

MINISTRY OF EDUCATION



---

**TECHNICAL UNIVERSITY**

OF CLUJ-NAPOCA, ROMANIA

**FACULTY OF AUTOMATION AND COMPUTER SCIENCE**

# **Bite Balance - Calorie Counting and Nutritional Analysis Mobile Application using Artificial Intelligence**

LICENSE THESIS

Graduate: **Timea NAGY**

Supervisor: **Sl.dr.ing. Claudiu DOMUTA**

**2024**



**TECHNICAL UNIVERSITY**

OF CLUJ-NAPOCA, ROMANIA

**FACULTY OF AUTOMATION AND COMPUTER SCIENCE**

DEAN,  
Prof. dr. ing. Mihaela DINSOREANU

HEAD OF DEPARTMENT,  
Prof. dr. ing. Honoriu VĂLEAN

Graduate: **Timea NAGY**

**Bite Balance - Calorie Counting and Nutritional Analysis Mobile Application using Artificial Intelligence**

1. **Project proposal:** A mobile application that calculates the total number of calories from photos by recognising the respective food in user-taken pictures. It also suggests calorie needs based on weight, age, gender, and other factors. The app calculates the total amount of nutrients such as proteins and fibres, along with the daily requirements based on the individual's goals. Additionally, it recommends other food items that should be consumed on that day, such as a banana or a serving of rice, with the goal of making calorie intake calculation more easy.
2. **Project contents:** Presentation page, advisor's evaluation, introduction, objectives, bibliographic research, analysis and theoretical foundation, detailed design and implementation, testing and validation, user's manual, conclusions, bibliography and appendices.
3. **Place of documentation:** Technical University of Cluj-Napoca, Automation and Applied Informatics Department
4. **Consultants:** N/A
5. **Data of issue of the proposal:** October 10, 2023
6. **Data of delivery :** July 12, 2024

Graduate: \_\_\_\_\_

Supervisor: \_\_\_\_\_

**TECHNICAL UNIVERSITY**

OF CLUJ-NAPOCA, ROMANIA

**FACULTY OF AUTOMATION AND COMPUTER SCIENCE**

**Declarație pe proprie răspundere privind  
autenticitatea lucrării de licență**

Subsemnatul(a) Nagy Timea, legitimat(ă) cu CI seria CJ nr. 401264 CNP 6010302125848, autorul lucrării Bite Balance - Calorie Counting and Nutritional Analysis Mobile Application using Artificial Intelligence elaborată în vederea susținerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea Automatică și Informatică Aplicată din cadrul Universității Tehnice din Cluj-Napoca, sesiunea Iulie 2024 a anului universitar 2024, declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

Data

11/07/2024

Timea NAGY

Semnătura



**TECHNICAL UNIVERSITY**

OF CLUJ-NAPOCA, ROMANIA

**FACULTY OF AUTOMATION AND COMPUTER SCIENCE**

## **SINTHESIS**

Of the thesis paper with the title

### **Bite Balance - Calorie Counting and Nutritional Analysis Mobile App using Artificial Intelligence**

Graduate: **Timea NAGY**

Supervisor: **SI.dr.ing. Claudiu DOMUTA**

1. **Theme requirements:** Developing a software solution that calculates calories and consumed nutrients from user-taken pictures, calculates the total calories and nutrients the user should consume based on users goal and physical data, and displays the consumed/recommended caloric and nutrient rate.
2. **Solutions:** Implementing a cross-platform mobile application in Flutter with Firebase & JavaScript back-end, connected to a local database.
3. **Obtained results:** A mobile application that has a user management system (login, register), has a profile page with detailed information about the user, has a history page that detaliates consumed foods and its nutrients categorised in meals and on the main page the user can see detailed information about the total consumed nutrients of the current day.
4. **Tests and verification:** The application was tested during the implementation process, with different use cases and user credentials.
5. **Personal contribution:** The personal contributions include designing and developing the user interface, creating all back-end connections to the database, CRUD operations, integrating secure Firebase authentication, training and integrating the food recognition model and also analysing other fully functioning applications.
6. **Bibliographical sources:** The sources of documentation for this project were mainly scientific articles and also websites and articles that provide information for the development process and detailed code examples and also tutorials.

Graduate: \_\_\_\_\_

Supervisor: \_\_\_\_\_

# Contents

|  |           |
|--|-----------|
| <b>Chapter 1 Introduction</b>                              | <b>1</b>  |
| 1.1 General Context . . . . .                              | 1         |
| 1.2 Objectives . . . . .                                   | 1         |
| 1.3 Specifications . . . . .                               | 2         |
| <b>Chapter 2 Bibliographic Research</b>                    | <b>3</b>  |
| 2.1 Study of nutritional analysis . . . . .                | 3         |
| 2.1.1 Macronutrients and Daily nutritional needs . . . . . | 3         |
| 2.1.2 Dietary plans . . . . .                              | 5         |
| 2.1.3 Nutrient calculations, formulas . . . . .            | 6         |
| 2.2 State of art . . . . .                                 | 8         |
| 2.2.1 CalorieMama . . . . .                                | 8         |
| 2.2.2 Lifesum . . . . .                                    | 9         |
| 2.2.3 MyNetDiary . . . . .                                 | 9         |
| 2.2.4 Cronometer . . . . .                                 | 10        |
| 2.2.5 SnapCalorie . . . . .                                | 11        |
| 2.2.6 Conclusions . . . . .                                | 11        |
| 2.2.7 BiteBalance . . . . .                                | 11        |
| <b>Chapter 3 Analysis and Theoretical Foundation</b>       | <b>12</b> |
| 3.1 Flutter Software Development Kit . . . . .             | 12        |
| 3.1.1 Dart programming language . . . . .                  | 12        |
| 3.2 Firebase Software Development Kit . . . . .            | 12        |
| 3.2.1 Firebase Authentication . . . . .                    | 13        |
| 3.2.2 Cloud Firestore . . . . .                            | 13        |
| 3.2.3 Cloud Storage - Realtime Database . . . . .          | 13        |
| 3.2.4 Hosting . . . . .                                    | 13        |
| 3.3 JavaScript . . . . .                                   | 14        |
| 3.3.1 Node.js . . . . .                                    | 14        |
| 3.3.2 Express.js . . . . .                                 | 14        |
| 3.4 Microsoft SQL Server Management Studio . . . . .       | 14        |
| 3.5 Artificial Intelligence . . . . .                      | 15        |
| 3.6 Machine Learning . . . . .                             | 15        |
| 3.7 Image Recognition . . . . .                            | 15        |
| 3.8 Transfer Learning . . . . .                            | 16        |
| 3.9 Convolutional Neural Networks . . . . .                | 16        |
| 3.10 TensorFlow . . . . .                                  | 17        |
| 3.11 Keras . . . . .                                       | 17        |
| <b>Chapter 4 Detailed Design and Implementation</b>        | <b>18</b> |
| 4.1 Application's architecture . . . . .                   | 18        |
| 4.2 Application Design . . . . .                           | 20        |
| 4.3 Implementation details . . . . .                       | 24        |
| 4.3.1 Image recognition model - Python . . . . .           | 24        |
| 4.3.2 Front-End - Flutter . . . . .                        | 26        |

|   |           |
|---|-----------|
| 4.3.3 Back-End                            | 31        |
| 4.3.4 Database Design                     | 35        |
| <b>Chapter 5 Testing and Validation</b>   | <b>37</b> |
| 5.0.1 Testing the BiteBalance application | 37        |
| 5.0.2 Testing the food recognition model  | 37        |
| <b>Chapter 6 User's Manual</b>            | <b>38</b> |
| 6.0.1 Requirements and Installation       | 38        |
| 6.0.2 Running the application             | 38        |
| <b>Chapter 7 Conclusions</b>              | <b>39</b> |
| 7.0.1 Obtained results                    | 39        |
| 7.0.2 My own contributions                | 39        |
| 7.0.3 Areas of future improvement         | 39        |
| <b>Bibliography</b>                       | <b>41</b> |

# Chapter 1. Introduction

## 1.1. General Context

In today's busy world, being able to precisely track every bite to maintain a healthy and balanced diet is a time consuming and challenging task, that most people do not have time for. The majority of people have the desire to loose or gain weight, which requires a detailed nutritional plan and an efficient calorie tracking method.

Each one of us wants to be healthy, but this requires a balanced and healthy diet as well. People need to realise that diet plays a major role is being healthy. By writing this paper and developing this app, my goal is to help people realise how important keeping a healthy diet is. Normally, dietary advice should be given by a dietitian, an expert in the domain, but there is also need for a more affordable method which is accessible to everyone.

I choose to develop a mobile app that helps users reach their dietary goals. By adding the functionality of recognising food from user-taken photos, the application makes calorie tracking easier by eliminating the monotonous task of manually logging each bite consumed by itself. Of course, this recognition and nutrient estimation can have slight errors, as the portion estimation is not so precise at the moment, but it gives an idea about the consumed amount of food, and provides a general feedback of the consumed nutrients. It is important to note that the main objective of Bite Balance is not to achieve a high degree of precision in measurements, but to provide an efficient way to helps users monitor their progress and to make calorie counting easier, accessible and more fun.

## 1.2. Objectives

The main objective of this paper is to create a cross platform mobile application that redefines the way people monitor their daily caloric and nutrient intake. The differentiating functionality that the app BiteBalance has, is the recognition of food items from user-taken pictures. Users can either take a picture of their meal and add caloric consumption from the picture, or they can enter manually the consumed food by name and the app adds its calories and nutritional information. This makes the app versatile and convenient, the user is able to take a quick photo and save the time required to enter each item on its own, or if the user can not currently take a photo or if the food has already been consumed and no photo was taken, users can simply input what they ate and log nutritional information that way.

Besides building the above mentioned application, another objective of this paper is to create a food recognition model with a high precision that is able to recognise food items effectively. The goal is to recognise not only one simple food, but also more complex ones, containing more ingredients thus more food items combined.

The application should provide all functionalities enumerated in the following section.

### 1.3. Specifications

This section contains the specifications of the Bite-Balance application. All the things the application should be able to perform are listed below:

1. User authentication, account creation for users and app state management depending on user actions. Users can log in with their existing Google account. The application is connected to Google services, which also offers password changing option and password reset option.
2. Email verification for validating account.
3. Daily caloric intake and nutrient needs calculation based on user data: height, current weight, weight goal, gender, age, activity level and personal goal (balanced diet, loose weight, get fit, gain weight, get toned). Research about different diets, and how the personal goal affects daily caloric and nutrient intake.
4. Displaying a history of consumed foods. This functionality will be displayed on a separate screen, under 'History' menu button. It will detail the consumed foods per meals, the total amount of calories per meal, and the nutritional facts of each food separately.
5. Calculation of daily consumed micro and macronutrients. These are calculated per day, and also per meal, displayed on the main screen and on the history screen.
6. Graphically displaying how much calories left needs to be eaten, from the daily amount of total calories that should be consumed depending on user physical data and personal goal and the total calories consumed already.
7. Categorising meals into breakfast, brunch, lunch, snack, dinner and supper options.
8. Option to delete an already added food item from the list. It should also remove the items nutritional factors from the daily consumed nutrients and calories.
9. Logging daily water consumption, suggesting daily water intake amount, depending on user data.
10. Multiple options for adding foods to a meal: by taking a photo, by uploading a photo from the gallery or by entering the respective food in a searchbar.
11. Recommending an amount of calories that each meal should contain. (Dividing daily caloric need by meals logically.)
12. Displaying a profile picture for each user and option to change it.
13. A separate section for displaying the users personal information and account details.



## Chapter 2. Bibliographic Research

### 2.1. Study of nutritional analysis

#### 2.1.1. Macronutrients and Daily nutritional needs

Food components are classified in two categories: macronutrients, which are present in high percentage in foods and micronutrients which are present in lower percentages. Between macronutrients, we can enumerate Carbohydrates, Sugars, Proteins, Fibers and Fats. And the most common micronutrients are vitamins and minerals which are essential to the human body.

#### Energy consumption, calories

Energy is the main fuel that helps sustain the human body and its functions. Energy is supplied by various macronutrients in diet: carbohydrates, proteins, fats and even alcohol is a precious energy supply for the body. Required energy intake depends on individuals personal lifestyle (especially physical activity), and Basal Metabolic Rate (BMR).

#### Carbohydrates

Carbohydrates, shortly carbs, are our main source of energy. The minimum amount of carbs required per day is considered to be 50g/day. For pregnant women, the minimum amount increases to about 100g/day. Carbs play a major role in producing glucose, which is the essential energy source for the brain and the red blood cells. Since the daily required amount of glucose is about 180g, and the human body can produce from non-carbohydrate sources about 130 grams, the daily minimum external carbohydrate intake should be at least 50g. The majority of people consume more than the recommended daily amount, which helps keeping the body's energy levels. Studies have been conducted that at least 55% of daily total energy should come from carbohydrates.

Carbohydrates play an important role the lives of people with physical activity, in high-endurance lifestyle and in high-intensity workouts mainly. Having a carbohydrate rich meal before a high intensity workout enhances performance. Athletes consume carbohydrate and sugar rich energy drinks to help obtain higher performance [1].

#### Proteins

Proteins are nutrients needed by the human body to help growth, maintenance and repair. Some protein rich foods are nuts, dairy, eggs and meat, especially chicken breasts. It has an important role in active peoples lives as proteins contribute in muscle growth and maintenance. For an average adult male that has 70kg, about 16% of its

body weight will be from proteins, i.e. about 11 kg. From these 11 kilos, about 43% will consist of muscle, and the rest will be distributed in skin and in blood.

Protein intake also highly depends on the persons lifestyle and activity percent. For moderately active people, the recommended amount of protein consumption depends mostly on weight and it is 0.8 g/kg of body weight, for muscle maintenance. It is important to note that very high amount of protein doesn't offer additional benefits, only drawbacks like kidney failure and bone degeneration over time.

For more active people or for those who desire muscle growth, more protein is required per day. B. Egan's [2] study about *Protein intake for athletes and active adults* suggests, protein intake of people performing aerobic or resistance exercise should range from 1.2 - 2 g/kg of body weight. The protein dosage should be equally distributed between meals, 0.25 - 0.40 g/kg, in order to obtain maximum stimulation of muscle protein synthesis, repair, growth and adaptation.

## Sugars

Glucose and fructose the main types of sugar. These occur naturally in fruits and vegetables, and other sweet foods such as honey. People usually tend to consume too much sugars which leads to diabetes.

The daily recommended sugar intake for adults should be no more than 30g. For children it should be even less. People should be aware that sugar is not only labeled as sugar, but is present in other forms in foods like glucose, fructose, dextrose, lactose.

## Fibers

Fiber has two forms: dietary fiber and functional fiber. Dietary fiber is present mainly in plants, while Functional fiber has beneficial effects on the human body. Studies conducted that the daily adequate intake for adults is around 38 grams for man and 25 grams for women. This amount varies for pregnant women and children. Additional information about fiber intake can be found in [2.1.3](#) Nutrient calculations, formulas section.

## Fats

Fats are the food components that provide energy for the body, help absorbing the fat-soluble vitamins and provides essential fatty acids, but consuming too much fat can lead to obesity, high cholesterol, blood pressure irregularities and even diabetes, as consuming unhealthy fats leads to insulin resistance.

There are saturated fats, typically found in nuts, meat and dairy, and unsaturated fats, which are considered to be healthier and are present in olive oil, fish and nuts. The latter helps in maintaining healthy cholesterol levels.

## Nutrient intake levels

The graph from Figure 2.1 visually represents the relationship between nutrient intake levels and associated side effects and risks. EAR representing the Estimated Average Requirement, RDA representing the Recommended Dietary Allowance, while UL being the Tolerable Upper Intake Level. At intakes below EAR, the potential risk of inadequacy is quite high, higher than 0.5, which should be avoided. Everything under the RDA can still have a risk of inadequacy. The healthiest range, where the risk is minimal

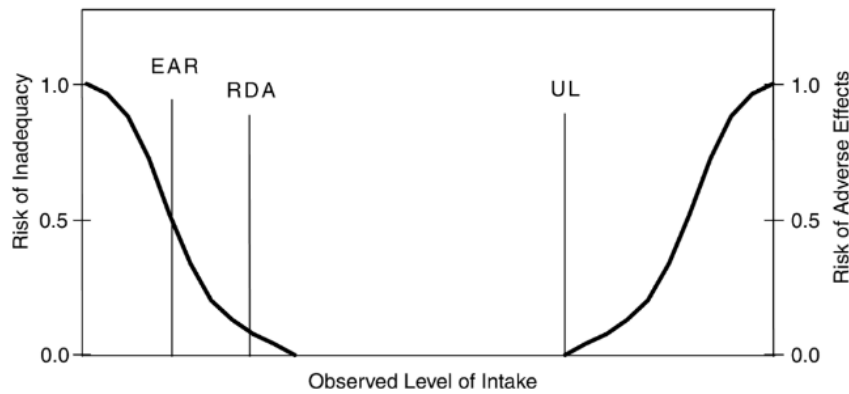


Figure 2.1: Risks and side effects of dietary inadequacy

is between the RDA and UL. Everything above UL is coming with adverse effects and may be dangerous [3].

### 2.1.2. Dietary plans

There are several dietary plans including very-low-calorie and low-calorie diets; low-fat and very-low-fat diets; moderate-fat, low-calorie diets; and low-carbohydrate, high-protein diets. Let's consider 4 diets that are relevant for the application.

### Weight gain

Kyle T. Ganson and Jason M. Nagata state valuable information in their journal *Weight gain attempts and diet modification efforts among adults* [4] about the importance of this phase in one's life, more and more people are making efforts to consume more calories on a daily basis. They found that people tend to consume additional fats as part of their weight gain journey besides following specific diets and modifying food intake. They also found that the 'bulking' cycle is a cyclical pattern of consuming a high amount of calorie dense foods for a certain period with the intention of muscle mass enchantment, may be beneficial.

People following a healthy weight gain journey should consider the following indications in mind, as recent research shows: [5]

- Weight gaining the most healthiest way means no plus kilograms overnight. People should gradually increase caloric consumption on a daily basis with around 300 to 500 extra calories per day.
- People should aim to eat smaller meals more often, and try adding healthy snacks such as fruits or non sugary biscuits between meals.
- Extra calories can be added to meals by using cheese, nuts and seeds, which are healthy and full of calories.
- High calorie content drinks (milkshakes) provide additional nutrition between meals. People should not replace a meal with a high caloric content drink.
- Having a balanced diet is the key to achieve a healthy weight. People should include in their meals healthy options from a variety of food groups: fruits and vegetables, starchy carbohydrates and dairy and alternatives.
- Having healthy and easy to prepare snacks, even on the go, such as yogurt.

- And should not rely on eating sweets and unhealthy snacks just for the sake of calories, as this can get you more drawbacks than goods.

### Weight loss

Weight loss is many people's goal and here are some key factors people should consider: [5]

- Trying to be active 30 minutes every day.
- Aiming to lose no more than 0.5 to 1 kg per week.
- Swapping sugary drinks with fresh water. Adding lemon is a great option for flavour.
- Cutting on foods with high sugar and fat content. Try swapping chocolate cereals with wholegrain alternatives.
- Sharing your goal with people you trust can help in maintaining your motivation

It not only matters what we eat, is equally as important is how much we eat of everything. The importance of portion size may matter more than what we eat, stated Barbara Baker in her journal [6]. Trying to replace all foods in our diet can be overwhelming. Taking smaller steps and enjoying our favorite foods, like chocolate, but in moderation is perfectly acceptable.

### High protein diet - weight gain and muscle growth

Building muscle with strength training is a good idea, as muscle has more weight than fat. Replacing fat with muscle is a longer process but definitely the best option.

A high protein diet is recommended for sportives and athletes. Protein should come from healthy protein sources which are: flesh of ruminants, poultry, fish, as well as eggs and dairy, legumes, nuts and seeds. Animal based protein sources are considered to be of higher quality and more effective than plant based ones, given its higher leucine content. A meal plan based on whole foods should be the base of a high protein diet [2].

### Weight loss and muscle growth

This diet can be a little more challenging as weight loss requires a caloric deficit and muscle growth imposes a caloric surplus for the muscle repair and growth. As Healthline's nutrition team states [7], the secret lies in proper nutrition and exercise. It is important to prioritize a healthy, lean diet, including adequate protein intake, healthy fats and nutrient dense food. For optimal muscle mass development, exercising is important. For weight loss and muscle growth it is recommended to combine both strength training and cardio workouts.

As the *Effects of Weight Loss on Lean Mass, Strength, Bone, and Aerobic Capacity* [8] study states, weight loss induced only by caloric deficit is not as effective as weight loss induced by exercise and caloric deficit. Exercising during a weight loss journey is important as it helps maintain or even develop muscle mass and increase aerobic capacity. Keeping the habit of exercising, definitely helps in avoiding regaining the weight, and also improves physical condition.

#### 2.1.3. Nutrient calculations, formulas

There is no exact amount for nutrient consumption, but a lot of studies and research have been conducted in order to determine an appropriate amount of nutrient

consumption. Caloric and nutrient needs differ for each individual, and depend on a lot of factors, such as geographic location, personal physique, activity level, personal lifestyle and goal.

According to *Standing Committee on the Scientific Evaluation of Dietary Reference Intakes and Subcommittee on Interpretation and Uses of Dietary Reference Intakes and Subcommittee on Upper Reference Levels of Nutrients and Panel on the Definition of Dietary Fiber and Panel on Macronutrients* [3] study, average reference intake values are presented in the following tables. In table 2.1, the total energy expenditure is presented in case of active people. For sedentary or moderately active lifestyles, the amount lowers.

Table 2.1: Reference intake values for energy

| Age         | Female        | Male          |
|-------------|---------------|---------------|
| 0-6 months  | 520 kcal/day  | 570 kcal/day  |
| 7-12 months | 676 kcal/day  | 743 kcal/day  |
| 1-2 years   | 992 kcal/day  | 1046 kcal/day |
| 3-8 years   | 1642 kcal/day | 1742 kcal/day |
| 9-13 years  | 2071 kcal/day | 2279 kcal/day |
| 14-18 years | 2368 kcal/day | 3152 kcal/day |
| 18+ years   | 2403 kcal/day | 3067 kcal/day |

In table 2.2 the reference intake for carbohydrates is presented, EAR being the estimated average requirement which meets the needs of 50% of people in a specific group. RDA being the recommended dietary allowance, the amount that should be consumed and that is enough for about 98% of population.

Table 2.2: Reference intake values for carbohydrates

| Age         | EAR       | RDA       |
|-------------|-----------|-----------|
| 0-6 months  | 60 g/day  | 60 g/day  |
| 7-12 months | 95 g/day  | 95 g/day  |
| 1-18 years  | 100 g/day | 130 g/day |
| 18+ years   | 135 g/day | 175 g/day |

Table 2.3 represents the daily total fiber consumption. Intake levels presented provide the greatest protection against coronary heart disease. We can observe that the highest need for fiber have younger women and middle aged man. For the total amount of fat in dietary reference there is no exact data that defines it, because the percent of energy consumed from fats can not be exactly defined.

Table 2.3: Reference intake values for total fiber

| Age         | Female      | Male     |
|-------------|-------------|----------|
| 1-3 years   | 19 g/day    | 19 g/day |
| 4-8 years   | 25 g/day    | 25 g/day |
| 9-13 years  | 26 g/day    | 31 g/day |
| 14-30 years | 25-26 g/day | 38 g/day |
| 31-50 years | 25 g/day    | 38 g/day |
| 51+ years   | 21 g/day    | 30 g/day |

## 2.2. State of art

In the following section a study of similar calorie counting and nutrition tracking mobile applications will be presented. The scope is to outline the benefits and drawbacks for each of them and to highlight why a mobile app like BiteBalance is needed. These mobile applications can be freely downloaded from Google Play or from the App Store.

While automatic food recognition is not an easy task, regarding the deformable shape of foods, there are already fully functioning applications that are able to do this, but only with a limited accuracy. A lot of food items do not have a predefined shape, especially the prepared ones, and different foods can look similar, while similar foods can look very different.

A comparative study on the accuracy of image recognition platforms for the recognition of food and beverages was conducted by S. Van Asbroeck and C. Matthys. It has resulted that after analysing different scenarios like bad lightning, cluttered space around food, non-standard container or unspecified angle. CalorieMama performed best, followed by Foodvisor and Bitesnap, with a total recognition percent of 63%, 49% and 46%. Clarifai and Logmeal performed noticeably well too [9].

### 2.2.1. CalorieMama

The app offers instant food recognition, powered by Food AI API. It implements deep learning and image classification strategies. It has a culturally diverse food identification method that is constantly improving. The app can be synced with Google Fit. Creates a custom calories plan, and gives users an estimate of when they are reaching their weight goal.

CalorieMama uses Food Recognition API, and gets the response in JSON format. It has macronutrients (calories, totalFat, totalCarbs, protein) and full nutritional information also (calories, totalFat, transFat, cholesterol, sodium, totalCarbs, saturatedFat, dietaryFiber, sugars, protein, monounsaturatedFat, polyunsaturatedFat, vitaminA, vitaminC, iron, potassium, calcium). The response is structured the following way: it has a main category, like Salad, and more items in it: Bacon, Bell pepper, etc. Each item contains the nutritional information, name, brand, serving size, unit and serving weight.

Basal calories burned (energy burned at rest) are calculated using The Mifflin St. Jeor Equation, which is different for men and for women. The equation for women is:

$$BMR = 10 \times \text{weight (kg)} + 6.25 \times \text{height (cm)} - 5 \times \text{age (y)} - 161$$

and for men:

$$BMR = 10 \times \text{weight (kg)} + 6.25 \times \text{height (cm)} - 5 \times \text{age (y)} + 5$$

The premium plan gives full access to detailed nutrition info, meal plans, cardio activity tracking and ad-free experience. Has the functionality to add a meal by taking a photo of the food, then selecting manually from more text options the best match from the photo [10].

### 2.2.2. Lifesum

The app has basic free options and also has a premium plan which offers additional options for users. The user has to pay the premium plan to view detailed micro-nutrients.

Amongst the free options, the app offers personal details customization, water consumption logging, meals categorisation and also suggests caloric needs. The user can adjust daily needed macronutrients and caloric needs only on premium plan. The landing page and the detailed macronutrients page can be seen in Figure 2.2 below.

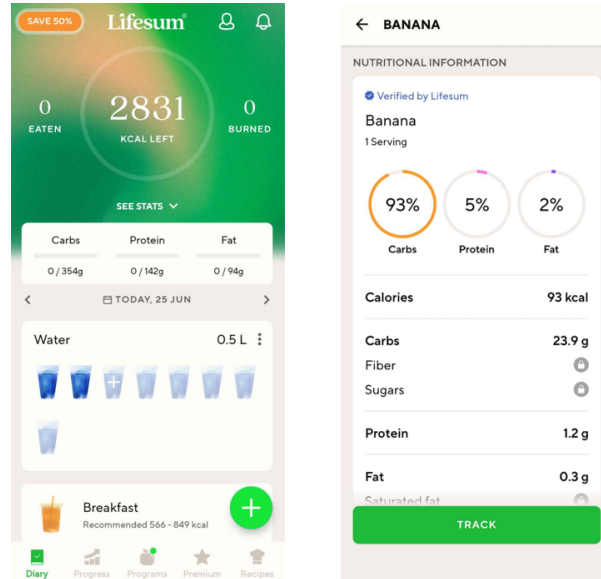


Figure 2.2: Lifesum landing and macronutrients page

The app also offers fruit tracker and vegetable tracker options, recommends intervals (30min) of exercise, it calculates eaten calories, burned calories based on exercise, and calories left to be eaten. It has a built-in intermittent fasting timer, barcode scanner and the user is able to set dietary needs and preferences (allergies to different foods like nuts, milk, eggs, vegan, vegetarian plans, lactose or gluten intolerance).

When adding something liquid, and orange juice for example, the user can select the desired amount from a list of available options: regular, large serving, bottle or by milliliters.

The app also offers different recipes according to the chosen lifestyle (balance, fasting, sugar detox, etc) but only available on the premium plan. Also offers the option to take a test available in the app and help determine the best suitable recipe recommendation for users needs [11].

### 2.2.3. MyNetDiary

MyNetDiary is similar to the above mentioned ones, it also has a Premium plan which offers daily feedback and advice, macros tracking, recipes and diet plans (Low carb, KETO, High-Protein, Low-Fat, Mediterranean, Vegetarian, Vegan).

On the free plan, it offers PDF reports, suggests menus with different calories and weight plans. There is a My foods section where you find a list of your favorites, frequent foods, recent meals, custom foods and my meals. It has the option to change water containers (half glass, full glass, bottle) depending on users personal preference.



In Figure 2.2 the Graphical User Interface can be seen. It shows the landing page of the application, from where the user can add consumed foods based on 4 categories (Breakfast, Lunch, Dinner and Snacks), and the user can view the macronutrients ratio consumed that day. When clicking on the Breakfast option, the user can see a list of already consumed foods, each food item is accompanied by a figure representing it. From there, the user can log a food by speech, by writing, or by scanning the barcode. When adding meal elements to a meal category, the app recommends popular foods for easier user experience.

On the last image from Figure 2.3 the users weight and caloric goals are illustrated. The app makes a plan for the user, on when he will reach his target weight and with how many calories per day [12].

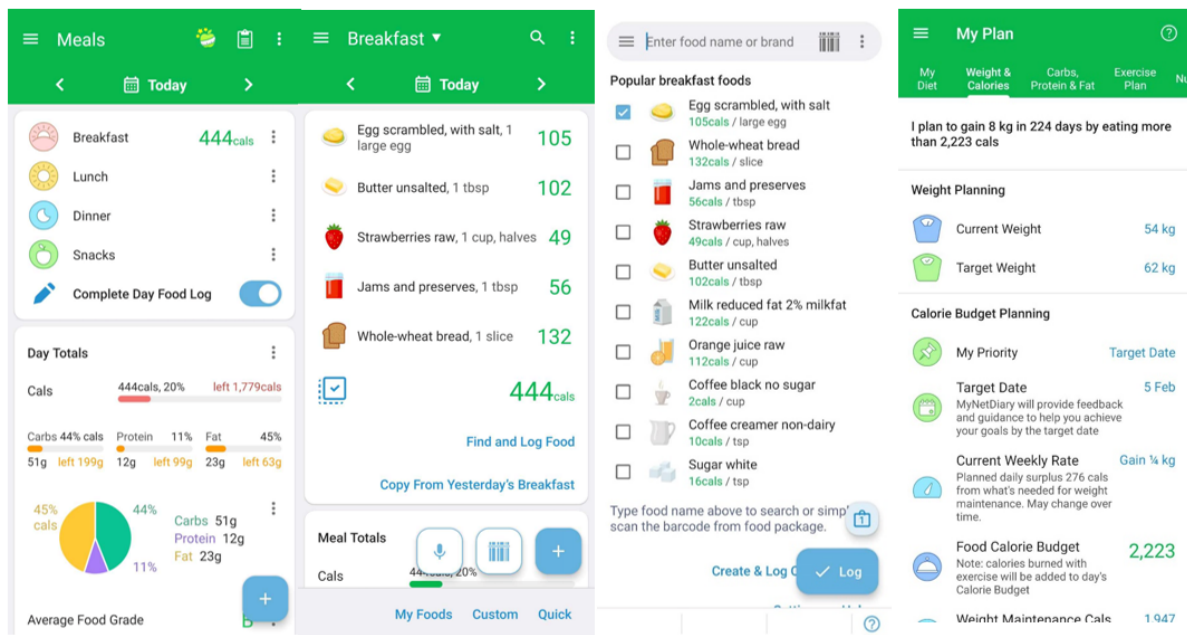


Figure 2.3: MyNetDiary Graphical User Interface

#### 2.2.4. Cronometer

Cronometer is similar to the above mentioned apps, the user can get detailed reports on: Calories consumed, Calories burned (from BMR and preset activity level), Caloric balance, Macronutrient targets (energy, protein, carbs, fat), Highlighted targets (vitamin C, fibers). The user can change the time of the report (1 day, 1 week etc). The charts section contains charts about weight progress and consumed calories.

The user can set the corresponding activity level (No activity, Sedentary, Lightly Active, Moderately Active, Very Active, Custom), and can set a weight goal. The application sends reminder notifications for users and has barcode scanner, alcohol measuring and day streak of logging food, which is good for motivation to use the app.

It also has a Gold plan that includes: no ads, set recurring foods, recipe importer, nutrition scores, fasting timer, custom charts and biometrics. The recipe importer is a useful functionality that no other application currently has, it helps users generate meals by importing a recipe from the internet as an URL and adds all of its ingredients and measurements as a custom recipe.



As a side-note, the only criterion for password is to be 12 characters long [13].

#### 2.2.5. SnapCalorie

SnapCalorie app is using AI technology to identify and categorise items from the photos. It has a unique dataset of 5000 images of meals which is used to train the deep learning model. The differentiating factor that SnapCalorie uses is the portion estimation technology by using depth sensors available on mobile devices. They claim that with the use of this technology of portion size estimation, caloric information error is reduced by 20

The app has photo logging available only for Premium members, as a free member, you get only one photo logging per day, which is obviously not enough for a motivated user. It has text and barcode logging for free. It has options to edit macro breakdown and caloric needs. The user can choose what he prefers to see on screen: calorie focus or macro-nutrient focus. It has a streak meter. Each day the user logs a food, a day is added to the streak, this way, it motivates users to log their food consumption every day. Email verification works by a code which you get on your email and have to enter in the app, not by clicking on the link from the email [14].

#### 2.2.6. Conclusions

After inspecting other applications, I find that there are already good and functional applications on food recognition and calorie estimation, however, I find that another application could integrate all the useful functionalities mentioned above, in a single app.

The apps already integrate barcode scanning, some of them have image recognition and also portion estimation technology, which is useful in providing more exact measurements and caloric recommendations. The apps focus on providing an insightful user experience by having more options to add foods, additionally they have premium plans which offer premium services for users willing to invest in their health and nutrition. These apps help keep users motivated by having the functionality of daily streaks and sending notifications around mealtime.

#### 2.2.7. BiteBalance

BiteBalance will integrate food recognition and calorie estimation from images and from text input too, to help enhance user experience. Additionally, it will have a detailed consumption history, as well as water tracking functionality. The water tracking functionality should also recommend the necessary amount of water for each user. It will calculate caloric intake and nutrient needs for each user personally from the users physical data and personal goal and preference regarding their diet. The app will have a section where nutritional facts will be graphically displayed such as calories left to be consumed and already consumed nutrients.

## Chapter 3. Analysis and Theoretical Foundation

### 3.1. Flutter Software Development Kit

The Flutter SDK is an open source framework created by Google that helps developers to simplify the building process of apps and create visually appealing applications. With Flutter you can create cross-platform applications from a single code-base. The application can run on Android, iOS, Web and as a Windows desktop application also.

Because Flutter is a widget based framework, every element on the screen consists of a widget and is wrapped around different widgets to enhance its visibility and to adjust its size, padding and other characteristics. Flutter offers a lot of visual elements such as text fields, buttons and containers. These UI elements are represented as widgets. It has animations available for the components on the screen, to enhance user experience with a visually stunning app.

It also offers a rigorous state management solution that helps developers easily manage the state of the application and modify the screen accordingly, to ensure that the application's different parts are synchronized. State management is the process of managing the appearing screens based on the user's actions and based on user input, to keep the app dynamic and responsive. So we need to adapt the application to be able to differentiate which screen to load next, and to ensure that the correct data is appearing on the screen.

By choosing Flutter as a development framework, you can easily integrate other services to your app offered by Google such as: Firebase, Google Play, Google Pay and more. Flutter uses the Dart programming language [15].

#### 3.1.1. Dart programming language

The Dart programming language is in active development, offered also by Google. It is an open-source object-oriented language with fast compilation. It is known for its portability as it can be compiled to native machine code and is supported by most platforms. It supports productive app development process, because it has the functionality of hot reload and hot restart which means you do not need to stop and restart your app to see a visual change in the code. It simplifies the development process.

It is a concise and safe programming language as it has null-safety mechanism (makes sure variables always have a non-null value), and synchronization (async-await) mechanisms, making it an event-driven language.

### 3.2. Firebase Software Development Kit

Firebase SDK is a powerful tool offered by Google to help create back-end solutions for applications. Used together with Flutter software development kit, it offers a wide

range of tools and services for software development, such as data storage, secure authentication options, and app hosting also, providing a full base for building an application's entire back-end. Firebase also offers a real-time No SQL database that synchronizes data across clients in real time [16].

### 3.2.1. Firebase Authentication

Firebase Authentication allows developers to simplify the authentication process in applications. It offers a readily available tool for managing user login and sign-up in different ways. It has secure credential transmission mechanisms as well as data encryption and hashing in order to protect user data and privacy.

It has the functionality of real-time authentication state management. The user makes changes such as logs in, logs out or resets the password and the application responds automatically. For example, if the user logs in, then closes the app and reopens it, the state of the app will not change, the user doesn't need to log in again, it stays logged in. Only after logging out, the application requires log in again.

By using Firebase authentication, developers can easily retrieve other data from the server related to the user, such as profile picture, account creation timestamp or last sign-in timestamp.

It supports various methods like email and password authentication, phone number authentication, Google authentication, Facebook authentication and a lot more other social platforms [17].

### 3.2.2. Cloud Firestore

Cloud Firestore is a service offered by Firebase. It is a NoSQL data storage service which is flexible and scalable for mobile applications, web applications and server development. Data is synced across all devices with the help of realtime listeners and it also offers offline support, the app connects to the database even offline, without an internet connection.

Given the NoSQL structure, data is structured in documents which are organized into collections. Each document has a key. Documents contain subcollections and nested objects, which can contain fields like integers, strings, or lists [18].

### 3.2.3. Cloud Storage - Realtime Database

Realtime database is also a data storage service, as the name suggests, but the difference between Realtime database and Cloud Firestore is that Cloud Firestore is recommended for more complex applications and organizations with rich data-model. It is highly scalable, while Realtime Database is more suitable for smaller applications with limited data storage needs and simple lookups.

While Cloud Firestore stores data as a collection of documents, Realtime database is structured as one large JSON tree [19].

### 3.2.4. Hosting

Besides the above mentioned services, Firebase also offers a powerful hosting platform. It is useful in deploying apps globally. It has zero-configuration SSL for security and fast content delivery mechanism [20].

### **3.3. JavaScript**

JavaScript is an interpreted programming language, which runs on the clients browser, mostly used in building powerful and visual web-pages with interactive real time controls.

As a key feature of JavaScript, I would mention that it is a dynamic language, that supports object construction at runtime, variable parameters list and function variables. It also supports multiple programming paradigms at once, such as object oriented programming, imperative and declarative programming styles.

It is well known for its client-side web development but it has front-end and back-end capabilities also. It is powerful also in server side scripting, but it needs additional frameworks such as Node.js and express.js which will be detailed below. We also use cors (Cross-Origin Resource Sharing) and mssql mechanisms to fasten the connection process to Microsoft SQL Server Management Studio and to enable effective resource loading systems and data transfers between the browser and the server.

In this paper, we will focus on elaborating its server-side back-end capabilities, as that part relates mostly to this project.

#### **3.3.1. Node.js**

Node.js is a tool that provides a platform for JavaScript server-side development and allows it to run outside of the browser. It can be used to make the connection between your application and the database, or different API-s.

It is known for its asynchronous and non-blocking capabilities, it doesn't wait for one operation to complete before moving on to the next, it handles more connections simultaneously. It is also scalable because of its capability to handle more concurrent connections efficiently, and single threaded, it creates only one process for handling all the requests, without creating new threads for each request.

#### **3.3.2. Express.js**

Express framework is known for its simplicity, scalability and flexibility. Developers can easily define endpoints and routes within their application. Additionally, it provides a strong middle ware framework making functionalities like authentication easier to develop and maintain.

### **3.4. Microsoft SQL Server Management Studio**

Microsoft SQL Server Management Studio is a powerful database management studio for managing databases locally or on the cloud. It helps developers configure, monitor and maintain SQL databases, to build queries and to deploy applications. Developers choose to use SSMS because of its robust performance, security and scalability.

It has an object explorer window, a template explorer window, visual database tools that help build queries, tables and diagrams, directly from the application.

By being a product of the Microsoft company, SSMS can easily be integrated with other Microsoft products such as Windows, Azure and .NET [21].

### 3.5. Artificial Intelligence

Artificial intelligence, firstly appeared in 1956, had many ups and downs during its development process. Roughly 60 years it was on standby due to lack of computational resources and hardware limitations such as storage and computing speed. In the recent years, as technology evolved in many areas, such as cloud computing and big data technologies, and we had the necessary resources, AI has reached its potential and has become a part of our daily lives.

Artificial Intelligence can be categorised into 3 main categories, regarding the learning task: supervised learning, unsupervised learning and reinforcement learning [22].

### 3.6. Machine Learning

Machine Learning is a specific area of Artificial Intelligence. As Batta Mahesh states in his journal *Machine Learning Algorithms - A Review* [23], Machine Learning is a scientific study of algorithms and statistical models that computers use to perform a task without being explicitly programmed. Machine learning algorithms are used for: image processing, language processing, predictive analytics and data mining.

With the newly available huge datasets, the demand for machine learning is in rise. The purpose of machine learning is to learn from available datasets, then be able to recognise what he learnt on newly available data. Algorithms learn some kind of pattern from the input dataset, then try to apply the pattern for prediction and classification.

There is no algorithm that can be named best. For different problems there are different algorithms that fit best that specific problem. The difference between supervised learning and unsupervised learning is that in case of supervised learning, there is a bid training labelled dataset consisting of training data and validation data, and results can be either correct or incorrect, while in case of unsupervised learning, algorithms are left to their own to discover the structure of the data. It gets an input raw data, it interprets it, applies a learning algorithm, processes it and from there the output results. There is no training or validation dataset, the algorithm learns on his own from input raw data.

There are a lot of algorithm for ML, such as Decision Tree, Support Vector Machine, Principal Component Analysis, K-Means Clustering, Reinforcement Learning, Neural Networks and more. NN try to learn the way the human brain learns, they can adapt to changing inputs, the network generates the best possible result without redesigning the network structure [23].

### 3.7. Image Recognition

Image Recognition is a specific application of Machine Learning. As authors Zhang Jie, Li Shu Liang and Zhou Xi Liu stated on the International Conference on Computer Vision in their paper Application and analysis of image recognition technology based on Artificial Intelligence in 2020, image recognition can be defined as the technology that uses computers to process, analyze and understand and image, to identify different patterns and objects. It is a practical application of the deep learning algorithm [22].

In paper *Food image recognition with convolutional neural networks* [24] image recognition is categorised in two main methods.

The first one is the conventional method of including image pre-processing, feature extraction, feature selection and classification. An example is the BoF bag-of-features model proposed by Marios M., by computing dense local features and classifying the food images with a linear support vector machine(SVM) classifier. This model achieved a 78% classification accuracy [24].

The second image recognition method is the deep learning method which gained more popularity these days. This method uses Conventional Neural Networks. An example is Alex Krizhevsky's deep CNN (DCNN), which is trained through ImageNet in 2010, with an accuracy of 84.7% [24].

### 3.8. Transfer Learning

Transfer Learning is an improved machine learning method that can use the knowledge that already has been learned to solve given problems. An improved image recognition rate can be achieved using transfer learning: the CNN should be pre-trained with large image datasets, and only after to be trained with the specific dataset that needs to be classified, as we can observe in Figure 3.1 from *Towards Data Science* [25] source.

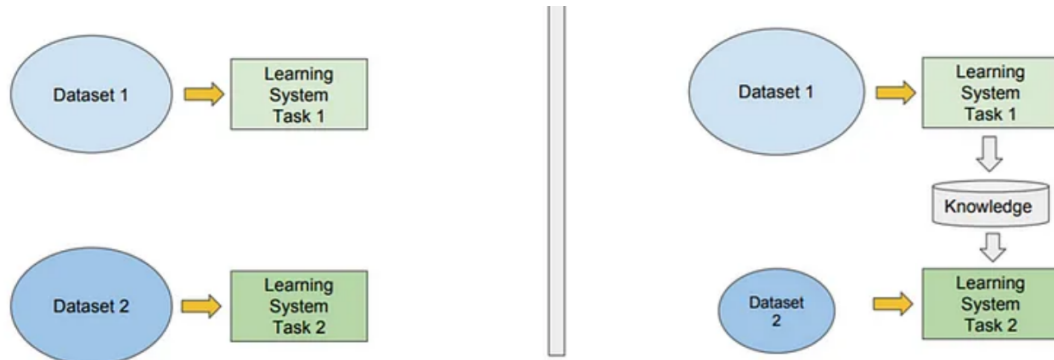


Figure 3.1: Traditional learning vs Transfer learning

As Torrey states in his *Transfer learning* [26] study, transfer learning attempts to improve on traditional machine learning by transferring knowledge learned in one or more tasks and using it to improve learning in a new task. Negative transfer is when using a transfer method decreases the performance of the learning task. The challenge we have to face is to produce positive transfer between tasks. It is better to use transfer learning in tasks that are appropriate, this way, positive transfer will more likely occur than negative transfer.

### 3.9. Convolutional Neural Networks

Convolutional Neural Networks have been used in many visual pattern recognition systems and more: speech recognition, face recognition, natural language processing and so on. A CNN has a multi-layer structure and can be divided in 3 main layers: the input layer, the hidden layer and the output layer. The input layer is the unprocessed image, the output layer is the result of the processing, while on the hidden layer, the image recognition happens and can have a complex multi-layer nonlinear structure, typically containing convolutional layers. The role of the convolutional layer is to convolve the

input image and the filter, to suppress noise and to enhance the original image. Another important hidden layer is the active layer which helps when solving more complex systems, by adding nonlinear factors into the network to help the network adapt more easily to complex problems [27].

CNN has three main features relative to classical neural networks: local receptive field, weight sharing and pooling. A local receptive field is the small area of the input layer, connected to each neuron in the CNN. Each connection between these two has an offset and a parameter weight. Weight sharing is the fact that in a local receptive field, each neuron shares the same weight, with the scope of reducing the amount of network parameters and training time. The pooling layer's purpose is to reduce the amount of parameters by compressing the image. This layer is typically behind the convolution layer.

The optimal selection of the CNN-s parameters is crucial at the first steps to ensure the optimal network training. The selection must be good to ensure that the network is not under-fitted or over-fitted. By making more and more tests, the parameters can be modified to get the best scenario and to improve the accuracy of the recognition [27].

### 3.10. TensorFlow

TensorFlow is an open source platform for machine learning development released by Google. It has a comprehensive, flexible set of tools, libraries and resources, making it a good choice for various tasks: image processing, speech recognition, natural language processing and even predictive modeling. It has limited performance relative to other machine learning tools but it is constantly improving with new releases of the library. It allows developers to easily build and deploy ML-powered applications. It is a flexible platform, running on multiple platforms and hardware configurations, these include Linux, macOS, Windows, Android and iOS.

It can be used in a Python environment. It has a low-level programming interface, it is able to perform fast gradient computation and it has a detailed control over neural network construction. Other machine learning tools include OpenCV, Torch, these were between the first ones, developed around 2000-2002, and Accord.NET, Pattern, Caffe and others, the newer versions.

In TensorFlow, ML algorithms are represented in the form of computational graphs. Each graph is a directed graph with nodes and edges, where each node represents an operation and each edge represents data flow between operations. An operation can represent a mathematical equation, a variable or constant, a control flow directive, a file I/O operation or a network communication port.

Tensors are the data flow between operations, represented by the edges of the graph. Mathematically it looks like a one or two dimensional matrix. The number of dimensions of a tensor is called rank, the number of components in each dimension is called the shape of the tuple. SparseTensors are more space-efficient data structures [28].

### 3.11. Keras

Keras is a Machine Learning library that provides powerful and abstract building blocks that help building a deep learning network [29]. Keras was built using the TensorFlow library and supports both CPU and GPU computation.



## Chapter 4. Detailed Design and Implementation

### 4.1. Application's architecture

The detailed architecture of the BiteBalance application can be seen in Figure 4.1: Application's architecture. The figure highlights the main components and softwares used in developing the app. The user opens the application and enters their credentials. The credentials are verified by the Firebase Authenticator service. It manages user login, sign-up and forgot password options.

After the user is logged in, their data are retrieved by the server. The app makes a request to the server, the server is connected to the database, retrieves required information about specific user from the database, then returns the data to the Flutter app. When the user logs a food, the app accesses CalorieNinjas API, retrieves caloric and nutritional info about the requested food, then returns it to the app. When the user takes a picture or uploads a picture about a food, the application uses the Keras TensorFlow model to identify the food on the photo, then accesses CalorieNinjas API the same way as before, to retrieve caloric and nutritional information, then returns it to the app.

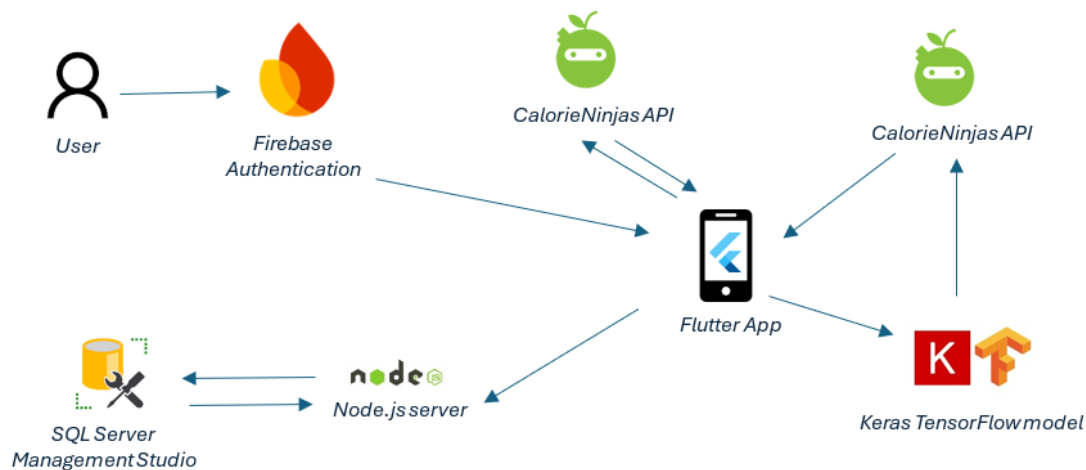


Figure 4.1: Application's architecture

The application is designed to run on Mobile devices (Android and iOS), but it will be able to run as a Windows Desktop application or on Web Browsers also. It is specifically optimised for an Android device.

In Figure 4.2 the use case diagram is presented. The use case diagram highlights the features of the application, and the options of the user from each state.



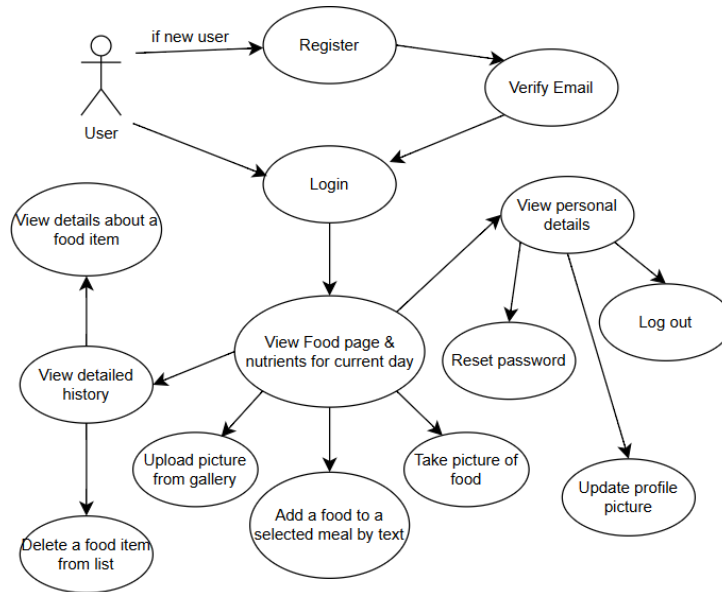


Figure 4.2: Use Case Diagram

The sequence diagram represented in Figure 4.3 illustrates how the components of the system work together to achieve the end result.

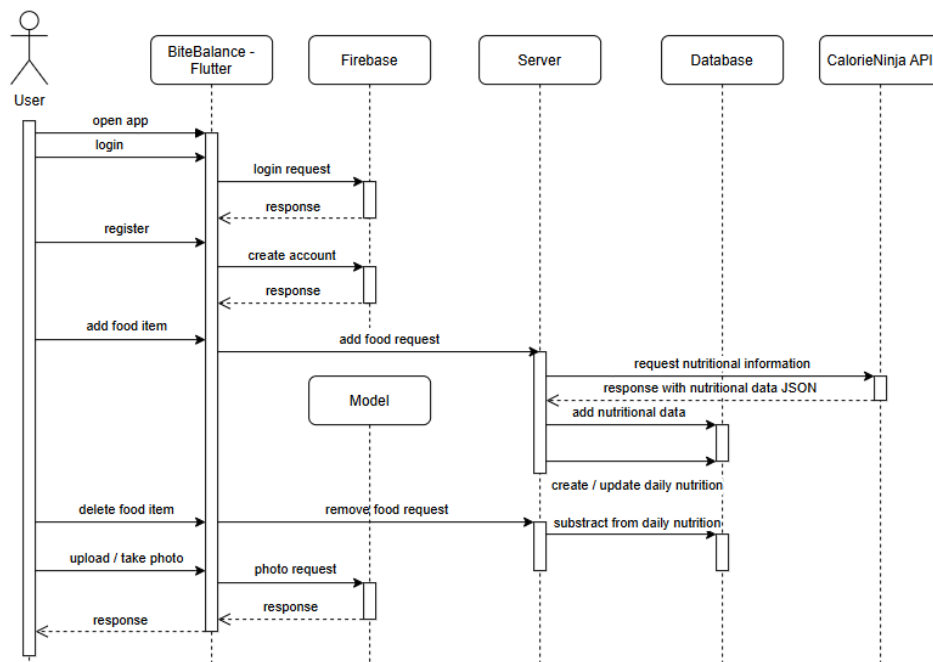



Figure 4.3: Sequence Diagram

## 4.2. Application Design

The front-end is designed to be user friendly, familiar and colorful. On the login page, there are options for the user to log in with entering email and password, to access a forgot password screen where he can enter his email and get a link to change the password, option to log in with already existing google account, or the option to create a new user account. Login and register pages can be seen in Figure 4.4.


After the user creates an account, an email is sent to the email address, with a verification link. Only after the link is verified, the user can continue to log in to the application. Until then, a screen is visible with a simple text message: 'Please verify your email address. And a button to resend the email. From this screen the user also has the option to continue with another account by pressing the respective button.

### Log In




Forgot password?

Login


log in with Google

Don't have an account?  
Register

### Register Account



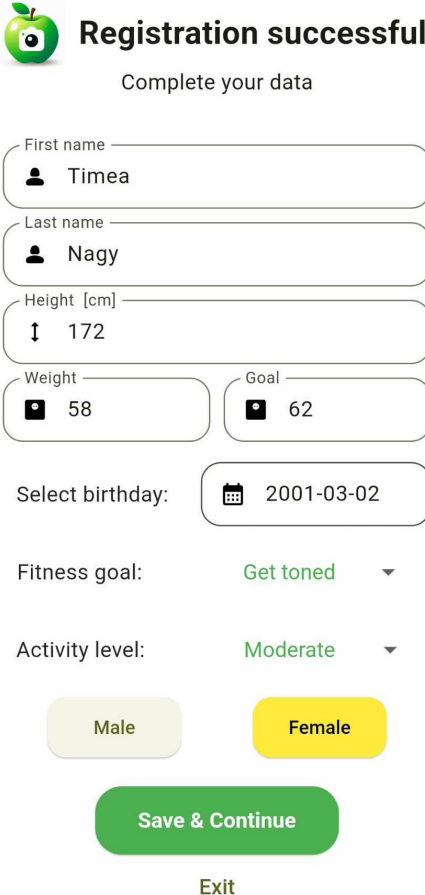
Register

Already have an account?  
Login

Figure 4.4: Login and register screens

In Figure 4.5 the complete data screen is presented where the user completes additional information after creating an account.

After the user logs in, or creates a new account and then logs in, the main screen appears, which is visible in Figure 4.6. On the main page the user has multiple possibilities. There is the food section, which is expandable. The user can view the total calories that he can consume for the day, there is a visual representation of the ratio of how many calories he already consumed and calories left. The user can see a detailed nutritional analysis of the consumed nutrients for the day, when he expands the food section. Each



**Registration successful!**

Complete your data

First name: Timea

Last name: Nagy

Height [cm]: 172

Weight: 58

Goal: 62

Select birthday: 2001-03-02

Fitness goal: Get toned

Activity level: Moderate

Male Female

Save & Continue

Exit

Figure 4.5: Complete data screen

nutrient item is visually displayed, according to the ratio of consumed/recommended to be consumed grams. It is highlighted with yellow, green, or red. Yellow represents the beginning of the interval, green represents the end of the interval and the full interval, and red represents overconsumption.

There are 3 icons that help the users add food for the day. The first icon is a camera icon, from where the user can take a picture of the food, the app will recognise the food from the photo and add it to the database, together with the food's caloric and nutritional information. The second option is to add a photo from the gallery. The app proceeds the same after analysing the food from the photo. The third option is the log manually button which is a popup window, from where the user can add a food as text. The user selects the desired meal (breakfast, brunch, lunch, snack, dinner, supper) and inputs the text, for example: two bananas or sandwich. The app adds the respective food and its nutritional data to the database.

On this screen is also the view for the water consumption. The user can add either a glass of water (250 ml) or a bottle of water (500 ml).

On Figure 4.6 the profile section is also visible. In this section, information about the user is displayed. Data is separated into account details and personal details. Under account details section, there is the birthday, the gender and the email address and under personal details section, further data is displayed about the user's physique (personal goal, activity level) which is used later in calculating caloric needs for the user.

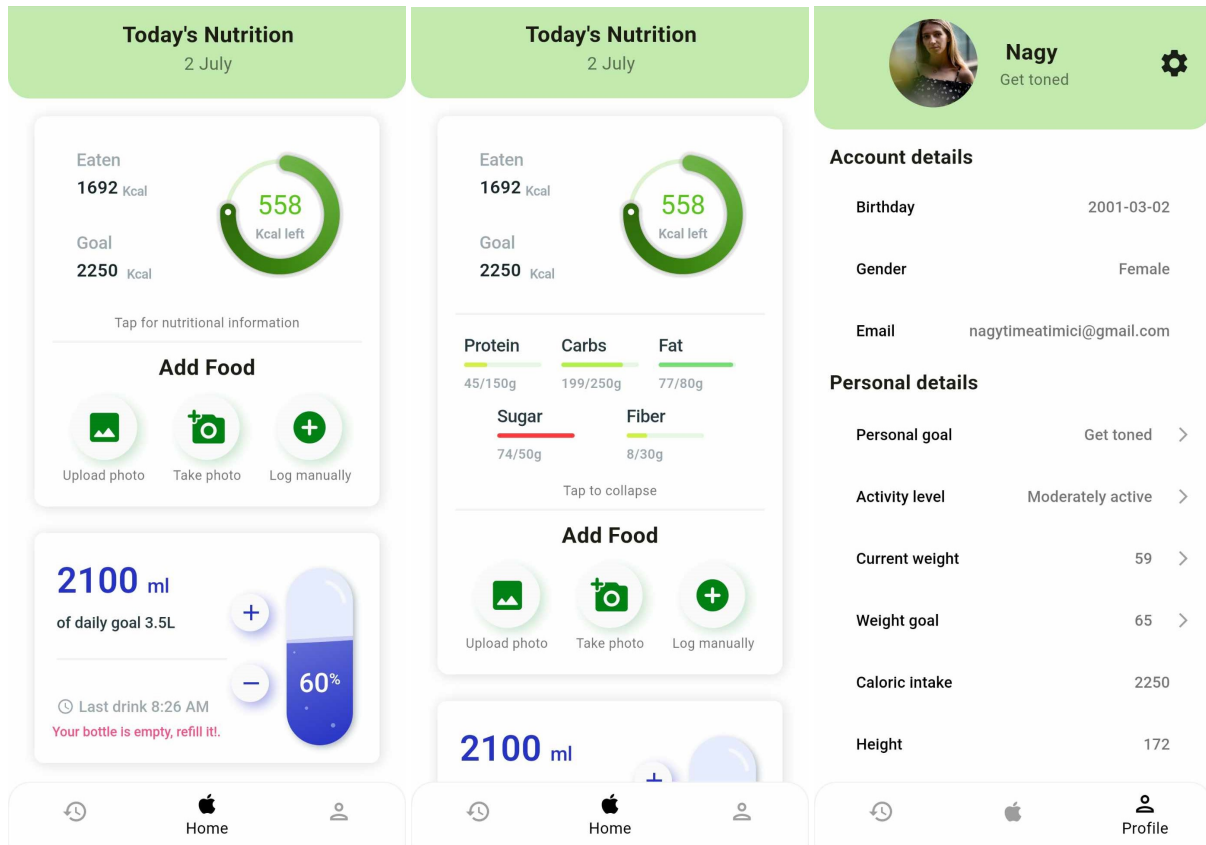


Figure 4.6: Login and register screens

On this screen, under the setting button, 3 option are available: changing the profile picture, changing the password and logging out.

On Figure 4.7 the History section is visible. This section provides detailed information about consumed food items. It is structured into 6 meals (breakfast, brunch, lunch, snack, dinner, supper) and each meal contains a list of the respective food items that has been consumed. Each meal item has its caloric recommendation and actual caloric consumption displayed. Caloric recommendation is only a suggestion from the app, on how much calories is recommended to be consumed during the respective meal.

When a meal is clicked on, it expands and the user can see a list of food items with its caloric info. When further expanding the food item, there is more information available regarding the portion size, carbohydrates, proteins, sugars, fats and fiber composition of the food.

On the top of the screen the current date appears, and two arrows. With the use of the arrows the user can navigate between days and view the food consumption for the past days.

When clicking on the 'trash' button, the user is able to delete a food element, which he no longer desires to be in the list. The deletion of a food from the history screen also deletes the foods nutritional data from today's consumption (which is displayed on the main screen).

On Figure 4.8 account verification email and password reset email are presented.

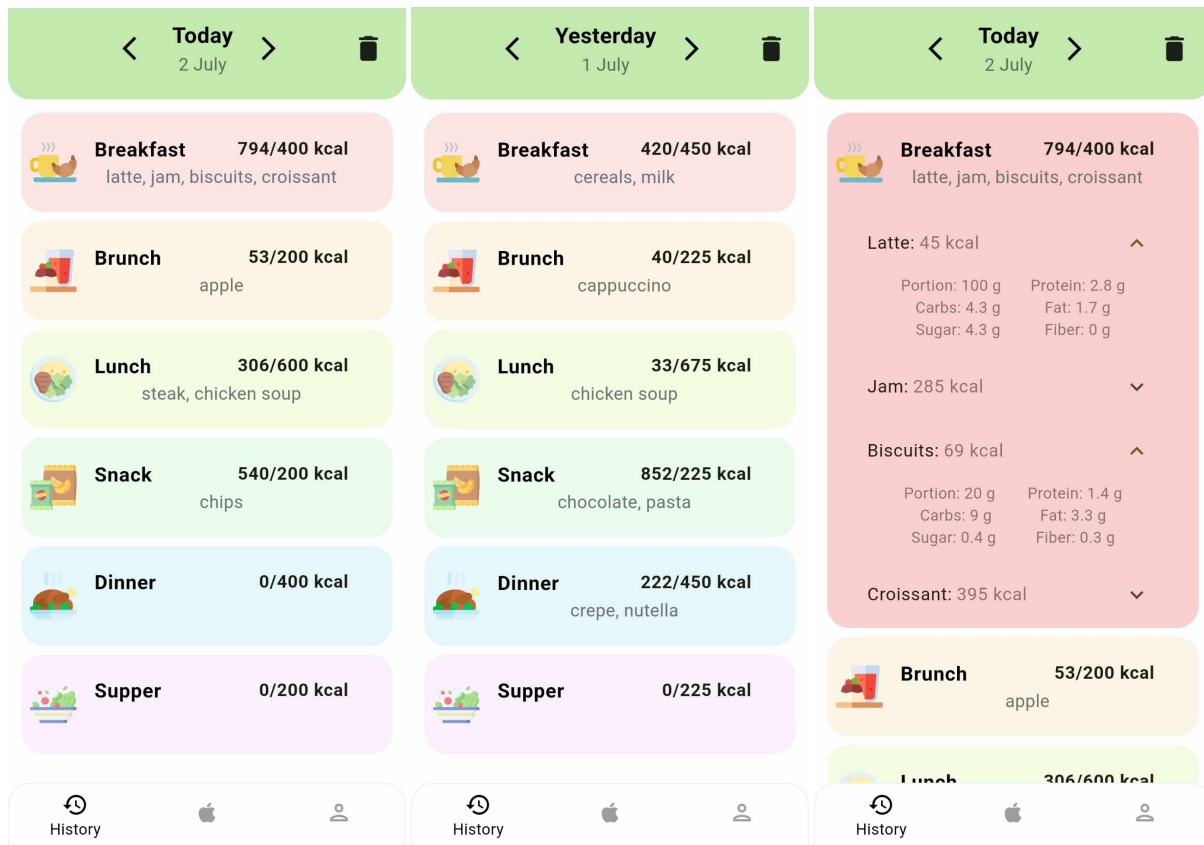


Figure 4.7: History screen

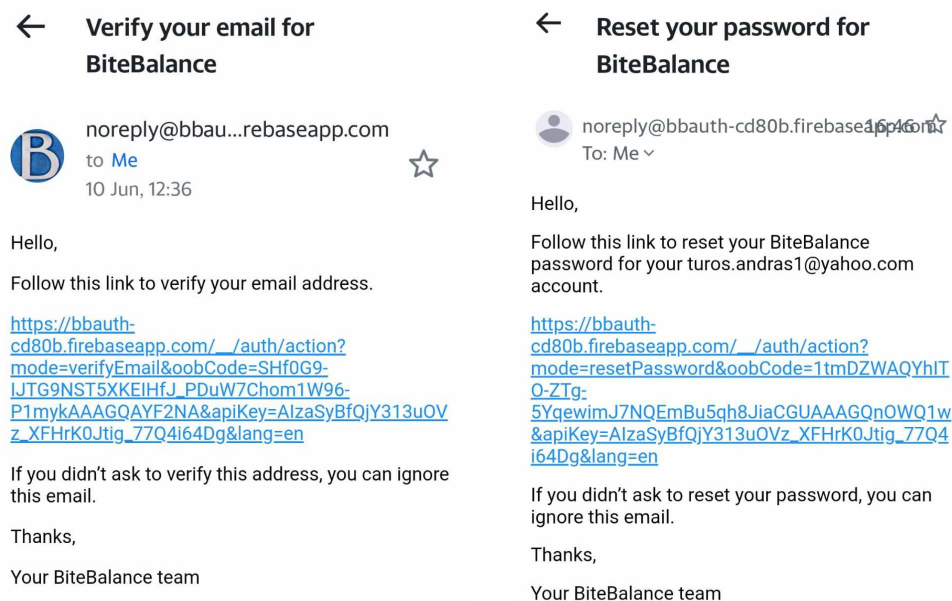


Figure 4.8: Account verification email and password reset email

### 4.3. Implementation details

#### 4.3.1. Image recognition model - Python

The image recognition model is implemented in Python. I downloaded an already trained neural network model, MobileNetV2 which is directly available from python, and further trained it for food recognition with the food 101 dataset. The food 101 dataset is a large dataset consisting of 101 types of food, each type of food having 1000 images, and can be found on Kaggle [\[30\]](#). Each image is similar to a user taken one, it may not be exactly in the middle and has disturbance elements such as kitchenware, hands, table, etc. The foods in the dataset are complex foods like pizza, and other full meals.

Let's break down the steps of training an image recognition model:

##### 1. Image preprocessing

All images from the food 101 dataset go through a preprocessing stage, which involves rescaling the images into smaller size to reduce training time. The dataset is separated into a training set (80% of data) and validation set (20% of data).

##### 2. Labelling

We have a labels.txt file that contains all 101 food names. This file is generated through a python code, from the class names.

##### 3. Image recognition base model

The model which I used for training is the MobileNetV2 model from the tensorflow library. I took batch size as 64, this represents the number of images I inputted into the neural network at once. The neural network is built based on the transfer learning method. As base model, we took the MobileNetV2 model from the keras library, and freeze all the weights, so that we kept the already existing features of the model and added our training.

```
model_base = tf.keras.applications.MobileNetV2(  
    input_shape = (IMAGE_SIZE, IMAGE_SIZE, 3),  
    include_top = False,  
    weights = 'imagenet'  
)
```

#### 4. Neural Network Architecture

We added a few layers to the Neural Network.

- Convolution layer, that tries to understand the patterns of the image
- Dropout layer, a layer that prevents the network from overfitting
- Pooling layer that reduces the data significantly, to prepare the network for the final dense layer
- Dense layer, which predicts the class probabilities

The layers are added as in the next code presents:

```

myModel = tf.keras.Sequential([
    base_model,
    tf.keras.layers.Conv2D(32,3, activation = 'relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(101, activation = 'softmax')
])

```

In the above code, *relu*, Rectified Linear Unit is an activation function that helps overcome the vanishing gradient in problem and helps the model perform better by learning faster.

The dropout regularization is set to 0.2, which means that during training, 20% of the input units are set to zero. This is done to prevent overfitting by reducing the reliance between the neurons.

The *softmax* activation method is used to ensure that the sum of probabilities will be exactly 1.

## 5. Optimising & training the model

The model is optimised with the Keras Adam optimiser designed specifically for the training of neural networks. When training the model, the epoch number is set to 2. I found that this is the lowest optimal number that still produces a high order accuracy model. It represents how many times the training process goes through the images. In the following code the training code is observed and in Figure 4.9 the output of the training process is represented.

```

hist = model.fit(
    train_gen,
    epochs = 2,
    validation_data = val_gen
)

```

```

Epoch 1/2
1263/1263 ————— 2918s 2s/step - accuracy: 0.2619 - loss: 3.1315 - val_accuracy: 0.3858 - val_loss: 2.4761
Epoch 2/2
1263/1263 ————— 2606s 2s/step - accuracy: 0.4009 - loss: 2.3597 - val_accuracy: 0.4053 - val_loss: 2.3751

```

Figure 4.9: Output of training process in Python

## 6. Converting the model

The model is converted to a tflite format compatible with Flutter development kit, that is lightweight for a mobile application.

### 4.3.2. Front-End - Flutter

#### Configuring Flutter project

The front-end is implemented using Flutter SDK. The application can be run on emulators or on a physical device connected to the computer. The application was tested in both cases but during the development process a physical device was mainly used.

The first step is to configure the coding environment. For developing this project, Visual Studio Code was used as IDE. The general steps of configuring a Flutter project compatible and connected to Firebase are presented below:

1. Installing Dart and Flutter extensions in VS Code.
2. Creating a flutter project by running the following command:

```
flutter create project_name
```

3. For running a Flutter project on a physical device, Android Studio also must be downloaded and configured.
4. Installing needed dependencies and libraries. The dependencies must have consistent versions, as one can depend on another. For developing this project the following dependencies and versions were used:

```
flutter:  
  sdk: flutter  
http: ^1.2.1  
flutter_launcher_icons: any  
english_words: ^4.0.0  
flutter_svg: ^2.0.9  
shared_preferences: ^2.0.8  
provider: ^6.1.2  
image_picker: ^1.1.1  
path_provider: ^2.0.15  
firebase_core: ^3.1.0  
firebase_auth: ^5.1.0  
google_sign_in: ^6.2.1  
cloud_firestore: ^5.0.1  
tflite: ^1.1.2  
tflite_flutter: ^0.9.5  
tflite_flutter_helper: ^0.2.0
```

5. Importing needed packages as first lines of code in files.
6. Creating a Firebase project, and setting it up for the created Flutter project. The setup process includes adding a google-services.json file to the Flutter project (android/app folder), which is available in the newly created Firebase project, for connecting the two projects. The json files contains information about the Firebase project.



## Front-End Development

For developing the front end, the official Flutter development page [31] was consulted, where official widgets, different views and templates are available for building a Flutter application.

The user interface's water management part and the display of eaten calories, caloric goal and the visual implementation of calories left was designed with the help of a template available on GitHub [32]. The template was modified and reconfigured according to the specific needs of BiteBalance application.

For code maintainability, every different code section is written in different files. The files are structured in different folders: login, food, profile, UI. In the UI section, different templates are made that are used throughout the app, for example a custom dialog, a food element or a meal element.

In food folder, there are files that handle food management through the app. The main screen is named `food_screen.dart`, and there are additional smaller views that are displayed on this screen, such as `add_food_view.dart`, `consumed_kcal_view.dart`, `water_view.dart` and `wave_view.dart`. These all help in building different sections of the user interface.

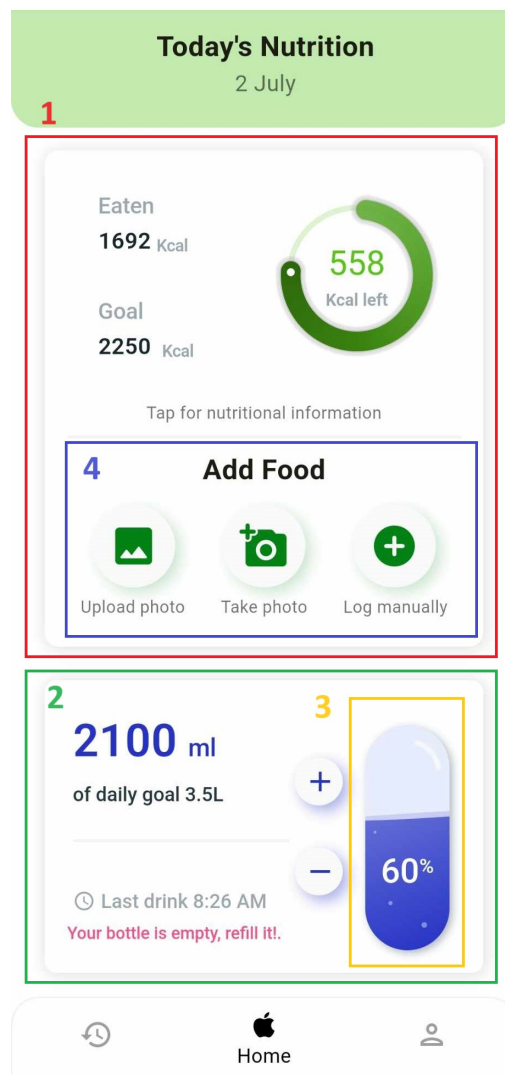


Figure 4.10: Main screen components

As we can observe in Figure 4.10, the first section highlighted with red is built with `consumed_kcal_view.dart`, and it contains the fourth section highlighted with blue which is coded in file `add_food_view.dart`. The water section is highlighted with green and is present in code file `water_view.dart`. It contains the `wave_view.dart` section highlighted with yellow which handles the illustration of the flowin water container.

Dependencies *tfite*, *tfite\_flutter* and *tfite\_flutter\_helper* are used to add the created model that recognises food to the app. We use *firebase\_core*, *firebase\_auth*, *google\_sign\_in*, and *cloud\_firestore* packages to connect the app to Firebase service.

### Constructing a page

As it was mentioned before in the theoretical part, in Flutter, everything is a widget. Every element is wrapped around different widgets to style it. In the app state management was used. A class needs to extend a `Stateful Widget` in order to be instantly modified on the screen, when something in the back changes or when another element is clicked and that has an effect on the UI, without reloading the page. Pages that need to be changed are built with `StreamBuilder` function.

The application has a bottom navigation bar with 3 elements, which means 3 pages are available through the app.

As an example, let's detailate how the login page is constructed.

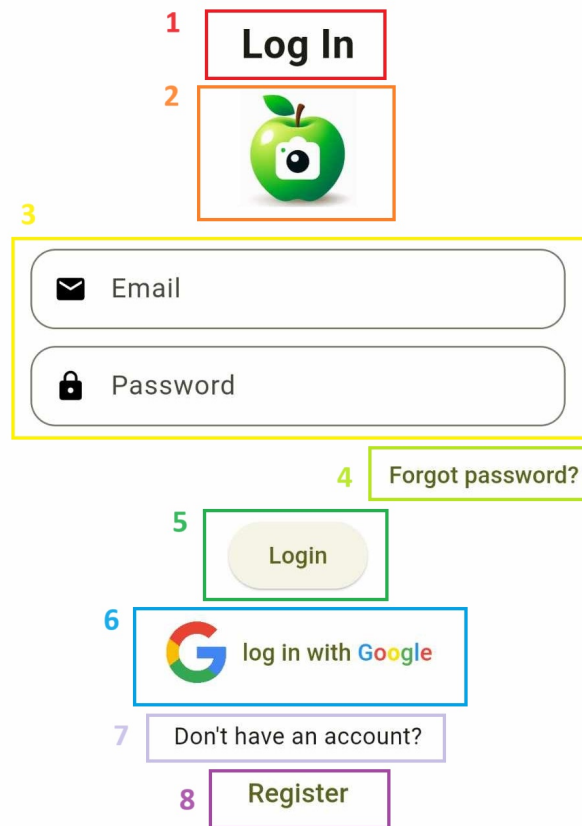


Figure 4.11: Login page elements

In Figure 4.11 the visual elements are highlighted. The class name which is used to build the screen is `LoginScreen` and it extends a `StatelessWidget`. It is located in `login_screen.dart` file. It has as an attribute the `_auth` Authentication Service, which handles user authentication methods. Other attributes are the `emailController` and the `passwordController` that extract text from the visual input fields from the screen and transmit it towards backend to process the information. The main widget is the `build` widget which returns a `Scaffold` elements that is the container for the page. It is aligned to center with a `Center` element and the fields are arranged in a `Column` element. On top of the page, a 'Log In' text is displayed with the help of a `Text` widget, highlighted with red and nr. 1 on the figure, and the logo of the application with the help of an `Image Asset` widget, highlighted with orange and nr. 2 on the figure. The elements are spaced with the help of a `SizedBox` widget. For code reusing purposes, an `InputField` element was defined in the UI folder of the project. It is reused in the register screen also. It has a controller element, a main text element, an information text element that is displayed when no text is entered in the field, it also has an icon element and an obscure text element which can be true or false depending on the text entered. If the text will be a password, the obscure element will be true and the text will not be displayed when typed into the field, only with dots. With the help of this `InputField` element, the components email and password input fields were defined. In Figure 4.11, they are highlighted with yellow and nr. 3.

After defining two email and password input field components, a small text element will be defined that has the role of a button. It can be clicked if the user forgets its password, and redirects the user to a page from where the user can enter its email address and will get a password reset link to its email. On the figure is highlighted with light green and nr. 4.

Next, a login button (green, nr. 4) is defined with the help of an `ElevatedButton` widget. It uses the `loginUser` function, which awaits for the `loginUserWithEmailAndPassword` method to complete from the Authentication Service, with parameters extracted from email and password controllers. After the method completes it return a user variable. If the variable is null, the user was not logged in and a dialog with an appropriate message is shown. If the user was logged in, it is redirected to the home page of the application.

The next element on the screen is the Google login button, highlighted with blue and numbered as 6. This button uses the `loginWithGoogle()` method defined in the Authentication Service file. In case of logging in with google, we need to verify if the user is a first time user or a returning user. If the user is a first time user, it is redirected to the complete data screen, else, to the main screen. The button is composed of an image with google logo, and 7 other text elements, from which 1 contains the text *log in with*, and the other 6 represent one letter from Google, each coloured differently.

The next element on screen is a text element highlighted with number 7 and lilac on the figure, it is a simple text component that indicated that the user can Register, if it doesn't have an account.

The last element consists of a Register button built with a simple `TextButton` widget, which is highlighted with purple and number 8 on the figure. It redirects the user to the register page.

Other pages are constructed similarly, with the same logic, by building it with appropriate widgets.

### Structure of main.dart

The main.dart file is the entry point of the application. It is an asynchronous function, handling asynchronous operations, such as await, mainly used when waiting for a response from a request. It has the following lines in the beginning:

```
WidgetsFlutterBinding.ensureInitialized();  
await Firebase.initializeApp();
```

These lines of code ensure that the application is properly initialised. The first line ensures that the flutter widgets are initialised, the second line ensures that the Firebase container project is properly set up and running.

The app then runs the main widget which is MyApp. It has a MaterialApp return type. This widget acts as a container for the following widgets that will be defined. The theme colours of the app are set, the home page is specified and the routes are handled. Routes map route names to screen widgets. Each route corresponds to a specific screen.

The logic of the MainPage widget is build with a StreamBuilder depending on the state of the user. If there is no data about a user being logged in, the LoginScreen is returned. If the email of the user is not verified, the VerificationScreen is returned, until the user verifies the email address from the mailbox. If the user is logged in successfully, The HomePage is returned and if the connection is in the waiting state, a circular progress indicator will be shown.

The structure of the HomePage is the following: it is a LayoutBuilder having a bottom navigation bar. The bottom navigation bar has 3 options: HistoryScreen, FoodScreen and ProfileScreen. The navigation between these screens is managed here.

### Implementing image recognition model in Flutter

The logic of implementation of the image recognition model can be found in the add\_food\_view.dart file. It uses the tfmlite flutter helper and tfmlife flutter packages. Tflite is the type of the model that is supported by Flutter for developing mobile applications.

There is a classifyImage function that gets an image as input and processes it by running it through an interpreter. The interpreter returns the response as a matrix format (tensor), with the classification scores of each item. The output is then processed and the most appropriate label is selected from the list, that matches most the image. The predicted class will be the class with the highest probability score.

Other relevant functions used are: loadLabels(), loadModel() and preprocessImage(). As the name suggests, the first two functions load the model and the labels, and the last one handles preprocessing the image which consists of decoding the image, resizing it with a bilinear method, normalizing the image and converting it into a TensorImage format accepted by the model.

### 4.3.3. Back-End

The back-end part consists of the server that connects and retrieves data from the local database, and also handles API connections. It is implemented in JavaScript with node.js and express. The back-end also has a part that handles user authentication, that is implemented in Firebase & Flutter.

The user authentication was chosen to be implemented separately from the rest of the back-end for security reasons. Flutter & Firebase already provide a secure authentication system that works well for deployed applications, providing token-based authentication, secure data storage and session management logics, that can be easily implemented. Firebase follows industry standards for secure authentication, including encryption, token validation and secure data storage.

### Calculating nutritional information and caloric needs from user data

The calculation of caloric needs and nutritional information is based on the following factors: age, height, current weight, weight goal, gender, fitness goal and activity level. Activity level ranges from 1 to 5. (1 - Sedentary, 2 - Lightly active, 3 - Moderately active, 4 - Active, 5 - High intensity). Fitness goal can be losing weight, gaining weight, getting toned (which includes a high protein diet and gaining some muscle weight), getting fit (which includes a high protein diet and losing some weight) and having a balanced diet.

The first step is calculating the BMR (Basal Metabolic Rate) using the Harris-Benedict equation which for men is:

$$BMR = 88.362 + (13.397 \times \text{weight (kg)}) + (4.799 \times \text{height (cm)}) - (5.677 \times \text{age (y)})$$

and for women:

$$BMR = 447.593 + (9.247 \times \text{weight (kg)}) + (3.098 \times \text{height (cm)}) - (4.330 \times \text{age (y)})$$

The next step is defining the Total Daily Energy Expenditure, TDEE, which is related to the activity level, and is calculated by multiplying the above calculated BMR with an activity factor. The activity factor ranges between 1.2 and 1.9. We will consider the following formulas:

- Sedentary:  $TDEE = BMR \times 1.2$
- Lightly active:  $TDEE = BMR \times 1.375$
- Moderately active:  $TDEE = BMR \times 1.55$
- Active:  $TDEE = BMR \times 1.725$
- High intensity:  $TDEE = BMR \times 1.9$

To adjust weight goal, we added between 250-500 calories/day in plus to the Total Daily Energy Expenditure for weight gain and subtracted around 500 calories/day from the Total Daily Energy Expenditure. Balancing the macronutrients is a complicated matter, but we tried creating a balanced diet for overall health and high protein diet for muscle growth, supporting muscle growth and repair. Additional factors can be considered by further studying the case.

It is important to note that this calculation is just an approximation, exact measures and calculations can be given by a specialised licensed nutritionist.

## CalorieNinjas API

CalorieNinjas provide a free plan for up to 10000 API calls/month. The input can be a simple text or a photo with a text. For BiteBalance, a simple text input is used on where the portion size must be specified first (one piece, 50 grams, one slice, etc.), then the food item. If no portion size is specified, then the default one is considered, which is 100 g. More food items can be added in a single input. The following endpoint is used:

`/v1/nutrition`

And the parameter of the API request is the text which contains the foods entered by the user. An example of an input text can be seen in Figure 4.12, together with the response. The response is structured in JSON format with items, each food item having its nutritional information.

two apples and one slice of pepperoni pizza

**Nutrition Results**

| Name            | Serving Size | Calories | Total Fat | Saturated Fat | Cholesterol | Sodium | Carbohydrates | Fiber | Sugar |
|-----------------|--------------|----------|-----------|---------------|-------------|--------|---------------|-------|-------|
| Total           | 475g         | 499      | 13.6g     | 5.7g          | 27mg        | 774mg  | 85.8g         | 11.1g | 41.3g |
| apples          | 364g         | 194.5    | 0.6g      | 0.1g          | 0mg         | 3mg    | 50.1g         | 8.6g  | 37.7g |
| pepperoni pizza | 111g         | 304.5    | 13g       | 5.6g          | 27mg        | 771mg  | 35.7g         | 2.5g  | 3.6g  |

Figure 4.12: CalorieNinjas API response

The API request is handled in Flutter, in the `food_screen.dart` file. It is handled in a function with the name `requestFood()`, it is an asynchronous function with a `Future<String>` return type. We transmit the text entered to the request by a `foodController`, which extracts the entered text in the input box by the user. The request is built with the help of a `http.get` function, where we transmit the endpoint of the API, the query (text entered by user) and as a header, we transmit the API key, which was obtained after registering to CalorieNinjas website. After the API has a response, it handles different scenarios: If the response is successful, it processes data accordingly. If there's an error, it handles that as well.

After we obtain the response, we decode it with a `jsonDecode` function, and it is ready to be added to the database. It is added to the database through the following endpoint with a `http.put`.

`http://IPAddress:3000/nutrientss/$userId/$date`

As header, we pass the type of the content, and as body, we pass the values extracted from the previous request. After finishing, the response is also handled on successful and error scenarios, with corresponding messages.

## User Authentication - Firebase

The user authentication is implemented in Firebase. After connecting the project to Firebase, authentication was enabled from the Firebase console. The connection process consists of having a Firebase account, creating a separate Firebase project that acts as a container for the Flutter project, configuring the two projects, adding Firebase services to the flutter project, re-building the flutter project, and importing necessary plugins.

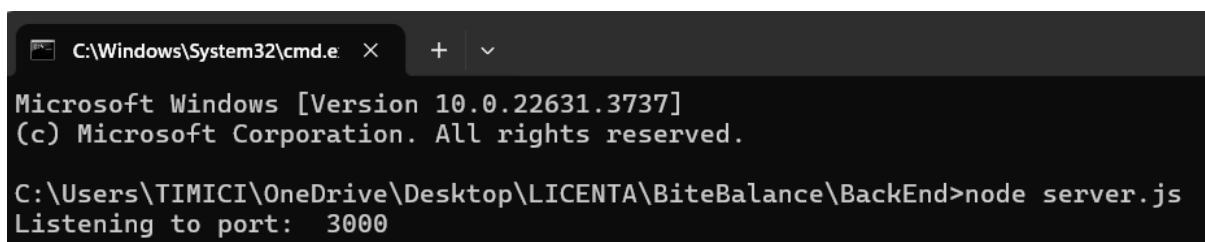
After that, a file was created, which handles all login services. It has separate functions implemented for getting the current user, for sending a password reset link, for sending an email verification link after creating an account, for logging in with google account, for creating a user with email and password - this is used to implement register functionality, for logging in with email and password and for signing out from the account. The log in with google functionality creates a google user variable, waits for the sign in process, provides as credential the access token. If the user is a first time user, the function return true. This is used in the app to redirect the user to the complete data screen.

All important information about the user such as userID is stored in a file name `globals.dart`. This file acts as a container where I stored the userID, the total calorieIntake, the personalGoal and the email address. These variables are set at login step and are used throughout the app. When the user logs out, these variables are reset.

If a user logs in for the first time after creating an account, it is redirected to a complete data screen, where he needs to input some information about his weight, height, etc. These are important in calculating the caloric and nutritional needs depending on users fitness goal.

## Database Connection

The code section that contains the connection to the database is implemented in a `server.js` file. It can be run from the command line as follows in Figure 4.13 with the **node server.js** command.



```

C:\Windows\System32\cmd.e  X  +  v
Microsoft Windows [Version 10.0.22631.3737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\TIMICI\OneDrive\Desktop\LICENTA\BiteBalance\Backend>node server.js
Listening to port: 3000
  
```

Figure 4.13: Running the server

In developing the backend, express, cors and mssql/msnodesqlv8 components were used. Express is a framework for node.js that handles routing and request/response management. I used it in handling http requests between the BiteBalance Flutter application and the database, and for testing the CalorieNinjas API. CORS enables communication between different domains, in this case the two components of the application, the backend and the frontend. Msnodesqlv8 library was used to ensure compatibility and to interact with Microsoft SQL Server Management Studio. I handled database connection pools and data querying with it.

The backend has get and put methods that handle data requests and data passing through the app:

- Getting all users from database, for verification purposes.
- Updating the data of the users, used in the step after registration, for completing additional data about the user.
- Getting userID from Firebase UID
- Getting a specific meal for a specific user on a specific date. Used in displaying data on history screen.
- Getting the total calorie intake based on userID. Used in displaying on the main screen the total number of calories that needs to be consumed.
- Getting the total amount of nutrients (Calories, proteins, carbs, fats, sugars and fibers) for a specific user on a specific date.
- Setting the nutrients of a specific user on a specific date. This step is used when the user adds a food as consumed. It retrieves the nutrients consumed that day so far, and adds the nutrients of the newly added food as sum of the two.
- Getting all nutrients for all users, for verification purposes.
- Getting the user data by ID.
- Getting the fitness goal of a specific user. This step joins two tables: fitnessGoal and user and searches for the specific fitness goal of the user. In user table, the goal is noted as an integer, and in fitnessGoal table, each integer has a specific meaning.
- Getting all meals by ID. Used in the app when displaying history screen.
- Getting the calories of a specific user on a specific date for a specific meal. Used when displaying on the history screen, the total number of calories consumed for a meal.
- Inserting a meal into the meal table.
- Deleting a meal from the meal table. This step subtracts the nutritional detail of the meal from the total daily nutrients consumed.

Let's detail how adding a food to the database and adding its nutrient information to the total nutrients consumed/day works.

Firstly, the code sets up an Express route:

```
app.put('/nutrientss/:userID/:date', (req, res))
```

It extracts the userID, the nutritional information and the date from the URL request parameters. It connects to the SSMS database using a sql.ConnectionPool. It executes a SQL query to retrieve existing nutrients from the database, for a specific user on a specific date. It then calculates the new nutrient values by summing up the already existing ones with the newly arrived ones.

If no matching row exists in the database (no food was added before and this is the first food of the user on the specific date), the code inserts a new row with the given nutrient values. If a matching row exists, the code updates the row with the new calculated values.

Then, the code sets the appropriate response headers and sends success (status code 200) or error (status code 500) messages.

The rest of the functions work similarly to the above described one.



## 4.3.4. Database Design

**Local database - SSMS**

The data in SSMS is organised in multiple tables, to help structure the meals and the users. The table definitions are shown in the tables below. Relationships between tables can be seen in Figure 4.14.

In table *User* we store user-related information about identification (name, firebase user ID - to connect to firebase account), physique, and nutritional needs that are calculated based on the physical data of the user (height, weight, gender).

In table *Meal*, we store information about each food entered in the application. Each entry has a specific ID, a name (banana, bread, peanut), a userID, a mealNameID, a date and caloric and nutrient information.

In table *Nutrient* we store information about each day's total consumption. When the user adds a food, its caloric and nutritional information is added to the table. When the user deletes a food, its caloric and nutritional information is subtracted to the table. The *lastDrink* entry notes when the user has drunk water last time.

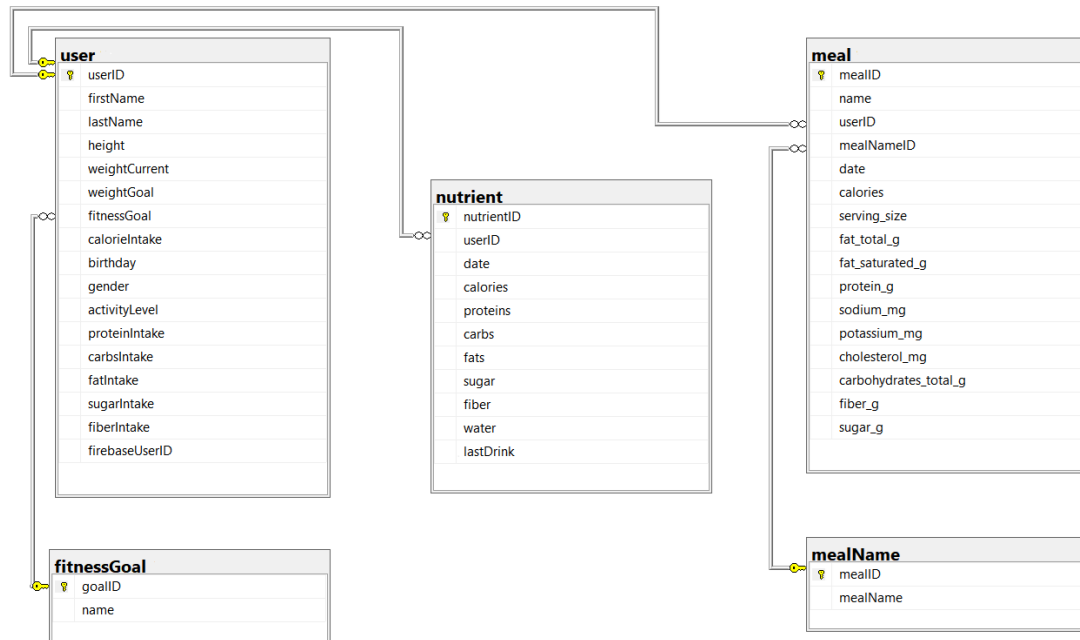


Figure 4.14: Database relations

Table *User* has the following data-types and structure *User*( *userID* int, *firstName* varchar(50), *lastName* varchar(50), *height* int, *weightCurrent* int, *weightGoal* int, *fitnessGoal* varchar(255), *calorieIntake* int, *birthday* date, *gender* char(1), *activityLevel* int, *proteinIntake* int, *carbsIntake* int, *fatIntake* int, *sugarIntake* int, *fiberIntake* int, *firebaseUserID* varchar(255), PRIMARY KEY (*userID*) );


Table *Nutrient* has the following data-types and structure *Nutrient* ( *nutrientID* int, *userID* int, *date* date, *calories* int, *proteins* int, *carbs* int, *fats* int, *sugar* int, *fiber* int, *water* int, *lastDrink* time(7), PRIMARY KEY (*nutrientID*) );



Table *Meal* has the following data-types and structure *Meal* ( *mealID* int, *name* nvarchar(50), *userID* int, *mealNameID* int, *date* date, *calories* float, *serving-size* float,

*fat\_total\_g float, fat\_saturated\_g float, protein\_g float, sodium\_mg float, potassium\_mg float, cholesterol\_mg float, carbohydrates\_total\_g float, fiber\_g float, sugar\_g float, PRIMARY KEY (mealID) );*

## Firestore

Authentication data is stored separately on Firestore server. The rest of the data regarding users and meals is stored in Microsoft SQL Server Management Studio. On Firestore, the CRUD operations are handled by Firestore. In Figure 4.15, we can see the way the data is stored in Firestore. Each user has an identifier, which is the email address, the provider, which for the present application can be either email or google, the creation date of the account, the last sign in date and a specific user ID.



| Identifier               | Providers   | Created ↓   | Signed in   | User UID                     |
|--------------------------|---|-------------|-------------|------------------------------|
| nagytimeatimici@gmail... |  | 10 Jun 2024 | 2 Jul 2024  | q6YwtMN6ZZVMtKaB0CV315...    |
| nagytimea123@yahoo....   |  | 10 Jun 2024 | 17 Jun 2024 | fWkcijPg73WI49eNF40C33ypz... |

Rows per page 50 1 – 2 of 2

Figure 4.15: Firestore Authentication data storage

## Chapter 5. Testing and Validation

### 5.0.1. Testing the BiteBalance application

The application was tested thoroughly during the development process on Android device and also after, on different devices. It is working optimally on an Android device, as it was designed specifically for that.

#### User journey within the application

- The user opens the app for the first time, the Login page appears. If the user doesn't have an account, goes to the Register screen by pressing the register button available on the screen and registers its account. The user must introduce a correct email address, and a password that must be 8 characters long, contain at least one uppercase letter, one lowercase letter and a number, the same password must be entered in both email and email verification fields. If these criteria are not met, a popup message will appear.
- After registering, the user gets an email verification link and the app will be on email verification screen until the user confirms the email address. If the email doesn't arrive or something is wrong, the user has the option to continue with another account from this screen.
- After this step, the user will be redirected to the complete data screen, where he can enter information about his physical data and personal goal.
- If the user doesn't have an account, he can also choose to log in with his already existing Google account, instead of registering with email. If it will be a first time user with the Google account, the user will be redirected to the complete data screen.
- After the user completes the requested data, he is redirected to the main screen of the application, from where he can view his personal data, he can add foods by text or by photo, view a detailed meal consumption history or view the detailed consumed nutrients for the respective day. These steps can be done in any order.

### 5.0.2. Testing the food recognition model

The testing of the food recognition model was done only manually, by uploading pictures to the application and getting the response. The model performs acceptably well in recognising the 101 food types present in the dataset.

## Chapter 6. User's Manual

### 6.0.1. Requirements and Installation

The general requirements for running a Flutter application are as follows:

#### Windows

- Windows 7 operating system, or later, 64-bit
- 2 GB free space recommended, 1.6 GB minimum
- Flutter environment set up
- Text editor IDE available, recommended Visual Studio Code

#### Android

- Android Jelly Bean or later version

#### iOS

- iOS 12 or later version
- To install the application, extract the content of the zip file into a folder.

### 6.0.2. Running the application

To run the application, firstly you need to run the server. Navigate in the extracted folder to BiteBalance/BackEnd, open a command line to this location and type:

```
node server.js
```

If everything is ok, the command line should display:

```
Listening to port: 3000
```

After running the server, the next step is running the flutter project. Choose a physical device, iOS or Android, connect it to the laptop with a wired connection. Make sure the device and the laptop are both connected to the same wifi network. This step is important to ensure proper data flow between devices. Open a command line to the following location: BiteBalance/FrontEnd/bite\_balance and type:

```
flutter run
```

After some time, the application should start and should be ready to use.

## Chapter 7. Conclusions

### 7.0.1. Obtained results

In conclusion, a mobile application was developed which handles user state management effectively, it recognises some food from user taken pictures, it calculates nutritional and caloric needs based on user information, it calculates total calories and nutrients consumed per day and also has a detailed history section and a profile section.

The application obtained is a great tool in helping individuals manage their weight by effectively tracking calories and nutrition, while keeping people motivated to continue the journey towards their desired weight goal and fitness goal. In my journey I faced similar challenges, and I considered that caloric and nutritional intake tracking is a difficult process. I found that is important to have friendly applications like BiteBalance to help people live a healthier life.

### 7.0.2. My own contributions

My own contributions to this project were developing the user interface of the application, creating the dataflow logic, designing the user's journey through the app, connecting the food recognition model with the Flutter app, connecting the Firebase container application to the Flutter project, implementing user authentication and database connections, designing the database structure and researching other similar applications, to have an idea about what would the market need in nutritional analysis and calorie tracking applications.

### 7.0.3. Areas of future improvement

I could name several future improvements, as the application is not perfect. It has some functionalities but for a great user experience it could have more. I have some ideas that could improve the application.

First of all, a more general model could be implemented that recognises a wider range and variety of foods. Currently, only 101 categories of foods are being recognised, which is not quite enough for real life daily usage. The training dataset could be more diverse. The application could categorise recognised food into categories like fruits, vegetables, meats etc.

Secondly, the food recognition model could have a portion estimation technology also. This technology also exists in other applications and could have been implemented to help the caloric and nutritional estimation accuracy. Currently, recognised food items are considered to be 100 g, which is not quite accurate.

Additionally, it would be easier for users to have a list of available foods from where he can select a food, categorised into food categories. Currently, the user can only enter the name of the food he consumed, or to input a photo.

To have a precise application that helps and assists people, a nutritionists advice would be beneficial, to improve the calculation of nutritional and caloric needs. Besides, it would be nice to incorporate a section that lets people connect and chat with dietitians, but this is a more advanced idea.

Another thing we could consider is the ability for the user to change information about his goal, weight, email address or other relevant information. Currently, these information can not be changed once entered.

As a final improvement I would fully develop this application and deploy it so it could be available on platforms like Google Play and Apple Store, to be available for everyone to download and use it.

Amongst these improvements, the error management system could be improved, to provide more popup messages, the food addition screen could be improved to have a nicer design, and the structure of the application could be improved to refresh faster, after changing something like adding a food element to the list.

## Bibliography

- [1] J. Mann, S. Truswell, and L. Hodson, *Essentials of human nutrition*. Oxford University Press, 2023.
- [2] B. Egan, “Protein intake for athletes and active adults: Current concepts and controversies,” *Nutrition bulletin*, vol. 41, no. 3, pp. 202–213, 2016.
- [3] S. C. on the Scientific Evaluation of Dietary Reference Intakes, S. on Interpretation, U. of Dietary Reference Intakes, S. on Upper Reference Levels of Nutrients, P. on the Definition of Dietary Fiber, and P. on Macronutrients, *Dietary reference intakes for energy, carbohydrate, fiber, fat, fatty acids, cholesterol, protein, and amino acids*. National Academies Press, 2005.
- [4] K. T. Ganson, J. M. Nagata, L. Vanderlee, R. F. Rodgers, J. M. Lavender, V. M. Hazzard, S. B. Murray, M. Cunningham, and D. Hammond, “Weight gain attempts and diet modification efforts among adults in five countries: a cross-sectional study,” *Nutrition Journal*, vol. 21, no. 1, p. 30, 2022.
- [5] “Managing your weight.” <https://www.nhs.uk/live-well/healthy-weight/managing-your-weight/>, Accessed June 2024.
- [6] B. Baker, “Weight loss and diet plans: Several types of diet plans produce at least short-term weight loss; portion size may matter more than what we eat.” *AJN The American Journal of Nursing*, vol. 106, no. 6, pp. 52–59, 2006.
- [7] “Weight loss mistakes.” <https://www.healthline.com/nutrition/weight-loss-mistakes>, Accessed March 2024.
- [8] E. P. Weiss, R. C. Jordan, E. M. Frese, S. G. Albert, and D. T. Villareal, “Effects of weight loss on lean mass, strength, bone, and aerobic capacity,” *Medicine and science in sports and exercise*, vol. 49, no. 1, p. 206, 2017.
- [9] S. Van Asbroeck and C. Matthys, “Use of different food image recognition platforms in dietary assessment: Comparison study,” *JMIR Formative Research*, vol. 4, no. 12, p. e15602, 2020.
- [10] “Caloriemama.” <https://play.google.com/store/apps/details?id=com.azumio.android.caloriesbuddy>, Accessed and Downloaded July, 2024.
- [11] “Lifesum.” <https://play.google.com/store/apps/details?id=com.sillens.shapeupclub>, Accessed and Downloaded January, 2024.
- [12] “Mynetdiary.” <https://play.google.com/store/apps/details?id=com.fourtechnologies.mynetdiary.ad>, Accessed and Downloaded January, 2024.

- 
- [13] “Cronometer.” <https://play.google.com/store/apps/details?id=com.cronometer.android.gold>, Accessed and Downloaded January, 2024.
  - [14] “Snapcalorie.” <https://play.google.com/store/apps/details?id=com.snapcalorie.alpha002>, Accessed and Downloaded January, 2024.
  - [15] “Flutter documentation.” <https://flutter.dev>, Accessed June 2024.
  - [16] “Firebase documentation.” <https://firebase.google.com>, Accessed June 2024.
  - [17] “Firebase authentication documentation.” <https://firebase.google.com/docs/auth>, Accessed May 2024.
  - [18] “Firestore documentation.” <https://firebase.google.com/docs/firestore>, Accessed April 2024.
  - [19] “Firebase realtime database documentation.” <https://firebase.google.com/docs/database>, Accessed July 2024.
  - [20] “Firebase hosting documentation.” <https://firebase.google.com/docs/hosting>, Accessed July 2024.
  - [21] “Microsoft sql server management studio.” <https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16>, Accessed June 2024.
  - [22] J. Zhang, S. L. Li, and X. L. Zhou, “Application and analysis of image recognition technology based on artificial intelligence – machine learning algorithm as an example,” in *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, 2020, pp. 173–176.
  - [23] B. Mahesh, “Machine learning algorithms-a review,” *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.
  - [24] W. Zhang, D. Zhao, W. Gong, Z. Li, Q. Lu, and S. Yang, “Food image recognition with convolutional neural networks,” in *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, 2015, pp. 690–693.
  - [25] <https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a>.
  - [26] L. Torrey and J. Shavlik, “Transfer learning,” in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.
  - [27] Y. Tian, “Artificial intelligence image recognition method based on convolutional neural network algorithm,” *IEEE Access*, vol. 8, pp. 125 731–125 744, 2020.
  - [28] P. Goldsborough, “A tour of tensorflow,” *arXiv preprint arXiv:1610.01178*, 2016.
  - [29] N. Ketkar and N. Ketkar, “Introduction to keras,” *Deep learning with python: a hands-on introduction*, pp. 97–111, 2017.



- [30] “Food 101 image dataset.” <https://www.kaggle.com/datasets/dansbecker/food-101>, Accessed December, 2023.
- [31] “Flutter official templates.” <https://www.fluttertemplates.dev/>, Accessed November 2023 - February 2024.
- [32] “Ui template.” <https://github.com/mitesh77/Best-Flutter-UI-Templates>, Accessed February, 2024.