



---

## GRADE TRACKER

---

### Final Year Project B.Sc.(Hons) in Software Development

BY

TOMMEY FAHERTY

LAURA FLAHERTY

GITHUB REPO:

[HTTPS://GITHUB.COM/TOMMEYFAHERTY/GRADETRACKER](https://github.com/TommeYFaherty/GradeTracker)

WEB APPLICATION:

[HTTPS://GRADETRACKER-6F72C.FIREBASEAPP.COM](https://gradetracker-6f72c.firebaseio.com)

MAY 10, 2020

**Advised by Kevin O'Brien**

DEPARTMENT OF COMPUTER SCIENCE AND APPLIED PHYSICS  
GALWAY-MAYO INSTITUTE OF TECHNOLOGY (GMIT)



# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Methodology</b>	<b>7</b>
2.1	Initial plan . . . . .	7
2.2	Implementation . . . . .	9
2.3	Meetings . . . . .	10
2.4	Development tools . . . . .	10
2.4.1	Github . . . . .	10
2.4.2	Firebase . . . . .	11
2.4.3	Overleaf . . . . .	11
2.5	Practical research . . . . .	11
<b>3</b>	<b>Technology Review</b>	<b>13</b>
3.1	Introduction to Technology Review . . . . .	13
3.2	Ionic React . . . . .	14
3.3	Redux . . . . .	15
3.4	Firebase . . . . .	17
3.4.1	Hosting: . . . . .	17
3.4.2	Authentication . . . . .	17
3.4.3	Database: . . . . .	18
3.5	Github . . . . .	19
3.6	Visual Studio Code . . . . .	19
<b>4</b>	<b>System Design</b>	<b>21</b>
4.1	Introduction to System Design . . . . .	21
4.2	React . . . . .	22
4.3	Redux . . . . .	23
4.4	Authentication . . . . .	23
4.4.1	How it Works . . . . .	23
4.4.2	User Interface: . . . . .	24
4.4.3	Firebase Integration . . . . .	24
4.5	Hosting . . . . .	26
4.6	Navigation . . . . .	26
4.6.1	Side Menu . . . . .	27

<b>5</b>	<b>System Evaluation</b>	<b>29</b>
5.1	Introduction to System Evaluation . . . . .	29
5.2	Objectives . . . . .	29
5.2.1	Creating a Progressive Web Application . . . . .	29
5.2.2	Authentication . . . . .	30
5.2.2.1	Testing the Authentication . . . . .	30
5.2.3	User Experience . . . . .	31
5.2.3.1	Testing User Experience Components . . . . .	31
5.2.4	Firebase database . . . . .	31
5.2.5	Calculation of grades . . . . .	32
5.3	Evaluating The Technologies Used . . . . .	32
5.3.1	Introduction to Evaluating The Technologies Used . . . . .	32
5.3.2	Features of Firebase . . . . .	33
5.3.3	Features of Ionic React . . . . .	33
5.3.4	Features of Redux . . . . .	33
5.3.5	Benefits of TypeScript . . . . .	34
5.4	Application functionality . . . . .	34
5.5	Limitations . . . . .	38
5.5.1	Introduction to Limitations . . . . .	38
5.5.2	Web based app vs mobile app . . . . .	39
5.5.3	Measure against objectives . . . . .	39
<b>6</b>	<b>Conclusion</b>	<b>41</b>
<b>A</b>		<b>47</b>
A.1	Github Repo . . . . .	47
A.2	Application URL . . . . .	47

# List of Figures

1.1	Current Logo for GradeTracker . . . . .	3
1.2	Potential Logos . . . . .	5
2.1	Storyboard 1. . . . .	8
2.2	Story Board 2. . . . .	8
2.3	Story Board 3. . . . .	8
2.4	Initial Plan. . . . .	9
2.5	Initial Plan. . . . .	9
3.1	Hierarchy of importance . . . . .	14
3.2	Diagram of how Redux works . . . . .	16
3.3	Steps for Firebase authentication . . . . .	18
4.1	Authentication diagram . . . . .	24
4.2	Routing authentication features . . . . .	24
4.3	App.tsx design . . . . .	26
4.4	How react router works in SideMenu.tsx . . . . .	27
4.5	Format of SideMenu.tsx . . . . .	27
4.6	The layout of the function renderMenuItems . . . . .	28
5.1	Typescript Code Suggestions . . . . .	34
5.2	Log In . . . . .	35
5.3	Register page . . . . .	35
5.4	Registration in progress . . . . .	36
5.5	Course Management Page . . . . .	36
5.6	More input boxes appear . . . . .	37
5.7	All inputs entered . . . . .	37
5.8	Home Page . . . . .	38
5.9	Side Menu . . . . .	38



# Chapter 1

## Introduction

Our grade tracking web-based application was the idea chosen as the purpose for our project, as we thought it would be an extremely useful service. Several colleges do not provide an in-depth look into the long term track of each students grades. We believe it can be a large benefit to students who aim to improve. We decided that a grade tracking app would be beneficial since it is a place to store a users grades but also display their long term progress. This can be eye-opening for students not doing their best or reassuring to those who are doing well. Reassurance leading to higher motivation, which is one of the outcomes we hope for users to achieve.

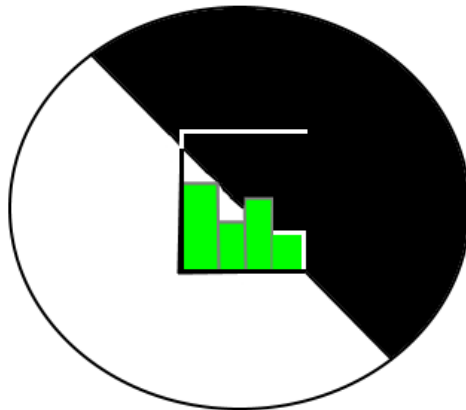


Figure 1.1: Current Logo for GradeTracker

We had discussed several ideas for our final year project but this idea resonated with us the most. It had the highest possible value in terms of long-term potential, learning outcomes, utility and scalability. We were also genuinely curious as to the work required to create a well designed web application, since we had

a relatively clear idea of what we wanted. The project had potential to be a long-term project, so after the year is finished if we wanted to continue working on it we could. We could even improve it with more features. This felt like it would be a more logical and rewarding experience because a lot of the projects we would work on in the future would likely be long-term projects. The learning outcomes had the highest potential to be the most rewarding factor. Since we had decided to use a new language as the basis for our project, we had learned a lot just from starting it. So much of what we were coding up was new to us, the entire project was a worth while learning experience. It was more worth while than attempting a project where we were more proficient in the language. We would have likely learned less and wouldn't have had the chance to broaden our knowledge in software. The utility of this project was the focus for us when discussing what we wanted to base our project on. We had been complaining about the issues of the system we were using at the time, for monitoring our grades. While it was provided by the college, because it was an official grades page there was no option for students to edit it. This was problematic since it wasn't mandatory for lecturers to use this feature. It meant some grades ended up in emails while others ended up on the grades page. Even then, some grades which were on the grades page were weighted incorrectly for the majority of the semester. This meant keeping track of ones grades was an external process anyways, so we thought that making a web application would be a good solution. We initially were planning to make the demographic for this college students. However, upon further discussion we realised that this had the capability to be useful for secondary school students too. Secondary school is already an incredibly stressful time with the outrageous amount of work, while teenagers are learning about themselves. In an education system where learning is geared towards a specific type of student, we figured that our product could help. It could especially be helpful when the work load is overwhelming and topics have little to no relation to each other. It would allow them to see what topics they struggle in and where specifically (written exams, practicals, orals, aural etc.). We had a meeting to determine how different the experience needed to be for college students in comparison to secondary school students. An example we had come up with is that college terms are split into years sectioned in semesters. While semesters are seldom mentioned when in secondary school. We also had the discussion of how weighting grades mainly applies to college students. Weighting grades in secondary school didn't make much sense since their final grade is solely dependent on the Leaving Cert exams. We came to the conclusion that entering it as a GPA made sense to avoid confusion if it wasn't weighted. After deciding to develop the grade tracker to be inclusive to secondary school students, we realised it might even be more beneficial to them if developed correctly. This leads into the last of the previously mentioned values, scalability. This project has the ability to grow and have several features added. At its core it is an application based on data presentation and manipulation. If we were to expand on the concept we could have different interesting ways of displaying results. For instance, we could use chart or pentagon based graph. We could also attempt to calculate a time efficiency estimate in terms



of studying. Another feature to help guide students when studying. These are just a few quickly thought of ideas for possible future features. In terms of scalability for user base, Firebase has a maximum of 100 simultaneous users at once. This is what the free version provides which is currently enough for our small two person project.

Upon developing the web application, we exchanged ideas in relation to the visual design of the web application. One of the things we debated on was the logo. We had a few ideas for one as seen in figure 1.2. What we had landed on was somewhat of a combination between option 1 and option 5. Option 2 and 3 were simple so we decided to opt for one of the more complex designs. Option 6 however was too complicated and neither of us felt confident in our sketching abilities, so we meshed the remaining two options which gave us the image in figure 1.1. While the image felt unfinished it was only a minor part of the project in terms of the work load.

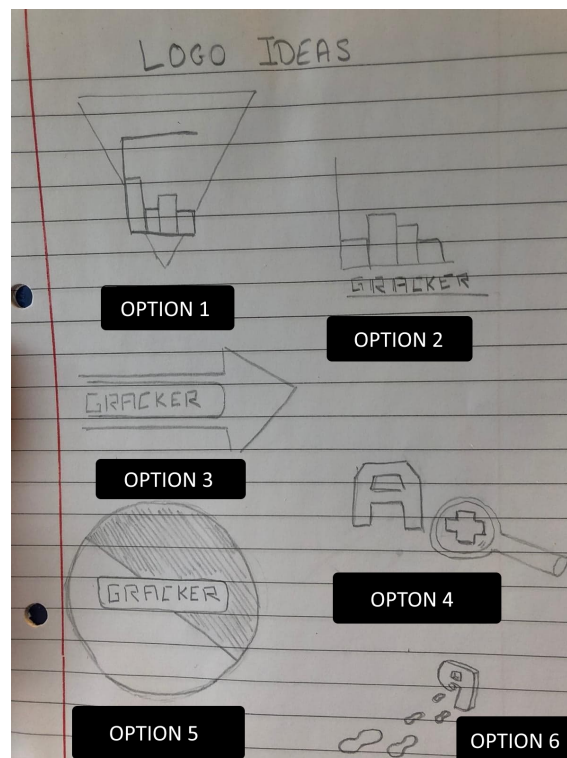


Figure 1.2: Potential Logos

We had briefly used Ionic before during one of our semesters, learning the basics and using TypeScript. It was primarily helpful in knowing where to go for more information. However we found ourselves relearning a lot of the information in Ionic since it was a long time since we had used it. Ionic React was the newest

addition to the Ionic framework. There was fewer tutorials and helpful guides for Ionic React than other options because it was so new. It appealed to us as we were looking into using React and React-native for our project. While researching we found that Ionic Reacts official framework was released. We thought it would be a good experience to attempt using this new framework to attempt our project since Ionic is a very popular framework. We had also looked into React-native and were very close to choosing it instead. We didn't end up choosing React-native however, because we felt it was overly complicated to use. It required us both to download a new IDE and would have taken more time to learn from scratch. The Ionic framework was much more intuitive and easy to learn. We also had some slight issues with hardware memory and speed, so we were hoping to use something minimally intensive like Visual Studios Code. We had already downloaded Visual Studios Code from multiple previous projects. Originally we looked at React-native because it is a mobile application framework. We were intending on having this mainly be a mobile application. We realised that it would serve as an overall better product as a web based application. With Ionic, it resizes the web application on a mobile device so it was less restrictive since it could be accessed on both a mobile device and desktop. It wouldn't require a download but would require an internet connection. We thought this trade was worth while. The application is targeted towards students. Most colleges have accessible wifi for attending students. At the time of writing this, most local secondary schools do not allow students to access their wifi. However, most student have roaming data with their mobile service provider. This appeared to be better than the alternative, since hardware storage is still an issue for many people currently.

In this dissertation we will cover our methodologies, a technology review, system design and a conclusion.

**By referencing the table of contents, these are the writers of each sections:**

*Tommey* → [1, 2.2, 2.3, 2.4, 3.2, 3.3, 3.6, 4.1, 4.2, 4.3, 5.1, 5.2.1, 5.2.4, 5.2.5, 5.3.3, 5.3.4]

*Laura* → [2.1, 2.5, 3.1, 3.4, 3.5, 4.4, 4.5, 4.6, 5.2.2, 5.2.3, 5.3.1, 5.3.2, 5.3.5, 5.4, 6]

## Chapter 2

# Methodology

### 2.1 Initial plan

For the development of our project we used the Scrum methodology. Scrum is based on Agile's principles [1]. Agile software development encourages responding to change over following a plan. However, a plan is still a good basis on where to start. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly. This is reflected in our weekly meetings. See Section 2.3 for further information on our weekly meetings. The scrum uses an iterative approach to development. Team members break down the end goals into smaller goals at the beginning [2] of development. After the initial story board was created, we had to negotiate the priorities, scope, and schedule.

When the purpose of the project was identified, we quickly realized that an in-depth plan was necessary. We needed to identify all the features that the application would possess and distribute the task of developing the application equally among us. A plan is necessary to increase efficiency, directs the tasks, helps with organization and is a motivation for development. The storyboard:

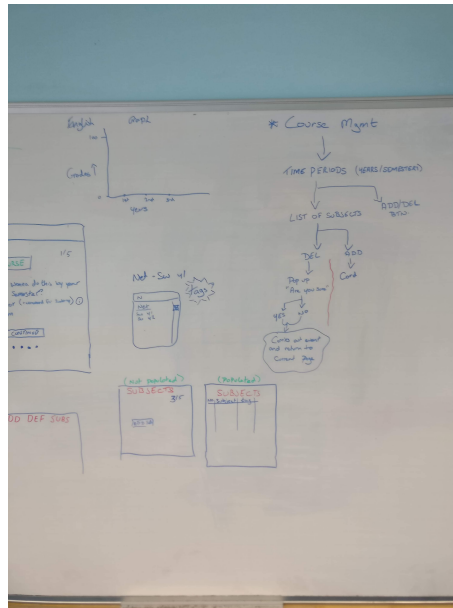


Figure 2.1: Storyboard 1.

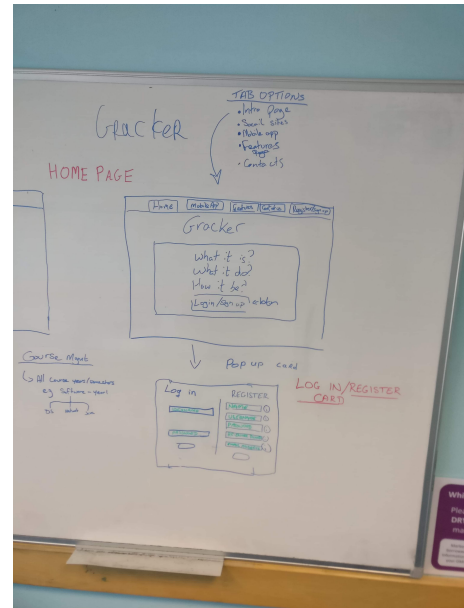


Figure 2.2: Story Board 2.

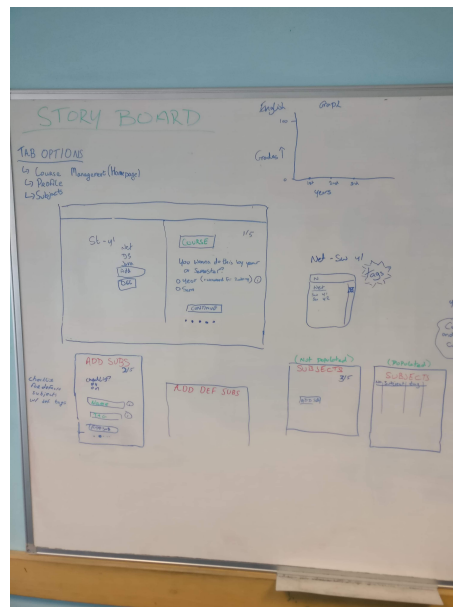


Figure 2.3: Story Board 3.

### Why is Storyboarding Important?

Storyboarding our project was very important. A storyboard will show if the ideas for the project works. It provides a basis for the plan. It shows where the direction of development is going. Our story board shows the path that the user will take when using the application. It provides a visual representation of the user's experience with using the app. Our storyboard helped with identifying errors at an early stage. It shows what order pages and features need to be developed in.

We had an initial plan that we created after the storyboard was developed. However, as outside factors disrupted our project development, the plan needed to be changed. The plan deviating a few times is a reactive response to deviation in expected outcomes. This is a feature of scrum.

The initial plan was created in October 2019.

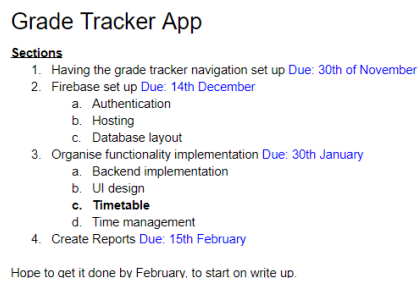


Figure 2.4: Initial Plan.

The revised plan was created in January 2020.

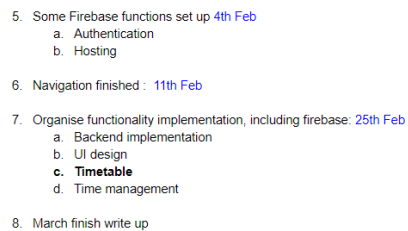


Figure 2.5: Initial Plan.

## 2.2 Implementation

When we were designing our plan to structure how we wanted to develop our app, we did not know a lot about how React works, Ionic's role in development and knew very little about Firebase and Redux. So as we learned more about

each component and began implementing them, our plans adapted and changed as we progressed. In some cases adding the other features to our project was straight forward, but there were cases where we needed to stray slightly from previous examples so that it would work in our application. For example we had considerable difficulty attempting to implement the Firebase database. Exchanging the data for an individual user from our app to Firebase was very complex and varied depending on how we built our application.

## 2.3 Meetings

Throughout development we met up quite frequently to assess all the work being carried out. We discussed what are plan was for the week, any difficulties we were running into and aired any confusion in relation to the development or design. We met up multiple times a week in our college library. We had booked a room where we could talk and display our work and future plans. We were able to freely discuss the what we felt was necessary and what wasn't. This prompted us to acknowledge the most important aspects of the project and prioritise those over more menial tasks with less value to the overall project. This was an efficient way to organise and segregate the workload as communicating in person meant there we were less likely to have issues with miscommunication.

## 2.4 Development tools

During the development of our grade tracking app we used several developmental tools. These tools we used made the development of the project easier for us in different ways. Collectively they made the process of completing a task faster and more efficient.

### 2.4.1 Github

One of the tools we used was Github. For the majority of our time in college we have been using Github for not only group project but for our respective projects too. In the case of this group project Github allowed for us to arrange and organise all of the work for this project into one area. There are something everyone needs to be aware of when using it, like how branches work and pushing and pulling changes. Ultimately Github makes the process for sharing work and implementing it into the current version very easy. It allows for us to flag issues we are having, create documentation and track the work contributed by each member. Having this set up throughout the year was very helpful but it became extremely helpful in the case of the COVID-19 outbreak that inhibited a lot of people from working. We no longer could meet after that in person and had to rely on our own hardware. Github relieved some of the stress of changing our routine and plan. We had all of our work in an accessible area which we could collectively work from.

### 2.4.2 Firebase

Another tool we used was Firebase's feature to host a version of our web application. This was a great way to be able to quickly show our progress with the application. All that was necessary was to build the version of the web app on a local machine and then deploy it to Firebase. Then instead of pulling out a laptop and running all of the code locally, we were able to show our current progress as long as we knew the URL.

### 2.4.3 Overleaf

Using Overleaf was solely for the purpose of writing the dissertation, however we decided to include it here as it was incredibly valuable. It worked similarly to Github in the sense that it allowed us both to make edits and commits to our work. It also gave us the ability to see how everything looked as we made edits and we were able to work on it simultaneously.

## 2.5 Practical research

We used a practical form of research for this project. Practical research means actual doing or using of something rather than theories and ideas [3]. This means that we used or watched videos of how Ionic React web applications work. This was to ensure that the new technology would work with our objectives. We used practical methods of testing the application. One of Ionic's features is that it allows for live development. Any changed made in the code was reflected locally on the developmental server (localhost:8100). We made sure the developmental server was working and that there were no coding errors before pushing changes to Github. This was to ensure that our project partner did not end up "pulling" any dead code, before beginning their development. We also practically tested the colour scheme of the application. We knew that the colour scheme needed to be bright, so we tested different bright colours. Bright colours are an attractive feature in applications and we know this because of our own use of various form of media. We mostly tested any ideas we had practically.





## Chapter 3

# Technology Review

### 3.1 Introduction to Technology Review

The objective of this project is to provide an easily navigable application, while challenging ourselves to use technology that we are not completely adept in. This chapter illustrates our research on various similar technologies and the reasons why we have ultimately chosen the aforementioned technologies. There is a justifiable reason for each technology chosen. We have weighed both the pros and cons against each other, measured the importance of each and what features would elevate our application. In doing this, there is an assuredly justifiable reason for each technology chosen. The main technologies used in this project are Ionic React, Redux and Firebase.

We consulted the image below (Figure 3.1), when choosing the technologies appropriate for this application. The application itself is number one. The app is to be an easily accessed application for students to track their grades in an less intimidating manner. The application and it's purpose is on top of the hierarchy.

The technology used in the product is number two on the hierarchy of importance. Any technology chosen is meant to enhance and improve the application. We also wanted to use technology that we have had limited interaction with in our four years of studying Software Development in the Galway-Mayo Institute of Technology. The reason for choosing technology that we may be unfamiliar with is to expand our knowledge and recreate the scenario of what learning a new computer language in a workplace environment is like.

Number three on the hierarchy is the technology used in the development process. This technology may not be featured in the code for the application itself but has helped tremendously in developing and building the application. It may make it easier for two-person development, or it may be an IDE (Integrated Development Environment) that facilitates application development. This technology needs to work well with the technology on the middle tier of the hierarchy and with the developer of the application

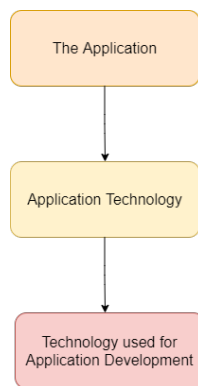


Figure 3.1: Hierarchy of importance

## 3.2 Ionic React

For this project we used Ionic React. When we started this project, React was the newest addition to the Ionic Framework. Its release was announced on October 14th 2019.[4] We were hoping to use a language that we hadn't used prior so we could get the most out of this project. We had decided to look into React and were initially choosing between React.js and React-native. We were leaning towards react-native as we wanted the user to have the option of using a mobile version of this application. However, we were instantly presented with some issues. React-native was relatively complex with very few consistent tutorials or forums with help in relation to what we were hoping to accomplish. Due to this we started looking into React.js when we had seen Ionic had a new just announce their official release of Ionic React. We had briefly used Ionic with TypeScript for a semester in second year, so we were reasonably familiar with how Ionic worked. Since it was a very new Ionic framework we were unsure as to whether we should use it or not, because it was likely going to have a similar problem to react-native where there was not enough help online. However we found that Ionic's Documentation on their website was very useful and easy to follow. There were even a few articles on the subject. Due to its popularity, it didn't appear to be as much of an issue as we thought it might of been.

Ionic React is a very well made and easy to use Ionic Framework. Its hefty amount of documentation leaves very few questions to be asked in relation to the web design. It also is very easy to see how quickly one could make a web page after getting familiar with it. It allows one to make nicely formatted pages which can be customised. The pages are also automatically reformatted based on the screen size of the user's device. This is a very useful feature since in this modern age, it's impossible to tell if a user will be accessing the page via computer, mobile, tablet or some other device. We used SCSS files for customisation of the web page components and wrote all the code in TypeScript.

We chose TypeScript over JavaScript because we read that there were some benefits to using it. For example, it points out errors at time of development reducing the chance of run-time errors. We had also used TypeScript before and we were comfortable with object oriented programming languages. While we were aiming to choose a language we hadn't used before to improve our skills, we felt using this new Ionic framework and Firebase was already a step in the right direction to improve our abilities. We also weren't overly-proficient in using TypeScript since we had only used it for a short time in our second year of college.

### 3.3 Redux

Along with Ionic React we used Redux, which is a state management tool commonly used with React. Having a state management tool is incredibly useful. When the application grows more components are introduced. Keeping track of the states shared across components becomes more cumbersome. There are three main features which allow us to manage the states. They are reducers, actions and stores. The reducers take the current state of the application and preform an action. They then replace the state with a new state. The actions are how data is sent from the application to the store. The store is where the state is kept. Redux keeps all states in a single store instead of something like flux which has multiple stores in the application. Redux uses a single store to avoid in application management issues. With multiple stores it can be difficult to entail what each store contains. As seen in the figure 3.2, the user interacts with the view. From there actions can be dispatched to have the application preform a specific task. The Reducers the takes in the arguments from the default initial state and the action and then figure out what changes need to be made. The new state is then passed to the view where it is re-rendered to show the new up to date data.

Some of the features that Redux has are the motivating factors for developers to use Redux. With Redux it uses encapsulation. In terms of state management, this means that it manages the store interaction logic leaving less work for the developers. This is one of the most significant features for Redux, as it gives developers an easy set up from the start for state management. Redux is also optimized. This means that that the performance of the code will be more efficient when rendering components in React. The components will only re-render the data if it is necessary. Which means unless the data has actually changed, Redux will acknowledge there is no reason to re-render the page because the display will be the same. Cutting out needless work for the application resulting in better performance and further optimization. Redux allows for a consistent vision of how the application can function. Maintaining predictability throughout the development of a project is very beneficial. It makes working on the project easier and allows the developers to quickly become more familiar with it. Redux does a great job with this quick to be when merging with React. It is made to work with React so evidently the workflow is seamless between them. We had

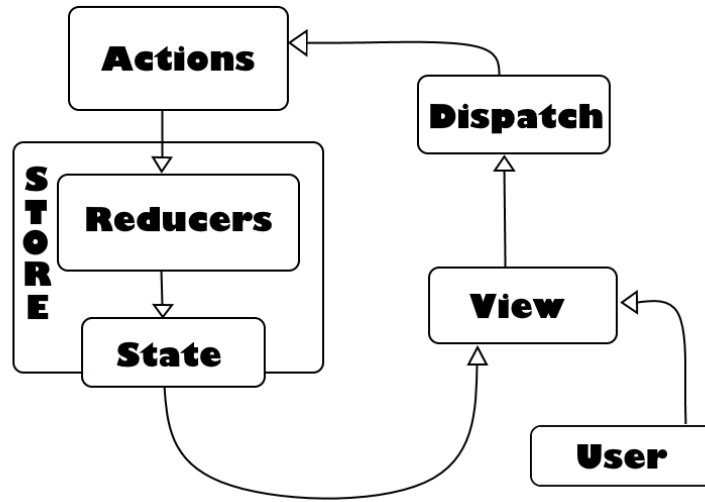


Figure 3.2: Diagram of how Redux works

been able to create a reducer and actions page giving the project a simple layout. It made it easy to make changes or additions to handling data when it was all neatly displayed in one place. Reliability is a huge element to take into consideration when deciding to use a product. While Redux has good representation among the React community, it seemed worthwhile to check if its compatibility had any differences with Ionic's framework. Even though Ionic-React was a new framework, tutorials detailing the use of Redux with ionic-react were quick to be posted. Which isn't overly surprising considering how commonly it seems to be used in React projects. Since it calls attention to its optimised features as a means of persuading and reassuring developers to use the product, it indicated it to be a good product. When adding features into a product, optimisation is incredibly important. It allows for the said project to better perform and in turn be more reliable. One of the final topics I will mention that Redux does well, is that it provides consistent maintainability for displaying states of the page. It is imperative to many projects that the user's experience is not hindered by having old data displayed or slow updates on pages. These seemingly small issues can be enough to turn a user away from a product if for example, it happens on every web page. It then can quickly become overwhelming and the sole focus of an aggravated user. With Redux it makes the developers job easier in this sense. Separating tasks using reducers and actions, assuming set up correctly, grants easy changes or additions to the state management. This being another crucial aspect to look at before implementing a product into a

project.

## 3.4 Firebase

This application is highly dependent on Firebase. Firebase is Google's mobile platform that helps you quickly develop high-quality applications [5]. Firebase has multiple products and our app uses three of them: hosting, authentication and Realtime database. The Firebase website provides extensive documentation which allows the developer to learn about the integration of Firebase into an application.

Firebase is an extremely popular product with over 1.5 million people using its product which is a very appealing reason to use it. It has been around since 2012 so many people who have problems or issues with Firebase have aired them onto websites such as Stackoverflow [6] and Youtube [7]. Where there are problems, there are solutions and both these websites provide a robust amount of information regarding most situations which any issue may arise. This was an extremely appealing reason to choose the Firebase product.

### 3.4.1 Hosting:

The idea of the app is that it can be accessed easily. The main way to do this is to host it on the internet. Before the decision to use Firebase was made, a few other options were considered. One of these options was the apache server. The Apache HTTP Server ("httpd") was launched in 1995 and it has been the most popular web server on the Internet since April 1996 [8]. It is estimated to be used by 55% of the internet. There are many advantages of using the Apache web server. Like Firebase, Apache provides useful documentation [9] and there is technical support available on many websites. It is highly reliable and can be installed easily. However, there is one major drawback and that is its compatibility. Our app is developed mostly in Ionic and React and Apache does not support that. On the hierarchy of importance, our app is number one. This means that the app itself has the highest priority and therefore hosting is not as important. Apache can not easily integrate with React so the decision to use the hosting feature of Firebase was made.

### 3.4.2 Authentication

Authentication is a very important part of any application. It is a necessary way to ensure that the user is correctly accessing their inputted information. It secures the users information. Most apps need to know the identity of a user. Knowing a user's identity allows an app to securely save user data in the cloud and provide the same personalized experience across all of the user's devices [10].

There are three steps to follow for implementing Firebase Authentication in an application. Set up the sign in methods. Customize the UI for signing in. Use

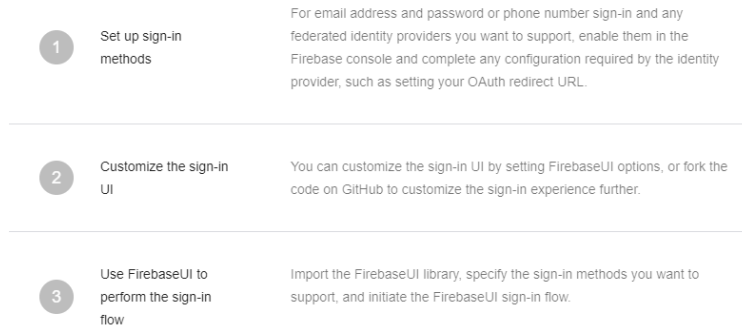


Figure 3.3: Steps for Firebase authentication

FirebaseUI to perform the sign-in flow.

The extensive documentation in which Firebase provides on their authentication product alone, is the reason we chose Firebase. There are many videos and tutorials available on the world wide web which detail how to integrate firebase into a web app. This makes it much easier on the developer.

### 3.4.3 Database:

The database is the most important part of our application. The user stores their grades' information on the database. Firebase offers two cloud-based, client-accessible database solutions that support realtime data syncing: [11]

#### 1. Realtime Database

- Firebase's original database
- Can record client connection status and provide updates every time the client's connection state changes.
- Used for JSON unstructured data.
- Simple, low latency.

#### 2. Cloud Firestore

- Product released in Ocotober 2017.
- Used for more in-depth queries or interactions with data, including offline local data.
- Used for large, complicated amounts of data.
- Allows for multiple databases to a single Firebase project.

Both solutions provide a reliable availability, however the Cloud Firestore was chosen for this project. This is because it is newer, and we wish for the user of our application to be able to easily access local data when they are offline.

## 3.5 Github

Github was key in our development process. Github is a prime example of the third tier in the hierarchy of importance 3.1. It did not necessarily contribute to the actual application; however, it did ease development. Github is a repository. The application's code is stored on the repository. This allows for the developers to “clone” the repo onto different computers, allowing the developer flexibility in coding.

Github also allowed to developer to see what changes have been made in each version or “commit”. This is a straightforward method of seeing what the other developer has contributed, changed, or fixed. It is a perfect tool for collaboration.

## 3.6 Visual Studio Code

We mainly used Visual Studios Code throughout development of this project. This is another example of the third tier on the hierarchy of importance as shown in figure 3.1. We had the option of using Reactide which is a dedicated IDE for React web development. The IDE was released in late 2019 [12]. We opted against using this since it felt unnecessary when we were already comfortable with Visual Studios Code and had used it for developing other Ionic projects.





## Chapter 4

# System Design

### 4.1 Introduction to System Design

This chapter will outline the decisions we made and route we took in creating this application. We will explain the thought process behind the structure of the application and alternative methods we decided against. The system design is a crucial aspect when developing a project. We figured it would be optimal to determine the structure of the project early on so we may implement ideas around our prioritised goal. After we had a clear idea of what we wanted the application to be, we researched different software to use and implement into the project. Obviously, it was an easier process to choose and eliminate after we knew what we needed to look for based on how we wanted the application to work. Once we were confident with the options we had narrowed it down to, we began working on creating Gracker. We tried to keep our code neat and structured, and the files segregated in a sensible way. We valued an organised set up when constructing the project and we tried to keep this ideology consistent throughout development. One of the nice things about working in a small project is that since there were so few of us communicating and ensuring understanding was a lot easier than if it were a larger group. However, in saying that, we still had faced problems in relating to the structure of the project. Upon revisiting some technologies on a more in depth scale and learning new ones, discovering a way to structure everything in a cohesive way was difficult. As we learned more progressing through the project, we questioned better our methods of arranging things for possible better ways. After a certain point, it was illogical to dedicate time to restructuring as there was a lot of work to be done and nothing in our project appeared egregious. The main topics which the system design revolves around in our project are React, Redux, Firebase and the visual aesthetic of our app.

## 4.2 React

When we began using Ionic React, it took us quite some time to understand how it operates. Revisiting Ionic made it somewhat easier to delve in to the set up and knowing where to find documentation on certain pieces of information. Ionic React consists of cross platform UI components and native functionality. The components are well documented on the Ionic React website. Ionic has great documentation for each of the frameworks we've used. From our experience on the Gracker project, the implementation of React and Ionic were very smooth and made a lot of the UI design work very easy for us to work on and change once everything was set up. The main reason we discussed using Ionic React is because we initially planned for users to be able to use our web application on their mobile phones. Ionic has a lot of the tools needed to make the process easier in terms of development. It resizes the web pages based on what device the user is on. This is perfect for us as it was one of our earliest plans to not have the user be limited by their device to use our site. We especially wanted this feature as so many website these days are accessed on the go. People use their mobile data to browse. It is incredibly convenient so we didn't want to inhibit or discourage a user from accessing the website. When a site is clearly not made to be on multiple platforms it is very obvious. It can make navigating what would normally be a very nice and simple website, feel very clunky and awkward. Our idea for our project is targeted at students, individuals who would be typically busy. Being able to check information on the go is a common occurrence for students. This is why we needed to take this aspect so heavily into consideration.

The web pages consisted of several of the features from Ionic React such as IonCards, IonTitles, IonContent and many other Ionic components. They were a simple way to construct a page and edit the css later. This is how a lot of Ionic based site are created. We designed it so the user would start off at the home page where information about the web application would be displayed. We then prompt the user to create an account to use the features our web-application provides. It is necessary for the user to have an account as the site stores information for each user. Signing up requires them to have an email address and from there they can access course management page. The course management page is where the user enters information on what course they're doing, the subject and each assessment with details about them. The course management page is solely for adding information like a new course, or new subjects. There is then a subject page which we planned to have show the user each relevant subject and the respective assessments for each. The final web page we have is the features page which is where we planned to go more in detail on the web application and its features. We decided this would be a good place to dedicate to the features for users to get a preview of what Gracker can do. We also wanted to put planned features on this page along with links to contact info. This currently is just the GradeTracker email which we used to create a Firebase account.

## 4.3 Redux

Using Redux was a new experience for us when implementing it into our project. Redux was used for updating web pages when live data changes are made. We also used it to allow the user to log in with our Firebase or register a new account. The log in button in the top right of the Home page shows the user options to log in using a password and username. The application then takes the password and username and checks if it exists in the Firebase database. If the username or password is entered wrong, it is not specified which is wrong. This is very common amongst online websites with account management, as it makes guessing email and password combinations much more difficult. After checking if Firebase contains the username and password when the user is attempting to log in, it dispatches the "SET\_USER\_STATE" action. The Reducer then takes in arguments from the initial state and from the action to determine what changes need to be made. This sets the user's account name to the email they provided along before the '@' symbol. So the user with the email **'BillyJoel@gmail.com'** would have the account name **'BillyJoel'**. After the user has logged in, the log in button is replaced by a log out button and the user is greeted by their username. Originally we hoped to have each user save and update data using Firebase and Redux, however we had only implemented so much of our plan. We got the log in system working, but learning to save the data was slightly different.

## 4.4 Authentication

### 4.4.1 How it Works

When the page is first loaded, the user is directed to the log in page, where they enter their username and password. If the user does not have an account, they should be told to go to the register page. Here they should enter username, email, and password. Once logged in or registration is completed the user is directed to the course management page. It should also be noted that once user is logged in the home page changes to an introductory page. The introductory page describes what "Gracker" is.

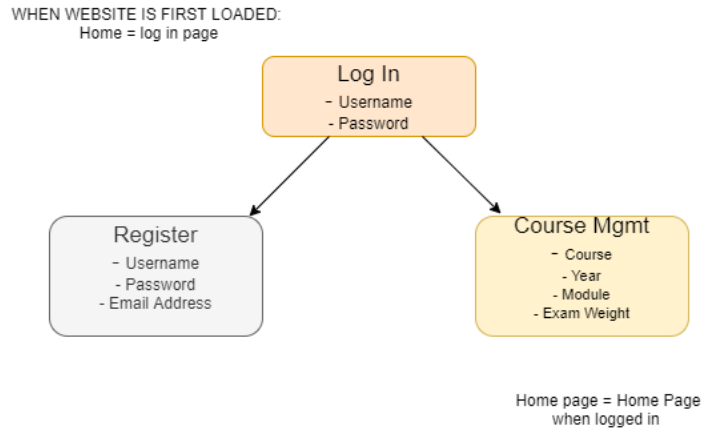


Figure 4.1: Authentication diagram

#### 4.4.2 User Interface:

When the page is first loaded, the user is directed to the log in page, where they enter their username and password. If the user does not have an account, they should be told to go to the register page. Here they should enter username, email, and password. Once logged in or registration is completed the user is directed to the course management page. It should also be noted that once user is logged in the home page changes to an introductory page. The introductory page describes what "Gracker" is.

To integrate the log in and register features of the application, the user interface needs to be adapted first. All pages need to be imported added to the React Router in App.tsx this includes LogIn.tsx and Register.tsx. Registration and log in very similar. Both pages take inputs for verification to go to the Course Management page. All inputs need to be declared and set eg **const [email, setEmail] = useState("")** to allow for function variables for verification.

```

<Route path="/LogIn" component={LogIn} exact={true} />
<Route path="/Register" component={Register} exact={true} />

```

Figure 4.2: Routing authentication features

#### 4.4.3 Firebase Integration

First to add firebase functionality to the application, the app needs to be registered on the firebase website to get access to the SDK. The firebase package needs to be added to the application using the Node command eg **npm i fire-**

**base.** Then a file for configuring firebase is created in the src folder (firebaseConfig.tsx). The details from registering the application are added to the configuration files. The application and firebase are now connected.

```
import * as firebase from 'firebase'
const config = {
  apiKey: "AIzaSyDMADOeA2pJhub886aa96wDgXn1gGpFJ8o",
  authDomain: "gradetracker-6f72c.firebaseio.com",
  databaseURL: "https://gradetracker-6f72c.firebaseio.com",
  projectId: "gradetracker-6f72c",
  storageBucket: "gradetracker-6f72c.appspot.com",
  messagingSenderId: "114427874629",
  appId: "1:114427874629:web:eccea7b695c65a3407022c",
}
firebase.initializeApp(config)
```

firebaseConfig.tsx contains multiple function including getting the current user, logging the user in and out and registering the user. Each function contains a firebase.auth(). method.

- firebase.auth().signInWithEmailAndPassword(email,password)
- firebase.auth().createUserWithEmailAndPassword(email,password)
- firebase.auth().signOut()

Each firebase function is surrounded by a try catch block to ensure the “catching” of errors, such as incorrect inputs. Each function from the configuration file needs to be imported into the log in and registration pages, to connect the firebase configuration settings to the inputs on both pages. There also needs to be a method in the .tsx files to connect the methods in firebaseConfig.tsx. The following is a code snippet of the logIn function in logIn.tsx which connects to the logUser function in firebaseConfig.tsx

```
async function loginUser(){
  setBusy(true)
  const res:any = await logUser(email, password)
  if(res){
    dispatch(setUserState(res.user.email))
    history.replace('/CourseMgmtPage')
    toast('Logged in!')
  }
  else {
    toast('email and password combination is not a match')
  }
  setBusy(false)
}
```

## 4.5 Hosting

Ionic provides comprehensive documentation on how to host an ionic app using Firebase. The two main requirements of a Progressive Web App are a Service Worker and a Web Manifest [13]. There are only a few simple steps to follow.

The service worker needs to be enabled in the index.ts file. To do so **service-Worker.register();** is added. The **ionic build** command is then ran so that the build directory is ready to deploy

The Firebase CLI is installed next using the **npm install -g firebase-tools** command. The CLI provides multiple prompts, asking what CLI features are to be used ( "Hosting: Configure and deploy Firebase Hosting sites."), to select a firebase project for the application (Choose the project you created on the Firebase website.) and what to use on as the public directory (Enter "build".) The last thing needed is to make sure caching headers are being set correctly. To do this, add a headers snippet to the firebase.json file.

```
"headers": [
  {
    "source": "**",
    "headers": [
      {
        "key": "Cache-Control",
        "value": "public, max-age=31536000"
      }
    ]
  }
]
```

Then the app is build and deployed using commands in the terminal

## 4.6 Navigation

The side menu is an essential component in this application because it allows for navigation between the different pages in the application. To allow for this there needs to be a router involved, followed by an IonSplitPane to separate the Menu and the Page on app.tsx

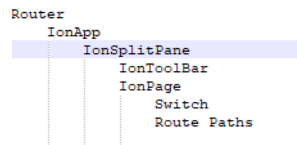


Figure 4.3: App.tsx design

### 4.6.1 Side Menu

A page interface and all pages necessary are declared in SideMenu.tsx. Note that this code was adapted from a blog post on Medium [14]

```
interface Page {
  title: string;
  path: string;
  icon: string;
}

const pages: Page[] = [
  { title: 'Home', path: '/', icon: 'home' },
  { title: 'Course Management', path: '/CourseMgmtPage', icon: 'information' },
  { title: 'Subjects', path: '/SubjectsPage', icon: 'information' },
  { title: 'Features', path: '/FeaturesPage', icon: 'information' },
];
```

To actually navigate between the pages the react router needs to be imported.

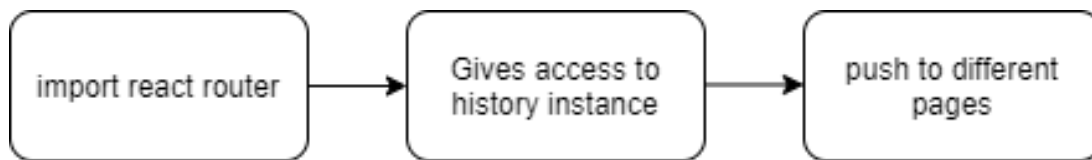


Figure 4.4: How react router works in SideMenu.tsx

To keep track of the current active page, the React Hooks API uses "states"

```
const [activePage, setActivePage] = useState(pages[0].title);
```

The code for the SideMenu is quite simple. The layout is as follows

```
IonMenu
├── IonHeader
│   ├── IonToolBar
│   │   └── IonTitle
│   └── IonContent
│       └── IonList
│           └── Call the RenderMenuItems() function
```

Figure 4.5: Format of SideMenu.tsx

**The renderMenuItems function:** This function maps the pages, toggles the menu, shows the labels of each page on a button and calls the navigation function.

```
IonMenuToggle
  IonItem Button
    Calls navigateToPage
    changes colour when clicked to show its active
```

Figure 4.6: The layout of the function renderMenuItems

The navigation function used the method shown above in Figure 4.4. It uses the history instance and setActivePage

```
const navigateToPage = (page: Page) => {
  history.push(page.path);
  setActivePage(page.title);
}
```



## Chapter 5

# System Evaluation

### 5.1 Introduction to System Evaluation

The main focus and objective of this web based application was to give a user an easily accessible web platform to store their grades. We wanted to be able to provide this for user's with students specifically in mind. We had them in mind while developing this application because we, as students, noticed this as an issue for us. Being able to spend time and resources to dedicate to solving a problem we faced made a lot of sense for us. We were able to highlight the gripes we had with previous versions of similar applications and services, and avoid going down that route while developing. We had several meetings throughout the inception of our project. We had several ambitious ideas for way in which this application could differ from other grade tracking applications. Primarily we wanted to implement the basic functionality as there was a lot of work for two people to get a service up and running with technologies we hadn't used before. We chose to do a project in this challenging was so we could gain the experience in coding with a new language. We also realise it is important to know how to go about a project with new technologies implemented. This is very common amongst many software companies. Being able to be adaptive and learn on the go is a very valuable quality. We were able to get insight as to how we would cope in a situation like this and experience in doing so. In this chapter we are going to look through the functionality of our project and compare it to what we hoped to achieve at the beginning.

### 5.2 Objectives

#### 5.2.1 Creating a Progressive Web Application

The process of creating a progressive web application (PWA) was extensive. We researched different aspects before the physical development of the project. Using Ionic allowed us to have the toolkit we needed to build the PWA. The

Ionic React framework used different UI components which made creating each page easy. Editing each individual component later on was easy too with the SCSS files. We wanted the user to be able to create a profile using their email and giving a password. From there they could go to the course management page where they could enter the details about their course. Subsequently they would enter the details revolving around their existing modules too. Allowing them to enter any amount of assessment per module and the weight for each. If a weight wasn't entered we planned to calculate the weight of everything, assuming it is all weighed evenly. This could be changed later. After all the information on the students modules is entered the subjects pages would display the current year and progress with all of the ongoing modules. The modules grades would be editable from here along with the weight. This makes it easy to keep the viewing page (subjects) separate from the editable page (course management) while keeping editing to a minimum on the subjects page. They only have ability to edit the essentials, which are things that can change along the year like a grade or an assessments weight.

## 5.2.2 Authentication

The log in function of the application was necessary in this project. It ensures that each customer experience is personalised. When the page is loaded there is a log in page, and an option to register to enable user authentication. There is also a log in button on the top right of the page. This button navigates to the log in page. The log in page and register page successfully achieve their requirements. Once a user is registered their details are stored in Firebase. So, when a user logs in, their username is displayed on the course management page. When a user has successfully logged in or registered, they are then brought to the course management page. The log in button also changes to a log out button. When a user is logged out, they are redirected to the home page and the course management no longer greets them.

### 5.2.2.1 Testing the Authentication

#### How the Log In and Registration was tested

Each step for the log in was incremental. This is evident in the commit section on the project's Github repository [15]. After each step there was a test to ensure the log in function was progressing satisfactorily.

- When both the log in page and the register page were created, they needed to be recognised by the app's side menu.
- Username, password, and other variables need to be stored data locally to check if they are set successfully (This sets up for Firebase later)
- Log In button (on top right corner of the toolbar) redirects user to log in page
- The link to register page on the log in page successfully redirects user.
- Passwords are hidden and not in plain text

- Use console / toast to determine if the log in is a success or failure
- Use toasts to check if password is viable (at least 6 characters)
- Use toasts to check if registration was successful

The log in and registration functions perform in a timely manner. This function does not "hang" or take a very long time to execute. The user's internet speed could also be a factor in execution time however the log in functionality should work without any long pauses or waiting times.

### 5.2.3 User Experience

The user experience was another important part of developing this app. There are university endorsed grade tracking applications that currently exist, but they may have other features, such as learning materials. The purpose of this application was to have an app that is deemed suitable for a student user. It should be easy to navigate to, store individual subjects/modules, the name of individual assignments, what weight each assignment has, what the user's grade was and how to calculate the "real" grade. Eg. User's assignment was worth 50%. The user's received the result of 80%. Their actual grade is equal to 40%. All this information is presented to each user without any complicated language or convoluted layout.

The user experience is as acceptable as the project's limitations (See section 5.5) allow it to be. All pages are clearly shown in the side menu. The log in button is clearly shown on the top right corner of the app. On the course management page is where the user input's their individual courses, what year they are in, what subject they are currently studying and what grade they got in that subject.

Initially on the course management page, the app greets the user. There is also only one input box. Once the user enters the name of the course that they are studying, all other input boxes are shown. All the data which has been entered into the input boxes are shown at the top of the page below the greeting.

#### 5.2.3.1 Testing User Experience Components

The following points are necessary to ensure that the user has a pleasant experience when utilizing the application.

- Ensure that menu shows all the pages
- What page is shown is the page highlighted in the side menu
- When course is entered in course management, other input boxes appear
- All links work
- Colour scheme is not busy
- No unnecessary clutter

### 5.2.4 Firebase database

We had decided to attempt using Firebase as the database for Gracker. When we were looking into React we had seen that a commonly used database in

conjunction with React was Firebase. It also had other features such as hosting and authentication which we were curious to try out. We planned to save the data to the Firebase database, use Firebase to host the up to date version of the web application and use Firebase authentication to verify user log-ins. While we had issues with saving the data across to Firebase, the user authentication and Firebase hosting worked well. We had worked out how we were hoping to save the data in JSON formatting.

### 5.2.5 Calculation of grades

The method we wanted to use for calculating grades was very simple. Each module would have any number of assessments. Each assessment weighted for different amounts, the sum of each assessments weight should be equal to 100. However, we would not stop the user from inputting a weight which would bring the total weight of a module to greater than 100. This is because of instances where extra credit can be gained. Instead the user would be notified with a message that the over all weight for a module is over 100 or vice versa if below. The user can enter the weight for each assessment in each module. If no weight is entered, calculations will be done to assume each assessment is worth equal weight in the module. These weights can be changed anytime. If there are two assessments put in and one has a user input weight and the other does not, calculations will be done to set the weight of the undefined assessment to the remaining amount. For example, if there is an "MCQ" assessment with a weight of "30%" and the a "written exam" assessment with a undefined weight. If the "written exam" is saved without any specified weight for the exam, it will be calculated based on the other assessments. In this case "MCQ" is the only other assessment therefore the "written exam" assessment's weight is equal to  $100\% - 30\% = 70\%$ . This is done for every module. It allows for the overall grade/GPA to be calculated efficiently.

## 5.3 Evaluating The Technologies Used

### 5.3.1 Introduction to Evaluating The Technologies Used

It was always our intention to use technologies that we have had a very limited experience with. Firebase, Ionic React and Redux all fit into that category. All three of these technologies are intertwined with each other. Ionic React uses Firebase to host it's applications and to authenticate users. As an Ionic React app grows it becomes important to store data in a linear way. This is where Redux comes in. Redux stores states. For example, if a user logs in, Redux sets the user state. When the user logs in Redux dispatches an action, sets the user state, selects the username and it is then displayed at the top of the course management page. Ionic React was only released in the summer of 2019. Because Ionic React is the base for this project everything else is built upon it. Learning how to integrate older technologies like Firebase into a new technology

like Ionic React emerged to be a very difficult yet rewarding challenge.

### 5.3.2 Features of Firebase

Firebase is an important feature of our application. It has multiple capabilities and we used it for hosting and authentication. The implementation of the Firebase database proved to be more challenging than we previously thought. The database was meant to store the user's course, year, subject and grade. However, this information is now stored at the top of the course management page.

#### **Authentication:**

As previously described in Section 5.2.2 the authentication was a key feature in the finished product. It successfully logged the user in, registered the user and could log the user out. This was all achieved once Firebase was imported into the project and the configuration file (`firebaseConfig.js`) was complete. The Firebase predefined methods were the core to all authentication functionality.

#### **Hosting:**

Hosting allows the user to access the web application without having to download the git repository. The accessibility objective was achieved here with the successful hosted web application. Once the Firebase CLI was imported into the project, Firebase guided the developer on steps for hosting via command prompts. Ionic React also provides documentation on hosting with Firebase. [13]. The hosting allows the user to access the application wherever they are (provided that they have an internet connection and a browser).

### 5.3.3 Features of Ionic React

The Ionic React framework was what we used to create the web pages and locally host a version before pushing it to Github. We used many of the different UI components on the web pages. It felt very rewarding using Ionic React once we got more comfortable with it. When we were making additions or edits, we were able to view the changes live from the local version. This was incredibly useful for all types of changes especially the for the development of the UI. Since we were using React we felt the need to use Redux with it. They were very commonly paired together when building web based applications. Managing the states on an application where the user can update their data at most points meant that the page would likely need to be re-rendered to stay up to date.

### 5.3.4 Features of Redux

This was our first experience using Redux in a project. Neither of us had used it prior so it was a very new topic for us to learn about and how it worked. Learning about Redux was one of the more challenging tasks in the project. It was difficult to fully understand all of the other concepts that were necessary to be aware of when learning about Redux. There are several concepts which Redux uses such as states, reducers, dispatching, actions and stores. Then also

learning how Redux is used with React. It is very beneficial to learn about, but also time consuming. Redux was assessed to be necessary to our project as we felt it would make state management easier to control as we were learning about it. We used Redux to enable user logins. Once the user is able to sign in with an email and password and redux allows us to assign a username to them based on the email they entered when signing up. Once they revisit the Home page they are then greeted by their username.

### 5.3.5 Benefits of TypeScript

React uses typescript instead of Javascript. Typescript is a superset of of Javascript which means that we can also use Javascript in any .ts file. Typescript is newer so it supports long-term maintenance. Typescript offers code suggestions which was a very helpful feature in developing the application. As you can see in Figure 5.1 below. Typescript also shows any errors as soon as they appear. The error is indicated by a red curvy underline. Typescript also offers self-documenting code. This means that it is easier for other Typescript developer to understand, what you are doing. Features like interfaces, enums and types are all apart of Typescripts self-documenting code

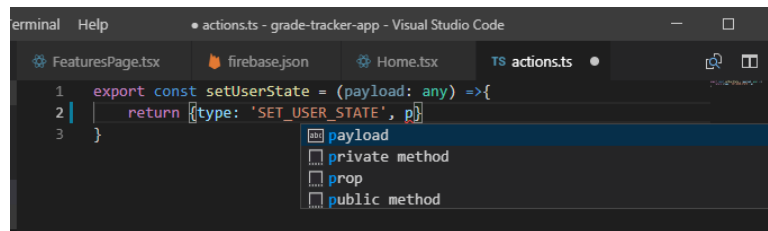


Figure 5.1: Typescript Code Suggestions

## 5.4 Application functionality

This section uses images to illustrate the functionality of the application and emulate the user experience.

When the user first enters the web application's url into the browser they are brought to the Log In page.

Menu	Log In
Home	Email
Course Management	Password
Subjects	<div>LOG IN</div>
Features	Don't have an account? click <a href="#">here</a>

Figure 5.2: Log In

If the user does not have an account they are instructed to click a link that redirects them to the register page

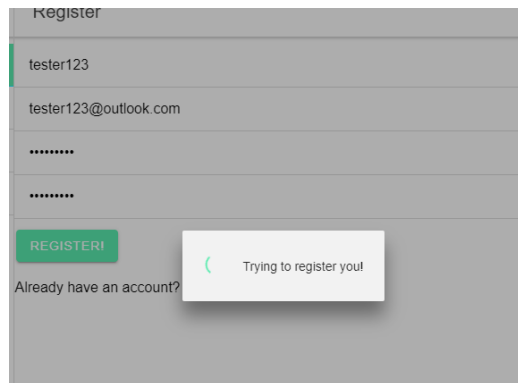
Menu	Register
Home	Username
Course Management	Email address
Subjects	Password
Features	Confirm Password

REGISTER!

Already have an account? click [here](#) to log in!

Figure 5.3: Register page

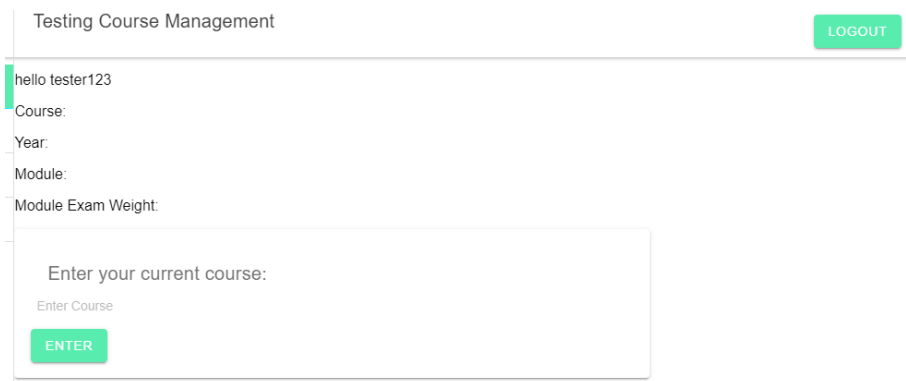
If the log in or registration details are incorrect there is a toast displayed at the bottom of the screen explaining the issue.



The image shows a registration form titled "Register". It contains input fields for "username" (filled with "tester123"), "email" (filled with "tester123@outlook.com"), "password" (masked with "\*\*\*\*\*"), and "confirm password" (masked with "\*\*\*\*\*"). A green "REGISTER!" button is at the bottom left. A white modal box with a green checkmark and the text "Trying to register you!" is centered over the form. Below the button is the text "Already have an account?".

Figure 5.4: Registration in progress

Once the user is registered or logged in they are then redirected to the Course Management page. At the top of the Course Management page the application greets the user.

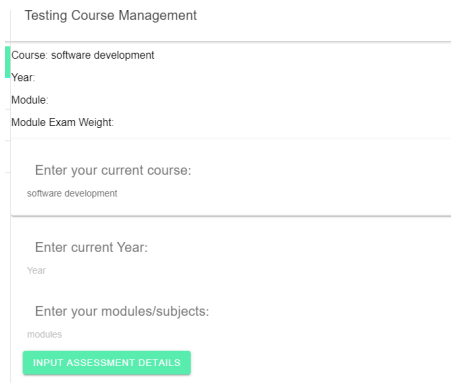


The image shows the "Testing Course Management" page. At the top right is a green "LOGOUT" button. Below the header, the user is greeted with "hello tester123". There are labels for "Course:", "Year:", "Module:", and "Module Exam Weight:". A large input box contains the text "Enter your current course:" and "Enter Course". A green "ENTER" button is at the bottom of the input box.

Figure 5.5: Course Management Page

There is only one input box showing currently. When the current course is entered and the enter button is pressed, more input boxes are shown.





Testing Course Management

Course: software development

Year:

Module:

Module Exam Weight:

Enter your current course:  
software development

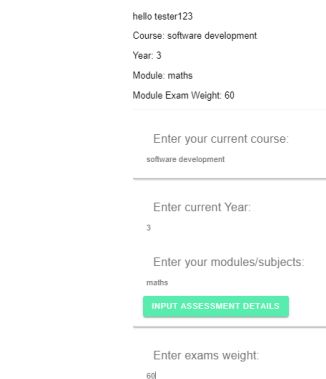
Enter current Year:  
Year

Enter your modules/subjects:  
modules

INPUT ASSESSMENT DETAILS

Figure 5.6: More input boxes appear

When all details are entered they are stored at the top of the page, below the user greeting.



hello tester123

Course: software development

Year: 3

Module: maths

Module Exam Weight: 60

Enter your current course:  
software development

Enter current Year:  
3

Enter your modules/subjects:  
maths

INPUT ASSESSMENT DETAILS

Enter exams weight:  
60

Figure 5.7: All inputs entered

The log in button changes to a log out button on the course management page. When the log out button the user is redirected to the home page. The home page explains the functionality of the application.

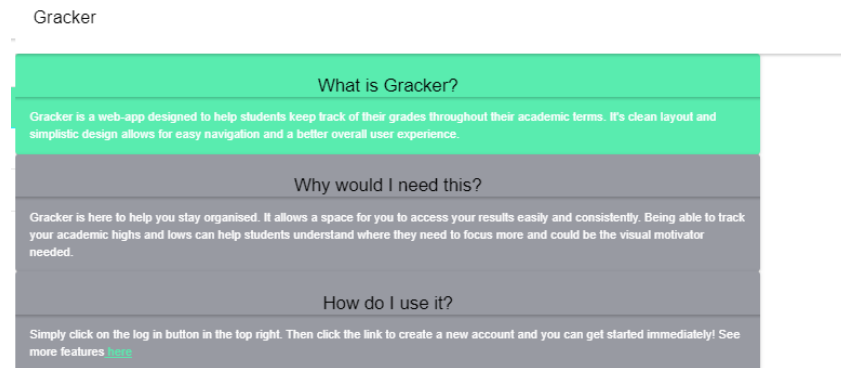


Figure 5.8: Home Page

The user can navigate through the app using the side menu feature. The side menu is hidden on smaller screen such as on smart phones but when the user swipes right, it appears.

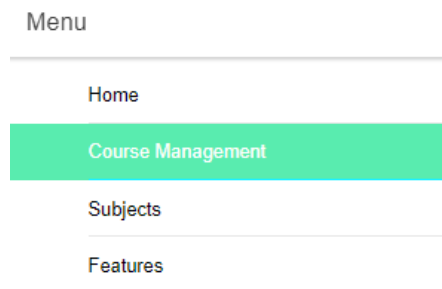


Figure 5.9: Side Menu

That is all the functionality of our application. Information is clearly presented for the user, and the application can be easily accessed.

## 5.5 Limitations

### 5.5.1 Introduction to Limitations

All applications have limitations. Some limitations may be on a functional level and other limitations may arise after the application is finished. Unfortunately, limitations exist in this application. There are limitations that we foresaw before undertaking the project and there are limitations that arose while developing the application.

In the initial plan we had hoped to add a timetable function to the application. The purpose of the timetable is self-explanatory however it would add an extra layer of complexity to the application. The application's purpose is to be clear of any clutter. The timetable may hinder this. The timetable could also be an advantage to the user too. Having a place where the user can not only track their grades but also track what time and location each lesson takes place could be beneficial to the user.

The calculation function was not successfully implemented in the application. While the weight of the grade for an assignment is stored in a database, it does not calculate the total grade. The purpose of this function was to rid the user of any confusion in their "real grade". The user's information (course, year, module, grade weight) is not represented when the user logs in. The information is stored but it is not shown / rendered when the user is logged in.

There are no reports or graphs used to depict the user's progression in education. The graph would be a visual representation of the user's educational career. It would clearly show which subjects the user needs to put more effort into and which subjects the user is already excelling. The reports or graphs could also be shown to a parent or guardian to as an indication of their child's advancement. The web application is currently accessed through a web browser. If the user could download the application, they may be able to calculate their grades offline. The enablement of offline use would be very helpful to users. Users would no longer depend on differing internet speeds. Offline use would also be convenient for a timetable function. The user could quickly identify where they need to be.

While this application has limitations, it has achieved some of the original goals outlined in the introduction. The app is easy to use, easy to access and it is not cluttered. It's colour code ensures that it is bright and attractive to the user's eye.

### 5.5.2 Web based app vs mobile app

We chose to do a web based application over a mobile application for several reasons. A web application was more accessible, we wanted to explore Ionic's new framework and we were able to use several features from Firebase in conjunction. We deployed it to Firebase which hosted Gracker and we used Firebase's authentication feature for users logging in and signing up. The idea for this project appeared to be suited for both a web app and a mobile app. Both options had benefits but we were able to experiment with more technologies by choosing a web app. Learning more about web based development is important as the internet is such a huge part of the world today.

### 5.5.3 Measure against objectives

We had several objectives from the beginning of this project. We heavily leaned into the most beneficial one which was learning to use new technologies. Once we had thought out the idea of developing a grade tracking app, we set a series

of objectives we wanted to accomplish along side learning the new technologies. We wanted to use features from Firebase such as the hosting, authentication and database. We hoped to fully understand Redux and Ionic React and why they are used together. We wanted to provide a web application which looked neat, simplistic and was efficient. We successfully implemented the use of Firebase hosting and the authentication for logging in. We had a much more difficult time setting up the database with our app. Since there were so many different ways to implement the database into the application, we struggled to find a way that was done in the way we needed it done. Our knowledge of Redux and React and how they work together has increased greatly. We kept our application very simple with minimal clutter and easy to navigate with a burger menu. We realised that burger menus seemed to be very common amongst many websites. We assumed this was for good reason and concluded that out of the multiple ways to allow a user to navigate the app, it was the most convenient. It allowed more room for small screens, quick navigation to pages and is commonly enough used, that most users understand how a burger menu works. We also chose a simple colour scheme with only two colours as we did not want to be overwhelming to the user visually.

## Chapter 6

# Conclusion

The objective of this project was to develop an application for students in both second and third level institutions to easily track their progression in education. The web-based application is an easily accessed and easy to use. Most third level institutions provide a service where the lecturer inputs the student's grades, however often there is an element of confusion involved with this. The grade that is input often does not reflect the weight it may carry for the student's final grade. Each assignment carries a different weight. The grades in this service may also be scattered around the website, or only showing the result of some assignments. Our application provides students a place where they can store all their information regarding their grades, their subjects and their course and change it as they wish. With all their information clearly represented they can correctly assess how they are doing in each subject. With graphs or reports the user can see which subject they are excelling at and which subject they need to work harder in. This provides a greater indicator in how the user's educational career is progressing.

We wished to create this application by using technologies that we had very little to no experience with before. By challenging ourselves to learn about these new technologies outside the classroom, we wished to prepare ourselves for our future careers. Often in a new job – especially in the IT industry, a new employee might possess only a small amount of knowledge of the technologies that their employer uses, however they are expected to quickly adapt to how the company works. So, by learning about new technology without the assistance of a lecturer, we are preparing ourselves to enter the workforce. The aim was to develop this application between October of 2019 and May 2020.

The below is a list of outcomes

- Learned about new technologies that we were previously unfamiliar with.
- Learned how to communicate with a project partner over the course of a 6 month period for the purpose of continuing the development of the project

- Firebase is hosting the application.
- The layout of the application is clear. There is no unnecessary clutter
- When the user first enters the web application's URL into the browser they are brought to the Log In page
- If the user does not have an account they are instructed to click a link that redirects them to the register page
- If the log in or registration details are incorrect there is a toast displayed at the bottom of the screen explaining the issue.
- Once the user is registered or logged in they are then redirected to the Course Management page
- At the top of the Course Management page the application greets the user
- Only one input box is shown initially on the Course Management page
- When the current course is entered and the enter button is pressed, more input boxes are shown.
- When all details are entered they are stored at the top of the page, below the user greeting
- The log in button is clicked it changes to a log out button on the course management page.
- When the log out button is clicked the user is redirected to the home page.
- The homepage explains the functionality of the application.

We feel that we have learned a lot from developing this project. Of course, learning about new technologies was a large reason as to why we chose the technologies we did (Ionic – React, Redux and Firebase). But we also learned about working on a long-term group project. Working as a team of two, we have recreated a likely scenario that would occur in a working environment.

We have gained insight on how to learn brand new technologies without the guidance of a lecturer or professional. The documentation for each technology quickly became our most visited web pages. We have learned that some documentation is easier to understand than others. For example, the Firebase documentation [10] was clearer than the Redux documentation. [16]. Understanding documentation largely depends on what existing knowledge the reader possesses. Technologies like Firebase have existed for quite a while but Ionic React is a relatively new technology so there is a limited number of tutorials available, so documentation was extremely important here.

On team projects planning is essential. There is an overall plan and then there is further plans within the plan. Every time we had a project meeting, we had to discuss what we had to organize and what each person was responsible

for. Planning is one thing but following it was another. We quickly discovered that there are other outside forces which can hinder our application development. Issues such as other college deadlines, personal reasons, and a lack of access of development tools all inhibited our progression. We both learned that addressing and overcoming these forces is essential.

It is my belief that there are multiple opportunities for future investigation. Once all the intended functions are completed, the application could become a serious product and be used by students all over the world. This application has potential to be a world player in the educational application section on any app store. There would need to be a careful balance between keeping the intended purposes and any future monetizing. For example, the purpose of this app was to be clear of any “clutter” like pop up ads and keep it simple for the user to use. If this application were to be monetized it would need advertisements to fund it. Perhaps, to honour the initial purposes of this application, advertisements would be vetted to be plain and simple. The ads could be in the side menu below the list of pages. The web application could also have a parent setting. A parent or guardian may have different uses for a grade tracker application. They could input their children’s grade, and anonymously compare their child’s progress to other children. With the information that they have stored in their web application they could also see which subject their child needs help in.

The web application could also just be further personalized for the user. The user might change the background colour or choose a different layout for their app. This possible personalization feature may appeal to more artistic users. It would also be good to investigate further into other Firebase authentication features. Firebase allows authentication with a multitude of social media sites. The user may be able to connect their Twitter or Facebook account with the application. This may create a more social aspect of tracking grades. Perhaps users can see how they are progressing compared to someone from a different educational institution.

Although we did not fully reach our objectives, we did reach the important ones that may not be tangential. Learning how to work with another person over the course of six months is vital. We learned how to accommodate to each other’s skills set. In an area where one person may have a weaker skill set, the other person may excel in. In an area where both skill sets may be weaker, both people had to work together to find a solution. In an area where both skill sets excel, both people had to carefully distribute the workload, so no one was doing most of the work.

Learning about Ionic React, Redux and Firebase in our fourth year of Software Development will definitely be a benefit to us in the future. Ionic React was released in August 2019, only a few months before we started undertaking this project. There is a rise in popularity with Ionic type apps and React is growing every day. Initially redux was not something that we were familiar with, yet quickly understood that is essential in saving states and is a “must” with complex Ionic React applications. Firebase is already a massively popular technology. It has a multitude of uses and we only scratched the surface of its capabilities. Firebase is Google’s mobile platform and a lot of users of Firebase

have already had some issues integrating Firebase or aired their confusion on online platforms. We learned how to take from the available solutions online and apply it to our own application. While no problem was identical, we could adapt the solution to match our purpose. Without a doubt, the knowledge that we have gained from this project will stick with us forever.



# Bibliography

- [1] The agile manifesto. Available: <http://agilemanifesto.org>.
- [2] Scrum-in-software-development. Available: <https://www.scrum.org/resources/what-is-scrum>.
- [3] Practical-research. Available: <https://www.scribd.com/document/352255786/Practical-Research-2-Quantitative-Research>.
- [4] Ionic-react announcement. Available: <https://ionicframework.com/blog/announcing-ionic-react>.
- [5] Firebase website. Available: <https://firebase.google.com>.
- [6] Stack overflow. Available: <https://stackoverflow.com>.
- [7] Youtube. Available: <https://www.youtube.com>.
- [8] Apache web server website. Available: <https://httpd.apache.org>.
- [9] Apache documentation. Available: <https://httpd.apache.org/docs>.
- [10] Firebase doc guide. Available: <https://firebase.google.com/docs/auth/?authuser=1>.
- [11] Choose a database. Available: <https://firebase.google.com/docs/database/rtdb-vs-firestore>.
- [12] Reactide — linkedin. Available: <https://www.linkedin.com/company/reactide>.
- [13] Progressive-web-apps-react. Available: <https://ionicframework.com/docs/react/pwa>.
- [14] Side menu. Available: <https://medium.com/@BartInTheField/lets-add-a-side-menu-and-a-toast-to-an-ionic-react-application-6b503955fadf>.
- [15] Project github repo. Available: <https://github.com/TommeyFaherty/GradeTracker>.
- [16] React redux docs. Available: <https://react-redux.js.org>.



# Appendix A

## A.1 Github Repo

<https://github.com/TommeyFaherty/GradeTracker>

## A.2 Application URL

<https://gradetracker-6f72c.firebaseio.com>