# 3. Experiment Design

## 3.1 Introduction

This chapter provides the insight into the abstract layers of the application NDMA and the routes to its development. This includes an overview of the technology researched in chapter two will be used in NDMA’s development. The technology would be evaluated against the business and technical requirements drawn up previously. The overview includes specifications sourced from NDMAs architecture to the full stack development and the chosen methodology suited to NDMA’s needs.

Software design is the process of creating and defining certain parts of technical solution to a dedicated problem. For NDMA, this process required a number of steps. These steps were ensured to define and structure the hierarchy of the overall design of NDMA. The steps taken are outlined:

The problem domain of NDMA was looked at. The problem tackled was “How to assist people in making the correct dietary choices”. After the problem was identified, time was put to create, and define, the concepts of the initial technical design of NDMA. NDMA would be the solution to the problem described. This included some of the initial technical, feature and potential user requirements. This also would be the phase where the timeframe and project scope, for NDMA, is defined. After this design, a few different types of users were queried for some expected user requirements. They would have been made aware of the domain area NDMA was within as they were queried.

After the gathering of the user requirements has passed, a transcribe from user to feature requirement happened. This occurred to ensure there was a feasible way to develop each requirement by the user. It was also to ensure there was a mutual understanding, between what the user communicated, and what the developer believes is the best approach to develop such feature. After completing this, research into the technical requirements happened again. This is to ensure the feature requirements described by the user were enabled to be developed. While this was going on, an overview of all the feature requirements was also analysed. This was to understand which requirement can be completed in the given project scope.

After completing both the technical analysis, and the scope of the feature requirements against the project scope of NDMA, a few software methodologies were analysed. They were compared and contrasted to see which would best fit the requirements of NDMA. This analysis would incorporate the key dates to deliver viable results of NDMA. Each of the methodologies strengths and weaknesses were analysed against NDMA requirements. After choosing the suitable methodology, the development stage of NDMA began.

This is by looking at each of the feature requirement for the current scope of NDMA. A few versions of the prototypes were drafted up in the attempt to ensure the requirement has been met. These prototypes of NDMA were then reviewed and improved on. This was the attempt to ensure the user requirement has been properly met. Each part of the process would go through an iterative stage until their version is satisfactory. If part of an NDMA requirement couldn’t be reached, a acceptable version would be pushed through. It would be made notified of the reasons behind the version of NDMA and why this version over other ones.

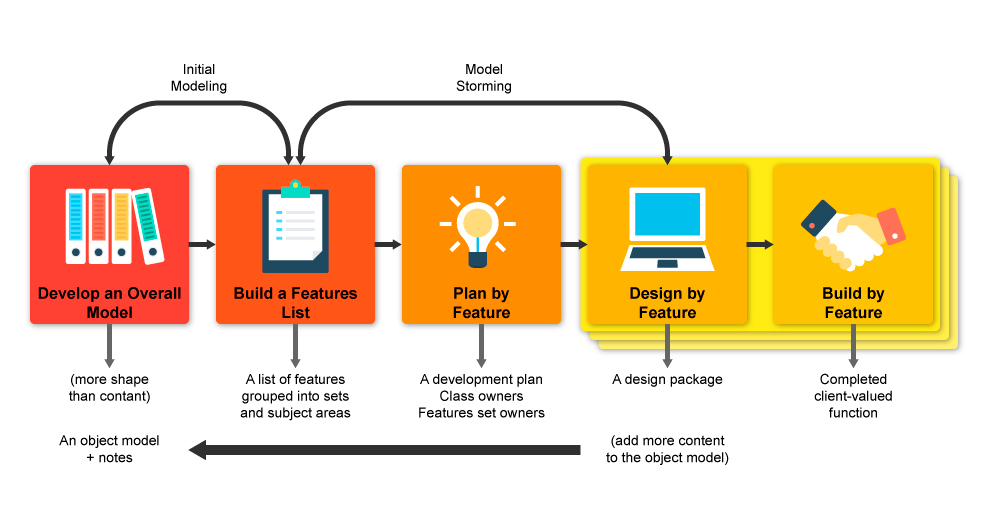
After each acceptable feature was developed, they would integrate into NDMA’s system. Once the scope has been concluded, the application would wrap up and be documented. These would be passed over in the key dates for review. All these processes would repeat until final scope has been reached. This would indicate the final acceptable version and the associated documented to be sent for review.

## 3.2. Software Design

**Software Methodology**

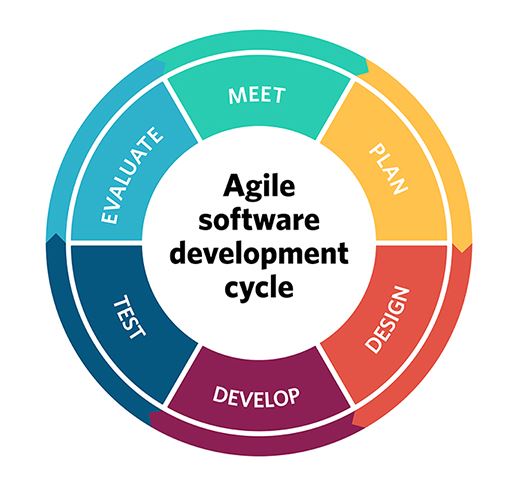
Software Methodology is the way of approach to a software design. Each approach would have defined steps and phases. This is to ensure the optimisation of the software. The approaches, which is used in industry, that were investigated for NDMA seem to have iterative type of process. The methodologies investigated are Feature Driven Development (FDD) Methodology, Prototyping Methodology and Agile Methodology. Like other methodologies, each of the mentioned have their different ways of handling the solution. If one methodology is derived from another, there would be some differences and some overlapping similarities. Other examples used in industry are the Waterfall Methodology and Test-Driven Development (TDD) Methodology. The suitability of the methodology is dependent on the design and technical needs of the project.

**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.

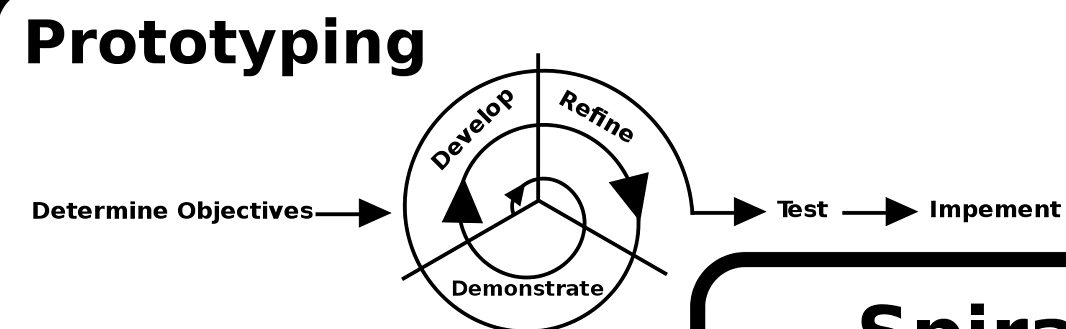
**Agile**

This methodology is another adaptive methodology being looked. Created as the alternative 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.



**Compare and contrast**

Each of the described methodology has its similarities and differences when compared. The similarities of Agile, Prototyping and FDD is the ability to develop swift, definable versions of software in a short space of time. It also easy to incorporate user feedback within each methodology. This is essential for NDMA. Part of each differences has mixed results for NDMA. For FDD, the features are continuously defined through the application lifecycle. This was similar in behaviour with NDMA. This is not part of Prototyping and Agile. For these methodologies, it is indicated all the requirements were known at the beginning. For each feature, the Prototyping would create a throwaway version initially. This is to get an understanding of the desired functionality of the feature. This is not part of the FDD methodology nor Agile and is a requirement of NDMA. Agile itself defines the number of tasks within each scope and the scope deadlines. This aspect of Agile is another requirement of NDMA. Because of the three requirements for NDMA was split across the methodologies, it was difficult to choose one. Protoyping was also the only one with defined terminologies for single developers. The other methodologies were defined for team projects.

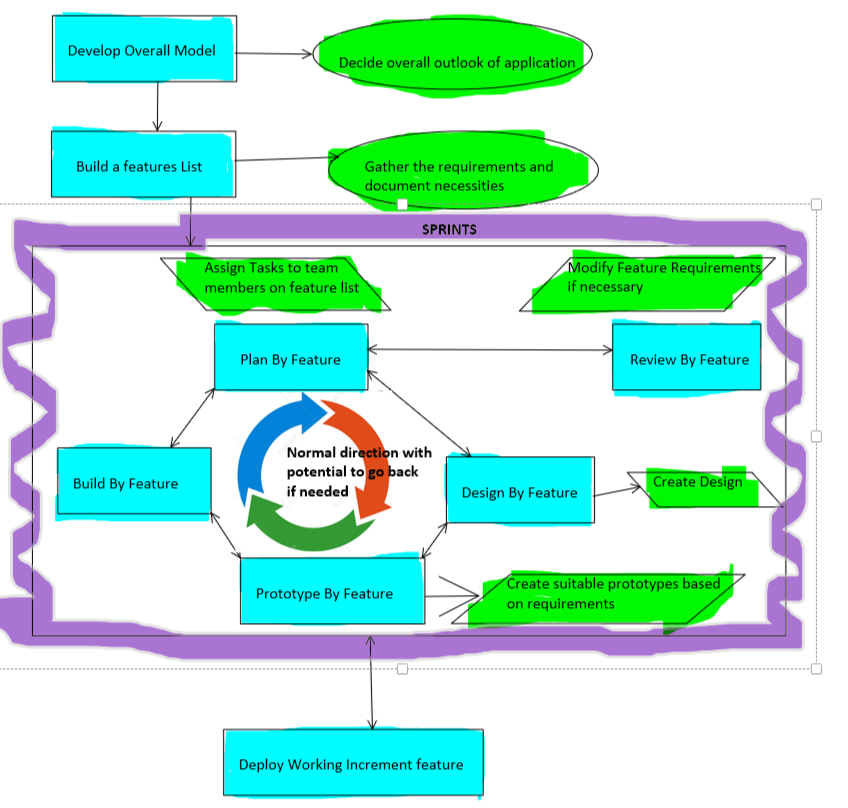
**Creating a new methodology**

After investigating each methodology, the decision was made to make a new methodology to meet NDMA requirements. Such methodology would be made from the different segments of the researched methodologies that were applicable to NDMA. For example, the prototyping aspect from the Prototyping Methodology was found to be necessary part of the project. The requirements of the application were both found at the start and during the interval of the application’s lifecycle. The start requirements were reviewed to see of the target could be reached. This approach was similar to the Agile Methodology. However, it was known at the start not all the requirements were known at the start. From this, it was to be expected some new requirements were to be discovered in the application lifecycle. This approach is like FDD. Gathering all this information, the three were combined to birth a new methodology. The name of the methodology created was Agile Feature Prototyping.

**Agile Feature Prototyping**

Agile Feature Prototyping was the chosen methodology was 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.

**Agile Feature Prototyping Diagram**

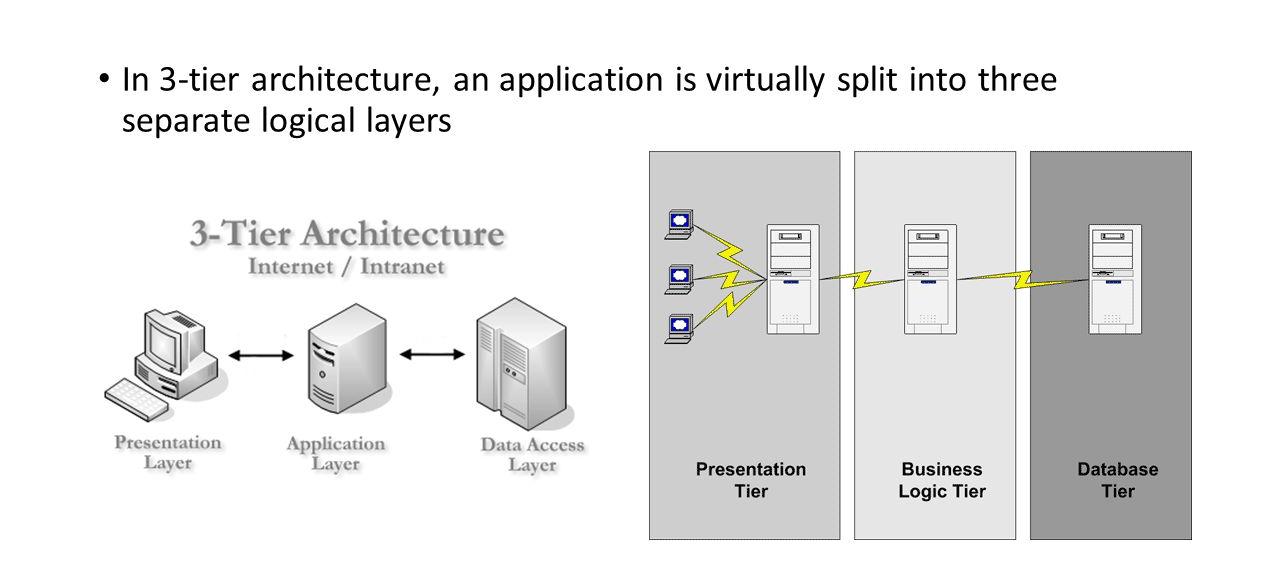


**Overview of System**

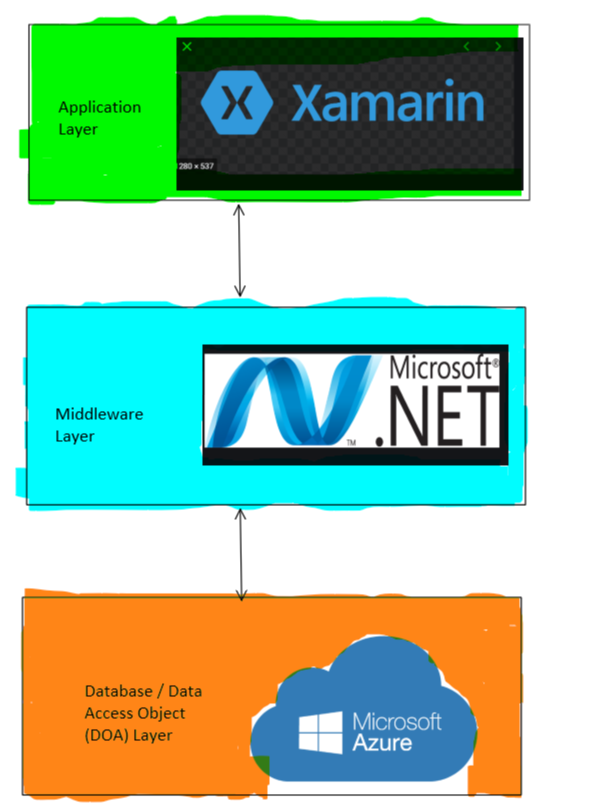
NDMA’s technical architecture is designed using a Three-Tier Client-Server Architecture. This is to ensure the logic of concerns are decoupled for minimal conflict between the three layers of software. The conflict, which could occur of the systems aren’t decoupled, would be one of two cases. The first is the overwriting of key files needed for the application to run efficiently. The second would cause a dependency within the layers themselves. If this occurred and one of the dependencies failed, the entire system would crash.

Each of NDMA tier layers are known as the presentation layer, the application layer and the database layer. The presentation layer is managed by the client machine and can be referenced as the front-view. The application layer is managed by the application server and can be referenced as the middleware. Finally, the database layer is managed by the database server. And can be referenced as the backend. Each terminology for each part of the application are interchangeable. The methodology for building the application is the Agile Feature Prototyping.

The mobile application UI of NDMA is the client side of the application. The Cloud database and the remote APIs, which NDMA connects to, are the server-side communication for the system.



My Version of the Three tier architecture



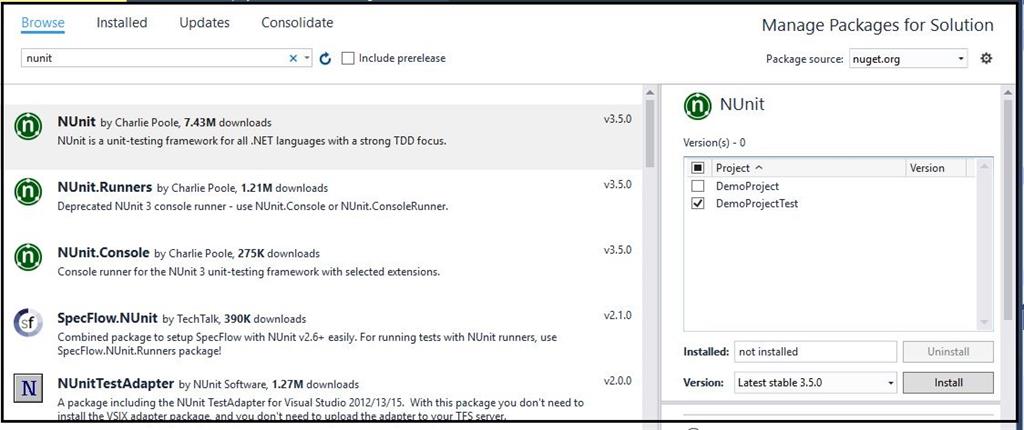
## 3.3. Software Test plan

The test plan for NDMA was to use automated and manual testing on the three different layers in each tier of the application. The system in its entirety will use Acceptance testing in the full stack development. Carrying out such tasks would ensure each layer of NDMA interact with one another correctly. There was the plan to make the state of NDMA architecture as close to the minimum operations standards. Tools, such as Benchmarks, Performance analysis or profiling tools, which are freely provided by Google, and other big tech companies, was used in conjunction with the debugging process of the application.

For the front-end of the application, an auto testing application was developed. The technologies used in this development was the frameworks of Selenium and NUnit. The Selenium framework allows the tests developed to automate tasks similar in behaviour to that of a user. The framework also allowed the testing system to be decoupled. This allowed testing of specific NDMA components without affecting other areas and layers. As the result of the mentioned capabilities, Selenium was chosen as the module testing.

NUnit is an open source Unit testing framework released under the MIT license for the .Net framework. Unit testing involves taking certain components of the application, test their performance against certain cases and assert them against expected outcome. This could be testing certain operations of the application and can use self-managed inputs as part of each case. NUnit, when combined with Selenium, was able to test each desired component of NDMA in automated session and assert their outcome.





For the middleware, NUnit testing was integrated. Black box, White box and Grey box testing was employed inside the middleware will be used here to ensure the performance and efficiency are optimised simultaneously with correct information. This is combined with ensuring the application can withstand faults, whether it is a technical issue or an injection attack from the user from the front-end of the application. Ensuring NDMA is de-coupled as necessary, so Subsystem testing will be incorporated. This is to ensure each application area only communicates in the way they are designed to do, combined with handling items they need to handle with only.

For database testing, DbFit was be used. This framework uses test-driven development as the methodology. It also encapsulates both unit testing and integration testing as part of its features. This allowed the test cases to capture specific information of the relational database. As a result, we were able to test the remote storage using the same test scenarios with different cases. This allowed test cases to be designed for availability and consistency of the remote application.

Manual testing was also captured. A set of instructions was provided to each person whom tested the application. This was to ensure the accuracy of each test case are as max as possible. The instructions were specific enough to direct the testers what is to be tested, yet vague enough to allow the tester free roam of how to test. This provided a in depth overview of NDMA internal behaviour and design.

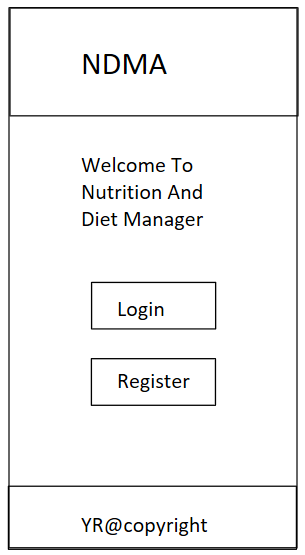
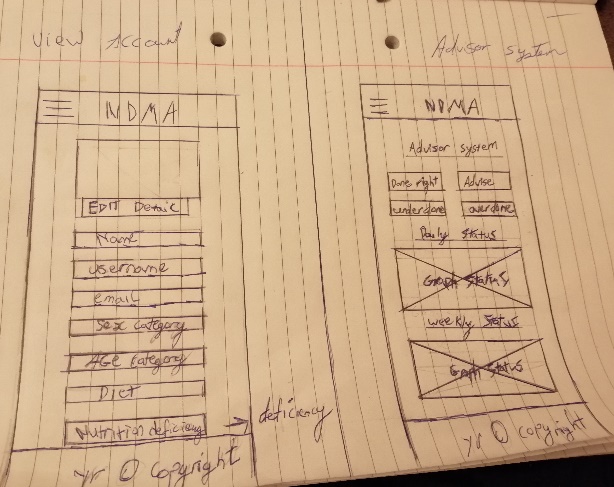
## 3.4. Front-End

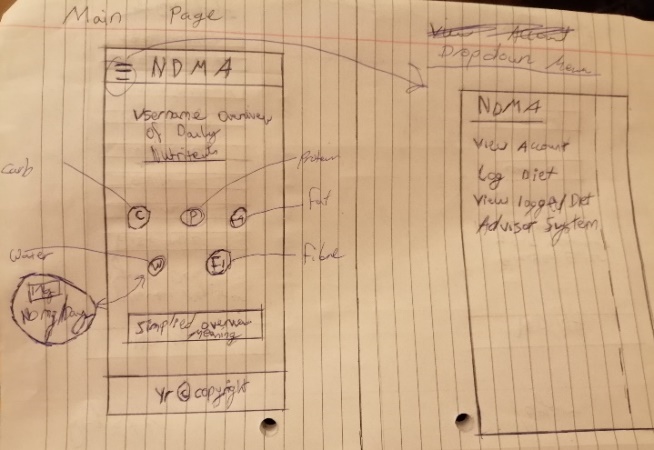
The front-end aspect of NDMA is the presentation layer of the system architecture. It displays the information in which the application will look like to the user. This is critical to NDMA. This is due to the complexity of the application being geared towards the user experience, user interface and design.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

**Low Fidelty Prototype – Paper prototypes**

Paper prototypes were conjured as the initial presentation ideas were fleshed out. These were the primary views of NDMA. As part of the first iteration, low-fide diagrams outlined the first look of the application. They would assist to capture the visualisation of the layout of the application in concept.

An insight into some of the drafts.

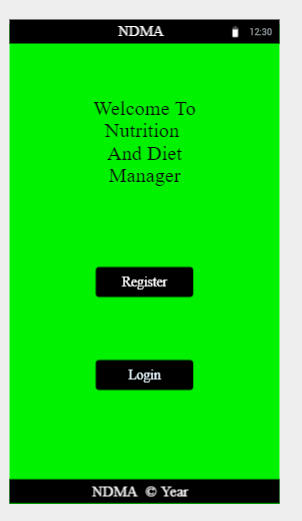
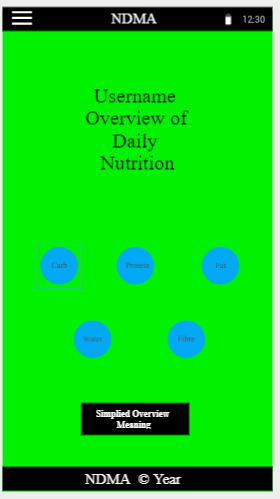
 



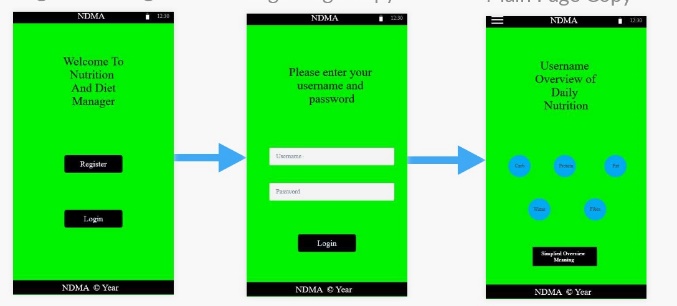
**Medium Fidelty Prototype - Wireframe**

After the paper prototypes were designed and reviewed, the next step was to create wireframes. This was considered the medium fidelty prototype of NDMA. The first purpose of the wireframes was to display the newer visualy aesthic looks of the prototype. The wireframes were combined afterwards into a storyboard. This was to display the aesthic features of the NDMA and its’ associtaced behaviour. Both the storyboard and the wireframes were created from the online webiste software called FluidUI. The link for the software is: <https://www.fluidui.com/> .

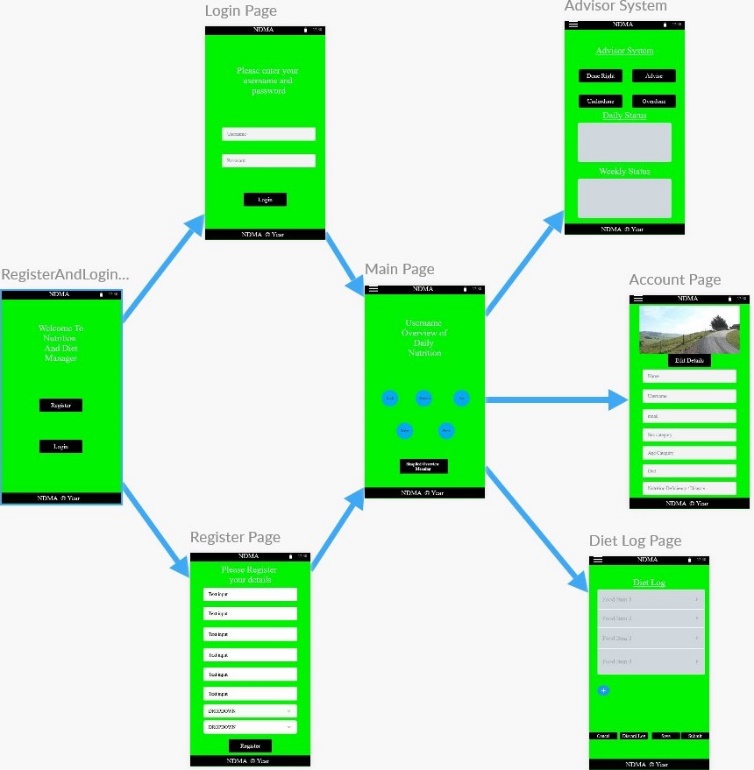
**Some of the Wireframe layout**

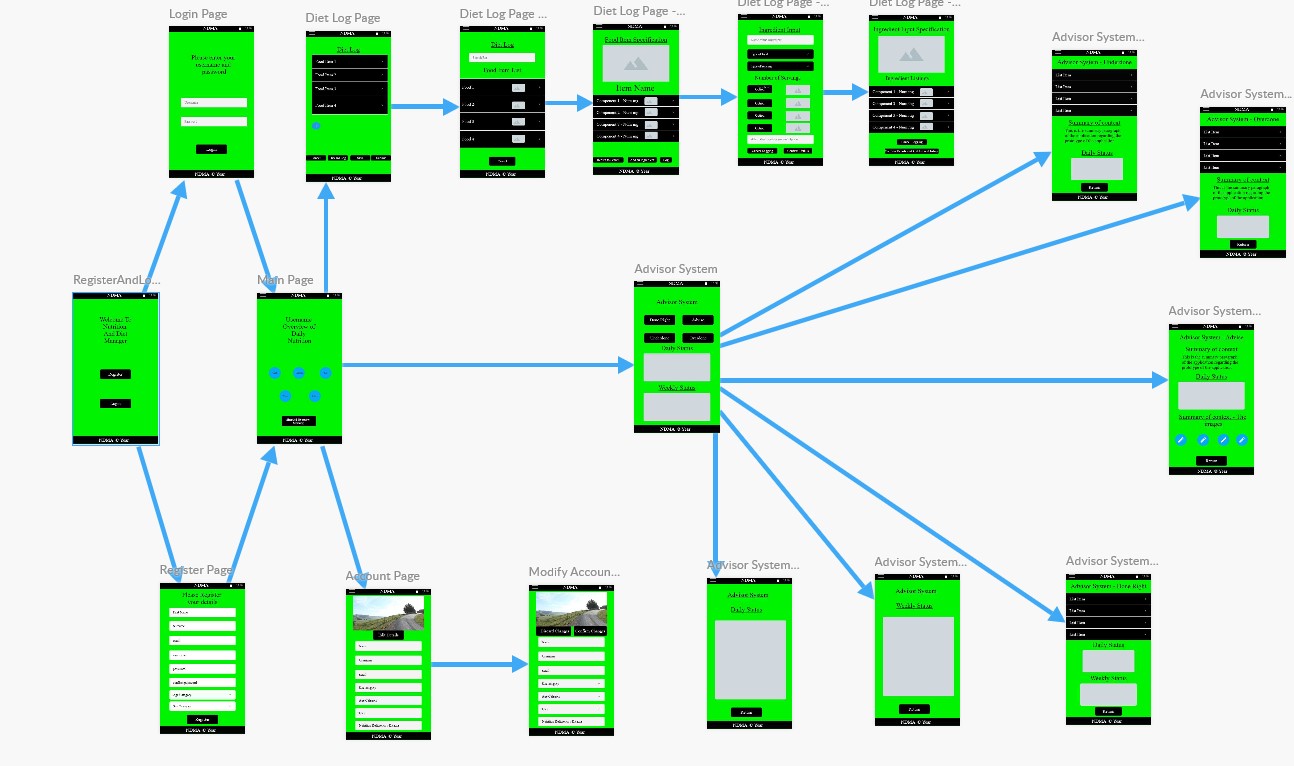
**First Storyboard Iteration – To display the login feature**



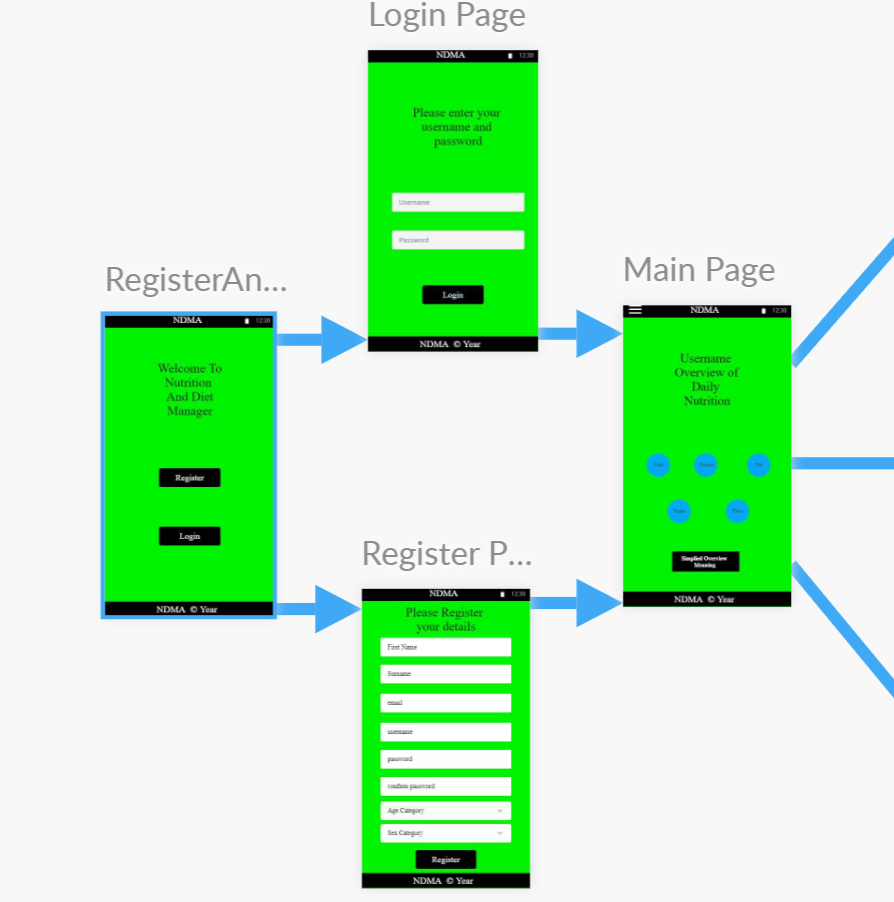
**Second Storyboard Iteration**



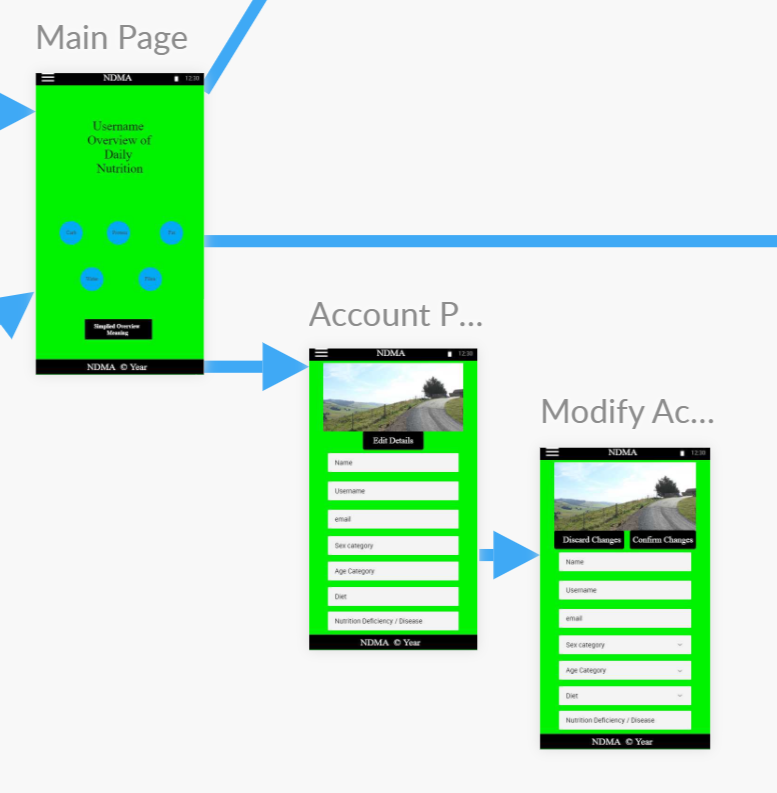
**Third Storyboard Iteration**



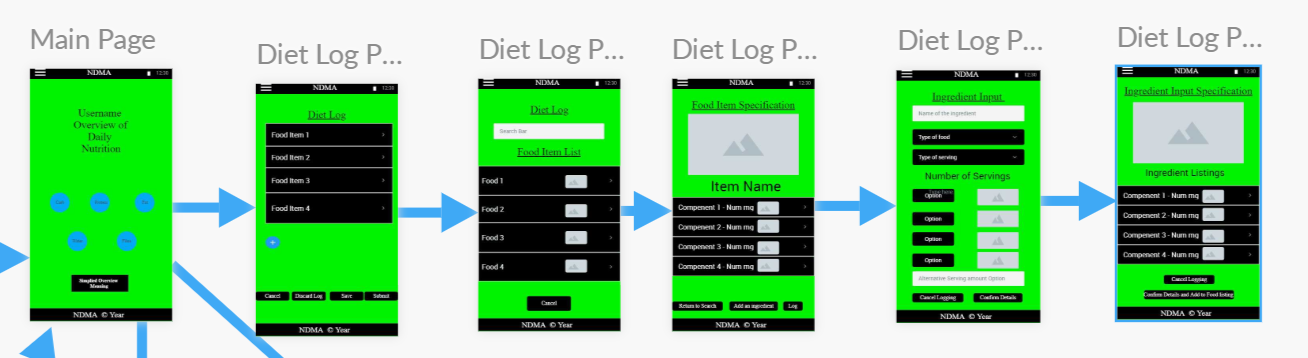
**Storyboard to display the login and register feature of NDMA, which directs to home page (first functionality breakdown)**

****

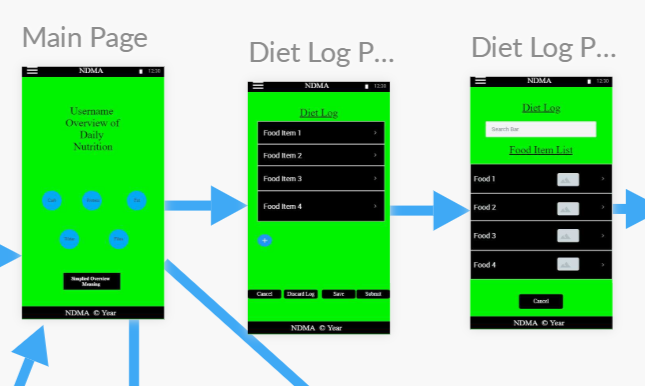
**Storyboard to display the user getting access to their user details functionality (second breakdown)**

****

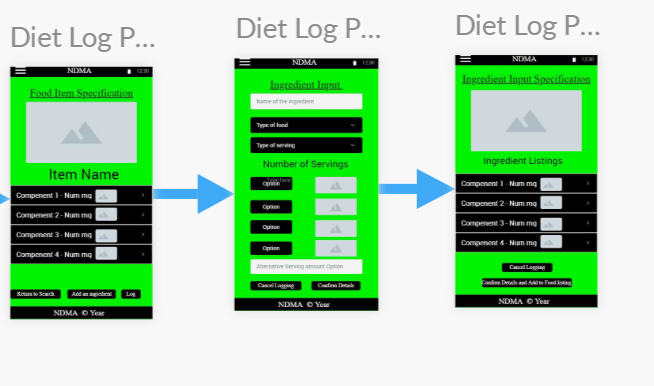
**Storyboard to display the logging system of NDMA (Third breakdown)**



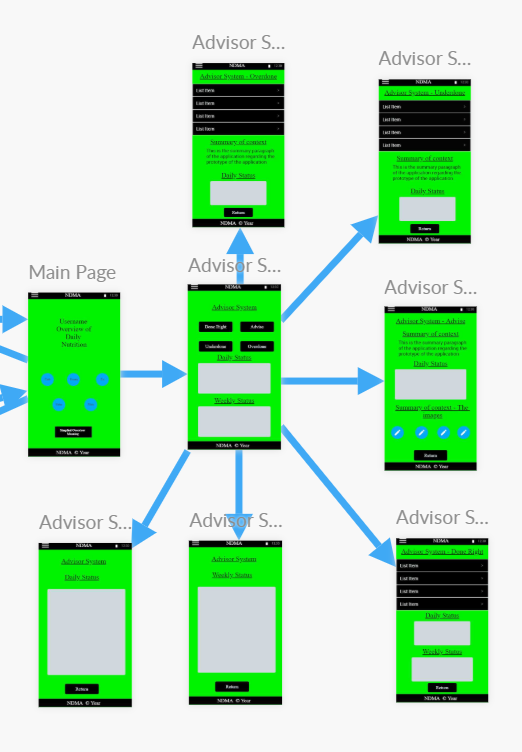
**A)**



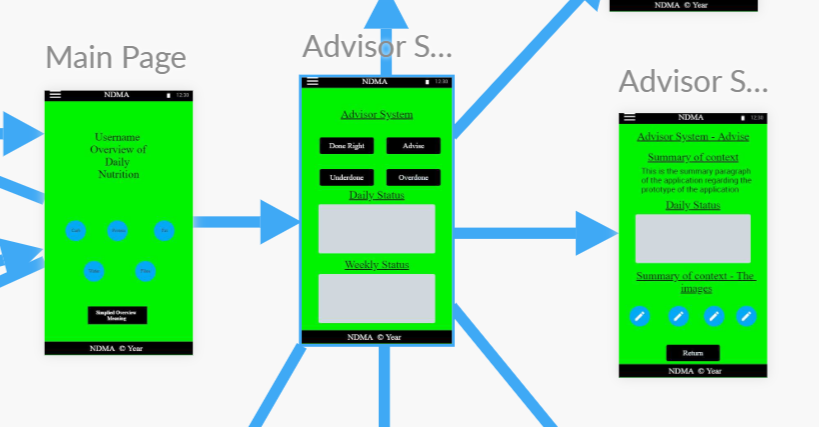
**B)**



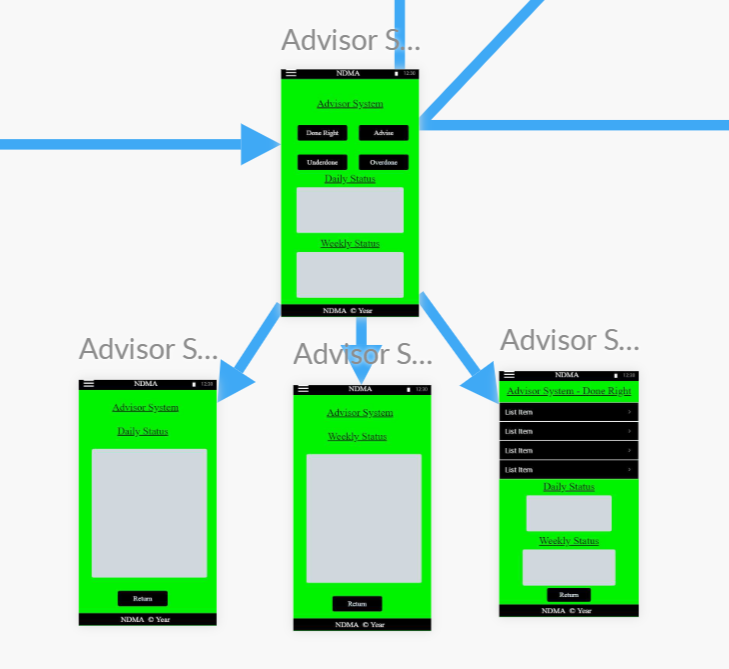
**Storyboard to display the advisor system of NDMA (Fourth breakdown)**



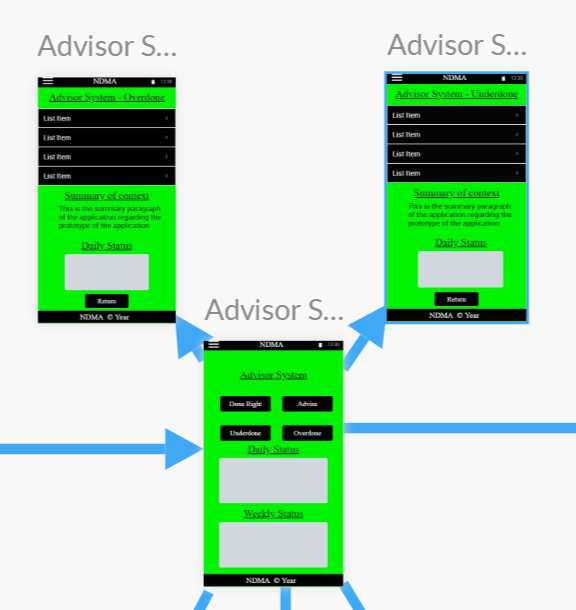
**A) To display access to the advisor system from home page, also display the overall advise page of the advisor syste,**



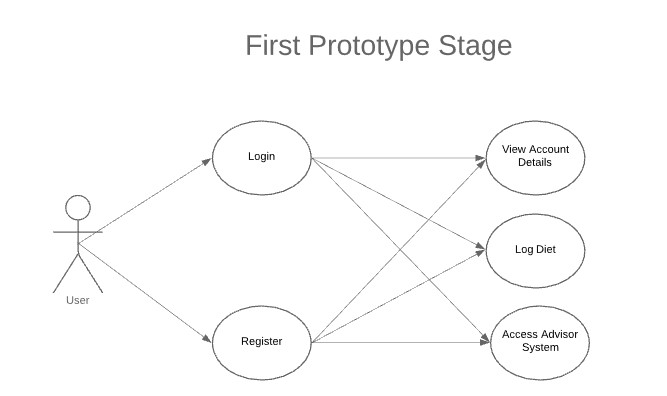
**B) To display the Daily and weekly trends of the logged data, also display the done right consumed data**

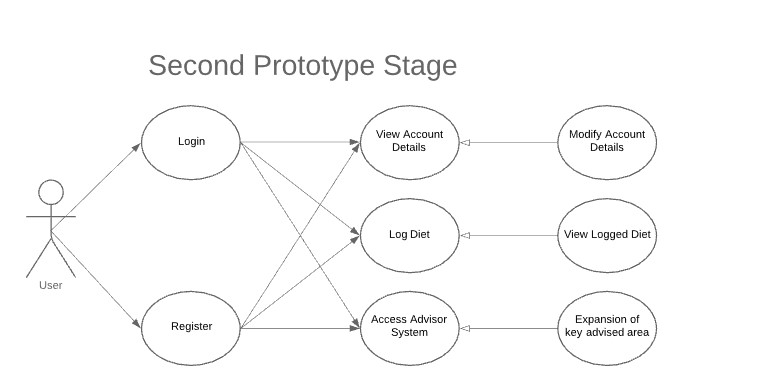


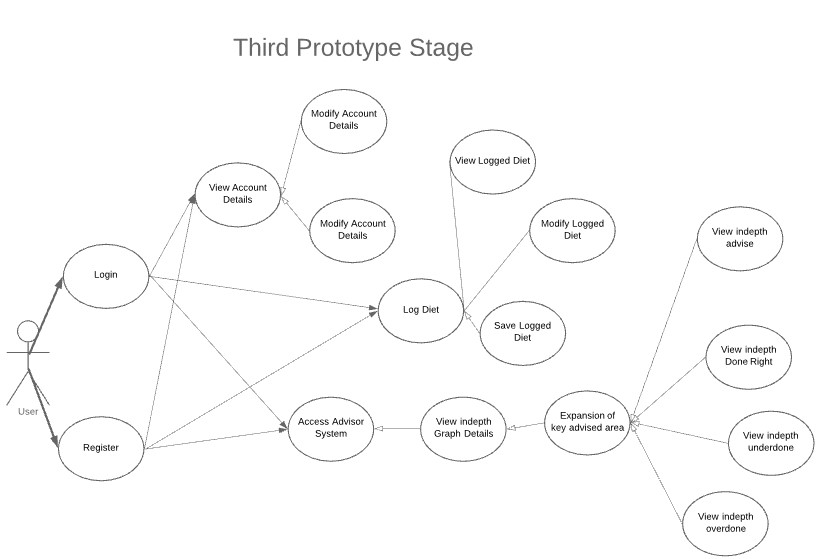
**C) To display the functionality of what food what been overconsumed and underconsumed**



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 NDMA. 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. Example are either the users’ logged diet or advice from the system.

In the case of the NDMA, 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 screenshot of a cell phone

Description automatically generated

## 3.7. Conclusions

The overall system of NDMA was analysed in this chapter at a high level. This included the system architecture structure. The full stack development of the system was discussed too. NDMA overall architecture encompass of a front-end, middleware and the backend design were laid out. Lastly, the chosen methodology for the system was expressed. A lower level analysis will be covered in depth in the next chapter, of which share the same themes. They will also cover the problems and potential changes encountered in the development process too.