar displaying "http://address.something". The page has an orange background and a white "X" icon in the top left corner. The title "Productivity Management" is at the top. Below it, the word "Sign Up" is centered. There are three input fields: "Email", "Password", and "Confirm Password". At the bottom, there are three social media icons: Facebook (f), Google (G), and Twitter (bird).

This is the registration screen. Here users can sign up with their own email and create a password that meets the verification criteria. Users who are already signed up cannot register with the same email twice.



4.2 Application Architecture

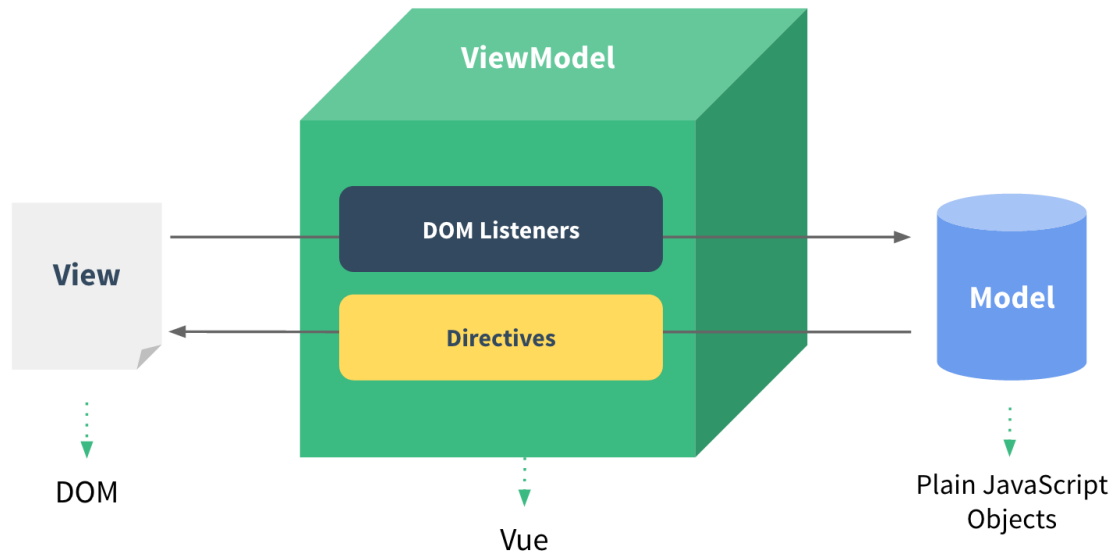


Figure 5 Vue Architecture (Vue.js, n.d.)

Front End Technologies

- Vue.js
- BootstrapVue
- CSS
- JavaScript
- Node Package Manager
- Chrome DevTools
- Git
- GitHub

Back End Technologies

- FullCalendar.io
- Firebase Authentication
- Firebase Hosting
- Cloud Firestore



4.3 Database Structure

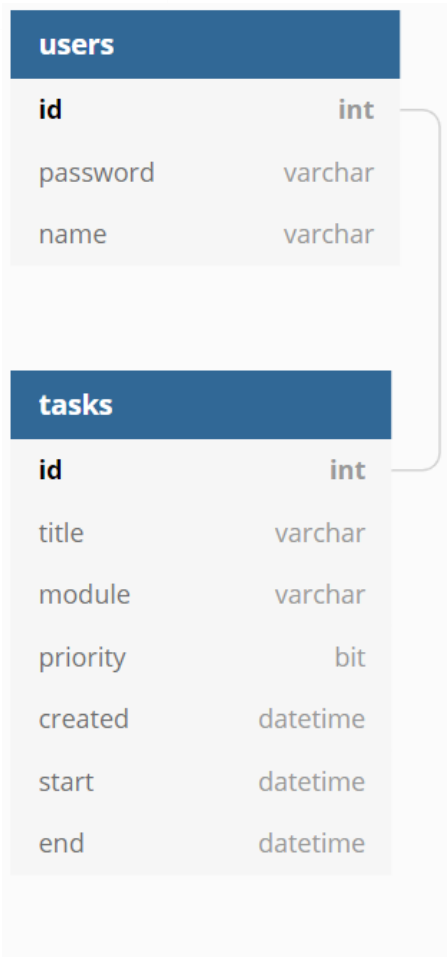


Figure 6 ERD Diagram



Chapter 5 Time Management

5.1 Introduction

Development on the project followed the Agile methodology. A backlog of user stories was assigned to 1 - 2-week sprints. These stories were then assessed using the MoSCoW prioritization technique and completed in order of their priority level. Below is a week by week record of the work undertaken from the very beginning of development on the project.

5.2 Semester 1

Plans for my initial project fell through so I moved to working on my own application. The project I decided on was a productivity / time management application. This will consist of a user-created task list that can be modified and assigned to dates on a calendar using drag and drop functionality.

After researching and comparing different frameworks, I decided to use Vue.js for the application. I was partially drawn to Vue as it was a relatively new framework and gained significant traction in recent years. As I had no prior experience with Vue, this also helped to provide me with more substance for my literature review. I also completed a set of initial wireframes. I will continue to develop these as I gain more clarity on what form the final application will take.

Other tasks in the preparation phase included setting up a OneDrive folder in order to share my files with my supervisor, creating a Trello Board and developing some User Stories. During this period, I was also developing a simple Hello World application using Vue.js to get a start with the framework.

I created a GitHub repository and began experimenting with Vue in order to get a better feel for it and become familiar with the core concepts. As I was without access to my laptop, I did this via an online code editor. I also worked extensively on my literature review, due for submission in January.

Towards the end of the semester, my attention was diverted to completing assignments for other modules while preparing for exams. During this time, I was working on my literature review and further developing my wireframes based on discussions with my supervisor.

5.3 Semester 2

For the duration of the month of January, I was focused on completing my literature review as well as making a start on coding the application in order to have a simple demo ready for the Interim presentation at the end of the month. I was also in the process of putting together PowerPoint a presentation. Due to the exams being moved to after Christmas, I was forced to balance working on the project with completing my remaining Semester 1 assignments.

Last week, I completed work on my literature review and submitted it for grading. I have also made a start on developing a demo for the application and have created an early draft of my PowerPoint presentation. For the next few days, I will be focused entirely on fleshing out my application demo as much as I can, as well as putting the finishing touches on my PowerPoint presentation.

At the end of January, I completed my interim presentation. The prototype I created consisted of a simple task list using Vue with which users can add to, assign a priority and delete from. However, this



was in a very crude and its primary intention was to illustrate that development on the application had begun, as well as to give a general idea of where the project is going.

Following the presentation, I began looking into a calendar API that was compatible with Vue to use for my application, as well as continuing work on refining my prototype. I decided that I would use the FullCalendar API for the calendar component of the application and implemented a basic configuration. As I was using Vue, documentation for this API was relatively limited.

I began researching and experimenting with the API in order to add drag and drop functionality as well as the scaling feature which would allow me to easily scale the duration events of the calendar. Once I successfully configured the FullCalendar API to activate drag and drop, work began on the highest priority user story: assigning events to a date on the calendar upon creation in the task manager component. To achieve this, I set up a Firestore database in order to be able to send and pull the data. I also spent some time implementing Bootstrap into the application.

Once I set up a database with Google Firestore and configured the calendar component of the application to pull from the database, the table in the database could now be updated when an event was created in the task management component. The calendar component could now pull from this data upon loading. I also implemented some Bootstrap styling, replacing the text links with icons and polishing the presentation of the task management component.

After this, my next task was to implement a delete function into the task management component in order to allow the user to remove items from the database, while continuing to work on my project documentation. I worked on my project report and added a delete function in order to delete events from the database. The next task was in fleshing out the priority level value in order to have the event priority reflected on the calendar via its colour.

Start and End time values were implemented into the Time Management form via BootstrapVue Timepicker components. The event placement on the Calendar component was now reflective of the Start and End time values selected by the user on the Create Event form.

I successfully added login and register components to the application. I also redesigned elements of the Task Management component using BootstrapVue. The Create Event form was now accessed via a button that opened a Modal containing the form.

Once the page was restyled and the login was implemented, I worked on ensuring that each user could only access their own unique data. Individual users could now log in and perform CRUD operations on their data. Finally, I began putting the finishing touches on the application. In the final weeks of development, I began applying the final layers of polish onto the application before submission.



Chapter 6 Problems / Challenges Encountered

Dropped Project

I had initially expressed interest in working on an industry application and had several meetings with the person running the project to discuss development. However, due to their deadlines conflicting with my own, as well as development requiring coordination with developers in other countries and the complexity of the application in contrast with the short timeframe, I decided I wanted to be more in control of my project.

Hardware Difficulties

Early in the year, I encountered significant hardware issues on my laptop as two columns on the keyboard had stopped functioning. Further issues with the keyboard lead me to send it away for repairs on three occasions throughout the year, impeding development on the application.

Reliance on Virtual Machines

As a result of being without a computer for large portions of the year, I was forced to rely on the virtual machines provided by the college. Unfortunately, I had further difficulties when trying to download the software I needed which seemingly could not be resolved. Later in the year, I found that directly pulling the project from GitHub once all the packages had already been installed bypassed these issues, however, it did not matter as the Virtual Machines were retired shortly afterwards.

No Access to Resources

As the college had been forced to close for a number of weeks and social distancing restrictions were put in place, I was left without a means to work on my project for an entire week. Thankfully, my laptop arrived in proper working order 7 days following the closure and this was no longer an issue for the remaining weeks of the semester.

API Documentation

While the FullCalendar API website provides extensive documentation, a lot of it is difficult to locate and navigating through the documents can be confusing. As well as this, very little is given in the way of Vue implementation beyond implementing the API with the most basic configuration. This has impeded development to a certain degree as I have been forced to blindly experiment with various aspects of the code in order to implement certain features, as well as search for alternative resources online.

New Tools and Tech

The most notable challenge that I encountered during the project was coming to grips with new technology. I decided to use the Vue framework for the development of the application in order to enhance my learning experience. This meant that development would proceed at a slower pace as I attempted to learn the technology while simultaneously building the app. This was less of an issue as development proceeded, however, and the efficiency of the framework (e.g. the virtual DOM) allowed me to partially make up for time lost in the early stages of development. Vue's extensive documentation also allowed me to come to grips with the technology at a relatively early phase.



Chapter 7 Learning and Outcomes

Final Application

The final iteration of the application delivers on the essential promises that were set out in the pre-development phase. The Virtual Planner provides users with an easy-to-use time management tool for planning and managing events. Users can register and log in via their email and password in order to access the main components of the application and interact with their events.

From the event management component, users can create new events and append data such as the date of the event, start & end times, description and select whether the event is high priority or not via a checkbox. From here, users also have the ability to update and delete their created events.

Once an event is assigned a date and time in the event management component, the event is then appended to the calendar component. Here, users can assign the event to new dates on the calendar via drag and drop functionality. The event can also be scaled via drag and drop, in order to change its duration. Hovering over events on the calendar triggers a popup event that reveals expanded event information. Any changes made in the calendar component are immediately reflected in the respective database document.

New Technology

One of the major goals I set myself for this project was to implement a new technology into the application that I hadn't used before. I was successful in this endeavour and the application was coded almost entirely using the Vue framework. This posed a great challenge as the window for completing the project was already relatively small given the other modules and assignments that were due to be completed over the course of the year and learning a new framework from scratch would prove to be time-consuming. Having weighed the pros and cons, I decided that the positives outweighed the negatives and that taking on the challenge would prove worth it for the learning experience.

Thankfully, I was correct in my decision and coming to grips with Vue has proven to be a worthwhile endeavour that has also allowed me to add to my CV. While the learning curve certainly slowed down development on the application to a certain degree, conquering the challenges that come with learning a new framework proved to be a thoroughly satisfying and rewarding experience. I enjoyed my time working with Vue due to its nicely segmented layout, speed, simplicity as well as the use of virtual DOM, which allowed me to see my changes reflected on the page as soon I as made them. This helped me to combat the learning curve to a certain extent, allowing me to make of for some lost time through the efficiency of the framework.

Applying Agile

Over the course of development on the application, I carried out work on the project while following the Agile methodology. I had previously applied the Agile framework during the development of my third-year project and more in depth during my work experience. This year's project presented me with an opportunity to apply concepts I was previously introduced to and had merely partially executed on past projects to the development of my own individual project from beginning to end.



At the start of development on the project, I created a set of user stories for my product backlog. Following discussions with my supervisor, I then prioritised these user stories using the MoSCoW prioritisation technique. These were aspects of SCRUM that I had executed many times before and was comfortable with carrying out. However, this marked the first time I carried out these tasks in a non-group setting and I feel an understanding and efficiency at carrying out these components of SCRUM are enhanced as a result.

Once the product backlog had been completed, I set about organising my user stories into sprints of 1 – 2-week periods. During the development of my third-year project, my team tended to lean towards two-week sprints, whereas during my work experience, my team tended more towards 1-week sprints. For my project, I leaned more towards the latter as I found that the shorter sprints kept me better focused as it forced me to work on the application every week, although this occasionally varied depending on external circumstances.

At the end of each sprint, I had a meeting with my project supervisor in order to give an update on the work that was carried out as well as discussing my progress. During each meeting, we would go through each of the user stories that were due to be completed in the sprint in order to assess whether they could be classified as ‘done’ or if more work needed to be carried out. Once this process was complete, we set about devising the next sprint and selecting the user stories that it would entail.

This was a relatively new experience for me. I feel this was an aspect of SCRUM that was not executed as efficiently on my third-year project as sprints often overran. Whilst I participated in code reviews and retrospectives during my work experience, my involvement was partially limited due to my inexperience as well as the length of the internship. In that sense, the meetings proved to be an excellent learning experience for and allowed me to hone my skills at carrying out sprints more efficiently.

What I Would Have Done Differently

Looking back on the project with hindsight, there are certain aspects of development that I might have approached differently if given the opportunity. One regret I had was not taking full advantage of the Vue UI. This is a tool designed to aid developers in creating and managing Vue projects. It allows you to efficiently install and configure project plugins and dependencies as well as running commonly executed scripts such as build and serve. The UI also provides a dashboard with analytics regarding the project’s assets, dependencies, status and more. I would have liked to have made more use of these features during the course of development.

Future Development

Due to time constraints and workload from other modules, it wasn’t possible to implement every feature that was discussed heading into the development and not every user story made it into the product backlog. If I were to continue working on the application in the future, there are a number of implementations I would consider. A feature that I had considered implementing during development but ultimately decided against was the ability for a user to sync their events with that of an external calendar e.g. Google Calendar. This would have required a fair amount of research and time-consumption on its own, so the idea was ultimately dropped to focus on the core application, however,



this is certainly an avenue I would be open to exploring in the future with time constraints no longer an issue.

Another feature that didn't quite make it into the application was the option to assign events to the calendar via drag and drop directly. Instead, in the final application, events are assigned to dates and times during creation, via a pop-up form. Events can then be moved around and scaled once they have already been created. While this feature wouldn't add any new functionality to the application, it would be a nice option to have implemented if I were to continue working on the application.

Conclusion

In conclusion, I feel I have taken a lot away from the development of PRJ400. I have acquired new technical skills that I can apply to future projects, as well an enhanced understanding of the principles of Agile and SCRUM. Finally, I have the completed application itself, a key component of my portfolio as I attempt to move on to the next stage of my Software Development career. Working on the project has proven to be a valuable experience as well as an experience I am sure to carry with me into future endeavours.

Bibliography

- Gore, A. (2018, May 7). *7 Vue.js Backends Compared*. Retrieved from Vue.js Developers: <https://vuejsdevelopers.com/2018/05/07/vue-js-backends-express-laravel-firebase-wordpress-django-rails/>
- Ivanova, S., & Georgiev, G. (2019). *Using modern web frameworks when developing*. Ruse: University of Ruse.
- Keeton, B. (2019, January 3). *The 11 Best Code Editors for 2019*. Retrieved from Elegant Themes: <https://www.elegantthemes.com/blog/resources/best-code-editors>
- Martin, S. (2019, July 5). *Angular vs React vs Vue: Which is the Best Choice for 2020? (Updated)*. Retrieved from Medium: <https://medium.com/hackernoon/angular-vs-react-vs-vue-which-is-the-best-choice-for-2019-16ce0deb3847>
- Saunders, E. G. (2019). *5 Strategies for Getting More Work Done in Less Time*. Cambridge: Harvard Business Review.
- Vivek. (2018, October 14). *7 advantages of using Vue.JS | The Progressive JavaScript Framework*. Retrieved from Inkoop: <https://www.inkoop.io/blog/7-advantages-of-using-vue-js/>
- Vue.js. (n.d.). *Getting Started*. Retrieved from Vue.js: <https://012.vuejs.org/guide/>
- Wheeler, K. (2018, July 13). *Jest vs Mocha: Which Should You Choose?* Retrieved from Noteworthy - The Journal Blog: <https://blog.usejournal.com/jest-vs-mocha-whats-the-difference-235df75ffdf3>

