FINAL REPORT
KETOGENETICS

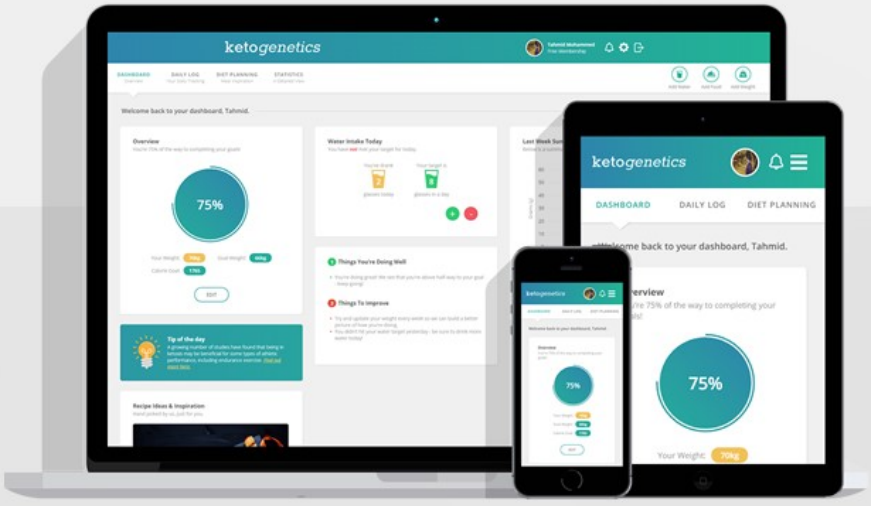
AN ALL-IN-ONE RESPONSIVE KETOGENIC DIET WEB APPLICATION THAT AIDS USERS BY HELPING THEM MONITOR THEIR CARB AND FAT INTAKE, MONITOR AND KEEP TRACK OF THEIR WEIGHT AND THEIR PROGRESS WITH STATISTICAL/HISTORICAL DATA.

SUPERVISOR

DAMIEN TILL

GROUP 10

MIZNA UPPAL
MOHAMMED TAHMID
DARDAN QUQALLA
OLUWASEUN OSIKOMAIYA
HANNA ROBLE



SOFTWARE PROJECTS IS52018C/S

Goldsmiths
UNIVERSITY OF LONDON

Ketogenetics

Group 10

Abstract

The aim of this project is to create “Ketogenetics” – this is a dynamic, Ketogenic diet-based web application. Users will be able to track their calories, macros, meals, weight and water consumption which will aid them in their weight purpose goal (whether it’d be weight loss, weight gain or weight maintenance).

The problem we are trying to solve is providing an application suited to those who follow a Ketogenic diet lifestyle. Currently, there are many weight purpose applications available on the market however none which specifically cater the Keto diet.

Acknowledgements

Our small team would like to thank our supervisor, Damien Till, for assisting us during the project and giving valuable feedback to constantly improve our work. His thoughtful insight helped us keep on track of the project from start to finish.

Contents

Introduction

Background

The origin of the ketogenic diet dates to the 1920s when doctors began using it to control seizures among patients suffering from epilepsy. The treatment was widely used for two decades, but with the modern era of antiepileptic drug treatment[1] its use declined dramatically (W. Wheless, 2008). Simply put, the diet consists of a "nutritional plan made of carbohydrates (carbs), fat, and protein" that eventually forces the body to enter a state known as ketosis. Ketosis is a metabolic state[2] that "forces the body use fat instead of carbohydrates for energy" (Diet Doc, n.d.), which has many proven health benefits, most notably, aiding in weight loss, reducing blood sugar levels, and more[3].

Motivation

After conducting research into the market, we concluded there were few keto-based applications which cater to the users' needs. The consensus was that there was not one application tailored to the thorough-planning of a ketogenic diet. Users would often find themselves having to download several applications, responsible for different aspects of the diet, such as meal planning, carb-tracking, etc. This was not ideal, evidently shown by results collected in the first term, as 66.7% of users in our questionnaire stated they do not use any keto-based application. Instead, the users were seeking an application which would incorporate all the features of those applications into one.

The initial idea was to create an application for those who follow a ketogenic diet and those interested in starting one. After getting in touch with keto-dieters on Reddit, we gathered opinions and ideas from both beginners and experts of the keto-diet. From that, we were able to summarise crucial components that are missing on the market and proposed a system which would aim to solve issues encountered by users.

Users pointed out that many available applications contained static pages, like a hard-copy cookbook. They were not personalised to the user at all and offered little to no interactivity. Instead, they served the mere purpose of providing information on the keto-diet and meals only. A few other applications existed which did provide the necessities for keto-dieters but required payment after installation. In addition, applications available on the market were only available for mobile devices and did not provide desktop-preview capabilities. These were the notable problems we were trying to solve throughout the development of our application.

Scope and objectives

The main objective of this project is to create a ketogenic diet-based dynamic web application which is usable, intuitive and functions well to aid the user in achieving their weight goals. The primary concern is to develop an application that will integrate the front-end and back-end system seamlessly together in hopes of maximising efficiency, responsiveness and aesthetics.

The core focus of our application, Ketogenetics, is to create personalised data profiles for each individual user. These personalised data profiles will be stored on a database, which will then be used by our application to automatically compute their targets for the day.

Users will be able to continuously update their data, such as current weight and daily macro intake, which will be logged and saved onto our system. Their data will sublimely be converted to a graphical representation on our system for the purpose of helping them understand their diet better. Furthermore, users will be able to view numerous recipes available online and their respective nutritional values, and also save their favourite ones.

Project Plan

The following are tasks are compulsory for this part of the project:

1. Analyse our previous primary and secondary market research.
2. Reviewing our designs and prototypes
3. Research technologies to aid the implementation of our web application
4. Decide on a methodology to follow and propose a development plan
5. Implement the front end of our application
6. Host our application
7. Implement the back end of our application
8. Implement methods to fulfil core logic layer of application
9. Testing the application
10. Evaluate the application
11. Research and propose future strategy for marketing

Research and analysis

In order to communicate with our audience better, the entire team researched using various websites and articles, to gain background knowledge of our concept; including learning new vocabulary and science of the Keto diet. This supported us when we summarised the requirements. It also helped us grasp the ethical issues which could arise from implementing this application (e.g health concerns) and measures we could take to protect our application and users.

Stakeholders

We were able to find different types of stakeholders who could offer valid feedback at different stages of our project. We found some on the “keto” and “ketouk” subreddits. Since there is a large audience available on there (1.3m on r/keto and 8.6k on r/ketouk); finding members who we could communicate with was easier than expected. We also found some stakeholders who we could communicate with face to face; some of these were either on the Keto diet, had been on it previously or aware of it to some extent. There were also some who were interested in using the application but were novice to concept of the Keto diet.

Research for requirements

We started with secondary research to understand our market before communicating with individuals. Our primary research allowed us to get to know our clients better and understand what they are looking for in our application. Doing this face to face allowed us to ask questions and get immediate responses; it gave our clients a chance to understand what we were trying to propose better.

The following table briefs what we did and what we learnt during research.

Type	What we did	What we learnt
Secondary	We analysed features that were commonly found on most fitness applications. We compared our competitive Keto applications and other non-competitive diet applications.	This allowed us to grasp the features which must be compulsory for our own product and add them to our list of requirements.
Primary	Communication using face to face interviews and surveys with stakeholders. We presented them with features for our application that are currently not on the market.	We gathered whether the implementation of certain features would make them likely to use the application. This allowed us to summarise requirements which they require and those they do not need.

We posted surveys on reddit for our other stakeholders to offer insight to our proposed system. Following this, our team was able to build a list of requirements for the application for the users. It was during this stage where we decided to eliminate many of the features we had planned to give to our application. A major change was that our exercise and diet planner application turned into a meal planning diet application.

System functionality

Final system requirements

Our system requirements were broken down into functional and non-functional requirements. The functional requirements are the items which are needed to ensure our system will execute as intended; they describe what the system should do. The non-functional requirements demonstrate how the system should behave and specify the system's quality characteristics.

The table below shows the essential and non- essential requirement used these to create our system requirements. They hold high value since they are demands from our stakeholders.

SRE ID	Requirement	Comments	Priority: H/M/L
SRE1	The application must be accessible on both desktop and mobile.	Application will be responsive and will work with platform/screen resolution.	H
SRE2	The application must be able to add food to users' daily log and give them control on custom foods they want to add.	Application will offer a way of manually inputting foods and manually adjusting calories, carbs, fats and also manually adjusting serving size.	H
SRE3	The application must be able to provide users with daily tips and tricks and provide feedback on how they're getting on.	The application will have a dedicated tips section as well as an improvements section where it will show users what they're doing well and thing's they can improve on.	H
SRE4	The application must be clear and accurate when displaying unit quantities.	The application will have accurate, up to date information at all times.	H
SRE8	Application must be able to track user water consumption	Application will be able to track water consumption. Users simply are able to add a glass of water at the tap of a button.	H

SRE1 – Accessibility

We want to make our application accessible to a wider audience; this means every single user, regardless of circumstance will get a similar experience. According to WHO (World Health Organisation), over 15% of the world's population have some form of disability. Majority of the Ketogenic diet-based application available on the market only cater mobile users therefore we need our application to be displayable on both mobile and desktop to ensure we are able to accommodate visually impaired users as well.

SRE2 / SRE3 / SR8 – User profiles

Our application will allow users to update the food they have eaten, the water they have consumed and their weight. The information they input onto the application will be stored to create the user's personalised profiles. This way the user will be able to track their progress and obtain feedback with how they are getting on. Our application will consistently take data from users to recalculate their macros for their specific goal.

SRE4 - Measurements

Majority of the features in our system require measurements therefore we need to clearly specify the unit and quantities for every piece of data we collect from the user. For our application, we will collect a mix of metric and imperial units and specify this to the user to ensure accurate data entry.

Functional requirements

Majority of our desktop requirements will be incorporated onto our mobile requirements.

In order to use our application, users will need to register/log (REW1/REM7/REM8). We will have a separate registration and login page with forms that will collect user's data and store onto a database (REW13). We will use this information to create personalised profiles for the individual. Users may choose to have more than one account therefore we are incorporating a functionality to let users "log out"; we will exhibit this feature through a button (REW5/REM11).

Users will have access to change their information and delete their account; this will be portrayed through a settings page (REW4). Majority of the user's data will be collected throughout the application to create graphical representations of the individual's progress; most of this would be overviewed on a "dashboard" (REW6). We will collect the data through forms and buttons (REM9) and display the graphs on dynamic "box-like" tiles (REW11). Tracking their journey will not limited to just the dashboard – users will be able to track their macros on a "daily log" page and the rest of historical data through a "statistics" page (REW7).

To fulfil our diet planning requirements, we will assemble recipes from popular Keto based websites and incorporate the data onto our own application (REW9). Users will be able to add these recipes onto their profiles through the diet planning page (REW8); they will also be given the option to add their own food items.

We have some extra requirements for the mobile version of our application such as ensuring the tiles on our application will fit regardless of screen size (REM1). The application will show a menu bar that can be accessed upon swiping (REM12). We have removed extra functionalities such the app being able to send reminders and notifications to users.

Desktop Functional requirements

The table below shows only the high priority requirements for this application from our project proposal (the medium and low priority ones have been removed). The priorities are based on the importance of the task and what users are expecting from this application. These are the set of requirements we are focusing on completing.

REW ID	Requirement	Comments
REW1	Must have a login page with registration option	This will need to be the first thing users see when they use our application
REW4	Website must contain a settings page	Any changes made should be made onto the database as well
REW5	Website must contain logout button	Logout option available for users otherwise keep user logged in on mobile and not on desktop when

		browser closed
REW6	Website must contain a dashboard that has the user's selected and personalised tiles/features	Dynamic, tiles must be flexible and display must be modern and exciting to use
REW7	Website must contain section dedicated to statistical information	Will display all the historical and current data in tiles
REW8	Website must contain page dedicated to meal planning	Will display user's breakfast, lunch and dinner with caloric needs and details regarding the meal itself Users should be allowed to make meals manually as well
REW9	Meal planning page must contain recipes which can be previewed on separate page	Users will be able to preview any recipe that they want to from the meal planning recipes page
REW10	Website will have a daily log page where they can see their day's progress	Users will be able to edit their daily logs and profiles which should be changed accordingly in database and other places on website where there is connection
REW11	Website will contain "tiles" with relevant user details	Water consumption tile, weight tile, food intake tile etc
REW12	Website will contain sections dedicated to water consumption	Water intake monitored and saved in database and presented to user graphically
REW13	Website will be linked to a database	Users' data profiles, their history, caloric and carbohydrate intake will specify meal

Mobile Functional requirements

The table below shows the core requirements for the mobile version of the application.

REM ID	Requirement	Comments
REM1	Tiles will fit regardless of mobile screen size	Must be clean, minimise user text
REM7	User will be asked for login with registration option available	No guest accounts
REM8	User will need to provide username and password and click login	Same process as desktop; database has all the accounts stored
REM9	Application will have a essential user data logging button available on all the pages	Essential user data: add water consumption, add food and add weight
REM10	User will need to stay logged in whilst using the mobile application	Use cache/cookies Users should also be allowed to log out when needed
REM11	Application will have a log out feature available	User must be able to log back in again
REM12	Application will show menu	Menu must be user friendly and easy to use

Non-functional requirements

We created these requirements once we started creating UML diagrams since we wanted to think more about extra features that are needed in our application which will improve the system's quality such as security, ease of use, scalability and response time.

The table below shows the non-functional requirements our team has decided to consider; it has been altered from the project proposal to show only the requirement and priority (explanation and further detail for these requirements can be found in the proposal).

REN ID	Requirement	Priority: H/M/L
REN1	Performance: System must be fast, and users should be able to access what they want in a few clicks.	H
REN2	Scalability: System will support an increase in number of users. An increase number of resources being used will not affect the performance of our application.	H
REN3	Users will be able to use the buttons and forms even when on mobile.	H
REN4	Capability: The storage and performance capabilities of our system must be extendable, if required by changes in our use cases.	H
REN5	Reliability: Our application should be able to perform, as intended, over a given period of time.	H
REN6	Maintainability: Front-end and back-end system should be developed in a clear structure and format.	H
REN8	Manageability: System will allow users to store several data entries without downgrading performance.	M
REN9	Usability: Application should provide ease of use for the user.	H
REN10	Availability: System must be available 24 hours a day, 365 days a year.	H
REN11	Security: Connection to website must be secure via HTTPs. User data stored must be encrypted.	H
REN12	Platform: Web application must be able to run on popular browsers such as Safari, Google Chrome, Microsoft Edge, Mozilla Firefox, and more.	M

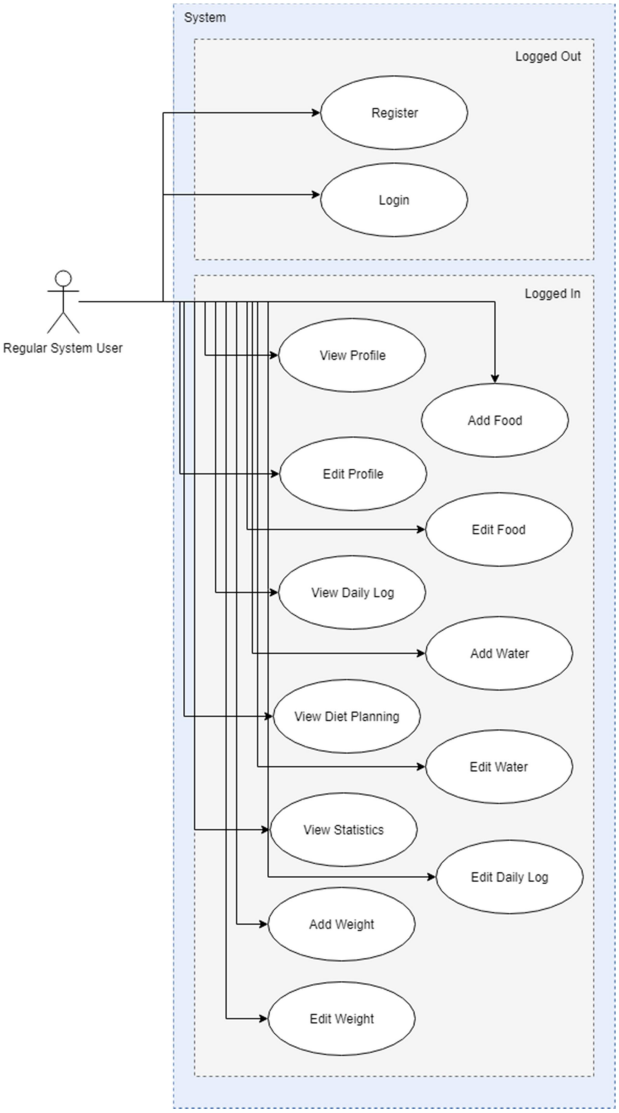
Use case diagrams

We created use case diagrams to model functionality of our system based on the requirements that we made. They demonstrate the relationship between actors and our system; it was during this stage we were able to clarify who would be using our system and for what reason. Constructing these helped our team analyse interactions which could take place. It identifies factors which may influence our system without worrying much about the details of how those functionalities would be implemented.

Login and Registration

This shows the main features an individual user will be able to see whilst using our application.

The following user case below models the actor's (regular user's) interaction when they are logged in and logged out of the system. User has the ability to register and/or login to the system; once logged, they have the ability to perform actions in the system.



Logged out

When they are logged out, the user will only have access to the "login" and "registration" pages.

Logged in

Once the user logs in, they will be taken to the dashboard. The dashboard will be the main page with simplified versions of components from the other pages.

Users will have the ability to add water, weight and food items to their profiles from this front page.

The design encapsulates the ability to view other pages so we will need a navigational menu to aid users around our web application.

Functionality of our web application

The following shows different features the user will have access to after logging into our system. Some of the features from our original proposed use case diagram have been removed based on the time we have to develop.

The following user case diagram below demonstrates detailed interactions of actor (regular user) with the system's features. It also illustrates how each feature is linked to one another.

After the user logs in, the first page they will have access to is the dashboard. There will be features on our application that are exclusive to "each" page whilst some features will be accessible on "every" page of the application. Repetitive functions of our application include the ability for our user to their log weight, food eaten and water consumed - these affect the user's macros and progress hence why they must be placed on every page. The summary for these functions would be provided on the daily log page.

Our user must be able to access every other page on the application regardless of the page they are on. This includes the ability to visit the settings page, diet planning page, statistics page and daily log page. Following this, a log out functionality must be displayed clearly on every page to increase ease of use.

From the dashboard, the user will be able to view different components of the application. It will give users an overview of their progress (their macros for the day, things they are doing well, tip of the day, statistical data). From settings page, they will be able to alter their profile (edit their profile picture, change their goal weight and even delete their account). Through "diet/meal planning page" then they will be able to view, sort, favourite and filter the available recipes. From statistics page,

they will be presented with graphical representations of their progress and goals (track their weight over time, the amount of water they had consumed on a particular date).

Changes to the dashboard functions

We removed the ability to edit the specific tiles you can view; instead we are proposing this concept for future releases instead due to our inability to complete this within time. The "view last week's summary" tile will no longer show a dual graph which demonstrates user's "weight" history with "carb and fat" history; instead this will be a demonstrated as a separate function on our daily log and statistics page. We want a featured recipe tile on the dashboard since we are removing this feature from our diet planning page.

Changes to the diet planning functions

We changed the name from "meal planning" page to "diet planning" page. We removed the ability to "search" for recipes and instead added a new functionality where users will be able to sort and filter the existing recipes. We no longer have a "featured recipe" on our diet planning page; this has been moved to our dashboard.

Changes to the daily log functions

Initially, we were going to base macros off BMI however this goes against purpose of the Ketogenic diet which promotes fat loss. We will collect user's body fat data and apply Mifflin st Jeor's formula, instead of Harris benedict, to work out the macros.

Mifflin st jeor's formula: 10 x weight (kg) + 6.25 x height (cm) – 5 x age (y) + 5

Changes to the statistics functions

We have decided on specific graphical representations for the features which manipulate user's data. Most of this will be applied on the statistics page.

We will use the following graphs to visualise:

- Pie chart for overview tile
- Pie/donut charts for showing user's macros for each day
- Line graphs to show user's macros over time
- Column graph to show carb versus fat history
- Line graph to timeline the weight progress
- Area graph for water consumption over time

Changes to the settings functions

We have kept the feature to change your goal weight on the settings page however users will be able to make use of this functionality on the dashboard as well. Instead of users choosing whether they are trying to gain, lose or maintain weight, they only need to input a goal weight (kg) and system will automatically create new macros. We also removed the ability to change your email. We have added a new functionality where users can delete their account.

Use case scenarios

The following tables depict some of our use case scenarios. It allowed us to capture the functional requirements of our system by considering the various cases a user may experience whilst completing a task. Identifying these helped us plan the visual design of our application and ensuring it was easy to use.

The table below shows cases which could occur when regular users attempt to login to use the system for the first time.

IDs used	ST1/ST2 and AC1
Scenario	Logging in after registration
Actor	Regular user
Cases	User can/cannot locate link to registration page / login page User can/cannot locate button to registration page / login page User's username accepted/not accepted User's password acceptable /not acceptable User's email already in use/ acceptable User decides they don't want to sign up half way User does/ doesn't inputs details regarding personal information correctly
Precondition	User opens the web application for the first time and begins registration
Trigger	User completes registration and proceeds to login page
Post condition	User manages to login to the web application
Successful scenario	User opens the web application and registers correctly and accurately as possible. After registering, they proceed to the login page where they remember their details and login successfully.

The table below shows how user will be able to set goals for weight and know how far they are from their goals. The system should adjust to these changes to create new profile for user.

IDs used	ST7 and AC4
Scenario	Tracking weight and goals
Actor	Regular user
Cases	User sets current weight User wants to edit current weight after setting it User cannot set their current weight, unsure how to User views their weight history User wants to view their weight history within a specified time-period User can/cannot locate section linked to weight history
Precondition	User is logged in and can view their dashboard
Trigger	User sets a new goal weight
Post condition	Users macros (carb, fat, protein and calories) change
Successful scenario	User sets a new goal weight from the main dashboard. The system adjusts the user's macros successfully to help user reach their new goal. The user can still preview and track their previous weight progress.

Development

Technologies Overview

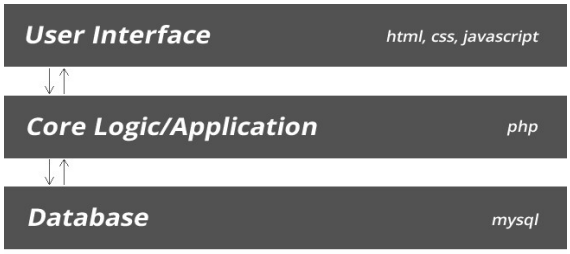
We researched plenty of tools and frameworks to aid the development of our final web application. Since time was a constraint, majority of the ones we settled with were ones we had experience with; this allowed productive implementation for a quality product.

The justification for each technology has been provided below.

Architecture

A layered architecture was used for our web application; the use of layers helped to control the complexity of our project.

The diagram below illustrates our layered architecture; based off the web architecture model.



all layers closed

Our 3 layers consisted of a user interface layer, a core logic layer and a database layer. Each time a request is made, it would pass through each layer until reaching the lowest to be fulfilled. Each layer is closed therefore there is no need to be able to bypass sub-components of the system; the layers communicate with the ones they are closet too. If a user decides they want to add a new weight then a request from the user interface layer is passed to the application layer, this is where any calculations needed would be applied. The result is captured onto the database once it reaches the bottom layer; it is stored for future use for other components of the web application. An example of this would be the storage of user's weight so that we can create a graph which visualises their history of progress (a line graph which showcases the change in weight over certain period of time).

Working with a layered architecture was an advantage to our team since it made splitting the tasks easier and enabled us to work on different parts of the application in parallel. It also allowed us to scheme methods for testing components of the system independently thus saving time planning.

The front end technologies we worked with were HMTL5, CSS3 and JavaScript. Our backend technologies were PHP and SQL.

HTML5 & CSS3

HTML & CSS are web mark-up languages that help layout the basic structure of a webpage. They provided us with the basic foundation for us to build our web application front-end. As our

application was a web-based application and there was a requirement that it need to be responsive (dynamically adjust to all screen sizes and resolutions), it made sense to use these languages.

We used vanilla HTML and CSS to build our front-end from the ground up for the entire application rather than using existing web frameworks such as Bootstrap. We needed maximum customisability and flexibility - due to our layout being complex we would have ended up needing to spend a significant amount of time making extensive modifications to existing frameworks. Vanilla HTML and CSS gave us the level of control we were looking for and helped us build an application that was responsive.

JavaScript

JavaScript is a web programming language that allows you to add dynamic behaviour, store information and handle requests and responses on a website.

We used JavaScript and many JavaScript libraries such as jQuery, FusionCharts, SweetAlert, Salvatorre, Tooltipster and more. Javascript was mainly used to perform calculations and arithmetic operations on the web application such as calculating total carbohydrate, fat and protein consumption, calculating nutritional information and producing user specific tips. Specific JavaScript libraries helped us tackle certain problems e.g for creating charts we used FusionCharts.js.

To make sure our application was fast; we decided to offset things like nutritional calculations to the client-side using JavaScript instead of dealing with them on the server-side using PHP.

FusionCharts.js

FusionCharts is a powerful JavaScript charting library that has a wide range of interactive charts and data-driven maps which allows you to create responsive charts in real-time. When sourcing a charting library it was important that the charting library was responsive (as our application is responsive), the charting library was highly customisable. The library could be integrated with our existing programming languages and software stack; it was easy to use and had good developer support. FusionCharts was the only library that met our criteria. It provided an unprecedented level of customisability so that we could match the charts with the theme of our web application and make them look like how we wanted. It also was responsive and had quick integrations with all our tech stacks as well as having comprehensive documentation, tutorials and guides. We used FusionCharts throughout our web application (on all main pages) to display and visualise user stats such as user macros for the day, user macros overtime, user fat vs carbs history, water consumption overtime and more.

SweetAlert.js

SweetAlert is a JavaScript plugin that allows you to create customisable modal popup boxes. We needed a clean way to inform users of errors without being obtrusive and a way of getting input data such as meal information. SweetAlert boxes were used throughout the application for user prompts and for users to quickly enter in data without having the need to go to a separate page/form. We picked SweetAlert as it fit our application theme and design and because it’s responsive and highly customisable.

Salvattore.js

Salvattore is a jQuery masonry layout grid. We used Salvattore.js to create the main grid of our dashboard and all other pages. Salvattore is responsive so automatically adjusts the grid to suit different user screen resolutions which was key - as our application needed to work on all devices and browsers. Salvattore saved us development time, creating a masonry grid layout from scratch would have taken up a big chunk of development time. It was easy to use and allowed us to create customisable rapid grid layouts that worked well on mobile and desktop.

PHP

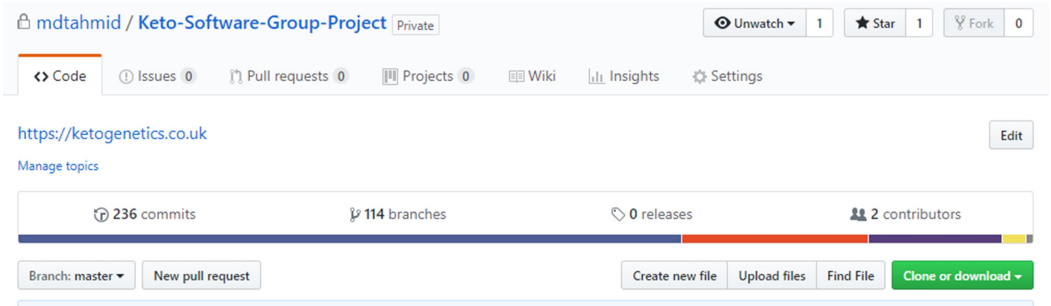
PHP is our back-end scripting language. PHP allowed us to merge our front-end UI with our backend database. It created the link between our front-end and back-end and allows us to interact with our database. We've decided against alternative technologies such as ASP.net because we need compatibility and support across all platforms and environments. Technologies such as ASP only support Windows based platforms and as our server is Linux based (cPanel) instead of Plesk based (Windows) it made sense to use PHP. PHP was used throughout our entire application - it manages everything from the user login/registration system and user sessions to fetching statistical information from the database so user graphs can be produced. It along with SQL powers our core features such as food logging and water tracking.

SQL

Our database of choice is SQL. This holds all the data for our web application from user registration and login details to user water consumption and food logs. We used SQL queries combined with PHP to perform CRUD operations such as retrieve user data from the database and perform calculations based on them such as calculating user calorie intake. We've chosen SQL over competing alternatives such as MongoDB as in our initial design we identified that it was important that our database was to be a relational database with column/row-based structure. Furthermore, SQL is fast, already has existing well defined standards, requires minimal coding and has great documentation and support for php (our backend scripting language).

Git and GitHub

Git is a version control system tool that allows you to manage source code history. We used GitHub which is a web-based git repository management tool to manage our code and code flow. GitHub helped us stay organised and allowed sharing source code amongst all team members.

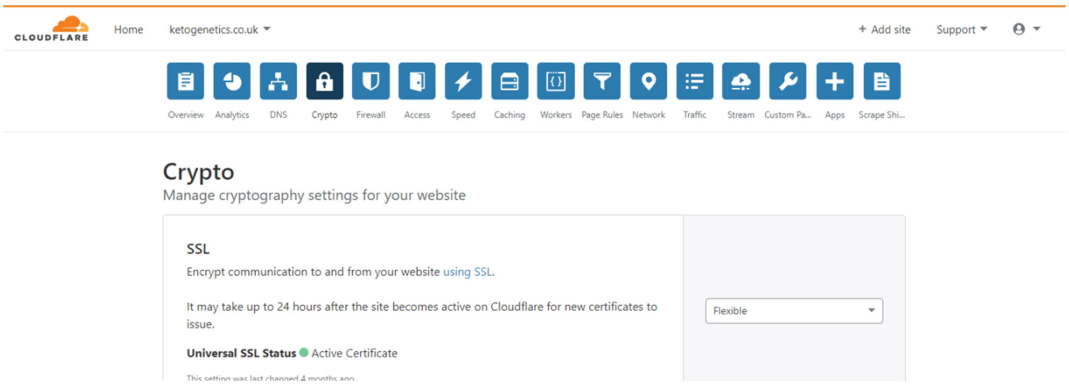


Buddy

Buddy software assists DevOps by automating code delivery and deployment. We used Buddy to create a pipeline between our GitHub repository and our GoDaddy servers to automate file upload to the servers. This made development streamlined as we had over 230 commits and manually updating servers each time would be time consuming and tedious. We used Buddy as it was easy to use and had support for the both our code management/version control platform and our hosting platform that we were using (GitHub and GoDaddy).

Cloudflare

Cloudflare is a DNS management service and CDN service that specialises in DDoS mitigation, website security and DNS services. We used Cloudflare to manage our DNS instead of our hosting provider (GoDaddy) as Cloudflare provides free SSL which was important to us since our application had a user registration system and needed to be secure.



Cloudflare also gave us fine grain control over our DNS such as the ability to create page/url rewrite rules and cache control – which helped speed up development. We used Cloudflare as it is free and integrates directly into our technology stack/web application.

Methodologies

Our team looked into different models before deciding on Agile. We needed an environment where we could keep looking for flaws and improve on them.

Waterfall model

This popular model was our first considered alternative. After research, we learnt it is too rigid for our project due to its linear and sequential design. It requires each phase to be completed before moving onto the next one which means that fixing our mistakes would be expensive and time consuming since we would test our product after the development of our application. This model was also unsuitable for our project since it does not accommodate a change in requirements.

Spiral

This splits the lifecycle into small parts and prioritises the sections with higher risk concentration. The model offers scalability and encourages earlier prototypes to be created however the requirements are not clarified. Unfortunately, this model requires involvement of highly skilled professionals whilst the ability of members in our team is a mix. It is also ineffective for small projects.

Agile

Our project followed an agile approach; we were constantly changing and evolving our requirements with different solutions. This model let us make corrections to our requirements to provide competitiveness whilst staying client-orientated.

We strived for continual improvement and had to be flexible with the idea of change. Most of the software development was completed through adaptive planning to ensure we could provide fast release of a working product that would satisfy our users.

Problems with Agile

Agile methodology promulgates a customer-centric approach to building software; our team responded to every request of our stakeholders. During our project, we were dynamically responding to change and not optimising for efficiency. Putting metrics on our work made us feel like we were progressing however it didn't mean we were doing as much as we could. We were contributing to our goals and strategizing ways to work faster whilst creating low impact unsuccessful material.

Our team was slightly able to handle this problem after an adjustment phase, especially when we started to let tasks go through carefully considered decision making. It meant there was no linear progress with our development; in some weeks we completed a lot of work whilst during other weeks we didn't do anything. However, on the long term, we were creating a high impact product.

Procedure

Preparing and planning

We split the team into two groups, 3 people to focus on development and the other 2 for testing.

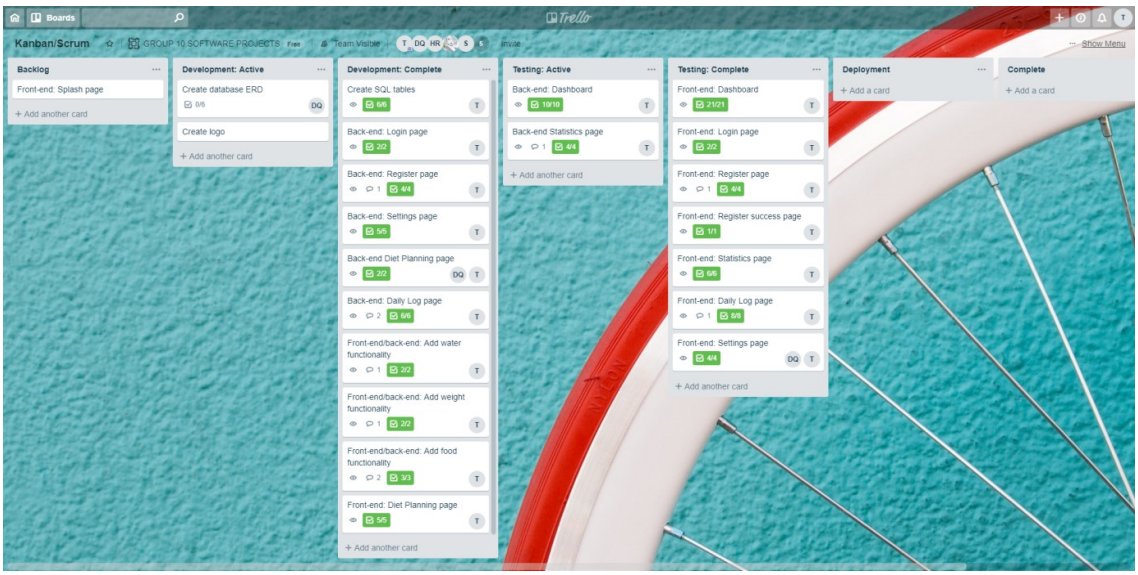
Due to the complexity of our project, we started during the Christmas break to allocate ourselves an extra 2 weeks for development. We used this valuable time to focus on researching and implementing solutions for the requirements. At the end of each two weeks, we could collectively report the areas we have worked on and tasks that have been completed in each team. We planned our next set of realistic and time manageable goals.

Scrum

Allocating approximately 6 weeks for development was not enough time to complete most of the objectives we had set for ourselves. With Scrum, we focused on 2 week sprints with exactly 3 sprints on total. At the end of each sprint, we evaluated and prioritised our aims for the next sprint. These were subject to change every time since there were always some tasks that were left incomplete from beforehand. Starting development 2 weeks earlier gifted us with extra time to embrace failure and learn from our mistakes whilst implementing our web application.

The Scrum Master created our Scrum/Kanban board on Trello with goals for the development. These were split into “front-end” and “back-end” tasks. Summaries and checklists were put onto each task card and assigned to the team members. Once it was complete, the member would tick it off from the pool; this way the others would know which tasks are remaining. If more than one member wanted to work on the same task then they had to communicate and work together on it.

The following image is a screen capture of our backlog.



Advantages of using scrum

It improved our development planning skills. It motivated us to prioritise what we needed to complete and manage our deadlines to ensure delivery of critical requirements. The sprints allowed us to acknowledge how much we had actually completed and if we needed to change our strategy at any point to get a quality working web application.

Disadvantages of scrum

The scrum master had a difficult role since they would need to add tasks and frequently change aims if team members were not completing the work. Scrum assumes that team members work with their own initiative and does not take skills into account. If a team member was unable to do the work, then it would be assigned to another member however this meant that particular members were doing more work than the rest.

Adopting scrum roles, rituals and values was bothersome since we weren't doing it right all the time. We were incapable of doing daily sprints and our reviews were published on alternate days with half done evaluations at the end of every two weeks. At some points in implementation, we weren't following any protocols instead just focusing on completing the proposed backlog.

Development log

We created and shared a development log which testers and developers checked and updated regularly. It allowed our team members to review what they have been doing and what needed to be done for the web application. This wasn't completely reliable since many times not everyone remembered to update everything.

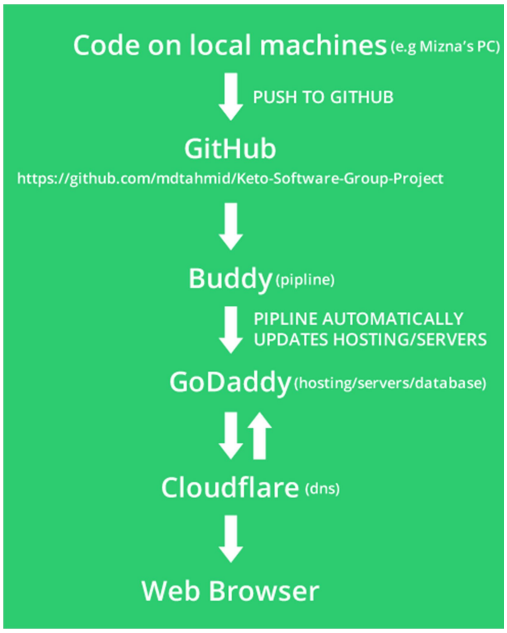
The following table below is an extract from our log of development.

Date	Description	Comments (things done well/ to consider/ to add or to fix)
16/12	Initial start-up file; includes HTML and CSS for Ketogenetics application. The layout is being decided; it contains the header and allocated spaces to add logo and other icons; there is minimal styling applied.	HTML and CSS is not enough for this web application; need to look into JS libraries: superslide, tooltipster.
17/12	Header navigation and menu added; used HTML and CSS to create. Images and new font had been added as well. Superslide and tooltipster has been added on but not used yet.	Keep code clean and wrap into containers
18/12	Mobile menu has been added using superslide.js	Stick to px for sizing and fix CSS using beautifier

Development environment

Our development environment was quite straight forward. We had a fast, streamlined development cycle so we could quickly push our new features and maximise all the time we had in the project.

The diagram below shows the flow of development.



Each person in the group coded on their local machines on each individual features of the application. When the feature was complete they would push to our GitHub repository.

Our pipeline at Buddy would automatically detect this push and upload the newly pushed files on the GitHub repository to our GoDaddy servers. The updates could be seen by going to: **ketogenetics.co.uk** on a web browser.

When changes were made to the application, the testing team would be alerted.

Our boards on Trello and all logs would be updated according to tasks which are complete. Members would be alerted to retrieve newly pushed files by using “git pull” before another “push”.

Changes to environment later into development

Some members found the management environment confusing so development leader took on an extra responsibility. Members would send the code to the dev leader who would manage the code and overlook the tasks. They would also make amendments when necessary to keep presentation and regularity of code consistent.

Managing tasks

In the previous term, we had some issues as a team with completion of tasks in accordance to deadlines. There were also issues where members knew what to do but weren’t sure how to do the tasks based on skill set and ability. Having a split team with two leaders meant the team members could work more closely with someone who could guide and monitor them with the tasks.

Human Resource Management (HRM)

Our small team of 5 meant that at various times during development, switching around was necessary to complete the tasks to ensure we were utilising everyone's skills. This meant there were times where members from the development group would aid in testing and testing members would aid in development. However, everyone knew which area they must prioritise and spend more time on.

The table below shows the assigned teams.

Name	Main team	Reasoning	Area of focus
Mizna	Testing	The member said they were comfortable with testing back end of the implemented solution and overseeing the overall testing of the product.	Leading the testing of the product, when switches were made; the switched member would refer to her for templates and guidelines. Member needs to research the testing methods that are required for this project and summarise a list of test cases for both mobile and desktop; plan out testing techniques and when they should be executed and conduct some user testing (member needs to ensure that all the results are logged and being provided to the development team who can review them and make changes to the application if needed).
Tahmid	Develop	The member said they were comfortable with creating the front end of the web application; he played a vital role in designing and demonstrated these extraordinary skills again in development. They will oversee the development of the product.	Leading the development of the product; upon switching, the development leader would point out tasks which needed to be completed; they would then be shown to him before he would upload it. He had to keep in touch with the testing leader for ensure everything was tested. Main focus must stay of the front end of the project with implementing the application ensuring the prototypes/ design matched final product.
Hanna	Testing	Member explained how she felt more comfortable with testing the product and researching and did not feel confident in the quality of her coding. She focused more on testing the front end of the application.	Main focus must stay on the front end of the application; with testing of individual pages, checking for broken links and checking the overall application was working as intended.
Dardan	Develop	Member said he felt comfortable with helping on the development of the project; he was comfortable with the tools that we	Will focus on developing the water consumption tiles and other tasks which the lead developer will assign during the actual development. Member will be assisting with merging of front and back end.

		planned on using.	
Seun	Develop	Member said in the previous term that he wanted to help more with development and technical elements of the project.	Will focus on developing the things to improve bits and other tasks which the lead developer will assign during the actual development.

Reflection on development

Agile was flexible and allowed us to make rapid changes throughout the development and testing phase. In the end, we were able to produce a working product that had been extensively tested and improved before user testing.

Problems faced during development

Lack of face to face communication was an underlying problem that our team failed to address during development. We organised to communicate more and vowed to phone calls and Skype conferences. Despite the encouragement, these pre-planned calls would be cancelled due to lack of member availability. At the start of the project, everyone spent a lot of time outside of labs and lectures to stay up to date with project however with other deadlines, assignments and exams this slowly declined throughout the year; it was at its worst during development and testing. We could have resolved this through regular reviews of our human resources and enforcing frequent meetings.

During development, we learnt that it is very easy to be left behind. There was a specific case where a member felt left out during the project which discouraged them to contribute and stay up to date. None of the team members were aware how serious this was until the end of development. Our supervisor guided us to brief all the members once more to keep everyone up to date. This can be avoided by sending regular briefings not just via Whatsapp/IM but other mediums such as email as well – to ensure everyone is getting informed.

What we would do differently

Our team relied heavily on digital communication instead of real time communication and collaboration tools. This led to time wasted where one member would be waiting on another member's replies – this increased the risk of not finishing critical tasks. We must focus on frequent meetings and face to face conversations to avoid this.

Splitting the team to complete the tasks did not work as effectively for us as we thought it would. Dividing a small group is not as efficient when using an agile approach. The split led to miscommunication between team members since we mainly focused on the tasks which were provided in our teams and overlooked work done by the other team. Switches were also made which meant that there wasn't much consistency in following our development plan.

In the future, we wouldn't bother with splitting the team since this will clarify the roles better; team members would gain more independence with what they produce as an individual for the team rather than for what their "half of the team" produces.

Design and implementation

Proposed design

Changes based on UML review phase

The appendix (appendix B.1) contains an overview of the necessary UML diagrams to understand the behaviour and structure of our web application.

The following table below summarises some of the important changes worth discussion that we considered for implementation.

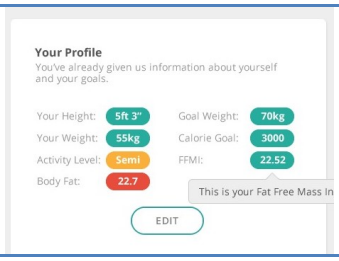
Issue we found	Solution
Class diagram issue: We learnt if the user does not add to the system, then they may not actually be able to take as much from it.	We needed to create elements which forced users to input data that would allow us to create a profile of them. This should be done at the start preferably when the users sign up to use our application.
Component diagram issue: Some components will affect majority of the application; making the application more complex. We have no way of determining accuracy of the application if little things like food input would change user's progress, macros, things to improve and other tiles.	We need to ensure that some features stay independent to avoid making the application too complex. We chose to remove the complexity of food since it is easier to link other components such as water intake and goal weight (these do not affect the application complexity as much). Users will be able to add food however this won't disrupt our system's analysis on their progress (for now).
Activity diagram: there isn't just one way to complete a certain task.	The behaviour of the user with the application could change if we were to provide the same features on different parts of the application. It is repetitive but makes the application easier to use.

Changes based on functional prototyping review

The appendix (appendix B2) contains a summary of the methodology of our prototypes and templates of our designs.

The following table below utilises feedback from the users. It shows some of the problems users found and our approach to the issues.

Feature	Problem	Improvement
	Users want to be able to see their goal weight on the dashboard. The overview tile shows progress but not actual goal.	Users should be able to see their weight, goal weight and calorie intake to reach that goal on the overview tile.

	The front dashboard shows the user's entire profile; this is the first thing they see every time they log in which can be seen as unnecessary.	We will remove the "full profile" tile from the dashboard.

The following table below shows some of the features our team would like to address after presenting prototypes. We made these changes upon judgement of our team's ability to implement certain features.

Problem	Change
Meal planning page contains recipes and when you click on a recipe, it should take you to full page with recipe in detail. However this will be too time consuming to implement and there can be issues concerning copyright since we are taking these recipes from other websites.	To avoid copyright and still give our audience access to recipes for meal planning, we have decided to link the recipes to the websites we are taking them from. These are credited websites such as "diet doctor". Instead of the user being directed to our own website page with the detailed recipe, it will be to the website with the recipe link.
One of the tiles on the dashboard is supposed to promote a recipe of the day; this means a new recipe every day. Either someone has to manually change this or we needed to develop an algorithm that takes a random recipe from our diet planning page and this would change every single day but there is an issue that the same recipe would keep appearing despite randomisation since we do not actually have a large pool of recipes.	Instead of recipe of the day, we decided to change this tile to promote the diet planning page for now. We decided add a carousel; the slideshow would cycle through "featured recipe" and a couple of new recipes.

Overview of final product design

Our team were fixated on creating a unique and fresh user interface as demonstrated through our final prototype templates and our final product. There is a consistent blue and green theme throughout the application to promote our health based application; this is made prominent by adopting a grey and white background. To keep our system clean and tidy, we opted for a special grid and tile feature which aids the separation of graphs and data representation.

The designing of our application is minimal and optimised to assist the functions. By plugging the main input collection buttons at the top of the application (and through "add" button for mobile), we are able to encourage users to make data entries which will enhance their experience. We played with "dark text on light" background and "light text on dark" background to certify the text is clear and easy to read.

We decided against excessive images and unnecessary explanations throughout the application to avoid an overcrowded user interface; this helped to maintain our user's focus. Our graphical representations are illustrated with small animations when users preview them. Our application is capable of leaving tips and pop-up boxes to help users navigate around our application.

Implementation

We will summarise our implementation process; from our approach and how we used our experience/results from previous stages to create a quality application. We will also discuss the problems we had during implementation and how we overcame this.

Implementation process

The process we went through to implement our application:

1) Front end: dashboard, login page, register page, success pages, statistics page, daily log page, meal planning page, settings page.

We focused on the front end of the application; in particular, the designing and layout of the application. After setting up the structure for our dashboard, we had a basic template that we used to build the other pages of our application.

2) Back end: SQL tables, login page, register page, diet planning page, daily log, settings

After implementing most of the features of the front end of our web app, we worked on the back end of our product. We knew how to create ERD diagrams from our designing phase; so developed our schema quickly and used the designs to create our SQL tables.

3) Front end/Back end

We worked on merging the front end functionalities with our backend database using our scripting technologies. Our focus was to allow users to add data to their profiles as implied in the requirements such as "adding water" for each day.

4) Extras: logo, touch ups and completion

We worked on extra features such as the "light-box pop up" forms and making sure that all the features of the application that we had started to work on were complete. Towards the end, we worked on our logo – though logos are usually important and can act as a trademark; our team paid more attention to technical aspect of our app such as ensuring macros were updating correctly for our users and meeting other basic requirements.

The technologies we used for the implementation of our web application have been specified under "technologies overview" in development section.

Project structure

Developers were repeatedly reminded to focus on splitting the code and sorting it into separate files and folders; doing this helped other team members locate features of our application for testing and when changes needed to be made. Presentation of code was also emphasised to reduce errors and

make code manageable. This allowed us to work on different parts of the application at the same time.

Front end implementation

We started with the implementation for the front end of our application using HTML5, CSS3 and JavaScript. Members used their own preferred text editors, ranging from Atom, Brackets and Notepad to help spot syntax errors; this made being able to pin point parts of the code that needed addressing to each other easier. Being able to "live preview" the application allowed us to visually see the changes we were making whilst coding, which sped up our implementation.

Changes to the front end

The designing for the front end of our application was complete during the prototyping phase. We proposed a clean, modern user interface however many of the features were complex so we made a lot of modifications. Some of these changes are noticeable such as our decision to go with the initial login in and registration pages designs (as discussed in our problems section below); whilst other improvements were quite small and specific to each tile on our application. Overall, we managed to enhance the ease of use through these adjustments.

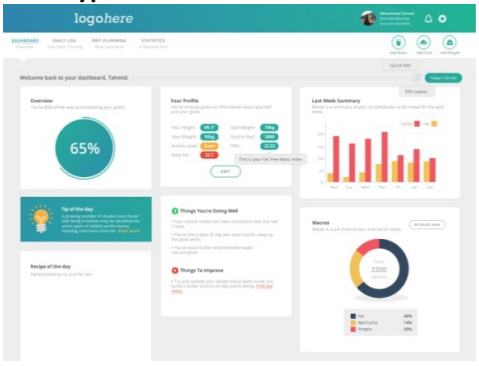
On our backlog, we proposed a splash page to make our application exciting every time users open our application however whilst implementing we found this was unsuitable for a web application since it would be annoying for a user who will use the application repeatedly; therefore removed this feature.

We didn't implement every single feature to the "maximum". Instead of producing a few quality "tiles", we tried to present all the tiles on the application and then worked on improving them after. Majority of our changes were made due to lack of knowledge on how to use certain tools to incorporate a certain function or because we did not have enough time to develop.

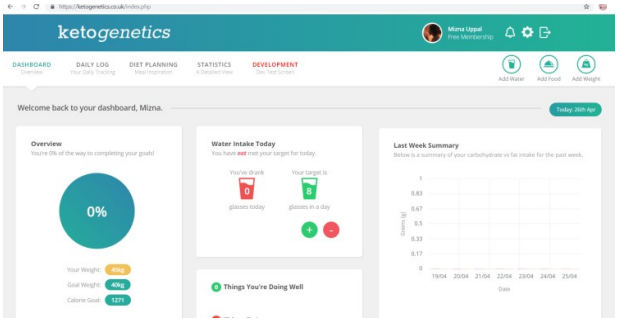
Final design product

The first part we created was our dashboard, this was supposed to showcase majority of the components of our system. The code for our dashboard acted as a template for the other pages; this made implementing the rest of the application easier.

Prototype dashboard



Final dashboard



In appendix (B3.1), you will find a comparison of the prototypes and UI screenshots for the final Ketogenetics product.

Back end implementation

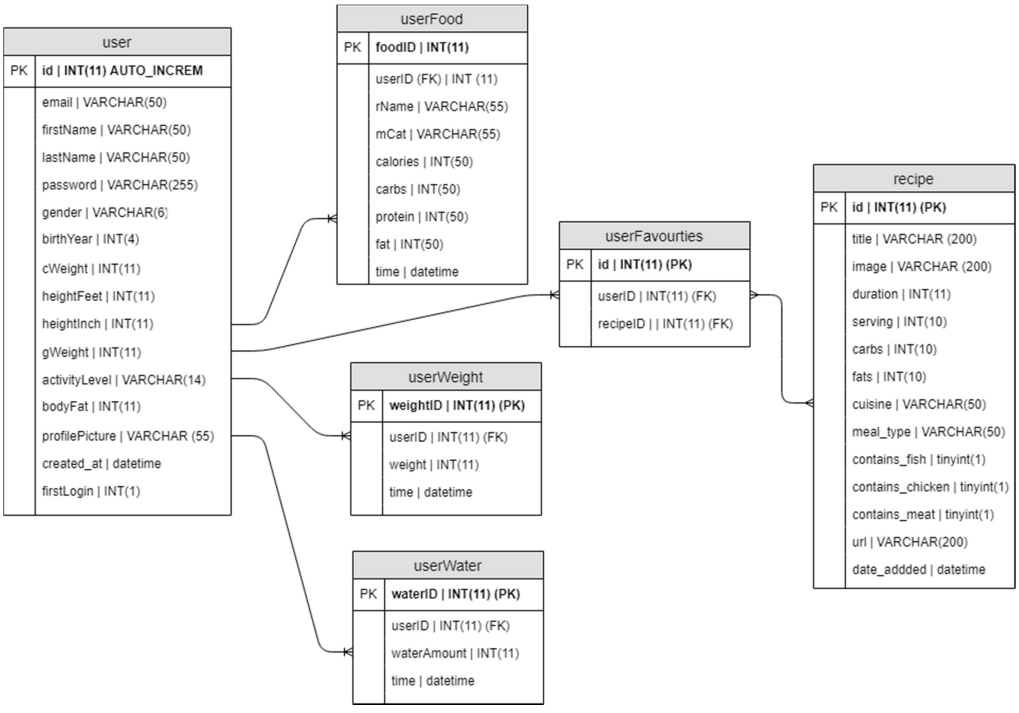
Our relational database of choice was SQL; it had a column/row based structure that allowed us to hold the data collected from the web application such as the user's personal details and progression data. We were able to manipulate the input information and perform CRUD operations. It was fast, required minimal coding to use and offered great support and documentation. Our server is linux based therefore we opted for PHP as the back end scripting language. PHP allowed us to merge our front end user interface with our back end database.

ERD DIAGRAM

We created this to finalise what information we will need to store in our database; we were able to identify entities, attributes and entity relationships. Creating this blueprint gave us a chance to recognise errors and other constraints. We were able to simplify our model to work out multiplicity of relationships before actually creating our database and doing any coding.

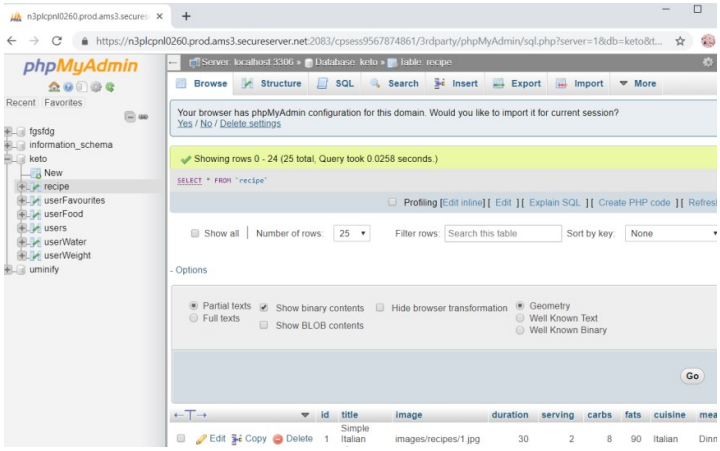
A higher level overview for this diagram has been provided in the appendix (B3.2) to help understand it.

The following ERD diagram shows the relationships between the different tables.

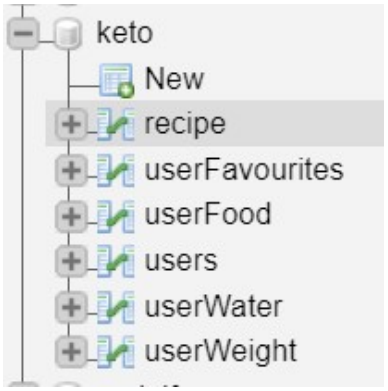


Building our database

We followed the schema to create our tables that would be capable of storing data needed for our application. This includes the user's data and recipes for diet planning.



Tables



The following recipes are stored under recipes table in our database; these will be used directly to aid users in their diet planning.

	id	title	image	duration	serving	carbs	fats	cuisine	meal_type	contains_fish	contains_chicken	contains_meat	url
Edit Copy Delete	1	Simple Italian pizza	images/recipes/1.jpg	30	2	8	90	Italian	Dinner	0	0	1	https://www.pizza
Edit Copy Delete	2	Pancakes with berries and whipped cream	images/recipes/2.jpg	25	4	5	39		Breakfast	0	0	0	https://www.pancakes-b
Edit Copy Delete	3	Simple beef lasagne	images/recipes/3.jpg	70	6	9	76	Italian	Dinner	0	0	1	https://www.lasagna
Edit Copy Delete	4	Coconut porridge with fresh berries	images/recipes/4.jpg	10	1	4	49		Breakfast	0	0	0	https://www.coconut-po
Edit Copy Delete	5	Mushroom omelette	images/recipes/5.jpg	15	1	4	43	British	Breakfast	0	0	0	https://www.mushroom-c
Edit Copy Delete	6	Keto garlic bread	images/recipes/6.jpg	60	20	1	9		Lunch	0	0	0	https://www.garlic-bre...
Edit Copy Delete	7	Battered hotdogs	images/recipes/7.jpg	45	4	7	68	American	Snack	0	0	1	https://www.mummy-dog
Edit Copy Delete	8	Baked salmon	images/recipes/8.jpg	20	4	3	90		Lunch	1	0	0	https://www.

The following shows us configuring our database locally in the earlier stages, using PHP.

Line 2 to 5 show the database credentials under the assumption you are running MySQL server with default settings. Line 7 shows us attempting to connect to a MySQL database and then checking the connection throughout lines 9-10.

```
1  <?php
2  define('DB_SERVER', 'localhost');
3  define('DB_USERNAME', 'tahmid');
4  define('DB_PASSWORD', 'softwaregroup15973');
5  define('DB_NAME', 'keto');
6
7  $link = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_NAME);
8
9  if($link === false){
10     die("ERROR: Could not connect. " . mysqli_connect_error());
11 }
12 ?>
```

Giving our application functionalities

Calculating user's macros

The collected input data is stored into variables on our front end, where we were able to apply our core logic through basic arithmetic knowledge and simple formulas.

The following code below is storing the user's gender and height.

```
640        var gender = "<?php echo($_SESSION["gender"]); ?>";
641        var heightFeet = "<?php echo($_SESSION["heightFeet"]); ?>";
642        var heightInch = "<?php echo($_SESSION["heightInch"]); ?>";
```

The following example includes Mifflin St Jeor Equation from our analysis. This allowed us to create the user's macros (these are the user's calories and carbohydrate/protein/fat intake).

The following if statements are to create calorie goals using the Mifflin St Jeor Equation.

```
656        if (gender == "Male") {
657            console.log("This account is male");
658            //This is the base TDEE
659            var calorieGoal = (10 * (latestWeight)) + (6.25 * (heightInCM)) - (5 * age) + 5;
660        } else {
661            console.log("This account is female");
662            //This is the base TDEE
663            var calorieGoal = 10 * (latestWeight) + 6.25 * (heightInCM) - 5 * age - 161;
```

Calculating user's goals

To help users with their progress, they were shown "things they are doing well" and "things that they need to improve".

Below are examples of messages the user will see on the user interface.

Our application takes the user's water consumption and informs users whether they are meeting their water intake goals. If user is performing well, their overall "achievement" percentage increases; if this is above or below a certain threshold, the user will get a message on their "things to improve board" with the advice.

The example below shows that if the user hasn't drank 8 glasses of water the previous day, they will be told to drink more water on their "things to improve".

```
753        if(newlySigned == false && waterYesterday < 8) {
754            addToIList("You didn't hit your water target yesterday - be sure to drink more water
               today!");
755        }
```

The example below shows that if the user is able to increase their overview percentage over 50%, then they will get receive a motivating message on their "things doing well".

```
757        if(achievementPercentageRounded > 50) {
758            addToWList("You're doing great! We see that you're above half-way to your goal -
               keep going!");
```

Improving details

Styling our functionalities

Initially, we spent majority of our time experimenting with libraries, playing with fonts, themes to create a high grade application which resembled our prototype templates. Some JavaScript plug-ins such as “notyf” came with their own premade stylesheets which saved us time designing and styling notification boxes.

The following screenshots below demonstrates how we added pop up notifications using notyf. The “add water” button is connected to a function which stores the user’s ID, the new serving of water consumed and the time the data was input onto our database. When this procedure is successful, then the system notifies the user that their “water consumed has been successful”.

Being able to record water consumption was a high priority requirement. This is the function to add water through a button.

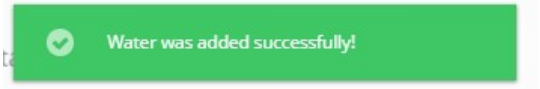
```
//This function deals with adding water via the add button
if(isset($_POST["waterAddButton"])){
    $id = $_SESSION["id"];

    $sqlWater = "INSERT INTO userWater (userID,waterAmount,`time`) VALUES ($id,
    1, now() + INTERVAL 7 HOUR)";

    if ($link->query($sqlWater) === TRUE) {
            echo '
            <script>
                var notyf = new Notyf({
                  delay:3500
                })
                notyf.confirm(\'Water was added successfully!\');
            </script>
            ';
    } else {
        echo "Error: " . $sqlWater . "<br>" . $link->error;
    }
}

//This function deals with deleting water via the delete button
if(isset($_POST["waterAddDButton"])){
    $id = $_SESSION["id"];
```

Outcome on the front end of the application.



Outcome on the back end of the application.

	waterID	userID	waterAmount	time
elete	3	5	1	2019-01-29 03:49:07
elete	4	5	1	2019-01-29 03:53:19

Ensuring responsiveness

Each style-sheet is specific to each page and feature in our application. This ensured our code was kept tidy and manageable during the front end designing of our project. Since we were coding from scratch, we needed to ensure responsiveness manually; hence why separating media queries for each feature made it easier to locate.

The following examples demonstrate how we ensured responsiveness of our application using media queries by individually adjusting to different screen sizes.

The following has been taken from main.css

```
@media only screen and (max-width:750px){
    .logo img{
        width:86%;
    }
    .logout,.profileText,.settings{
        display:none;
    }
    .iconContainer{
        padding:0 4px;
    }
    .logout,.menuTrigger,.notifications,.settings{
        padding:0 4px;
    }
    .profileImage{
        margin:0 6px;
    }
    .menuTrigger{
        display:inline-block;
    }
    .menuTrigger img{
```

The following has been taken from favourites.css

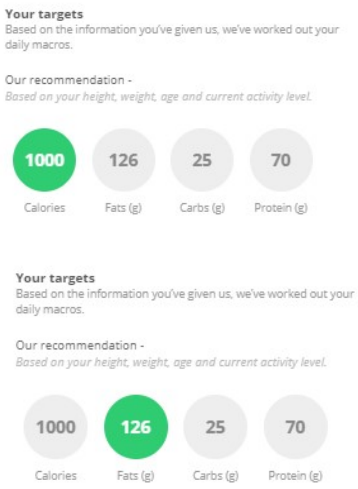
```
#grid {
    overflow: hidden;
    padding: 0px 18px;
    animation-duration: 1.5s;
    animation-delay: 0.3s;
}
@media only screen and (max-width: 767px) {
   #grid[data-columns]::before {
        content: '1 .column.size-1of1';
    }
    .column {
        width: 100% !important;
    }
    .item:nth-child(1) {
        margin-top: 18px !important;
    }
    #grid {
        padding: 0px 14px;
    }
}
```

Removal of redundant code

Having a “main” style-sheet granted a consistent theme throughout the application; it helped us avoid duplicate CSS code (excessively creating new classes and attributes with an existing style).

The following image (left) is an extract of styling code for targets on our daily log page (right). We carefully chose the names for our elements to make adjusting these in the future easier.

```
253  }
254  .targetCircle p {
255      line-height: 80px;
256      text-align: center;
257      font-size: 22px;
258      font-weight: 700;
259      color: #888;
260      border-radius: 100px;
261      background-color: #EEEEEE;
262      -webkit-transition: 0.6s ease;
263      transition: 0.6s ease;
264  }
265  .targetCircle p:hover {
266      background-color: #2ECC71;
267      color: #fff;
268  }
269  .targetExplain {
270      padding: 26px 36px 50px 36px;
271  }
272  .targetsOther {
273      padding: 6px 36px 0px 36px;
274  }
275  .targetsOther p {
276      color: #999;
277      font-size: 14px;
278      font-weight: 400;
279      font-style: italic;
280  }
281  .targetExplain p {
282      color: #bbb;
283      font-size: 14px;
284      font-weight: 300;
285      text-decoration: underline;
286      display: inline-block; /*Tooltip fix*/
287  }
288  #leanBodyMass {
289      font-size: 14px;
290      color: #999;
```



Animation

jQuery plugins

The speed at which our “pop up” notifications and forms appeared needed to be determined through the styling phase of our application. Using the jQeury library, we were able to enhance our animation speeds for smooth, beautiful transitions.

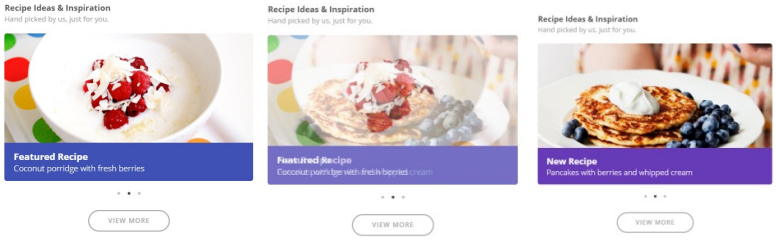
Slick is a responsive carousel jQuery plugin which supports CSS3 transitions which we used to add animation towards the featured recipes slideshow.

The following function controls our recipe tile carousel. Implementing the front end design version of this tile was not as time consuming as trying to merge the recipes with this functionality.

Using slick

```
//Function controls the re
$('.slick').slick({
    dots: true,
    speed: 1000,
    fade: true,
    arrows: false,
    autoplay: true,
    autoplaySpeed: 6000,
    lazyLoad: 'ondemand',
    cssEase: 'linear'
});
```

For fade effect

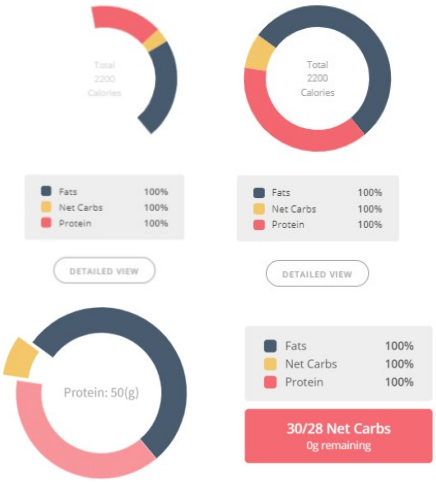


Built in features

Using fushioncharts.js was an advantage since it had built-in animation properties; after understanding the syntax, we were able to create effortless graphs which would open with professionally.

The following image (left) shows our code for "2D donut graph" which would visualise the user's macro goals. The entrance animation and duration has been determined to give polished appearance and animation results (right).

```
FusionCharts.ready(function() {
  var fusioncharts2 = new FusionCharts({
    type: 'doughnut2d',
    renderAt: 'chart-container2',
    width: '100%',
    height: '340',
    dataFormat: 'json',
    dataSource: {
      "chart": {
        "startingAngle": "310",
        "showLegend": "1",
        "defaultCenterLabel": "Total<br>2200<br>Calories",
        "centerLabel": "$label: $value(g)",
        "centerLabelBold": "0",
        "centerLabelColor": "#AAAAAA",
        "centerLabelFontSize": "18",
        "showTooltip": "0",
        "decimals": "0",
        "enableSmartLabels": "0",
        "showLabels": "0",
        "animationDuration": "3",
        "showValues": "0",
        "doughnutRadius": "80",
        "plotFillAlpha": "90",
        "showLegend": "0",
        "bgColor": "#DDDDDD",
        "bgAlpha": "0",
        "theme": "fusion"
      },
      "data": [{
          "label": "Fat",
          "value": "70",
          "color": "#34495E"
```



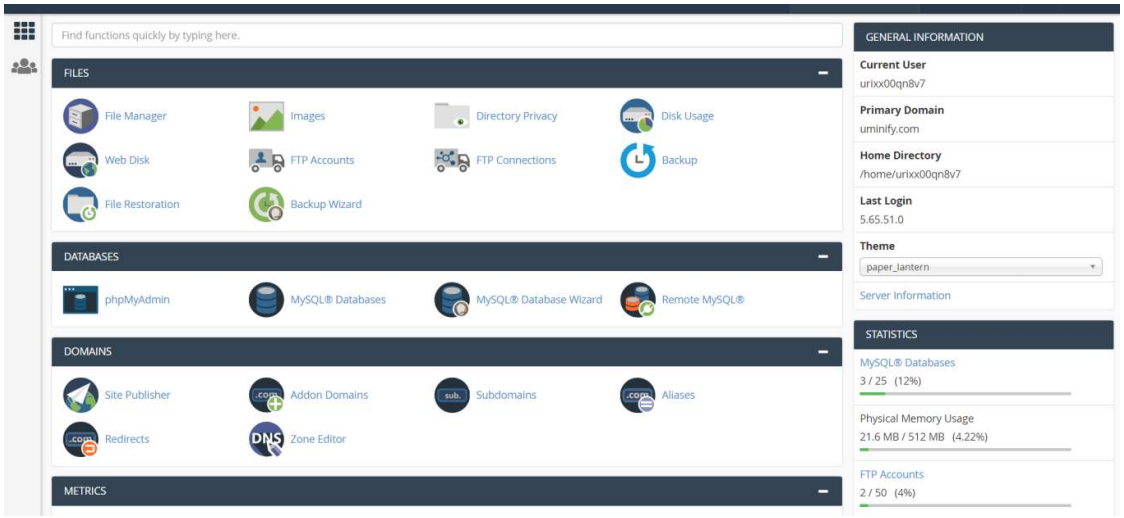
Hosting our application

Using Buddy, we created a pipeline between our main GitHub repository and our purchased severs on GoDaddy (our hosting provider). We gave CloudFlare permission to manage our DNS to take advantage of free SSL. Initially, we transferred our files to the server using FileZilla as our FTP client.

Cloudflare	Manages our DNS
Hosting provider (GoDaddy) ketogenetics.co.uk	Hosts our domain and servers
Database (GoDaddy)	Our database

Our control panel simplified the process of hosting our web application to our audience. It had an easy to use interface and we were given full control over different aspects of our application through easy commands for complex functions. Unfortunately, this proficient and powerful control panel lacks basic security since it relies on the server of the managing software, making it an easy target to hackers due to its encryption complications.

The following is a screenshot of our linux-based web hosting cPanel.



Overview of issues

Approach to problems

Every time we came across something we could not make or fix, we would research online to find a solution to our problems. Majority of the time, we managed to solve our issues. If we couldn't, we would leave it till the end as long it didn't affect overall implementation of our application.

Problem during implementation: front end

Some of the problems we had that we are going to discuss:

1) Ensuring our final application and prototypes looked similar
2) Ensuring responsiveness (application to fit mobile and desktop devices)
3) Changing our registration pages to original designs
4) Focusing on little details

Some of the features such as the responsive tiles and charts on our prototypes look simple however the complexity to code these was more challenging than we assumed. The unique tile and grid

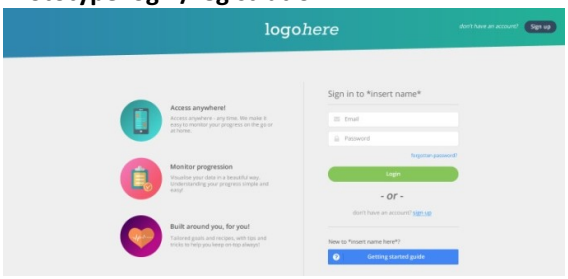
feature of our application was an element we could not remove; it was loved by our clients. We had problems trying to make this from scratch; it was taking too much time since we were not using any other frameworks (bootstrap) or templates (boilerplates). We found salvatorre.js as a solution which gave us the exact results we were looking for.

Another library we used excessively to match our prototypes was Fushioncharts.js – this was used to visualise the different graphs in our application such as the pie charts, line graphs, column and bar charts (specific examples include line graph to display the weight over time for individuals, presenting carbohydrate versus fat intake using column graphs, user's macros through pie charts).

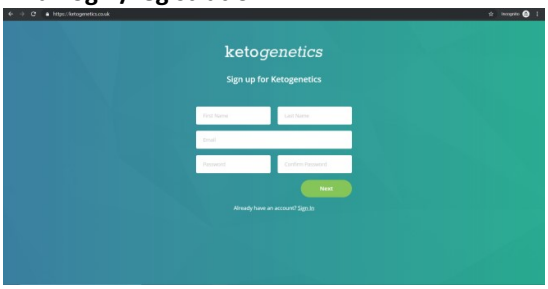
During early implementation stages, our login page for our final product was very similar to our colour desktop prototypes however we had issues with ensuring mobile responsiveness.

Developers were not willing to recreate the registration page after the creating the login page. The amount of detail that was being emphasised was not equivalent to the time being spent on those pages; we were not pleased with the results we were getting and there was so much more left to work on. One of the team members suggested looking at the older sketches that we came up with so we chose the design that our clients mentioned "resembled a login page".

Prototype login/registration



Final login/registration



We are glad we made this quick judgement since finishing the sign in and sign up pages earlier let us move to back end development faster which is where we had the most problems.

Problems during implementation: back end

Our inexperience with PHP and SQL was prominent during majority of back end implementation. One particular case was a bug in our SQL queries not being able to retrieve data; this took a lot of development time since we didn't know what was causing the problem. Our team was able to get in touch with GoDaddy and CloudFlare to check if this was a hosting/server side issue due to a recent software upgrade. The problem has been partially fixed however still pops up when users don't fully input every piece of data required on the forms.

We had trouble connecting to our SQL server; we learnt that the issue was related to poor configuration. We checked to make sure our target SQL server running and had an ability to listen to appropriate protocols.

When focusing on database connection for our web application we came across application context database, database pooling and new connection per request. This allowed us to establish the reason for common failures for connectivity in our web application.

Relying on a single database connection amongst all users creates a long running query which can grind our entire server to a halt; this would make our application unreliable and slow. Using connection pooling as our technique would require CPU usage and memory allocation with minimum available connections to attach to; this would allow us to work around traffic load since the pool would keep a number of connections open at all time and consistently grow as traffic increases.

We opted for a new database connection per request, this option works well for our moderately utilised system and is easier to upgrade to a pooler at a later date. Each database connection will require opening and closing a database connection therefore in practice our pages load faster. This created bugs in our back end development since we issues of hanging queries due to lack of control over connections to database. Later, we set up our database through with an app server and DB server, we were able to find this solution was easier to adapt with which it prevented system overloads.

Testing

Quality assurance

This section provides an overview of the testing completed to assure the quality of our application. We conducted several tests during and after development.

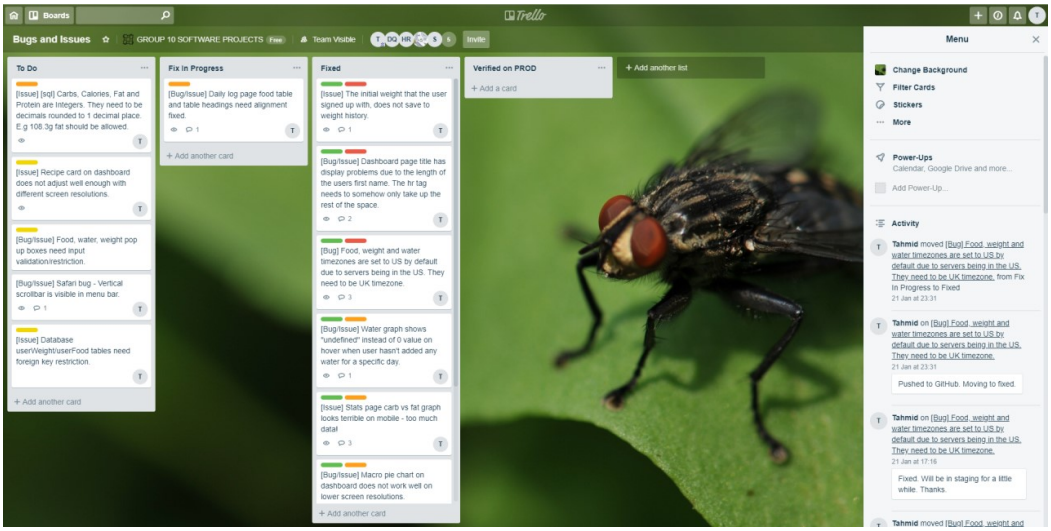
Testing criteria

We composed our testing criteria in the form of a checklist. This was based on our functional and non-functional requirements. We constructed various test cases to ensure quality delivery of Ketogenetics. We strived to obtain positive results during the testing of our “web application”.

Bugs and Issues Management

When problems were found, the tester would log the problem onto the “Bugs and Issues” Trello board where the team could track issues fixed and those that remained. When bugs/issues were identified, they would be added to the “To Do” column and given a severity rating from “Critical” to “Minor” depending on how quickly we felt the bug needed to be resolved.

The following image below is our “bugs and issues” board.



At the end of each sprint run we would assign developers to go through the bug/issues log prioritising the critical and major bugs and work on fixing them. Throughout the sprint we would work on the minor bugs as a normal part of development. This helped us stay on-top of bugs/issues and prevented us from having to deal with a huge backlog of them at the end of the project.

During development

We covered the following: code reviews and functionality testing.

Code reviews

Reviewing the code was completed with the authors of the code and quality assurance testers.

The developers provided testers with the code files; comments were used to relay where changes should be made and if anything wasn't working. Developers and testers were successful in regularly checking status of the code review log.

It allowed us to reduce errors by fixing them earlier rather than later in development. The method was time consuming and had to be completed regularly compared to the other testing methods completed.

The table below is an extract of our code review log. The developer would fill out their section and tester would fill out their part; comments column could be made by either.

Developer			Tester			Comments
Date	**Name**	**Status**	**Date**	**Name**	**Review**	
16/12	Tahmid	Completed	16/12	Mizna	**Bugs:** No bugs found; checked using validator. **Methodology:** 4, Concerns regarding HTML only for web application. **Presentation:** 5, Clear HTML and CSS layout.	Look into JS libraries
17/12	Tahmid	Completed	17/12	Mizna	**Bugs:** No bugs found **Methodology:** Good use of Javascript and separation of concerns **Presentation:** 4, the CSS layout needs fixing; padding needed from line 143 onwards	Keep code clean and wrap into containers
20/12	Tahmid	Completed	19/12	Mizna	**Bugs:** No bugs found **Methodology:** Consistent methods **Presentation:** 3, it is getting worse; fix the presentation of styling section.	Stick to px for sizing and fix CSS using beautifier

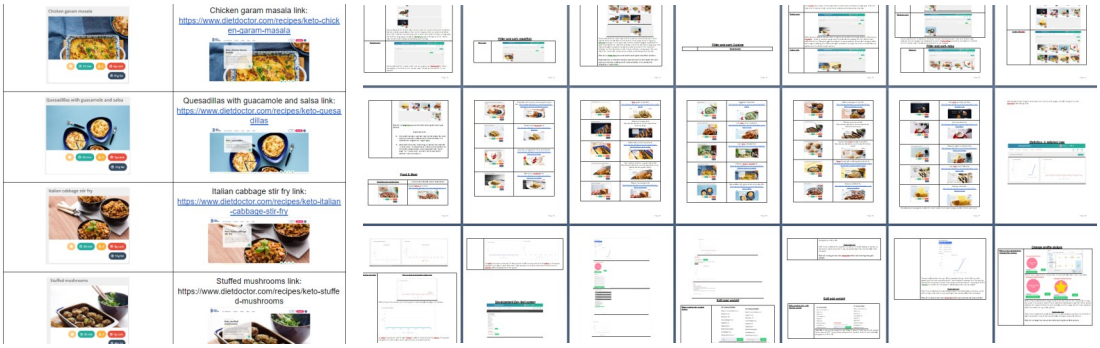
During and after development

We covered the following: functionality testing, security testing, and performance testing.

Functionality testing (Front end)

This was conducted to check whether the front end of the application performs as intended. One of the testers went through every single link and documented what appears depending on the link clicked. Tester found some links that were wrong links and a couple of dead links. We were able to fix these accordingly before our user testing phase.

The images below show how we organised the functionality testing documentation.



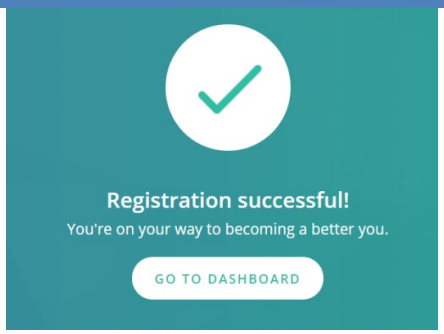
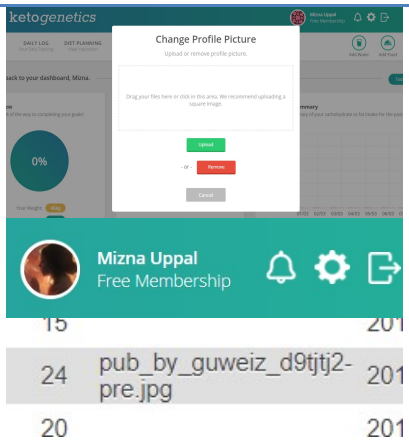
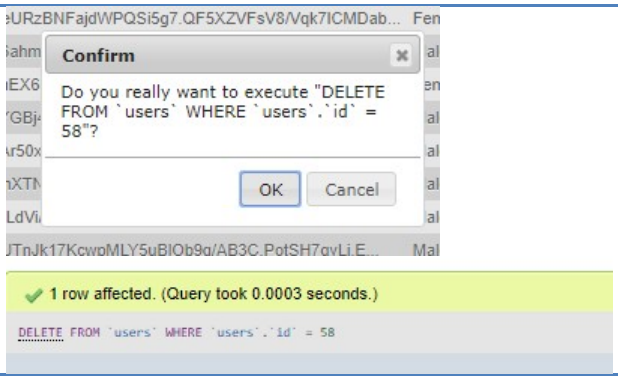
Functionality testing (Back- end)

This testing was done to ensure that the data which was input at the front end of application is stored into a database. We tested table structure, scheme, and structure; we were able to record the procedure and its efficiency.

The table below is an extract of the tests we conducted for our back end development

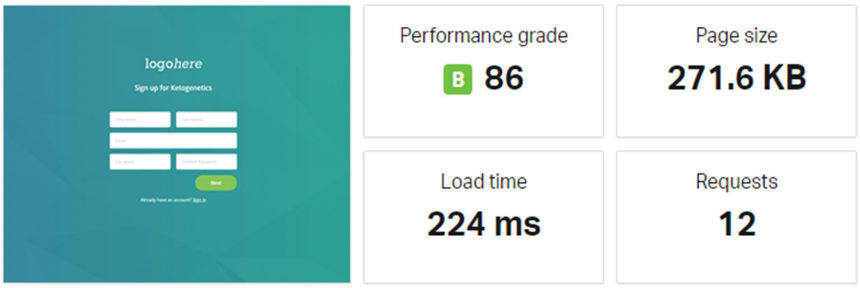
Test case	Description	Test Data	Expected	Results	Pass/Fail
BE3	Signing up accurately as possible to Ketogenetics	First stage *First name: Mizna* *Last name: Uppal* *Email: muppa002@gold.ac.uk* *Password: ****** Second stage *Gender: female* *Birth year: 1998* *Current weight: 48kg* *Goal weight: 42kg* *Height ft: 5* *Height inch: 1* *Activity level: Sedentary* *Body fat %:* 30	The new record would appear on the database since registration is complete	The new record appears on the database since registration completed properly	Pass
BE4	Adding a profile picture to my account	Click on my profile picture Drag profile picture Click on upload Click on "save"	My profile picture should be saved and stored onto database	Image saved and description (file image name) put in user record	Pass
BE7	Delete a record of user from the database	Delete the account with email newdeletethis@gmail.com	The record will be deleted and cannot be changed.	The record was deleted.	Pass

The table below follows the back end testing we conducted; it records results from procedure.

Test case	Screenshot	Comments
BE3	 Delete 55 laithkamaral@gmail.com laith kamaraldin $2y$10$63 Delete 56 muppa002@gold.ac.uk Mizna Uppal $2y$10$xX	Completing registration properly. New record has been added to the database.
BE4	 15 201 24 pub_by_guweiz_d9tjtj2-pre.jpg 201 20 201	Changing profile picture; updated on to the database.
BE7		Now attempting to delete the account. Row deleted therefore user deleted manually

Performance testing

We ran many website performance tests to ensure regardless of location and internet connection speed, our web application will load quickly and reliably. For our testing, we used pingdom which gave detailed website performance reports and insights. Our findings and performance reports show that our register page has a load time of 224 milliseconds, we have 11 response code 200s from the server (this means all resources were successfully loaded), our total content size is 271.6kb and pingdom.com has given the website a performance grade rating B meaning the website is faster than 85% of tested websites on the platform. Detailed report:
https://tools.pingdom.com/#5a1fe1ffb5400000



Chrome Dev Tools was an invaluable resource to us during the creation of the website as it provided useful tools such as the ability to emulate screen resolutions. This allowed us to view the website at different screen resolutions and helped us ensure that the application would be responsive (adjust to different screen resolutions accordingly). Furthermore, it allowed us to spoof user agents (this meant we could test on browsers that we did not have access to). Similarly, like pingdom.com it allowed us to throttle network connections to test and measure website performance and speed and make sure it was adequate.

After development

We covered the following: software quality, compatibility tests and load testing.

Software quality

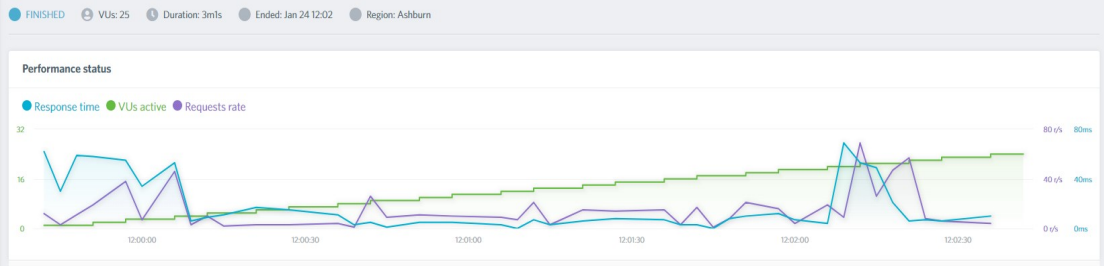
We ran all our html mark-up through the W3C html mark-up validation service. The W3C (World Wide Web Consortium) is the main international standards organization for the World Wide Web. We used their mark-up validation service to: check the validity of our html, ensure that all relevant html tags were closed, ensure there were no broken or invalid links and make sure the syntax was valid for all our pages. Our html code passed the W3C validation with no errors.

We ran all our CSS through the W3C CSS validation service to ensure valid css code with no syntax errors were present. All our css code passed the W3C validation test successfully, apart from some minor warnings (this was normal since they were browser specific vendor extension warnings).

Running our code through these validation services helped us identify errors, typos and incorrect uses of html/css. We determined if our code poses any risks in terms of usability; it allowed us to create a cross-compatible web application which is error free and scalable in the future.

Load testing

We conducted stress tests on our application to simulate how it would behave when hundreds of people are using it at the same time. We used loadimpact.com to run a stress test using their virtual servers to check if our website response time and performance dropped as the number of concurrent users increased. As shown in the graph below, there was no significant drop in https response time (in fact it seemed to get faster as the number of users increased). This is impressive since we are only using GoDaddys shared hosting plan where we are capped on server resources. The test results show that in the future our application should be scalable with an uprated/dedicated hosting plan and more powerful hardware.



Compatibility testing

We used “browsershots” to see what our application would look like on different browsers. This improved our insight for a cross-browsing compatible application whilst we verified our system’s behaviour on older versions.

Screenshot request group 1 for https://ketogenetics.co.uk/

Linux

Arora 0.11:	3 min
Chrome 37.0:	3 min
Chrome 38.0:	3 min
Chrome 48.0:	3 min
Chrome 69.0:	3 min
Chrome 73.0:	3 min
Dillo 3.0:	3 min
ELinks 0.12:	3 min
Epiphany 3.18:	3 min
Epiphany 3.4:	3 min
Epiphany 605.1:	3 min
Firefox 10.0:	3 min
Firefox 11.0:	3 min
Firefox 12.0:	3 min
Firefox 13.0:	3 min
Firefox 14.0:	3 min
Firefox 15.0:	3 min
Firefox 16.0:	3 min
Firefox 17.0:	3 min
Firefox 18.0:	3 min
Firefox 19.0:	3 min
Firefox 20.0:	3 min
Firefox 21.0:	3 min
Firefox 22.0:	3 min
Firefox 23.0:	3 min

Windows

Chrome 39.0:	3 min
Chrome 44.0:	3 min
Chrome 45.0:	3 min
Chrome 51.0:	3 min
Chrome 71.0:	3 min
Chrome 73.0:	3 min
Firefox 30.0:	10 min
Firefox 35.0:	10 min
Firefox 40.0:	10 min
Firefox 41.0:	10 min
Firefox 42.0:	10 min
Firefox 43.0:	10 min
Firefox 44.0:	10 min
Firefox 45.0:	10 min
Firefox 46.0:	10 min
Firefox 47.0:	10 min
Firefox 48.0:	10 min
Firefox 49.0:	10 min
Firefox 50.0:	10 min
Firefox 60.0:	10 min
Firefox 61.0:	10 min
Firefox 62.0:	10 min
Firefox 63.0:	10 min
Firefox 64.0:	10 min
Firefox 65.0:	10 min

Mac OS

Chrome 41.0:	9 min
Chrome 45.0:	9 min
Chrome 48.0:	9 min
Firefox 30.0:	9 min
Firefox 31.0:	9 min
Firefox 32.0:	9 min
Firefox 33.0:	9 min
Firefox 34.0:	9 min
Firefox 35.0:	9 min
Firefox 36.0:	9 min
Firefox 37.0:	9 min
Firefox 43.0:	9 min
Firefox 44.0:	9 min
Firefox 45.0:	9 min
Firefox 46.0:	9 min
Firefox 47.0:	9 min
Firefox 48.0:	9 min
Safari 9.1:	9 min

Functional requirements evaluation

We tested each requirement thoroughly through code reviews and functionality testing. We checked whether the system was able to perform as intended by entering different data values and then recording the results which we would review later on to improve the functionality. Examples of our application not behaving as intended include erroneous links, problems with back end functionality being unable to update user input, unresponsive buttons, inaccurate data calculations etc.

Majority of the features of our application were developed and tested before we conducted our user testing. During this process, we were able to learn how to efficiently handle errors and prevent them from reoccurring. It helped us identify where the bugs in our application were mostly concentrated and focus on implementing a method to fix this.

Creating a log of all our issues and problems ensured that if this project is picked up on in the future then the new developers and testers have some form of documentation so they are aware of problems that still need fixing; it can also help if the future developer faces issues that they do not understand how to handle since our team may have already done so.

The following table evaluates each requirement and what we did to fulfil them.

REW ID REM ID	Requirement	Evaluation/Improvement	Met
REW1 **REM7** **REM8**	Must have a login page with registration option	We created a login and registration page. The login page is the first page which the users see when they use our application. From this page, they are able to access the registration page. We have considered the possibility of adding guest accounts for our future releases to allow users to experiment with our application.	Yes
REW4	Website must contain a settings page	We have a settings page which can be accessed after the user logs in; they will be able to change their goal weight, profile picture and delete their account. We dropped features such as the ability to change calories and macros manually but those parts remain of the application for future development.	Yes
REW5 **REM11**	Website must contain logout button and mobile users to stay logged on	There is a log out button at the top of the application on the navigational bar, next to user's profile and settings page. When the user logs out, they will not be allowed to access the system until they log in again.	Yes
REW6 **REM9**	Website must contain a dashboard that has the user's selected and personalised tiles/features	We created dynamic tiles that are responsive to changes in screen size. These tiles create the layout of our application and portray user's progress and other data visually through graphs and goals. The dashboard page overviews majority of the features from the application however it does not let user's select their own personalised tiles, yet. This is a feature we have added to develop for future versions of Ketogenetics.	No
REW7	Website must contain section	Our application contains a specific "statistics" page that displays historical and current data on the "tiles". We have	Yes

	dedicated to statistical information	visualised: user's weight progress/macros using line graphs, user's water consumption through area graphs and user's carbohydrate versus fat intake with bar charts.	
REW8	Website must contain page dedicated to diet planning	The diet planning page contains a series of recipes which display the user's breakfast, lunch and dinner with caloric needs and details regarding the meal itself. Users are allowed to manually add their own meal/food items as well as long as they know the specific calories, carb/fat/protein for each item. We need to improve this functionality by lettings users choose from a list of food items in a database.	Yes
REW9	Diet planning page must contain recipes which can be previewed on separate page	Users can preview any recipe that they want to from the diet planning recipes page. Upon clicking, they are redirected to the link where the recipe was taken from. This removed our copyright responsibility and allowed us to add more recipes to our application to present for current version. We need to refine this and add functionality where users can put the specific recipes from our list onto their profiles.	No
REW10	Website will have a daily log page where they can see their day's progress	When users edit their daily logs, goal weight or input data in other features; the database automatically stores these tracked changes. The changes are used to visual graphs which users can observe to see day's progress.	Yes
REW11	Website will contain "tiles" with relevant user details	Our application contains tiles relevant to each page as proposed in the design (water consumption tile, weight tile, food intake tile etc)	Yes
REW12	Website will contain sections dedicated to water consumption	Water intake is monitored and saved in database and presented to user graphically through an area graph. We can improve how the user inputs the water they drink onto our system with different fun shaped glasses (as proposed in original designs).	Yes
REW13	Website will be linked to a database	We have successfully created a connection between our front end and back end. The web application is able to store users' data and create profiles based on their history, caloric and carbohydrate intake.	Yes
REM1	Tiles will fit regardless of mobile screen size	The tiles on our application are responsive and fit regardless of mobile screen size as demonstrated through our compatibility testing.	Yes
REM10	User will need to stay logged in whilst using the mobile application	Currently the application lets the user stay logged in if they do not log out; this was to accommodate our mobile requirement.	Yes
REM12	Application will show menu	Instead of swiping left feature to access menu; our application keeps the menu at the top. Users can swipe this section to view all the pages of our application.	Yes

Non-functional requirements evaluation

Our system is fast and available at all times as confirmed through our performance and load testing. We were able to create an application with estimated grade B for response time. However, despite the good results, there is still room for improvement especially since we will need to support the increasing number of users in the future, without affecting our resources.

Our code followed separation of concerns (SOC) to support DRY and avoid WET code; this was done by splitting our application into separate sections and eliminating redundancy through use of libraries and other tools.

Through user testing, we were able to confirm the reliability and ease of use of our system since it performs as intended. Our application has a clean, consistent and unique user interface as demonstrated in the feedback. The design quality and responsiveness of our product attracted a high average rating of 4 stars however these results are unreliable since the sample of users testing quality of our application was small.

The security of our application has been provided through password protected user accounts to store the personalised data and ensure only the user has access to it. We could improve the security through validation of emails provided; this is to avoid unwanted traffic since all the accounts being created will be associated to a real person.

Our approach to quality assurance was by verifying that we were meeting quality metrics. The following are defined as standard for web applications; these tied with our non-functional requirements.

REN ID		Evaluation/improvements	**Met**
REN1 **REN10**	System is fast as confirmed by our performance testing. System is available all the time.	Our web application is currently graded B in terms of performance and response; we need to improve the load time to ensure faster delivery.	Yes
REN2 **REN8**	System is capable of supporting an increase in number of users without affecting resources as shown through our load testing. System can store plenty of entries without downgrading load time.	There was no significant drop in https response time when we conducted our stress tests on system.	Yes
REN3	Our user testing confirmed users being able to use buttons and forms on mobile with ease.	Our application received an average rating of 4 stars which shows the ease of use whether on mobile or desktop.	Yes
REN4	It is possible to extend storage and capabilities of our web application. Our organised code is easy to correct and change in case of defects.	To improve the code, we need to ensure SOC to avoid DRY. There needs to be more well commented code.	Yes
REN5	System is reliable and performs as intended.	For a trustworthy application, we need to keep testing the system and keep it free from failure otherwise users will lose interest.	Yes

REN9 **REN6**	Application is easy to use as shown in users testing since front end and backend developed in clear structure. Our system has a clean, consistent, responsive, user friendly and modern UI design. We have a user guide to aid users.	If user approaches a link or image and they are not directed to correct place, it can annoy and lead them to avoiding our app. To check if we have any 404s, we can set up Google Webmaster tools on our website to check crawl errors (there are free 404 checkers available to aid).	Yes
REN11	Security of the system has been ensured with password protected user accounts/SSL certificate.	To improve validation and security, we need to ensure that the accounts being created are real people.	Yes
REN12	Our application is accessible on mobile and desktop. Our application is accessible on different platforms and browsers.	we tested on a range of different types/models. To ensure that the application is compatible, we need to continue testing it on as many different platforms and CPU architectures as possible; we need to explore older browsers as our choice of testing as well.	Yes

User testing and evaluation

This section evaluates our project's user testing during the development phase, including our formative evaluation.

We would like to point out the importance of the results from our designing and prototyping stages since majority of our communication with stakeholders took place during these phases. They provided us with validation by evaluating each individual prototype through answering questions; we used this feedback to create the design blueprints for our web application.

Usability testing

We tested with our users on two separate occasions through similar methods. Our usability testing and remote usability testing were conducted with 5 stakeholders. The first study was conducted on Monday 11th March and the second went through Saturday 16th – Sunday 17th March which was close to the final stages of our development.

The following table compares the methodology we applied for both user testing experiments.

Usability testing	Remote usability testing
Users were asked to fill out a sheet with total of 6 tasks. Instructions to complete the tasks were provided and we observed them individually whilst they used our web application in the lab.	After discussing with our supervisor, we conducted a similar study with a different set of users but the same set of tasks. This time, we did not provide instructions to complete the tasks and left the users to figure it out; the users completed the tasks in their own time at home. We told stakeholders to use VLC media player's screen capture, to record what they were doing.
The questions we asked were simple since we were looking for yes/no replies; the user would tick "yes" if they were able to complete the task provided or they would tick "no" when they reached a step that did not see occurring or could not achieve. If they ticked "no" then they would have to explain the procedure they followed in a separate box.	Users asked us questions regarding the application later on compared to the first set of group; some of these questions were related to the tasks whilst some were actually showed interest in the application.
The first test was both successful and unsuccessful. Our objective was to find out whether the users could fulfil tasks and use our application but we did not have a solid method for logging the details of the users therefore weren't able to conclude more from the study.	Our team went through the videos to compare the users' written results with the screen recorded results and the results we were seeing on our database. We followed up with users and asked them questions regarding anything we found interested. For example, some entered extreme body fat and we learnt many people do not know how to calculate this.

Tasks provided

The following examples are a couple of tasks provided to the first group. The second group received the same set of tasks, just without the steps in the instructions section.

The table below shows the 6 tasks given; they were related to the core features of our application.

no	Task	Aim for task
1	Registration on Ketogenetics	To see whether users are able to register successfully on to our application without facing errors and problems with validation
2	Logging out and logging in	To see whether users are able to log in and out of the application.
3	Adding a new goal weight	To see whether users can input a goal weight successfully to which the application would produce a new set of macros and goals for the users.
4	Adding a new glass of water	To see whether users are able to input their water consumption for the day.
5	Adding breakfast	To see whether users are able to add their food to their profiles.
6	Favourite meals	To see whether users can favourite their meals and access them.

This specific task below was for us to check whether users could actually sign up on our application.

Task 1	**Registration on Ketogenetics**			
Instructions	We need to assess whether you can register on our application successfully and then access it; please note the details that you registered with down since they might be required in the next few tasks. Step 1: Open Ketogenetics.co.uk Step 2: The first page you should see is the registration page Step 3: Sign up to our application however you like Step 4: You should automatically be logged into our application and see the dashboard			
Yes		**No**		(tick yes if each of steps take place otherwise tick "no")
If not, then please state why and the procedure you went through to fulfil task below and specify the step which did not occur.				

This specific task below was to check whether the users' details that they used to sign up with; were being stored and users were able to log in and out successfully.

Task 2	**Logging out and logging in**			
Instructions	We need to assess whether you can log in and out using the details you signed up with. Step 1: We will consider that you are already registered and logged into the application Step 2: Click on the log out button at the top Step 3: After you have been logged out, use the details you registered with to log in again Step 4: You should be successfully logged in and see your dashboard			
Yes		**No**		(tick one of the following boxes)
If not, then please state why and the procedure you went through to fulfil task below and specify the step which did not occur.				

Online survey

An estimate of 51 people filled out the questionnaire we provided them with and gave feedback regarding our web application. The results allowed us to collate how we were going to improve our application before our next user testing. The purpose of this test was to ensure that the quality of product was of high standard and users were able to use the application regardless of what device they were on.

Quality of application based on mobile and desktop: We needed to know whether the quality of our application is affected depending on if the user is accessing our application through mobile or on desktop. We sent a survey out to ask a larger audience.

We asked the users:

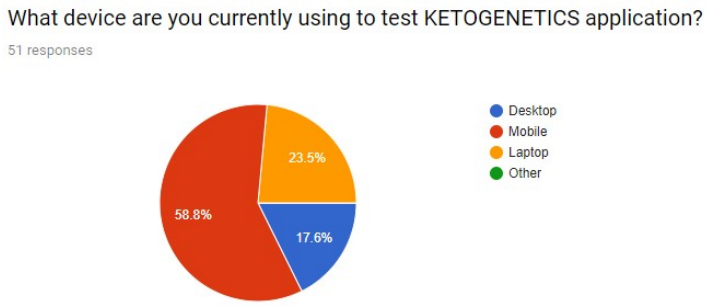
- ***Which device they were using when testing our application?***
- ***To rate how easy it is to use the application***
- ***To rate the quality of our application***

Using Google forms allowed us to check each individual response and see if the type of device can affect a user's experience on our application.

Results: mobile rating was lower than desktop

There is a chance these results are skewed since we had more mobile users testing our application compared to desktop and laptop users.

The image below shows the "type of device" results.

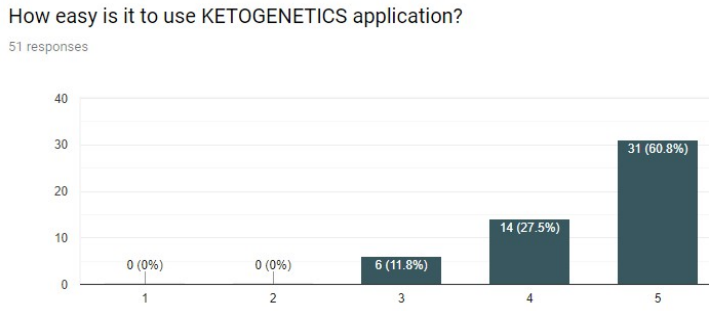


Regardless, we found that on average, laptop and desktop users were likely to give the application a higher rating compared to mobile users which meant we needed to work more on our mobile version (especially since there were a higher proportion of mobile users).

Results: It was easy to use and easiest for desktop/laptop users

Most of the users found the application easy to use with 5 being the highest rating. This concludes the test for "ease of use of application" to be a success since so far the responses showed it is not hard to use.

The image below shows the "ease of use" results.

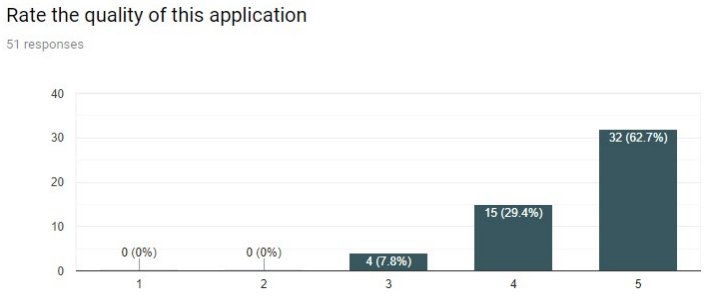


Desktop and mobile users found the interface easier to use compared to mobile users.

Results: our application is of good quality

The quality of our application currently stands at an average of 4 stars. This proved our application is easy to use.

The image below shows the "quality of application" results.



Desktop/laptop users rated the application with a higher rating compared to mobile users. Our team assumes this is because desktop and laptop users were able to preview the responsiveness of grid feature of our application.

The remaining results from this study can be found in appendix C.

Formative evaluation

Our user testing was completed through an online survey and then real life studies.

Number of users

We carefully chose 5 stakeholders to test our application for both our usability studies. For our online surveys, we needed a large audience to get maximum feedback for our product's quality depending on whether the user was on mobile or desktop hence why the number of users testing our application was not planned. Regardless, we received 51 responses with over 21 users providing improvements for the system.

Research

For the second study, we considered cognitive walkthrough, heuristic evaluation and usability testing but eventually opted for remote usability testing. We decided against popular moderation methods for our usability testing such as concurrent think aloud (CTA) and retrospective think aloud (RTA) since they require manual logging and because we were not physically with our stakeholders. Instead we used a method similar to retrospective probing (RP) when they sent their recordings.

Reflection on Usability testing

Outcomes from testing

Usability testing allowed us to see if the stakeholders were able to complete the tasks we had given them without any issues. We were clearly able to see exactly where our stakeholders were getting stuck. Some of our testers took this opportunity to expose the some problems our team had failed to identify. If we had conducted user testing earlier, we could have reduced the development and time costs.

Lessons learnt

For our remote usability testing, we followed up on users after the tests to ask them questions regarding the application; this method was flawed since it is likely that users may have forgotten by then therefore weren't giving accurate responses. We could have improved this by maintaining contact with the tester whilst they were testing the application and asking questions through an online medium.

Planning the experiments, research and preparation took longer than we expected and after not conducting the first study properly, we had to do it all over again. However, it was worth organising since we obtained more information than our first experiment. We had load of information but weren't sure what to do with it. Instead of settling down as a team to discuss what to change; we should have consolidated in writing up a test report. By writing it down, the design team would have been able to understand better.

It was disadvantageous that we only recorded the users for the desktop version and not the mobile version as well. Due to lack of time, we were not able to conduct a study which recorded the users and what they were doing to fulfil tasks on mobile visually.

Reflection on Online Survey

Outcomes from Online survey

Our online survey allowed us to make a fair judgement on quality of our application. Since we were short on development time, using this convenient method provided us with fast results overnight. We were able to gather that our application was responsive on different devices and were able to see whether this quality is affected whether user is on mobile or desktop.

We were able to learn the nature of how the application is used by an actual audience; we did this by channelling through the database and checking data entries being made by users after release of survey.

Lessons learnt

An issue with this was our inability to recruit the right participants for the online surveys. If the participant doesn't know anything about Keto or isn't interested in dieting but wanted to evaluate the application anyway, then they may not have the same opinion compared to our actual stakeholder. Despite receiving positive feedback, our results are still skewed. We could have avoided this by sending the surveys to specific users and Keto subscribers on reddit instead.

Following User testing

After collection of results, we were able to see the technical and functional problems with our application. With only a little bit time left for development, we decided to fix changes that we could and leave the rest to fix for future releases.

There might still be some issues with the application that even testing procedures can't identify; these are not bugs, rather the nature of how the application is used by an actual audience. These errors must be found and fixed before the deployment. The user testing was a great way to see how actual users make use of the application; this has made identifying a pattern and where errors could occur clearer. Example: users submitting extremely high body fat percentage or extremely low (e.g. 60% or 5%); though this is possible, we knew it was inaccurate from the sample of the audience we used; they were neither clinically obese or athletes.

Scalability of the application is an issue to work on. As the number of users, images and recipes increases, and the load time for the website does get slower (though this is not as noticeable, it is still something to look at especially if we are to cater a large audience). Example: as we progressively add more images, the application gets slower – whether it is profile images or recipe images. One way around this is to limit the size the user uploads as their profile picture.

We used their feedback to make changes that would improve the user performance before deployment.

The table below shows some of the problems we saw and the solutions we made.

Problem	Solution
Our previous back-end issue with SQL queries was coming back. There was a	We were able to fix the issue during the actual experiment since the user alerted us whilst test was

unique bug regarding the overview progress. After signing up, one of the users realised that their progress keeps increasing. This went over 200% so we had to find the root of the problem after user explained the details of their registration procedure.	ongoing. The user had entered a negative goal weight which caused a loop since they had reached their goal weight. This issue should have been addressed by the team during early validation testing but we are glad it was brought up anyway. After we fixed, the user was allowed to sign up again with same details and continue with the tasks.
Users were submitting extremely high or extremely low body fat percentages (e.g. 60% or 5%). Though this is possible, we knew it was inaccurate from the sample of the audience we used; they were neither clinically obese or athletes.	Some of the users genuinely did not know how to calculate bodyfat. We changed the application slightly so now there is a guide which teaches users how to estimate their body fat percentage. We also included techniques to measure body fat in our user guide.

Conclusion

Summative evaluation

Our aim was to create a web application that is able to cater Ketogenic dieters with their weight purpose goal. The system is supposed to be accessible on mobile and desktop devices. The application must create profiles for individual users including their macros and meal plans.

Requirements

Our team managed to complete all the prioritised requirements. This project was successful and our final application shows this. Our goal was a working application that meets our MVP therefore we weren't expecting brilliant quality; our focus was always on the technical aspect of the web app. However, our results demonstrate that the consistent design elements and responsiveness of the application are just as outstanding.

Design

There is a consistent theme running throughout the application making the interface easy to use. We utilised industrial standards with the navigation menu at the top, logo at the left top corner, log-out button at the top right, settings at the top and profile icon at the top. Following standards makes our application more familiar to users who would enjoy using it more than if we had decided to be more creative with the layout and structure. Throughout this project, our team focused tailoring the application to the needs of the user rather than how we wanted it.

Implementation

The final application resembles our prototypes; it has a modern design with unique grid layout as we had planned. Building from scratch using HTML5, CSS3 and JavaScript gave us maximum flexibility and customisation. Working with these languages was an advantage since the team was familiar with them, making it easier to create the application.

Front end development brought us problems regarding design of the application; when we started, we knew we did not have enough time. To solve the issue of implementing a grid style layout, portraying pop up messages and visualisation of graphs, we found different JavaScript libraries to do the work for us. During the back end development, we read tutorials to learn how we could connect and manipulate our back end database with our front end user interface; we ended up using phpMyAdmin, which is a free software tool written in PHP. Back end development took the longest since it was everyone's first time manipulating a database with web interface.

Security

Security of our application was overlooked throughout the project; there are many attack vectors such as SQL injection, Cross-site scripting, CSRF and remote file inclusion. Our team explored techniques to sanitise input and output to eliminate vulnerabilities and unauthorised manipulation of our application. Unfortunately, most of the methods we found were unpractical since most web applications will always be in a state of development. We opted for web application firewalls (WAFs) to protect the application from security threats and block attack attempts. "Hash and salt" technique allowed us to store the users' passwords securely; they cannot be easily guessed.

Hosting

Finding the right hosting infrastructure was critical since we required a compatible server platform with scalability and security built in. During the production of our system, we did not plan for hosting the web application however we needed to know about the potential of our application and how it worked on different devices. Hosting on personal hardware was a security threat and since our responsive web application was not a static website so we could not host this through Github or cloud storage services such as AWS S3, Google Cloud, and Microsoft Azure. After some research, we settled with CloudFlare for DNS management and SSL. CloudFlare automatically protected our website from malicious traffic. It distributes our content quickly by sending it through a global content delivery network which ensures that our pages load faster. A benefit of using CloudFlare was that it provided a free SSL certificate which served our website over HTTPS, boosting the security of our application.

Future work

Meeting the requirements and creating a working application does not mean the "Ketogenetics" is complete; there will always be room for improvement.

Security

We could improve the security of the application by adding a validation feature when registering. Currently, when users sign up to use our application, they can sign up with any email; this includes a non-existent email. Having a validation feature which checks that the user is "real" with an exclusive email account enhances the security of our application.

Backlogs for future releases

Before development, we devised detailed backlogs whilst settling on our MVP. These contain potential features which could be added onto our core features. In the future, we could add more to our application and work through with the releases as specified in backlogs. Examples for expansion

include social interaction with other users with chatting function, improvement to the tile features and integration of fitness devices with our application.

Ethical

The purpose of this application is to aid in weight purpose goals through a Ketogenic diet lifestyle; however this diet is still controversial with health concerns involved. If a user decides to use our application with health issues which makes this worse then they might decide to hold us accountable. We have mentioned consulting a doctor on our user guide however this is still not enough. We should create terms and conditions which the users must tick before they can register to use our application.

Performance

Another vital improvement for future development is upgrade in web performance. As more users begin to join, the load time for our application will increase. CloudFlare offers powerful load balancing with CloudFlare workers which could aid scalability and maintainability of Ketogenetics.

Marketing

Additionally, we need to refine our promotional plan before executing our web application into the market. There are plenty of ways such as search engine/app store optimisation and networking to reach out to influencers and tech blogs who would review our application.

Bibliography

1) W. Wheless, J. (2008). [online] Onlinelibrary.wiley.com. Available at: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1528-1167.2008.01821.x [Accessed 28 Apr. 2019].
2) Diet Doc. (n.d.). What Is the Origin of the Ketogenic Diet? | Diet Doc. [online] Available at: https://www.dietdoc.com/diet-tips/ketogenic-diet-oriTgin/ [Accessed 28 Apr. 2019].
3) Adult obesity and type 2 diabetes. (2014). [ebook] Public Health England. Available at: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/338934/Adult_obesity_and_type_2_diabetes_.pdf [Accessed 29 Apr. 2019].
4) Pingdom website performance tester | pingdom.com | https://tools.pingdom.com/#5a1fe1ffb5400000 [Accessed 27/03/19]
5) W3C HTML mark-up validation service | https://validator.w3.org/ [Accessed 20/04/19]
6) CSS Validator | https://jigsaw.w3.org/css-validator/ [Accessed 20/04/19]
7) Updating headers on cloudFlare | https://blog.cloudflare.com/update-response-headers-on-cloudflare-workers/ [Accessed 20/04/19]
8) Build Fire, App promotion | https://buildfire.com/free-app-promotion/ [Accessed 18/04/19]
9) IBM web app performance | https://www.ibm.com/developerworks/web/library/wa-webappperformance/

Appendices

Appendix A: Introduction

A1: Motivation

The following table addresses a few of the problems we are solving; there are plenty more reasons covered in our project proposal.

Problem	Reasoning
Majority of the available applications contain static pages, similar to a hard-copy cookbook	These "apps" are built to support users with their meal plans ("Keto diet app" on google play app store). They are not personalised to the users at all and offer little to no interactivity; instead they serve the purpose of providing information on the Keto diet and meals only
Some Keto based applications provide necessities to Ketogenic dieters however require payment after installation	We discovered a couple ("KetoDiet", "MyKeto"; both found at apple app store) were advertised as "FREE" but require the users to pay to use majority of the features.
Applications on the market are only available for mobile devices and there are none with desktop-preview capabilities	Popular fitness applications such as "fatsecret" and "myfitnesspal" are well known and often credited for their ability to showcase user progress on both mobile and desktop; hence why Ketogenetics must be displayable on different devices.

A2: Detailed project plan

12. Analyse our previous primary and secondary market research.
    - Evaluate our research on competitive applications to identify their strengths and weaknesses
    - We need to review our stakeholders' needs.
13. Reviewing our designs and prototypes
    - Ensure our proposal meets our requirements
    - Discuss weaknesses in our plan that we need to change
14. Research technologies to aid the implementation of our web application
    - Use the prototypes as a guideline to find and compare technologies which we can use to reach our desired designs.
    - Research into different libraries and frameworks to get a product which resembles our prototypes. Find technologies to manage and streamline our repository and files
    - Research databases we can use and back end scripting languages
15. Decide on a methodology to follow and propose a development plan
    - Research different models to ensure final delivery of working product
    - Discuss techniques to maximise completion of tasks
16. Implement the front end of our application
    - Create the first layer of our architecture model with high focus on the design of user interface. It must be easy to use.
17. Put our application up online
    - Research cloud server hosting/ Research for deployment
18. Implement the back end of our application
    - Develop a schema and use this proposed design to build the database
19. Implement methods to fulfil core logic layer of application
    - Research formulas that could be applied
    - Create a connection with database and merge the front end with back end
20. Testing the application
    - Test the application during and after implementation to ensure quality of product.
    - Research and experiment with user testing methods
21. Evaluate the application
    - Assess the usability of application
22. Research and propose future strategy for marketing

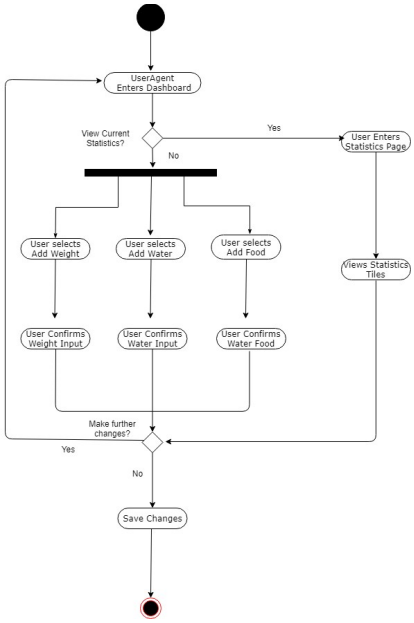
Appendix B: Design and Implementation

B1: UML design review

B1.1 Behaviour diagrams

Activity diagram

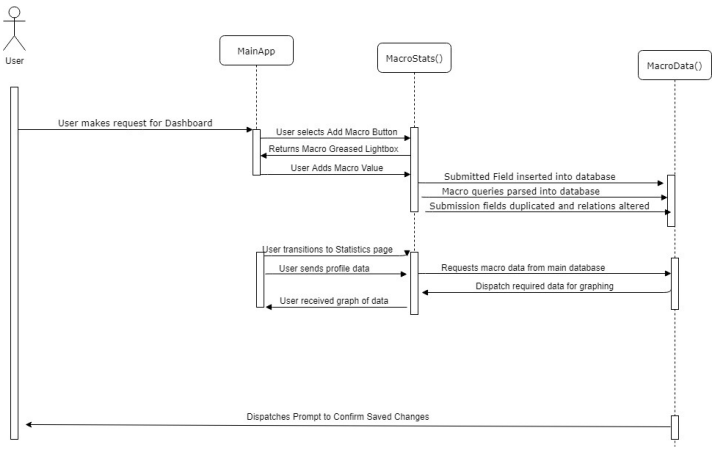
Activity diagrams capture the step by step process of the user's behaviour whilst using the system. This helped us understand the decisions which our users would face and what our application could provide to aid.



The following activity diagram shows the work flow of the user's ability to visit the statistic page and update their macros for the visualisation of statistical progress. The user logs in (pre-condition) and enters dashboard (system step). If they want to see their progress tiles, they would visit the statics page (expansion flow). If the user decides to stay on the dashboard, and do not want to view their progress, they may choose to update it instead (alternative flow). They can add their weight, water and food from the dashboard and save changes (returning flow).

Sequence diagrams

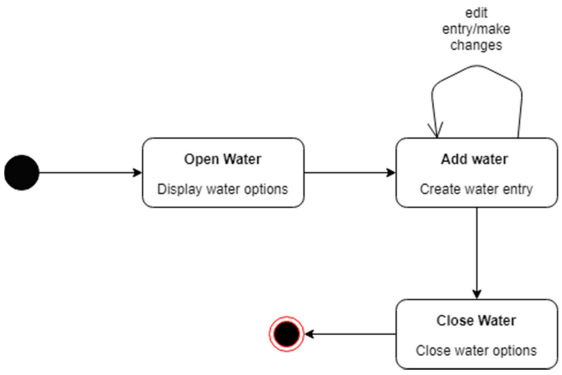
Sequence diagrams depict the user's interaction with the objects of system in a sequential order. This helped us understand how and in what order objects in a system function; certain events cannot take place before another event.



The following diagram shows the user's interaction with the main application's objects through the dashboard. The user could select a button which changes their macro data (their goal weight, water consumption, food intake and current weight etc) and this could alter their personalised profile data. They could change their profile picture which would also alter their profile data. Their progress is being saved all along.

State chart diagram

State chart diagrams clarify the purpose of system. They model the lifetime of the data object; the defined states are controlled by internal and external events.



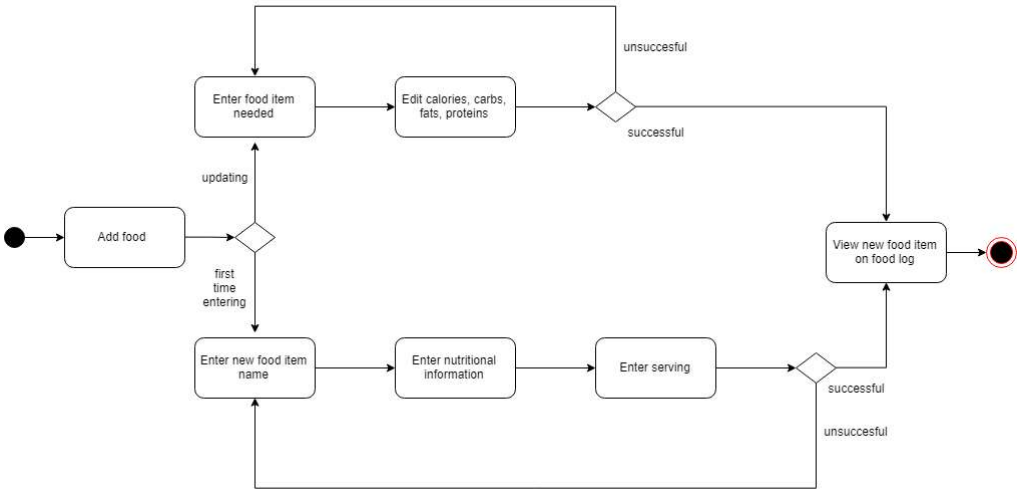
The following state chart diagram models water consumption of the user. We start with an idle state. The following states can be seen as requests; when user clicks on water button, a box would pop up. This displays the option to add water consumed to user's profile. The entry can be edited and changed as user inputs. Once they save changes, they can close. This completes the lifecycle for data object which is transferred to database. If the user wants to remove their glass of water, they would need to go through another cycle. Adding food and weight will follow similar approach.

B1.2 Structural diagrams

Component diagram

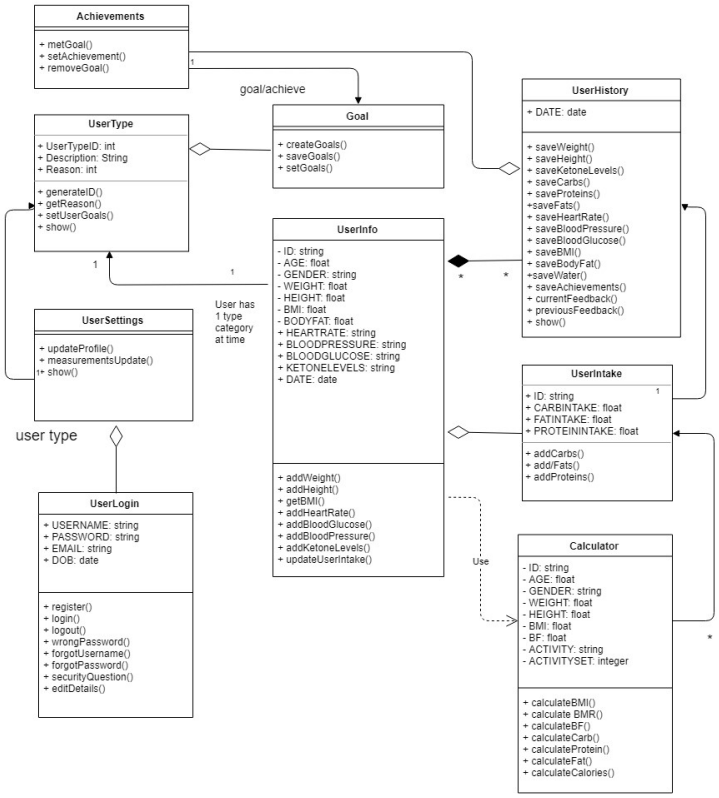
Component diagrams do not describe the functionality of the system but describes the components which are needed to create those functionalities.

The following diagrams models food in our system. The user may choose to add food which will be stored onto the database for the user's profile; this is a functionality of our system. The diagram demonstrates what we need if the user is entering for the first time or updating. It shows the extra components that are needed such as adding/editing calories, carbohydrate, nutritional information, servings etc and each individual piece of information affects other parts of the system. Depending on the success, this would be logged onto system. Component diagrams allowed us to understand the relationship amongst artifacts (food input) and the application.



Class diagrams

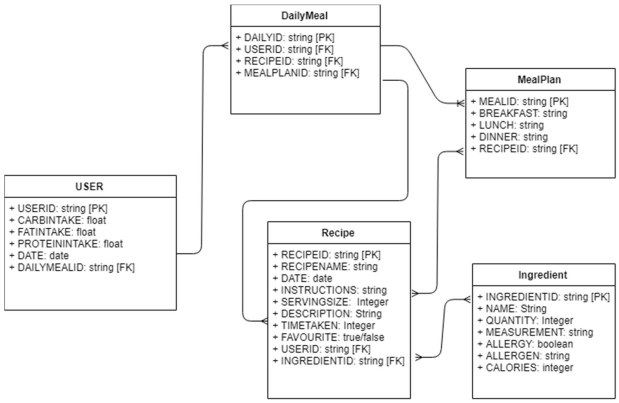
Class diagrams describe the operations and attributes of our application; it is modelled with a static view and analyses the responsibilities of our system. We learnt if the user does not add to the system, then they may not actually be able to take as much from it.



The following diagram shows the links in the system to create the users' profiles. The user information that the user provides would allow us to create other features. We can see which features are necessary for other features to exist; this allowed us understand which features we need to work on first. The user's profile uses a calculator to create a bio-profile. This is used by other parts of the system such as user's history, goals and macros. Other features such as achievements are reliant on goals and nothing else. This means if there is no user history record then system cannot create goals and if there is no goal being met then there can't be any achievements to produce and show.

ERD diagram for meal planning

ERD diagrams model the structural design of a database; it shows the relationships between the entities. Creating this allowed us to understand the back end of our application better since now we had a working plan drawn. It also gave us skills to create a scheme for our actual application.



The following diagram illustrates a small part of our database (for meal planning). The "user details" table which contains the user's macros data would store the user's daily meals. The daily meal table is connected to a meal planning table and recipe's table; these are what users can add to their profiles as consumed. Recipes are connected to individual ingredients (this allows user can search recipes based on food items etc). Individual recipes can be added to favourites as well. Meal plans are created using the recipes and daily meal's table can access these meal plans.

B2: Prototype design review

B2.1 Conceptual prototyping

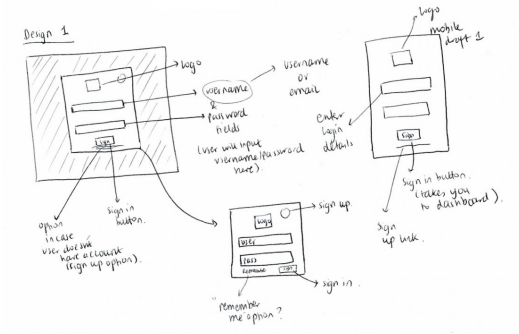
Our conceptual prototypes for our web application consisted of paper wireframes; these were cheap, easy to make and quick to dispose. The front end group created multiple designs for different screen sizes and devices; these did not portray features of the application in detail.

We presented the basic designs to our stakeholders who expressed their opinions on the structure and layout of the system. They responded to our open questions and made comments on individual designs. The front end team collected the feedback and presented it to the rest of the team and we worked on improving the designs based on suggestions.

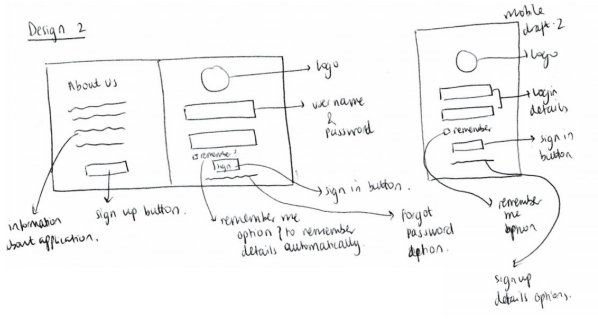
Our project proposal documents our prototyping and changes we made to our conceptual prototypes before we moved to functional prototyping. An example includes changes to our login page.

The following diagrams below illustrate the designing for our login and registration pages. These were shown to stakeholders who found the sketches easy to understand and gave feedback.

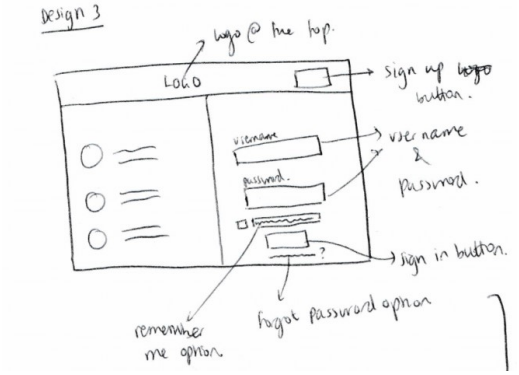
Design 1 – login page



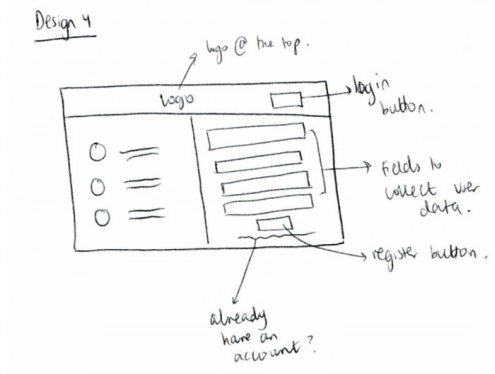
Design 2 – registration page



Design 3 – login page

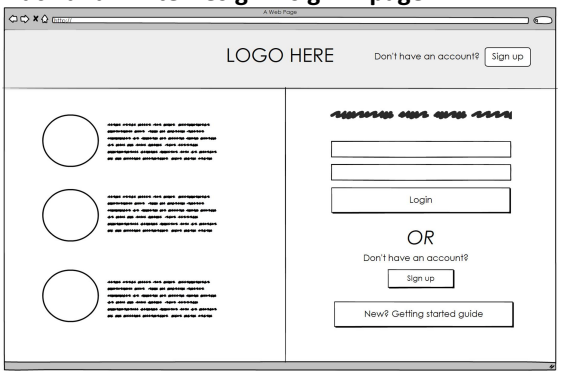


Design 4 – registration page



Initially, our “sign up” and “sign in” pages were going to be one page however a user suggested that separate pages for logging in and registering would be easier to use so we incorporated this idea into our application and finalised this on the black and white sketches in next stages (these are shown in the table below).

Black and white Design – sign in page



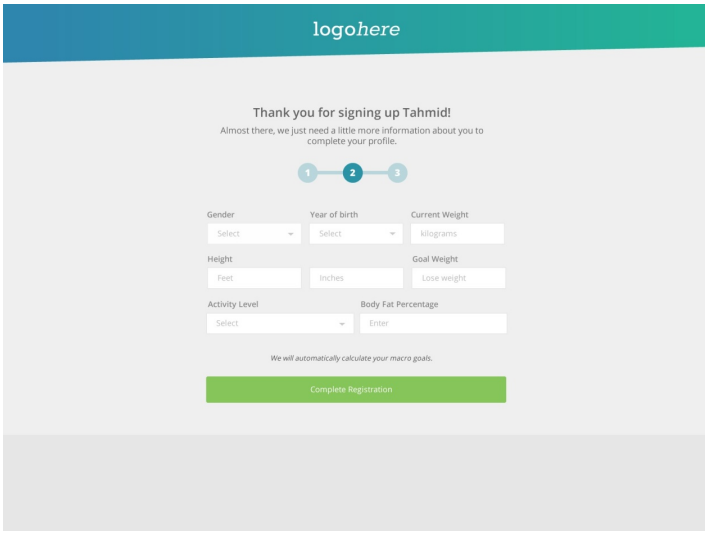
Black and white Design – registration page

B2.2 Functional prototyping

These were designed using Fireworks, Photoshop and Adobe xd. We created templates on Fireworks and Photoshop and then transferred them over to Abode xd, where we linked the static pages, giving users the ability to navigate through the prototype as they would on the web application. During the exercise, we were able to see the how users behaved with our prototype.

The stakeholders had to answer some open questions regarding the usability of the proposed system. They had witnessed what the proposed system would look like before it was built so we used feedback to make changes for the actual development.

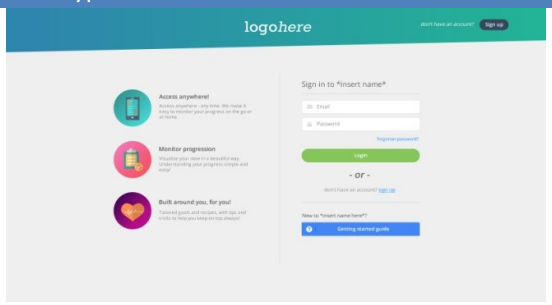
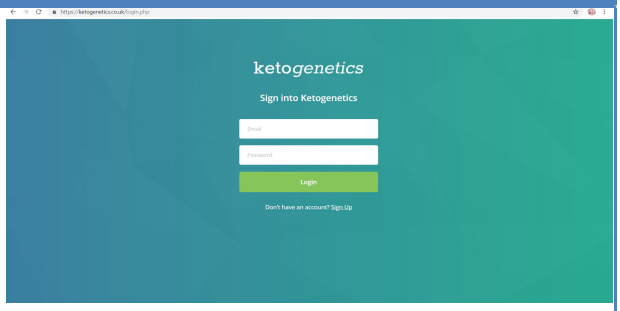
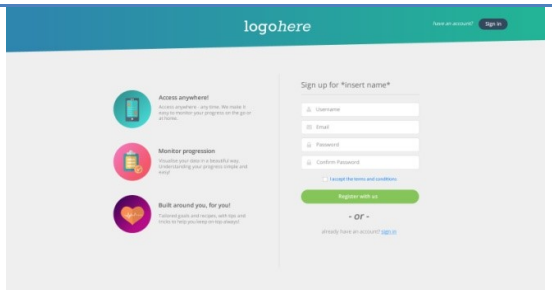
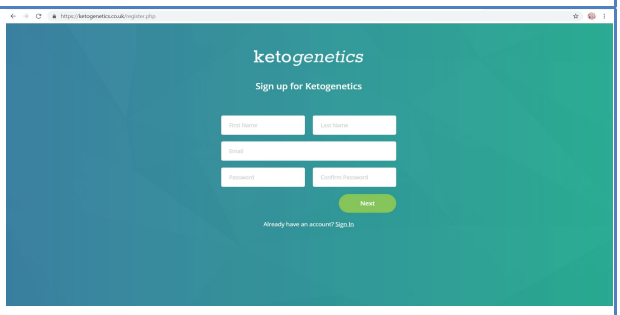
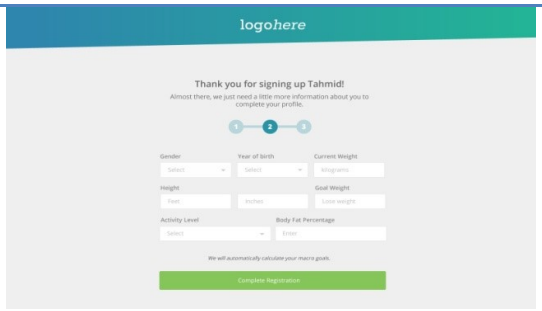
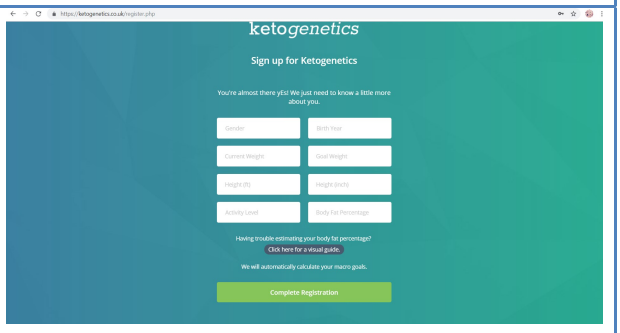
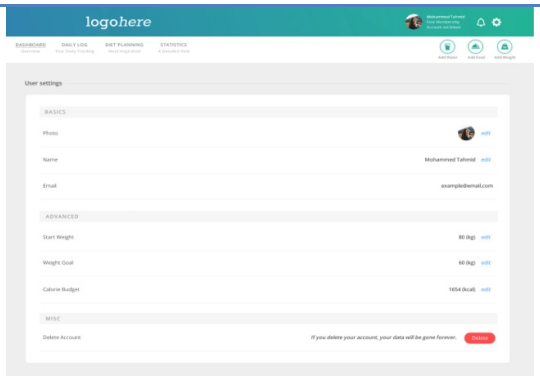
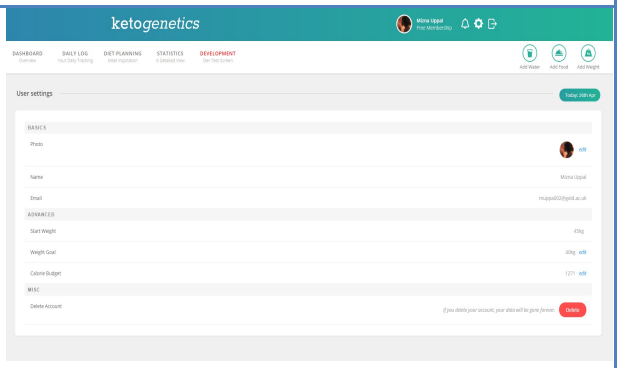
The prototype below shows registration page of our application.



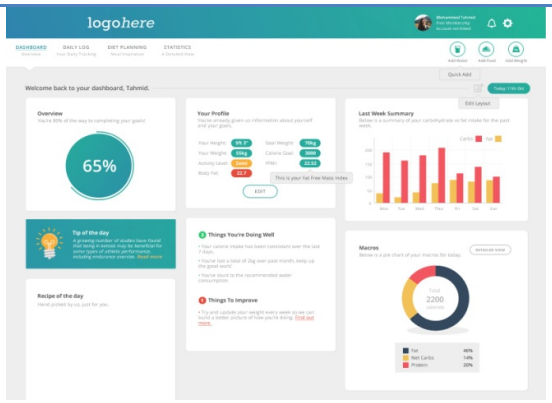
The prototype below shows dashboard page of our application.

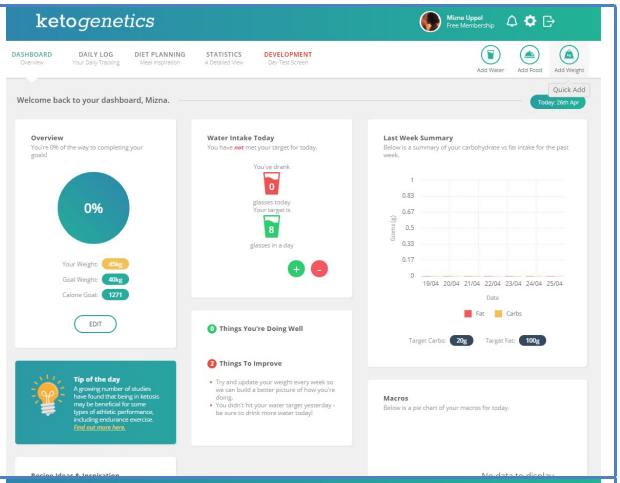
B3: Final product design

B3.1 Final product vs prototypes

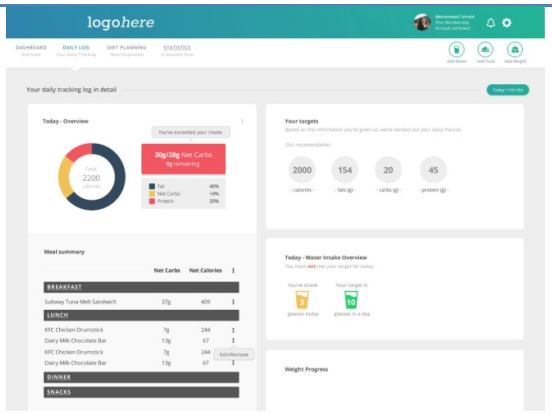
	Prototypes	Final
Login page		
Registration page		
Sign up page		
Settings page		

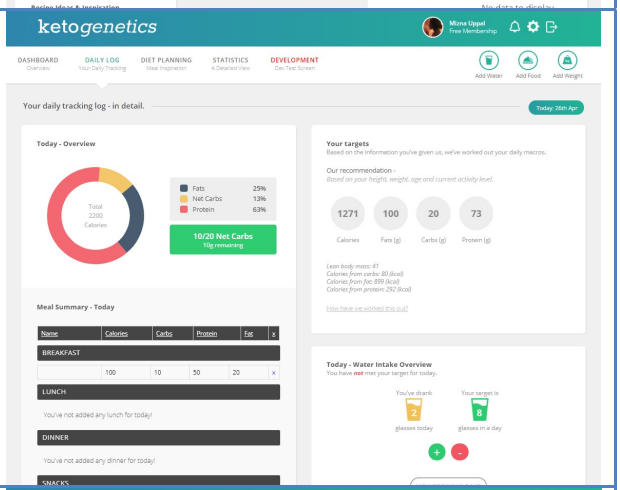
Dashboard page



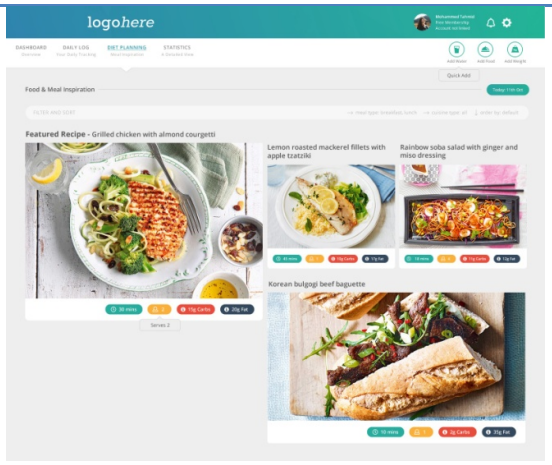


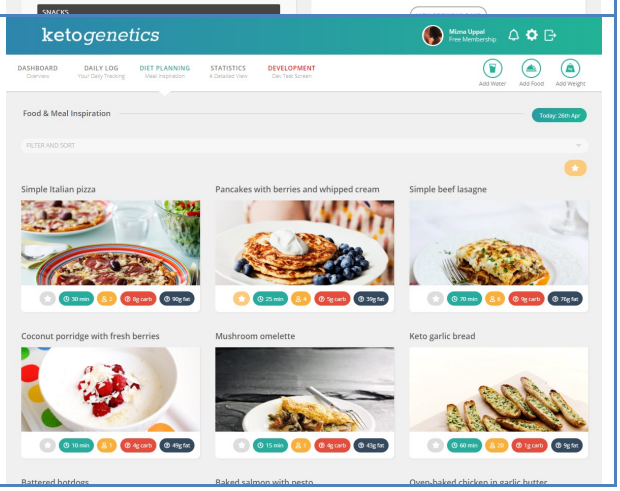
Daily log page

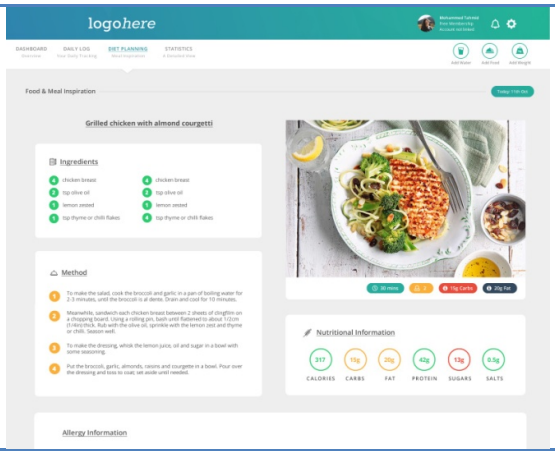
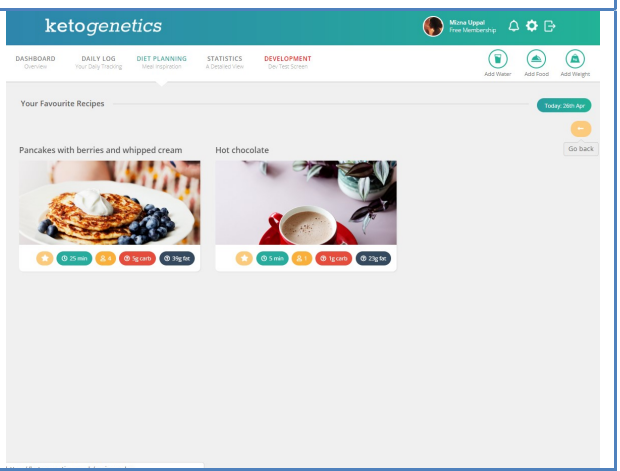
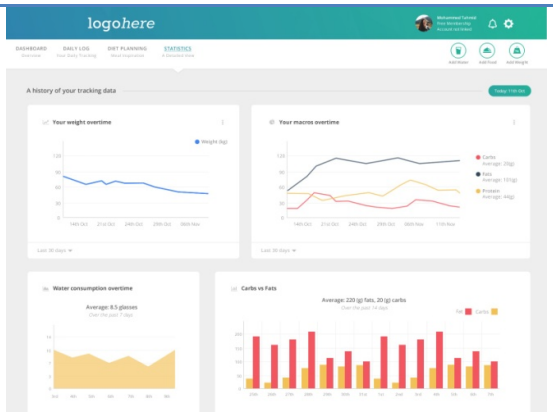
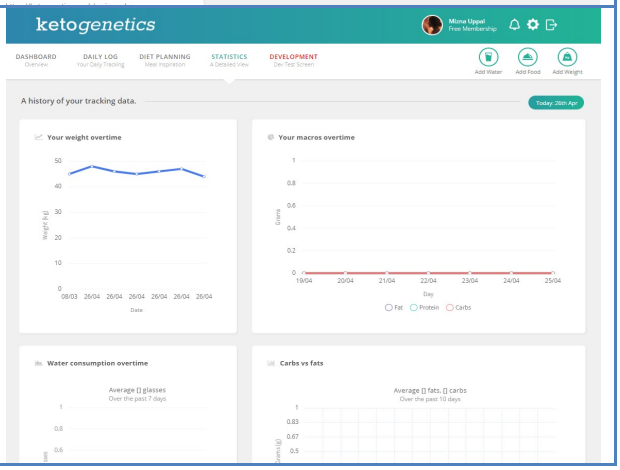




Diet planning page





Diet planning page for each recipe		Feature removed Clicking on the recipe redirects to external link that has the recipe from the site we took it from.
Favourite recipes	Feature created after suggestion from clients. It was not on our prototypes.	
Statistics page		

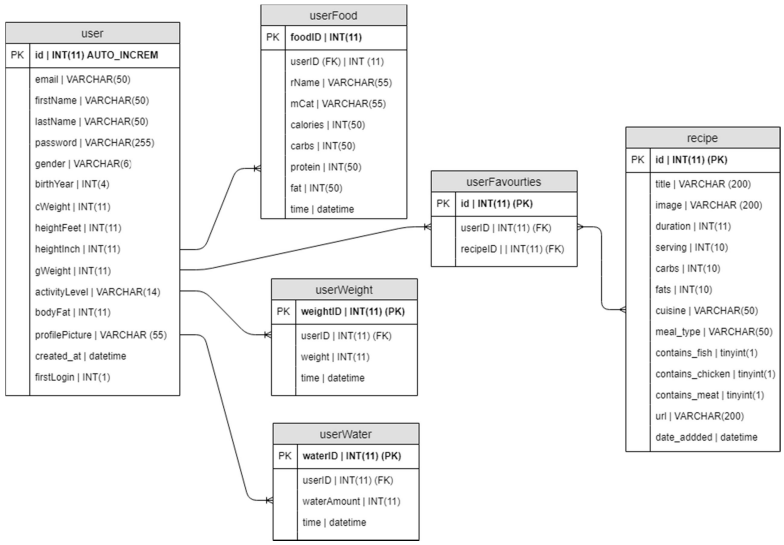
B3.2 ERD diagram explained

Our “user” table will contain all the user related data; this includes personal and progress information. The user table will be related to another table (userWeight) which will record the weight being added each time; we will assign each input weight with a timestamp to create a timeline with weight progress.

We will have a “userFood” table which will be used to calculate the user’s macros after user inputs their calories, carbohydrate, fat and protein intake for each meal/food item consumed. Each meal will be assigned a timestamp so we can manipulate data stored in this table to visualise the user’s intake goals and overall progress over a certain period. We will have a “userWater” table to store the amount of water the user is consuming. Every time the user inputs a new quantity of water consumed, it will be assigned a timestamp and stored onto our database for that user.

Users will be able to add recipes that are stored in the “recipe” table to their “favourites”; these favourite recipes will be stored on a separate “userFavourites” table along with each user’s ID to filter each individual’s favourite choices of recipes.

The following ERD diagram shows the relationships between the different tables.



B4: Implementation

B4. 1 Updates extract

Extract of implementation updates

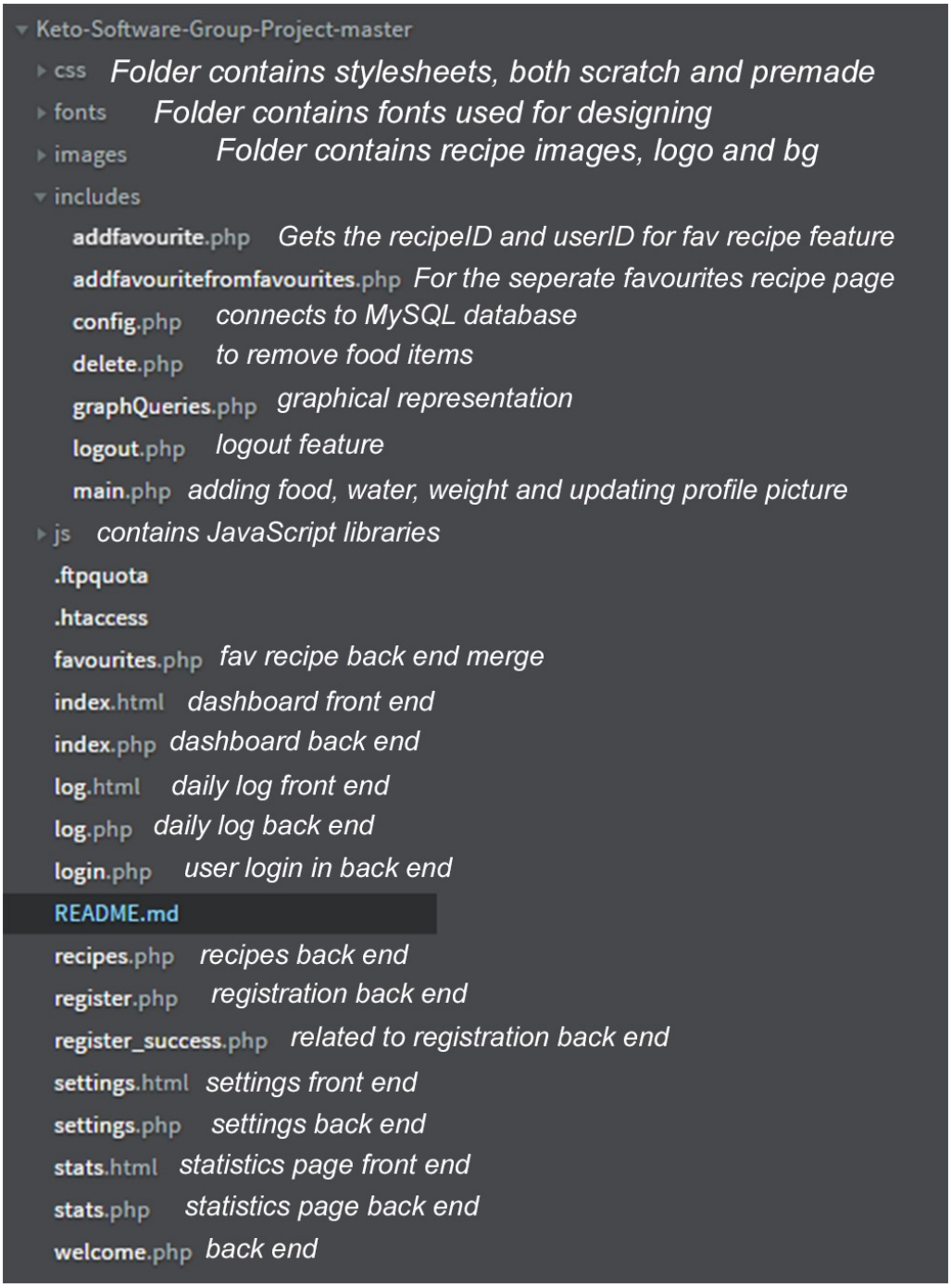
These have been taken from development log to demonstrate our implementation procedure and goals we were meeting.

From 25/12	***daily macro and calorie total calculations** ***mobile bug fixes** ***sweetalert forms** ***added water tile to dashboard** ***login page revamp** ***server-side bug fixes**

	***more additions to the dashboard (it should be now 80% complete frontend)** ***more additions to backend/calculations (lean body mass, recommend protein, carb and fat intake numbers and the calories you should be consuming from these) as well as percentage progress for weight.** ***formulas adjust with your activity level and gender** ***major bug was fixed where users' initial sign up weight wouldn't be recorded to weight history.**
From 28/12	*implementation of charts on dashboard using Fusion Charts library *Fusion Charts for carbs vs fats has also been implemented on welcome.php (chart will fill up as days go by as you add food) * backend (welcome.php) and frontend (index.html) completed * bug related to server time zone vs our time zone but there's no fix for this as we do not have dedicated hosting *change of formula to calculate goal calories - changed to Mifflin St Jeor Equation instead of Harris-Benedict formula and the usual code cleanup and bug fixes.
From 01/01	* slick carousel on the dashboard for the recipes section, still experimental and may be changed in the future * 2 new graphs on the dev page - water and weight graphs (that's almost all the graphs done!). * 4 graphs fully functional * graphs do not throw any server-side errors in the Apache error log *several bug fixes fixed and pushed (update on Trello board) * similar to Photoshop/Fireworks design * interactive and animation too
Changes/updates between 07/01 - 10/01	* Changed how water is added to the database. Now a new row is created for each glass of water. This allows us to implement the quick add and remove buttons. * Added and implemented quick add and remove buttons for water * Added and implemented ability edit/change goal weight * Added navigation triangle * Separated some common js to separate file * Added statistics page (front-end) * Graph type and style changes * Other minor style/css changes across pages * Other minor bug fixes
Changes/updates between 10/01 - 14/01	* Separated food log into meal categories - breakfast, lunch, dinner, snacks. * Added section to flag up if you've exceeded your carb intake * Improved how carbs vs fats graph looks on mobile * Moved css to external file * Added ability to remove profile picture * Added daily log page * Fixed Firefox inputbox spinner issue * Fixed some spelling and grammar issues * Other minor style changes * Other minor bug fixes
Updates from 22/01 to 08/02	* Fixed upload picture form bug * Front-end/back-end merge for all pages * Fixed stats page graph for mobile users * Fixed bug with sql queries/ contact with GoDaddy and Cloudflare. * Fixed empty form submission bug for weight * Added notyf.js confirmations when adding weight, food or water * Added recipes page and favourites system * Added basic recipe filters * Added links to recipe page on other pages * Added settings page front-end * Added tips and tricks back-end

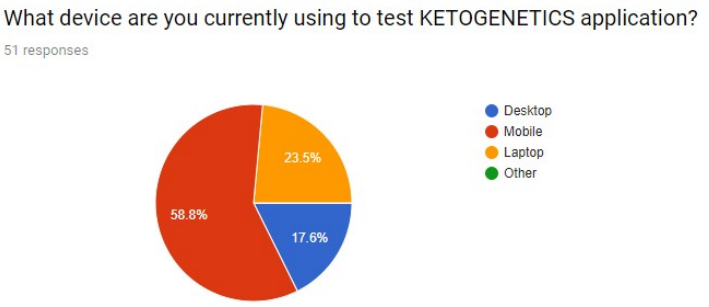
B4.2 Structure

The image below shows how we structured our files and work. It summarises what each file contains.

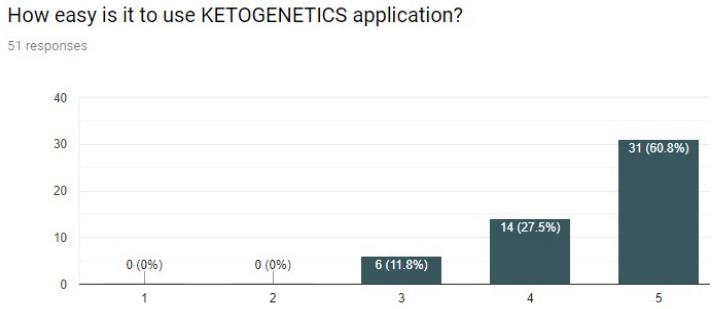


Appendix C: Testing and evaluation

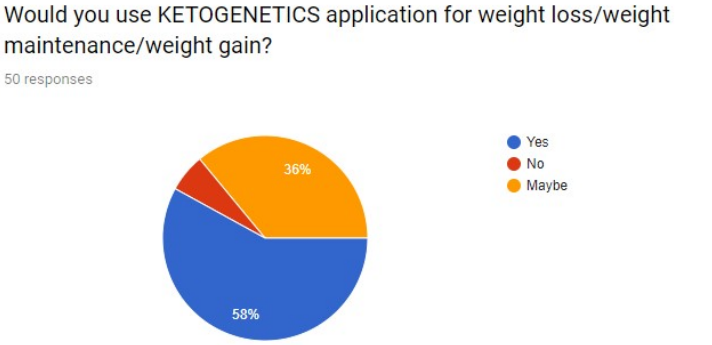
C1.1 Results from Online survey



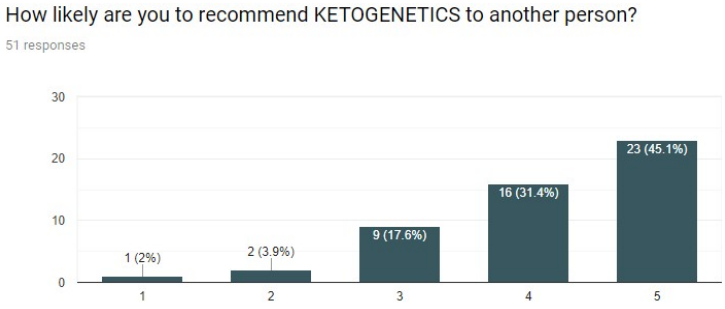
We asked the users which device they were using when testing our application. Using Google forms allowed us to check each individual response and see if the type of device can affect a user's experience on our application. We found on average laptop and desktop users were likely to give the application a higher rating compared to mobile users which meant we needed to work more on our mobile version.



Most of the users found the application easy to use with 5 being the highest rating. This concludes the test for "ease of use of application" to be a success since so far the responses showed it is not hard to use.



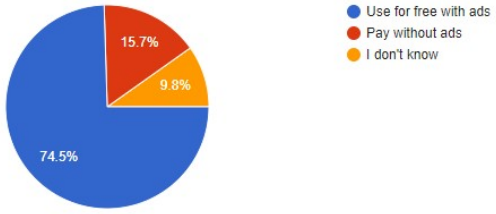
We asked the users if they would use our application to which we received positive outcome. Nearly 60% of users are willing to use our application and over 90% is willing to try.



Some users are likely to recommend this application whilst others aren't. We gave users a chance to give feedback with improvements regarding the application towards the end so that we can see what they didn't like – which would improve our chances of making application even more marketable and desirable to audience.

Would you rather use this application for free with ads or pay without ads?

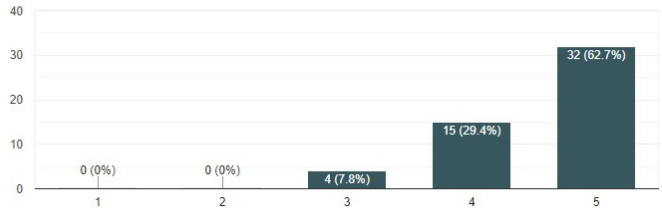
51 responses



We wanted to know whether the users would be willing to pay and were surprised *some* were considering our application can still be immensely improved.

Rate the quality of this application

51 responses



The quality of our application currently stands at an average of 4 stars. This proved our application is easy to use.

C1.2 Feedback

What went well whilst you were using KETOGENETICS application? Were there any features which you liked?

26 responses

- The layout
- Nice layout
- This was a really good app, well done guys!
- I loved the little round circle which lets you add stuff
- THIS WAS AMAZING
- Like the simple interface, very easy to use and gets the job done.
- The app looks clear and concise
- This was really good, I loved it
- I liked the layout of the application
- really like the responsiveness of the application
- Professional looking user interface
- Everything was very intuitive

I liked the colour scheme and layout

The response time

The layout and theme was consistant throughout using the app. I liked the graph feature which specified how much water/carbs intake that i have taken

It was nice and easy to use

It was very easy to use

It was easy to register and to operate

Very intuitive and professional looking

Amazing service

The application looks good and has nice concept, with improvements it will be pretty great

Good quality

The animation stayed consistent (pop up) and also the colours used. The navigation between each pages were very easy as they remained in all pages.

The animation stayed consistent (pop up) and also the colours used. The navigation between each pages were very easy as they remained in all pages.

I like the idea that I can add glasses of water that I had in a day

What do you think could be improved? Anything you didn't like about the application?

23 responses

Was confusing (would put 5ft8 but would say enter a number)

I am not sure how to improve, it was good

Maybe let users choose dark mode since the application is so bright when using at night on phone

N/A

When signing up, the user has to enter all the weight and height information. User should be given an option to add these later if they need to.

Nothing that I can think of

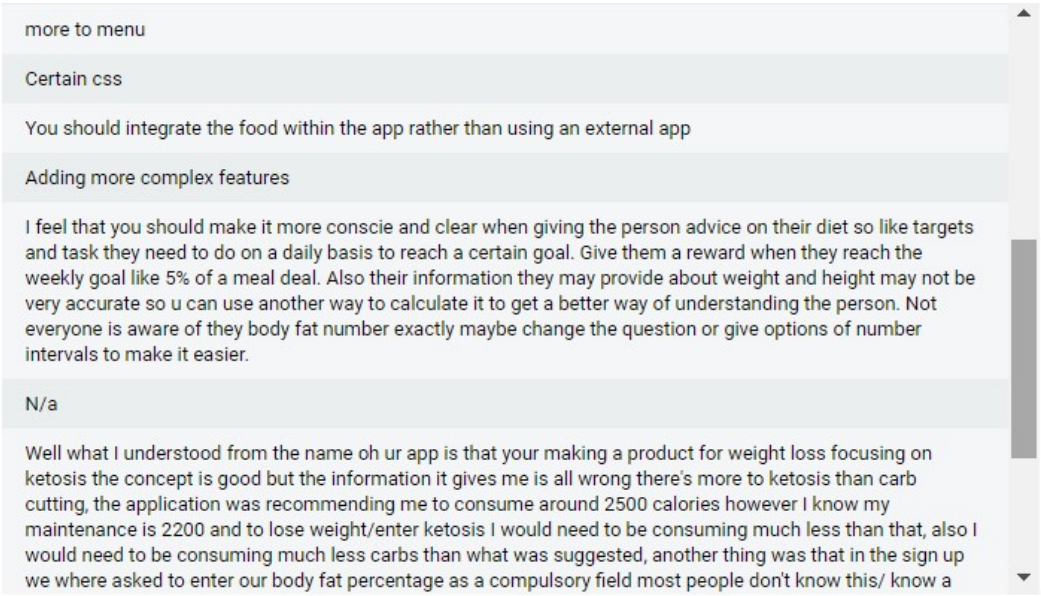
I want to be able to sign up without putting all my personal information in

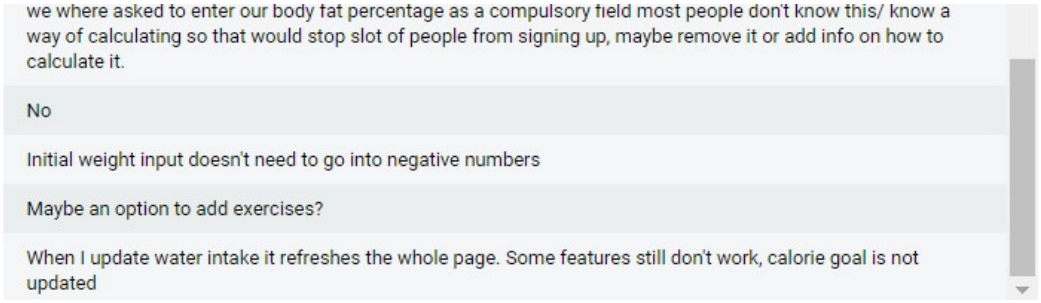
logo should be better

could use more pictures like the recipes page

Recommended calorie intake is too low, might be calculating something wrong.

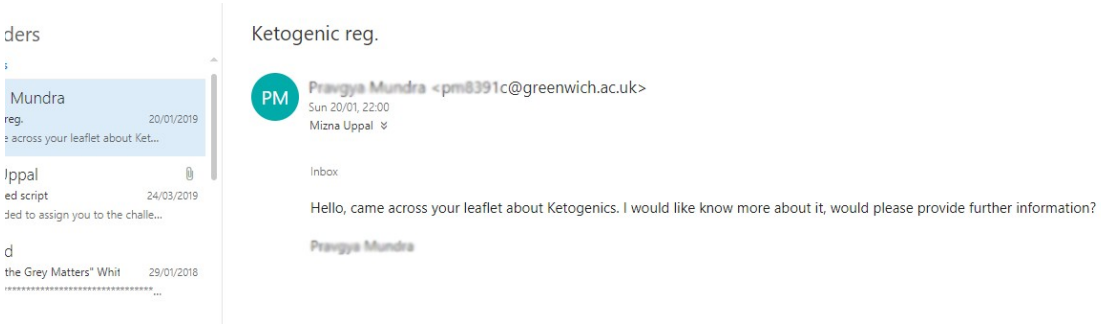
I didn't like the fact that you can only put your height in feet and that you have to put your body fat percentage, which I don't know and I think many people don't know as well



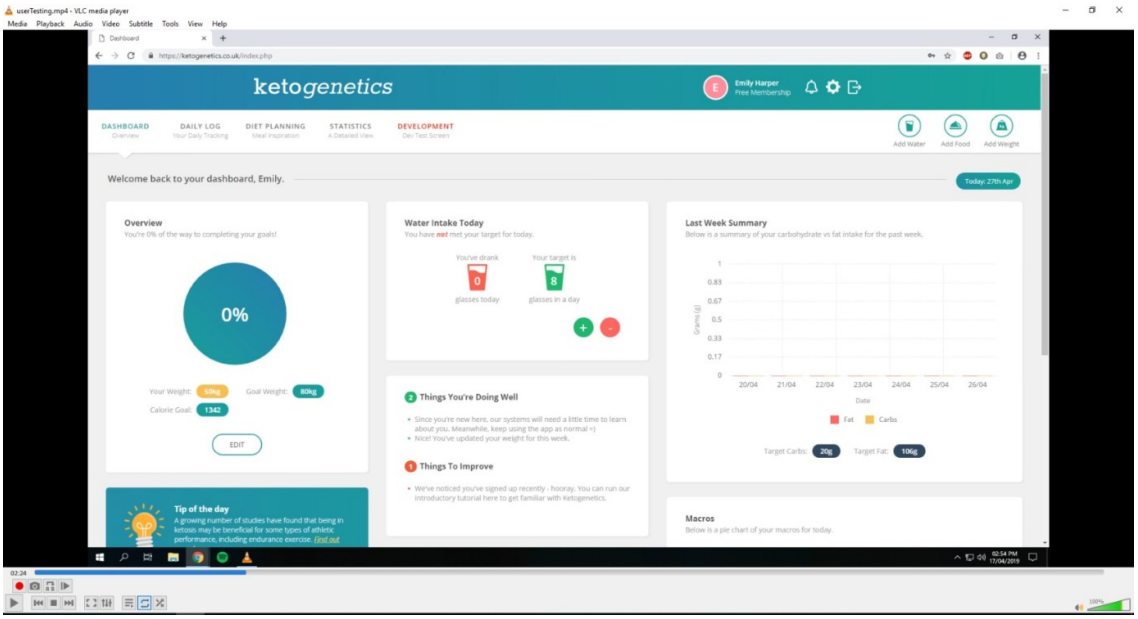


C2: Usability testing

C2.1 collecting usability testing users



C2.2 Screenshot of user's sent video



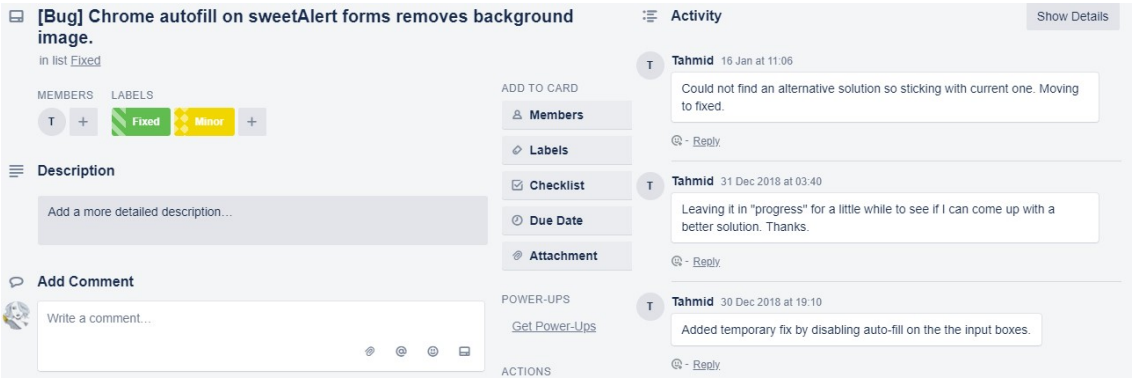
C2.3 Remote usability testing detailed

Exclusive to second study	Reflection
We had some new stakeholders. These people knew about the Keto diet and were interested in testing our application.	These members gave an unbiased and fresh perspective compared to the stakeholders who had seen our prototypes and earlier designs which allowed us to focus on what our real end users need.
It allowed us to visually observe the natural thought process of our participants as if they were actually using our application; we were able to see patterns for popular features (nearly 3/5 logged their water consumption for the day before we even reached this particular task and 4/5 logged this more than once; making it one of the most popular features).	We learnt random detailed information such as the amount of time it takes to complete the tasks that had been specified. Apart from most clicked, we also found features that were ignored and weren't appealing to users (they would not bother clicking on these during the experiment).
This test helped us to recognise the potential of our application based on its layout. For example, The features at the top of the application had a greater chance of being clicked on and explored compared to tiles at the bottom of the page.	We considered changing the layout of the tiles in our application to ensure what we want the user to click on is actually placed at the top.

C3: Quality testing

C3.1 fixing bugs

The following image below shows the developer commenting on an issue found by the tester which needed resolving.



C4: Deployment

C4.1 Goals and critical success factors

The following are relevant tasks for a successful deployment of Ketogenetics:

Critical goals	
Improvement to security features	The system keeps the user logged in and they need to log out manually; this needs to be changed so that the user is given an option regarding whether they want to stay logged in.
More work needs to be done for the personalised user experience	The application needs to be able to collect data and become adaptable to the user's interests/needs. Examples include: being able to suggest more improvements and things they are doing well.
Better method for calculating macros and calories using body fat	Though the current method being applied was found to be good (based on our research), our user testing and evaluation show this can still be a problem that needs resolving. Many people don't know how to calculate body fat especially those who are new to the Keto diet. We should provide an extra measure during the registration phase which can estimate body fat for the users.
Other goals	
Improvement to database of food	Back end of the application can be improved by allowing food items that user inputs to be added; details of the food item as well.
User reminders	Give the user reminders to update their water or weight/ ensure user is meeting macros.
Social	Feature for users to be able to interact with other users who are also using the application; this would allow them to share recipes, share progress etc and comment on each other's journeys.

C4.2 Future releases

The following are based on initial requirements (taken from backlogs) that can be added for future releases. We will consider implementing some of these to ensure that the software is kept relevant through future versions.

Meal planning	- **Being able to search for a recipe** - **Being able to add your own recipe** - **Commenting/liking recipes** - **Sharing recipes**
Macros	- See other people's macro tiles and compare - Change measurement for carbs, proteins, fats intake - Add vitamins and minerals
Water	- Add more methods to measure water intake - Add other drinks consumed since this impacts water consumption
Weight	- View other people's weight progress - Compare goals with others - Motivation tiles - Track body fat to actual weight - Different ways to measure weight (not just BF)

Appendix D: User Guide

Table of Contents

Introduction

PLEASE CONSULT WITH YOUR DOCTOR, OR OTHER QUALIFIED HEALTH CARE PROFESSIONAL BEFORE STARTING A KETOGENIC DIET.

This is the official user guide for Ketogenetics.

Ketogentics is a dynamic diet lifestyle web application designed for the purpose of aiding our clients in weight loss, weight gain and weight maintenance. This user guide will cover and explain several aspects of our application. This includes signing up, accessing the dashboard, viewing and tracking your macros, water intake, and many more other features.

What is a ketogenic diet?

The principle behind a ketogenic diet is simple. Cut down your carbs and eat more fat to burn more fat. Although this may seem counter-intuitive, the diet encourages the body to enter a state known as **ketosis**. Ketosis is a metabolic state in which the body fuels on ketones and stored fats for energy rather than glucose. The following diagram below shows what a ketogenic diet consists of.

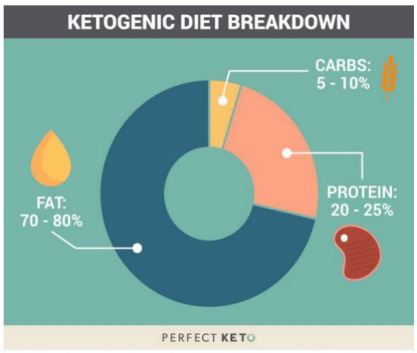


Figure 1 - Ketogenic Diet (source: https://perfectketo.com/keto-macro-calculator/)

Why should I go keto?

The incentive to start a ketogenic diet may be different for anybody, considering its various health benefits. The ketogenic diet's origin dates all the way back to the 1920s. Since then, thorough research has been conducted on why it may be proven to be a more preferable choice compared to traditional diets. Below are a few links which explain the benefits of a ketogenic diet…

- Ketogenic diet benefits
- Why is the keto diet good for you?
- The Benefits of The Ketogenic Diet

New Users

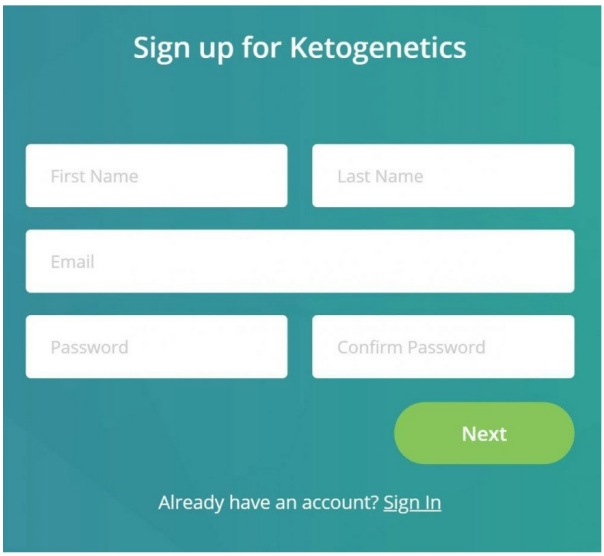
So, we hear you are interested in starting a ketogenic diet? Well, you are at the right place. Whatever your goal is, our web application will be sure to ease the process for you. The first step is to create your account.

User account details

Access our website by clicking here.

On the very first page you will be prompted to enter your account details and sign up. This will include your first name, last name, email and password *(as shown below)*. Remember to enter the details correctly as you will **not** be able to change them later.

If you do already have an account, click on the *"Sign in"* link or click here.

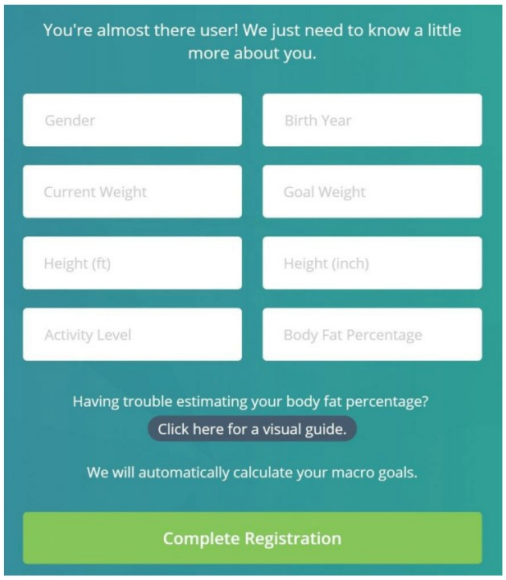


Personal details

The next thing you will need to do is to fill out some personal details on yourself such as your weight, height, activity level, etc.

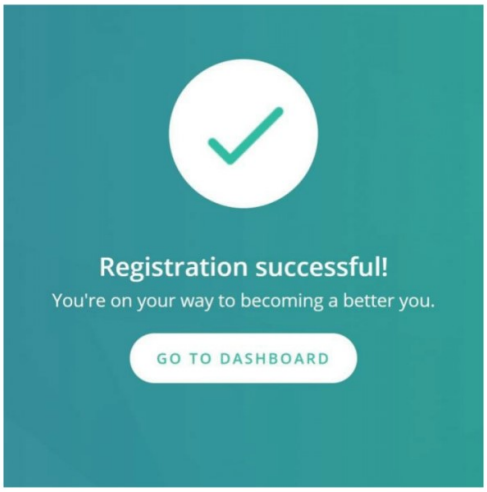
Please do ensure the values you enter are as accurate as possible.

These values will be used by our application to calculate your macros for the day. The macros targets are based on a formula and will be different for every individual.



Success

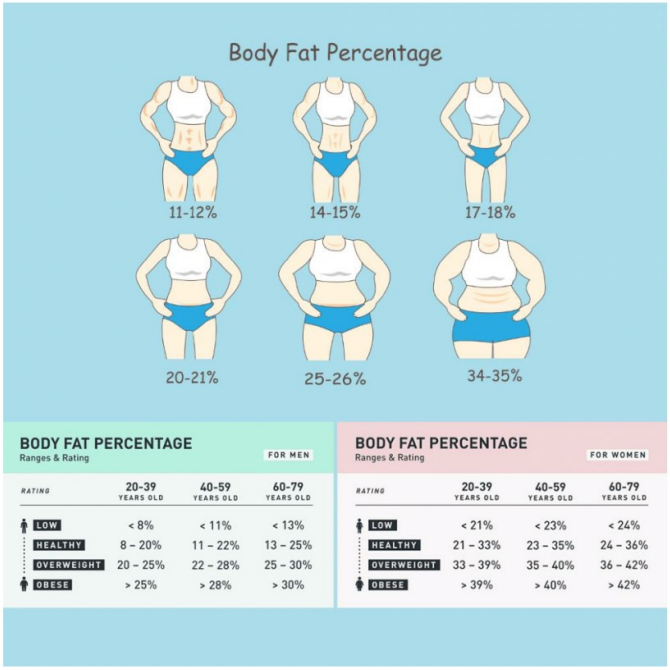
You are now ready to start using Ketogenetics and are on your way to becoming a better you.



Calculating body-fat

For some users, difficulty may arise when prompted to enter their body fat percentage. We would advise to use the following site: Body Fat Calculator or the following diagram below.

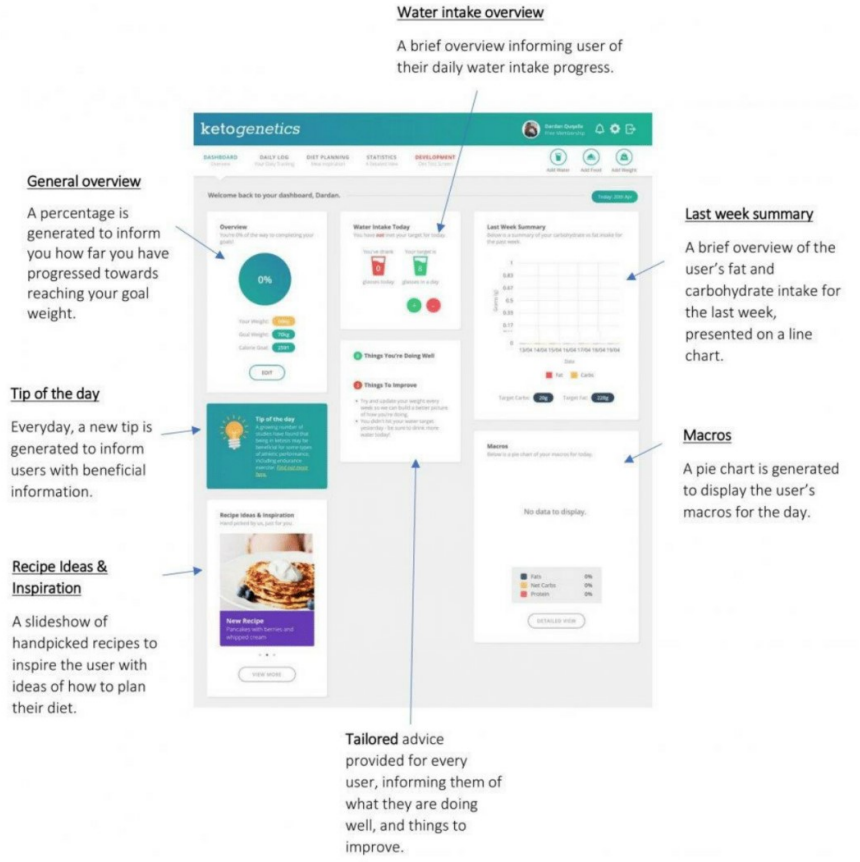
Please note: the link and diagram will only give you an estimate of your body fat. For a more accurate result, please consult a professional.



Dashboard

The dashboard has been designed with the aim of providing a simplistic overview of the user's data. From the main dashboard users will be able to observer their daily macros, water intake, and even update them too. In addition, users will be able to observe daily tips, tailored advice, and handpicked recipe.

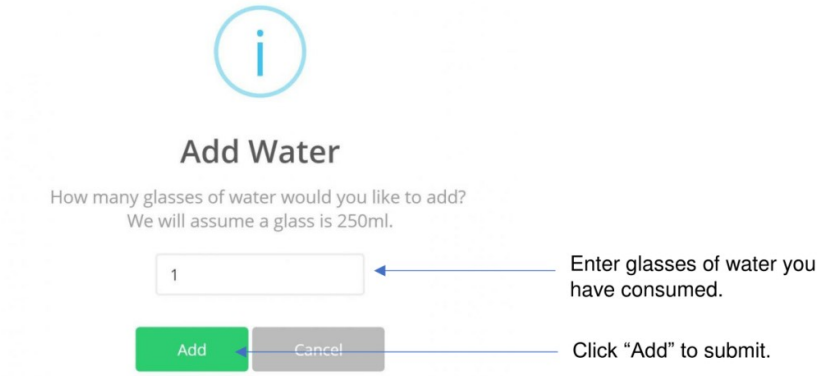
Overview



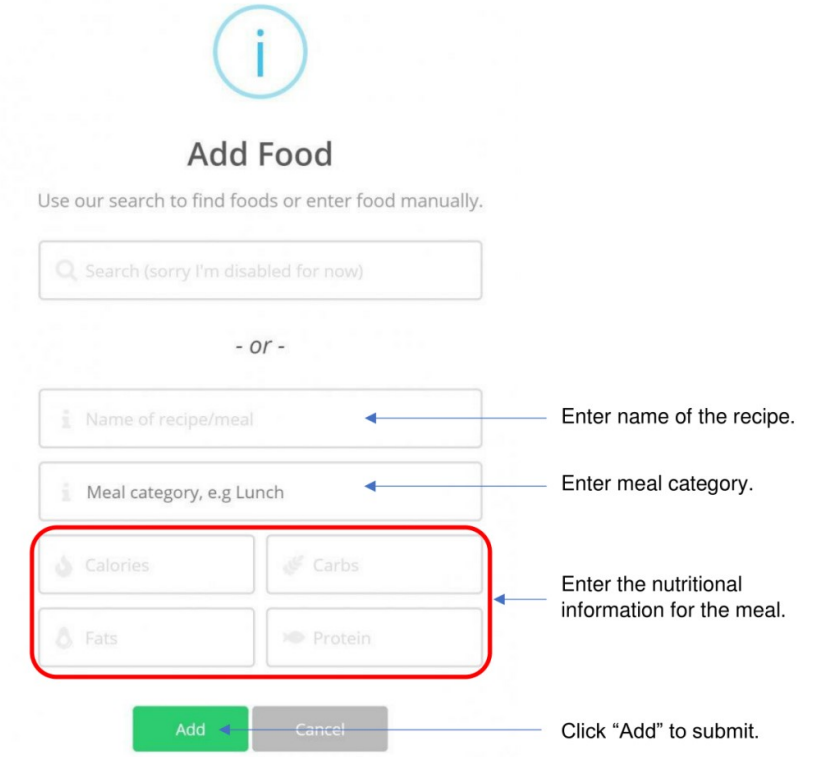
Quick-add

Quick-add allows users to quickly update their macros, water intake, and weight.

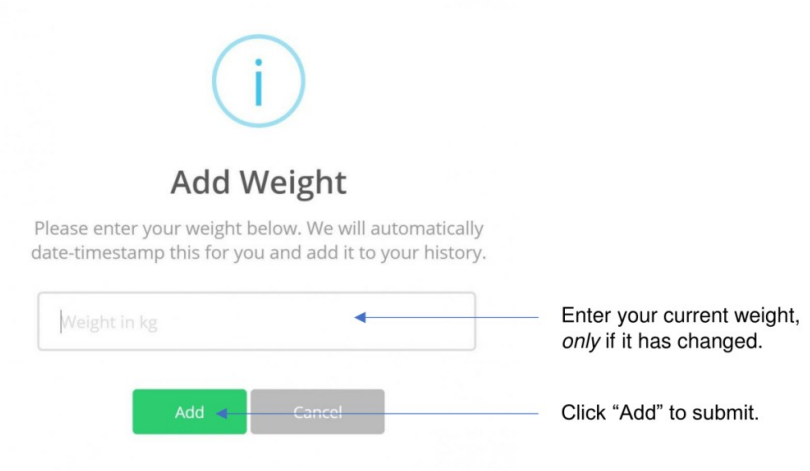
Updating daily water intake



Updating daily food intake

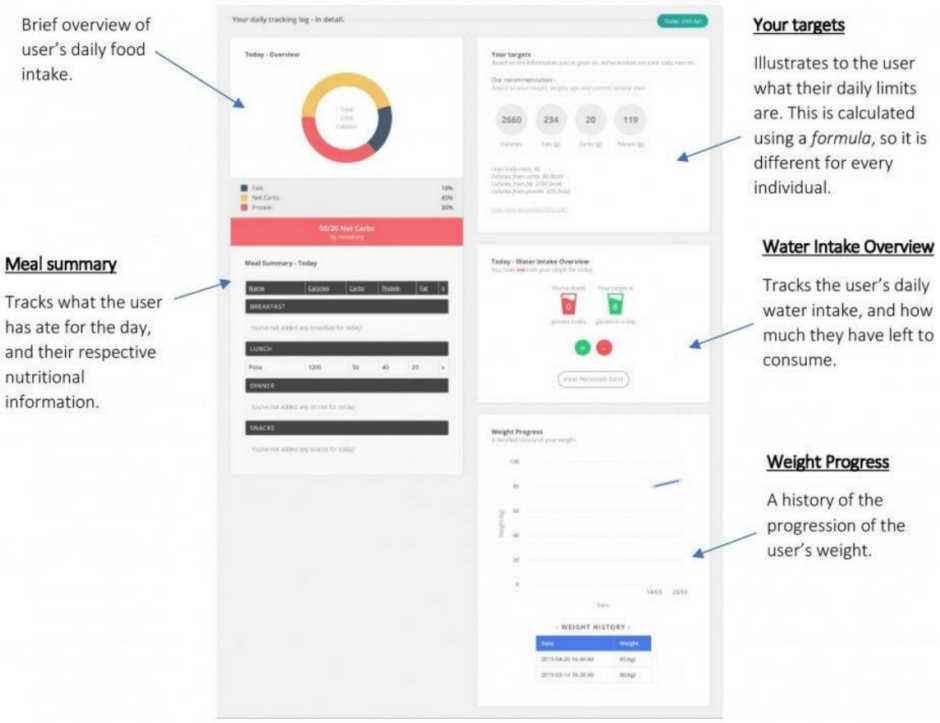


Updating weight



Daily Log

This is where your daily data will be stored. You will also be able to see your daily targets for the day, water intake overview, meal summary, and weight progress.

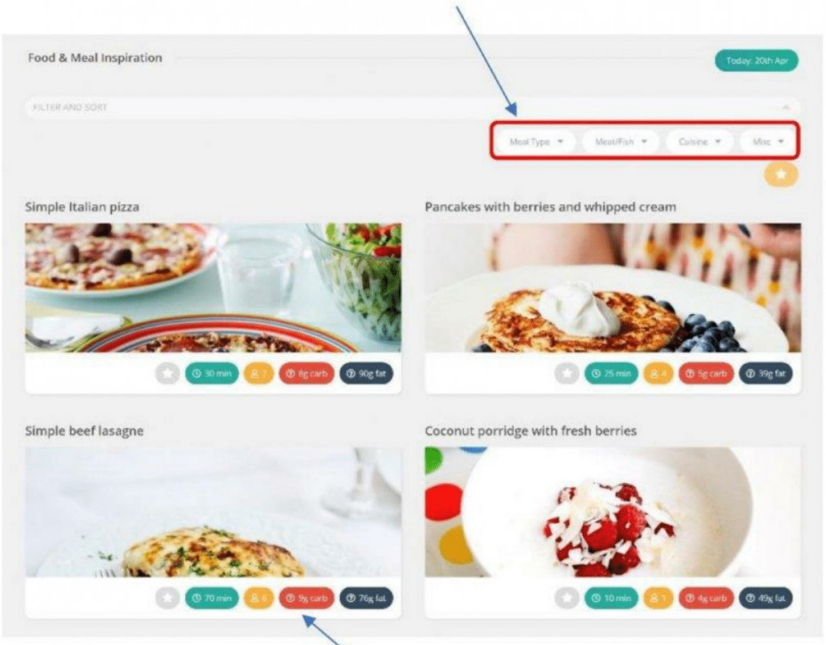


Diet Planning

The diet planning page focuses on providing users with recipe ideas and inspiration. On this page, users will be able to filter recipes depending on their preferences, and even favourite them too. How convenient!

Filter

Filter out what recipe ideas you are looking for. This could be based on meal type, meat/fish, cuisine, etc.



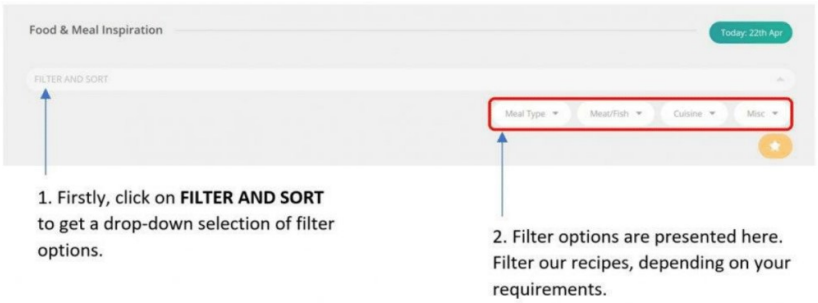
Recipes

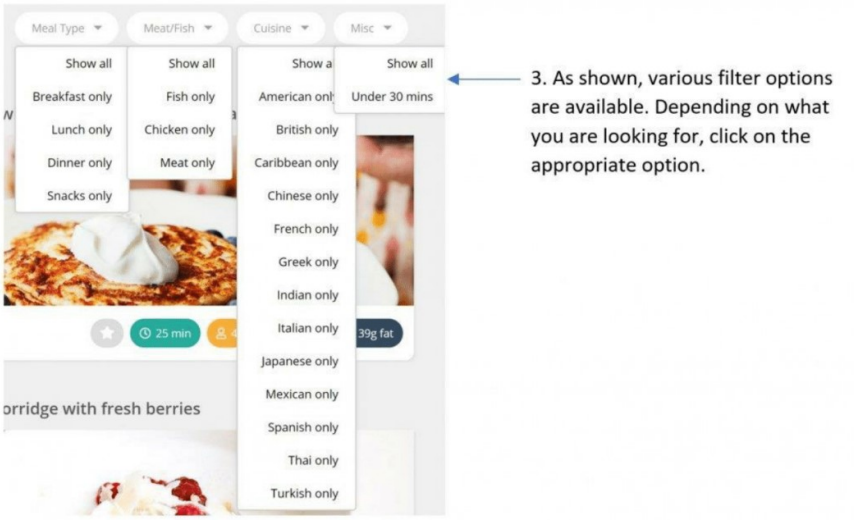
Recipes are presented in an easy format. Information such as cooking time, servings, nutritional information are provided.

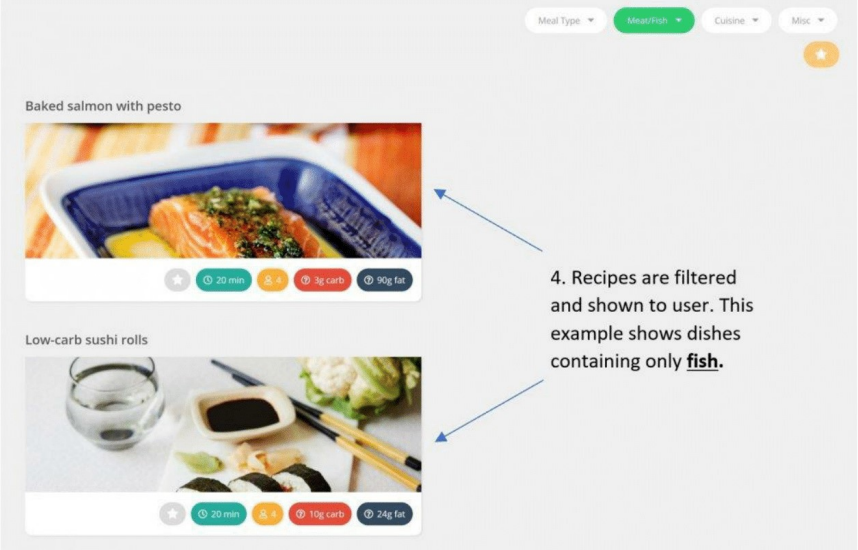
Filtering Recipes

Users can filter our recipes depending on their preferences. You can filter out recipes based on:

- Meal type: Show all, Breakfast only, Lunch only, Dinner only, Snacks only.
- Meat/fish: Fish only, Chicken only, Meat only.
- Cuisine: British only, American only, Chinese only, French only, etc.
- Misc: Show all, Show under 30 mins *(cooking time)*.



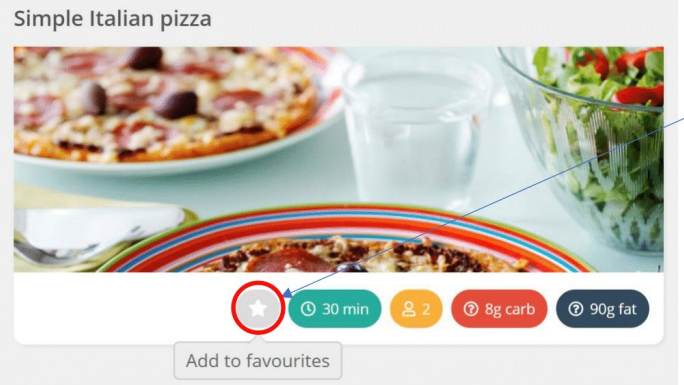


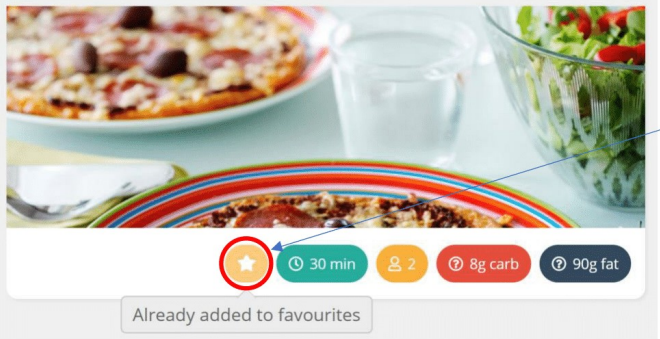


Favouriting Recipes

We have included a feature to favourite recipes so that you can view them later. It is very simple to do so.

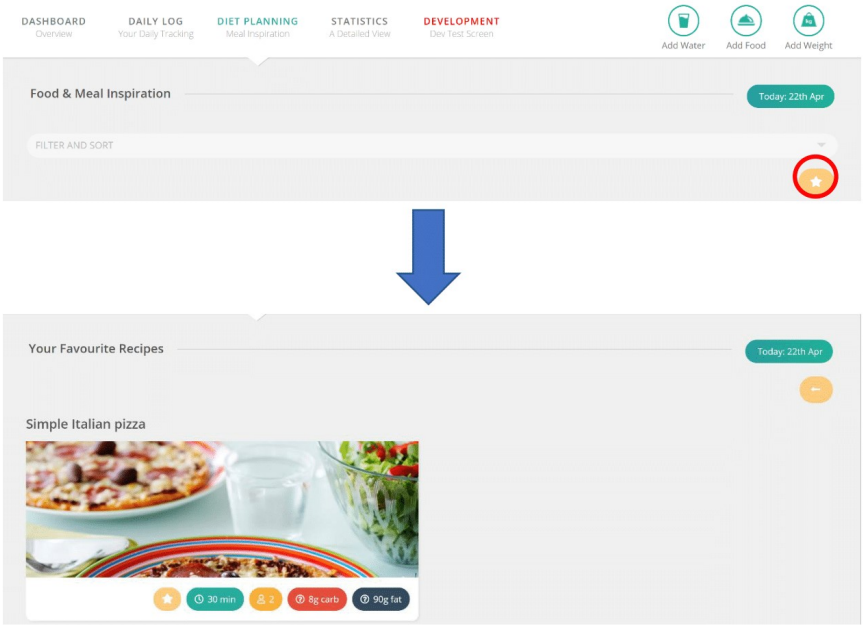
Firstly, find a recipe you like on the diet planning page.





The star icon will turn gold, indicating the recipe has been added to your favourites.

To view your favourited recipes, click on the star icon at the top of the page *(circled below)*.

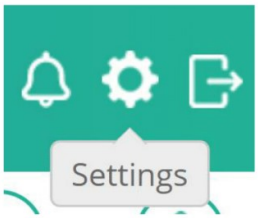


Misc.

Editing weight goal.

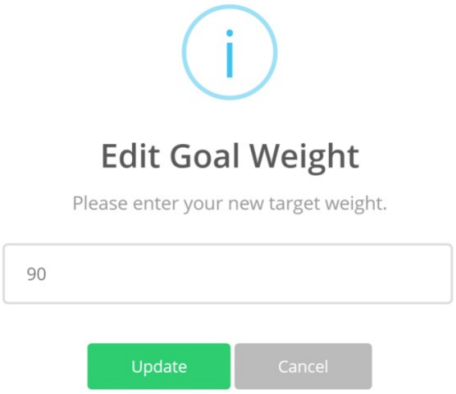
Changing your weight goal is simple.

1. Click on the settings icon at the top-right of the page.



2. Click on ***edit*** link for the Weight Goal option.

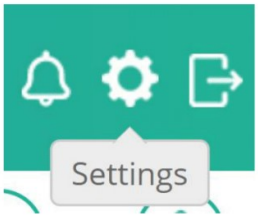
3. Enter your new weight goal.



Deleting your account

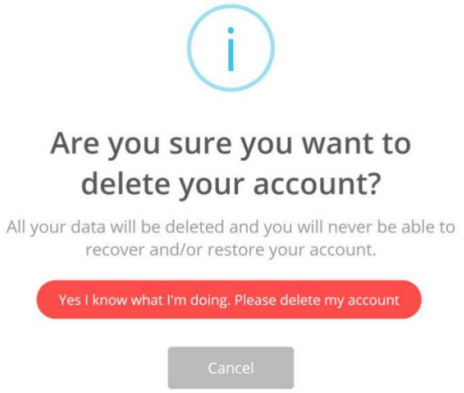
We would be sad to see you go, but if you do, there is an option to delete your account.

1. Click on the settings icon at the top-right of the page.



2. At the bottom of the settings page, you will see the delete option. Click on the delete button.

3. Confirm the deletion of your account.



4. Account has been successfully deleted.