I have been asked to develop a software system for the Sport Industry. The Client has requested that the application be able to access workouts uploaded by the company, track their diets and ask the trainers questions.

From these vague specifications, I began to develop a number of User Stories to represent all the software requirements I would need to effectively and efficiently develop this web application.

**User Stories:**

User Stories are crucial in software design as it gives developers a roadmap to what they have to code, this saves a large amount of time in development. It is this saved time that can be used later on to ensure testing reveals as many bugs within the software as possible. This section provides a number of user stories, viewed from the perspective of a user to derive features needed for the software.

1. As a user, I would like to view exercises posted by trainers so that I can understand the correct movements in a specific exercise.
2. As a user, I would like to log my workouts so that I can keep track of my progress.
3. As a user, I would like to have a profile that I can view or change to keep track of my progress.
4. As a user, I would like to be able to track my calories through a food diary so that I can view how many grams of carbohydrates, proteins, fats, etc.
5. As a user, I would like to be able to add my recipes/foods to be used in the food diary later.
6. As a user, I would like to be able to edit foods that best tailor what I have available to me.
7. As a user, I would like to be able to edit my daily caloric intake goal so that it best suits my height and weight and my overall goal (gain or lose weight).
8. As a user, I would like a forum that I can post questions on or reply to questions so that I can get help with a specific problem or offer help to anyone in need of it.
9. As a user, I would like to create my own workout routines using exercises posted by trainers.
10. As a user, I would like to participate in different workouts to discover different types of sport such as powerlifting, bodybuilding, running, etc.

These User Stories would be the basis for my design. From here I began to move onto translating these stories into formal descriptions/models that could be used by all project stakeholders and more importantly the developers and testers. These Models would include a full set of UML Diagrams: Use Case, Sequence Diagrams, Class Diagrams and Architecture Design. A mock UI Design will also be constructed to ensure full transparency across all development sectors of the software. Lastly, a simple instruction set of System Tests that can be used to validate the User requirements will be created to ensure bugs are minimal and that the software will perform as desired.

**UML Diagrams:**

UML Diagrams are very important when it comes to designing software systems. They provide a clear direction for programmers to accomplish all requirements for a specific software and also provide clear context to what a software should be able to do.

**Use Case Diagram:**

When tackling the use case diagram, it was important for me to really use the user stories to fine tune exactly who would access what feature but also to allow ease of access to these users. While it is important to have features for the user, an overly large amount of them can seem daunting the new users. This is why I decided to strip down the software to its base components and features, as you can see below it’s clear how the actor (user) is able to access certain features.

A close up of text on a white background

Description automatically generated

I tried my upmost to ensure ease of access to each user, an example of this is along the food diary use case. When adding a food to the diary, we have options for either custom food whereby the user can input the nutritional value located on packaging or to make things easier, the user can simply scan the barcode which returns all nutritional information without spending a great deal of time fine tuning information.

**Sequence Diagram:**

Due to the extensive processes that happen throughout this software, I broke down the sequence diagrams into smaller, more accessible diagrams.

The following sequence diagram describes the process of adding food using the barcode scanner feature. A database of nutritional information taken from decoded barcodes would be required to complete this process. The database I am using is a MongoDB dump from openfoodfacts (<https://static.openfoodfacts.org/data/openfoodfacts-mongodbdump.tar.gz>). This dump contains over 1 millions products and provided ease of access, limiting the amount of work need to complete this feature.

**Barcode Scanner Sequence Diagram:**

**A close up of a map

Description automatically generated**

Food Information

Next I moved to create a sequence diagram for the forum option within the software. This option is designed to allow users to ask questions and view previously answered or unanswered questions.

The forum section is very important as it allows users to communicate with each other, share any issues plaguing them as well as receiving expert advice with regards to specific problems. For this forum to work, we must assume there is a database which we can use to store any questions put forward by users but also to have some policing with regard to questions. The administrator service would allow user submitted questions to be reviewed and checked for any inappropriate contents. From here the question will either be approved or not approved. If approved the question will be posted and saved to the database, if not approved the user will receive a notification indicating why it was not approved. It will then be deleted from the database.

**Discussion Forum Sequence Diagram:**

**A close up of a map

Description automatically generated**

**Class Diagrams:**

Class diagrams are an extremely useful way of showing the structure of a software through its classes, attributes and methods but also showing the relationship between all of these. The UML class diagram below showcases all the main processes which occur within the software.

This class diagram is based off the user itself. Everything cannot exist without a user to populate the information, with exception to *“Routines”* as the routines section remains public so others can also participate in the same routines if they so wish. Both *User* and *UserInfo* both contain basic information regarding the user, but they also allow the software to calculate specific caloric, macronutrient and basic nutrient goals in grams based on their overall goals, lose weight, gain weight or maintain weight. This simple system allows the software to quickly calculate exactly how many calories, carbohydrates, proteins, etc. a user should be consuming daily based on their goal, height and weight, but also their activity level. Their activity level allows the software to estimate the users BMR (Base Metabolic Rate) and it is indicated by an integer value:

1. Not Active (Sedentary Job with no daily exercise).
2. Lightly Active (Sedentary Job with 1-3 days a week exercise).
3. Active (Sedentary Job with 5+ days a week exercise).
4. Very Active (Active Job with 3-5 days a week exercise).

We use all these variables along with the *Harris-Benedict Equation* to calculate the user’s daily caloric needs and then spread the ratios of macronutrients to show how many carbohydrates, proteins, etc. the user requires daily.

**UML Class Diagram:**

A close up of text on a white background

Description automatically generated

The Food Diary system allows the user to add a meal to their day by using a barcode scanner to scan ingredients/whole meals to retrieve all their nutritional information. This food is then added to that meal for the day and the nutritional information of the user is update to account for the number of calories, carbohydrates, proteins, etc. consumed from that meal.

Lastly, I have implemented a workout routine system that will allow the user to create a customer workout based on their restrictions but also on their strengths. When creating a routine, the user can view the entire database of exercises.

Within each exercise we have the exercises name, a description of how to perform the exercise and also an indicator to show which muscle group that exercise hits. This will help the user craft workouts that hit the same muscle grouping for maximum muscle exhaustion or if they desire to perform a full body workout, then this will allow them to craft that workout ensuring all muscle groups are hit. This also has other applications, for example if a user had an injury to their shoulder then this would allow them to set and perform exercises that exclude that muscle group which will in turn reduce pain, risk of injury and also help to heal those muscle groups.

The forum system is very basic. It allows any user to post a question that can be answered by anyone at any time. It also allows the user to view all posts by other users and reply to them if so desired.

**Architecture Diagram:**

The main reasons we use Architecture Diagrams are to ease use collaboration, increase inter-developer communication but more importantly it gives the project a vision. This vision shows everyone involved the entire system to provide developers and project managers guidance to help reach their final goal of deploying the software to users.

**A close up of text on a white background

Description automatically generated**

This diagram shows all the packages within the software system as well as the dependencies of each. This Multi-Layered Application relies on communication with the server to retrieve data such as a list of food items, list of questions asked by users as well as the users themselves. This allows the user to be completely connected to the entire system over a simple connection.

Each layer of the system is represented by models which each have their own dependencies, and within each model we have all the packages that are active within the system. When looking at the server model we can see the Data Access package allows each user’s software to connect to any database that is required. For example, if the user wishes to add food to their current meal, they would scan the barcode located on their packet of food. This barcode information would then be queried to the Food Database and return all the nutritional information related to that product containing that barcode. This is just a simple example, but the same process carries across all the packages including Users, Workout Routines and Forum packages.

**User Interface Design:**

A bunch of different colors

Description automatically generated

UI (User Interface) Design is an important process when constructing software as it helps create fewer software issues, it can increase involvement of the user with the software, allow for a substantial increase in functionality perfection and lastly creating a bounded link between the brain of the customer and the software. It is important that the customer and software are linked as it will allow the user to seamlessly flow through the software. In a perfect world, the UI would create an almost symbiotic relationship between the user and the application so that the user would not even have to make a conscious decision to switch between screens or perform a certain activity. UI Design strives to make this concept a reality.

When first visiting the software, the user will have to login or register if they do not have an account. I tried to use minimal information from the user as the form takes up a large portion of the page. Due to the minimal information required, only one page is necessary for registration which saves redirect usage across other screens. Once the user has passed this section, they will be greeted by the Home Screen which features 4 list items, Food Diary, Workouts, Forum and Settings. I wanted the user to be able to access all portions of the software without them having to navigate between a large number of screens. This section is the hub for the entire software and therefore includes a simple search bar to navigate quickly to any screen in the application.

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

My UI design for this software is based off a personal application. Every user will have a similar base set up for their software, but some parts of the software will change for each user. For example, in the food diary section the macronutrient pie chart, daily calorie intake and meals will change as these are based off the user’s personal settings. Things like daily calorie intake will change based on height, weight, activity level as well as the user’s goal, but features such as the barcode scanner will remain the same as it is an integral part of the software.

A screenshot of a cell phone

Description automatically generated

The sport industry is vast and has a great number of niche areas. For years I have been involved in running, powerlifting, bodybuilding and many other off springs of the sport industry. Throughout the years, I have noticed how difficult it is to get good quality guides and programs so that helped me in designing the Workout section of the software. Any user can start any type of program, all of which are accessible from the Explore tab within the Workouts Screen. If the user prefers to create their own routines with custom information, they can do that the My Routines tab.

A screenshot of a cell phone

Description automatically generated

As I have previously mentioned, it can be difficult to find information on certain topics. The Forum section allows users to discuss topics related to them and to ask questions about issues they are having. The forum works similar to a messaging system whereby the Title of the question (topic) is displayed within a list of others that anyone can view. This screen also offers a redirect for a user to ask their own question that uses a minimalistic form fill-in section.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Lastly, the Settings Screen of the app is probably the most crucial of all as it allows the user to customize their experience to suit their personal preference. Simple operations such as changing units of measurement can be key to pulling the user into the software. It also allows the user to change their information which is key to a sports-oriented software. Values such as weight can change a user’s Daily Caloric Needs which is an essential feature of the software, another example would be the user’s goal. If they have changed their goal from losing weight to gaining weight, then this must be changed in the settings to re-calculate Daily Caloric Needs and Macronutrient Ratios to ensure the user can reach their goal.

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

**System Tests:**

System Tests are one of the most important processes when designing a software. It is crucial that as many bugs are found within the software as possible before release of said software. There are a number of different software faults such as:

* Algorithmic faults
* Syntax Faults
* Documentation faults
* Stress or overload faults
* Capacity and boundary faults
* Computation and precision faults
* Throughput or performance faults
* Recovery faults
* Timing or coordination faults
* Standards and Procedure Faults

While for small programs it is possible to ensure there are no bugs but unfortunately for bigger software systems this is a large task and takes a great deal of time. The most common faults are Algorithmic or Computation and Precision as shown by Hewlett Packard that suggest this is the case 50% of the time.

When outlining system test, first we must look at the user requirements. This process is streamlined by the user stories we created at the beginning of the project. The following User Requirements must be tested:

1. View exercises from professionals that give detailed information on the exercise.
2. Log workouts that affect daily progress.
3. Profile that can be viewed or changed at any time.
4. Track calories and macronutrients through a food diary.
5. Add recipes/foods to be used in the food diary.
6. Edit foods.
7. Edit Daily Caloric Intake goal.
8. Forum to post questions or reply to questions.
9. Create custom workout routines using exercises posted by trainers.
10. Participate in different workouts based on other sports.

**Server-Side Requirements:**

This section covers all requirements that require a server-side connection. When performing these actions, we must ensure no information is lost in the process. Before we begin testing specific database connections, we must always do Load Testing of the web server, so a measurable response is returned based on putting pressure onto the system.

*Load Testing Test Cases:*

* Default load settings.
* Creating a bassline using an expected load.
* Creating a peak using increased loads that may occur during rush times.
* Creating a stress load based on the maximum load the server can handle.
* Creating a spike test to ensure the server can handle a sudden high load spike.
* Simple smoke test to endure the test are correctly configured and the server is responding to requests.

Next we must look at verifying the user’s login credentials. These test cases must only be performed once a system is in place that stores encrypted passwords to the user database.

*Login Test Cases:*

* Invalid Email and Password.
* Invalid Email but valid Password.
* Valid Email but invalid Password.
* Valid Email and Password.

Lastly, we must test the database itself. This process includes structural testing of the database to minimize all possible bugs within the system.

*Database Test Cases:*

* Schema Testing to validate the mapping format of the table.
* Table – Column Testing to ensure mapping of the backend is compatible with mapping of the front end.
* Stored Procedure Testing to ensure standard coding conventions and exception/error handling were followed.

**Black Box Testing:**

Black box testing examines functionality within a software without going into its internal structures or workings. Black Box testing covers unit, integration and acceptance testing which will allow us to cover the majority of the software. This section will specifically cover all aspects of input within the software, then comparing the actual output of the test and expected output of the test to ensure everything is working correctly. The requirements covered in this section include: **Requirements 2, 3, 4, 5, 6, 7, 8 and 9**.

*Input/output Test Cases:*

* Daily Caloric Intake Goal/Profile – Age, Height, Weight, Activity Level:
  + Equivalence Partitioning whereby input data is divided into regions. Each region will be tested to ensure returned values coincide with expected values.
  + Combination Testing to ensure calculated data is correct from different combinations of input data.
* Adding/Removing/ Editing Foods:
  + Boundary Value Analysis to ensure food servings fall within boundary settings.
  + Combination Testing to test all possible combinations and how they will affect recalculated data based on newly added food.
* Forum:
  + Random Testing to ensure data is successfully outputted.
* Workouts:
  + Combination Testing to ensure any set of data can be joined within a new collection.
  + Boundary Value Analysis to ensure repetitions/sets of exercises are correctly partitioned.

**White Box Testing:**

White box testing is the opposite of black box testing whereby the internal structures and workings of the software are tested to ensure all paths are followed and that data flow is correct. Please note, for some of the following test cases, the server-side operations have already been tested. The requirements covered in this section include: **Requirements 1, 4, 9**.

*Test Cases:*

* Viewing Exercises:
  + Statement Testing to ensure query functions are operating correctly.
  + Branch Testing to ensure every path in these functions are tested thoroughly.
  + DU Pair Testing to ensure data flow is correct and optimal.
* Tracking Calories/Macronutrients:
  + Statement Testing to ensure calculation/ratio functions are operating correctly.
* Custom Workout:
  + Branch Testing to ensure every path of custom workout can be explored and tested.

From these extensive test cases, the software has been successfully tested using Server-Side, Black Box and White Box tests.

This program’s aim is to allow the user to track their entire lifestyle within the app, everything from the meals they eat to the exercises they perform but also I wanted to ensure that even fitness beginners could access all of the content and have each exercise, food, etc. be explained thoroughly.

Throughout this report, I have outlined everything that would be required to develop the software based on the specification and set of system requirements. Beginning with the use case diagram, this paved the way for the rest of the design process as everything can be related back to these cases. The user requirements are derived from this list also which are key for the design of the classes, user interface as well as system tests.

The sequence diagrams show key processes when in contact with the databases as well as within the features of the software. This is further expanded through the Class Diagrams and Architectural Design. The User Interface is a key part of any software as this is where the user will always be. As I mentioned previously, my aim was to try make the User Interface and the User’s relationship almost symbiotic to allow ease of use. Lastly, the System Tests helped finalize it by locating and ridding the software of as many bugs as possible.

Overall, I believe that through all the processes outlined above, I have effectively and efficiently designed a software that could be followed easily by developers, project managers and clients alike.