Part 3 Interim Report

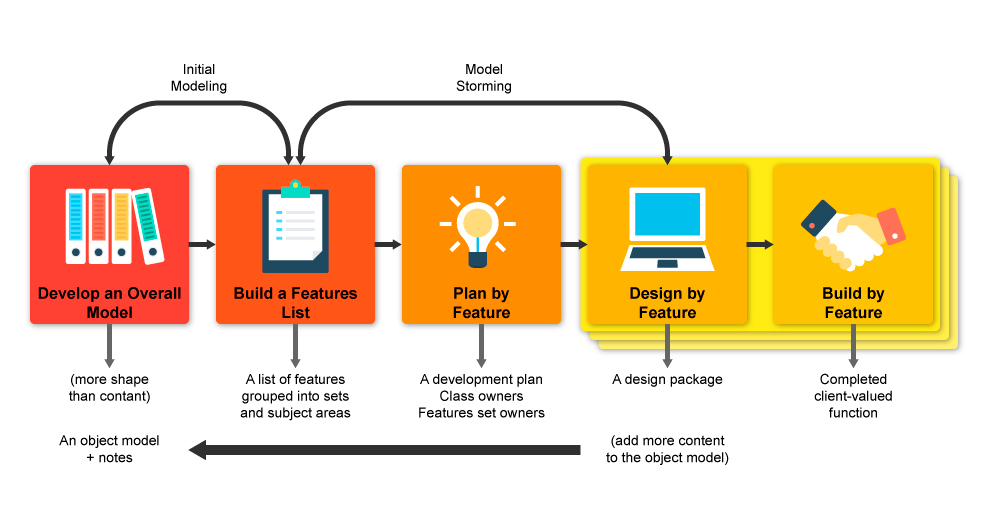
# 3. Prototype Design

## 3.1 Introduction

This chapter provides the insight into abstract layers of the application and the route of its development. An overview of how the technology researched in chapter two will be provided and analysed against the requirements of the applicable. This insight includes specifications sourced from the system’s architecture to the full stack development and the applicable chosen methodology.

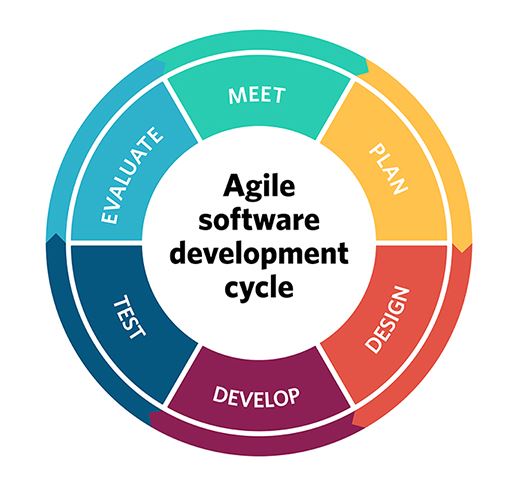
## 3.2. Software Methodology

**Feature Driven Development**

This methodology is one of the few adaptive methodologies which focuses on five short iterations and steps. Each iteration lasts approximately two weeks. The first three covered at the start are to “Develop an Overall Model”, “Build a Features List” and “Plan by Feature”. The last two covered at the end are to Build by Feature and Design by Feature.

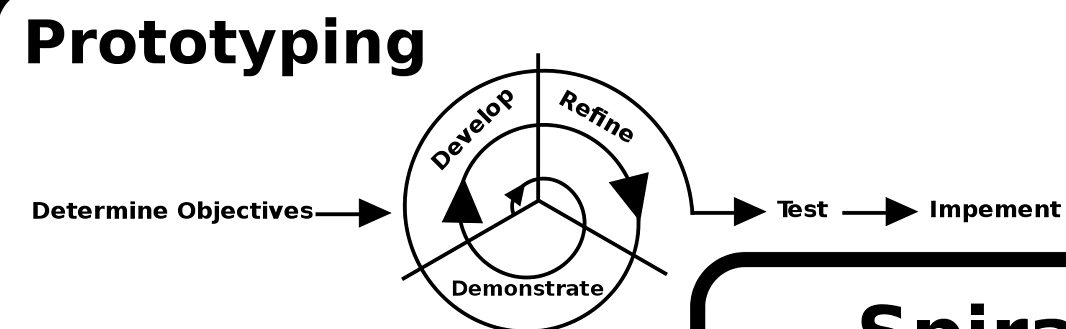
**Agile**

This methodology is another adaptive methodology being looked. Created as the alterative to the waterfall method, this involves both the potential users and the stakeholders in closer range for influence on the project itself. Key aspects of the project include ensuring the user is actively involved in the process, the team is enabled to make independent decisions, allow adaptable requirements in a fixed timeframe, enable the capturing of the requirements at the high level, iteration over small developed releases, frequent delivery of products, completion of the feature before moving on , applying the 80/20 rule, provide room for rigours integrated tests and clear comprehensive, collaborative & cooperative approach between all stakeholders.



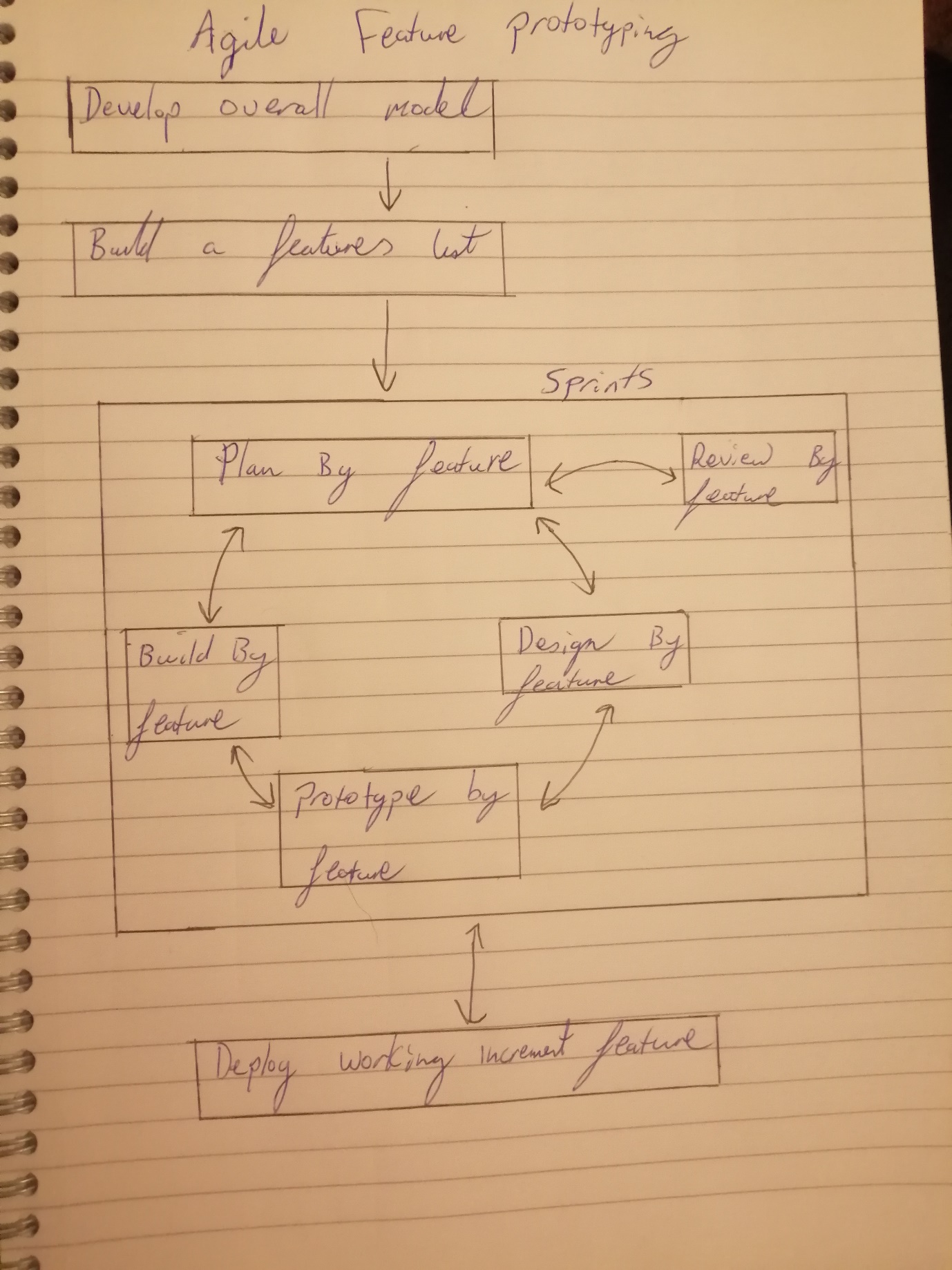
**Prototyping**

This methodology is used when attempting to create a solution to a proof of concept or a business requirement. It is normally adapted, developed, redefined and refined until the requirements have been gathered and understood to be correct. Once that happens, the final version will expand on the prototype in its implementation.



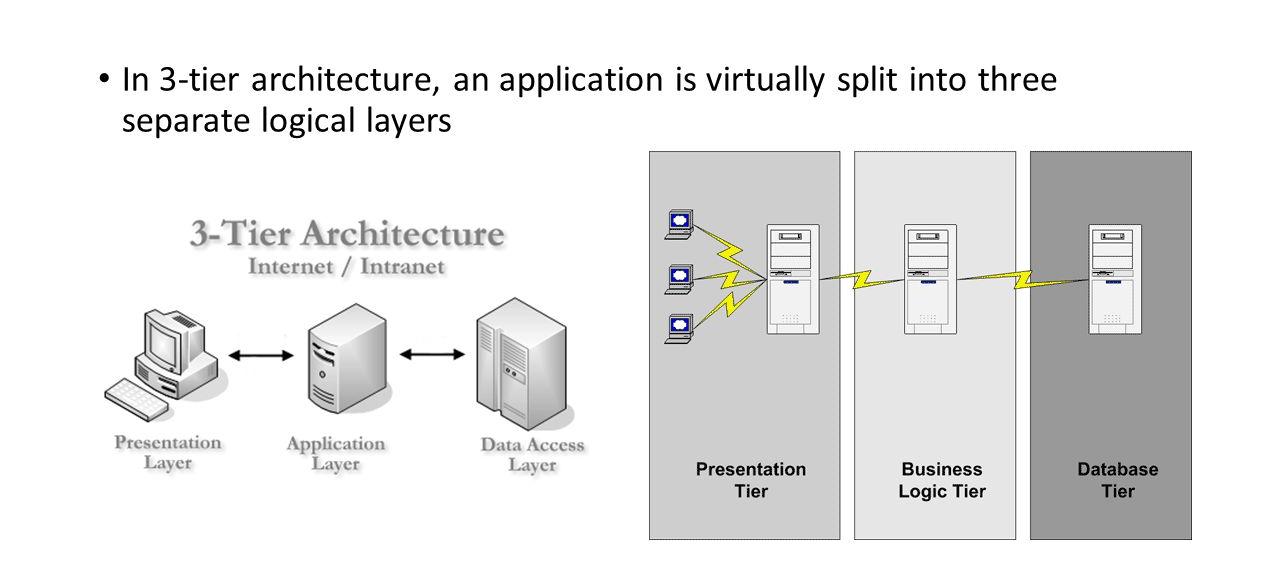
**Agile Feature Prototyping**

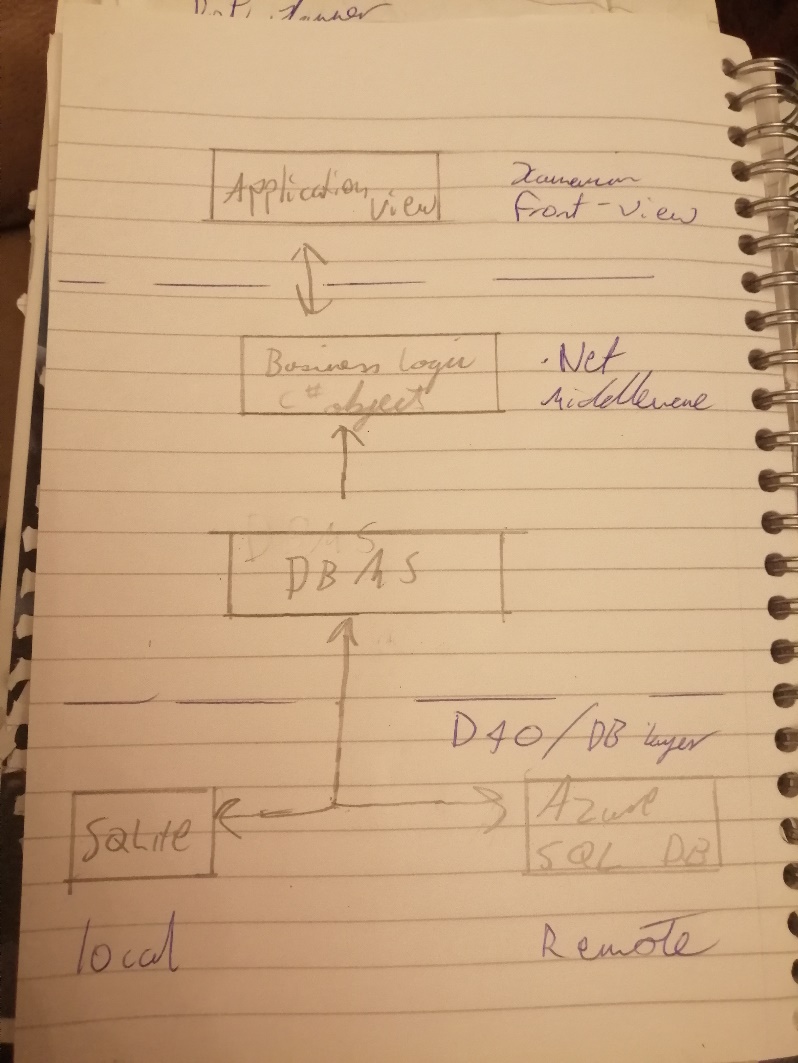
This is the methodology chosen and is created from each part of the researched methodology discussed. The methodology starts out like the Feature Driven Development methodology, where it captures the overall model and gathers the features as the first two steps at the start. Once completing that, it goes through iterations of sprints, like the agile methodology. Within each sprint, it would plan out each feature with the user for design purposes, which is followed by prototyping process and then returns to the planning for review. Since this is a circular motion, any part of the step can return to the source it came from for review, which is an aspect of agile during sprints. Since prototyping is included as this step, this is where prototyping methodology is used. The curricular can be ran through a few times before it is satisfied with the feature result before deploying it in increments. The next feature is worked on then.



## 3.3. Overview of System

The systems’ technical architecture is designed using a Three-Tier Client-Server Architecture. This is to ensure the logic of concerns are decoupled for minimal conflict between the three layers of software, known as the presentation layer, the application layer and the database layer. The presentation layer is managed by the client machine, the application layer is managed by the application server and the database layer is managed by the database server. The presentation layer can also be referenced as the front-view, the application layer as the middleware and the database layer as the backend. The previous mentioned terminologies can interchange. The methodology for building the application is the combination of FDD, Agile and Prototyping (as described previously)





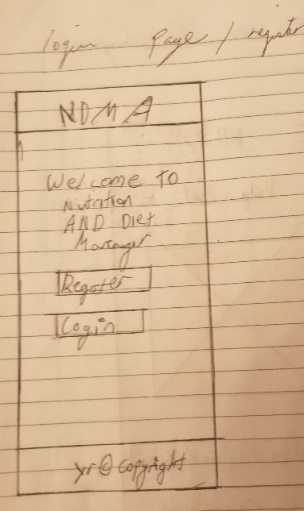
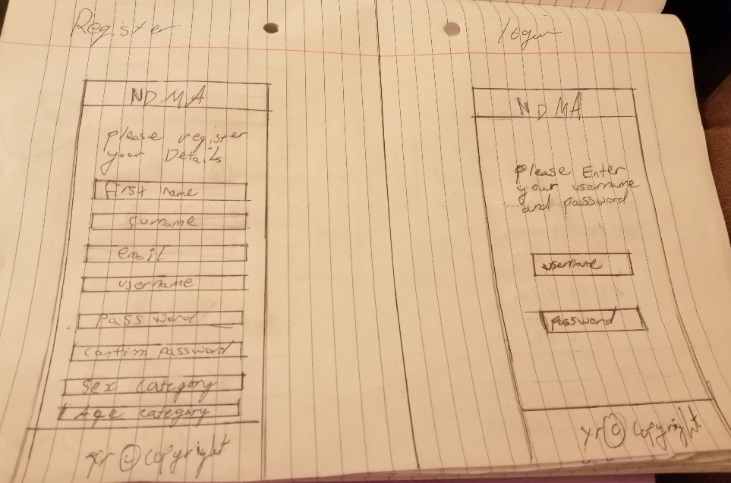
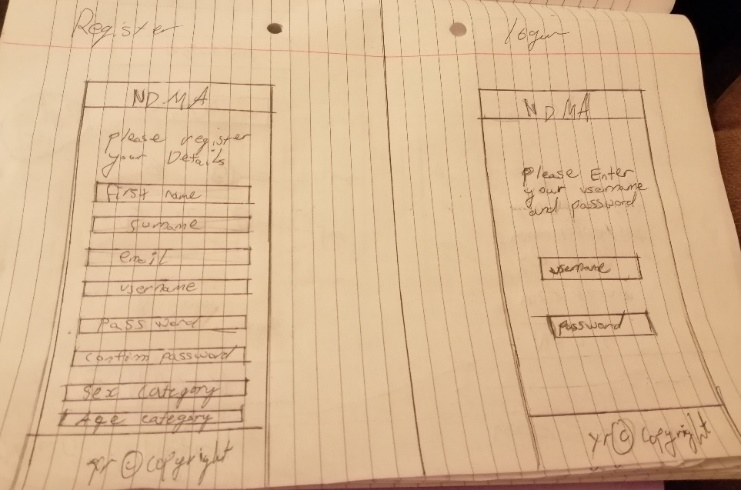
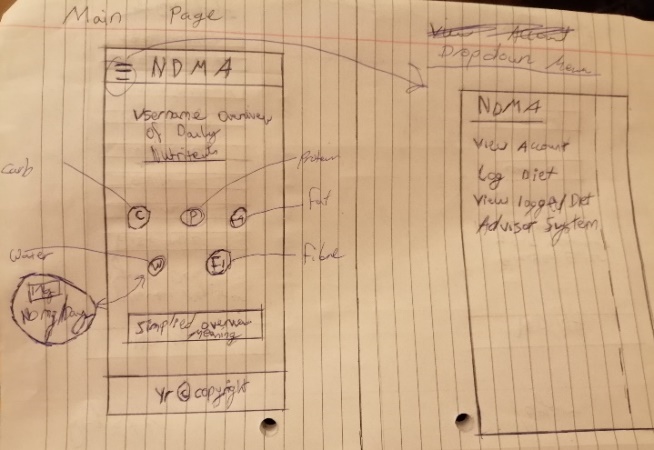
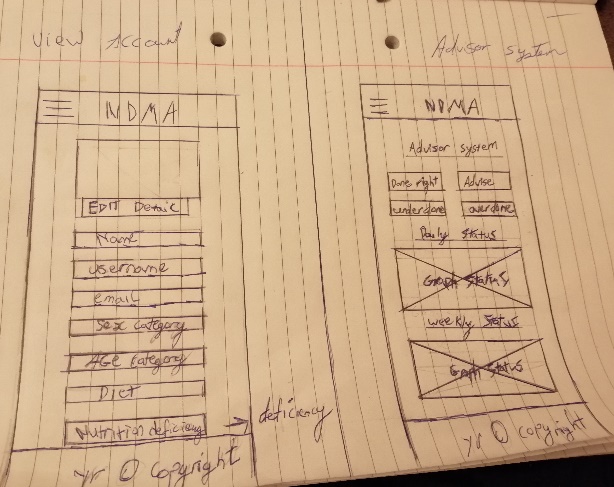
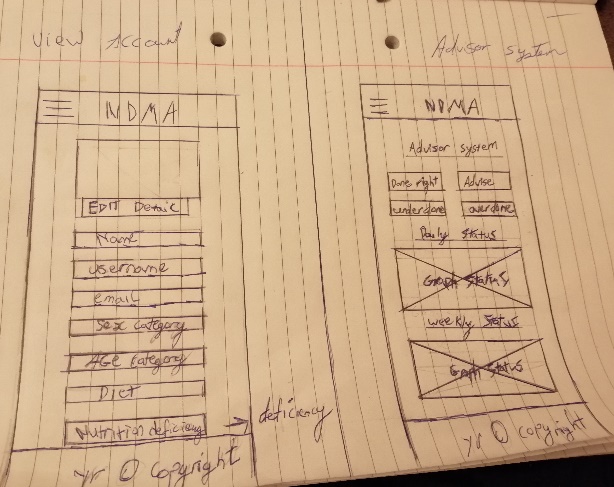
## 3.4. Front-End

The front-end aspect of the application is the presentation layer of the system architecture. It displays the information in which the application will look like to the user. This is critical to the system as the complexity of the application is geared towards the user experience, the user interface and design. The functionality of this layer must be polished to properly capture the input of the user in retrospect to their diet.

**Low Fide Prototype**

Paper prototypes were conjured as the presentation ideas were fleshed out. As part of the first iteration, these low-fide diagrams outlined the first look of the application.

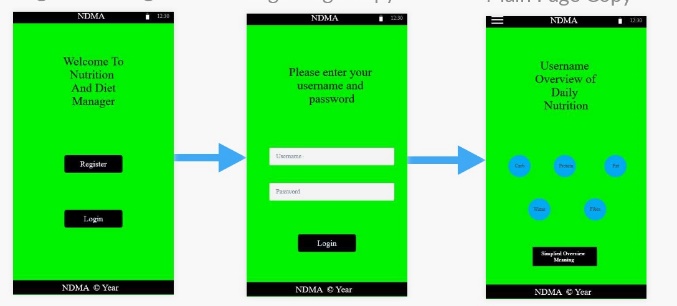
A look at some of the drafts

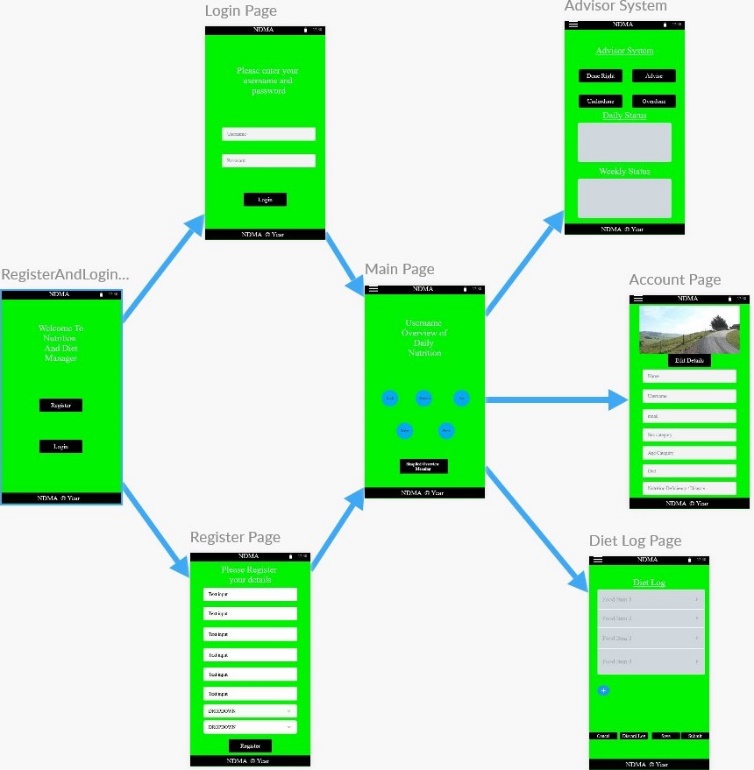
**Medium Fidelty Prototype**

Afterwards the layout of midium fidelty version of the prototype were created, both to display the newer look of the prototype with adding visual aesthic features to the application and the associtaced behaviour of the application. The prototypes and the associcted stroyboards were created from an online software called FluidUI.

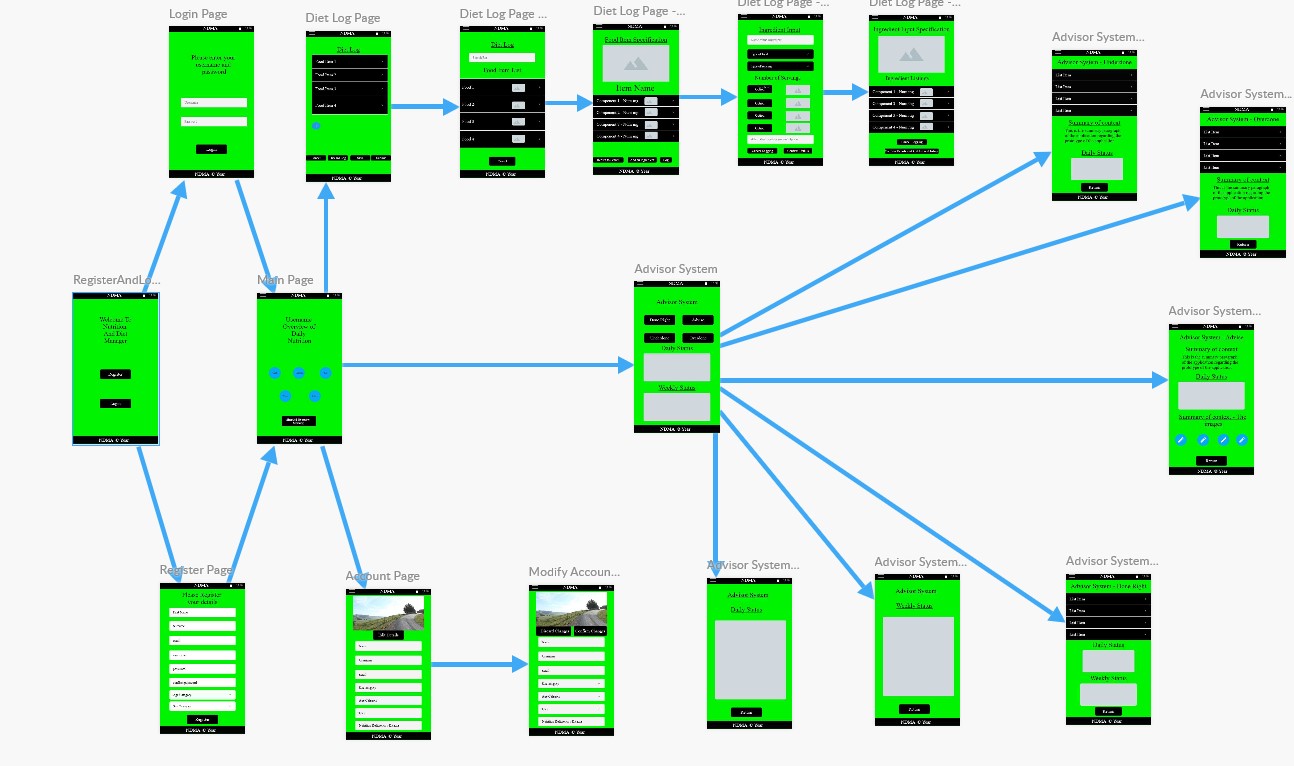
**First Storyboard Iteration**



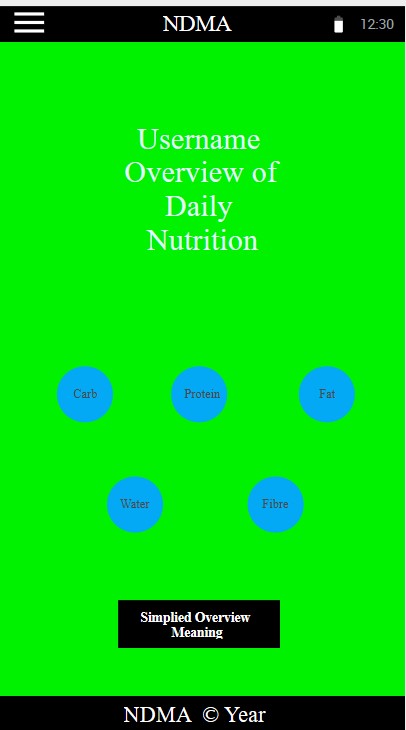
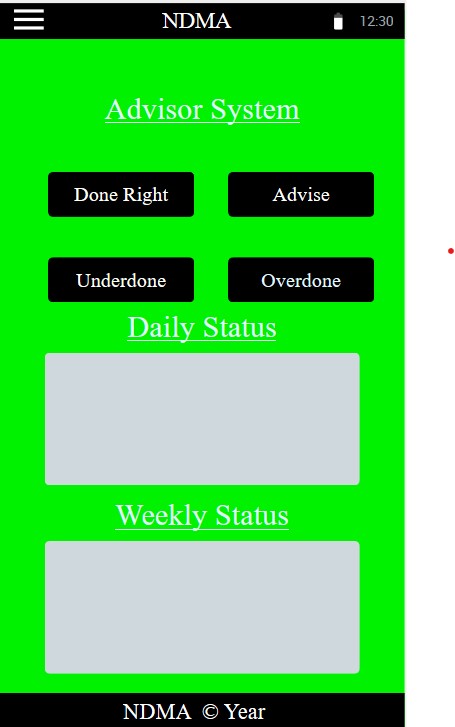
**Second Storyboard Iteration**



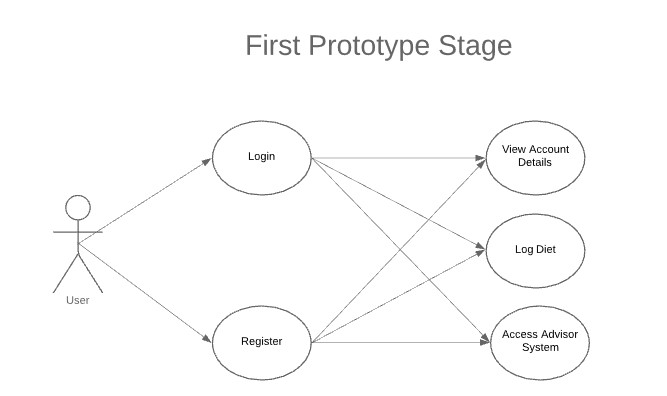
**Third Storyboard Iteration**

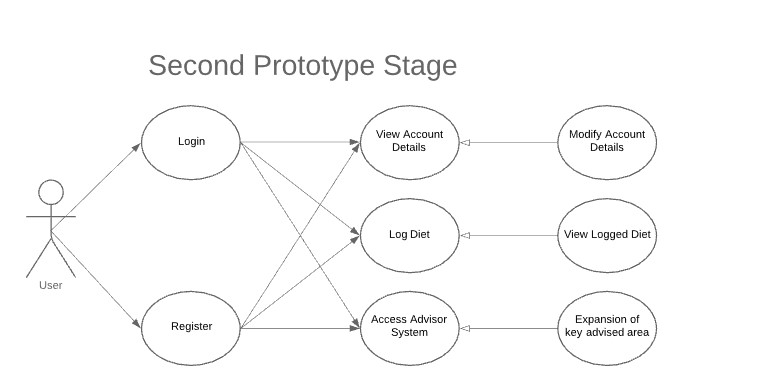


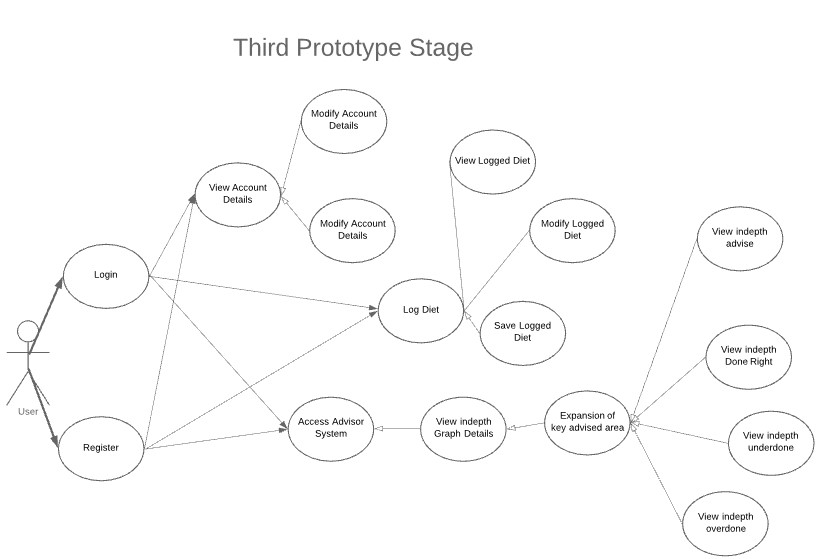
**Some of the Wireframe layout**

Use Case Diagrams were created alongside the applications prototypes and are used to represent the system behaviour with the user interaction. They also demonstrate both compulsory functionality and extendable ones for each part the user interacts with. This was drafted in three stages.







## 3.5. Middle-Tier

The Middle-Tier aspect, also known as the middleware and the application layer, is the logic aspect of the behaviour of the application. It controls the functionality of the application and communicates with the front-end or the backend as necessary. Because the system is a mobile application, the middle-tier provides the directions to get from one screen to the next. It also retrieves the request information from the database to provide to the user for overview, such as their logged diet schedule, or the advice from the system.

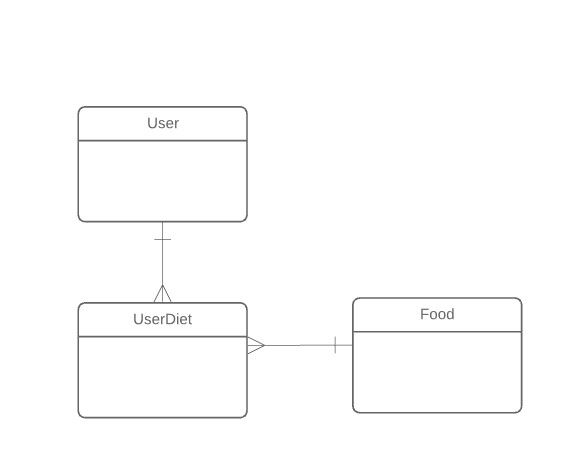
In the case of the designated system, the .Net Framework will be the key for the overview of the system.

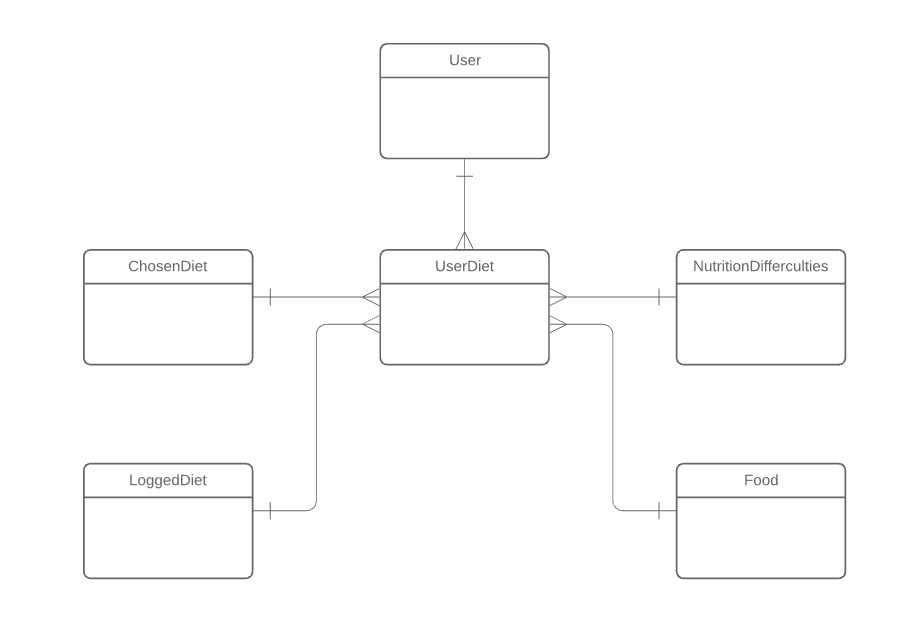
## 3.6. Back-End

The backend, also known as the Database layer, is the Data Access Route to all the storage within the system, such as the user credtials and their associated logged diets. This will be accessed from the Database managament system from the middleware section of the system. For local storage, SQLite will be used as the temperaroy storage system, in case the system cannot access the remote storage section. This would be handled by the Azure SQL database as the remote storage.

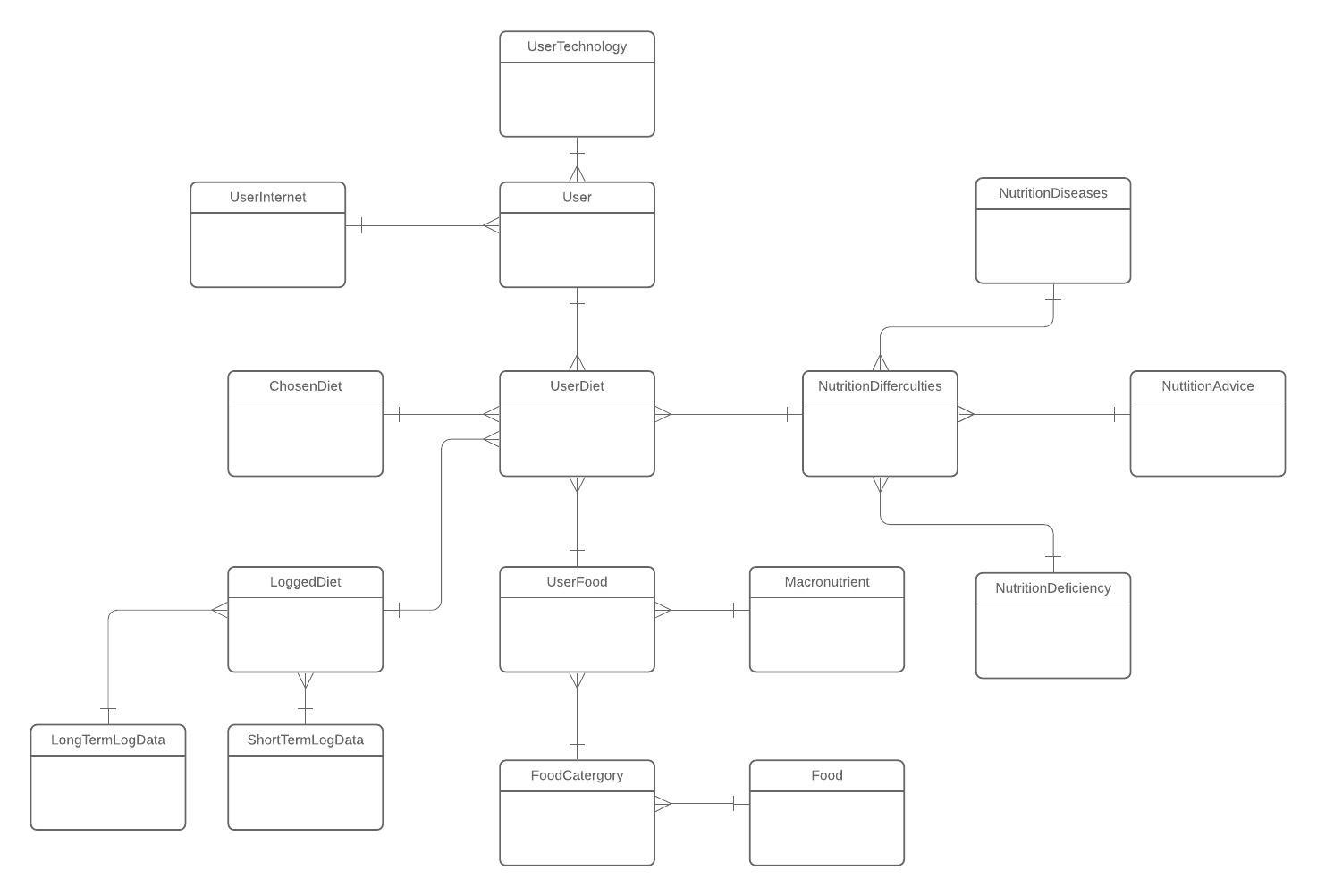
A few Entity Relational Diagrams have been complied below in different iterations to display the overview of the application backend behaviour.

**First Iteration**



**Second Iteration** 

**Third Iteration**



Maybe class diagrams

## 3.7. Conclusions

The overall system was analysed in this chapter in high level. This involves from the system architecture to the full stack development, encompass of the front-end, middleware and the backend, of the system and finally, the chosen methodology for the system. A lower level analysis will be covered in depth in the next chapter which will cover many of the same themes covered in this chapter. They will also cover the problems and potential changes encountered in the development process too.