Phaze

Technical Manual

Kevin Cogan

Kevin Bortas

Table Of Contents

1. Introduction

1.1. Overview

Phaze is a mobile and web-based application that is intended for the general public to improve their fitness and well-being with the goal to educate and alleviate obesity in the world. This is achieved through tracking and recording the user's activity and consumption of the vital micronutrients of food, allowing the users to make more informed dietary decisions. The development of both the android mobile and web application ensures that our users can access our there information anywhere at any time. Phaze provides many user-friendly features that aim to make fitness fun. These features include:

Micronutrient Breakdown:

This feature provides users with micronutrient information, such as fats, sugars and proteins, based on the searched food. This feature will quickly inform users on the food's nutrition allowing the users to make healthier choices in their eating habits.

Food identification:

This feature allows users to identify the caloric and micronutrient information of food by simply taking a picture of the food. Our software will identify the food and display the macronutrient breakdown to the users making it quicker and easier to get information on unlabelled food items.

Foot-step counter:

This tracks the activity of the user by counting the number of footsteps of the user. The footsteps are then converted into calories and displayed to the user so they can eat more or less depending on their fitness goals.

Food Diary:

This feature allows the user to track the foods they have eaten for breakfast, dinner, and lunch every day, week, or month. In the food diary, we provide a breakdown of caloric information to the users for each section. Helping users to keep track of their dietary habits.

Phaze Website:

This is a web-based application that allows the users to find out more information about Phaze on our homepage. The website allows users to manage their account and to view their current activity and micronutrient intake for the day.

The system uses several programming languages:

1. **Java:** was used to develop the mobile application
2. **Python:** was used to develop the Convolutional Neural Network (CNN) and the Flask server for the API and backend for the Phaze website.
3. **Javascript, HTML, CSS:** was used in the development of the frontend of the website.

1.2. Glossary

Micronutrients: are calories, fats, proteins, and carbohydrates.

Activity: is the number of steps a user makes.

CNN: stands for Convolutional Neural Network. This neural network is known for being good at image classification.

Flask server: this is a python based backend server that can be integrated with a variety of tasks.

API: stand for Application Programming Interface. This links different software and hardware applications.

HTML: Hypertext Markup Language is the frontend programming language that is used to design web pages.

CSS: Cascading Style Sheets is a way of styling an HTML page.

Front-end: These are the visual elements of an application that users can interact with.

Back-end: This is what adds functionality to applications and runs in the background.

Training dataset: This is used to train the neural network to make accurate predictions.

Validation dataset: This dataset checks how accurate the predictions of the neural network are.

Testing dataset: verifies the accuracy of the final model with the images in this dataset.

Filters: determines the number of kernels to use for convolutional of the inputted image that is converted into a matrix.

Feature map: is the result after applying a filter to the image that is inputted into the neural network.

Kernel Size: in a CNN refer to the height and width of the window that is processing the matrix of the image. Note: kernel_size must always be odd.

Strides: this determines how much the widow in a filter shifts on the x and y-axis of the matrix.

Padding: ensure that the output of the Conv2D is of the same width and height as the dimensions of the inputted matrix.

MaxPooling2D: is used to reduce the spatial output of the previous step, in our case it is for the Conv2D function.

Activation function: determines what is to be sent to the next layer/neuron based on the output of Con2D. The he_normal is perfect for deeper neural networks.

Kernel_initializer: this is used to control the values of the Conv2D class before the training starts. This helps to train deeper and more effective neural networks.

Overfitting: is when your model is too similar to the training dataset. As a result, the CNN will learn specific images instead of learning patterns causing difficulty in identifying images the CNN has not been trained on.

Dropout: helps to reduce the overfitting in the CNN model when training. This is achieved by changeling output neurons/nodes of the hidden layer to zero during the training Phaze of the neural network. A value is passed into the dropout function and this is the probability of the function being activated to do the task described above.

GlobalAveagePooling: generate the average output of each feature map from the previous layer. This helps to reduce overfitting.

Dense Layer: provides only one output for each neuron to the next layer.

Kernel_regularizer: helps to reduce overfitting and by adding penalties to weights of neurons that may cause overfitting.

Data Augmentation: a process of distorting the original image to improve the training accuracy of the CNN model.

Epoche: is an iteration through the dataset.

Pool_size: is an input of the MaxPooling function in TensorFlow. This input determined the window size that the map function will be selected from. For example, a pool_size = (2,2) will find the largest value in two by two windows.

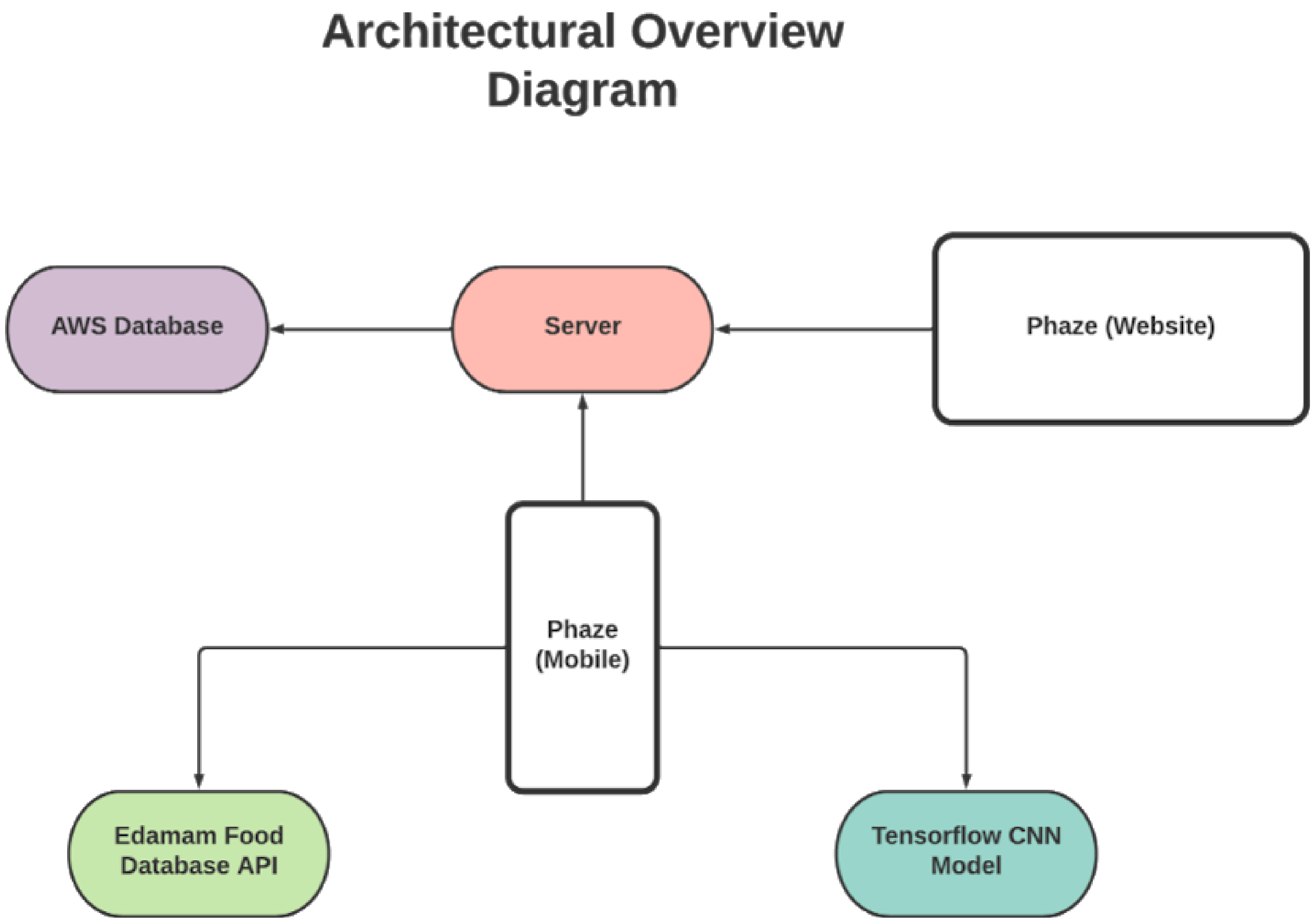
Fully Convolutional Layer: this is a layer that only performs convolutional operations.

Bottleneck Layer: is a layer that has less neuron/nodes than the previous layer.

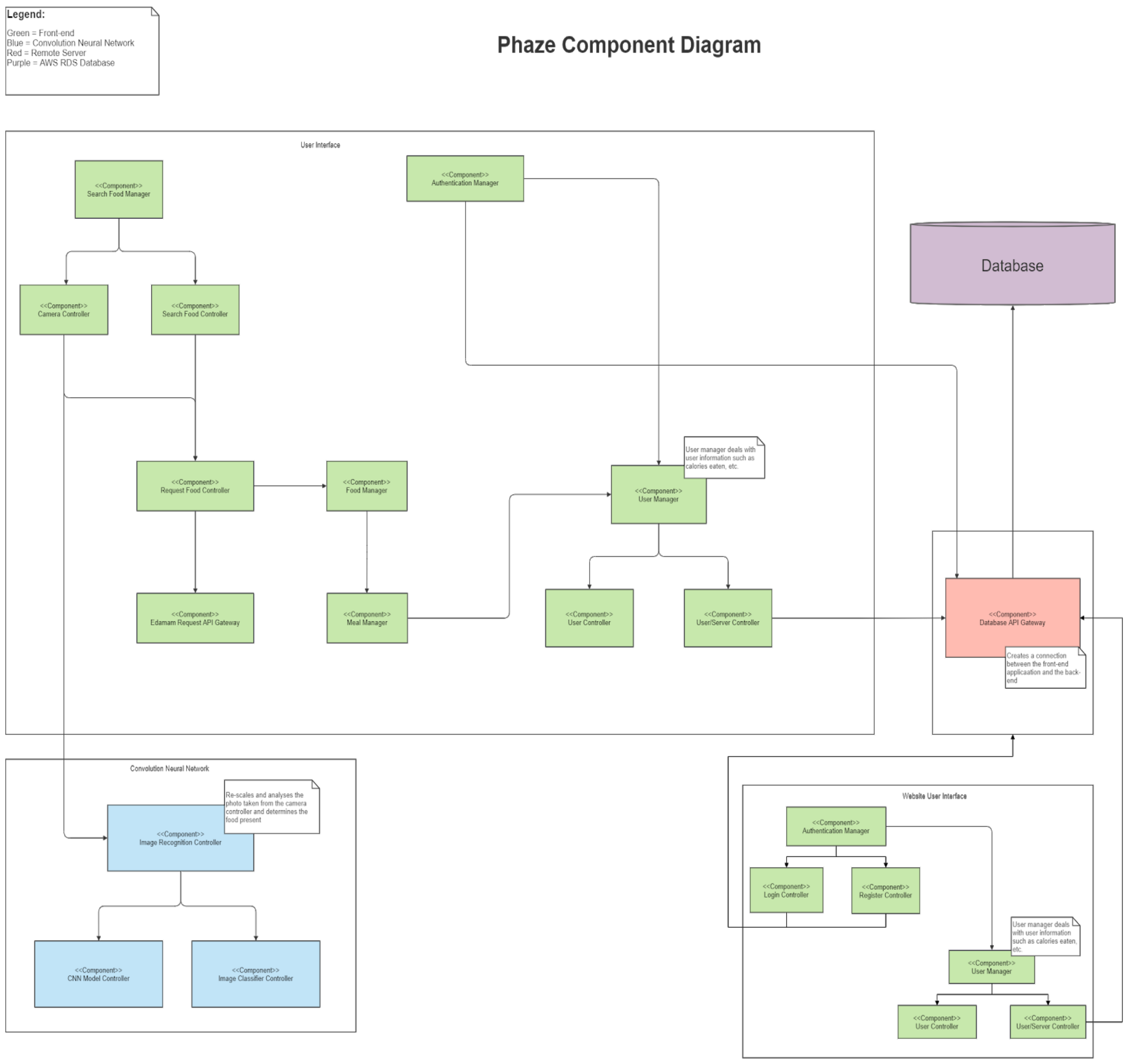
Transfer learning: is the method of using a pre-trained model as a starting point to create a new model with the new data.

2. System Architecture

2.1. Phaze Architectural Overview Diagram



2.2. Phaze Component Diagram


Legend:
Green = Front-end
Blue = Convolution Neural Network
Red = Remote Server
Purple = AWS RDS Database
Phaze Component Diagram
User Interface
<<Component>> Search Food Manager
<<Component>> Authentication Manager
Database
<<Component>> Camera Controller
<<Component>> Search Food Controller
User manager deals with user information such as calories eaten, etc.
<<Component>> Request Food Controller
<<Component>> Food Manager
<<Component>> User Manager
<<Component>> Edamam Request API Gateway
<<Component>> Meal Manager
<<Component>> User Controller
<<Component>> User/Server Controller
<<Component>> Database API Gateway
Creates a connection between the front-end applicaation and the back-end
Convolution Neural Network
Re-scales and analyses the photo taken from the camera controller and determines the food present
<<Component>> Image Recognition Controller
<<Component>> CNN Model Controller
<<Component>> Image Classifier Controller
Website User Interface
<<Component>> Authentication Manager
<<Component>> Login Controller
<<Component>> Register Controller
User manager deals with user information such as calories eaten, etc.
<<Component>> User Manager
<<Component>> User Controller
<<Component>> User/Server Controller

Search Food Manager:

This contains the components needed to search for a food

Camera Controller:

This is part of the Camera Fragment and allows the user to take a picture

Search Food Controller:

This allows the user to manually search for a food

Request Food Controller:

This controls the Edamam API request

Edamam Request API Gateway:

This Gateway connects the application with the Edamam Database.

Food Manager:

Deals with the food information

Meal Manager:

Deals with the meals eaten by the user such as breakfast, lunch, etc.

User Manager:

This manages the user information

User Controller:

This controls the user information locally

User/Server Controller:

This connects the user with the remote server and the AWS Database

Authentication Manager:

This manages the login and register features.

Login Controller:

This contains the necessary components to log in to the application

Firebase Login Gateway:

This connects the application to the Firebase Database

Register Controller:

This contains the necessary component to register for the application

Firebase Registration Gateway:

This connects the application to the Firebase Database

Database API Gateway:

This allows communication between different systems involved.

Database:

This is the AWS RDS Database

Image Recognition Controller:

This contains the Convolutional Neural Network components

CNN Model Controller:

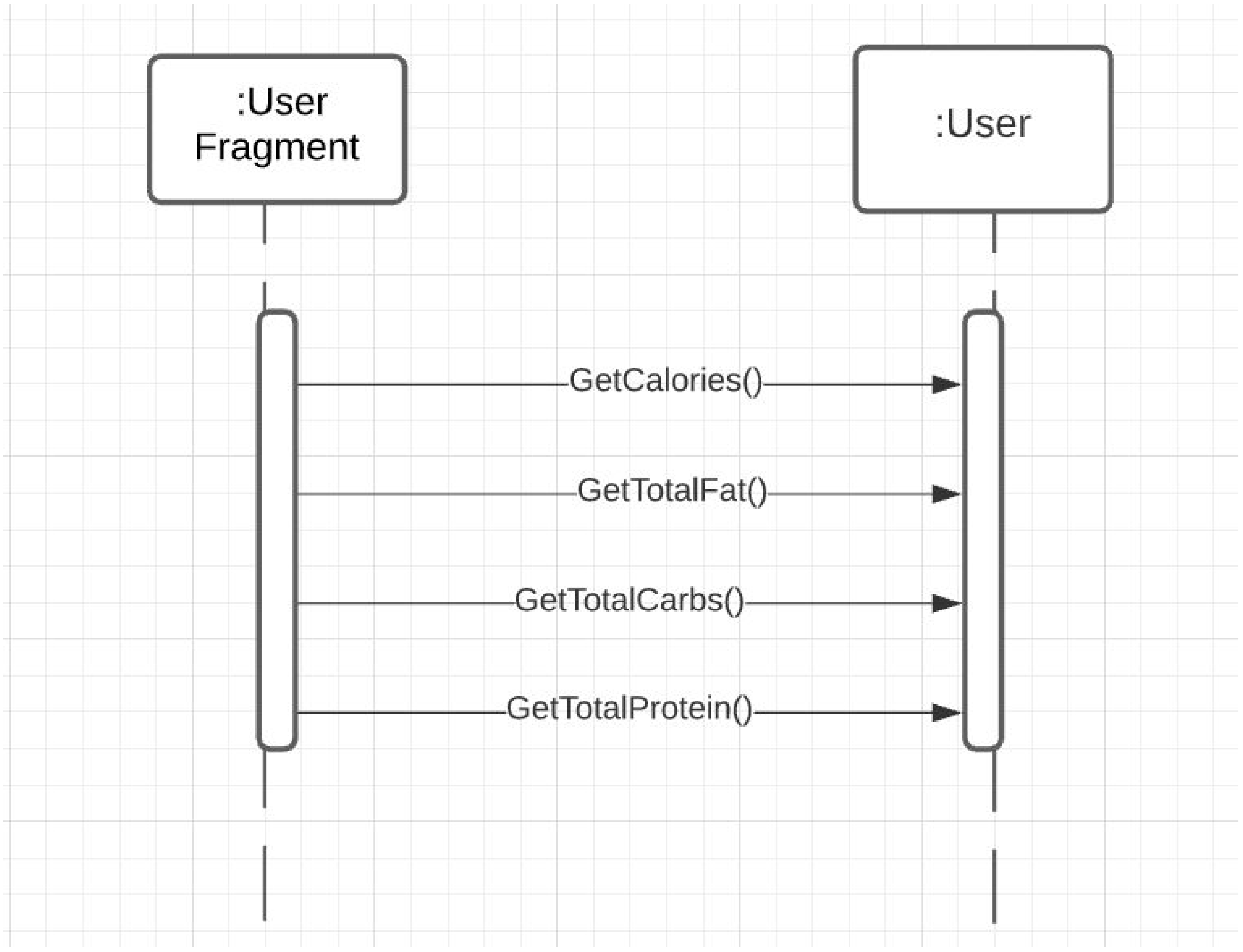
This is our CNN Model

Image Classifier Controller:

This controls the components needed to resize the image taken by the user

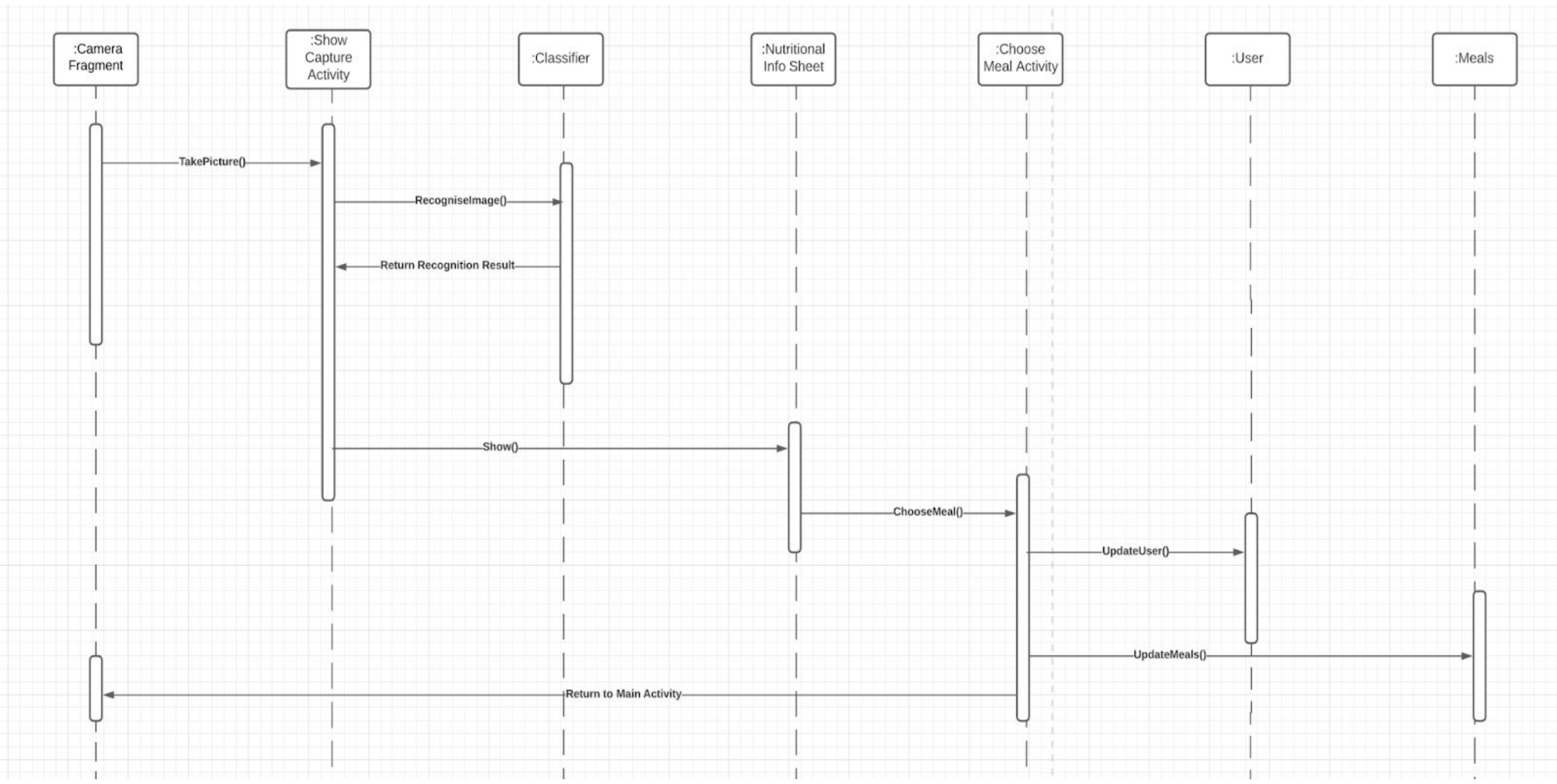
2.3. Mobile Application Sequence Diagrams

Micronutrient Breakdown



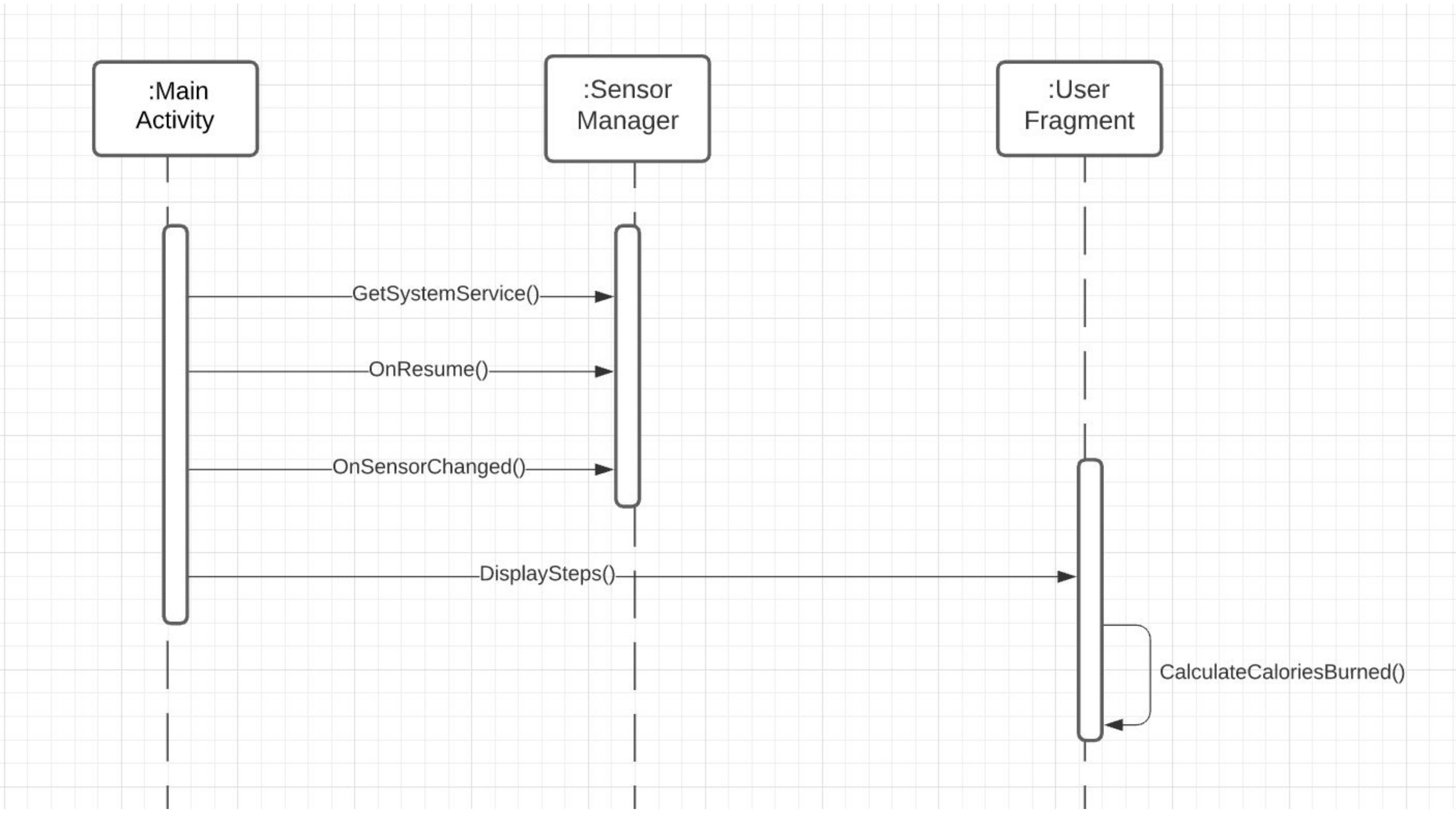
The User Fragment uses the User class to retrieve the stored nutritional information such as the calories, protein, fats and carbohydrates and displays them on the fragment.

Food identification



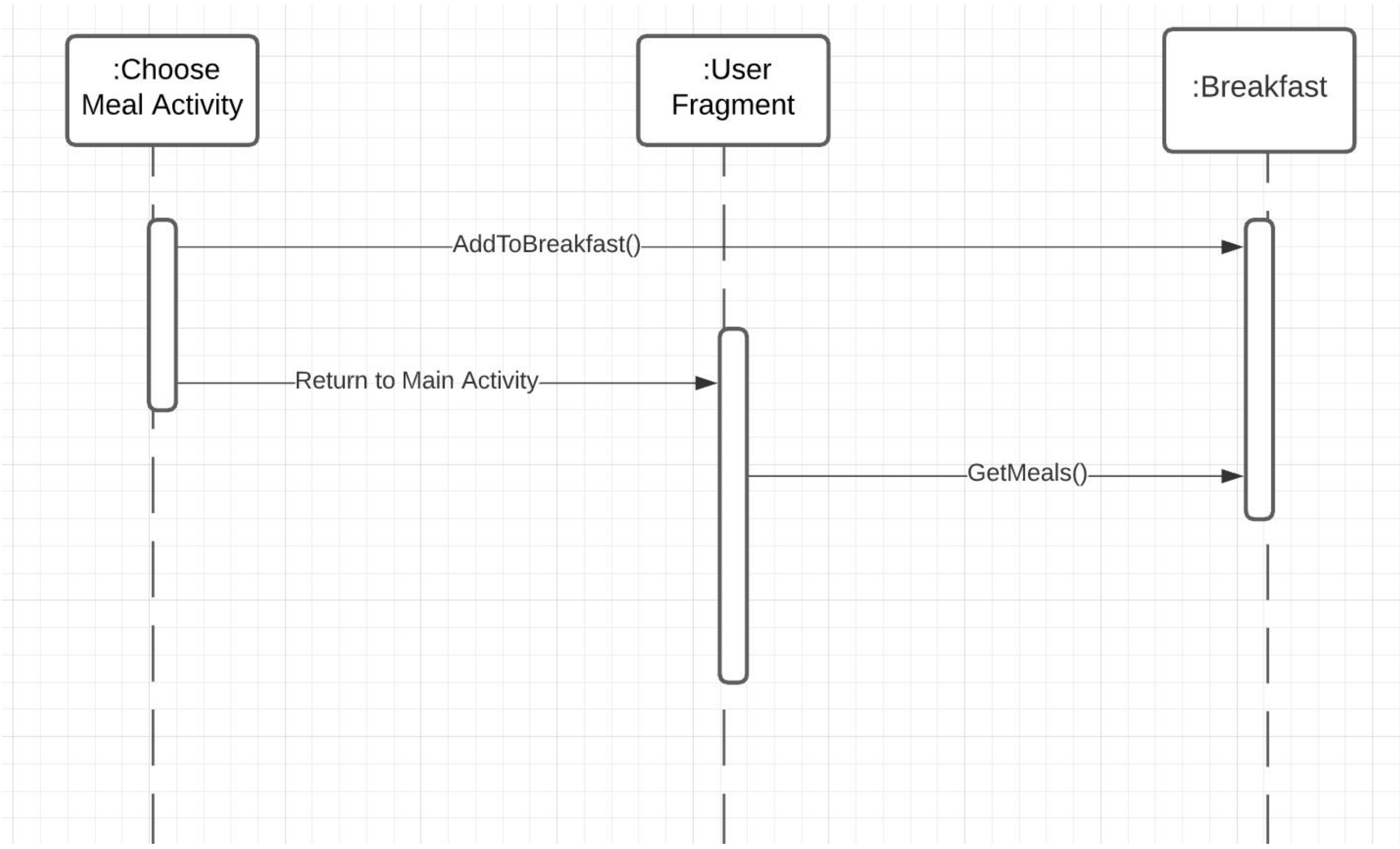
The Camera Fragment executes the takePicture() method when the user presses the take picture button. This picture is then displayed in the Show Capture Activity and passes image through the Classifier which rescales and resizes the image and uses the food classification CNN to create a prediction. This information is displayed using a Bottom Sheet Dialog called Nutritional Info Sheet. The user can now add that food to a meal using the Choose Meal Activity. The nutritional value of the chosen food is calculated and stored in the User class and the user is returned to the main page.

Foot-step counter



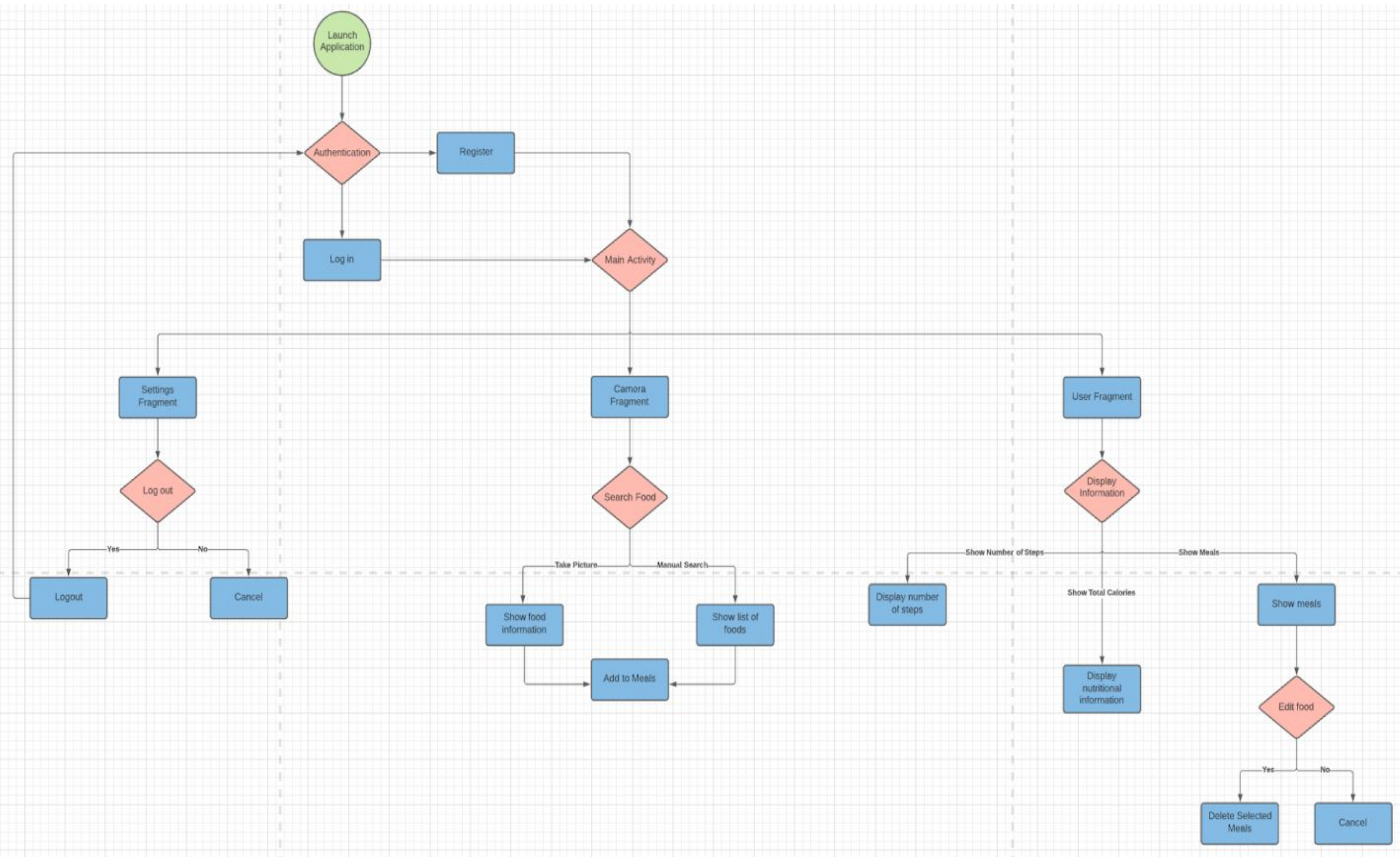
This tracks the activity of the user by counting the number of footsteps of the user. The footsteps are calculated in the Main Activity using the mobile phone’s internal Sensor Manager class. The Main Activity accesses this information using the getSystemService(), onResume() and onSensorChanged() methods, which then displays this information on the User Fragment and uses this to calculate the calories burned.

Food Diary

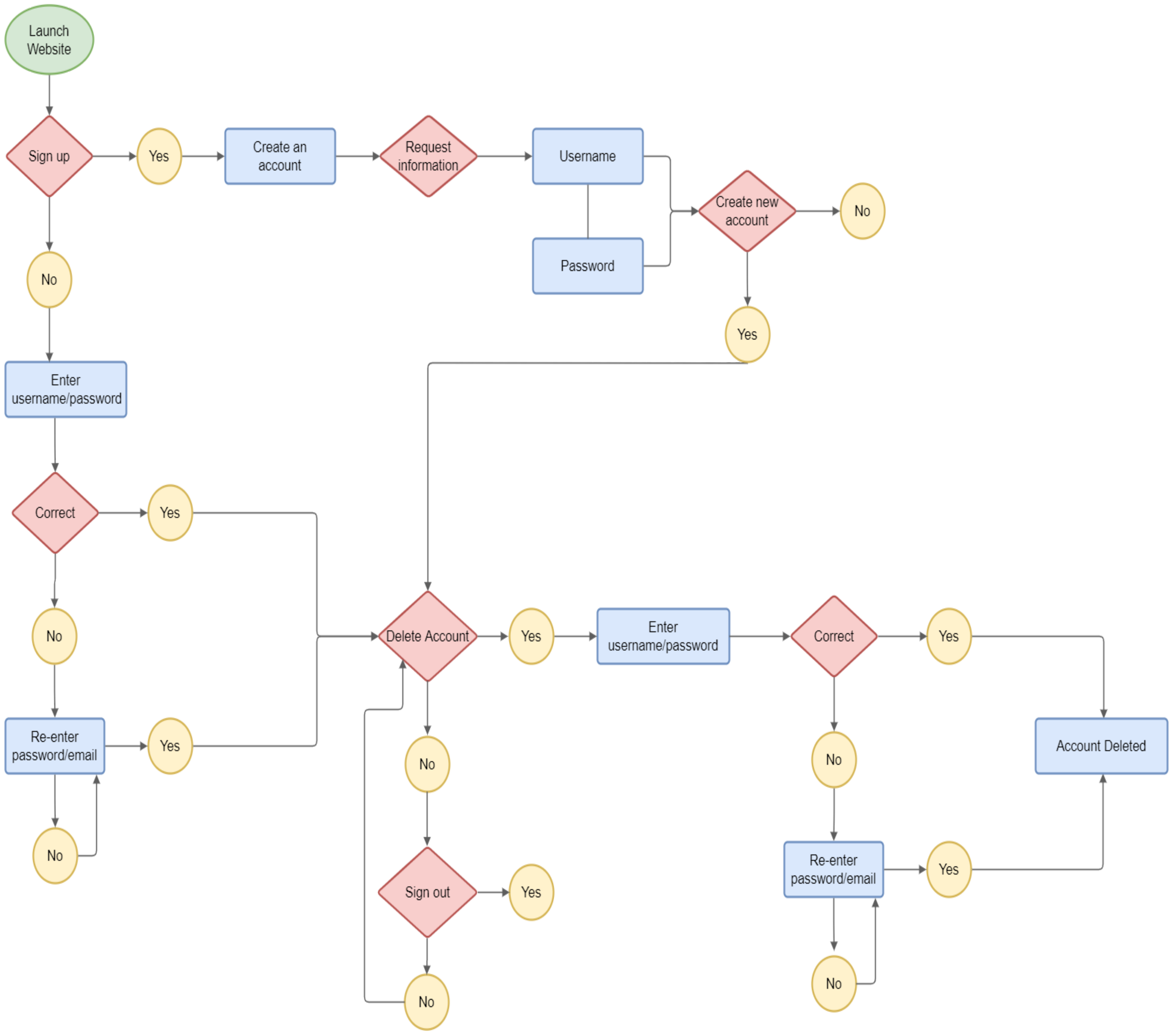


This feature allows the user to keep a record of the food that they have eaten. When a user has selected food and serving size, the user has the option to add to a particular meal (in this case the meal is breakfast). The food is stored in the corresponding meal class. The user is then returned to the main page and they can access this information on the User Fragment.

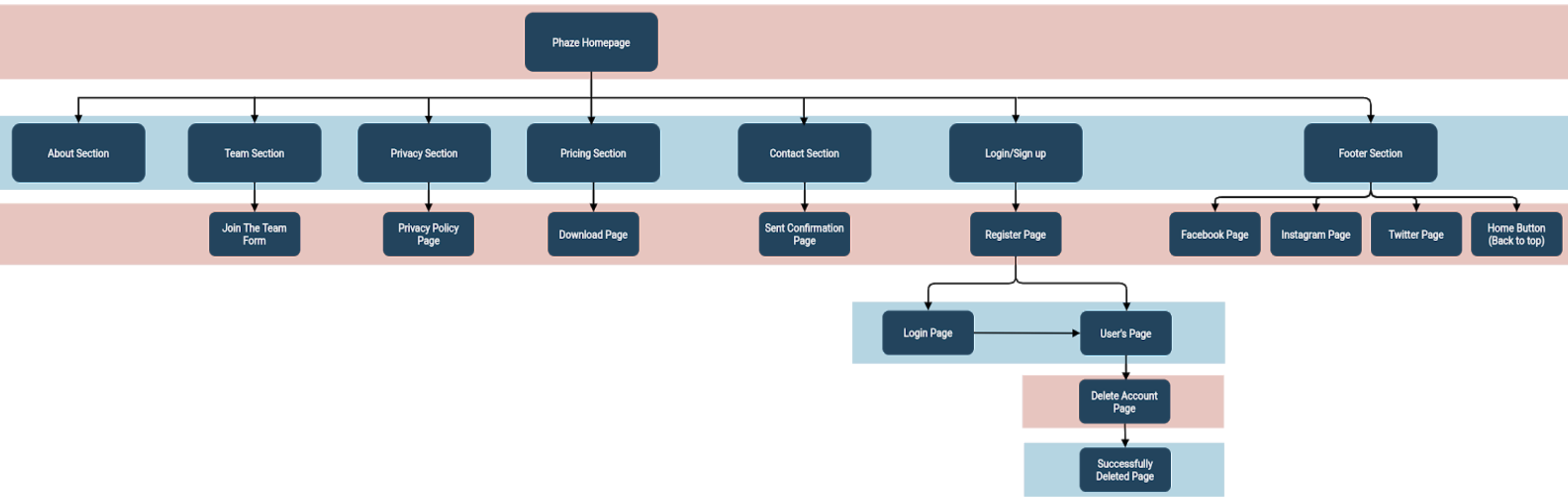
2.4. Mobile Application Flow Diagram



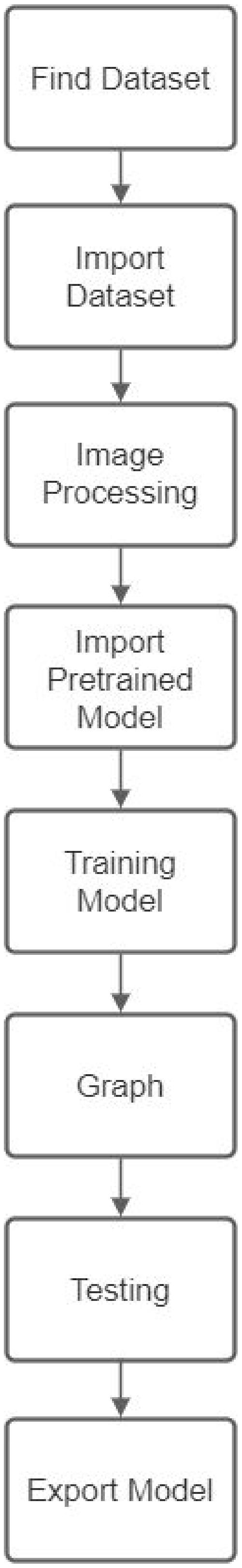
2.5. Website Flow Diagram



2.6. Website Site Diagram



2.7. Data Transfer CNN Flow Diagram



1. **Find Dataset:**

   The CNN program was sourced from a food dataset from the TensorFlow dataset called ‘food101’. This dataset contains 101 different food classes with 101,000 images in total. This dataset was sufficient for the identification of the most common meals.

2. **Import Dataset:**

   The ‘food101’ dataset was imported into the CNN via an API request to the TensorFlow database using the TensorFlow library function, tensorflow_dataset.load(). This function will load the dataset into your environment and will be ready to use once it is fully downloaded.

3. **Image Processing:**

   The images need to be processed in a format that will enable us to pass them into our CNN.

   1. Dividing the imported dataset into a training, validation, and testing dataset. It is important to ensure each dataset has unique images as this will reduce overfitting occurring.
   2. The images then have to be scaled down and converted into matrices.

4. **Import Pretrained Model:**

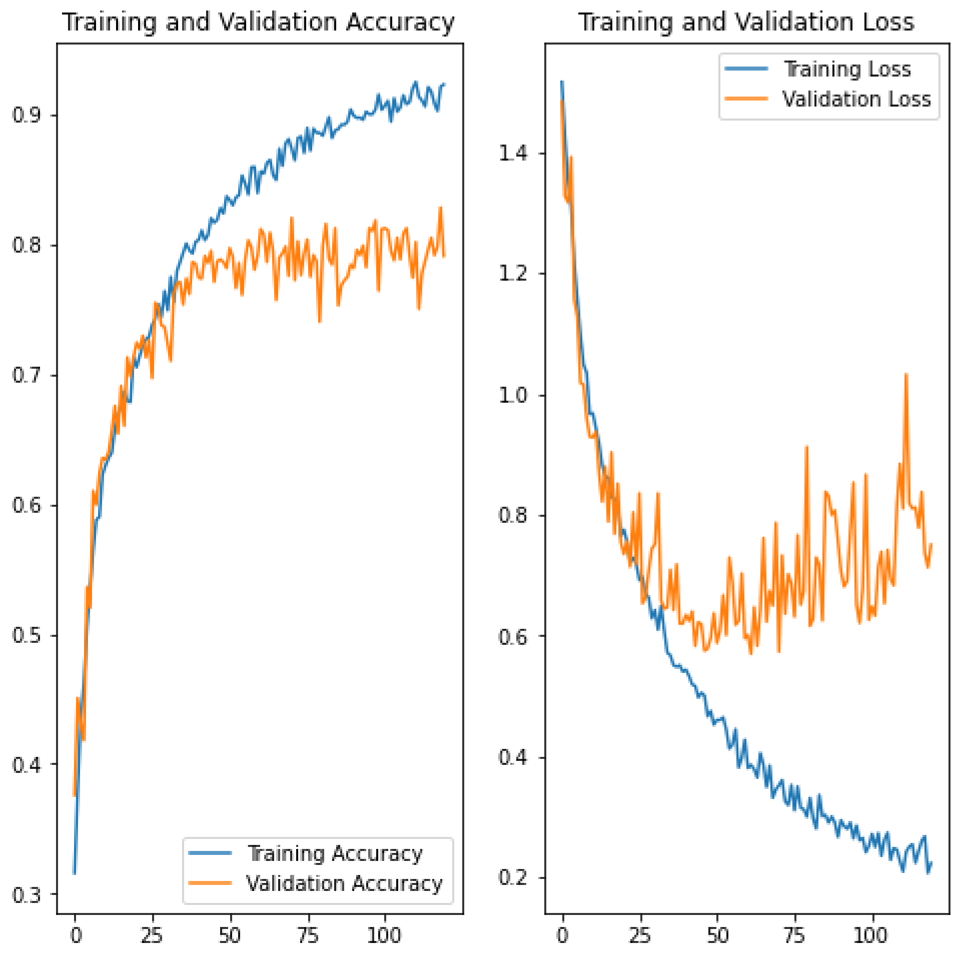
   The pre-trained model used is Mobilenet_v2. This model initially uses a full convolutional layer of 32 filters which is then followed by 19 bottleneck layers and is specifically designed for mobile devices due to its speed and fewer parameters needed to run more accurately.

5. **Training Model:**

   Transfer learning is used to train the Convolutional Neural Network due to the computational power needed for such a large dataset. This enabled us to train our CNN faster with more accuracy.

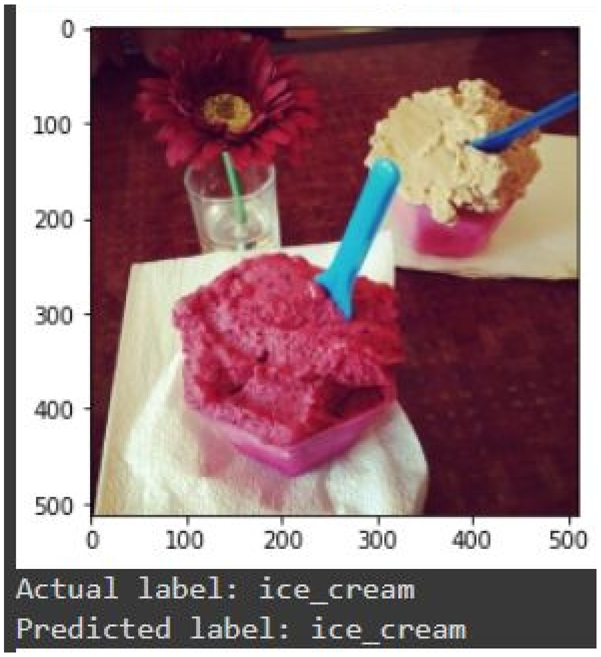
6. Graph:

Graphing was used to visually see the behaviour of our model so we could make more informed decisions on adjustments to the model. We graphed the accuracy and the loss to ensure loss was consistently decreasing and accuracy was consistently increasing.

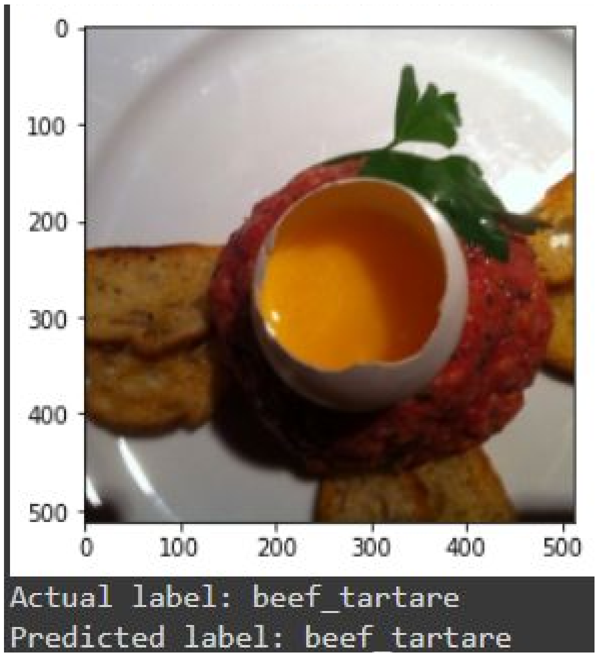


7. Testing:

Testing was done by calculating the model accuracy by using the test dataset to calculate the overall prediction accuracy. Furthermore, automated testing was conducted by comparing the image with the actual label and predicted label.

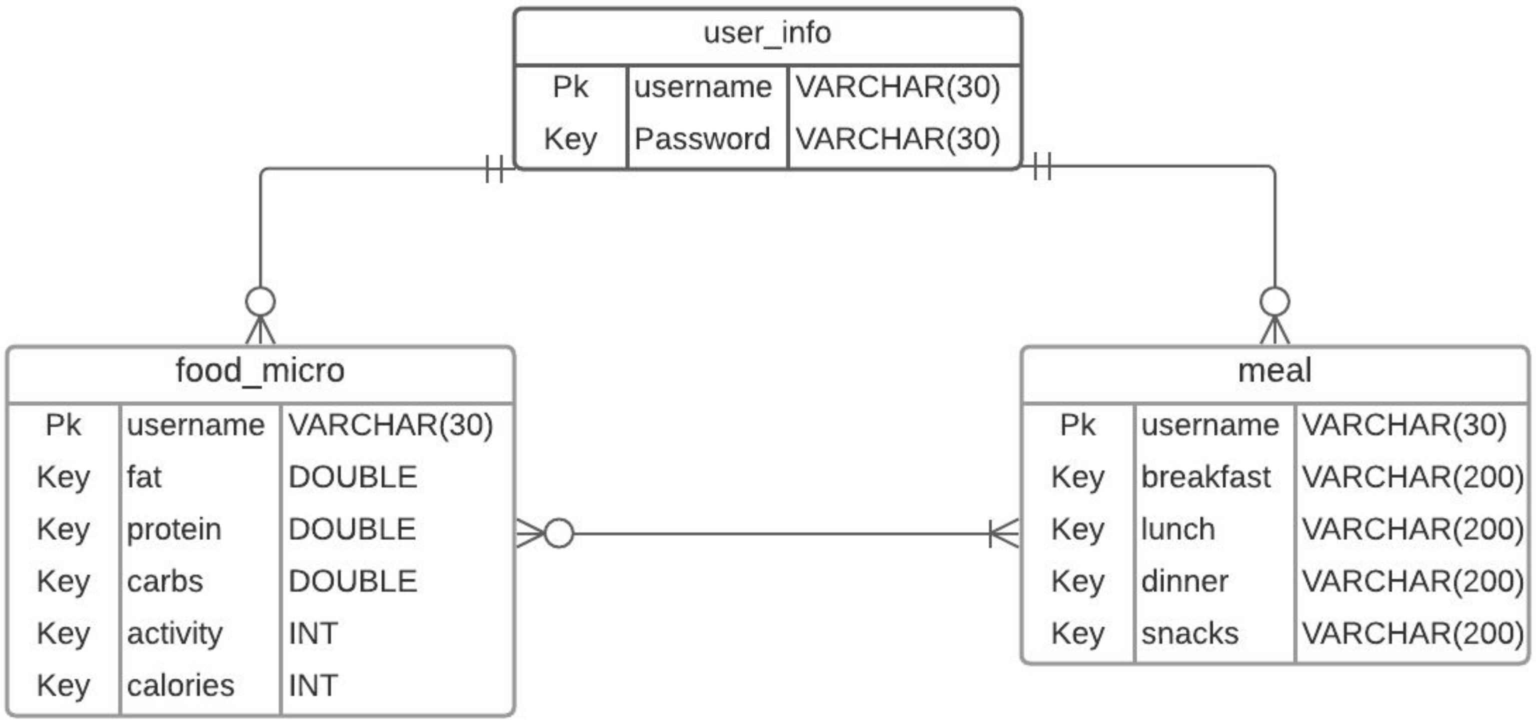


```
Actual label: ice_cream
Predicted label: ice_cream
```



```
Actual label: beef_tartare
Predicted label: beef_tartare
```

2.8. Database Entity Relationship Diagram



3. Testing

3.1. Unit Testing

A large emphasis was placed on unit-testing at the beginning of the project as this allowed quick debugging of errors and easy refactoring which will become more helpful as the code for the system grows bigger. We used best practices of software testing by concentrating on edge cases as this is where most errors are found. Unit testing has been completed on ally components in the system

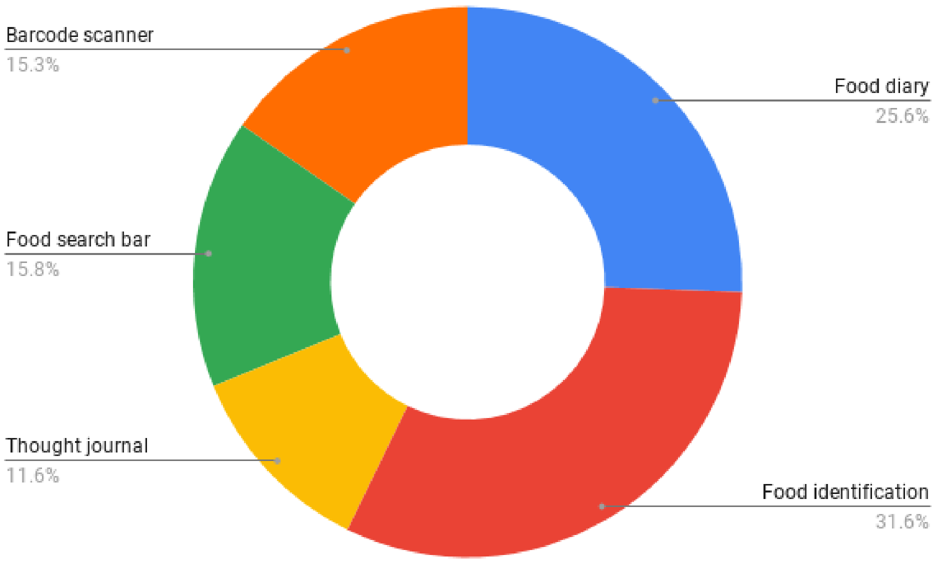
3.2. Integration Testing

Throughout our project's lifecycle, we conducted integration testing on our system so once we have developed our other systems we ould be able to combine these systems with minimal errors. Integration testing was conducted on:

- Android application
- Web application
- Amazon Web Service (AWS) Relational Database
- Flask Server
- Hosting Service for the flask server - AWS Elastic-Beanstalk

3.3. Needs Assessment

The Phaze team wanted to ensure we were developing a project that is of use to the potential users so we conducted a survey outlining features that the user would like to use. Once the survey was completed and analysed we could make a more informed decision on what features would benefit the end-users the most.



3.4. User Testing

The Phaze Team conducted multiple user acceptance testing one after the development of the prototype of the application and again once the improvements were implemented. The goal was to create a user-friendly, intuitive mobile and web application that offers functionally to users helping them have a better quality of life. The Phaze team plans to continuously conduct surveys after a major development in the life cycle of the project.

We performed the first survey on fifteen participants of different backgrounds. We targeted people of a non-technical background as they will potentially encounter more problems as they are not experts in the field of computing. Secondly, we targeted people of different age groups, people ages between 18 - 24 and 45 +. This spread will hopefully give use more information helping us make more informed decisions on improvements on the Phaze application. (Surveys links can be found in the appendix)

3.5. World Wide Web Consortium (W3C) Testing

Website testing was done via the W3C website validator to ensure that our code complied with the W3C standards. Our code has passed all tests set by W3C.

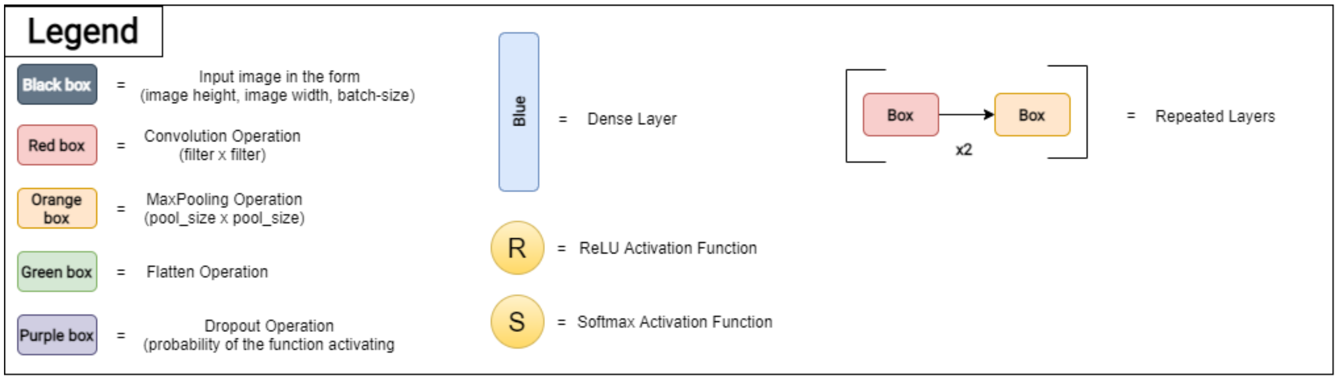
3.6. End-to-end

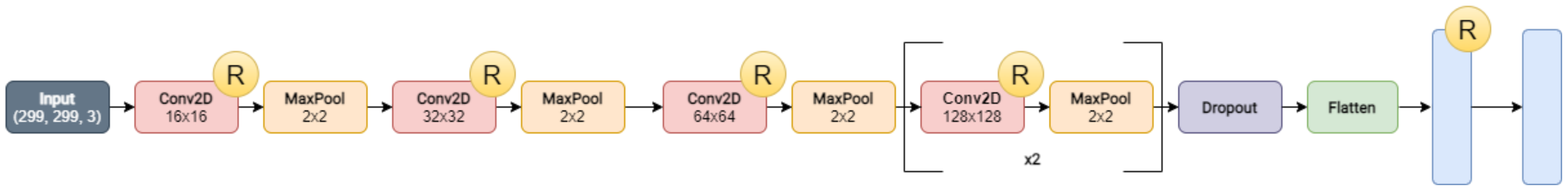
In the last week before the project deadline, the Phaze performed comprehensive end to end testing on the Phaze system. This enabled us to test the system as a whole and to catch any bugs that we have not noticed previously. Due to the ridge and frequent unit-testing and integration testing we performed we did not find any bugs in the final end-to-end testing.

4. Problems

4.1. Computation Power

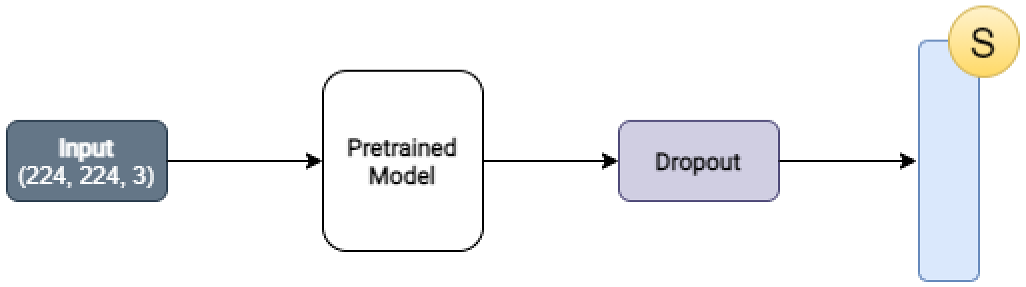
Originally we developed a neural network without a pre-trained model (mobilenet_v2). This model requires all the steps in the CNN flow diagram with the addition of data augmentation. The model achieved a validation accuracy of 84% with five food classes and a validation accuracy of 76% with 6 food classes. The addition of one extra food class of 1000 images per food class added an hour of training time and epochs. The structure of the model that achieve these results is shown below:


Legend
Black box = Input image in the form (image height, image width, batch-size)
Red box = Convolution Operation (filter x filter)
Orange box = MaxPooling Operation (pool_size x pool_size)
Green box = Flatten Operation
Purple box = Dropout Operation (probability of the function activating
Blue = Dense Layer
R = ReLU Activation Function
S = Softmax Activation Function
Box → Box x2 = Repeated Layers



After predicting the current training rate we deemed that this method would take a considerable amount of time to test our model which is not viable for the project development.

Solution



Transfer Learning was the solution to this problem as we could leverage the computation already conducted in the pre-trained models and use that to train our new model. We achieved accurate results of 95% accuracy for 101 food classes in under an hour of training the model.

4.2. Hosting the Flask server - Problem with sending data method.

Initially, when setting up the Flask API server to manage the user's account information we configured the server to send information to the Flask server through the URL. An example of how the server accepts the URL can be seen below:

```
@application.route("/<usr>")
def user(usr):
    data = usr.split("%")
    username = data[0]
    password = data[1]
```

We encountered a problem with trying to host this method of passing information this way as the AWS ElasticBeanstalk was not compatible with the format. Furthermore, with more consideration of the design, we deemed this design to be a security issue as we will be sending sensitive account information such as usernames and passwords.

Solution

We decided to pass the information via `request.form` this offered more security as well as making the software compatible with AWS Elasticbeanstalk meaning we could host the flask server online. An example of the new format the Flask server accepts information can be seen below:

```
@application.route("/login", methods=["POST", "GET"])
def login():
    if request.method == "POST":
        info = request.form
        username = info["Username"]
        password = info["Password"]
```

4.3. Firebase vs AWS MySQL Relational Database

We initially used a Firebase NoSQL database due to the ease of use and integration with the android mobile application. The purpose was to authenticate, create, and delete user details from the database. However as our system became more complex we wanted to store more information such as meals, micronutrients, user activity and much more. We require more control over how we could store and retrieve this information.

Solution

We set up the MySQL Relational Database with AWS and python connector library to query the database. This enables us to tailor the table and queries specifically to our needs. Enabling us to provide plenty of information to our users via the website or mobile application.

4.4. Google Cloud Services - Image Classification

We found AI Vision which is a Google Cloud Service tool that allows customers to quickly and easily use image classification. We set this service up and began testing it with sample images of food. We then realised a flaw with this service. Although it was very accurate it firstly identified every object including plates and other objects but it also used very poor classification labels for foods. For example, a pick of steak would return the label ‘Meat’. This was a problem due to the lack of precision of the label.

Solution

Our solution was to write our own Convolutional Neural Network (CNN) when we can have more control over the labels and what kinds of objects we want to identify, in our case food class. After using several libraries such as Pytorch and Keras we finally settled with TensorFlow where we developed our first neural network that worked perfectly. Since then we have applied more advanced features to make our neural network more accurate.

4.5. Barcode Scanner - Google Vision API

One of our aims for Phaze was to implement a barcode scanner and return the nutritional information of that particular food/product. We discovered a Google Vision API which was able to achieve this and we began to implement it within our application. However, we encountered a design flaw and a limitation of an android application when we created 2 camera objects within our Camera Fragment. The first camera object was in charge of using the physical device’s camera and allowing the user to take a picture while the second object was using the Google Vision API. The barcode scanner worked as it should however now we were unable to successfully take a picture and use our image classifier.

Solution

To combat this issue we needed to redesign our implementation of the barcode scanner. We decided to use only one camera object (the original camera object) and to allow the user to take a picture. Now instead of revealing a barcode to the camera, the user must take a picture of the barcode and this image will be analysed in the Show Capture Activity, where it is either passed to the image classifier or given to the Google Vision API which will identify the barcode.

5. Deployment

5.1. Android Studio Test Download

- **Get the source code**

```
git clone https://gitlab.computing.dcu.ie/bortask2/2021-ca326-phaze.git
```

- **Set up Android Studio**

1. Download Android Studio: https://developer.android.com/studio
2. Open the application using Android studio
3. Connect a physical device using a USB cable or create a virtual emulator
4. Run the application using the selected device

5.2. Google Play Store Download

1. Open the Google Play Store application on your Android device
2. Search for “Phaze”
3. Download the application if there is sufficient space on your device and you are using the most up to date version of the android operating system
4. Once downloaded, launch the application

5.3. Manually Run Specific Components

- **Get the source code**

```
git clone https://gitlab.computing.dcu.ie/bortask2/2021-ca326-phaze.git
```

5.3.1. Manually Run CNN

1. Open `image_classification` folder
2. Dependencies - automatically install dependencies listed below:

```
pip install -r requirements.txt
```

Programme	**Version**
matplotlib	3.2.2
matplotlib-venn	0.11.6
tensorflow	2.4.1
tensorflow-datasets	4.0.1
tensorflow-estimator	2.4.0
tensorflow-gcs-config	2.4.0
tensorflow-hub	0.11.0
tensorflow-metadata	0.28.0
tensorflow-probability	0.12.1

3. Execute the code on the command line:

   ```
   python3 transfer-learning_image-classification
   ```

5.3.2. Alternative: you can click on the link below and run the code immediately:

https://colab.research.google.com/drive/1xqqB4PdPVrYWuwqtFL8oM5SKyMopo9YQ?usp=sharing

5.3.3. Manually Run The Flask Server - Locally

1. Open `flask-api` folder

2. Dependencies - automatically install dependencies listed below:

   ```
   pip install -r requirements.txt
   ```

Programme	**Version**
`click`	`7.1.2`
`Flask`	`1.1.2`
`itsdangerous`	`1.1.0`
`Jinja2`	`2.11.3`
`MarkupSafe`	`1.1.1`
`mysql-connector-python`	`8.0.23`
`protobuf`	`3.15.1`
`six`	`1.15.0`
`Werkzeug`	`1.0.1`

3. Execute the code on the command line:

   ```
   python3 application.py
   ```

5.3.4. Manually Run The Flask Server - Remotely

1. Open `flask-api` folder
2. Run the command on your command line to zip the files state below:

   ```
   zip application.py requirements.txt
   ```

3. Login to AWS ElasticBeanstalk by clicking on the link below:

   https://eu-west-1.console.aws.amazon.com/elasticbeanstalk/home?region=eu-west-1#/environment/dashboard?applicationName=3YP-API&environmentId=e-3be3ysxhgv

4. Enter the Root user and Password :

   Root user: bothkevins@gmail.com

   Password: BortasCogan2020!

5. Select the Upload and deploy button
6. Select the zip file and upload the file and wait upto 5 minutes as the programme is deployed on the AWS server. Once completed with no errors the application is now remotely deployed.

5.3.5. Manually Run The AWS MySQL Relational Database

1. Login to AWS RDS by clicking on the link below:

   AWS Database login link

2. Enter the Root user and Password :

   Root user: bothkevins@gmail.com

   Password: BortasCogan2020!

3. Select the click Action drop-down button where you will have a variety of options.

   These include:

   - Start/Stop
   - Reboot
   - Delete
   - Create read replica
   - Take snapshot
   - Restore to point in time
   - Start database activity stream

4. If the database is not active select Start and the database will begin to run.
5. To modify the tables on the remote database server download MySQL Workbench and login with these credentials:
   - Connection Name: database-users
   - Host Name: database-users.cwhtfwf8upc3.eu-west-1.rds.amazonaws.com
   - Username: admin
   - Password: BortasCogan2020!
6. Once login in you will be able to execute SQL commands on the database.

Future Work

If given more time, we would like to create more features and enhance the current features of the application and website. These ideas include:

1. Adding more foods to the food classification feature
2. Expanding the database to store weekly, monthly and yearly user activity and micronutrient intake
3. Improve the food step counter by using other hardware such as GPS.
4. Create and add and remove functions to the website food diary.
5. Food diaries get feedback from users in the text and use sentiment analysis to offer recommendations.
6. Use `Werkzeug` to create encrypted posting of data on the network or get a domain with SSL Certification so there is no security breach when entering the password.

Finally, we would like to look further into Data Protection Legislation to see if we can store more data on users so we can create more intuitive software features for the users.

Appendix

Clickable links for the surveys below can be found in the 'docs' directory in the repo.

Needs Assessment Survey

1. https://docs.google.com/forms/d/190NSDpDtV7D5sHQITXaVXIlXctU-iyNdpzj3voLJg30/edit?usp=sharing

User Testing Survey

1. First Survey:

   https://docs.google.com/forms/d/16e817wz_YPAHgOinwbUQ5n8mkp89aNvACwzBOjoq-vU/edit?usp=sharing

2. Second Survey:

   https://docs.google.com/forms/d/1ygMuoIrlDV_R4Q_yTE4ogMJO2CBG04Wg0Ry5yOsooA4/edit?usp=sharing