Palestine Polytechnic University

College of IT and Computer Engineering

Graduation Project

Healthy Body

Team Members

- Mohammad Abureesh
- Intisar Qafesha
- Qusay Khalayleh

Supervisor

- Ezdehar Jawabreh

Abstract

Many people like to take care of their health and want to lose weight or increase their weight so they used to search for the required information on the Internet or get it from social media and a lot of this information is incorrect and they may get confused or distracted from their goals because they take their information from many different untrusted resources. So we build a mobile application to give accurate and up-to-date nutrition information. The nutrition system allows users to communicate with the nutritionists to gain a diet plan, allows subscribers to track their progress in the diet plan and calculate calories for any food type the user requires, provides a map that shows restaurants that provide healthy food and sports clubs. These features may be helpful for those who are motivated to achieve different goals related to their health. The nutrition system offers a complete solution to nutrition and fitness tracking in an easy-to-use app. It's available for both IOS and Android platforms.

الملخص

يحب الكثير من الناس الاهتمام بصحتهم ويريدون إنقاص وزنهم أو زيادته وللقيام بذلك يلجأ الناس للطريقة الأسهل والأسرع والأوفر بالبحث عن المعلومات الصحية على الإنترنت أو من خلال وسائل التواصل الاجتماعي والكثير من هذه المعلومات في غالب الأحيان غير صحيحة أو غير دقيقة. وايضا قد يتشتتوا نتيجة لهذا الكم الهائل من المعلومات وبالتالي لا يستطيعوا تحقيق أهدافهم والسبب أخذهم لمعلوماتهم من العديد من المصادر المختلفة غير الموثوق بها. لذلك قمنا ببناء تطبيق جوال لإعطاء معلومات تتعلق بالتغذية الصحية دقيقة وموثوقة. يتيح نظام التغذية للمستخدمين التواصل مع خبراء التغذية للحصول على الحمية المناسبة لهم ،وايضاً يسمح للمشتركين بتتبع تقدمهم في النظام الغذائي أو الحمية المختارة لهم من قبل الخبراء، ويتيح النظام أيضاً حساب السعرات الحرارية لأي نوع من الأغذية الذي يحتاجه المستخدم، ويوفر كذلك خريطة توضح مواقع المطاعم التي تقدم طعامًا صحيًا والنوادي الرياضية أيضاً. قد تكون هذه الميزات مفيدة لأولئك الذين لديهم الرغبة في الحفاظ على صحتهم ولياقتهم البدنية ووزن مثالي. يقدم نظام التغذية نظام لتتبع الحميات الغذائية واللياقة البدنية في تطبيق سهل الاستخدام. متاح على منصات IOS و Android.

Chapter1

Project introduction

Table of Contents

1. **Introduction**
2. **Research problem**
3. **Overview**
4. **System importance**
5. **System objectives**
6. **Scope of the system**
7. **Actors**
8. **Similar systems**
9. **Summary**

1.1 Introduction

In this section, we will talk about the problem and discuss it, then make an overview of the system, and declare the system importance and what we can offer to the customer, then we will mention the system objectives, its scope with the target people, system actors, how it differs from other nutrition applications and summary at the end.

1.2 Research problem

The problem is that some people get confused between social media platforms or different apps to maintain their health or achieve health goals that they set for themselves and these platforms may be untrusted, the lack of interest or knowledge of calorie importance, and how to calculate them.

1.3 Overview

The project aims to build a nutrition system for smartphones and a website to control the system. We provided features that made it easier for users to maintain their health by getting nutritionists and athletic trainers to offer their services to users, enable them to calculate their daily caloric needs, inform them about the calories of different foods, schedule their food routine, and make them aware of the locations for restaurants that provide healthy food, sports clubs.

1.4 System importance

- Facilitates daily routine scheduling process.
- Easy access to healthy restaurants and gyms.
- Facilitate the process of viewing the calories of multiple types of foods.
- The ability to calculate the daily calories needed by the user.
- Calculate calories for any food type the user enters in the system.
- Facilitates the communication process between nutritionists and users.

1.5 System objectives

- Nutritionists are able to track customers' (subscribers) progress.
- Subscribers are able to track their progress in fat loss.
- Adding maps in the application to locate restaurants that provide healthy food and locate sports clubs.
- Schedule a daily, weekly, or monthly routine of diet that users choose.
- Adding a calorie calculator to calculate the daily calorie requirement based on weight, height, and intensity of weekly activity.

- Calculate calories for any food type the user enters in the system.
- The ability to review the calorie guide for all types of foods.
- Organizing the communication process between the client and the nutritionists by adding the chat feature in the application.

1.6 Scope of the system

In the beginning, we seek to provide the application to all people in Hebron with all its features (Providing customers with gyms and healthy restaurants locations) and for the West Bank in general with most of its features and target nutritionists to work on the system interested in fitness and nutrition to use the application.

1.7 Actors

- Users(Subscribers)
- Nutritionists
- Athletic trainers
- Admin

1.8 Similar systems

- **myfitnesspal :**

  Advantage:

  1. analyze and provide you with a breakdown of each individual portion of food that you input into it: highlighting how much fat, protein, and carbohydrates are in them and then how many calories you've consumed.
  2. myfitnesspal provides the number of calories that you should be consuming each day.

Disadvantage:

1. Occasionally barcode scans don't work.
2. Needs to be updated often; updates can be inconsistent.

Difference: Getting advice from nutritionists and fitness trainers besides features like calculating your daily need for calories.

- **FatSecret**

Advantage:

1. The interface is simple and easy-to-use.

2. Regular challenges give you extra boost of motivation

Disadvantage:

1. App can occasionally freeze and slow your phone down.
2. Lack of nutritional information on occasion, especially when it comes to whole food.

Difference: FatSecret doesn't provide a map showing healthy food restaurants, sports clubs, and supermarkets that provide healthy ingredients.

Chapter2

Requirements Specification

Table of Contents

1. **Introduction**
2. **Project Requirements Definition**
3. **Functional Requirements**
4. **non-Functional Requirements**
5. **Distributing the functional requirements**
6. **System administrator requirements**
7. **Users requirements**
8. **Nutritionists requirements**
9. **Guest requirements**
10. **System requirements**
11. **Use Case Diagram**
12. **Use Case Templates**

2.1 Introduction

In this chapter, we mention the functional requirements, the non-functional requirements, analyze the use case chart, provide the details of each use case, and provide a class diagram for the system. Functional requirements will be divided among the system representatives who are administrator, users, nutritionists, gust, along with system requirements.

2.2 Project Requirements Definition

2.2.1 Functional Requirements

R1: Nutritionists are able to track customers' (subscribers) progress.

R2: Subscribers are able to track their progress of fat loss.

R3: Adding maps in the application to locate restaurants that provide healthy food and locate sports clubs.

R4: Schedule a daily, weekly, or monthly routine of diet that users choose.

R5: Adding a calorie calculator to calculate the daily calorie requirement based on weight, height, and intensity of weekly activity.

R6: Calculate calories for any food type the user enters in the system.

R7: The ability to review the calorie guide for all types of foods.

R8: Organizing the communication process between the client and the nutritionists by adding the chat feature in the application.

2.2.2 Non-functional Requirements

R1: The date format must be as follows: month.day.year.

R2: The system is easy to use. 4 hours max to get familiar with the system.

R3:All authentication tokens are saved on a local device for comparison and need user permission to gain access.

R4:All web pages should be able to load within three seconds.

2.2.3 Distributing the functional requirements in the system to the following roles

- System requirements
- Nutritionists requirements
- Administrator requirements
- Users requirements
- Guest requirements

2.2.3.1 System administrator requirements

1. Add types of food to the system
2. Adding a new location to the system
3. Adding new nutritionists
4. Update location data
5. Update types of food added to the system
6. Remove nutritionists from the system
7. Delete a location from the system map
8. Delete posts
9. Delete users
10. Show all nutritionists
11. Show all users
12. Approval of adding nutritionists to the system

2.2.3.2 Users requirements

1. Log in to the system
2. Explore maps for healthy restaurants and gyms.
3. Edit profile information
4. Delete account
5. Subscribe to nutritionists
6. Calculate body mass index(BMI).
7. Adding new feedback for nutritionists
8. Communicate with a nutritionist
9. Review the calorie guide for different foods
10. Track the progress against the set goal
11. View location data on a map
12. Browse the map
13. Calculate the calories needed for the daily needs
14. Calculate calories for a specific food

15. Explore posts by randomized nutrition experts
16. Browse posts for the followed nutritionists
17. Send feedback about the system
18. Sending a request to the system administrator to change the account type from user to nutritionist

2.2.3.3 Nutritionists requirements

1. Log in to the system
2. Edit profile information
3. View the calories for all foods registered in the system
4. Send feedback about the system
5. Create a diet plan for a specific subscriber
6. Track subscribers and review their information
7. Cancel a user subscription
8. Add a new post
9. Delete a post
10. Edit a post
11. Communicate with subscribers
12. Browse the map
13. View location information on a map
14. Explore posts by other nutritionists
15. Review all feedback

2.2.3.3 Guest requirements

1. Explore popular posts
2. Create user accoun

2.2.3.4 System requirements

1. Keep your login information
2. Send notifications
3. Calculate the daily water needs of the user.
4. Informs users if they have perfect weight or not.

2.3 Use Case Diagram

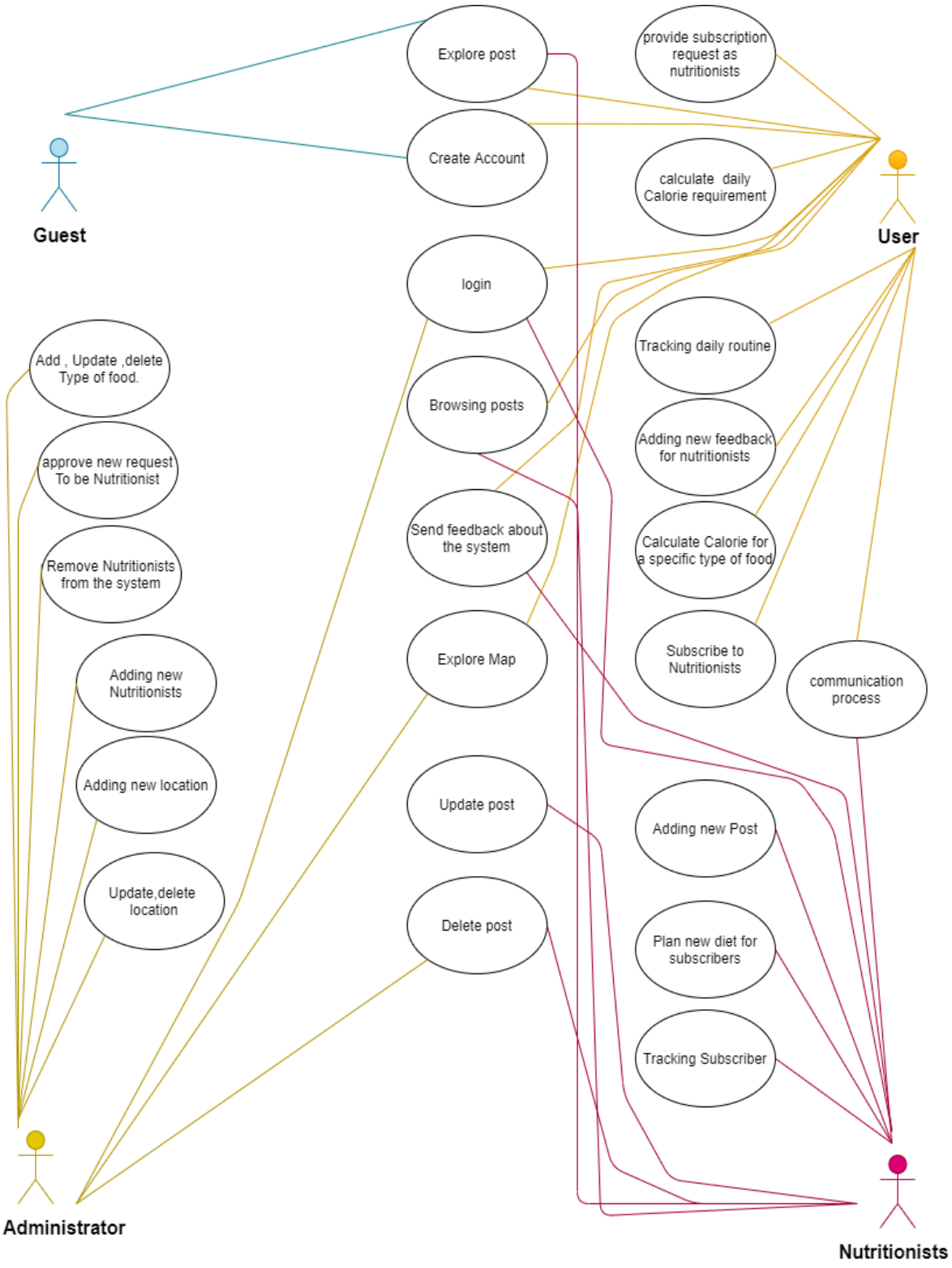


Figure2.1 use case diagram

2.3.1 Use Case Templates

Use case	Create Account
Actor	Guest.
Description	Describes how guests sign up for the nutrition registration system.
Inputs	Enter the username, email, password.
outputs	The system redirects the user to the user page user's home page.
Procedures	● Users enter the URL for the sign-up page. ● The system displays a sign-up page that asks users to enter a unique username, email, and password. ● The user submits their data to the sign-up system by clicking on the sign-up button. ● The sign-up system checks the uniqueness of the username and email. ● System stores the user's data in a database. ● The system redirects to the home page for each user. ● The system sends a verification email.
Exceptions	Username isn't unique. ● The system shows an error message. Email isn't unique. ● The system shows an error message.

Table 2.1 Use case diagrams to create an account**.**

Use case	Login
Actor	Administrator, Users, Nutritionists.
Description	Administrator, Users, Nutritionists logs in toe nutrition system to access the functionality of the system.
Inputs	Enter a valid username and password for the user's account.
outputs	The system redirects the user to the user's home page.
Procedures	● Users enter the login page. ● The system displays a login page that asks users to enter their username and password. ● The user submits a username and password to the login system by clicking on the login button. ● The login system checks the username and password from the database. ● The system displays the home page for each user.
Exceptions	Invalid username. ● The system shows an error message. Invalid password. ● The system shows an error message.

Table 2.2 use case template for login.

Use case	Explore posts

Actor	Guest, Users.
Description	Explore the posts that get the most popularity through a special page in which posts are displayed by some nutritionists who have a high rating, it is not possible to interact with these posts only when they are displayed.
Inputs	A list of posts rated high display from the system.
outputs	A box containing the name of the nutritionist and the recipe that he uploaded. If this recipe is available on a picture, it will be displayed with its description.
Procedures	● If the actor is only a guest, from the login screen they scroll down to explore the posts to get an overview of the app and its content. ● If he is a registered user in the system, he will go to the Explore screen by clicking on the Explore icon. Note: These procedures are illustrated in the screenshots in chapter 3
Exceptions	● The display of posts on this page is dependent on an internet connection. ● If the connection is not available, a message is displayed asking to connect to the Internet.

Table 2.3 use case template for creating account explore post.

Use case	Browsing posts
Actor	User, Nutritionists.
Description	Browse posts only for the actors mentioned above, this feature is the main page in the application in which the posts of the pages that the

	user follows are displayed and the posts that are uploaded by the nutritionist are displayed.
Inputs	The system brings a list of posts of nutritionists that the user follows and displays them by date from oldest to latest.
outputs	A list containing the posts that the system has brought and each post is displayed all its contents in addition to the possibility of interacting on the publication such as sending feedback to the nutritionist through the post or adding it to the favorites list.
Procedures	By pressing the Home button in the navbar, you go to the main screen to display a list of the posts provided by the system.
Exceptions	● If there is no internet connection, no new posts are displayed. ● If there are posts cached they will be displayed. ● Instability in communication leads to prolongation of the period required for displaying posts

Table2.4 use case template for browsing posts.

Use case	Send feedback about the system
Actor	User, Nutritionists.
Description	Sending a proposal to develop the system or any feedback to the system administrator
Inputs	Add text describing the problem with the ability to add an image (screenshot)
outputs	Sent to the system and all feedback is displayed in the list at the system administrator.

Procedures	● Log in to the system. ● Press the menu button in the corner. ● Choose from the menu "Feedback". ● A dialog box opens, titled Send Feedback. ● Enter the text (problem or suggestion). ● Adding an image (screenshot) is optional. ● Click on the Submit button.
Exceptions	● Pressing the Submit button without entering any content (text entry is required). ● No internet connection is available. ● Closing the dialog before pressing the Submit button.

Table 2.4 use case template to send feedback about the system.

Use case	Explore Map
Actor	Administrator, User, Nutritionists.
Description	View a map containing restaurants that serve healthy food and display sports clubs.
Inputs	
outputs	The system displays restaurants that serve healthy food and sports clubs on a map and details of each location.
Procedures	● Log in to the system. ● From the navbar choose the map icon. ● Moving to a new screen where the map is displayed.

Exceptions	● No internet connection. ● GPS is not enabled.

Table 2.5 use case template to explore the map.

Use case	Adding a new post
Actor	Nutritionists.
Description	Nutritionists can add posts.
Inputs	Nutritionists enter text and can upload images.
outputs	System display post into the user's home page.
Procedures	● Nutritionists log into the system page. ● The system authenticates nutritionists and starts the session. ● Nutritionists choose to add posts. ● Nutritionists add posts by entering text or uploading an image from the internet or their devices. ● The system acknowledges the post added.
Exceptions	The system fails to authenticate the nutritionists. ● the system informs the nutritionists and doesn't allow the nutritionists to proceed. The system fails to add posts because there is no internet connection. ● The system informs the nutritionists.

Table 2.6 use case template for adding new posts.

Use case	Updating post
Actor	Nutritionists.
Description	Nutritionists can update posts that are already added.
Inputs	Nutritionists can add text and upload new images
outputs	System display post into the user's home page.
Procedures	● Nutritionists log into the system page. ● The system authenticates nutritionists and starts the session. ● Nutritionists select edit on the post that they wanted to update. ● Nutritionists can delete or add text and upload new images or delete an image. ● Nutritionists click on the save button. ● The system updates the post.
Exceptions	The system fails to authenticate the nutritionists. ● the system informs the nutritionists and doesn't allow the nutritionists to proceed. The system fails to add posts because there is no internet connection. ● The system informs the nutritionists that can't update the post.

Tabel 2.7 use case template to update the posts.

Use case	Deleting post
Actor	Nutritionists.

Description	Nutritionists can delete posts that are already added.
Inputs	
outputs	The system deletes posts from the user's home page.
Procedures	● Nutritionists log into the system page. ● The system authenticates nutritionists and starts the session. ● Nutritionists select delete on the post that they wanted to delete. ● Nutritionists click on the delete button. ● The system deletes the post. ● The system disappears from the users' pages.
Exceptions	The system fails to authenticate the nutritionists. ● the system informs the nutritionists and doesn't allow the nutritionists to proceed. The system fails to add posts because there is no internet connection. ● The system informs the nutritionists that can't delete it.

Table 2.8 use case template to delete the posts.

Use case	Communication process.
Actor	Nutritionists, users.
Description	Nutritionists can communicate with the user by messages on a chat.
Inputs	Users can send and receive messages to a nutritionist that he/she selected to communicate with him/her and vice versa.

outputs	The system displays these messages on user's and nutritionist's chat.
Procedures	● The user selects a nutritionist that wants to talk with him/her. ● The system displays the nutritionist's profile. ● Users select the talk to the experts' button to start chatting. ● The system displays messages on nationalist's messages when login to the application. ● The system sends a form to users to fill it. ● The nutritionist can replay and send a message at a predefined time that the nutritionist chose it. ● The nutritionist and user can send and receive messages.
Exceptions	The system fails to authenticate the nutritionists. ● the system informs the nutritionists and doesn't allow the nutritionists to proceed. The system fails to add posts because there is no internet connection. ● The system informs the nutritionists that can't delete it.

Table 2.9use case template for communication process.

Use case	Approve new request to be nutritionists
Actor	Administrator.
Description	Before entering the system, any nutritionist must be approved by the system administrator. After reading and watching the required things.
Inputs	
outputs	After verifying the request, new nutritionists are added to the system.

Procedures	● Log in to the system administrator control panel. ● Go to the list of requests. ● Open the requests that were submitted. ● Verifying the requests. ● Click on the "Accept" button.
Exceptions	● Submit a request that contains incorrect data. ● The lack of an internet connection to enter the system control panel.

Table 2.10 use case template for approving new requests to be nutritionists.

Use case	Remove nutritionists from the system.
Actor	Administrator.
Description	Removing a nutritionist from the system by the system administrator.
Inputs	
outputs	Remove an expert from the system.
Procedures	● Log in to the system control panel. ● Go to the option to remove an expert from the system. ● Click the remove button.
Exceptions	No internet connection available.

Table 2.11 use case template to remove nutritionists from the system.

Use case	Adding new nutritionists

Actor	Administrator.
Description	Instead of submitting a request, the system administrator, through the control panel of the system, adds a new nutrition expert.
Inputs	Create a new user account and give it the Privileges of a nutritionist with entering all the necessary data.
outputs	Introducing a new nutritionist to the system.
Procedures	● Log in to the system control panel. ● Enter the option to add a new nutritionist to the system. ● Create a new user account. ● Enter the necessary data to grant the account the powers of a nutritionist. ● Click on the Add New Expert button.
Exceptions	● Lack of internet connection. ● Wrong data entry. ● Availability of another nutritionist with the same name.

Table 2.12 use case template for adding a new nutritionist.

Use case	Adding a new location
Actor	Administrator.
Description	To enter a healthy food restaurant or a new sports club in the system by entering all the necessary data.
Inputs	New location coordinates, photos of the location, description.

outputs	A new location on the map.
Procedures	● Log in to the system control panel. ● Entering the option to add a new location. ● Enter the coordinates of the location. ● Enter a location category. ● Enter a picture for the location. ● Enter a description of the location.
Exceptions	● Lack of internet connection. ● Wrong data entry. ● The existence of the same location.

Table 2.12 use case template for adding new location.

Use case	Update location
Actor	Administrator.
Description	If the information on specific location changes, the system administrator makes the necessary updating.
Inputs	New location coordinates, photos of the location, description.
outputs	Update this location on the map.
Procedures	● Log in to the system control panel. ● Entering the list of locations. ● Choose the desired location. ● The data for this site is displayed in a new window to make the necessary updating. ● Press the update button.

Exceptions	● Lack of internet connection. ● Enter invalid data.

Table 2.13 use case template for updating location.

Use case	Delete location
Actor	Administrator.
Description	Deleting a specific location that no longer provides the required services.
Inputs	
outputs	Remove location from the map.
Procedures	● Log in to the system control panel. ● Entering the option to delete a location. ● Choose the location to delete. ● Click on the Delete button. ● Confirm the deletion.
Exceptions	● Lack of internet connection. ● Enter the wrong coordinates.

Table 2.13 use case template for deleting location.

Use case	Provide subscription requests as nutritionists.

Actor	User.
Description	The user submits a request to transfer his account to the account of a nutritionist. After submitting the required matters, if they comply with the conditions, then they are agreed upon by the system administrator.
Inputs	● A copy of the ID card. ● Photograph. ● At least one phone number.ss ● Personal data. ● Pictures of papers proving that this user has experience in the field of nutrition (university degree).
outputs	Adding an expert request to the system control panel.
Procedures	● Log in to the system. ● Go to the menu. ● From the menu go to settings. ● From the settings, go to my account. ● Choose the option to transfer the account to a nutritionist account. ● Enter the required data. ● Click on the button Submit.
Exceptions	● Enter the wrong data. ● No internet connection is available.

Table 2.14 use case template to Provide subscription requests as nutritionists.

Use case	Calculate calories for daily needs
Actor	User, Nutritionist

Description	Users and nutritionists should be able to calculate daily caloric needs.
Inputs	
outputs	Daily caloric requirement.
Procedures	● Log in to the system. ● From the menu in the corner enter the Calorie Calculator. ● Enter the necessary data. ● Click on the button calculate calorie
Exceptions	Enter invalid data

Table 2.15 use case template to calculate calories for daily needs

Use case	Tracking progress
Actor	User.
Description	It enables the user to track his daily routine through charts that show him the number of steps, the meals he eats, and charts that show daily caloric consumption.
Inputs	
outputs	
Procedures	● Log in to the system. ● Go to the profile. ● Charts Review.

Exceptions	Wrong data entry

Table 2.16 use case template to track progress.

Use case	Add new feedback for Nutritionists
Actor	User.
Description	Send feedback or suggestions to a nutritionist.
Inputs	● The name of the nutritionist to whom the feedback will be sent is added by the system. ● Text containing feedback
outputs	Add the feedback on the nutritionist's page.
Procedures	● Log in to the system. ● Through the nutritionist's post or his page, choose the option to send feedback. ● A dialog opens, for sending feedback. ● Enter the content the user wants to submit in a text box. ● Click on the Send button.
Exceptions	● Wrong data entry. ● No internet connection is available.

Table 2.16 use case template to add new feedback for Nutritionists.

Use case	Calculate calorie for a specific type of food

Actor	User, Nutritionist.
Description	The nutritionist and the user can calculate the calories for a specific food through a menu in the system that is added by the system administrator.
Inputs	Name of the food.
outputs	The number of calories in the food chosen.
Procedures	● Log in to the system. ● From the menu in the corner. ● Choose the option to count calories for food. ● Move the food page. ● Food selection. ● Click on the calculate calorie button.
Exceptions	No internet connection is available.

Table 2.17 use case template to calculate calorie for a specific type of food.

Use case	Subscribe to nutritionists
Actor	User.
Description	The user can partner with a nutritionist to create a special diet. And to reach the ideal weight.
Inputs	

outputs	Adding this nutritionist to this user, and displaying his posts on the home page for this user.
Procedures	● Log in to the system. ● Search for a specific nutritionist or sister from the exploration page. ● Entering the nutritionist's page ● Click on the Subscribe button.
Exceptions	No internet connection is available.

Table 2.18 use case template to subscribe nutritionist.

Use case	Plan a new diet for subscribers.
Actor	Nutritionist.
Description	The nutritionists create a plan for specific Subscribers.
Inputs	● Required meals. ● Determine the calories. ● Several meals are required. ● Duration of the plan. ● Diet type.
outputs	Adding a plan for a specific subscriber, adding this plan to the user's home page.
Procedures	● Log in to the system. ● Go to the profile. ● Choose the option to add a new plan. ● Enter the plan data. ● Choose a specific subscriber.

	● Click on the Add Plan button.
Exceptions	● No internet connection is available.

Table 2.19 use case template to plan a new diet for subscribers.

Use case	Tracking subscriber
Actor	Nutritionist.
Description	The ability of a nutritionist to track subscribers through the plans that the nutritionist sets for them and by communicating with them via chat.
Inputs	Subscriber's username.
outputs	Just reviewing the stages reached by the subscriber. And follow the progress of the subscriber. reaching the target or not.
Procedures	● Log in to the system. ● Go to the profile. ● Go to subscribers. ● Move to a new screen to display subscribers. ● Choose a specific subscriber from the list of subscribers. ● long press to view the subscriber's status.
Exceptions	No internet connection is available.

Table 2.20 use case template for tracking subscribers.

Use case	Adding a new type of food
Actor	Administrator.
Description	The administrator enters a new item of food into the system through the system's control panel.
Inputs	● The type of food. ● Select the category. ● The calories it contains. ● Description of food.
outputs	Adding a new food item to the system.
Procedures	● Log in to the system control panel. ● Go to the Food option. ● Choose the option to add new food. ● Enter all the required information. ● Data verification. ● Click on the "Add a new item" button.
Exceptions	● No internet connection is available. ● Wrong data entry.

Table 2.21 use case template for adding a new type of food.

Use case	Update type of food
Actor	Administrator.

Description	Updating the data for a specific food item.
Inputs	Choose the name of the food from the specified category.
outputs	New item.
Procedures	● Log in to the system control panel. ● Go to the Food option. ● Choose the option to edit an item. ● Choose the category from the dropdown list. ● Food selection. ● Opens a window containing this item's information. ● Make the necessary updating. ● Click on the "Edit Item" button.
Exceptions	● No internet connection is available. ● Wrong data entry.

Table 2.22 use case template for updating a new type of food.

Use case	Delete type of food
Actor	Administrator.
Description	Deleting a specific food item from the system.
Inputs	Choose the name of the food from the specified category.
outputs	Remove food items from the system.

Procedures	● Log in to the system control panel. ● Go to the Food option. ● Choose the option to delete an item. ● Choose the category from the dropdown list. ● Food selection. ● Click on the "Delete Item" button.
Exceptions	No internet connection is available.

Table 2.23 use case template for deleting a type of food.

Use case	Calculate body mass index(BMI).
Actor	User, Nutritionist.
Description	calculate body mass index in one number that can help tell if someone's weight is healthy.
Inputs	Enter the height of the user. Enter the weight of the user.
outputs	BMI of the user and indicates that a person is within the normal weight range for his or her height and displays that a person is underweight, normal, overweight, or obese.
Procedures	● Log in to the system. ● Enter calculate BMI option. ● Enter weight and height. ● Enter calculate BMI.

Exceptions	• No internet connection is available.

Table 2.24 use case template to calculate BMI.

Chapter3

Software Design

Table of Contents

1. **Introduction**
2. **Context Diagram**
3. **System Architecture**
4. **Alternatives for our technologies**
5. **Class Diagram**
6. **Database Design**
7. **Database tables**

3.1 Introduction

This chapter includes an explanation of the design, tools, and structure of the project, where the components and parts of the system will be detailed, to give a complete idea of all parts of the system.

3.2 Context Diagram

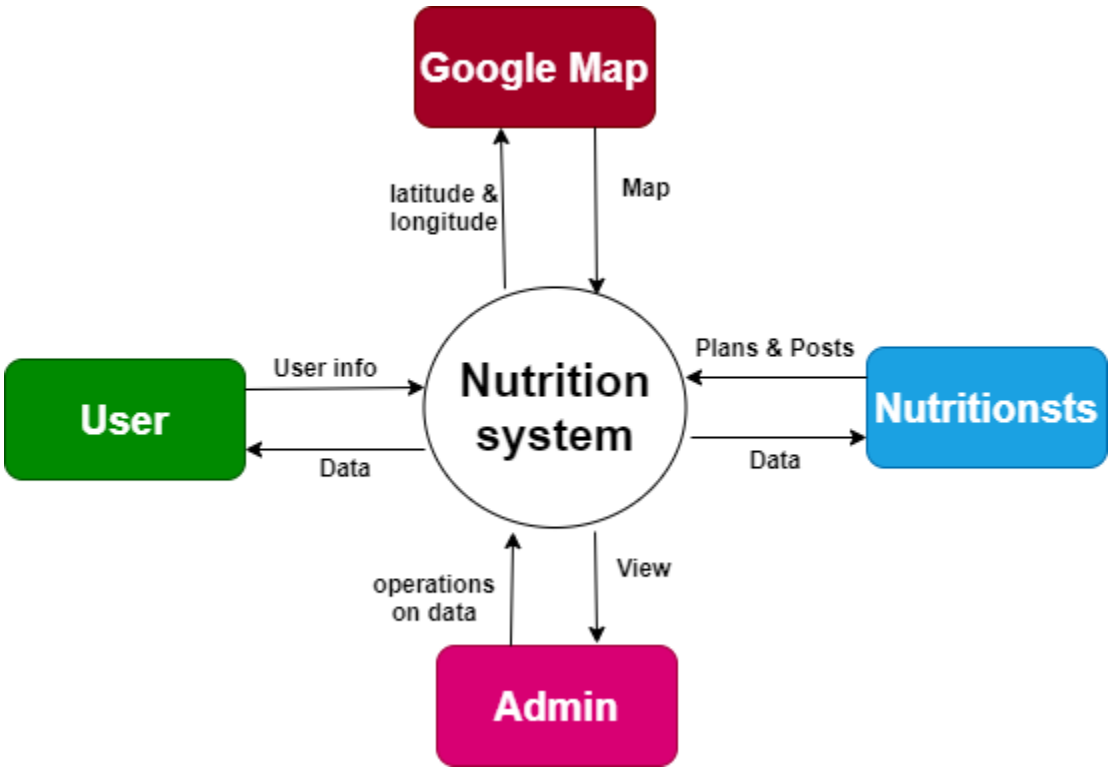


Figure 3.1 context diagram.

3.3 System Architecture

Our system is divided into two-part :

Part1 Mobile Application (Technology used is Flutter framework)

3.3.1 Mobile architecture using Flutter

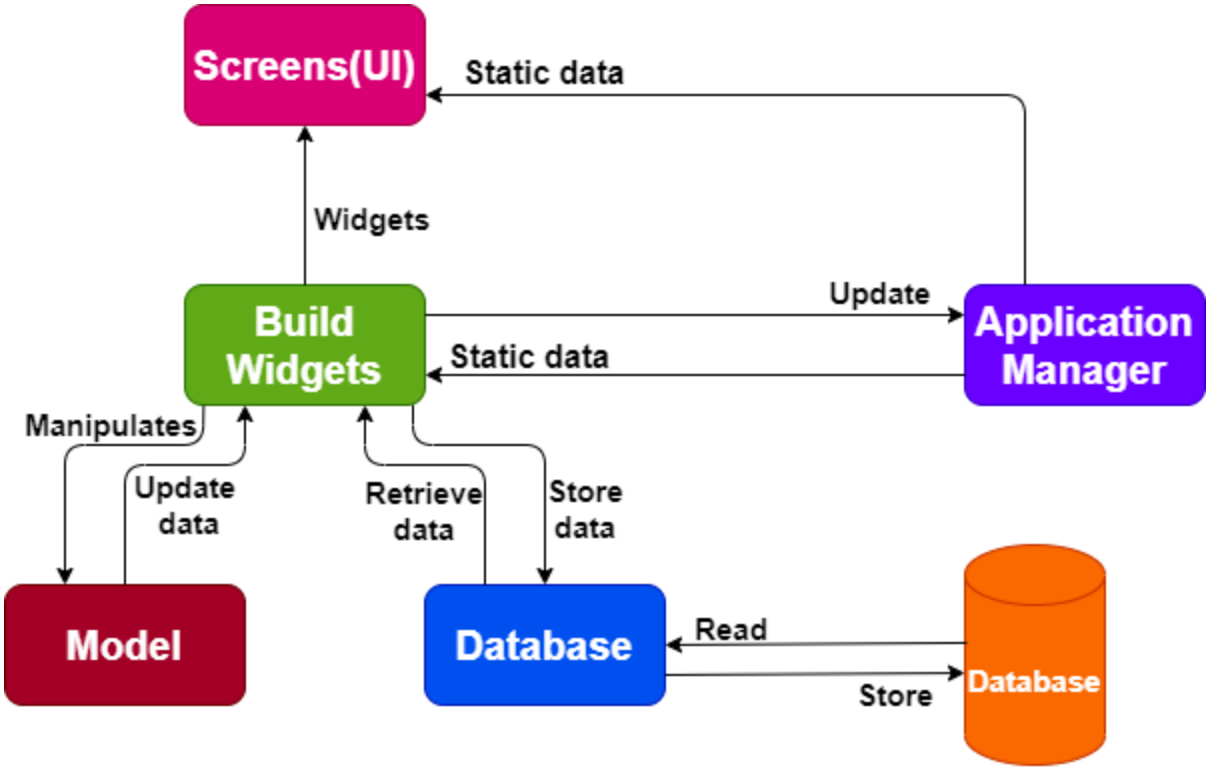


Figure 3.2 Mobile architecture using Flutter

3.3.1.1 Widget

Everything in Flutter is a widget that means a description of a part of the UI, and it's a way to declare and construct UI and it may handle user interaction.

3.3.1.2 Component details

This section describes each layer in mobile architecture

A. Screens(UI)

It contains the main screens that are displayed in the app. For more information about these screens see section [3.8].

B. Build Widgets

Building some widgets that are used in all application screens (Shared components) or build widgets for a specific screen.

C. Model

This layer defines the interfaces or interactions between different classes, inheritance, encapsulation, and other object-oriented interfaces and features. See the class figure[3.5].

D. Database

Used to store data on the server or local database (SQLite).

E. Application Manager

It contains many files that are used in managing the application, some of which are related to the graphical interface such as color manager, some files to manage data in the application.

3.3.1.3 Advantages for flutter architecture

- Build widgets layer provides state management and UI logic separation.
- Component-Based architecture reduces the cost of development and maintenance.
- It is reusable which means it can be used to reusable components to spread the development and maintenance cost across several applications.
- It increases the security of the whole system via reuse.
- It is easy to maintain and update the implementation without affecting the rest of the system.
- If the new compatible versions are available then it is easy to replace the existing versions without any impact on the other components.

<u>Part 2</u> Website for control panel (Technologies used are MEAN stack)

3.3.2 Architecture For Control panel Website (MVC)

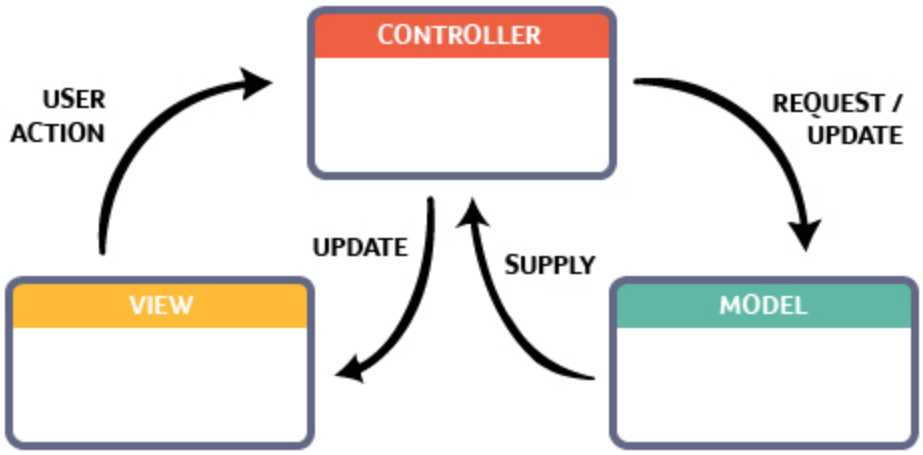


Figure 3.3 Architecture For Control panel Website (MVC).

3.3.2.1 Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: model, view, and controller. Each of these components is built to handle specific development aspects of an application.

3.3.2.2 Component details

A. Model

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data.

In conclusion: We are going to have a model layer that is going to represent all the models required for the system. See section [3.5].

B. View

The View component is used for all the UI logic of the application.

For creating views you may use template engines, frontend framework which is our choice, or other technologies.

In conclusion: Our view layer for the website will be represented by the frontend framework (Angular) and the mobile application will be represented by the flutter framework. See section [3.8].

C. Controller

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component, and interact with the Views to render the final output.

In conclusion: We have a controllers layer that is going to represent all the controllers required for the system

3.4 Alternatives for our technologies

* We used MEAN Stack for our website and Flutter for the mobile application.

A. Website VS Mobile application:

We built a website for the control panel because it makes it easier for the admin to add, edit, and delete data and provides the ability to access it from any device.

We built the rest of the system as a mobile application because mobile phones are more popular than personal computers which are going to increase our potential user's percentage.

B. MEAN Stack VS other web stacks in general and specially MERN (R for React.js)

MEAN: MySQL, Express.js, Angular, Node.js

MEAN Stack enables you to make scalable high-performance single-page applications that render on the client-side.

Angular is a complete solution framework that makes it easier for you to build complex applications than react.js for example because react.js needs third-party libraries a lot more than angular.

C. Flutter

- Flutter VS Native technologies

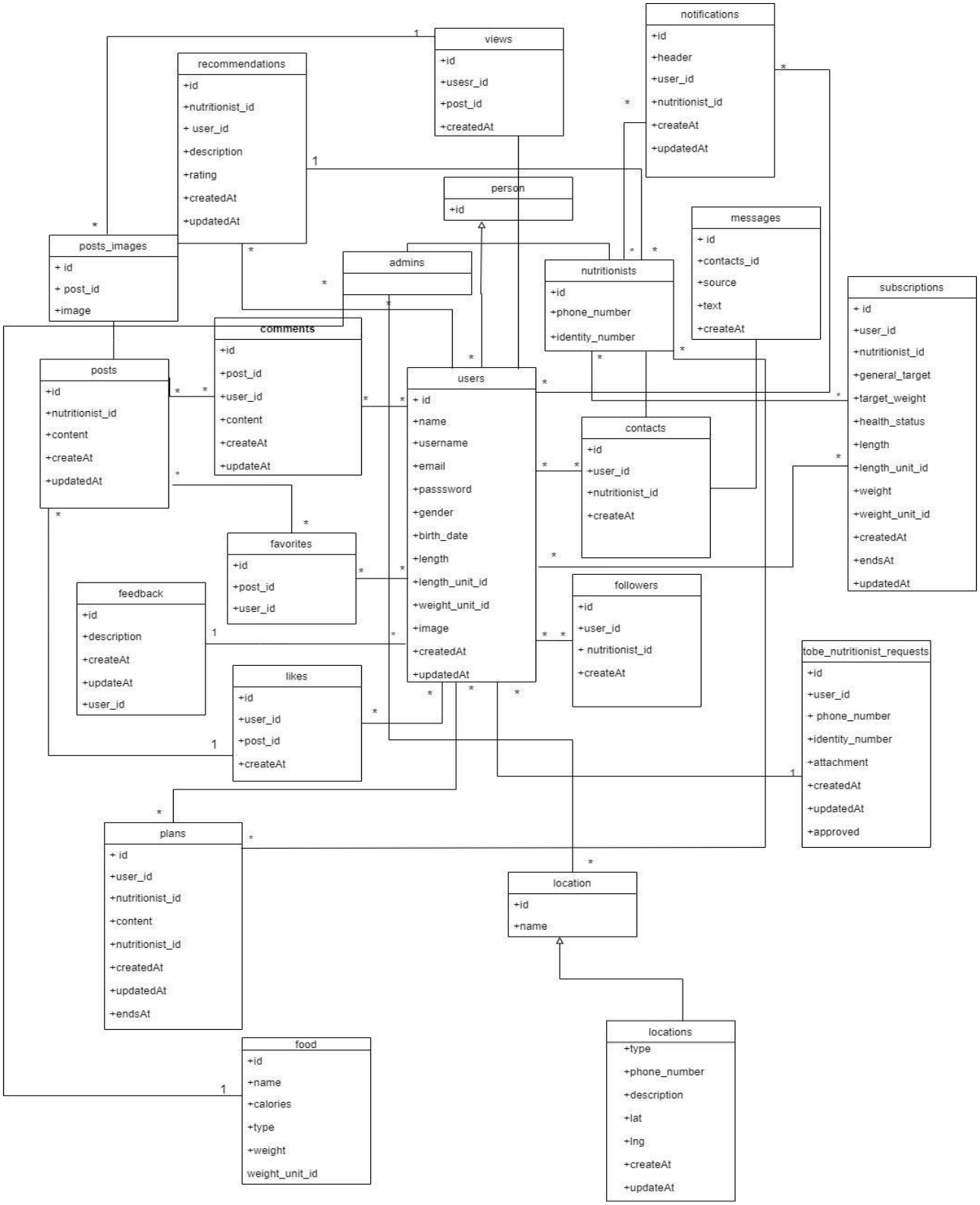
  Native technologies aim to build applications for a specific platform (Android or IOS).

  Flutter aims to build multi-platform applications (Android and IOS) and that saves a lot of time and effort.

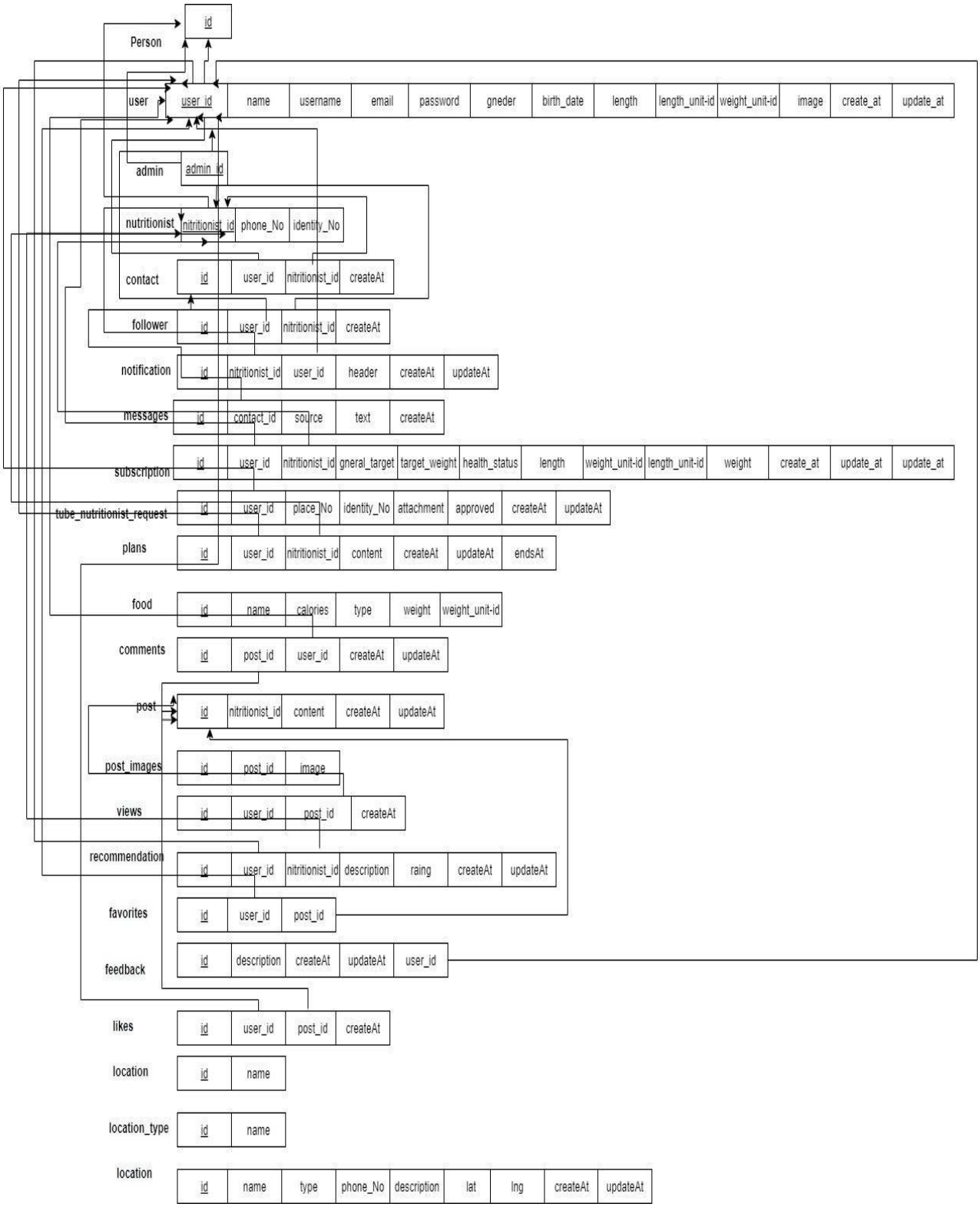
- Flutter VS other cross-platform technologies

  Flutter is cross-compiled technology and that means that you write in Dart language and the compiler produces Native code for you (Android and IOS). That makes it high performance and an enhanced user experience.

3.5 Class Diagram


notifications
+id
+header
+user_id
+nutritionist_id
+createAt
+updatedAt
views
+id
+usesr_id
+post_id
+createdAt
recommendations
+id
+nutritionist_id
+ user_id
+description
+rating
+createdAt
+updatedAt
person
+id
messages
+ id
+contacts_id
+source
+text
+createAt
posts_images
+ id
+ post_id
+image
admins
nutritionists
+id
+phone_number
+identity_number
subscriptions
+ id
+user_id
+nutritionist_id
+general_target
+target_weight
+health_status
+length
+length_unit_id
+weight
+weight_unit_id
+createdAt
+endsAt
+updatedAt
comments
+id
+post_id
+user_id
+content
+createAt
+updateAt
posts
+id
+nutritionist_id
+content
+createAt
+updatedAt
users
+ id
+name
+username
+email
+passsword
+gender
+birth_date
+length
+length_unit_id
+weight_unit_id
+image
+createdAt
+updatedAt
contacts
+id
+user_id
+nutritionist_id
+createAt
favorites
+id
+post_id
+user_id
feedback
+id
+description
+createAt
+updateAt
+user_id
followers
+id
+user_id
+ nutritionist_id
+createAt
likes
+id
+user_id
+post_id
+createAt
tobe_nutritionist_requests
+id
+user_id
+ phone_number
+identity_number
+attachment
+createdAt
+updatedAt
+approved
plans
+ id
+user_id
+nutritionist_id
+content
+nutritionist_id
+createdAt
+updatedAt
+endsAt
location
+id
+name
locations
+type
+phone_number
+description
+lat
+lng
+createAt
+updateAt
food
+id
+name
+calories
+type
+weight
weight_unit_id

3.6 Database Design



3.7 Database tables

In this section, the name of the table in the database and the description related to this table are displayed

Table Name	Description
Person	This table is used to store information shared between users, nutritionists, and system administrators.
User	Save the information of the user.
Nutritionist	Save information about the nutritionists.
Admin	Only to store the ID of the system administrators.
Post	Save posts information that was published by the nutritionist.
Favorite	Save post information that the user selected as favorite.
Comment	To store user's comments on a specific post.
Users Recommendations	Save user's recommendations for specific nutritionists.
Feedback	Save the user feedback about the system.
Plan	To store a user diet plan made by a specific nutritionist.

Notification	Save notifications information sent for a specific user.
Messages	Store sent messages between the user and a nutritionist.
location_type	Save the type of location if it is a gym or restaurant.
subscriptions	Save the users that subscribe to a nutritionist.
tobe_nutritionist_requests	Save the users who want to be nutritionists.
Food	Save different food types of information.
contacts	Save the users that are subscribed to particular nutritionists.
followers	Save the users that follow a particular nutritionist.
posts_images	Save the information about the images for posts.
likes	Save the information of the users who make likes for the post.

Table 3.7.1: Person table

Field Name	**Field Type**	**Null**	**Field length**
id	int	No	20

Table 3.7.2: User table

Field Name	Field Type	Null	Field length
userId	int	No	20
length	double	No	3,2
weight	double	No	3,2

Table 3.7.3: Nutritionist table

Field Name	Field Type	Null	Field length
phone_number	varchar	No	10
identity_number	int	No	9

Table 3.7.4: Admin table

Field Name	Field Type	Null	Field length
adminId	int	No	20

Table 3.7.5: Post table

Field Name	Field Type	Null	Field length
id	int	No	20
nutritionistId	int	No	20
content	varchar	Yes	50

createAt	datetime	No	
updateAt	datetime	No	

Table 3.7.6: Favorite table

Field Name	**Field Type**	**Null**	**Field length**
Id	Int	No	20
userId	int	No	20
postId	int	No	20

Table 3.7.7: Comment table

Field Name	**Field Type**	**Null**	**Field length**
id	int	No	20
post_id	int	No	20
userId	int	No	20
content	varchar	No	50
createAt	datetime	No	
updateAt	datetime	No	

Table 3.7.8: User recommendations table

Field Name	Field Type	Null	Field length
id	int	No	20
user_Id	int	No	20
nutritionist_Id	int	No	20
description	varchar	No	50
rating	double	Yes	2,1
createAt	datetime	No	
updateAt	datetime	No	

Table 3.7.9: Feedback table

Field Name	Field Type	Null	Field length
id	int	No	20
userId	int	No	20
createAt	datetime	No	
updateAt	datetime	No	
description	varchar	No	60

Table 3.7.10: Plan table

Field Name	Field Type	Null	Field length
id	int	No	20
userId	int	No	20
nutritionistId	int	No	20
content	varchar	No	100
createAt	datetime	No	
updateAt	datetime	No	
endsAt	datetime	No	

Table 3.7.11: Notification table

Field Name	Field Type	Null	Field length
id	int	No	20
nutritionistId	int	No	20
userId	int	No	20
header	varchar	No	50
createAt	datetime	No	

updateAt	datetime	No	

Table 3.7.12: messages table

Field Name	**Field Type**	**Null**	**Field length**
id	int	No	20
contact_tId	int	No	20
source	int	No	11
text	text	No	
createAt	datetime	No	

Table 3.7.13: Place location table

Field Name	**Field Type**	**Null**	**Field length**
id	int	No	20
name	varchar	No	30
description	varchar	No	50
type	int	No	2
lat	DECIMAL	No	8,7
lng	DECIMAL	No	8,7

Table 3.7.14: Food table

Field Name	Field Type	Null	Field length
id	int	No	20
name	varchar	No	30
weight_unit_id	tinyint	No	2
calorie	DECIMAL	No	2,7
type	varchar	No	20
weight	double	No	3,2

Table 3.7.15: views table

Field Name	Field Type	Null	Field length
id	int	No	20
user_id	int	No	20
post_id	int	No	20
createdAt	datetime	No	

Table 3.7.15: posts_images table

Field Name	Field Type	Null	Field length

id	int	No	20
post_id	int	No	20
image	varchar	No	200

Table 3.7.16: contacts table

Field Name	**Field Type**	**Null**	**Field length**
id	int	No	20
user_id	int	No	20
nutritionist_id	int	No	20
createAt	datetime	No	

Table 3.7.17: followers table

Field Name	**Field Type**	**Null**	**Field length**
id	int	No	20
user_id	int	No	20
nutritionist_id	int	No	20
createAt	datetime	No	

Table 3.7.18: tube_nutritionist_request table

Field Name	Field Type	Null	Field length
id	int	No	20
user_id	int	No	20
phone_number	int	No	20
identity_number	varchar	No	10
attachment	varchar	No	200
createdAt	datetime	No	
updatedAt	datetime	No	

Table 3.7.19:likes table

Field Name	Field Type	Null	Field length
id	int	No	20
user_id	int	No	20
post_id	int	No	20
createAt	datetime	No	

Table 3.7.20:subscriptions table

Field Name	Field Type	Null	Field length

id	int	No	20
user_id	int	No	20
nutritionist_id	int	No	20
createAt	datetime	No	
updateAt	datetime	No	
endsAt	datetime	No	
weight_unit_id	tinyint	No	2
weight	int	No	3
length	decimal	No	5,2
Approved	tinyint	No	1
health-status	varchar	No	300
general_target	Varchar	NO	300
target_weight	Decimal	NO	5,2

3.8 Screens(views)

Mobile views

- **Explorer Page for guest users**

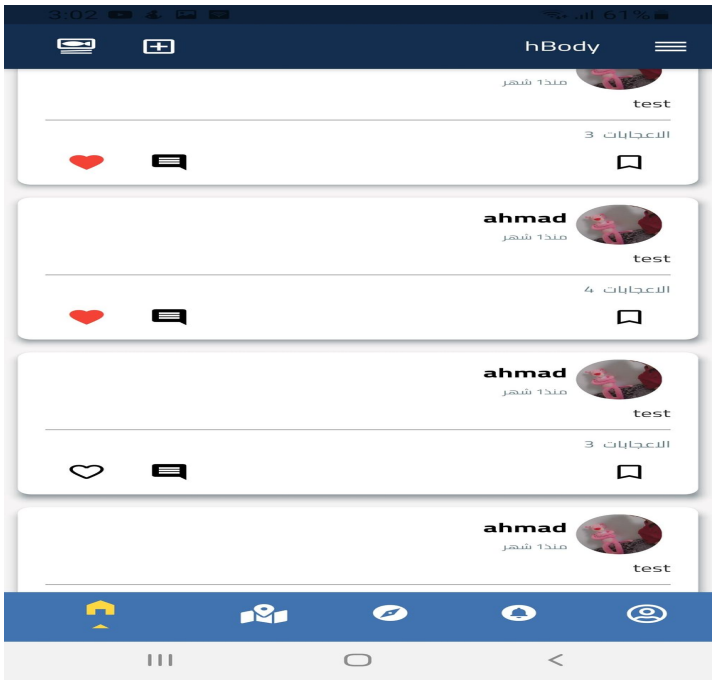


Figure 3.1 explorer Page for guest users.

- **Profile Page**

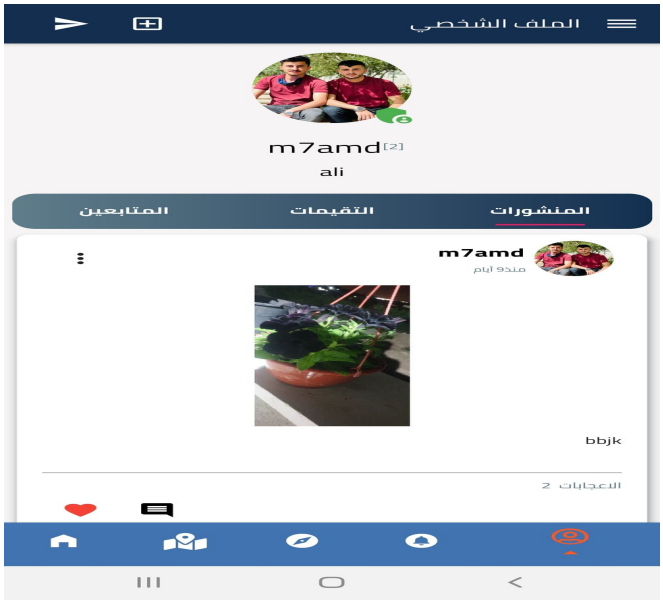


Figure 3.4 profile page interface

- **Map**

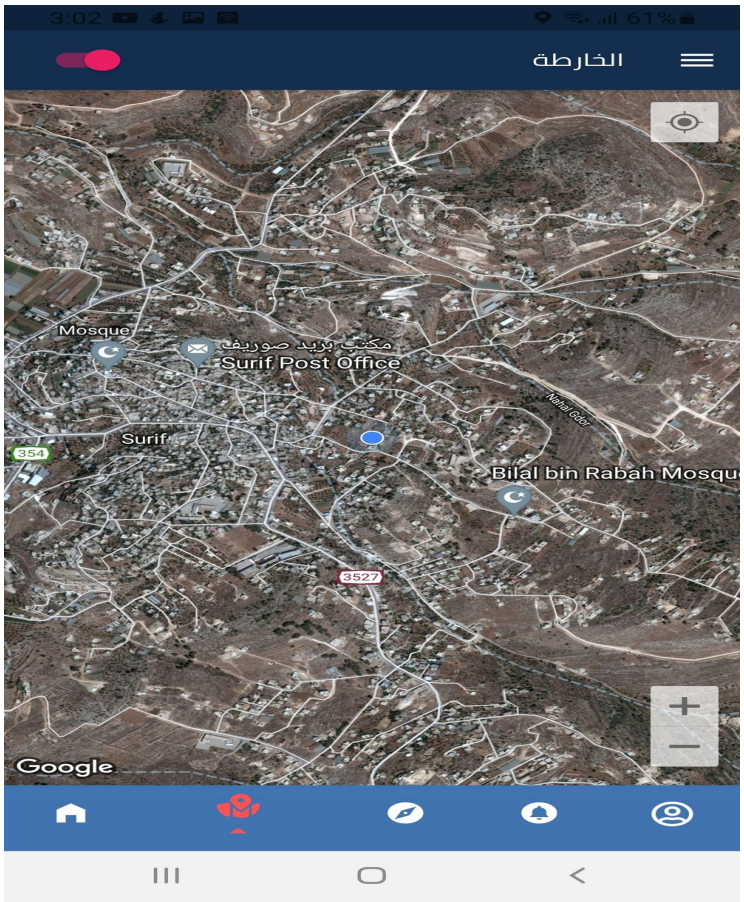


Figure 3.2 map interface

Chapter4

Software Implementation

Table of Contents

1. **Introduction**
2. **Programming environment and system building.**
    - **Mobile application**
    - **Website**

4.1 Introduction:

In this chapter, we will talk about what we did to transfer the project from the theoretical stage to a real functional app, and this includes the two parts of the project: the website and mobile application.

4.2 Programming environment and system building:

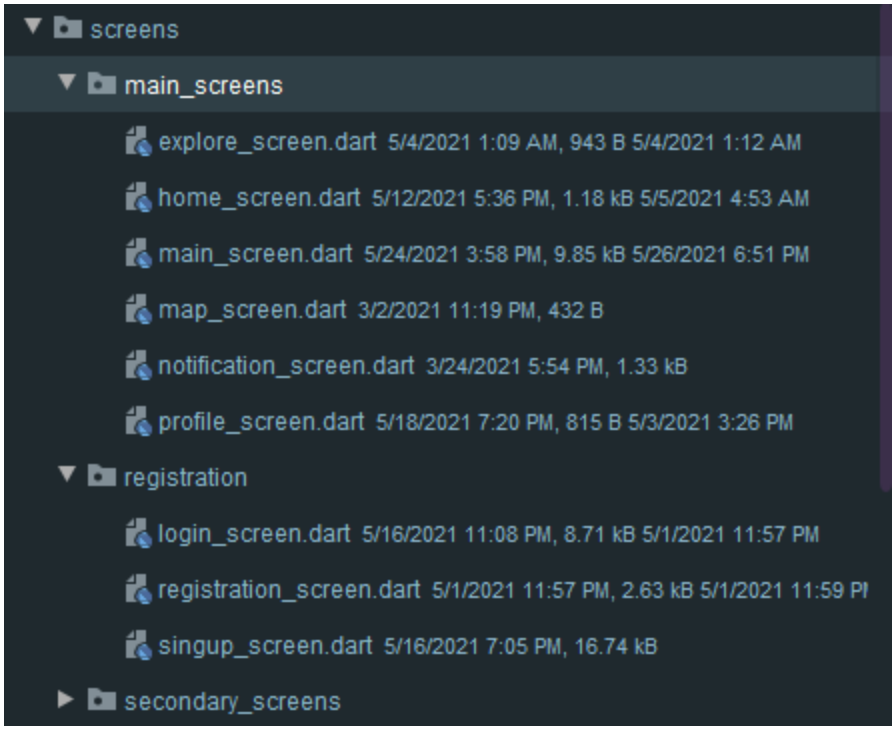
We will explain the method that we followed in building the system and the programming environment that were worked on in building the two parts of the project.

4.2.1Mobile application:

The component architecture was adopted in building a mobile application for Android and IOS devices using flutter language, and the method of building the system will be explained in this part.

- **Screens(UI)**

  It contains the main screens that are displayed in the app.

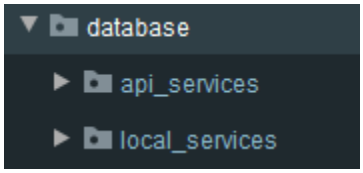


- **Build Widgets**

  Building some widgets that are used in all application screens (Shared components) or build widgets for a specific screen.

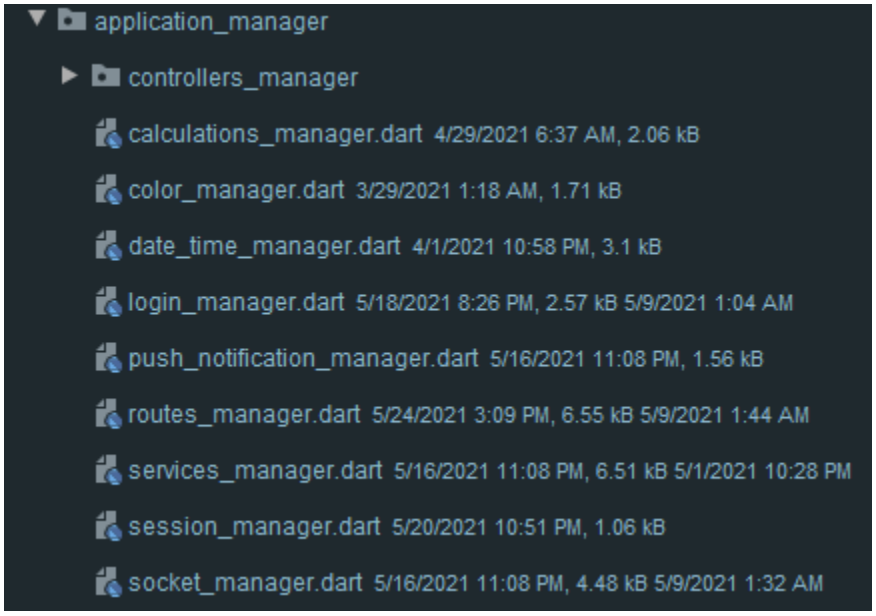
- **Database**

  Used to store data on the server or local database (SQLite).



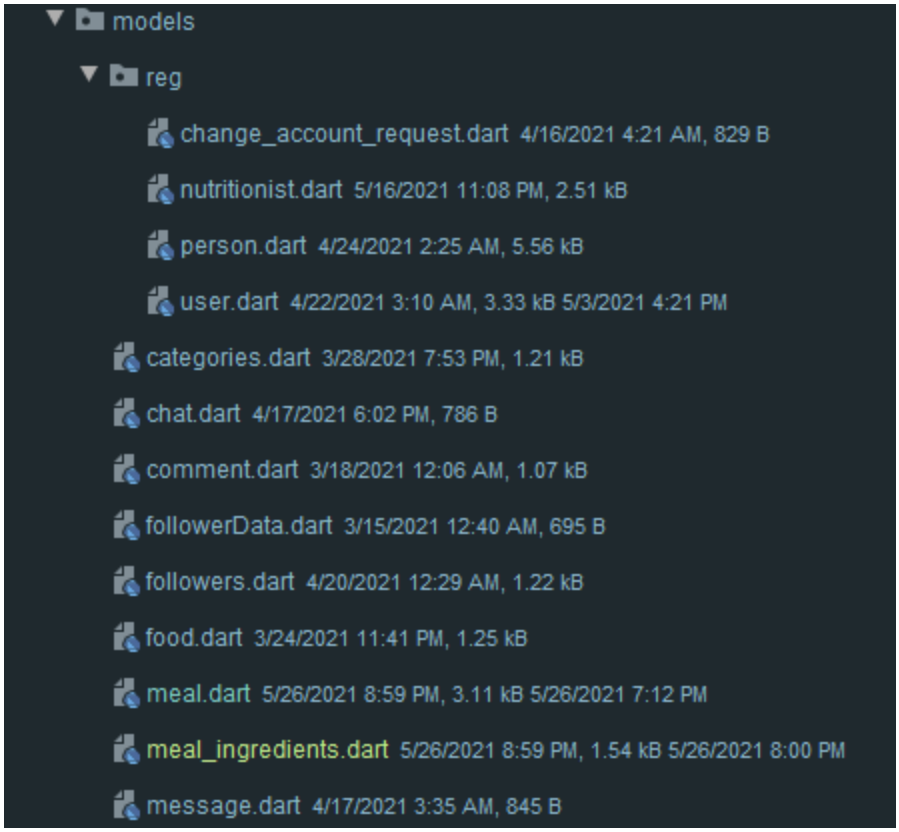
- **Application Manager**

It contains many files that are used in managing the application, some of which are related to the graphical interface such as color manager, some files to manage data in the application.



- **Model**

This layer defines the interfaces or interactions between different classes, inheritance, encapsulation, and other object-oriented interfaces and features.



Chat functionality implementation

We will describe the implementation of a chat function that enables users to communicate with nutritionists. The mobile application would act as a client to send messages to the server by using socket-io package**:**

```
void initSocketEvents() {
  if (null != this._socketIo) {
    this.clearSocket();
  }

  this._socketIo = IO.io('$_connectUrl', this._getOptionsBuilder());

  this.openConnection();

  InitControllers.chatController.setIsTyping = false;

  _socketIo.on(ERROR, (error) {
    print('Socket ex: error=>$error');
  });

  _socketIo.on(CONNECT_ERROR, (error) {
    print('Socket ex: connectError=>$error');
  });

  _socketIo.on(ON_MSG_RECEIVED, (message) {
    // InitControllers.chatController.myMessagesList
    //     .add(Message.fromJson(message));
    InitControllers.chatController.resetPageNumber();
    InitControllers.chatController
        .loadMyMessagesListByChatId(this._contact.getContactId);
    //InitControllers.chatController.setMsgLength();
  });
```

To fetch data from the backend, you need to provide the required authorization headers like fellows.

```
Map<String, dynamic> _getSocketHeaders() => {
      'auth': userToken,
      'withCredentials': true,
      'inactivityTimeout': Duration(seconds: 5),
      'secure': true,
    };
```

Main method

This main method for mobile application starts with **setEnabledSystemUIOverlays** method that Specifies the set of system overlays to be visible when the application is running. **setPreferredOrientations** method Specifies the set of orientations the application interface can be displayed in. Get_Storage Package is a very fast, extra light weight & synchronous key-value pair package that will help you to store app data in memory such as, if user is signed in, if yes, then all the details about the user.

```
void main() async{

  //To hide state bar when
  SystemChrome.setEnabledSystemUIOverlays([SystemUiOverlay.bottom]);

  //To force application Stay in portrait mode
  SystemChrome.setPreferredOrientations(
      [DeviceOrientation.portraitDown, DeviceOrientation.portraitUp]);
  //end

  //To initialize the getStorage to get access to the storage
  await GetStorage.init();
  await InitControllers.loginMangerController.checkIsFirstLogin();
  InitControllers.connectionController.setIsConnected();

  //if device connected to internet fetch units <lengthUnits,weightUnits>
  if(InitControllers.connectionController.isConnected.value){
    FetchUnits.loadUnits();
    await InitControllers.homeController.loadFollowingData();
  }

  //To run application
  runApp(HBody());

}//end main
```

MaterialApp class is a convenience widget that wraps a number of widgets that are commonly required for material design applications. It builds upon a WidgetsApp by adding material-design specific functionality, such as AnimatedTheme and GridPaper.

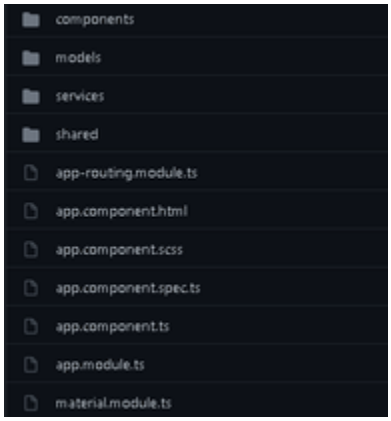
```dart
class HBody extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      debugShowCheckedModeBanner: false,
      home:Obx(
          ()=>InitControllers.loginMangerController.isFirstLogin.value
              ? RegistrationScreen()
              :MainScreen(),
      ),
    );

  }//end build method

}//end MyApp class
```

4.2.2 Website implementation:

The NodeJs platform was adopted in building a website that runs code written in typescript to create a RESTful API that receives requests from a mobile application or from a system administrator's web page, and the method of building the system will be explained in this part.

Node js:

Server part folder (API Services) built using NodeJs.It contains the main operations of the system such as dealing with the database, registering users, and carrying out their operations received from the application (Requests), which include deletion, addition, or modification. Or from the Admin Web Page.



Component:

Components are the main building block for Angular applications. Each component consists of:

- An HTML template that declares what renders on the page

- A Typescript class that defines behavior
- A CSS selector that defines how the component is used in a template
- Optionally, CSS styles applied to the template

Model

corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data.

Shared component:

Contain components that are shared for the whole project.

Service:

Contain logic that fetches data from the server.

Implementation of Registration function:
Stores users information like username, password, image on the server and generates a token for authenticating a user for each request to the server.

```
const register = async (req, res) => {
    const username = req.body.username.toLowerCase()
    const email = req.body.email.toLowerCase()
    let userData = {...req.body, username, email, birth_date: new Date(req.body.birth_date)};
    let image: string;

    let _user: any = await User.findOne({
        where: {
            username
        }
    })
    if (_user) {
        if (req.file) await unlinkAsync(dest + req.file.filename);

        return res.status(400).json({message: 'Username is taken.'});
    }

     _user = await User.findOne({
        where: {
            email
        }
    })
    if (_user) {
        if (req.file) await unlinkAsync(dest + req.file.filename);

        return res.status(400).json({message: 'Email is taken.'});
    }
```

```
    image = req.file ? req.file.filename : null;
    const password = await bcrypt.hash(userData.password, 10);

    userData = await User.create({ ...userData, password, image });

    const keys = Object.keys(userData.dataValues).filter(key => key != 'password');

    const token = generateToken({...userData.dataValues, ...DEFAULT_AUTHORIZATION});

    res.status(201).header('x-auth-token', token).json({
        ..._.pick(userData, keys),
        ...DEFAULT_AUTHORIZATION,
        token
    });
}
```

Chat implementation:

To enable communication between users and nutritionists we use **Socket.IO** primarily using the WebSocket protocol which is the internet protocol that allows full duplex communication between a server and clients. The server may send data to a client without the client initiating a request.

```
const chat = io => {
    io.use((socket, next) => {
        const token = socket.handshake.headers['auth'];
        validateToken(token, socket.request, next);
    });

    io.on('connection', socket => {
        socket.on('join-chat', async (contacts_id: number) => {
            const user = socket.request.user;

            const _contacts: any = await findContacts(contacts_id);
            if (!_contacts) return socket.emit('error', 'Contacts not found.');

            if (_contacts.user_id !== +user.id && _contacts.nutritionist_id !== +user.id)
                return socket.emit('error', 'Access forbidden');

            socket.join(contacts_id);

            socket.on('typing', ({name, typing}: {name: string, typing: boolean}) => {
                if (typing) {
                    socket.broadcast.to(contacts_id).emit('typing', `${name} is typing...`);
                } else {
                    socket.broadcast.to(contacts_id).emit('typing', ``);
                }
            });
```

Chapter5

Software Testing

Table of Contents

1. **Introduction**
2. **Testing of mobile applications**
3. **Testing the website.**

5.1 Introduction

This chapter will talk about the system testing stage, the important stage to ensure that the system works without any problems, and to ensure that the system requirements are fully completed, as this stage comes after the stage of designing and building the system.

5.2 Testing of mobile application

The mobile application was checked by a flutter test package to ensure how they worked and the results of the examination were successful. The following tables and snapshots are a review of some of the tests.

- **Test of the body mass index function:**

```dart
void main() {
  //To test getBMIStatus() in <CalculationManager>
    test(
        'BMI result is 15.0 , then BMI status should be under weight ["نقص في الوزن"]',
        () {
      final bmiStatus = CalculationsManager.getBMIStatus(15.0);
      expect(bmiStatus, 'نقص في الوزن');
    });
} //end main
```

#	case	input	Expected output	Actual output	Pass/Fail
1	Increase in the weight	15.0	نقص في الوزن	نقص في الوزن	Pass
2	Decrease in the weight	19	نقص في الوزن	وزن طبيعي	pass

- **Test of the login**

```dart
void main(){

  //To test login() in <RegistrationApiServices>
  test('login() should be return Nutritionist',()async{
    final user = await RegistrationApiServices.login('ali','123456RRRR');
    bool value = user.isNutritionist;
    expect(value,true);
  });

}//end main
```

#	case	input	Expected output	Actual output	Pass/Fail
1	Correct username and password	username=”ali” password=”123456 RRRR”	true	true	Pass
2	Wrong username and password	username=”alii” password=”123456”	false	false	pass

- **Test of the unique username in the database for registration**

```dart
void main(){

  //To test checkNameOrEmailIsAvailable() in <FetchData>
  test('isAvailableName should be false ,'
      ' because this name already is used from another user',()async{
    final checkOn = 'username';
    final isAvailableName =  await FetchData.checkNameOrEmailIsAvailable('ali',checkOn);
    expect(isAvailableName,false);
  });

}//end main
```

#	case	input	Expected output	Actual output	Pass/Fail
1	enter username already in database	username=”ali”	false	false	Pass
2	enter username doesn’t exist in database	username=”ameer”	true	false	pass

- **Test of the difference between current date and entered date:**

```dart

void main(){

  //To test getTimeDifference() in <DateTimeManager>
  test('value should be equal "now" ["الآن"]',()async{
    final currentDate =DateTime.now(); //passed
    //DateTime.parse('2021-03-01 16:19:14'); //failed
    final value = DateTimeManager.getTimeDifference(currentDate);
    expect(value,'الآن');
  });

}//end main
```

5.3 Test of website

The test of the backend of the website was done by using a working environment (Jest) developed by Facebook, which is dedicated to examining lines of code written in JavaScript and it also works with node js and typescript, the examination was successfully passed. The following is an explanation of one of the checks that were performed. The following image shows a login of a user in the system:

- **Test of login :**

```
it('should login the user using email', async () => {
        const res = await request.post('/api/login')
        .send({
            username_or_email: 'test@test.com',
            password: 'test123'
        });

        expect(res.status).toBe(200);
        expect(res.body).toHaveProperty('token');
    });

    it('should not login', async () => {
        const res = await request.post('/api/login')
        .send({
            username_or_email: 'testt@test.com',
            password: 'test123'
        });

        expect(res.status).toBe(400);
        expect(res.body.message).toBe('Invalid email or password!');
    });
```

#	case	input	Expected output	Actual output	Pass/Fail
1	enter correct email and password	email='test@test.com' password='test123'	User has a token	User has a token	Pass
2	enter wrong email or password	email='testt@test.com' password='test123''	Invalid username or password	Invalid username or password	Pass

Chapter6

Results and Future Plans

Table of Contents

1. **Results.**
2. **Future plans.**

6.1 Results:

This project was built based on the needs of individuals who have an interest in his/her health care and perhaps the result of this system is facilitating access to healthy restaurants and gyms, facilitate the process of viewing the calories of multiple types of foods, ability to calculate the daily calories needed by the user, calculate calories for any food type the user enters in the system and facilitates the communication process between nutritionists and users.

6.2 Future plans:

In the future, we look forward to adding important features to the system, the most important of which are:

- The map of the restaurant and gym include the whole of Palestine and the Arabian world.
- The user enters the ingredients and the system generate a meal of these ingredients.
- The mobile application includes videos of sports activities.