# Phaze Blog

## November

- We started working on our application, Phaze. We installed Android Studio and began researching how to use it.
- In order to create accounts for the users, we found a NoSQL database known as Firebase which we decided to use for user Authentication and Authorization into the application. Having set up a Firebase Database, we developed a simple login feature for our application. The NoSQL aspect of Firebase proved to be useful for getting this feature up and running in the early development stages as there was no need for a "middle man" server communicating between the front-end and back-end and a Firebase database was quick and easy to set up and integrate.
- We began developing the main activity of our android application, creating 3 empty page fragments known as the Settings Fragment, Camera Fragment and User Fragment. This acts as the home page in our application and gives the user access to all of the application's features.
- We embarked on developing the main activity further, starting with the Camera Fragment. we created a camera view and set this as the default fragment to be viewed when the application is launched. This page uses the mobile phone's camera and allows the user to take a picture and display it within the application. One of the issues we encountered when developing this feature was that the photo size would often be too big to fit on the screen, which would cause the application to crash when displaying it. We solved this by finding the best fit screen size and setting the camera parameters to that before activating the camera object.
- While waiting for AI development, we worked on developing a user page on the main activity of our application. we implemented a statistical library from Github known as MPAndroidChart (https://github.com/PhilJay/MPAndroidChart ) to develop a pie chart that holds the user's nutritional information such as total calories and macronutrients ate. This allows the user to better visualise the nutritional value of their eaten food.
- We then created a User class that stores the user's recorded nutritional information. Every method and variable in this class is static, which restricts us from having multiple user objects (apart from the one which is logged in) and we are able to access this class from anywhere within the application. The pie chart from MPAndroidChart uses this class to update and display the user information accordingly.

## December

- We began developing a food classification feature using the Google Cloud Service called AI Vision
- We needed to retrieve the associated nutritional value of the recognised food. We found a food database known as Edamam (https://www.edamam.com/) and decided to use its API. We linked this with our application using a Java library known as OkHTTP in order to communicate with the database. This allowed us to retrieve the recognised food's nutritional information and display it once the AI is working correctly.
- Next, we needed a way to store the food under certain meals that the user would eat. We developed an additional feature that allows the user to add the food to a specific meal such as breakfast, lunch, dinner or snacks. We then updated the User class with the new nutritional information and displayed it on the user page using the MPAndroidChart library.
- While we were working on improving our Neural Network, we developed a search feature that gives the user the option to manually type in food into a search bar and retrieve its nutritional information, which also uses the OkHTTP library to communicate with the Edamam Database. This uses the same RequestFood class as the image classifier to make our application more maintainable and to reduce the amount of duplicate coding. We then used the same method as above to add it to the User class and display the information.

## January

- After developing a number of food classification models. We finally developed an image classification model, which was able to identify 5 food classes with over 80% accuracy, we decided to implement that within our android application by converting the model to a .tflite file. The hardest part about this process was learning how to translate the Python image resizing and rescaling code into Java. However, after a lot of research, we found a solution.
- Now that we had a working prototype of our application, we decided to work on some lower priority features. We created a feature that allows the user to adjust the serving sizes of their eaten food. The user can choose the serving sizes using multiple values such as grams, ounces, pounds as well as per item and save them to the User class.
- We implemented another activity that displays the food eaten on a given meal (E.G Breakfast) and their corresponding nutritional values that the user has selected.
- We created a delete option that allows the user to remove recorded foods from their diary and updates the User class. We tried implementing a slide to delete feature for the recorded foods, however, this proved way too

difficult and buggy to achieve and so the idea was replaced with a simple button.

## February

- After testing our new working CNN model which was able to identify 101 different foods with an accuracy of over 95%, we decided to implement it within our Android application. However, our old image resizes and rescale code did not work with this new model can give us inaccurate results. This forced us to rewrite that code from scratch.
- We implemented a barcode scanner, which gives the user the option to take a picture of a barcode and queries the Edamam database for that product and displays its information.
- We wanted to create a step counter which uses the mobile phone's internal sensor to detect movements. It then uses this information to calculate the number of calories burned. One issue with this was that not all Android phones contain this internal sensor and this caused the application to crash if we tried to access it. This was repaired by first checking if the user's mobile phone has this sensor before trying to use it.
- Now that most of the necessary functionality was finished, we focused on making the application more aesthetically pleasing by introducing small icons and images as well as updating the colour scheme and introducing a Bottom Navigation Bar feature that allows the user to swap between different page fragments.
- We realised that the CNN model could detect pictures more easily and accurately if they were brighter. To combat this we created a flash feature where the user can toggle between flash on and off before taking a picture.
- The last page that needed work was the Settings Fragment. We began working on a simple settings page that links our application to different links on our website. It also gives the user the ability to log out.
- Up to now, we have been using Firebase, however, due to the fact that it is a NoSQL database we needed a more powerful and scalable solution to storing information. We created an AWS SQL Database in order to store the user accounts and associated information such as nutritional information, meals eaten, etc. We then switched over to fully implementing the AWS Database for authentication, registration and updating and uploading information to and from the database. The Java OkHTTP library helped us communicate with the Python Flask Server hosted on Beanstalk to send and retrieve any necessary information. We also added a delete account feature to the Settings Fragment which allows the user to permanently delete their account from the Database.