



# **Life Enhance – Self Improvement Web Application Final Project Report**

**TU856  
BSc in Computer Science**

**Dan Andrei Pirlea**

**C18444682**

**Supervisor: Giulia Vilone**

School of Computer Science  
Technological University, Dublin

**8<sup>th</sup> of April 2022**

## **Abstract**

Life can be very hectic at times, between work, family, and other commitments there is very limited time for yourself. Consequently, only 31.3% of Irish people are highly active<sup>1</sup>. Furthermore, 1 in 4 Irish people will suffer from a mental health issue at some point in their life<sup>2</sup>. Due to increasingly busier lifestyles, the food habits of Irish people have also started changing. 17% of the evening meals are no longer the main meal of the day, with smaller meals and snacks changing the evening meal habits<sup>3</sup>.

The goal for this project is to provide a fun and engaging web application that will help users keep track of their life. The web application is called “Life Enhance” and it combines various life planning tools in an entertaining way, which makes the web application feel more like a “game” rather than a chore.

The purpose of this project is to try get people to form a new, healthy habit. Forming habits is not an easy task, it is usually a very difficult and uninteresting process. Therefore, this application will hopefully make the habit-forming process be a more engaging one.

## Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

A handwritten signature in black ink, appearing to read 'Dan', positioned above a horizontal line.

Dan Andrei Pirlea

4<sup>th</sup> Of January 2022

## **Acknowledgements**

I would like to thank my project supervisor Giulia Vilone for her constant support and guidance throughout the project, as well as my family and friends who always made sure I'm on track.

## Table of Contents

Table of Figures .....	7
1. Introduction .....	8
1.1. Project Background .....	8
1.2. Project Description .....	8
1.3. Project Aims and Objectives .....	9
1.4. Project Scope .....	9
1.5. Thesis Roadmap .....	9
2. Literature Review .....	11
2.1. Introduction .....	11
2.2. Alternative Existing Solutions to Your Problem .....	11
2.3. Technologies you've researched .....	14
2.4. Other Research done .....	15
2.5. Existing Final Year Projects .....	16
2.6. Conclusions .....	17
3. System Design .....	18
3.1. Introduction .....	18
3.2. Software Methodology .....	18
3.3. Overview of System .....	19
3.3.1 Changes to Overview of System during development .....	20
3.4 Presentation Layer .....	20
3.4.1 Changes to presentation layer due to development .....	27
3.5 Application Layer .....	27
3.6 Database Layer .....	28
3.6.1 Changes to Database Layer due to development .....	29
3.7. Conclusions .....	30
4. Experiment Development .....	31
4.1. Introduction .....	31
4.2. Creating the React App .....	31
4.3. Front End Development(React) .....	31
4.3.1 Tailwind CSS .....	31
4.3.2 Toastify .....	32
4.3.3 Font .....	32
4.2.4 Backgrounds .....	33
4.2.5 Names .....	33
4.2.6 Homepage .....	34

4.2.7 React-howler .....	35
4.2.8 React Router .....	35
4.2.9 Page Header Component .....	37
4.2.10 Dashboard .....	38
4.2.11 Objectives.....	38
4.2.12 Your Stats .....	40
4.2.13 Campfire.....	42
4.2.14 Profile .....	43
4.3 Firebase .....	43
4.3.1 Setting up Firebase Project .....	44
4.3.2 Firebase Authentication.....	44
4.3.3 Firebase Storage.....	46
4.3.4 Firebase Realtime Database.....	47
4.4 Backend Development (Node JS).....	49
4.4.1 Creating the backend .....	50
4.4.2 Creating Routes/Endpoints .....	50
4.4.3 Middlewares .....	51
4.4.4 MyFitnessPal API.....	51
4.4.5 CORS.....	51
4.4.6 Axios .....	52
4.4. Conclusion.....	52
5. Testing and Evaluation .....	53
5.1. Introduction .....	53
5.2. Black Box Testing .....	53
5.4. Evaluation .....	54
5.5. Conclusion.....	55
6. Conclusions and Future Work.....	56
6.1. Introduction .....	56
6.2 Conclusions .....	56
6.2.1 Design Conclusion .....	56
6.2.2 Personal Conclusion.....	56
6.3 Future Work.....	57
6.4 GANTT Chart used.....	58
Bibliography .....	59

## Table of Figures

Figure 1 - Target values for physical activity <sup>1</sup> .....	8
Figure 2 - MyFitnessPal Web Application Food Section .....	11
Figure 3 - MoodPanda Web Application.....	12
Figure 4 - Remente Web Application.....	13
Figure 5 - Goalify Web Application .....	13
Figure 6 - Waterfall Model <sup>14</sup> .....	18
Figure 7 - Scrum Methodology <sup>15</sup> .....	19
Figure 8 - Low Fidelity Prototype Home Page(Not Logged In).....	20
Figure 9 - Low Fidelity Prototype(Logged In) .....	21
Figure 10 - Medium Fidelity Prototype Login Page.....	21
Figure 11 - Medium Fidelity Prototype Main Menu .....	22
Figure 12 - Medium Fidelity Prototype Goals Section .....	22
Figure 13 - Medium Fidelity Prototype Fitness & Food Section .....	23
Figure 14 - Medium Fidelity Prototype Mental Health Section .....	23
Figure 15 - Medium Fidelity Prototype Profile Section.....	24
Figure 16 - 1st Iteration Use Case Diagram .....	24
Figure 17 - 2nd Iteration Use Case Diagram .....	25
Figure 18 - 3rd Iteration Use Case Diagram .....	26
Figure 19 - Final Use Case Diagram.....	27
Figure 20 - 1st Iteration ERD .....	28
Figure 21 - 2nd Iteration ERD.....	28
Figure 22 - 3rd Iteration ERD .....	29
Figure 23 - Final ERD .....	29
Figure 24 - Toast.....	32
Figure 25 - Press Start 2P Font Example .....	33
Figure 26-Homepage Page.....	34
Figure 27 - Login Modal .....	34
Figure 28 - Register Modal.....	35
Figure 29 - Page Header.....	37
Figure 30 - Dashboard Page .....	38
Figure 31 - Objectives Page.....	39
Figure 32 - Objective modal .....	39
Figure 33 - Ternary operator used on the Objectives Page .....	40
Figure 34 - Your Stats .....	40
Figure 35 - Error message if no calories added.....	42
Figure 36 - Campfire Page.....	42
Figure 37 - Campfire Modal .....	43
Figure 38 - Profile Page .....	43
Figure 39 - Resetting password.....	45
Figure 40 - Empty Avatar and Default Display Name.....	46
Figure 41 - GANTT Chart .....	58

## 1. Introduction

### 1.1. Project Background

Life can be very hectic at times. Work, family and other commitments leave very limited time for yourself. Thus, only 31.3% of Irish people are highly active<sup>1</sup> and very few of them reach the minimum target values for physical activity recommended by the government which are as following:



Figure 1 - Target values for physical activity<sup>1</sup>

Furthermore, 1 in 4 Irish people will suffer from a mental health issue at some point in their life<sup>2</sup>. After a successful campaign done by TU Dublin (formerly DIT) between 2012 and 2016, which encouraged students to open up with their mental health issues, the number of students registering their mental health status rose by 700%<sup>2</sup>. The web application will hopefully help more people open up.

Due to increasingly busier lifestyles, the food habits of Irish people have also started changing for the worse. 17% of the evening meals are no longer the main meal of the day, with smaller meals and snacks changing the evening meal habits<sup>3</sup>. This clearly shows that people's lifestyles aren't as straightforward anymore. However, despite this, over 80% of consumers place a high level of importance on eating a balanced diet<sup>3</sup>. The web application will go hand in hand with the good intentions of the consumers by providing clear statistics of the user's nutrient intakes which will clearly show any deficiencies.

All these worrying trends seem to have the same origin, which is the busyness of adult life.

### 1.2. Project Description

The main goal of this project is to provide a fun and engaging web application that will help users quickly form healthy habits with limited impact on their daily schedules. The web



application will be called “Life Enhance” and it will combine all sorts of life planning tools in an entertaining way, which will make the web application feel more like a “game” rather than a chore.

The user will be able to track various aspects of their life such as mental health and physical health. The user will also be able to set up goals, which can be shared with their friends, thus both seeing each other’s progress for that specific goal.

The point of the web application is to be fun to use, hence its design will be similar to a game, where you are able to see your “level” and so on, as well as receive achievements. The web application will hopefully aid the formation of a habit in the user.

During the planning, development and testing process of the web application, an agile approach will be used. The agile methodology that will be used will be a modified version of SCRUM.

### 1.3. Project Aims and Objectives

The overall aim of the project is helping people form life improving habits in a fun and entertaining way. In order to achieve this aim, several milestones will be set, which will incorporate easily in the modified SCRUM approach that will be used. All the features of the web application will be divided into multiple 2 weeks sprints. This way concise milestones will be set every 2 weeks which will bring the project quickly back on track in case of any deviation.

The purpose of this project is helping people form their habits; therefore several features will be available on the web application, ranging from food tracker to a personal diary. The most important objective to keep in mind during the development of the project is the user’s entertainment. The features should be fun to use, the user should see going onto the web application as a form of fun rather than a chore.

Another objective of the project would be to teach people that habit forming does not always have to be a boring process and instead it can be invigorating, especially with the use of technology as it advances.

Life Enhance should be as the name implies, an enhancement to your life. Even minimally exploited, the web application, should still help the user in a little bit.

### 1.4. Project Scope

The scope for this project is simple, a self-improvement web application will be delivered by the end of April, as well as my dissertation, together with a YouTube demo. The interim report will also be delivered by the 4th of January.

### 1.5. Thesis Roadmap

**Literature Review** – This chapter explores background research done in the self-improvement field as well as any software that inspires the project.

**Experiment Design** – This chapter investigates the methodology I have chosen for the project, as well as diagrams of use-cases.

**Experiment Development** – This chapter looks into the development of the web application as well as any problems encountered during the process and any differences from the initial design.

**Testing and Evaluation** – This chapter describes how all the testing and evaluation was done.

**Conclusion and Future Work** – This chapter reviews the issues occurred during the entire duration of the project as well as any future work planned for the project.

## 2. Literature Review

### 2.1. Introduction

This chapter will explore background research done in the self-improvement field as well as any software that inspires the project or is similar to the envisioned web application. Examples of such software include MyFitnessPal and Mood Panda. The main focus was on web applications rather than mobile apps, thus self-improvement tools that do not have a web application were ignored, such as Strava and Google Fit, to name a few. Several technologies were also researched to be used in the creation of Life Enhance. The technologies are all modern and new. Research on software evaluation tools was also conducted, with a big focus on Nielsen's Heuristics. The reason for this is because the design of the web application will be detrimental to keeping the user engaged. Finally, 2 previous final year projects were reviewed.

### 2.2. Alternative Existing Solutions to Your Problem

#### MyFitnessPal

MyFitnessPal<sup>4</sup> is a smartphone app, as well as a website which tracks diet and exercise. During the account registration, users are asked what their fitness goal is, as well as several fitness related questions such as their weight, height, and gender. The users are then able to search a huge database of over 300,000,000 food items<sup>5</sup>, each with their own calorie count to track their diet and calories.

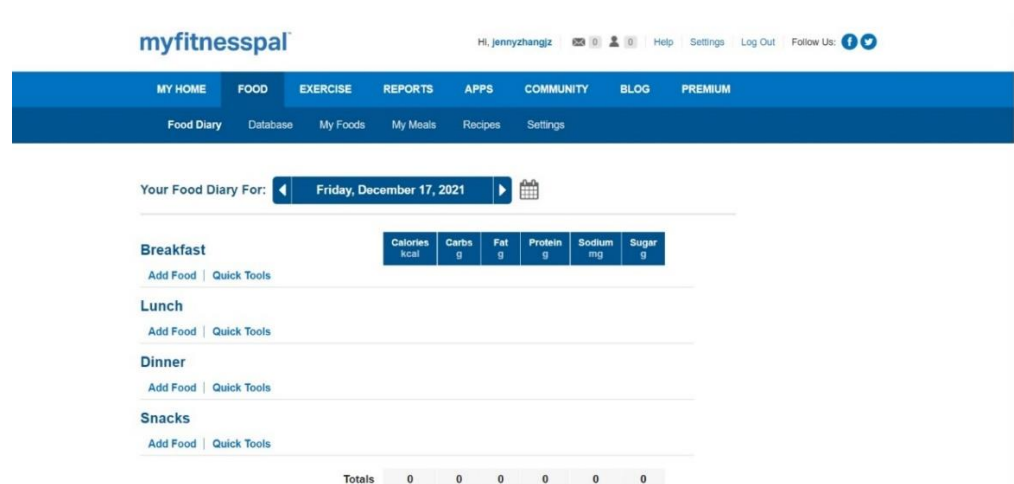


Figure 2 - MyFitnessPal Web Application Food Section

They are also able to track their exercises, with the ability of adding notes to today's exercise. The exercises are also pulled from a database; thus, the users can enter the amount of minutes/sets/reps done for a specific exercise and the calories burned are automatically added in.

The most useful thing about this application is the huge databases they have. This ensures a very smooth experience for the clients as they can find whatever food they want, no matter where they are from in the world. Unfortunately, the database is custom made and user

validated, so the accuracy of the calories can vary hugely. For example, there can be 5 “apple” values in the database, each with a different number of calories.

However, while the large database is advantageous, a disadvantage of this application would be that it does not allow the user to do anything else other than track food and exercise and also filling it everyday tends to feel like a chore. There do not seem to be any motivating elements on the application.

## MoodPanda

Moodpanda<sup>6</sup> is a very simple mobile and web application where the user is able to update his mood from a scale to 1 to 10, as well as adding an optional note giving a reason for the mood. The application is almost like a social network. Other users can “send hugs” to your status, which is like giving a “like” and comment/reply to your status.

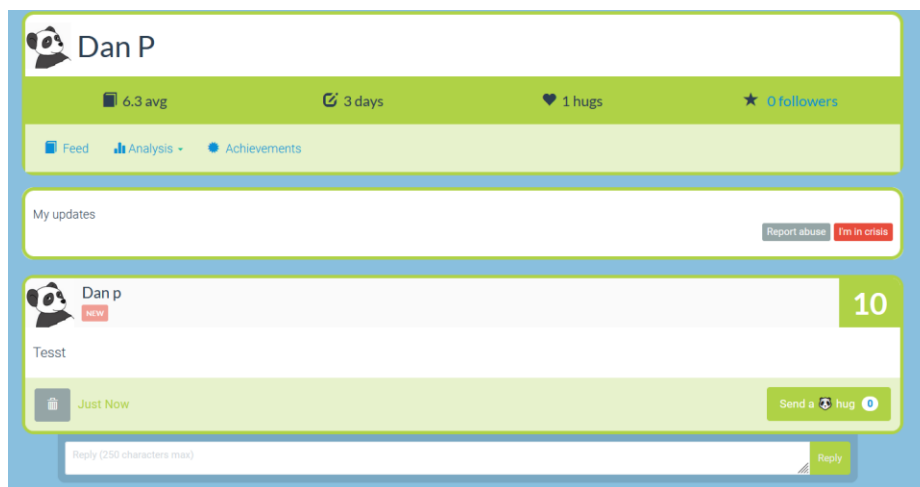


Figure 3 - MoodPanda Web Application

On the downside, this application is unmoderated. It is possible comment on anyone’s posts and also post anything you want. This can be a problem, especially since most people using a mood tracker application might not feel at their best and be very sensitive to negative remarks.

However, the application feels very personal and the community aspect is definitely a new touch in the domain of mood tracking apps. If the users see that there are also other people struggling, they can be motivated to keep using the app and track their moods.

## Remente

Remente<sup>7</sup> is a mobile and web application that provides a mental health tracker as well as a self care journal that helps with goal setting and achieving personal growth. Users can plan their day, track their goals, make a life assessment and set their “Life Balance”.

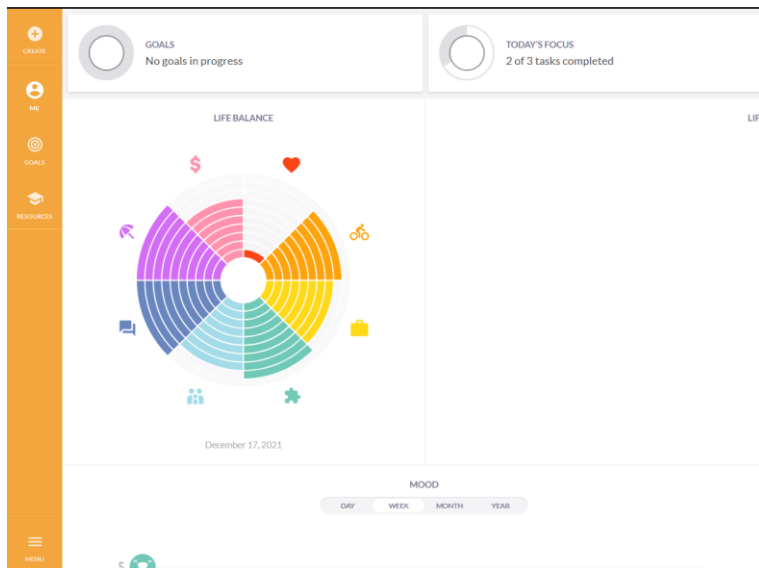


Figure 4 - Remente Web Application

This application is well done in terms of design and colours. It also provides a gamification aspect to it, as you are able to change your “Life Balance”. Life Balance does not provide much in terms of functionality. It simply allows the user to set how happy they are with a certain aspect of their life. For example, in Figure 4, the user has only 1 bar for the “love” icon, which means their romantic life might not be going great. The application is very fun to use and entertaining. However, it is not very intuitive and no tutorial has been presented at the start. The user interactions are a bit odd and it is a bit hard to get a grasp of how it works. A lot of users might feel overwhelmed by the amount of functionality and quit the application straight away.

## Goalify

Goalify<sup>8</sup> is a mobile application and web application which keeps tracks of your goals and habits. You can manage repeating tasks as well as to-dos. Furthermore, you can set up challenges with your friends and compare progress, as well as motivate each other using an integrated chat feature.

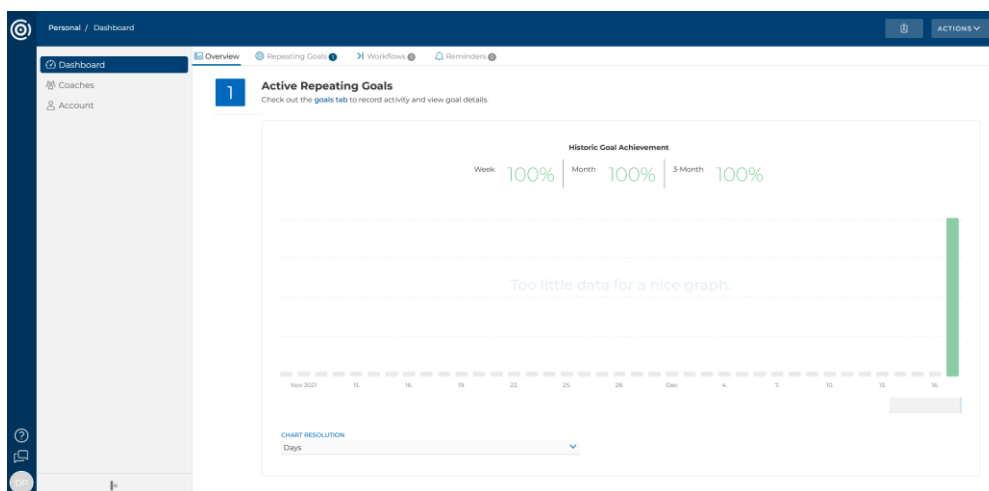


Figure 5 - Goalify Web Application

While the application has a really good potential, the UI is not engaging enough. It looks very formal and boring, and it is not intuitive. Its major potential comes from the capacity to share the goals between friends and track each other's progress. This keeps the users motivated in their self improvement journey.

## 2.3. Technologies you've researched

### 1. HyperText Markup Language(HTML)

Because Life Enhance will be a web application, HTML must be used. HTML defines the structure of the web page by using markup to annotate text, images and other content to display in the browser.

### 2. Cascading Style Sheets(CSS)

This is a stylesheet language that describes how elements should be rendered in the browser and the layout of the web page. This will be used to design the application's UI, in conjunction with other technologies.

### 3. JavaScript

This is used to program the behaviour of web pages. Over 97% of websites use JavaScript on the client side<sup>9</sup>. JavaScript not only allows the implementation of complex features on a web page, but it can be integrated with many libraries such as React to develop various functionalities of the web page, such as graphics and content updates.

### 4. React

This is a frontend JavaScript Library for building user interaction features. React allows to build complex UIs from small and isolated pieces of code called components<sup>10</sup>. The components have a render method that takes input data and returns what to display, as well as maintaining internal state data.<sup>11</sup> This library could be very useful in creating Life Enhance, due to its efficiency. For example, if React was not to be used, then the DOM tree would have to be changed accordingly to the application state. However, React does it automatically, which in turn creates a cleaner code.

### 5. Node JS

This is an open source backend JavaScript environment which executes JavaScript code outside the web browser. The advantage of Node JS when compared to other backend frameworks, such as PHP, is that it eliminates the waiting time between requests. Node JS will be appropriate for the development of Life Enhance, as there will be many API requests. Another reason for the usage of Node JS would be that it has a JavaScript syntax, which will go hand in hand with the frontend React.

### 6. PostgreSQL

This is an open-source relational database management system. It has earned its reputation for its architecture, reliability, data integrity and extensibility<sup>12</sup>. Because Life Enhance will store many user data, such as the user's profile and goals, then a relational database management system is a must.

### 7. pgAdmin

This is an open-source administration and development platform for PostgreSQL. This could be useful if PostgreSQL will be the decided database because it is the de facto GUI tool for it.

## 8. AWS (Amazon Web Services)

This technology provides on-demand cloud computing platforms on a metered pay as you go basis. This might be used if Life Enhance will be published online, rather than just locally on the machine. It is compatible with every technology listed above.

## 9. Google Firebase

This is a platform developed by Google which provides multiple tools and technologies for creating mobile and web applications. For Life Enhance, Firebase can be used as an alternative backend which provides a database and authentication system. It also offers real time databases, which means any change that happens on the server-side database is sent to all the users. Firebase is also known for rapid development as well, which might prove advantageous.

## 2.4. Other Research done

The success of this project is linked to the quality of the application's UI. This quality must be evaluated by using a set of appropriate metrics. Extra research was carried out to determine these metrics. The outcome of this research was Nielsen's Heuristics that will be used to evaluate Life Enhance's usability and engagement. The reason for that is because the UI will be very important during the development of Life Enhance to ensure it keeps the users engaged. Nielsen's Heuristics will help with that aspect because each heuristic is about making the application more presentable.

### 1. Nielsen's Heuristics

The main feature of Life Enhance will be its fun and game-like design. A heuristic evaluation is a usability inspection method for software that helps identify usability problems in the UI<sup>13</sup>. The 10 principles of Nielsen's Heuristics are as follows:

**Visibility of System Status:** The design should keep users informed about what is going on at all times.

**Match between System and the real world:** The design should not use any jargon, it should be easy to understand by the user.

**User Control and Freedom:** The design should provide "exits" at all points in case of an user related mistake.

**Consistency and Standards:** The design should be consistent, for example users should not have to wonder if two different words mean the same thing(i.e.: "Sign-up" and "Register", it should either be "Sign-up" or "Register" everywhere).

**Error Prevention:** The design should have elements to prevent error, for example "Re-enter Password" field.

**Recognition rather than Recall:** The design should not make the user remember elements from a previous interaction. For example, a menu should be easily navigable, so the user can simply go back and check the previous menu page again.

**Flexibility and efficiency of use:** The design should also cater to expert users, for example having a quick home button if you press the logo of the website. This will be very important when creating Life Enhance, because part of keeping the web application fun and entertaining are short interactions. The user should not have to spend too long filling in the application every day, otherwise it might get boring.

**Aesthetic and minimalist design:** The design should only have elements related to the UI.

**Help users recognise, diagnose and recover from errors:** The design should display the error messages in normal language and also give a suggestion on how to fix the error.

**Help and Documentation:** Documentation should be provided on how to use the system. For Life Enhance, a tutorial will be presented to the user upon his first login, and they will also be able to repeat the tutorial.

## 2. **National Physical Activity Plan for Ireland**

This document outlines the physical activity plan for Ireland and contains figures such as recommended amount of exercise. These figures can be used during the development of the web application in order to give government approved suggestions.

## 3. **Azure DevOps**

This is a Microsoft product which offers a platform for teams to collaborate on, ranging from testing, developing, and deploying. It is a tool used for agile methodology, thus it could be suitable for the development of Life Enhance, as the approached methodology will be a modified version of SCRUM.

## 4. **Trello**

This is a web-based application where users can create task boards, for example a “To do”, “In Progress”, “Done” and so on. This can be used as an alternative to Azure DevOps as it suits more an individual project, whereas Azure DevOps serves a purpose for multiple teams usually.

## 2.5. Existing Final Year Projects

This section will review some final year projects which are somehow similar to Life Enhance. There weren't any life improvements web applications, therefore the reviewed projects were of just two web applications, however the relevancy lies in the same use of technologies.

### 1. **Lisgrey House Restaurant Web Application – Matthew Magee**

The project consists of a responsive web application that implements a takeaway system and a menu filtering system while providing an intuitive interface. The complexity of this project comes from the fact that an external client will be involved. It is not trying to construct a stand-alone restaurant application that any restaurant could use, it must meet the needs and wants of the client.

The technologies used for the completion of this project were Django and Bootstrap as well as PostgreSQL for the database.

The weakness of this project was the implementation of the PWA as well as the email and phone number verification, however if the PWA issue is solved, then the user accounts will be able to have a verification step implemented. The PWA problem was basically that if a user tries to login or logout, they will be redirected to a cached page that will still think they are logged in/out. Because of this, user accounts had to be removed to keep the PWA functionality.



Other than these issues, the project was a major success and plans for future work have already been created by Matthew.

## **2. Park Manager: System for managing the operation of an amusement park - Bogdan Babiy**

The project allows customers to book into an attraction at a particular time while also being able to see how many other customers have booked in for the same time slot. Bogdan stated that the most complex part was implementing an array of components that utilize different programming languages and frameworks and creating a system that allows for easy communication between all of the components. The technologies used in this project were Node JS, Quasar JS and Mongo DB for the database. The weakness of the project was that there were quite a lot of bugs found during the Black Box Testing Phase. Another weakness of the project was that this entire idea is a bit unrealistic. The experience of the amusement park is to wonder around and just go on whatever ride you want whenever you want. If you must book each ride, it seems a bit unentertaining. A good strength was that the UI is very clean and simple, without the need of a user manual. Furthermore, scalability is huge in this project, and the framework used allows the application to be expanded to support virtually any device and platform such as desktop and mobile.

## **2.6. Conclusions**

After this research was conducted, the design process of the web application can start. The application's main features have been established, as well as the technologies that will be used during the development of it, assuming everything goes according to plan. The UI will be based of Nielsen's Heuristics.

### 3. System Design

#### 3.1. Introduction

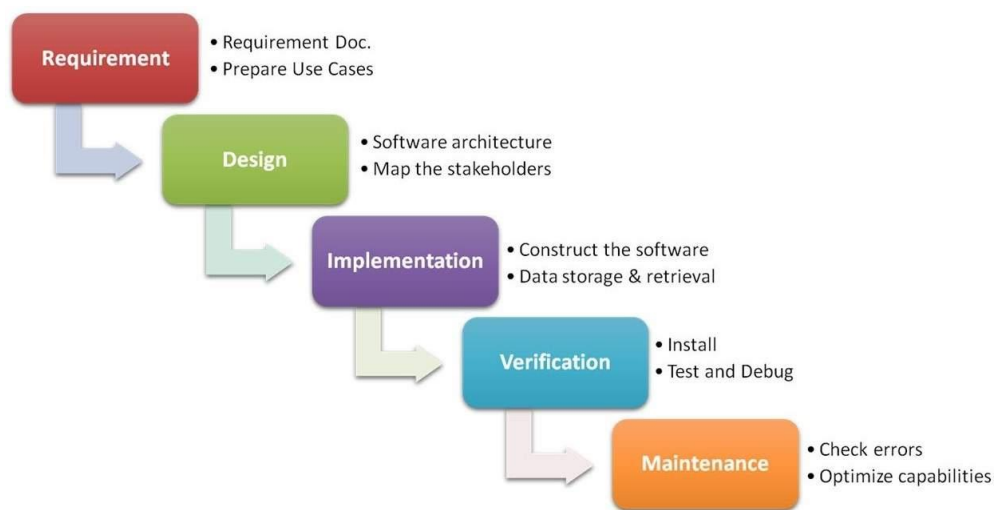
After the background research was completed, it is time to design some of the information researched. This chapter will discuss a number of software development methodologies as well as the chosen one, following on with an overview of the system, as well as some Use Case Diagrams and Entity Relationship Diagrams.

#### 3.2. Software Methodology

This section will go over a number of software development methodologies available as well as the chosen software methodology.

##### 1. Waterfall

This is the oldest methodology available. It has 5 stages which are as following:



*Figure 6 - Waterfall Model<sup>14</sup>*

This methodology is very simple and easy to understand. However, the problem with this is that a project will never go exactly according to plan. For example, the initial customer requirements might change, or perhaps during the implementation stage the Design might also need to be modified. Therefore, the waterfall model is an unrealistic approach to software development.

##### 2. Scrum

This is an agile development methodology. In order to understand Scrum, we need to understand what agile means. Agile is a methodology which is very similar to the waterfall methodology. However, instead of doing each stage for the entire product, you just do a few features at a time. In this way, if something changes, you won't waste months of development, instead you will just be wasting two weeks at most as corrections can be made during the development phase.

SCRUM is exactly like agile, however the work is divided into sprints, usually 2 week long sprints. At the end of each sprint, you should have a feature fully developed,

tested and deployed.

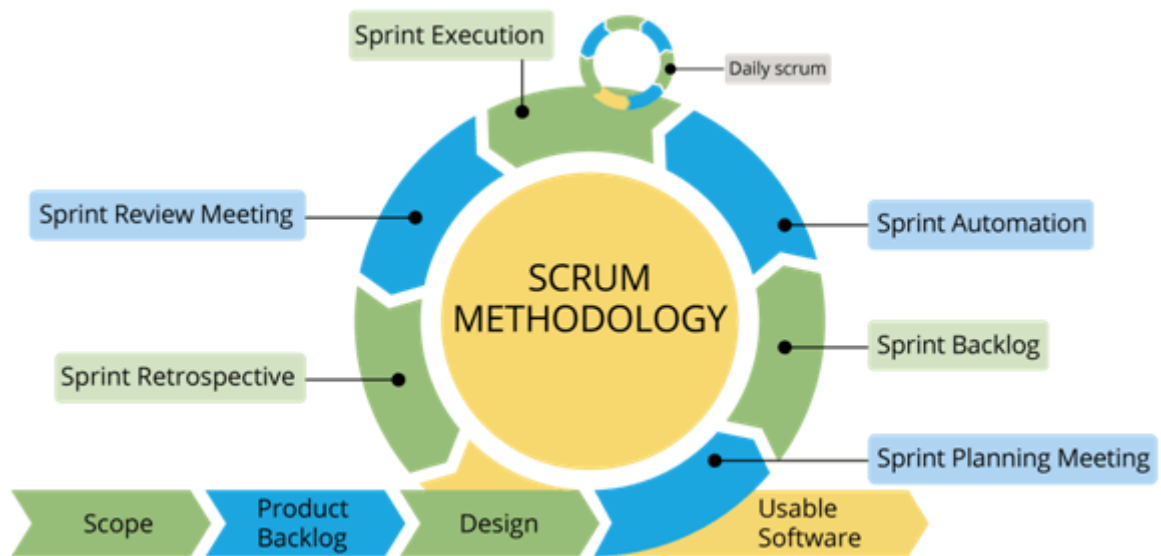


Figure 7 - Scrum Methodology<sup>15</sup>

The software methodology that was used in the creation of Life Enhance is a modified version of SCRUM. The reason why it was a modified version is because SCRUM is a team-oriented approach, whereas for this project there will only be 1 creator. But the main approach will be similar.

### 3.3. Overview of System

In addition to the modified version of SCRUM which will be used throughout the development of the project, a feature driven-development approach will also be utilised. For each sprint, a feature will be fully developed and tested. This works perfectly for Life Enhance, because the entire web application consists of a set of features (e.g.: Goals). The feature-driven development will be used in conjunction with the modified version of SCRUM, which means all features will have a priority as well as an Effort. The highest priority features will be completed first (e.g.: Login).

In terms of the technical architecture, a 3-tier architecture will be used. A 3-tier architecture consists of 3 layers: the Presentation Layer, the Application Layer and the Data Layer. There are a lot of advantages when using a 3-tier architecture system. For example, you can change 1 tier without affecting the others, since they are kept separate. This, in turn, makes the code much cleaner, as everything is kept separate. By using a 3-tier architecture, you are future-proofing your app as changes can be applied without affecting the other tiers. For Life Enhance, the following architecture will be implemented:

- Presentation Layer: ReactJS
- Application Layer: NodeJS
- Database Layer: PostgreSQL

Firebase will also be kept in the back of the head in case of issues with the current architecture.

### 3.3.1 Changes to Overview of System during development

During the development of Life Enhance, the system architecture was changed slightly. The software methodology was kept the same, however Firebase was used as the database instead of PostgreSQL. The reason behind this choice was that Firebase was used for the authentication and storage, therefore having the database be Firebase based as well would be a smart choice since integration would be easier.

## 3.4 Presentation Layer

The presentation Layer deals with the graphical user interface. Low Fidelity and Medium Fidelity prototypes were created to design the Presentation Layer, as well as a Use Case diagram.

### Low Fidelity Prototypes

The Low Fidelity prototypes were drawn on paper, and family, friends as well as the supervisor were asked their opinion on it.

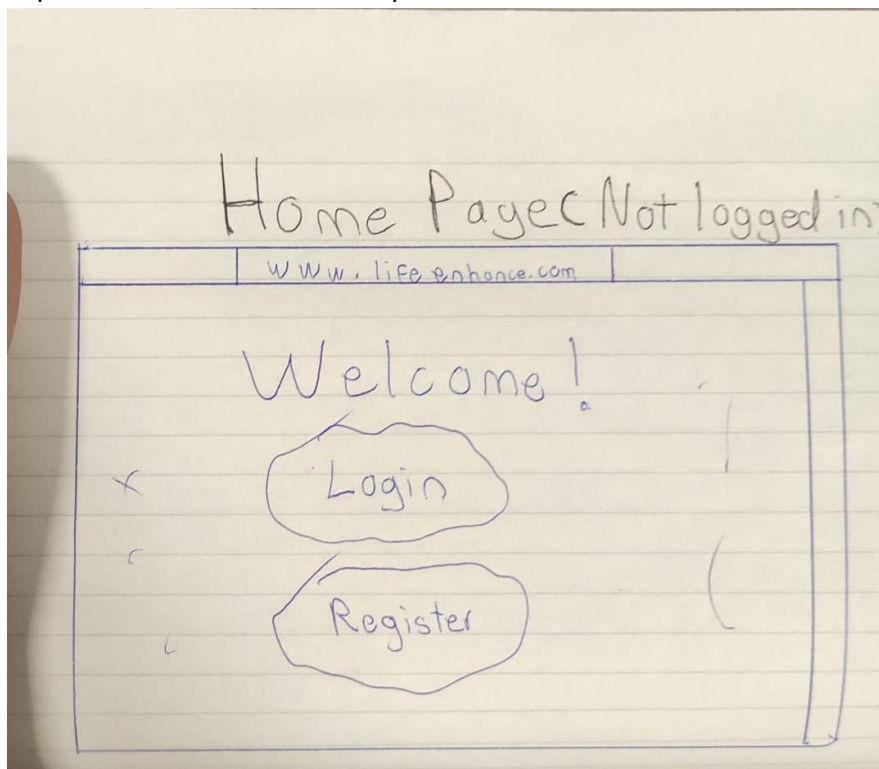


Figure 8 - Low Fidelity Prototype Home Page (Not Logged In)

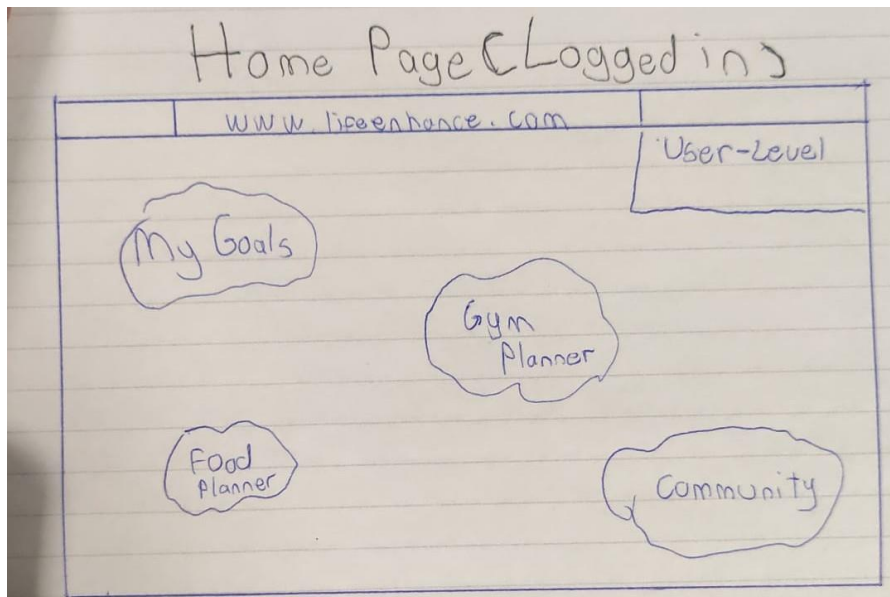


Figure 9 - Low Fidelity Prototype(Logged In)

The low fidelity prototypes were created with the idea that the application should be fun to use as well as having gamification elements, hence why this design as well as the “User Level” label in the corner.

### Medium Fidelity Prototype

Following the background research, Medium Fidelity prototypes were created using a web application called draw.io<sup>17</sup>.

Login Page:

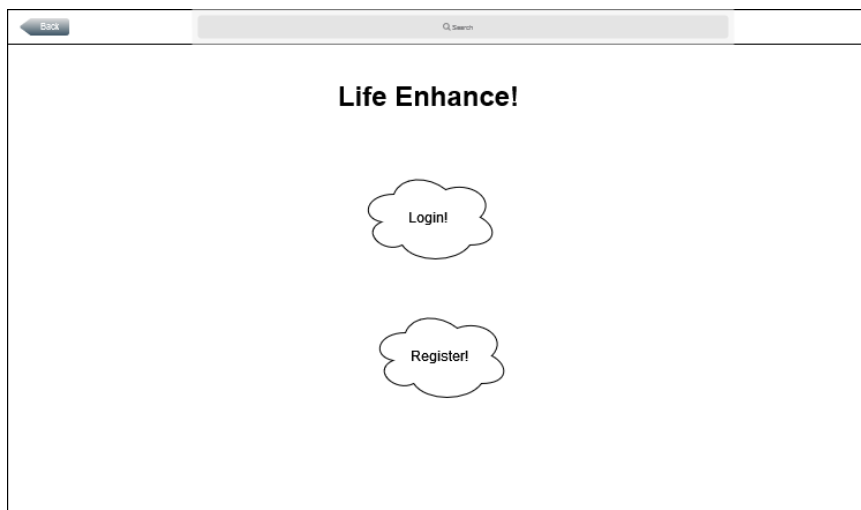


Figure 10 - Medium Fidelity Prototype Login Page

## Main Menu:

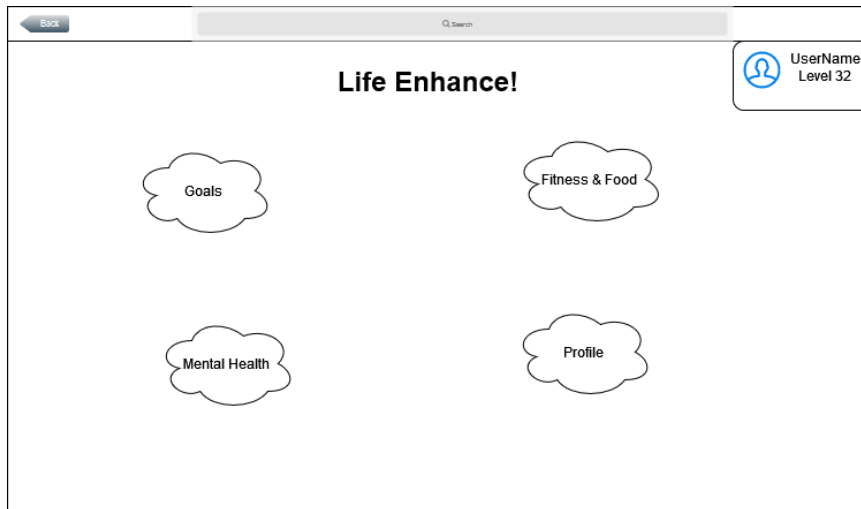


Figure 11 - Medium Fidelity Prototype Main Menu

## Goals Section:

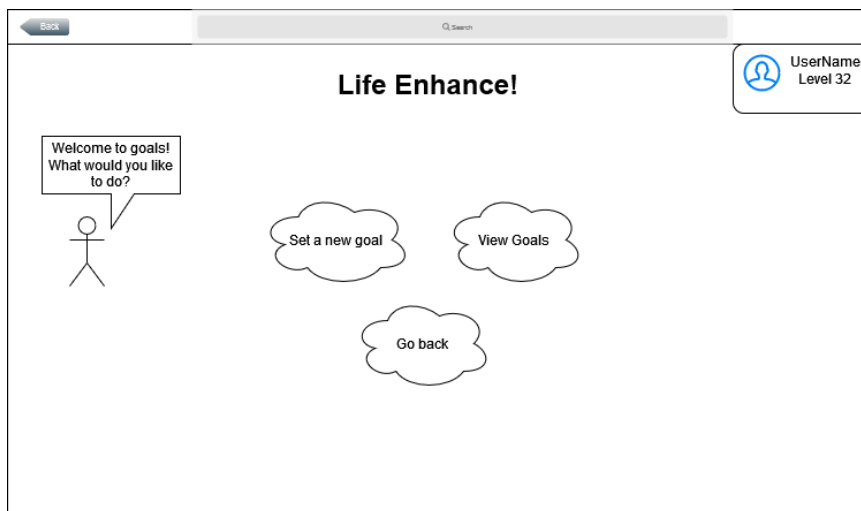
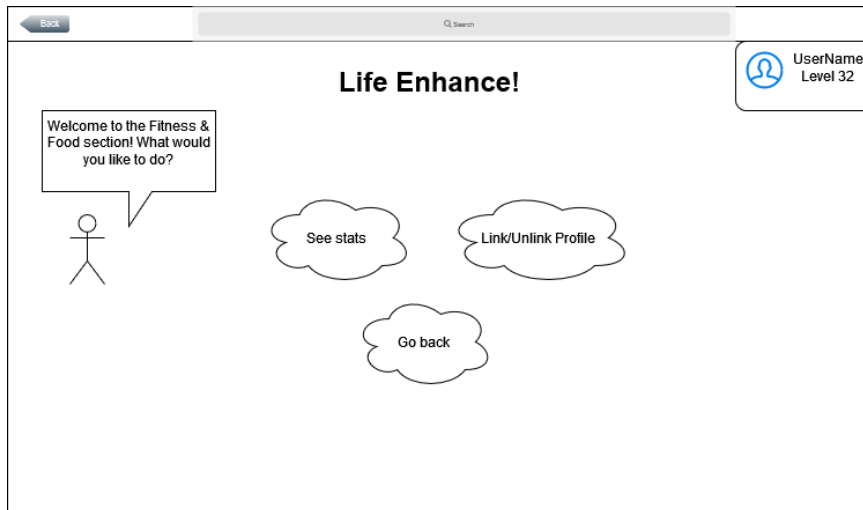


Figure 12 - Medium Fidelity Prototype Goals Section

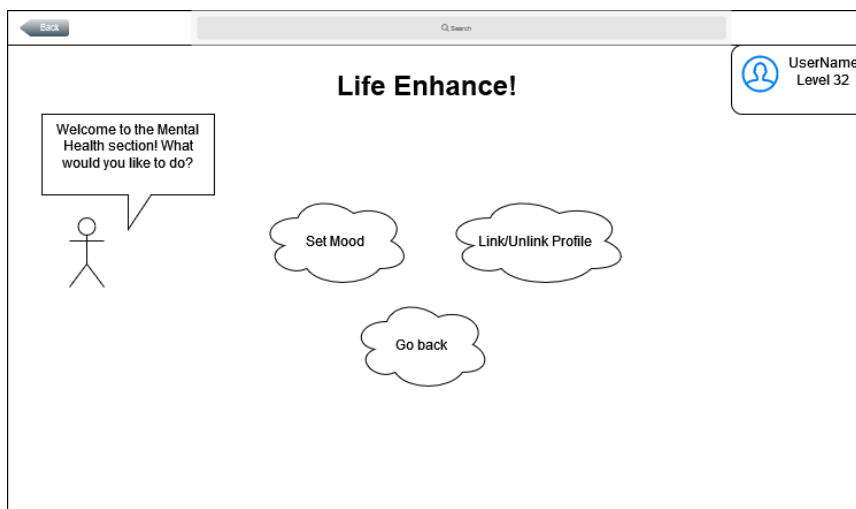
## Fitness & Food Section:



*Figure 13 - Medium Fidelity Prototype Fitness & Food Section*

MyFitnessPal API will be used during the development process if all things go according to plan, hence the "Link/Unlink Profile".

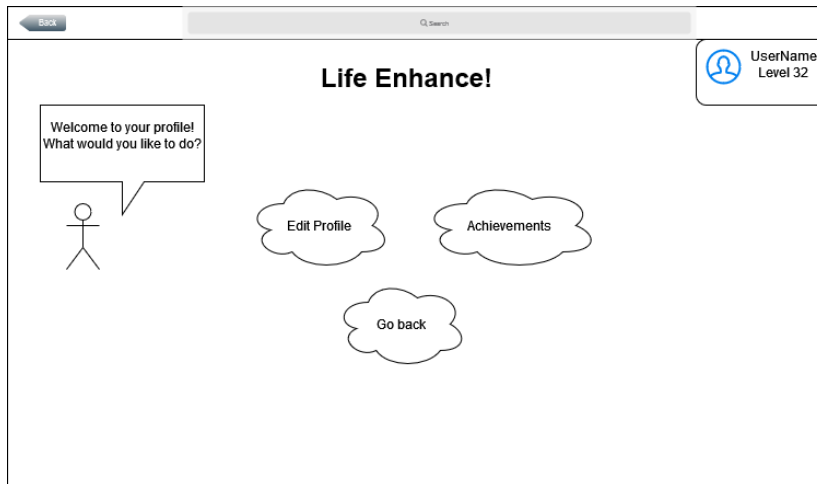
## Mental Health Section:



*Figure 14 - Medium Fidelity Prototype Mental Health Section*

Similarly, as above, an API will try to be used, however the screen might look different if no suitable API is found for this.

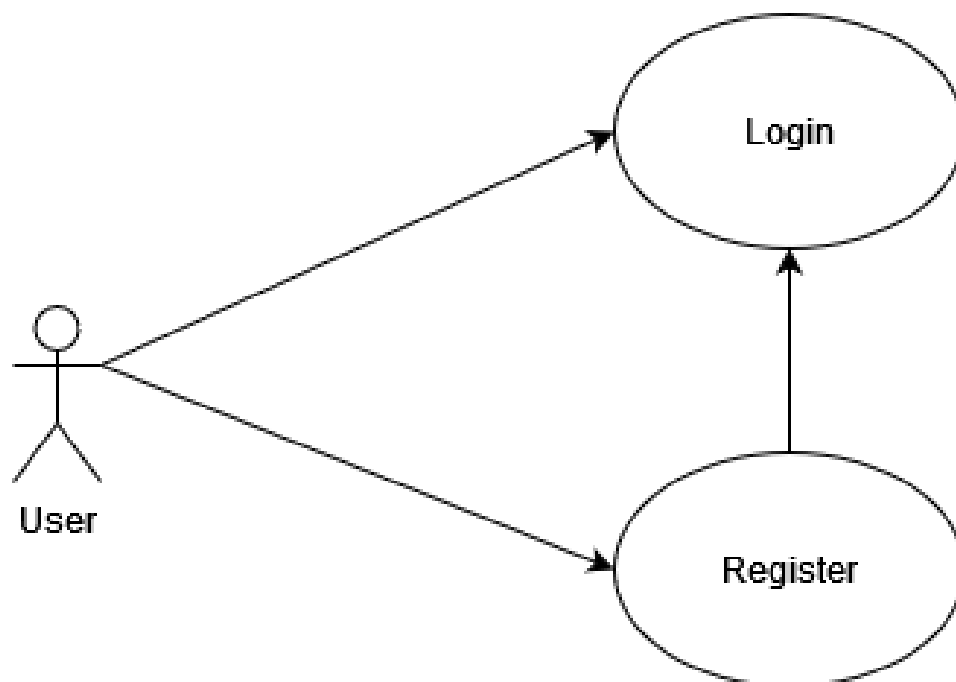
## Profile Section:



*Figure 15 - Medium Fidelity Prototype Profile Section*

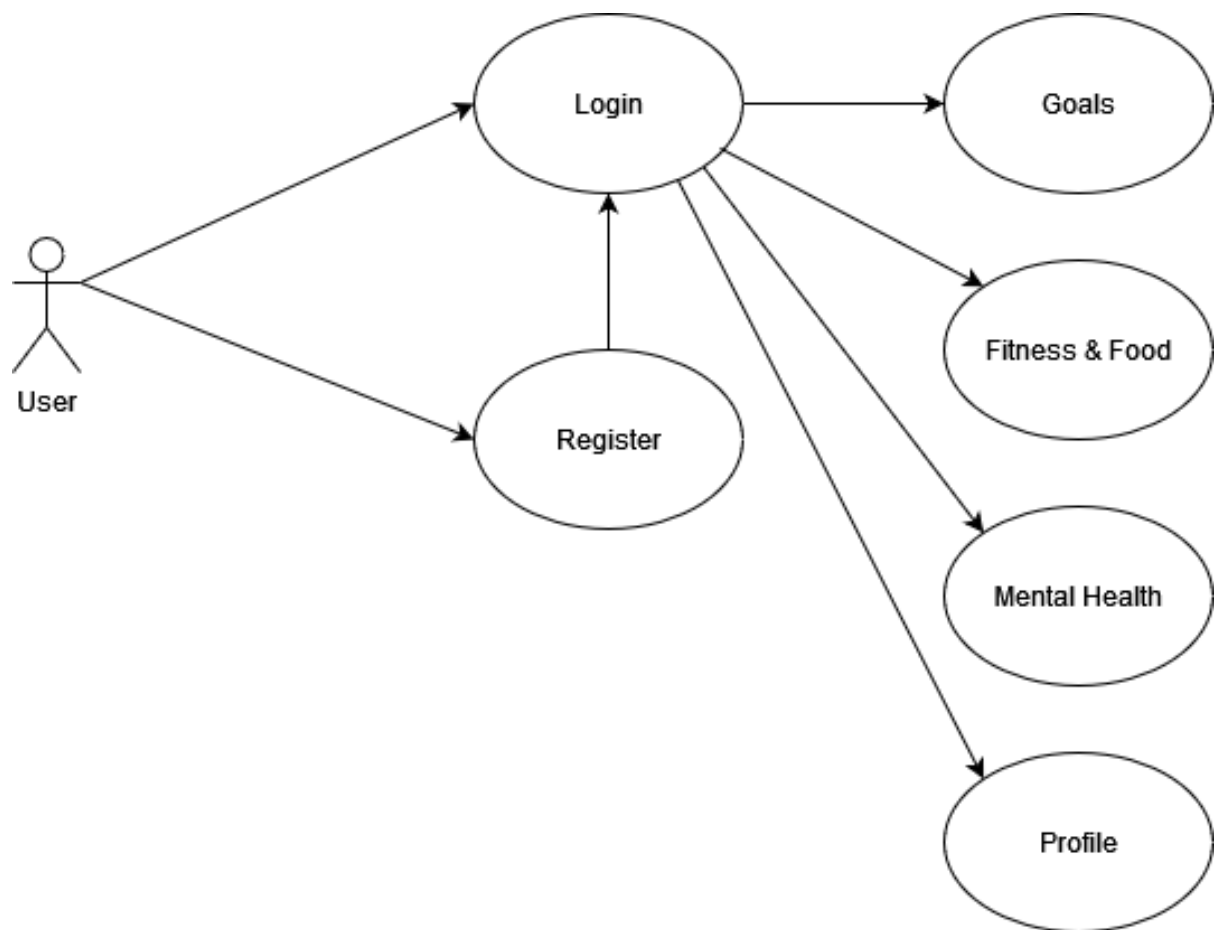
It is worth noting that the design might change quite a lot during the development process, however what is definite is that the game-like design will be there.

## Use Case Diagrams:



*Figure 16 - 1st Iteration Use Case Diagram*





*Figure 17 - 2nd Iteration Use Case Diagram*



Figure 18 - 3rd Iteration Use Case Diagram

### 3.4.1 Changes to presentation layer due to development

As stated in chapter 3.4, changes did happen to the presentation layer. An up-to-date Use Case Diagram can be observed in the figure below.

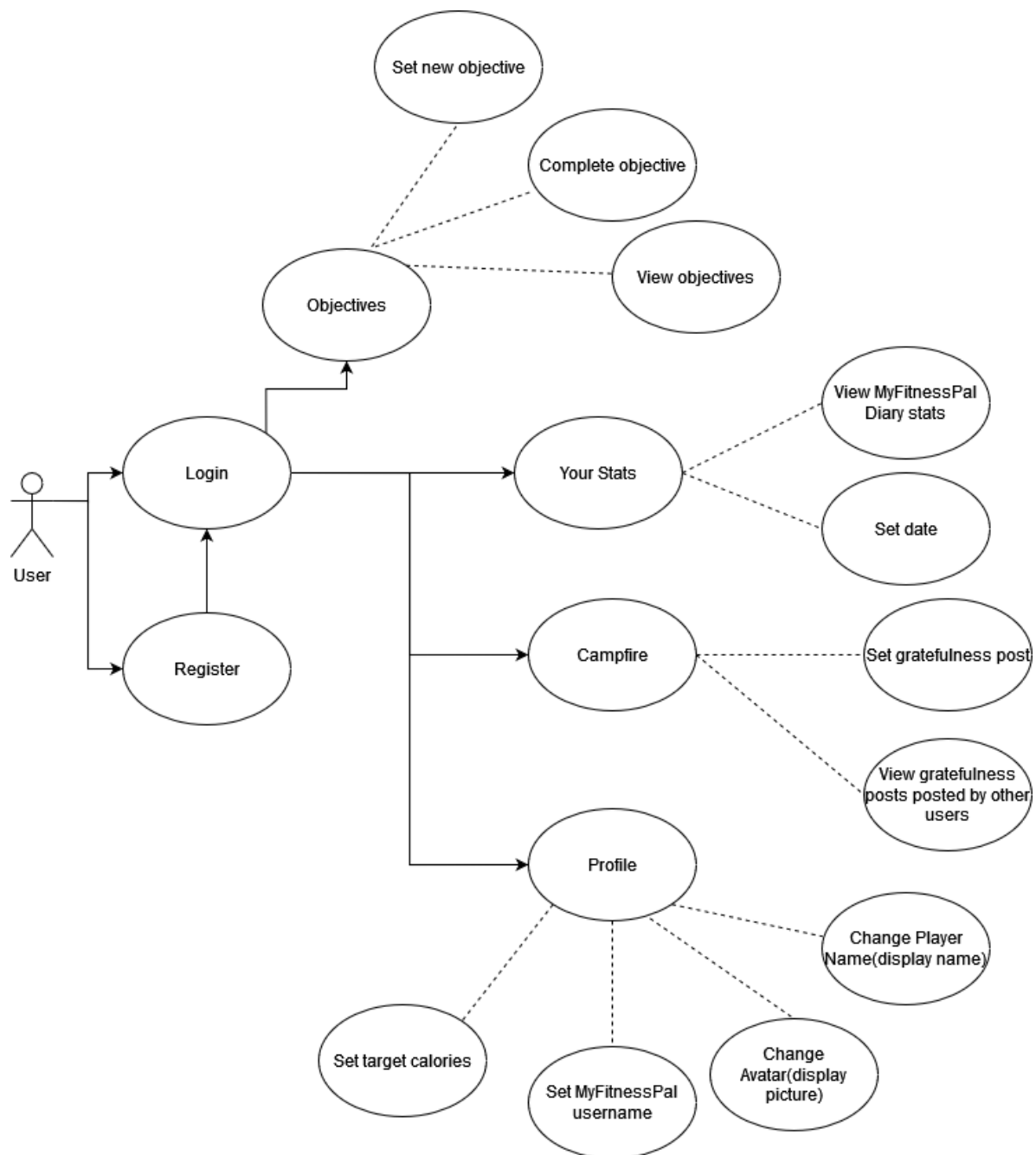


Figure 19 - Final Use Case Diagram

### 3.5 Application Layer

This layer controls the application's functionality. For example, if the user enters his username and password in the Login Screen (Presentation Layer), the presentation layer sends the user input to the application layer which in turn will send that info to the database layer in the form of a query. Then, the database layer will return the results back to the application layer, which converts it and sends it back to the presentation layer to

display it on the user's screen. Hence why the application layer is also called the "Middle Layer", it acts like the middleman.

### 3.6 Database Layer

This layer is simply the database of the web application. It is a place where data, such as the user profile will be stored. A simple Entity Relationship Diagram has been created in order to visualise the design. However, the database layer will drastically change during the development process as fields might be added or removed.

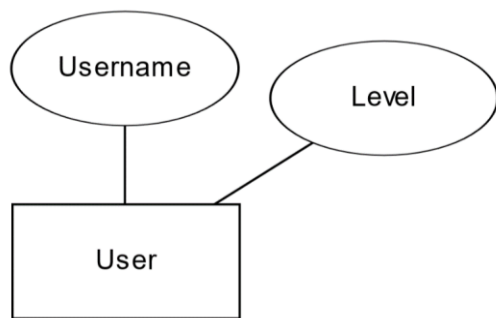


Figure 20 - 1st Iteration ERD

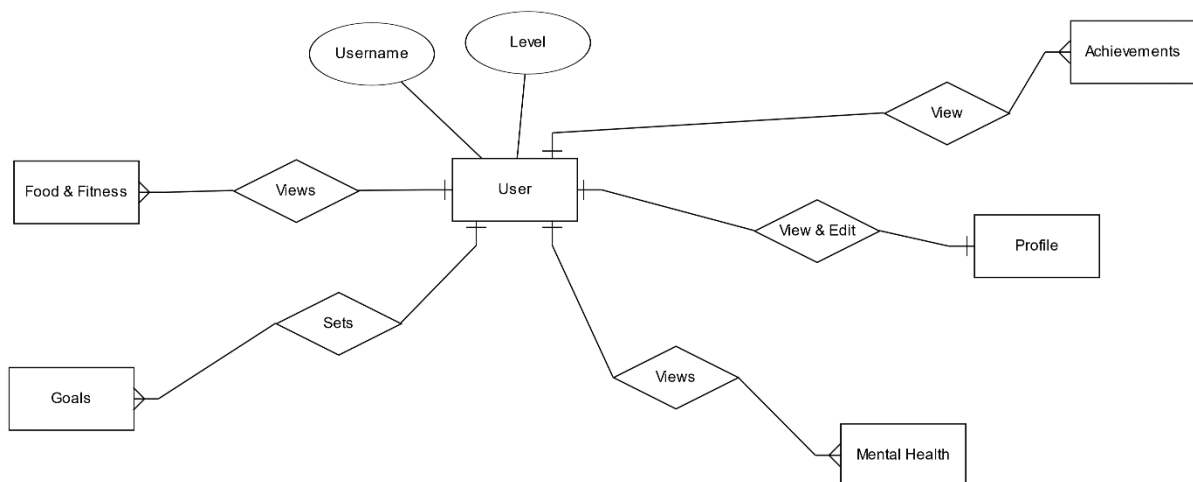


Figure 21 - 2nd Iteration ERD

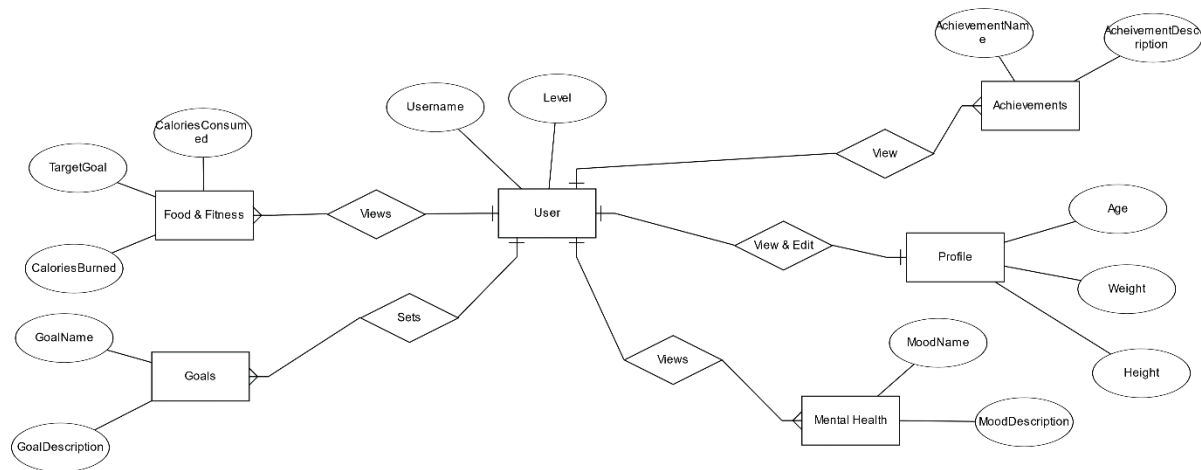


Figure 22 - 3rd Iteration ERD

### 3.6.1 Changes to Database Layer due to development

As mentioned earlier in chapter 3.3.1, PostgreSQL was not used as the database of choice. Firebase Realtime Database was opted for instead. Firebase Authentication is used to authenticate the user onto the application. It automatically holds the following fields:

- Uid
- Email
- Password
- Display Name
- PhotoURL(which can be used to display their profile picture)

This reduced the amount of information stored on Firebase Realtime Database, since it was incorporated in Firebase Authentication. Below is an Entity Relationship Diagram which displays the final schema.

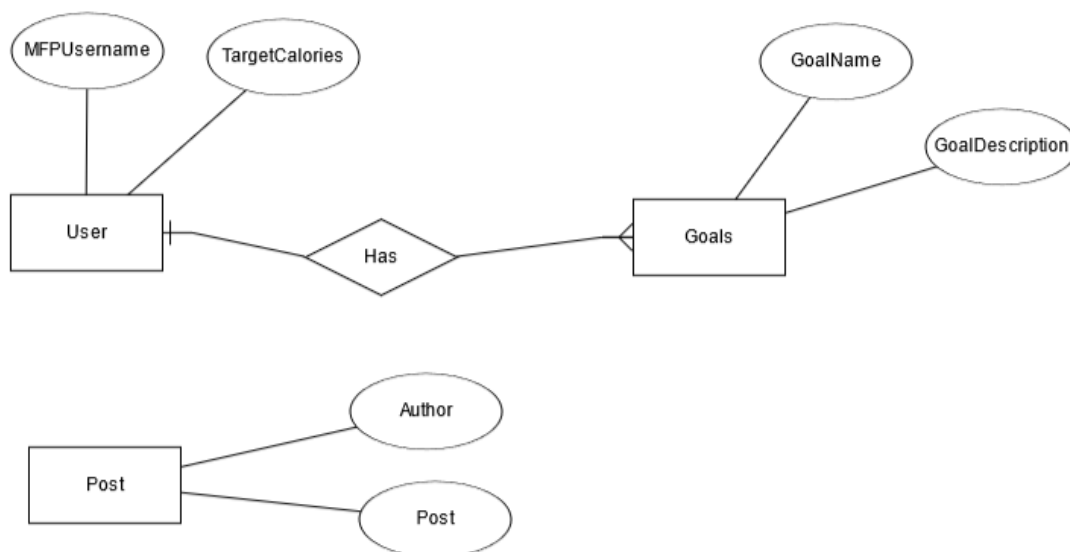


Figure 23 - Final ERD

This is much smaller than the initial ERD, but that is due to Firebase Authentication holding multiple other values.

### 3.7. Conclusions

This chapter discussed the design of the system, ranging from the software methodology all the way to the technological architecture used. Low Fidelity prototypes and Medium Fidelity prototypes have been created, as well as Use Case Diagrams, which will help during the development of the presentation layer. Entity Relationship Diagrams have also been created to support the development of the database layer.

The changes that occurred during development have also been discussed. A new Use Case Diagram has been presented as well as a new Entity Relationship Diagram, which best represent the final version of the system.

## 4. Experiment Development

### 4.1. Introduction

This chapter involves the development of the web application Life Enhance based on the system design created in chapter 3. Throughout its development, many of the core concepts and technologies changed from the initial design. All these changes will be discussed as well as justified.

### 4.2. Creating the React App

Node and NPM were installed to create the React application. Upon installing them, the following commands had to be executed

```
npx create-react-app life-enhance
```

```
Cd life-enhance
```

```
NPM start
```

### 4.3. Front End Development(React)

In this chapter the frontend development will be discussed, specifically all the technologies used in order to create the front end as well as any changes from the initial design.

#### 4.3.1 Tailwind CSS

This is a CSS framework for building custom UIs. It is highly customisable and quite simple to use with a very detailed documentation.

To install Tailwind CSS, the following commands had to be executed in the web application terminal:

```
NPM install -D tailwindcss
```

```
npx tailwindcss init
```

Afterwards, the template paths had to be configured. This was done by adding this to the tailwind.config.js file.

```
module.exports = {  
  content: [  
    './src/**/*.{js,jsx,ts,tsx}',  
  ],  
  plugins: [],  
}
```


Then, inside index.css, the following code was added:

```
@tailwind base;  
@tailwind components;  
@tailwind utilities;
```

To use Tailwind CSS, you specify the CSS inside the JSX using className, which is almost like inline CSS but made easier. Below is an example of Tailwind in action:

```
<h1 className="mt-4 text-2xl ml-2">What date would you like to see?</h1>
```

This displays the following:



**What date would you like to see?**

The mt-4 code stands for margin top 4 (which is Tailwind's custom size), text-2xl is the size of the text and ml-2 stands for margin left 2. Every single element in Life Enhance has been customised using Tailwind CSS.

#### 4.3.2 Toastify

All the error messages on the web application were displayed using toasts. Toasts are pop-up notifications which appear on the page, usually in the corner. Below is an example of a toast.

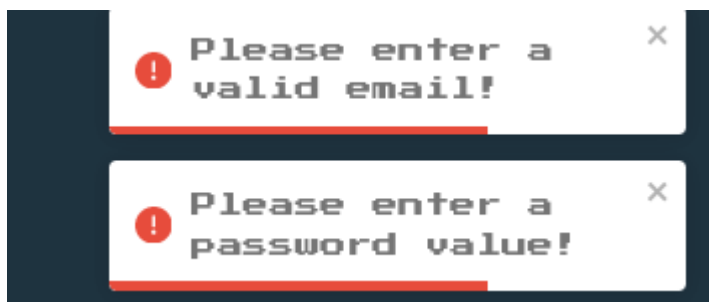


Figure 24 - Toast

These toasts were displayed using a library called "React-toastify", which was installed using NPM. To display a toast, you must call a method and ensure you have a toast container component in your JSX code.

Below is a snippet of code which demonstrates how to display a toast:

```
if(error.code === "auth/user-not-found") {  
  toast.error("No user found with that email");  
}
```

JSX:

```
<ToastContainer/>
```

It is also worth noting that toasts are used to indicate success on the web application, for example when the user updates his profile successfully, a message appears.

#### 4.3.3 Font

The font "Press Start 2P" was used to create a gamification element to the entire web application. This is a bitmap font which is based on 1980s Namco arcade games<sup>16</sup>. This font was installed by executing the following command inside the project's terminal:

***NPM install @fontsource/press-start-2p***

Then, in order to apply it to everything, the following code was added to App.css:



```

* {
  font-family: "Press Start 2P";
}

::placeholder{
  font-family: "Press Start 2P";
}

input[type="text"]
{
  font-family: "Press Start 2P";
}

```

The \* means that the font will be applied to absolutely every single element of the application. However, the \* did not work for placeholder and input text, therefore it had to be specified manually.

The figure below shows snippet of how the font looks like in Life Enhance:

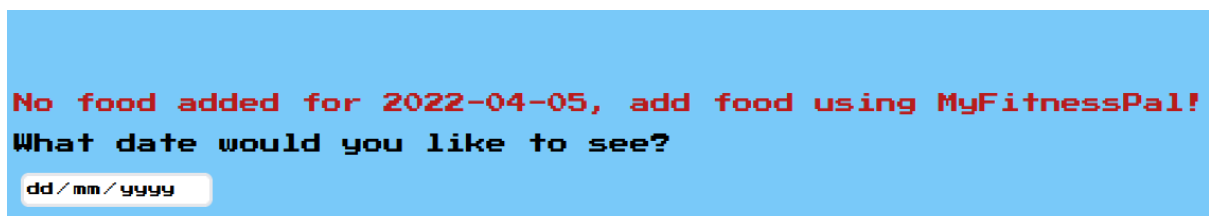


Figure 25 - Press Start 2P Font Example

#### 4.2.4 Backgrounds

Rather than having a plain colour as a background, pixel backgrounds were used. This was done with the purpose of adding further gamification elements to the web application. These backgrounds were taken from wallhaven<sup>18</sup> and shutterstock<sup>19</sup>. Finding these backgrounds was quite a tedious task because some backgrounds would not look nice against the page header, for example, if a background was too dark, the logo would barely be visible, therefore light backgrounds were used. Another advantage of using light backgrounds was that it gave a more inspirational and cheerful look.

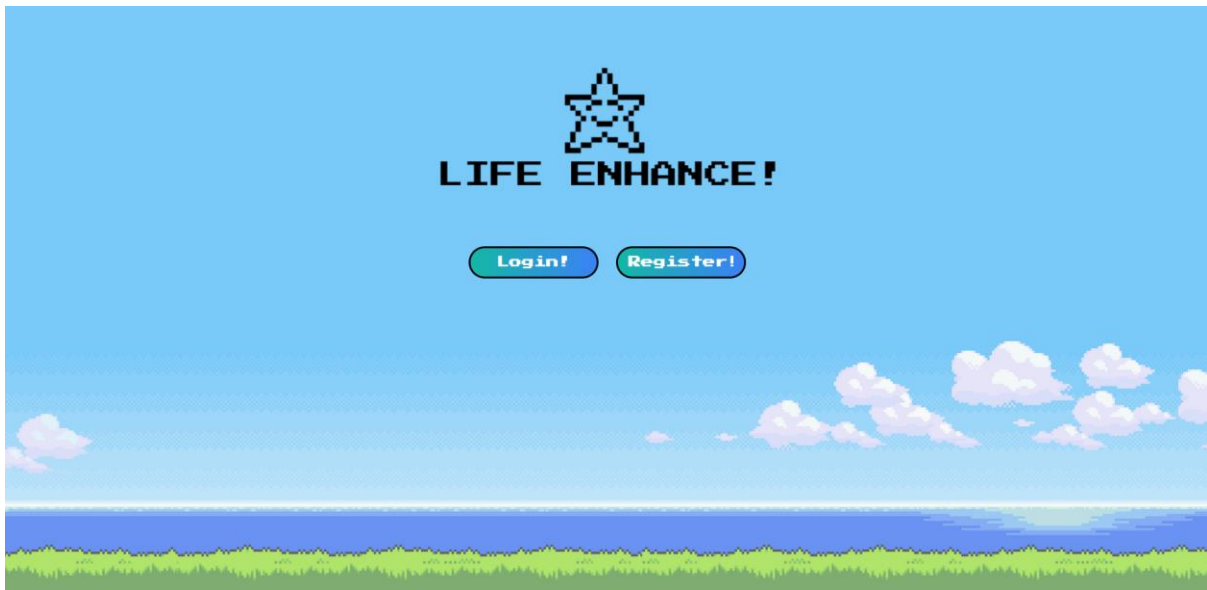
For the campfire page the background chosen contains a campfire to try to add a bit of immersion onto the web application. Similarly, the profile page contains a house, to replicate a “home” feeling since the user is on their profile.

#### 4.2.5 Names

In order to add further gamification elements to the web application, game-like names were used for various elements of the web application. In the original design, the following names were proposed: “Mental Health, Physical Health, Goals, Profile”. However, these have been changed to “Campfire, Your Stats, Objectives and Profile”. The username and the profile picture fields have also been replaced with “Player Name” and “Avatar”. This feels like you are playing a game based on your life.

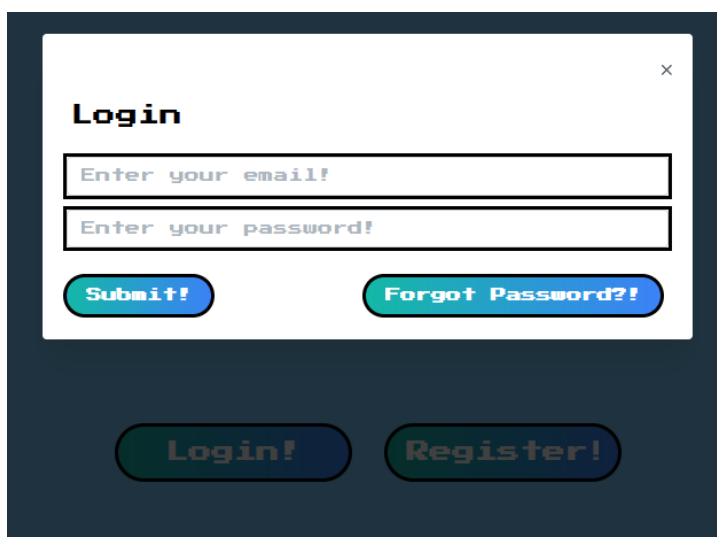
#### 4.2.6 Homepage

Figure 26 shows the homepage:



*Figure 26-Homepage Page*

The look of it is far more unique than a regular web application, opting for a videogame look. In the middle of the screen, there are 2 buttons. Each one of them opens a modal that contains the authentication system.



*Figure 27 - Login Modal*

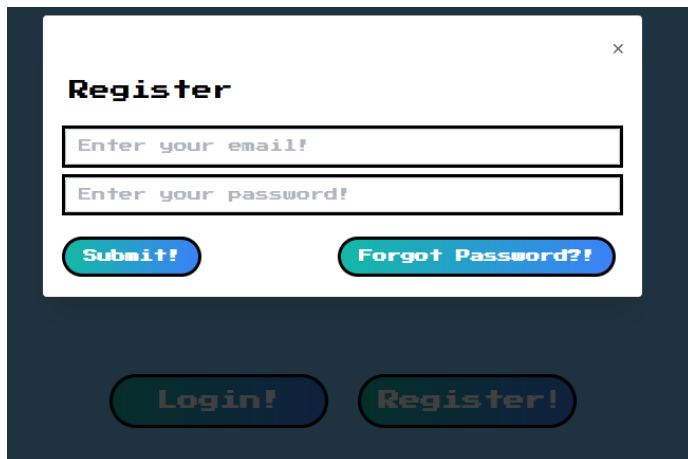


Figure 28 - Register Modal

The modals were created using a library called **Mantine**. Mantine is a React components library which contains over 120 customizable components. Mantine was installed using NPM. Below is a code snippet containing the login modal:

```
<Modal transition="slide-right" transitionDuration={500}
  opened={login}
  onClose={() => setLogin(false)}
  size="30%"
/>
```

Inside the modal you can then place various elements to create the login form and then you simply close the modal with the `</Modal>` tag. Mantine allows to customise the component quite thoroughly, for example the login modal has a “slide-right” transition, which means that the modal will appear from the left side of the screen.

There is error checking on every single action and field which is displayed using toasts.

#### 4.2.7 React-howler

This is an audio library and was installed using NPM. This was used to play a campfire audio on one of the pages in the web application. React-howler offers a lot of customisations to do with the audio. However, in the case of Life Enhance, the only thing that needed was to play a sound. This was done as simple as this:

```
let sound = new Howl({
  src: [campfireSound]
})
sound.play();
```

#### 4.2.8 React Router

This library was used to create routes for Life Enhance. It was installed using NPM. The use of routes made the application a single-page web application.

The first thing done was creating a `Routes.js` file which exported the name of all the pages that Life Enhance will use.

```
export const HOMEPAGE = "/";
export const DASHBOARD = "/dashboard"
export const GOALS = "/goals"
export const MENTALHEALTH = "/mentalHealth"
export const PHYSICALHEALTH = "/physicalHealth"
export const PROFILE = "/profile"
```

Then, inside App.js, the routes were specified:

```
function App() {
  return (
    <>
      <Routes>
        <Route path={ROUTES.HOMEPAGE} element =
{<Homepage></Homepage>}></Route>
        <Route path={ROUTES.DASHBOARD} element =
{<Dashboard></Dashboard>}></Route>
        <Route path={ROUTES.GOALS} element = {<Goals></Goals>}></Route>
        <Route path={ROUTES.MENTALHEALTH} element =
{<MentalHealth></MentalHealth>}></Route>
        <Route path={ROUTES.PHYSICALHEALTH} element =
{<PhysicalHealth></PhysicalHealth>}></Route>
        <Route path={ROUTES.PROFILE} element = {<Profile></Profile>}></Route>
      </Routes>
    </>
  );
}
```

The routes were imported the following way:

```
import * as ROUTES from "./constants/Routes";
```

And the pages were imported the following way:

```
import Dashboard from "./pages/Dashboard";
import Goals from "./pages/Goals"
import Profile from "./pages/Profile";
import PhysicalHealth from "./pages/PhysicalHealth";
import MentalHealth from "./pages/MentalHealth";
```

When this was first created, the gamification names were not thought of yet, which is why the page names and routes do not correspond to what's on the page.

#### 4.2.9 Page Header Component

Since the page header will appear in every page, a component was used, so the code can be reused. The page header contains a back button (unless you are on the dashboard), the logo and the avatar.



Figure 29 - Page Header

Tailwind CSS was used to customise all the header elements, as well as to position them. For the back button, a ternary operator was used. Since the web application does not contain a lot of pages, the back button simply navigates back to the dashboard if pressed. The back button is not displayed if the page we currently are on is the dashboard.

```
{
  (urlLocation.pathname!==DASHBOARD)?<BackButton></BackButton> : null
}
```

The avatar and display name are taken directly from the database; this will be discussed later in the backend chapter.

Another feature implemented while keeping in mind Nielsen's heuristic "Flexibility and efficiency of use" was to provide shortcuts which are hidden from novice users. In the example of the page header, shortcuts were implemented the following way: a shortcut to dashboard if the user presses the logo and a shortcut to the user profile if the user presses his own avatar picture. These are standards shortcuts used on almost every single web application.

Logo shortcut:

```
<div className=" justify-center h-64 mt-2 ml-[48rem]">
  <Link to={DASHBOARD}><img src={logo}></img></Link>
</div>
```

Avatar shortcut:

```
<Link className="break-words" to={PROFILE}>
  <Avatar className="w-28 h-32 mt-1 ml-2" src={user.photoURL}></Avatar>
</Link>
```

Another important code in the page header is the following:

```
let sound = new Howl({
  src: [campfireSound]
})
```

```
useEffect(()=>{
  if(urlLocation.pathname==MENTALHEALTH){
    sound.play();
  }else{
    Howler.stop();
  }
},[])
```

What this does is play a campfire noise if the current page is the mental health page(campfire). If it is not, it stops the sound.

#### 4.2.10 Dashboard

On this page, the user can navigate to all the pages in the web application. Below is the dashboard page

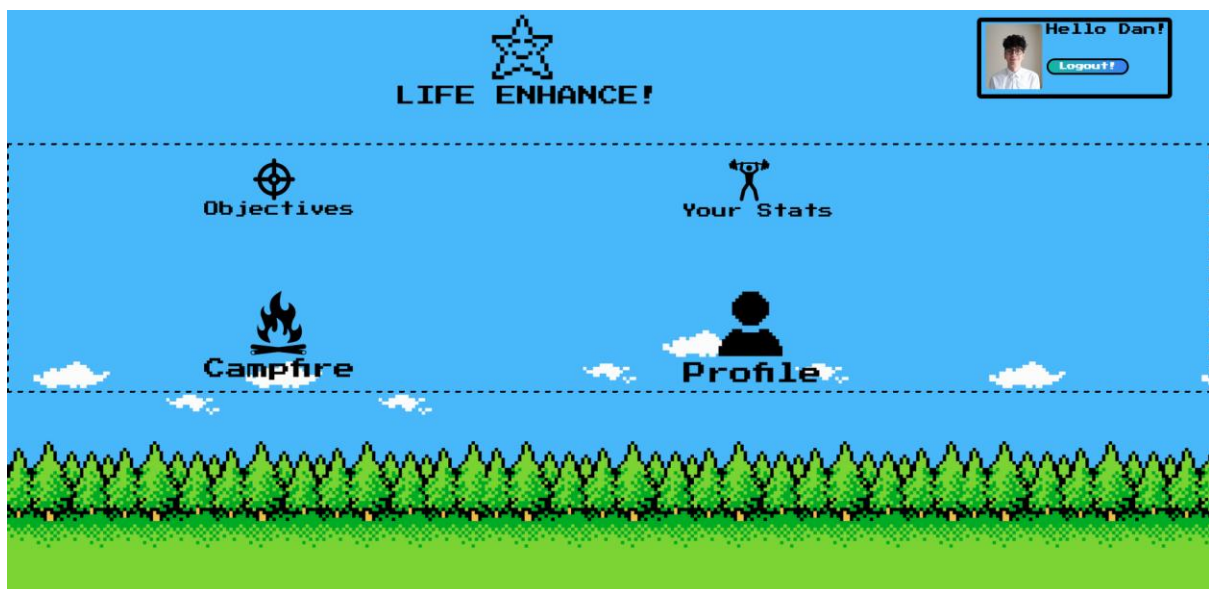


Figure 30 - Dashboard Page

The icons were created using a website called ucraft logo maker<sup>20</sup>. Upon the user selecting any of them, it redirects them to that page. Also note how the back button is not displayed in the page header, this is because you cannot go any more back than this.

#### 4.2.11 Objectives

On this page, the user can set objectives for himself, as well as mark them as complete when they achieved them. The figure contains a screenshot of the objectives page:

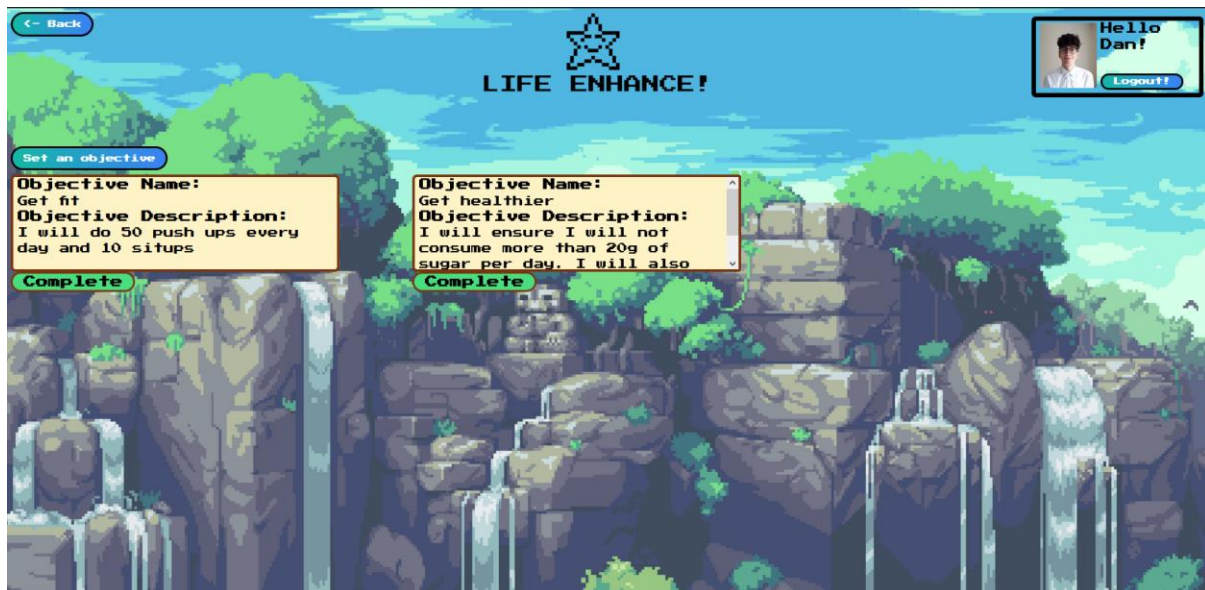


Figure 31 - Objectives Page

The setting of the objective is done using a Mantine modal. Below is how the modal looks:

Figure 32 - Objective modal

Each objective has a “complete” button, which the user can select in order to complete that goal. To ensure all objective boxes are the same size and width, the following tailwind CSS code was added to the div:

```
<div className="grid grid-cols-3 overflow-y-auto">
```

What overflow-y-auto does is create a scrollable bar in that allows the user to see the remaining text, in the eventuality of the text being too long. This can be observed in Figure 29, in the 2<sup>nd</sup> objective.



Ternary operators were also used during the frontend development of this page. Thus, if the user has not added any objectives, the following will be displayed:

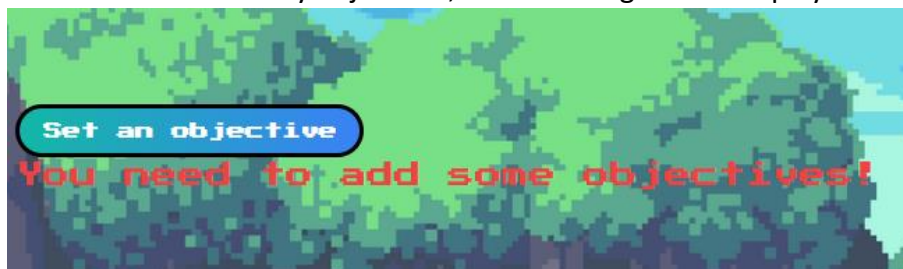


Figure 33 - Ternary operator used on the Objectives Page

#### 4.2.12 Your Stats

On this page, the user can see his MyFitnessPal details presented in a more unique way, specifically a bar chart and a pie chart.

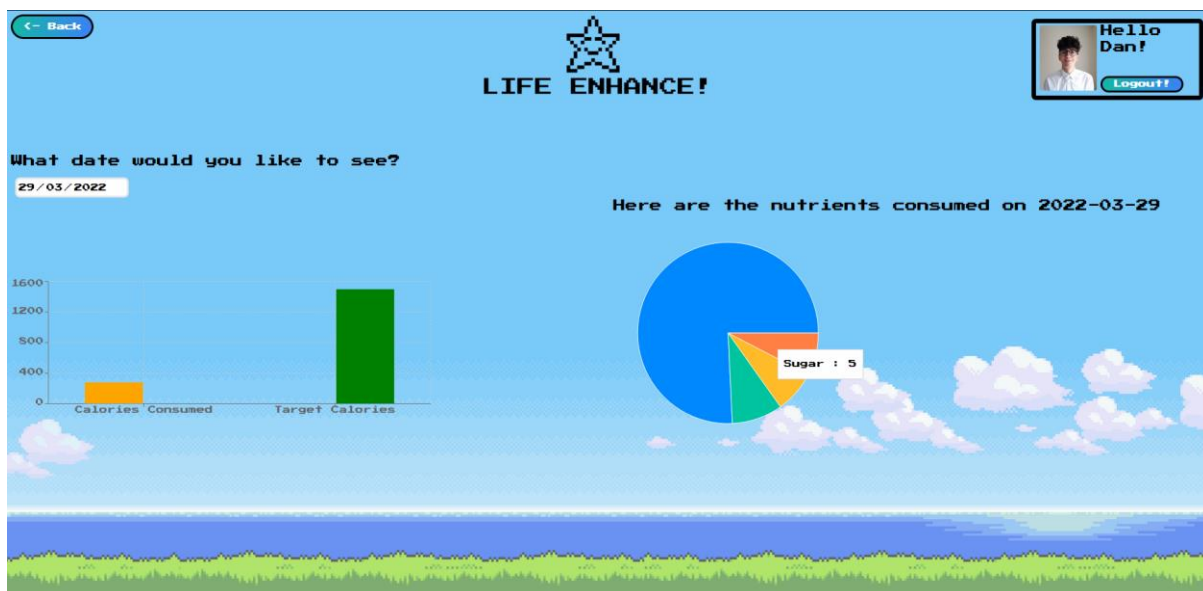


Figure 34 - Your Stats

The charts were created using a library called **Recharts**. This was installed using NPM. Below is the code for the creation of the bar chart.

```
<BarChart width={730} height={250} data={barChartData}>
  <CartesianGrid strokeDasharray="3 3" />
  <XAxis dataKey="name" />
  <YAxis />
  <Tooltip></Tooltip>
  <Bar barSize={100} dataKey="consumedCalories" fill="orange" />
  <Bar barSize={100} dataKey="targetCalories" fill="green"/>
</BarChart>
```

The way it works is it pulls data from an array (in this case "BarChartData") and displays it. Calories consumed value is taken directly from MyFitnessPal, more about this will be discussed later. Target Calories value is also taken from the database.



The pie chart works the same way, the values are taken directly from MyFitnessPal and are displayed in the form of a pie chart. Below is the code for the pie chart.

```
<PieChart width={1000} height={400}>
  <Pie
    dataKey="consumed"
    isAnimationActive={false}
    data={pieChartData}
    cx={200}
    cy={200}
    fill="#8884d8"
  >
    {pieChartData.map((entry, index) => (
      <Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />
    ))}</Pie>
  <Tooltip></Tooltip></PieChart>
```

It is similar to the bar chart. One key difference though is the following piece of code:

```
{pieChartData.map((entry, index) => (
  <Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />
))}
```

Unlike its predecessor, you are not able to add colours by just using the “fill” keyword. Instead, an array of colours had to be created:

```
const COLORS = ["#0088FE", "#00C49F", "#FFBB28", "#FF8042"];
```

And for each entry the colour was populated this way.

Furthermore, the date input was created the following way:

```
<input className="w-56 h-10 mt-4 rounded-lg border-4 ml-4" type="date"
  onChange={(event)=>setDate(event.currentTarget.value)} required/>
```

Although it is worth noting that the date’s hook default value is the current date. The current date is generated using the following function:

```
function getCurrentDate(separator='-'){
  let newDate = new Date()
  let date = newDate.getDate();
  let month = newDate.getMonth() + 1;
  let year = newDate.getFullYear();
  console.log("date date date is "+date);
  if(date<10){
    date = 0+date;
    return `${year}${separator}${month<10?`0${month}`:`${month}`}${separator}0${date}`
  }else{
    return `${year}${separator}${month<10?`0${month}`:`${month}`}${separator}${date}`
  }
}
```

The reason for “if(date<10) is because if the date was for instance 01-04-2022, then it would be presented as:

2222-04-1

While there is nothing wrong with that, the backend code which is explained further down the report, requires the date to be passed as an argument in the following format: YYYY-MM-DD. Thus, if the “day” part of the date is below 10, then a 0 is added in front of it to adhere to the required date format.

If there is no data from MyFitnessPal related to the selected date, then the following message will be displayed instead of the graph and the pie chart.



Figure 35 - Error message if no calories added

#### 4.2.13 Campfire

On this page, the user can say what they are grateful for and share it with other users. The user is also able to see what other users are grateful for. Below is how the Campfire page looks like:

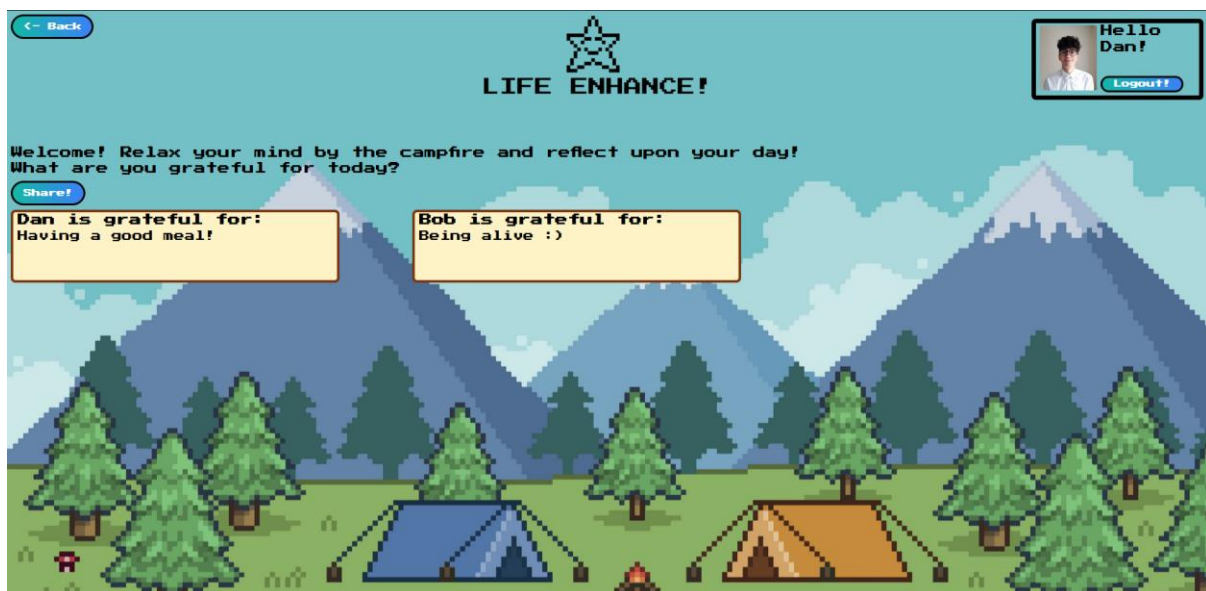


Figure 36 - Campfire Page

It is not possible to remove any of the posts, including your own, however only the most recent 6 are displayed.

Howler, which was discussed in the PageHeader component, is used here to play a campfire sound.

Like the objectives, by pressing the “Share” button, a Mantine modal opens which allows the user to share what they are grateful for today.

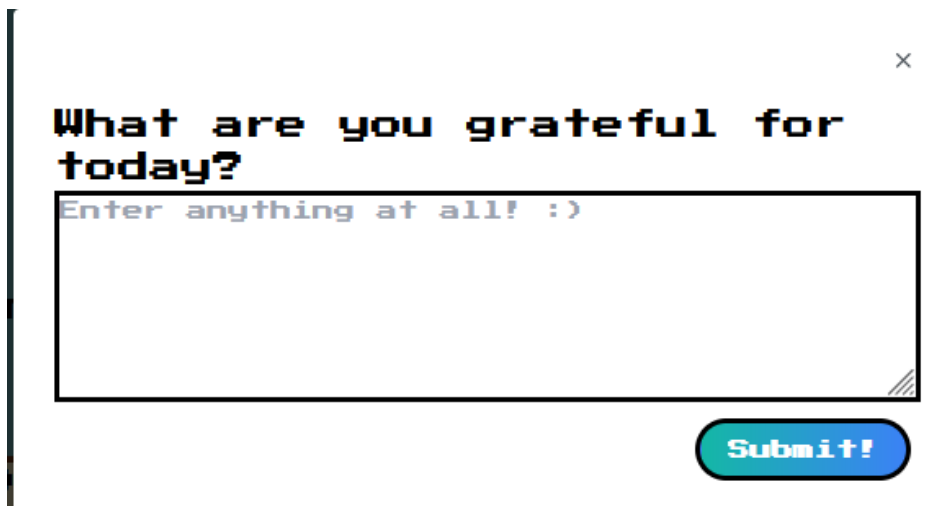
A Mantine modal window with a close button (X) in the top right corner. The text inside reads "What are you grateful for today?" in a large, bold, black font. Below this is a text input field with a placeholder text "Enter anything at all! :)". At the bottom right of the modal is a blue button with the text "Submit!" in white.

Figure 37 - Campfire Modal

#### 4.2.14 Profile

On this page, the user will be able to set their display name (Player Name), Profile Picture (Avatar), as well as their MyFitnessPal user and their target calories. Below is how the page looks like:

A screenshot of a profile page with a pixel art background featuring a wooden cabin, a tree, and a fence. The page has a blue header with a star icon and the text "LIFE ENHANCE!". In the top left corner is a blue button with a back arrow and the text "Back". In the top right corner is a user profile card showing a small avatar, the text "Hello Dan!", and a "Logout!" button. The main content area has a "Welcome to your profile Dan" message. Below this are two form sections. The first section, titled "Player name:", has a text input field and a "Submit" button. The second section, titled "Configure your account", has two text input fields labeled "MyFitnessPal Username(Must be Public):" and "Target Calories:", and a "Submit" button. A toast message "Avatar: No file selected." is visible near the first form section.

Figure 38 - Profile Page

All the fields have error-checking in place, which once again is displayed with the use of toasts. The values are sent to the database, which will be discussed later in the report.

### 4.3 Firebase

Firebase was used for Authentication, Storage and Database. The reason for this was because Firebase was relatively easy to set up and use. Furthermore, it was possible to create the entire application with the free-tier account.

#### 4.3.1 Setting up Firebase Project

Before being able to use Firebase, a Firebase project had to be created and the web application had to be registered to the project. The Firebase project was created using the Firebase console.

After the Firebase project was created, Life Enhance could be registered.

Then, Firebase was installed using NPM and initialised inside a firebase.js file.

#### 4.3.2 Firebase Authentication

Firebase Authentication comes with built-in methods that allows users to sign up and login, as well as access profile-management services such as forget password.

This is the code used to sign up new users:

```
createUserWithEmailAndPassword(getAuth(),email,password)
  .then((response) => {
    sessionStorage.setItem('Auth Token', response._tokenResponse.refreshToken)
    setRegister(false);//Closing the modal upon registering
  })
  .catch((error)=>{
    if(error.code === 'auth/email-already-in-use'){
      toast.error('An user with this email has already been created! Select "Forget Password" to reset your password')
    }
    if(error.code === 'auth/invalid-email'){
      toast.error('Please enter a valid email!')
    }
    if(password==="){
      toast.error("Please enter a password value!")
    }
    if(error.code === 'auth/weak-password'){
      toast.error("Passwords should be at least 6 characters")
    }
  })
}
```

As we can see, the function to create a new user takes in 3 parameters, `getAuth()` which is imported from “firebase/auth” and the email and password of the new user. The error codes are all handled by Firebase. The only thing needed to do during the development is checking for them and notifying the user about them if they occur. This can be observed in the following line of code:

```
if(error.code === 'auth/weak-password'){
  toast.error("Passwords should be at least 6 characters")
}
```

The `error.code` is passed back from Firebase if the function fails to sign up the new user. It passes the error code, which in this case is “weak-password”. The user is then notified using a toast that their password should be at least 6 characters.

The session is set inside this method. A session allows the user to access the rest of the web application, since they are signed up.

The login method is equal to the signing up one, but with different error codes.

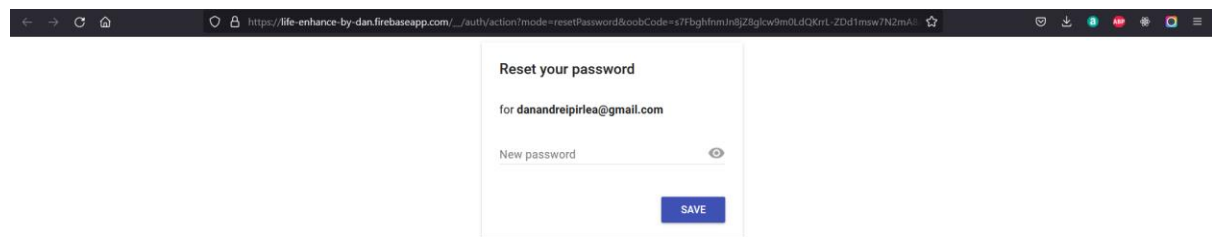
```
signInWithEmailAndPassword(getAuth(),email,password)
```

Another function imported from Firebase was the `handleForgotPassword()`. This only requires 2 parameters: the Auth and the email.

```
function handleForgotPassword() {  
  sendPasswordResetEmail(getAuth(), email).then(()=> {  
    toast.success("Password has been reset. Please check your email! ");  
    setRegister(false);  
  })  
}
```

When you request this, the user will receive an email with instructions on how to reset their password.

It is worth noting that the resetting of the password happens on an URL hosted by Firebase, not on your own web application.



*Figure 39 - Resetting password*

This is very advantageous as it means one less page to handle on your web application, making it more lightweight and easier to debug.

Firebase Authentication also supports sending a verification email before creating an account using the function `sendSignInLinkToEmail()`. However, this was not used during the development of Life Enhance, due to the unavailability of emails as well as the extra time needed to constantly confirm the user.

The email and passwords are the values the user inputs in the “TextInput” modal component. For example, this is the TextInput for the email:

```
<TextInput className = "mt-2 border-black border-4" value={email}  
  onChange={(event)=>setEmail(event.currentTarget.value)}  
  placeholder="Enter your email!" />
```

Every time the value changes, it sets the email hook to whatever is inside TextInput (which is done using `event.currentTarget.value`).

There are 2 more values associated with Firebase Authentication: the display name and the photoURL. When the user first creates an account, their display name gets defaulted to

“Newbie” and their photoURL to null (instead it is just an empty avatar, as seen in the picture below).



*Figure 40 - Empty Avatar and Default Display Name*

These values are changed inside the profile section, using the function below:

```
updateProfile(auth.currentUser, {  
  displayName: displayName, photoURL: photoURL  
})
```

The display name is simply taken from a hook which is changed by the user inputting details onto the TextInput and pressing the submit button, like the logging in and signing up functions.

However, for the photoURL, it is much more complex, and the use of Firebase Storage is needed. This will be discussed in the chapter below.

#### 4.3.3 Firebase Storage

Firebase Storage allows you to upload user generated content. In the case of Life Enhance, the user must upload a profile picture and then an URL should be generated that gets uploaded to the photoURL attribute of Firebase Authentication.

Firebase Console had to be used to set up Firebase Storage. The security rules had to be configured as well as the location of the Cloud Storage bucket. Since the web application is in its preliminary stages and no one has access to it, the security rules are defined as public.

Furthermore, the Storage Bucket URL which was generated for us had to be added to the firebaseConfig.js file.

To start using the Storage, a reference to the storage bucket was created using the line of code below:

```
const storage = getStorage();
```

This line is in the Profile Page code, the reasoning behind it being that the user can only update his avatar in their profile page. After this was done, a file reference constant was created, which holds the reference for the file. The way the file reference was built was the following way:

```
const fileRef = sRef(storage, user.uid + '.png');
```

sRef creates a “Storage Reference” which takes in 2 parameters, the “storage bucket”, which is “storage” and then the url. To make it easily distinguishable, the user’s uid was used.

The next step is to prepare the file. In terms of front-side code, the file was uploaded using an input of type file

```
<input className="ml-2" type="file" onChange={handleProfileUpdate}></input>
```

When the user uploads a file, the handleProfileUpdate function is called, which sets the photo hook to the photo the user uploaded

```
if(e.target.files[0]){  
  setPhoto(e.target.files[0])  
}
```

The reason behind the [0] was because the user should only be able to upload just a picture, and [0] is the first one.

Using the file reference and the photo the user uploaded, the function uploadBytes can be used to upload the photo onto the Storage Bucket.

```
await uploadBytes(fileRef, photo)
```

After the photo is successfully uploaded, the photoURL can be recovered from it, using the following code:

```
photoURL = await getDownloadURL(fileRef);
```

It uses the fileReference that got created earlier to get the downloadURL.

Finally, the photoURL is set in the photoURL hook and we can update the photoURL using the updateProfile function discussed earlier.

#### 4.3.4 Firebase Realtime Database

Although during the system design phase, PostgreSQL was the chosen database, Firebase Realtime Database was used instead. The reason for this choice was because Firebase Authentication and Firebase Storage have already been set up, so it fit nicely with the workflow (for example having the database config in the firebase.js file).

Firebase Realtime Database is a NoSQL cloud database. The data is stored in JSON format. During the system design chapter, the consensus was that a SQL database will be used, but since Firebase Authentication dealt with the users and their display name, there was not a need for an SQL database.

To create the database, Firebase Console had to be used. Similarly, to Firebase Storage, the security rules used are public and a location had to be chosen. Then, once again, inside the firebase.js config file, the databaseURL had to be entered.

The database was used to store many things, such as the objectives, the target goal, and the gratefulness posts.

The database usage is quite straight forward, below is a demonstration using the gratefulness posts. The first thing was to get the actual database. This is extremely similar to the `getStorage()` method, except it is for the database.

```
const db = getDatabase();
```

Then, the following method adds a new post onto the database

```
function handlePost() {  
  const auth = getAuth();  
  const user = auth.currentUser;  
  
  push(ref(db, '/posts/'), {  
    Post: post,  
    Author : user.displayName  
  }).then(r => {  
    setPostSetting(false);  
    navigate(MENTALHEALTH);  
  })  
}
```

The post is taken from the post hook, which is what the user had added when they pressed the submit button and their display name is also taken and set into the Author field.

However, to display the post, it must be pulled from the database. To do this, a posts reference constant was created, which is where exactly the posts are located.

```
const postsRef = query(ref(db,'posts/'),limitToLast(6));
```

An important thing to note is that the query method was used to only reference the last 6 posts. This was done to prevent too many gratefulness posts on the page

Then, the following function deals with getting the posts.

```
function getPosts(){  
  onValue(postsRef,(snapshot => {  
    snapshot.forEach((childSnapshot)=>{  
      data = {  
        ref : childSnapshot.key,  
        post : childSnapshot.val().Post,  
        author : childSnapshot.val().Author  
      }  
      posts.push(data);  
    })  
  }))  
}
```

The reason `onValue` was used here was because it returns the complete list of data as a single snapshot, which then you can access each individual children using a loop, which is exactly what the function is doing.

Data is just a variable which holds the JSON data pulled from the database.



The key (which is the ID of the post which Firebase automatically assigns) is then added to the data variable, as well as the post and author. The data object is then added to the posts array, which contains all the posts objects.

The following code is used to display the posts:

```
<div className="grid grid-cols-3">
  {
    posts.map((post)=>{
      return (<Post postKey={post.ref} post={post.post} author={post.author}></Post>)
    })
  }
</div>
```

A post component was created, which takes in 3 props, the key, the post itself and the author. In this code above, we iterate over the posts object array and for every single object we create a post that contains the props that were inside the post object pulled from the database.

The objectives are displayed in the exact same way.

There were also a few unique fields such as Target Calories and MyFitnessPal Username. They are unique because only 1 field exists for them, whereas posts you can have multiple of them. These fields were set using the “set” method, as shown in the code below:

```
set(ref(db, '/users/' + getAuth().currentUser.uid), {
  MFPUUsername: MFPUUsername,
  TargetCalories : targetCalories
}).then(r => {
  navigate(PROFILE);
})
```

It is the same as the above code, however you just set those fields, rather than “push” a new value.

Similarly, the following function was used to get a single value from the database

```
await get(child(dbRef, '/users/' + getAuth().currentUser.uid +
  '/MFPUUsername')).then((snapshot) => {
  if (snapshot.exists()) {
    setUserMFP(snapshot.val());
  }
})
```

In this code, it simply gets a snapshot of “MFPUUsername” and then the snapshot’s value is used to set the user’s MyFitnessPal hook. The reason this works is because there’s only 1 MFPUUsername value in the database for the user.

#### 4.4 Backend Development (Node JS)

Although Firebase was used for most of the backend development, in order to make API calls, Firebase Functions had to be used, however they do not come in the free-tier account. Therefore, Node JS was used instead.

#### 4.4.1 Creating the backend

In order to create the backend, the following commands were executed:

NPM init

NPM install express

NPM install nodemon

Then, the package.json file was modified, specifically the scripts' part, to contain the following code:

```
"scripts": {  
  "start": "nodemon app.js"  
},
```

Afterwards, a file app.js was created.

**Express** is a backend web application framework for Node.js. In simple terms, it makes web servers and backend easy to develop.

**Nodemon** is a tool that helps developers code Node.js based applications by automatically restarting the node application when any files get changed.

With Express being added, creating the backend was done simply by adding the following lines of code:

```
const express = require('express');  
  
const app = express();  
app.listen(3001);
```

Port 3001 was used since port 3000 was taken by the React application.

#### 4.4.2 Creating Routes/Endpoints

The main reason for creating this backend was to execute API calls. To create a route, a directory called routes was created and inside a "fitnessDiary.js" file was created.

```
const express = require("express");  
  
const router = express.Router();  
  
//our code  
  
module.exports = router;
```

The first 2 lines of code create the route and the 3<sup>rd</sup> line exports the route. Inside this code, there will be the method which pulls data from MyFitnessPal, which will be discussed in the following chapters.

#### 4.4.3 Middlewares

Middlewares are functions which execute when routes are being hit. The following example is a middleware that executes everytime the endpoint “/fitnessDiary” is reached and ensures the fitnessDiary route is being executed.

```
const fitnessDiaryRoute = require('./routes/fitnessDiary');  
app.use('/fitnessDiary',fitnessDiaryRoute);
```

The route is now setup and ready to be used

#### 4.4.4 MyFitnessPal API

The API for MyFitnessPal was requested at the beginning of the year on their official website. Unfortunately, there was no response. Because of this, MFP<sup>21</sup> was used, which is a third-party API for accessing MyFitnessPal diary data. The only disadvantage with this API is that it can only access your MyFitnessPal diary only if it is public.

This was installed using NPM. The following line of code was added to add it to the backend:

```
const mfp = require('mfp');
```

Inside the route created earlier, the following code was added in order to pull the information from MyFitnessPal

```
router.get('/',(req,res)=>{  
  const mfp = require('mfp');  
  let MFPUsername = req.query.username;  
  let date = req.query.date;  
  mfp.fetchSingleDate(MFPUsername,date,'all',function (data){  
    res.send(data);  
  })  
})
```

MFPUsername and the date are the parameters taken from the frontend when this function is called.

The function FetchSingleDate() is part of the MFP Third Party API, which gets all the diary data from a user’s MyFitnessPal profile. It accepts several parameters, the username, the date that you want to retrieve the information from and which nutrients you would like to pull. If you want all of them, the string ‘all’ is accepted, otherwise you can insert an array with the nutrients you want (eg: {calories,carbs}).

Finally, res.send(data) sends the data retrieved from the API call back to the frontend. More specifically, data is an object whose keys are the nutrient field names and values are each number consumed, as well as the date.

#### 4.4.5 CORS

The frontend and backend have different “origins”. For example, localhost:3000 is the frontend and localhost:3001 is the backend. Because of this, the frontend is not able to communicate to the backend due to security reasons. The package “CORS” can be installed using NPM, to relax the security applied to the API and allow the frontend and backend to

communicate between each other. The following code demonstrates how CORS is applied to the backend:

```
const cors = require("cors");  
app.use(cors());
```

A middleware executes this on every single endpoint of the backend.

#### 4.4.6 Axios

The way the API was called from the frontend was using Axios. Axios is a promise-based HTTP Client for Node Js and the browser. It was installed using NPM. Below is the Axios call created to get the data from MFP.

```
await Axios.get("http://localhost:3001/fitnessDiary", {  
  params: {  
    username: usermfp,  
    date: date  
  }  
}).then((response) => {  
  setCalories(response.data.calories);  
  setCarbs(response.data.carbs);  
  setFat(response.data.fat);  
  setProtein(response.data.protein);  
})
```

The parameters are being sent to the backend. Usermfp is taken from Firebase Realtime Database and the date is taken from an input type="date". After it gets the response, it can access the "data" and get the calories, carbs, fat and protein from the user's fitness diary. These values are then used to construct the graphs.

#### 4.4. Conclusion

The system architecture was changed quite heavily from the initial design, with a bigger focus on Firebase and less focus on Node JS. However, Node JS was still used to create routes and endpoints in order to connect the MFP Third Party API to the frontend. The database used was Firebase Realtime Database

As for the frontend, plenty of gamification elements were added to make the application engaging and fun to use. These gamification elements range from fonts to pixel background pictures.

Overall, the main gist of the web application is still the same, and although the technologies changed slightly during the development phase, it still serves the same purpose, to provide a self-improvement web application which is more engaging and interesting than its competitors.

## 5. Testing and Evaluation

### 5.1. Introduction

This section will discuss the plans for testing. Since an agile approach will be used, as well as a feature driven development, testing is crucial. Every single feature will have to be tested before being deployed.

The ways Life Enhance will be tested will be using white box testing and exploratory testing.

Evaluation is also a crucial element. During the background research, Nielsen's Heuristics were discussed. The evaluation will be against the discussed heuristics.

### 5.2. Black Box Testing

Black Box Testing was used to test the software. This is a testing approach which the tester is not aware of the internal details of the application, which inherently means it allows the system to be tested from the user's point of view.

Several testcases were created, which describe simple user stories, such as logging in, registering and setting up your profile. Below are the testcases created:

Test Case Number	Test Case Description	Test Case Steps	Expected Results
1	Create account	-Go to the web application -Select the register button -Fill in a valid email and password -Select the submit button	The user gets successfully registered and redirected to the dashboard
2	Log out	-PRE-REQ: Execute test case number 1 -Select the "logout" button beside your avatar	The user gets redirected back to the homepage
3	Login	-PRE-REQ: Execute test case number 1 and have an account -Select the login button on the homepage -Enter the details you used to create an account	User gets redirected to the dashboard
4	Set up an objective	-PRE-REQ: Execute testcase number 3 -Select the "Objectives" icon -Select the "Set an objective" button -Insert an objective name and description -Press Submit	The objective appears on the page
5	Create a gratefulness post	-PRE-REQ: Execute testcase number 3 -Select the "Campfire" icon -Select the "Share" button -Insert what you are grateful for -Press Submit	The post appears on the page
6	Set up your display name	-PRE-REQ: Execute testcase number 3 -Select the "Profile" icon	In the corner, the display name gets

	and avatar picture	-Insert a display name in the "Player Name" field and an avatar picture in the "Avatar" field -Select "Submit"	updated as well as the avatar that you uploaded
7	Set up your MyFitnessPal account	-PRE-REQ: Execute testcase number 3 -Select the "Profile" icon -Insert your MyFitnessPal username in the "MyFitnessPal Username" field, ensuring your diary is set to public and a numerical value for the target calories -Press Submit -Add food on MyFitnessPal on today's date -Select the back button -Select the "Your Stats" icon	The calories consumed should be displayed on the bar chart and the nutrients consumed should be displayed on the pie chart

All these testcases were executed by my friends and family on my machine, and they all passed test.

Furthermore, exploratory testing was also executed. Exploratory testing is an approach to testing which involves the user "playing around" rather than following a set of steps as described in the testcases. Using exploratory testing, several bugs have been discovered, some which have been fixed and some which have not. The ones that have not will be discussed in chapter 6, which deals with all the issues.

## 5.4. Evaluation

The evaluation was done against Nielsen's heuristics discussed in the background research.

**Visibility of System Status:** The way this was incorporated in Life Enhance was using toasts, which tells the users if their profile has been updated successfully for example or if any errors occurred.

**Match between System and the real world:** The design does not use any jargon, it is all easily understandable. The users that tested the web application have had no complaints regarding this matter.

**User Control and Freedom:** The design provides constant exits. For example, on "Your Stats" page, the user can leave the page by pressing the back button or the logo. Furthermore, modals can be closed by just clicking outside of them, the user does not have to look for the 'x' to close it.

**Consistency and Standards:** The design was consistent throughout. For example, the term "Objective" is used everywhere, including the modals, placeholders, and the objective component.

**Error Prevention:** Plenty of error prevention was added to Life Enhance, for example Firebase Authentication has several error-codes generated. In my design, I ensured all the error-codes have been handled accordingly with the use of toasts.

**Recognition rather than Recall:** The way the system was designed was to make the user's experience an easy one. For example, the way "your stats" was designed first, the user had to constantly add his MyFitnessPal username and "search for it". But this broke this heuristic, therefore a field was added in the profile where the user sets up his MyFitnessPal username once and it is stored in a database. This way, the user does not have to recall his username constantly.

**Flexibility and efficiency of use:** The design caters to expert users. For example, the user can press the Life Enhance logo in order to navigate back to the dashboard or his avatar in order to navigate to their profile. These interactions are common in pretty much every single web application in the world, so an experienced internet user will have no problems using these shortcuts.

**Aesthetic and minimalist design:** The design is minimal, there is not an overabundance of words and everything is kept short and simple.

**Help users recognise, diagnose and recover from errors:** The errors are all displayed and there is help for users to fix their problems. For example, if they have an incorrect password, the system tells them they can reset their password by selecting the forget password button.

**Help and Documentation:** The system can be used without documentation, however if future work is done, then a documentation will be needed to be set up.

## 5.5. Conclusion

Life Enhance was tested by multiple people using White Box Testing as well as exploratory testing. Furthermore, it was evaluated against the Nielsen's heuristics which were discussed in the background research chapter and it passed every single heuristic.

## 6. Conclusions and Future Work

### 6.1. Introduction

In this section, the final conclusions will be discussed, as well as any remaining future work that Life Enhance could use.

### 6.2 Conclusions

#### 6.2.1 Design Conclusion

The project's design has not changed too much in terms of frontend, since it was built in React, however the backend was changed a lot. During the Experiment Design chapter, the main backend was going to be Node JS with PostgreSQL, however it ended up being Firebase with a little bit of Node JS for API calls. When the experiment design was first written, Firebase was mentioned slightly and it was said that it will be kept in mind throughout the development.

Since an agile approach was utilised, the first feature that have been developed was the authentication system, due to its high priority. This was done using Firebase Authentication. Afterwards, the profile feature was developed. Since Firebase Authentication and Firebase Storage were straightforward to use, the decision to use Firebase for everything was taken. This way everything is integrated easily without too much hassle or overhead.

The Entity Relationship Diagram has also changed; however, this was expected to change. In the end, a no SQL database was used (Firebase Realtime Database), but that was not so bad since the database does not require much structuring and it is small.

The class diagrams have also changed. During the experiment design, there was a heavy focus on using APIs, but when developing, many of the API's were unavailable or confusing to use, with outdated documentation. Therefore, the only API used was the MFP API which pulls data from MyFitnessPal. This one was kind of necessary because MFP provides a huge database containing all the foods, which would be hard to incorporate into my project.

For the objectives and gratefulness posts, there was not much need for an API, since the code to develop it was not too difficult, especially with the use of Firebase.

There were a few features which also have been abandoned, such as the user gaining levels. Although this would have helped the gamification element, it felt like it served no real purpose. And since the goal of the web application is to be lightweight and fast to use, it did not really feel right to add this feature in.

The achievements were also abandoned, mostly since they would have taken a lot of time to design. The way it was meant to work was to have achievements such as "Slowly Getting There – Completed 5 goals". However, this feature was very low priority compared to the other features, and it was mostly a feature requiring imagination, since the achievements had to be created and designed.

#### 6.2.2 Personal Conclusion

In the end, the project felt like it was a success. There is a lot of work that can be added, but the beauty of a web application is that you are able to add as many features as you can. The



work will never really end since there's always new and innovative ways of enhancing your life.

The time constraint was a bit difficult, especially balancing all the assignments as well as real life responsibilities. It definitely feels like having more time would have been contributed to a more developed project, however the project in its current state is not too bad either. The end goal as described in the introduction chapter was to "help people form life improving habits in a fun and entertaining way". This goal was achieved, as the web application feels incredibly fun and engaging to use and does not force the habit-forming process onto the user. The web application also makes the user more likely to use less attractive apps such as MyFitnessPal since they know their data will be presented in an enjoyable way, along with their objectives and gratefulness posts, in an energetic interface.

### 6.3 Future Work

As described earlier, there is a lot of future work that can be done. The first future work which would make the web application more engaging and fun to use would be the ability to share goals with your friends. Due to time constraints, there was no time for this to be developed. However, it should not be too hard to develop it, because there is already code to share gratefulness posts with other users.

Achievements could also be developed, just to further add to the gamification element. But they would require creativity to add in.

A long-term goal would be to change the API call from Node to be done using Firebase Functions instead, however this would require a Firebase account which is not free tier, as well as a re-vamp of the current backend code.

In the 'Your Stats' page, it would also be nice to get a few more APIs from various other applications and display their stats there. For example, Google Fit API could be used to see how much you walked in a specific date.

Media queries should also be implemented, right now the background images just repeat themselves, however they should scale depending on the resolution of the user's machine. Similarly, a mobile friendly web application should also be created as a future goal.

The exciting thing about Life Enhance is that there will always be features to add. New life improving applications are getting developed constantly and their APIs could easily be integrated in, as well as various innovative technologies.

## 6.4 GANTT Chart used

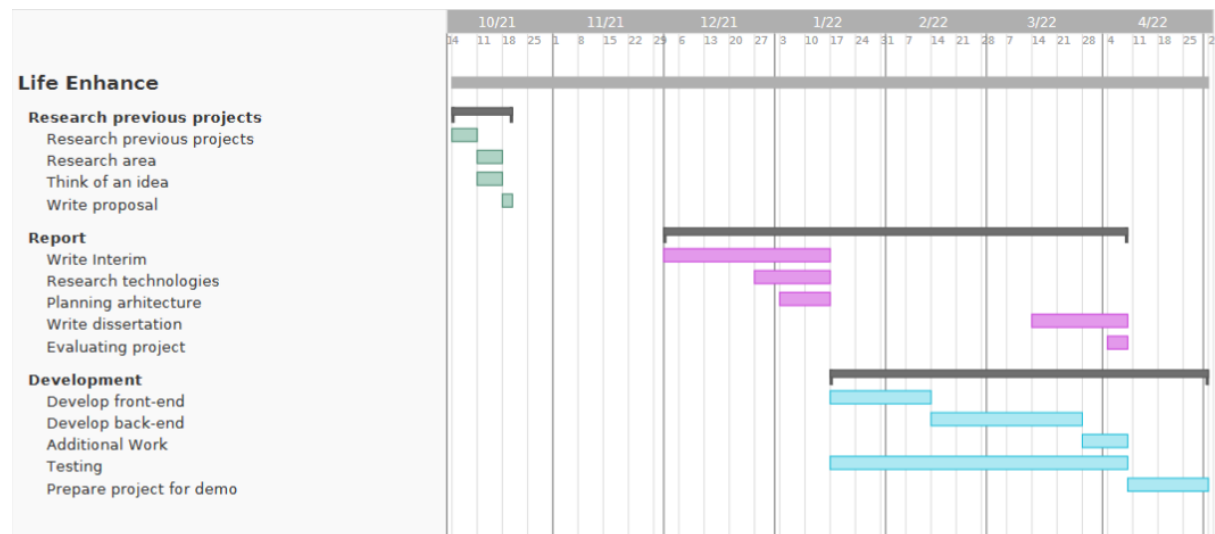


Figure 41 - GANTT Chart

## Bibliography

1. Ireland, H., 2022. *Get Ireland Active!*. [online] Assets.gov.ie. Available at: <<https://assets.gov.ie/7563/23f51643fd1d4ad7abf529e58c8d8041.pdf>> [Accessed 1 January 2022].
2. Spectrum Mental Health. 2022. *Mental Health in Ireland - Spectrum Mental Health*. [online] Available at: <<https://mentalhealth.ie/blog/mental-health-in-ireland>> [Accessed 15 January 2022].
3. Bordbia.ie. 2022. *What Ireland Ate Last Night*. [online] Available at: <<https://www.bordbia.ie/industry/news/press-releases/what-ireland-ate-last-night/>> [Accessed 15 January 2022].
4. Myfitnesspal.com. 2022. *MyFitnessPal | MyFitnessPal.com*. [online] Available at: <<https://www.myfitnesspal.com/>> [Accessed 15 January 2022].
5. Myfitnesspal.com. 2022. *Free Calorie Counter, Diet & Exercise Journal | MyFitnessPal.com*. [online] Available at: <[https://www.myfitnesspal.com/welcome/learn\\_more](https://www.myfitnesspal.com/welcome/learn_more)> [Accessed 15 January 2022].
6. MoodPanda. 2022. *MoodPanda App*. [online] Available at: <<https://www.moodpanda.com/>> [Accessed 15 January 2022].
7. App.remente.com. 2022. *Remente*. [online] Available at: <<https://app.remente.com/>> [Accessed 15 January 2022].
8. Reach your Goals. Change your Habits. 2022. *Goalify - Reach your Goals. Change your Habits. Download the free app.*. [online] Available at: <<https://goalifyapp.com/en/reach-your-goals/>> [Accessed 15 January 2022].
9. W3techs.com. 2022. *Usage Statistics of JavaScript as Client-side Programming Language on Websites, January 2022*. [online] Available at: <<https://w3techs.com/technologies/details/cp-javascript/>> [Accessed 15 January 2022].
10. Reactjs.org. 2022. *Tutorial: Intro to React – React*. [online] Available at: <<https://reactjs.org/tutorial/tutorial.html>> [Accessed 15 January 2022].
11. Reactjs.org. 2022. *React – A JavaScript library for building user interfaces*. [online] Available at: <<https://reactjs.org/>> [Accessed 15 January 2022].
12. Postgresql.org. 2022. *PostgreSQL: About*. [online] Available at: <<https://www.postgresql.org/about/>> [Accessed 15 January 2022].
13. Usabilitygeek.com. 2022. [online] Available at: <<https://usabilitygeek.com/heuristic-evaluation-introduction/>> [Accessed 15 January 2022].
14. UKEssays. November 2018. *Waterfall Methodology in Software Development*. [online]. Available at: <<https://www.ukessays.com/essays/computer-science/waterfall-methodology-in-software-development.php?vref=1>> [Accessed 15 January 2022].
15. Zaynabzahra, 2022. *SCRUM Methodology*. [online] Zaynab's Blog. Available at: <<https://zaynabzahrablog.wordpress.com/2017/10/07/scrum-methodology/>> [Accessed 15 January 2022].
16. Google Fonts. 2022. *Google Fonts*. [online] Available at: <<https://fonts.google.com/specimen/Press+Start+2P#about>> [Accessed 8 April 2022].

17. App.diagrams.net. 2022. *Flowchart Maker & Online Diagram Software*. [online] Available at: <<https://app.diagrams.net/>> [Accessed 8 April 2022].
18. Wallhaven.cc. 2022. *Awesome Wallpapers - wallhaven.cc*. [online] Available at: <<https://wallhaven.cc/>> [Accessed 8 April 2022].
19. Shutterstock. 2022. *Stock Images, Photos, Vectors, Video, and Music | Shutterstock*. [online] Available at: <<https://www.shutterstock.com/>> [Accessed 8 April 2022].
20. Ucraft.com. 2022. *Free Logo Maker | Create Your Logo Online with Ucraft*. [online] Available at: <<https://www.ucraft.com/free-logo-maker>> [Accessed 8 April 2022].
21. GitHub. 2022. *GitHub - fitnessforlife/mfp: A third-party API for accessing MyFitnessPal diary data*. [online] Available at: <<https://github.com/fitnessforlife/mfp>> [Accessed 8 April 2022].