

CA326

Phaze Blog

By Kevin Cogan

November 2020

15th - Started brainstorming ideas for our third-year project.

- Dieting App that uses some form of artificial intelligence.
- Mood tracking application that can tell the person sentiments based on facial recognition.
- Create an algorithm that can automatically complete a circuit through the uses of genetic algorithm/ machine learning algorithms. The cars will be able to work and be trained in different circuits.
- Create a web application that allows the user to adjust the textual and visual setting as well as adding additional features to help make any webpage accessible. This application should work on all web pages. This could be a chrome extension.

16th - General researched the feasibility and technologies used from the different ideas we brainstormed from yesterday:

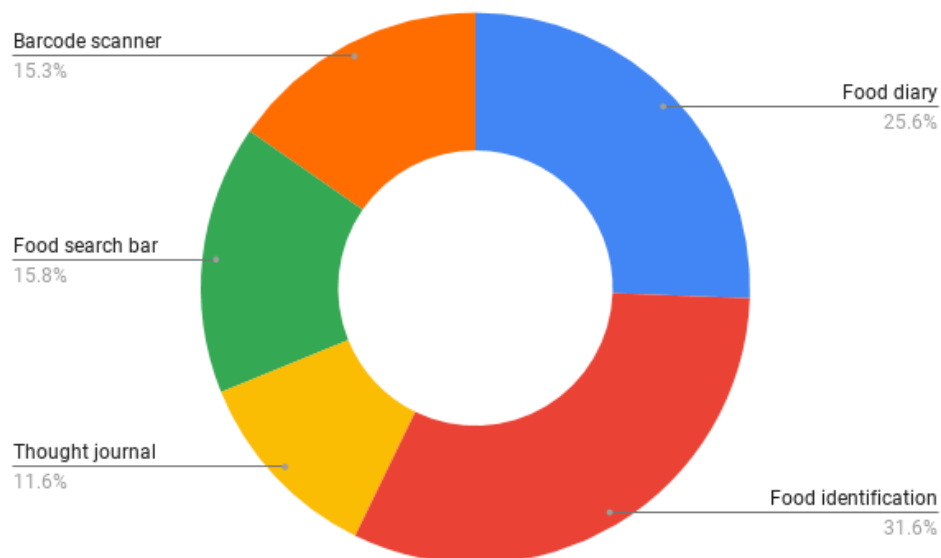
- **Dieting app:** we looked into what technologies would have to be used. We concluded that we will need some form of artificial intelligence, a database of food calories, and a server that can be hosted online.
- **Mood Tracking:** applications would need to use artificial intelligence with the potential use of a facial recognition library such as OpenCV. To get it to the users we can have the option to integrate this feature as a web application using HTML, CSS, and JavaScript or we can make it an android app with the use of Android Studio.
- **AI virtual Autonomous Car:** this would be a web-based application with the use of artificial intelligence such as neural networks or genetic algorithms. The visuals of the car and the circuit would be designed using HTML, CSS, and JavaScript. Another possibility is to use OpenCV to detect the circuit such as the black circle part of the image is the circuit and the green around the circle in the grass. We would use a colour picker along with artificial intelligence to allow the car to drive.

- **Web Page Layout Adjustment application:** We Will have to use React.JS that will interact with the DOM so that the webpage can be textual and visually adjusted. The application will have to be hosted on the google chrome store to be distributed to users. This means it has to comply with the google chrome store regulations and privacy policies.

17th - Selecting our idea:

- We selected the food identification application as it seemed the most feasible to develop and can scale up and add multiple different features over the development cycle.
- At the meeting, we identified problems such as:
 1. We need to get android applications
 2. Understand how to implement artificial intelligence in our application
 3. Set up a database with a database provider such as Amazon Web Services, Microsoft Azure, Google Cloud Services.
 4. How to connect the database with the android application.
 5. Where to find a database that contains micronutrient information about food.
 6. How to use Android Studios and the visual layout for the android application.

17th - We installed Android Studio and began researching how to use it. We also conducted a needs assessment of potential users to see what feature people would like to use in the app. The results can be seen below from the 15 participants that took part in the survey. This aided us to make a more informed decision on what features we should be developing.



18th - Purchased two different androids phones one with an old operating system and the other with the latest operating system. This will allow us to test the application on two different devices so we can potentially pick up any errors that may arise on either device.

19th - We began researching how we would begin developing the image classification feature with the use of artificial intelligence. We both agreed that we would approach the project in an agile fashion where we would have short sprints to get a task completed then conduct testing before we do the next iteration.

20th - I began researching using the Pytorch library to develop a Convolution Neural Network for image classification.

21th - The first step needed to create a Neural Network for image classification was that we needed to find a data set of images of food. We research a group called the U.S Department of Agriculture (<https://fdc.nal.usda.gov/>).

22nd - After analysing the data I downloaded it did not contain calorific information or images of food. I began searching for other ways to either get images of food or a database of calorific information about food.

23rd - I found a website called Kaggle (<https://www.kaggle.com/>) this website has a data set of 101,000 images of food for 101 food classes. The only way I could use the files was by getting the information via an API request. So I had to research how to make API requests.

24th - I started researching how to make an API request looking at various videos and websites:

1. Python Request Tutorial (<https://www.youtube.com/watch?v=tb8gHvYICFs>)
2. Python Request Library
(<https://www.nylas.com/blog/use-python-requests-module-rest-apis/#:~:text=How%20to%20Use%20Python%20Requests%20with%20REST%20APIs,-Now%2C%20let's%20take&text=The%20GET%20method%20is%20used,function%20to%20do%20exactly%20this.&text=The%20response%20object%20contains%20all,headers%20and%20the%20data%20payload.>)

25th - I downloaded the dataset into our coding environment successfully. Now I had to preprocess the images into a format that will be accepted by our neural network.

26th - I learned that the image had to be split into two folders, one called the training dataset folder and the other is a validation dataset folder. I had to divide the dataset around 75% of the image in the training dataset and the remaining goes into the validation dataset.

27th - I had issues with dividing the dataset into a format that would be accepted by our programme that I wrote with the PyTorch library so I continued to do more research into finding a solution.

30th - I was struggling to find a solution to try to find a format that will work with the PyTorch programme so I began to look for alternative solutions to the problem I have encountered.

December 2020

2nd - I found a premade image classification model on Google Cloud Services that allows users to identify images via an API request. The prediction model had an accuracy of 93%.

4th - I began building a python request script that will pass an image to the image classification feature by Google then the API returns the predicted items in the image.

```
import io
import os

os.environ["GOOGLE_APPLICATION_CREDENTIALS"]="Google-API-Service-Key/google_vision_service_key.json"

# Imports the Google Cloud client library
from google.cloud import vision

# Instantiates a client
client = vision.ImageAnnotatorClient()

# The name of the image file to annotate
file_name = os.path.abspath('resources/steak.jpg')
```

```
# Loads the image into memory
with io.open(file_name, 'rb') as image_file:
    content = image_file.read()

image = vision.Image(content=content)

# Performs label detection on the image file
response = client.label_detection(image=image)
labels = response.label_annotations

for label in labels:
    print(label.description)
```

7th - I began testing the Google image classification API on several images of food and I identified a problem that the images classification model was trained to identify any item so when I pass an image of a steak on a plate often the image classification API would return 'plate'. Furthermore, the label names for food were often too vague such as for an image of the steak it would often come up as 'meat'. I will now either have to come up with a solution to this problem or begin developing our image classification model.

I created a User class that stores the user's recorded nutritional information. The pie chart uses this class to display the user information.

10th - As I could not develop a solution to the Google image classification problem I decided to go back to developing our image classification neural network using a library. We began researching other libraries as we did not have much success with the PyTorch library. We found the Tensorflow library and began looking into the documentation of that library to see if we could create a neural network with the dataset we found earlier.

11th - We started our SRS documentation for the project and started researching based on the questions based on the template provided.

19th - Completed the SRS documentation.

23rd - Continued working on trying to preprocess the images using the TensorFlow library and understand the different functions in the Tensorflow library.

26th - Successfully processed the images into the training dataset folder and the validation dataset folder. Now we need to research processing the images further such as scaling them down and changing them into matrices.

27th - I learned the technical terminology behind image processing and image classification (<https://towardsdatascience.com/convolution-neural-network-for-image-processing-using-keras-dc3429056306>).

28th - I scaled the images down and converted them into matrices using the NumPy library. I also assigned the batch size based on research conducted on the internet.


29th - I had to create a model to train our neural network with the food dataset of images from the Kaggle API. I had the very basic knowledge of input, hidden, and output layers in the model however I did not fully understand the use of the model in practice. I also struggled at the beginning to understand the model functions in Tensorflow as this was the first time I have ever built a model. I decided to learn more about how to train a model and to attempt building one once I am more competent with the tasks that are required to build one.

January 2021

1st - After researching image classification neural networks models we began building a basic model so we ran the programme and saw how it behaved. We believed that the trial and error method would help us to understand the functions in the model better. After we had to compile the model before we began training the model. This was completed successfully and we trained our first neural network with our basic model with a validation accuracy of 20% for 10 epochs. We created a script that allowed us to test the image classification model on 300 images of food. This would present the food, the actual food label, and the predicted food label. We simply had to verify the labels correspond correctly. We plan to do this testing after we have created a model we believe we can use.

2nd - I started to graph the information such as the training accuracy and loss, and the validation accuracy and loss. This helped us to see the effectiveness of our model so that we would begin modifying the model to maximise the accuracy.

3rd - I began more layers to our neural network. The layers consisted of convolute 2D layers using the activation function 'relu' as it is proven to be effective with image classification. This



made these modifications greatly improved the accuracy as now I was getting an accuracy of upto 62% with two layers. I also noticed that when the accuracy and validation accuracy values were close the overall accuracy of predicting foods was high. I then tested the prediction with our automated testing process.

4th - I began adding more layers to the model this time I included maxpooling 2D followed by a dropout layer. At the end of the model code, I used a global average pooling layer followed by a dense layer with a 'softmax' activation function. In the final activation layer, I used a kernel regularizer that helps to penalise images that vary greatly from the other sets of images. Once the program was run for 200 epochs I noticed a great improvement of 84% for validation accuracy. We then wrote a quick program to test this with images from the internet. Once tested and the model was predicting the food well we decided to export the model by writing a simple shell script.


5th - I had to look for a food database with micronutrient information such as calories, fats, sugars, and proteins. I discovered a Food Database called Edamam (<https://developer.edamam.com/food-database-api-docs>). This API provided us with all the information on foods that I needed to create the features in our mobile application. I had to make a request to the Edamam API using an HTTP library. The documentation from Edamam was poor so I found code on GitHub that worked. Unfortunately, the code was written in python. So we converted the relevant python code into Java code. Once completed we successfully requested information from the Edamam API. After that, we just had to parse the relevant information from the retrieved JSON information.

10th - Kevin B created a basic Android studio login and registration page for our application. It simply takes the username and password of the user. I researched to find a database that will store the users' credentials so the next time they log into their account they can verify their credentials with the database. However, I do not know much about databases so I had to research different service providers such as Google Cloud Services, AWS, and Microsoft Azure.

11th - We found a Google Cloud Service called Firebase that was a no SQL database that allowed us to quickly and easily store users credentials in a cloud database. We set this service up using google documentation and successfully connected the application to the Firebase database.

13th - We realised the Firebase database was very restrictive and did not allow us to store the types of data we wanted so I decided to try to find other databases that provided better functionality.

We linked up the food classifier with the Edamam API and we were able to retrieve the recognised food's nutritional information and display it.



14th - I discovered the AWS Relational MySQL database that allowed me to have complete control over the information that was stored in the database. This method of setting up the database required more work to set up as I could not find a way to directly connect the database to the android application using java. Instead, I decided to create a server that acts as an API and it will query the database using a python connector then return the relevant data to the android application in JSON or string format.

15th - Today I set up the AWS MySQL database and used MySQL workbench to create a table and query from the AWS database to ensure that the database server was working correctly. After the successful tests, I decided to create a python script that would be able to query the database using a python connector. As well as this I was as working on trying to get the CNN to identify more food accurately


17th - I had issues querying the database however I discovered that the python connector version was not compatible with the older MySQL database. I fixed the problem by updating the MySQL database. I established a connection with the database today. Tomorrow I will look into trying to query the database using python connector SQL functions.

18th - I successfully queried the database using the Python connector. However, I encounter an issue trying to pass a variable into the SQL statement. So I decided to research the problem. I also wrote a python unit test script to test each of the edge conditions to find errors after making adjustments to codes. This was very beneficial as it reduces time doing regression testing.

19th - I found a way to pass variables into SQL statements and began looking into setting up a server to put the python connector queries.

20th - I created a Flask server to act as an API when the user is to query data such as in the login section of the application. The login verifies the username and password sent in by the user. The register section adds the username into the database if it is not in the database already and finally I create a delete account section that allows the user to delete all their information if they delete their account. This is fully functional on a local network testing the code with an online form and adding more tests to our unit test script. However, now I need to make it available to anyone around the world so I need to look into how to host the application online.

22nd - I will need to use an online web service to host our application such as AWS Elasticbeanstalk. After I host the application I will need to connect the android application with the hosted flask server via an HTTP request library in java such as OkHTTP.



We began doing our first user acceptance test with people of different age ranges and backgrounds. So we were conscious that most of our close connections come from a computing background who are very competent in computing so that would not be ideal for finding issues with the features or layout as they would be able to troubleshoot the problem by themselves. Instead, we targeted our connections from the non-technical fields such as business studies and people from an older age bracket as they would be more unfamiliar with technology.

23rd - Since Christmas, I have been trying to improve the neural network image classification model to identify more than just five food classes however after increasing the food classes the validation accuracy decreases and the time to train the model increases. It reached the point where it would take one day to test an adjustment to the model. This did not seem like a practical use of our time so I decided to look for potential ways to add as many foods as possible with the least amount of processing power and time.

24th - I found a method called Transfer learning that takes away the computational power and time to train a model. What this does is that it uses a pre-trained model to train our model with the food images that I pass in. This allows us to quickly train our model with all 101 food classes that I pass into the programme. Furthermore, it records an accuracy of 96% making our newly trained model accurate for all 101 food models. This is the perfect solution to our previous problem as I did not have the computational power such as RAM to train our neural network with one hundred and one thousand images.

25th - I am now looking into exporting our model and seeing how I can use the image classification model in the android studio application.

We found our participants and began giving them access to our first draft of the Phaze mobile application and survey.

26th - I began researching how to host a flask server on a hosting service such as AWS Elastic Beanstalk.

We then looked at the responses from our participant's surveys and it was clear that the users wanted a website for help and support, a flash feature on the application, more imagery, and a way to access the calorie and micronutrient information via the website

February

7th - After troubleshooting the problems with hosting the flask application on AWS Elastic Beanstalk. I have successfully hosted the API server that will query the database. I now just need to create a POST request script in java that will send the user information to be verified.

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.

12th - We realised that it was not secure to pass the username and password via the web link so instead I decided to use a form function that retrieves the post request in JSON format and will convert it to a string or integer in the flask API script. This required me to modify the code.

13th - I had issues trying to host the newly written code as the AWS Elastic Beanstalk server returned errors so we worked on debugging the issue. So I tried setting a Linux server using the Linode service provider.

14th - I discovered in the AWS Elasticbeanstalk dashboard there is a Logs link that will show all the commands run on the AWS server. This enabled us to see the error messages which quickly allowed us to solve the problem. After changing the database connector to a more common version (Python connector library) the was correctly hosted on the AWS server and I could call the API.

We created a flash feature where the user can toggle between flash on and off before taking a picture.

16th - I decided to store the users' micronutrient and calorie information on the cloud database so I had to add an update feature that allows users to add and take away values from their nutritional data stored online.

17th - Due to user feedback, I decided to create a website that will have information about the application and founders. I also wanted to put our privacy policy online so users can feel safe when using our application.

I also created a login, register, and delete function on the website that enables users to manage their account. Furthermore, I created a user page that allows the user to see their activity, micronutrient information and foods they have added to their food diary. Finally, from the survey

users wanted an easy way to contact the Phaze team for support so I created a contact form that allows us to see the query and to give the users a response accordingly.

20th - We have successfully linked the cloud database with the mobile application via the Flask API so the application can retrieve information from the database such as login and register information as well as calorie and macronutrient information.

21th - I created a new survey for the Phaze mobile application and the Phaze website for the second version application and our new website.

22nd - I contacted the same participants to take part in the second survey. I provided them with the new version of the mobile application and a link to the Phaze website.

25th - After reviewing the user feedback I noticed a considerable improvement in the responses from the users

28th - I worked on the Phaze graphics for the manual and website. I used my skills in Adobe Photoshop to make professional images.

March

1st - I began making minor improvements to the website such as adding a drop-down menu in the user page section.

3rd - We conducted our final integration testing between the components each of us is developing this included user testing and end-to-end testing.

4th - I conducted the final unit testing on the components such as the flask server and I ran a test on the website to ensure all elements were working properly.

5th - we conducted integration testing on the latest updates that we have made to ensure all the features are working properly together.

8th - We conducted end-to-end testing on our entire system to ensure it was working properly.

11th - We submitted all our work and documentation.