[redacted]

Middlesex University
Hendon, London

School of Science & Technology

Interim Report

CST3990 – Literature review & Initial steps

Author	***Supervisor***
[redacted]	[redacted]
Student ID	***Date***
[redacted]	[redacted]

Contents

Chapter 1: Literature Review

1.1 Introduction

Vision grants humans the luxury of perceiving and understanding the world around them and easily recognise objects. Computer vision on the other hand aims to mimic human vision by processing images (Sonka, 1993). The ability for a machine to recognise real-world objects resulted in the production of evolutionary systems such as autonomous vehicles, facial recognition, species detection and many other systems that play a major role in the advancement of today's society (Wäldchen, 2017). all these systems apply a common approach which mostly involves extracting 3D objects from the real world through hardware such as Cameras which then outputs two-dimensional images, resulting in enormous loss of data which in consequence increases the complexity of this area (Huang, n.d.,1996). In this literature, I will be reviewing publications that utilised Google Cloud's Vision AI to implement systems that use computer vision technologies for image processing. Due to Vision AI being released in 2015 the availability of publication on this particular technology is limited.

1.2 Relevant literatures

By its very nature, computer vision is a hard field to explore for many people as it involves understanding and producing complex computations. (Shinde et al., 2017) explored the potential of reducing power consumption from mobile devices that use intensive machine learning applications. The proposed solution comprises using Google Vision AI to scan business cards and extract their content, then provide a digital replica of it. They claimed that the sole reason for choosing Google Vision AI over other machine learning tools is that Google Vision would handle all computational operations on a server on the cloud which in consequence result in an energy-efficient system. in light of this, researchers developed a keen interest in such tool. A later study by (Rifiana Arief et al., 2018) outlined the challenges of large-scale digital document growth. To solve such a problem, they implemented a system to extract text from digital documents of students' thesis using Google Vision to eventually introduce an automation extraction framework. The study shows favourable use of Google Vision AI over other OCR technologies because of its 100% text recognition accuracy level.

Failure to comprehend financial concepts and risks that occur from such complications may cause a decision that could further harm a person's monetary condition. A study by (K. Saputra et al., 2019) proposes a solution for such a problem by implementing a mobile application that utilises Google cloud's vision AI to monitor and control financial spending. The functionality of the system works by extracting text from images of receipts that contain relevant content such as total, discount and change. The outcome of the proposed system outlines the effectiveness of such technology on user satisfaction, given a score of 3.35/4.

With the increasing interest in Google Vision, more researchers have shown keen interest in providing tools for visually impaired individuals. A study by (Bharatia, Ambawane and Rane, 2019) designed a smart stick that uses Google Vision to detect objects and faces. The smart stick includes a microcontroller that acts as the central control system, an ultrasonic sensor for obstacle detection. LDR sensor for car detection in dark environments, a camera module to capture images and a buzzer to provide noise alerts whenever an obstacle is detected. The system uses Bluetooth interfaced android application that handle all machine learning computations. The leverage this system has over other related tools is the decrease of workload over the system as most computations are handled by the android application separate from the primary system.it was also claimed that having most computations handled by a separate server optimises power consumption drastically. As a result, researchers used the system as a steppingstone. (Sugadev et al., 2019) proposed an obstacles avoidance solution for tough terrains using Google Cloud's Vision AI. The system uses a RaspberryPi camera from a raspberry pi mounted on an autonomous vehicle. The end goal was to eventually analyse the images and provide decisions for navigation. The study claims that the Vision AI not only provides details of what the objects in the image are, but in some cases the type of object is also given (soft or rigid) this helped shape the system to be more accurate. nonetheless, both studies outline the effectiveness of cloud-based computer vision tools such as Google Vision in order to improve the efficiency and power consumption of mobile applications that require intensive computation operations.

(Hernandez et al., 2019) exploit Google's Vision AI kit to implement a face detection system through a typical office camera that provides detailed and accurate information on people accessing certain areas within a workplace. The study focuses on the complexity of training such a model from scratch as well as the amount of time it takes for the model to be trained which was later identified that it took 6 hours to train the model on a small dataset. The training was done using Python and TensorFlow. (Hernandez et al., 2019) has mentioned the benefits of using Google's Vision AI kit by importance the existing model which allows for the ability to run machine learning applications with limited computational resources. These hypotheses identify the significance of utilising cloud-based machine learning tools as an alternative.
(Punia et al.,2020) developed a new method to detect coronavirus in hopes to differentiate between patients that are suffering from pneumonia using X-rays of the chest over thermal screening and the end goal was to produce a more efficient system of detecting coronavirus. However, it can be argued that the accuracy score of the model was between 66.67%-72.38% using the ResNet-34 model, which is poor due to the sensitive nature of the virus the model score must be high in order to provide

accurate predictions and avoid misdiagnosing a patient. (Bougourzi et al.,2021) on the other hand, used a different approach by utilising CT-scans using slice-level classification resulting in a higher model accuracy of 87.75%. both publications comment on the limitation of datasets available to train a more accurate model.

A later study by (Darma Putra et al., 2020) also explored the potential of utilising Google cloud Vision AI in the medical field. Darma claimed lack of systems for people unfamiliar with medical terms or recognising certain diagnoses themselves. Their proposed solution was to implement both a mobile and web application that uses different features of Vision AI such as label detection, object detection and web detection. the system takes an image of actinic keratosis for instance, the image is then analysed and a final prediction of what the medical diagnoses is shown in the image. The detection of diseases within images show promising results, however the model lacks accuracy in some cases such as images of pustular psoriasis. This study further outlines the potential advancements of such technology could have on the medical field and the world as a whole.

A study by (Sai Aishwarya Edupuganti et al., 2021) exhibits concerns for visually impaired individuals and the complication they face when consuming the medicine. Google Cloud Vision was also utilised in this use case to eventually implement a mobile application that uses the native camera to read out the text on medicine boxes so that visually impaired individuals can easily identify which medicine to consume without the need for any assistance. However, as the system deals with extremely sensitive data, the study fails to provide the accuracy of the system. Not measuring the system accuracy could lead to false information of the medicine to the user.

A more recent study exhibits concerns over the lack of computer vision assistive technologies for visually impaired individuals. (Bougourzi et al., 2021) utilised Google Cloud's Vision AI in combination with a RaspberryPi to process images and read out what has been detected to the user using text-to-speech technologies. Although the Vision AI is trained on large amounts of datasets, a study on the robustness of Google's Vision AI by (Hosseini, 2017) where consistently successful tests show that adding sufficient noise to images, results in an entirely inaccurate and different output. However, two years after (Hosseini, 2017)'s study was conducted. (Vaithiyanathan and Muniraj, 2019) applied Google cloud's Vision AI to extract text from images and use the voice-based output to give visually impaired users details of objects surrounding them, however, the difference in this study is that (Vaithiyanathan and Muniraj, 2019) were able to perform image pre-processing methods to remove noise and blur from the image to provide a more accurate detection. it can be argued that recent features of Vision AI indicate a more efficient way of implementing such technology through the usage of label detection where Vision AI analyses objects within an image and returns labels related to each object instead of only limiting the technology to extract text from images. such reasoning doesn't limit these hypotheses but expands on them to incorporate a self-idea.

(Shalva Thakurdesai et al., 2021) proposed a smart gallery management system that demonstrates the potential of Google's Vision AI. The main goal of the system is to allow users to upload images and then these images are classified and stored in a database with details of the content each image contains. These images can then be easily be searched using different objects within images. However, with large datasets, the system performance will see encounter intensive computations and slow image classification. an earlier study by (Markowitz, 2020) proposes a more robust solution for a similar problem that uses a different feature of the Vision AI known as product search that allows for a more efficient way of indexing images.

1.3 Conclusion

The purpose of this review was to outline the potential benefit of utilising Google Cloud's Vision AI to perform intensive machine learning tasks in a short amount of time rather than having to train a model that would be time-consuming to provide an accurate and efficient model. Most researches gathered provide compelling evidence of superior performance over alternative machine learning tools such as tesseract. The proposed solutions address the simplicity of implementing a complex application that deals with heavy machine learning tasks without having to comprehend intricate computations that a person would go through when working with machine learning models from scratch. In light of this, findings gathered lack exploration of potential features provided by Google Vision. Additionally, there is a lack of publication available for Google Vision AI, despite it being 6 years since the release of the technology. Therefore, I believe more research is required in this area to better demonstrate the effectiveness of such tool in today's society and potentially attract more users that shy away from machine learning applications due to its complicated nature, as well as provide a quality-of-life solution due to the lack of such technology in this area that uses Google cloud's Vision AI.

Chapter 2: Initial Steps

2.1. Overview

In this chapter, I will be walking you through the steps taken towards implementing the app and the current stage of the project. This chapter will identify the requirements for the functionality of the system as well as classify non-functional requirements to demonstrate how the system should deliver functionality. This section will also showcase designs that represent the flow and structure of the system.

2.2. System description

Computer vision is a complex area for many people. The process of training a model to recognise objects time-consuming as it involves collecting enormous amounts of data testing that data and adjusting hyperparameters to improve the accuracy of the model. With this in mind, programmers tend to shy away due to the complications it entails. With the increasing development of cloud applications. Performing machine learning operations has become much simpler with cloud tools such as Google Cloud's Vision AI and AWS Rekognition. These tools allow users to perform image classification tasks on models pre-trained on large amounts of datasets. In this project I will undertake the challenge of implementing a cross-platform mobile application that uses Google Cloud's Vision AI. The system will allow users to use the native camera to take a picture of a restaurant and extract text from the images and provide details of the place based on the content of the image. The system will also explore other features of Vision AI such as label detection to food images of places.

2.3. Google Vision AI

In 2015 Google introduced cloud-based image analysis API service defined as Vision AI. This API allows developers to easily integrate machine learning into their system, supporting features such as object detection, label detection, text detection face recognition and many other helpful features. The API can be interacted with through Java, python, node and C# or by making http request to its RESTAPI.

2.4. Requirements Specifications

2.4.1. Functional requirements

User's Perspective	System's Perspective
Users can create an account	Successfully authenticate user
Users can log in using third party auth providers e.g., Google	Only show the rest of the app when the user is authenticated
Users can search for a place	Store user details in Firebase
Users can use the camera to scan place	Obtain the user's current latitude and longitude
Users can set dietary preferences e.g., vegan, halal…etc	Use redux for higher-level state management
Users can see if the place serves food based on their dietary preferences	Resize image taken by the user to 640x6Users
User can view sentiment analysis on place reviews	Encode raw image to bUsers4
User can view images of scanned places place and filter based on objects in the image	Ensure the building is detected in images
User can see their scan history	Extract text blocks from images
	Display scanned place details to the user with sentiment analysis

2.4.2. Non-Functional requirements

- **Usability**: Interface must be minimal and easy to use
- **Performance**: Image classification mustn't exceed **5s**
- **Capacity**: Firebase storage mustn't exceed 20K reads and 50k writes
- **Compatibility**: App must be compatible with both IOS and Android
- **Documentation**: A well-written help and documentation section must be present
- **Scalability**: App must be able to handle large user demand

Name	Skanr **– An image classification app**
Description	This app will allow users to use their cameras to check if a restaurant serves food based on their dietary preference just by simply taking a picture of the front of the restaurant
Primary Actor	user scanning place
Scenario	A person is abroad and is trying to see if a restaurant serves halal food as well as see food images and reviews of the place. As the person is abroad, he is unable to search the restaurant's name as it is written in a different language. The person then opens Skanr uses the camera and takes a picture of the front of the restaurant
Precondition	User must be authenticated before using the camera
Trigger	From the home screen user presses the scan button from the bottom tab then on the camera screen user presses the capture button
Basic Flow	Once the user takes a picture of the front of a restaurant the app will classify the image and ensure a building is present, the text is then extracted from the image and is compared with places near the user's location. Once a match is found a results screen will show the details of the matched place along with an indication of the place serving food based on the user's preferences

Figure 1 – use case 1 – user scanning place

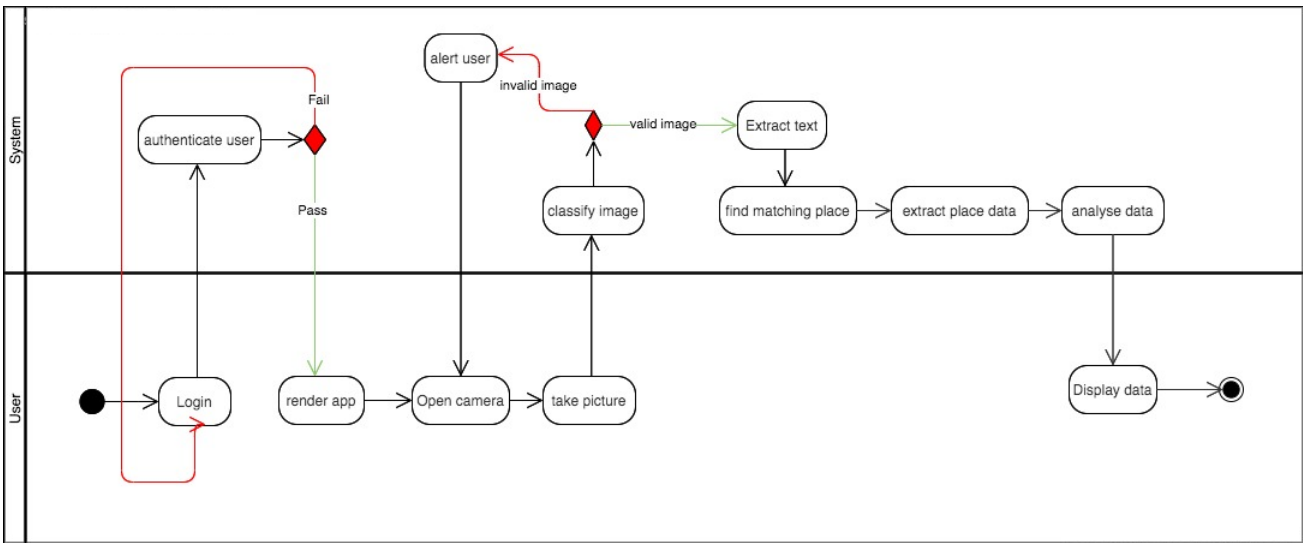
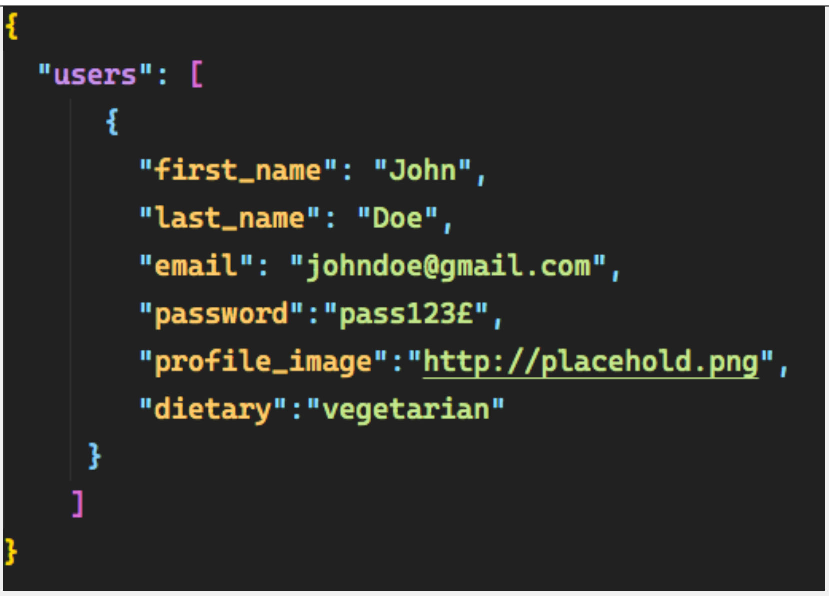
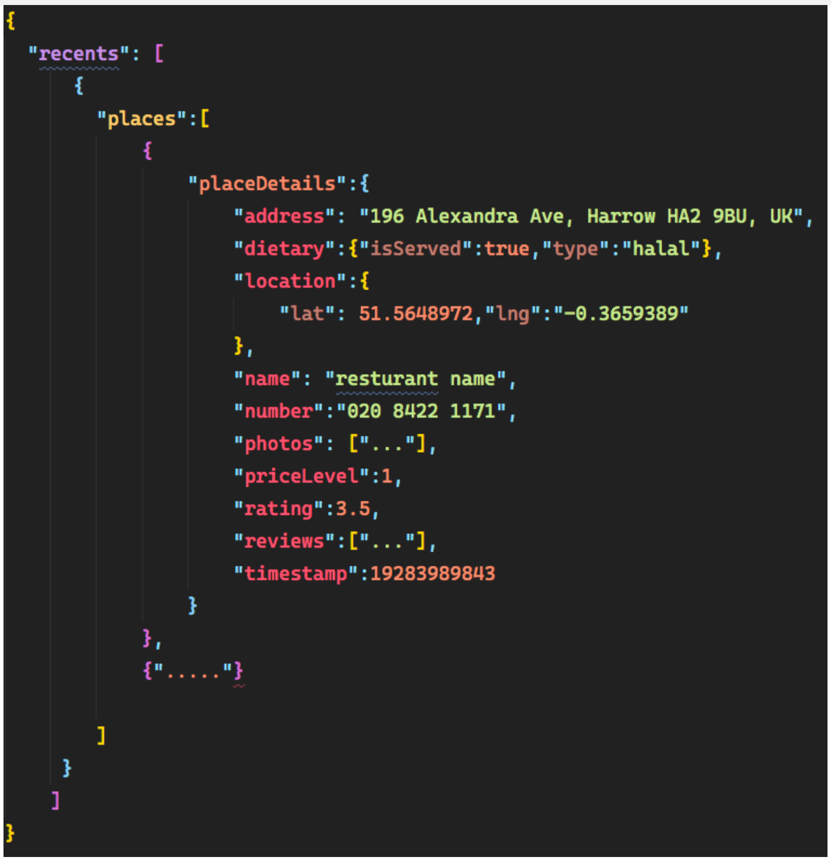


Figure 2 – Activity diagram of main functionality flow

2.5. Design

2.5.1 Database design

All data within the app will be stored in **Firebase’s** cloud storage in document format. The document will consist of two collections, a **user** collection and a **recents** collection.

Collection	***Document***
Users This collection will store information about each user such as their login details as well as their dietary preferences	
Recents This collection will store information about each place the user scans. It will include details such as the address, geographic location and other details shown on the right	

```
{
  "users": [
      {
        "first_name": "John",
        "last_name": "Doe",
        "email": "johndoe@gmail.com",
        "password":"pass123£",
        "profile_image":"http://placehold.png",
        "dietary":"vegetarian"
      }
    ]
}
```

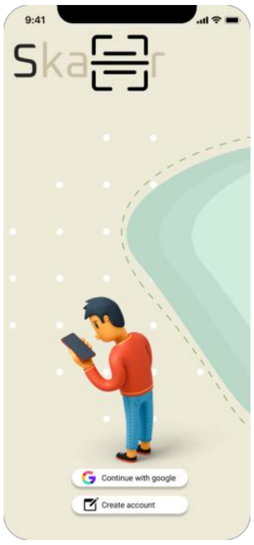
```
{
  "recents": [
    {
      "places":[
        {
          "placeDetails":{
            "address": "196 Alexandra Ave, Harrow HA2 9BU, UK",
            "dietary":{"isServed":true,"type":"halal"},
            "location":{
              "lat": 51.5648972,"lng":"-0.3659389"
            },
            "name": "resturant name",
            "number":"020 8422 1171",
            "photos": ["..."],
            "priceLevel":1,
            "rating":3.5,
            "reviews":["..."],
            "timestamp":19283989843
          }
        },
        {".....”}

      ]
    }
  ]
}
```

2.5.2 App UI Design

Using **Figma** I have created concept designs of the user interface for each screen.

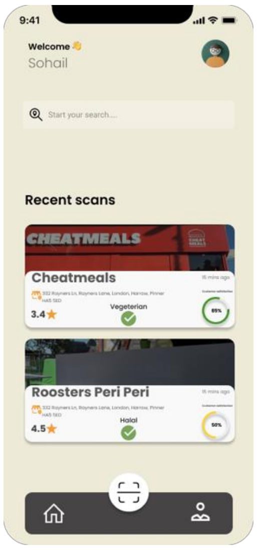
(1) Login Screen

This is the first screen the user is taken to after they have been authenticated. In this screen, the user is able to see their recent scans if there is any as well as navigate to either the account tab or open the camera screen



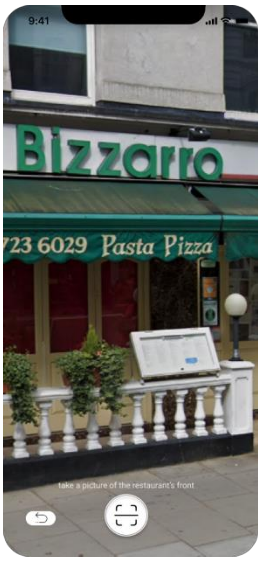
(2) Home Screen

This is the first screen the user is taken to after they have been authenticated. In this screen, the user is able to see their recent scans if there is any as well as navigate to either the account tab or open the camera screen



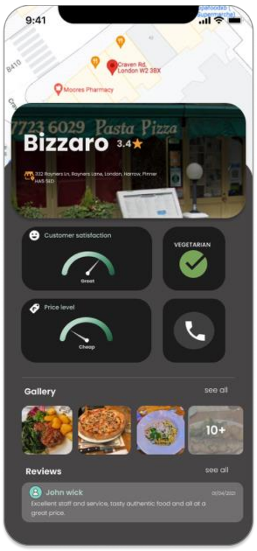
(3) Camera Screen

This screen opens up the camera view to the user with options to take a picture of the current view. The user also has the option to return back to the home screen.



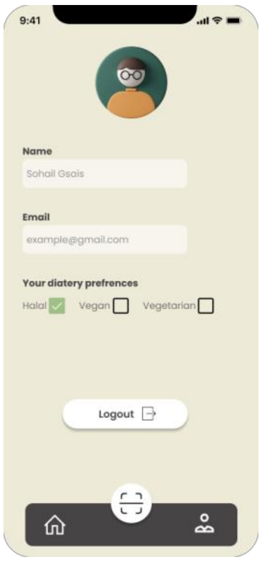
(4) Results Screen

This screen will show once the image classification is finished. The screen will display to the user the results of the place scanned from the image and indicate if the place serves food based on their set preferences. The screen will also show other details of the place



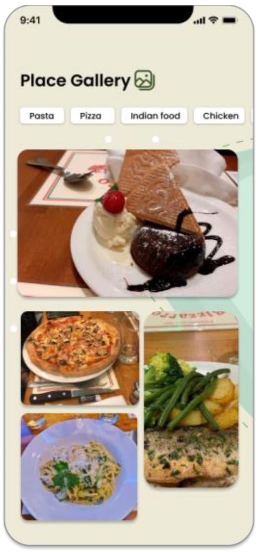
(5) Account Screen

This screen displays to the user the details of the currently logged in account as well as the options to update their details or log out of the account.



(6) Gallery Screen

This screen displays images of dishes posted by user within the google database that has been classified and filtered based on the objects detected within the image.



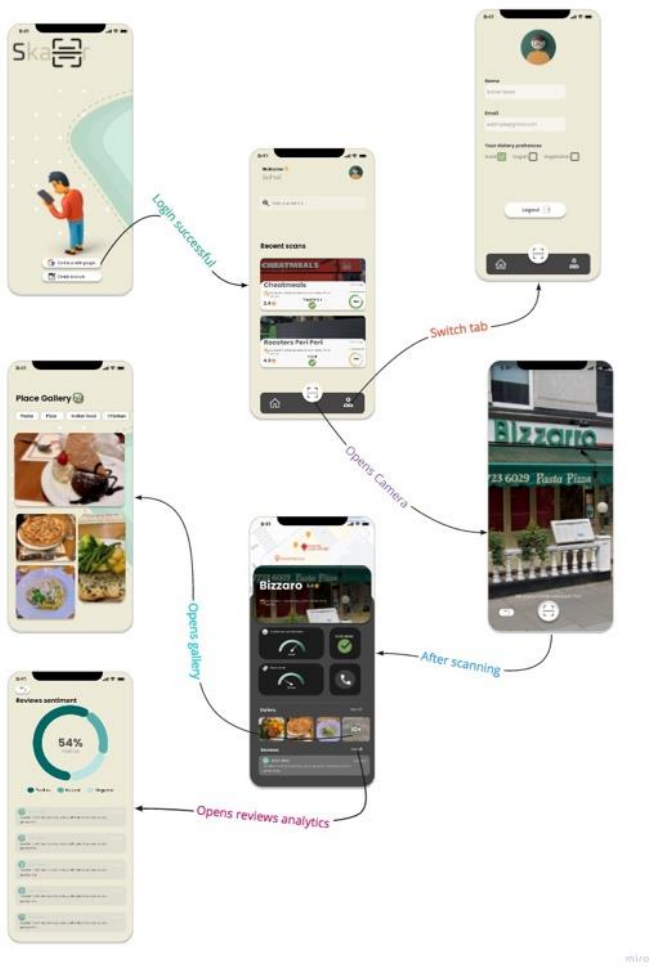


Figure 3 – Higher level system flow

2.6. Analysis of relevant systems

2.6.1 Google lens

The Google lens is a feature built within the google app that uses neural networks and pretrained models to classify images and detect objects within an image (Shapovalov, 2019).thus this feature provides endless possibilities, and it makes people’s life easier just by simply pointing their camera at an object around them and Google lens provides information based on what it detects. However, having tested these features there are some flaws and inaccuracies within the image classification. more specifically the place detection feature. As it does not appear to provide accurate details of the scanned place, but it only presents similar images. In **figure–4** the screen represents what Google lens detected when I took a picture of a place and chose to provide details of that place. The results given was only for similar images of the text and objects detected in the image.

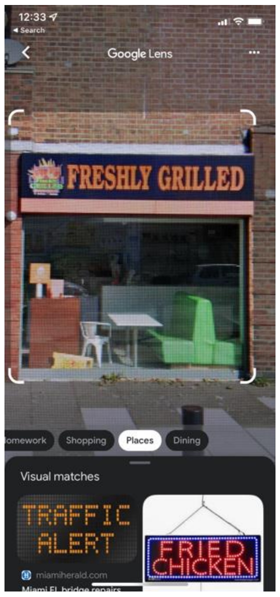


Figure 4 – Google lens place test flow

2.7. Current Stage

2.7.1 Project Setup

The initial steps taken to setup the project involved deciding which technology is best suitable for the given task. Thus, implementing native applications specific to each operating system is a time-consuming process. IOS applications, for instance, are built using Swift or Object-c and these applications can only be used for IOS devices, whereas Android applications are mostly written in Java. Therefore, building a cross-platform application using this approach means writing the same application for two different code bases (Fentaw, 2020). React Native on the other hand allows for a simpler implementation of cross-platform applications as the application is written in a single codebase and shared for both IOS and Android. With this in mind, it became clear that the best option for me is to implement the app using react-native.

2.7.2 Technology Stack

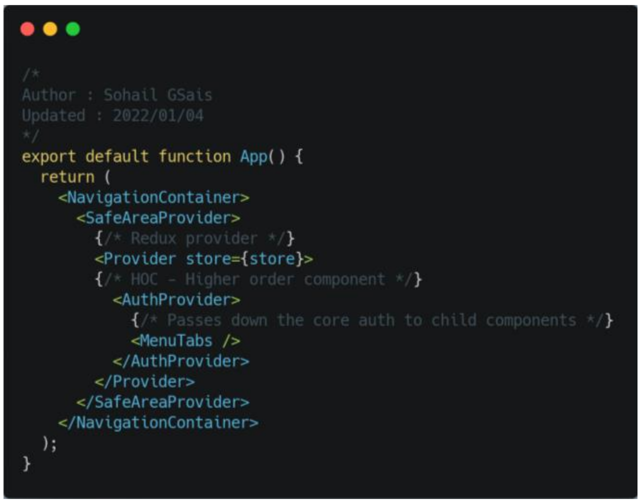
- **React Native**: Language used to implement native app
- **Redux:** Higher level component used as a global store for react-native
- **tailwindcss**: CSS library responsible for all UI styling
- **Firebase**: NoSQL database hosted on google cloud's firestore
- **GCS**: Google Cloud Services
    - **Vision AI**: used for image classification and text detection
    - **Places API**: used to gather place details
- **Sentiment API**: used to analyse sentiment from place reviews

2.7.3 Implementation

Authentication

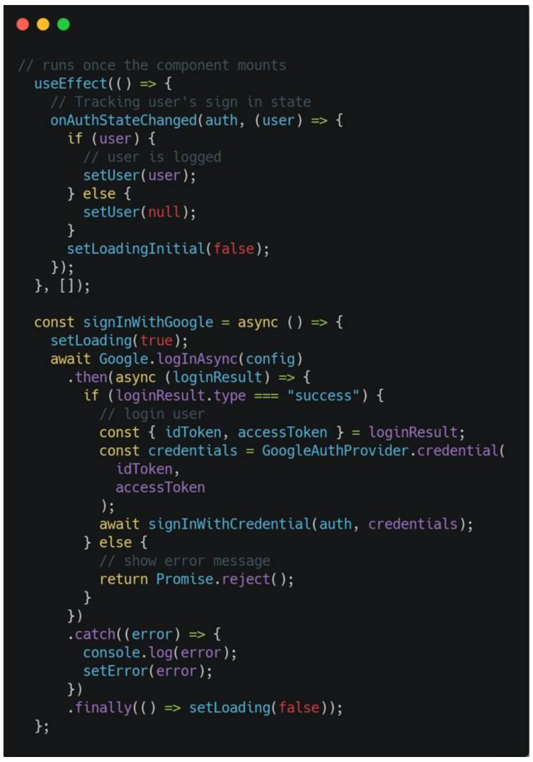
Some functionalities of the app require differentiating between users, as dietary preferences vary from one user to another. Thus, I have implemented two different authentication options. The first option uses Google cloud's different auth providers as a wrapper for the app which behaves as a higher-order component that renders the rest of the app as long as the user is authenticated.

Instead of reinventing the wheel, React-native makes it easy to encapsulate components and allow them to be shared throughout the project using an approach referred to as react hooks, which in simple forms are functions that can be used inside of react's functional components. Hence to be able to check if a user is logged and have access to that user's details through the app (Bugl, 2019). I implemented my own react hook. Figure-6 shows the code structure of the hook. The **useEffect** method is a pre-built hook by react that gets triggered once the component mounts. And in my case, the useEffect is triggered to track the user's login state, if a user is logged in then their details are shared throughout the app. The **signInWithGoogle** method in figure-6 handles the logic of authenticating the user by extracting the access token and the ID token once the user chooses a Google account to login with.



```
/*
Author : Sohail GSais
Updated : 2022/01/04
*/
export default function App() {
  return (
    <NavigationContainer>
      <SafeAreaProvider>
        {/* Redux provider */}
        <Provider store={store}>
        {/* HOC - Higher order component */}
          <AuthProvider>
            {/* Passes down the core auth to child components */}
            <MenuTabs />
          </AuthProvider>
        </Provider>
      </SafeAreaProvider>
    </NavigationContainer>
  );
}
```

Figure 5 – App components



```
// runs once the component mounts
  useEffect(() => {
    // Tracking user's sign in state
    onAuthStateChanged(auth, (user) => {
      if (user) {
        // user is logged
        setUser(user);
      } else {
        setUser(null);
      }
      setLoadingInitial(false);
    });
  }, []);

  const signInWithGoogle = async () => {
    setLoading(true);
    await Google.logInAsync(config)
      .then(async (loginResult) => {
        if (loginResult.type === "success") {
          // login user
          const { idToken, accessToken } = loginResult;
          const credentials = GoogleAuthProvider.credential(
            idToken,
            accessToken
          );
          await signInWithCredential(auth, credentials);
        } else {
          // show error message
          return Promise.reject();
        }
      })
      .catch((error) => {
        console.log(error);
        setError(error);
      })
      .finally(() => setLoading(false));
  };
```

Figure 6 – authentication hook

Algorithm

1. Capture screen
2. Resize the image to 640x640 pixels
3. Convert raw image to base64
4. POST request to Vision AI to image classification
5. Process response data
6. Check if the image contains a building label
7. Get nearby places based on user location
8. Loop through the array of text blocks extracted from the image
9. Loop through the array of places details nearby
10. Check if place name matches extracted text
11. If match is found place details are saved.

Image Classification

Before I have implemented the image classification logic, I. needed to Implement the camera logic. In which case I used Expo's Camera component and setup the required permission needed to give the app access to the camera along with simple controls needed for the camera such as picture button and scrollable filter options shown in **figure 7**. Once the camera logic was complete the next step was to implement the image classification logic using the Vision AI.

Image Pre-processing

As the Vision AI is a new tool the documentation was mostly aimed at application built natively with SDKs only available for Objective-C and Swift. Hence there's no available module for react native. However, the AI can be interacted with using **RESTful** API calls however before making a request there are some requirements to consider, the size of the image, the encode type and which detection feature(s) is needed. For instance, the image needs to be no more than 640x640 pixels. If we put this into perspective, the latest iPhone 13 screen resolution is 1170x2532 pixels therefore once the user takes a picture the camera will capture the entire screen meaning that the image size will around 1170x2532 which is too large and will affect the performance of the classification, and making the user resize the image doesn't seem like an efficient solution. Therefore, I have implemented a method with the help of an **Expo** library that allows fast and easy image manipulations. The method **resizeImage** in **figure 8**b resizes the image down to 500x500 right after the image is captured once the image is resized the method returns an object containing the URI of the resized image and the new width and height. Once the image is resized the final step is to convert the raw image to a **base64** simply by calling the **readAsStringAsync** within Expo's Filesystem by passing it the URI of the reseized image and the encoding type.

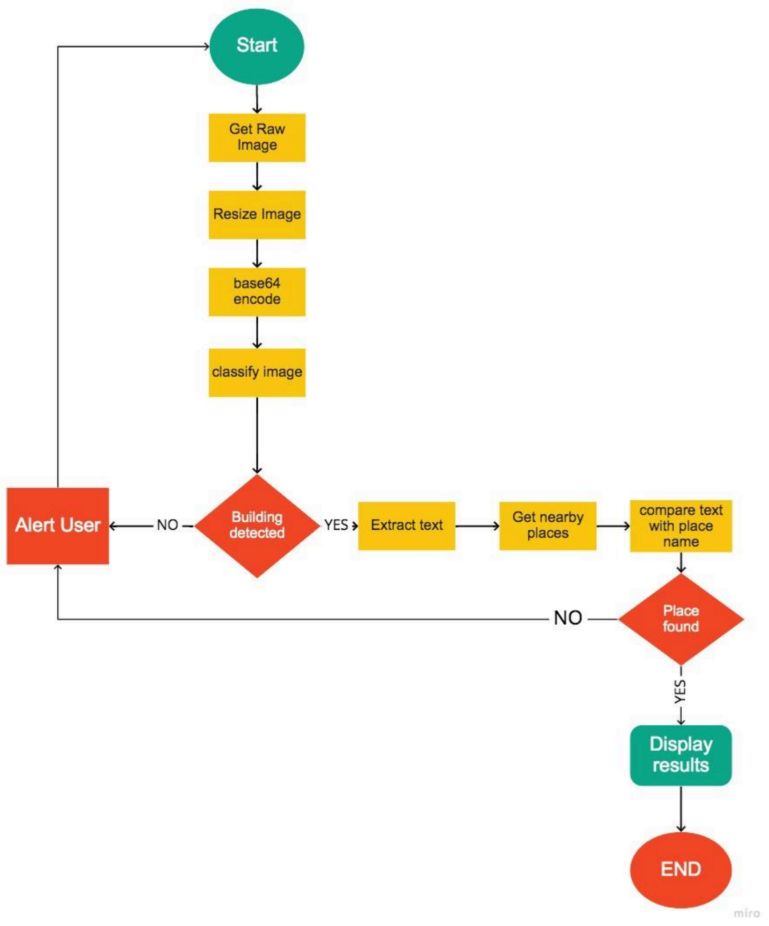
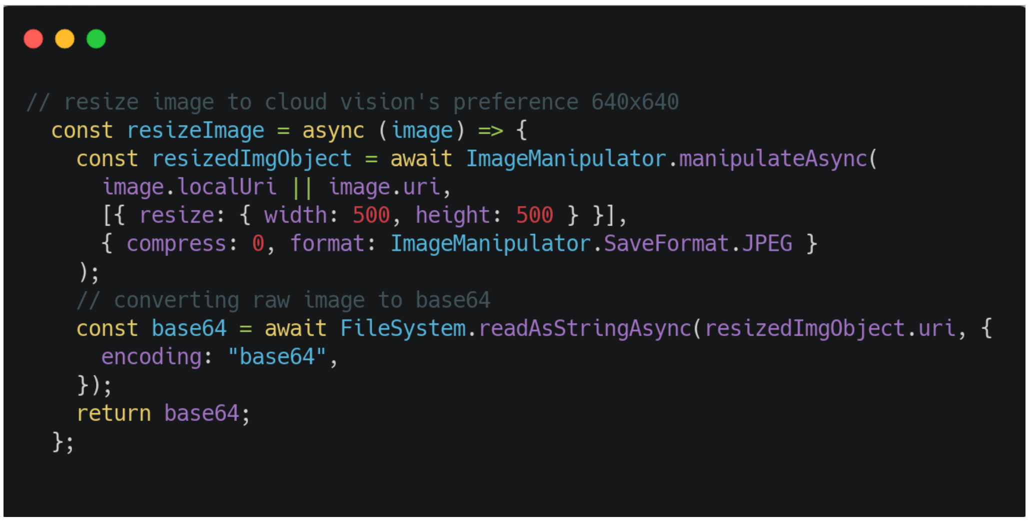


Figure 8(a) – flowchart of image processing algorithm



```
// resize image to cloud vision's preference 640x640
  const resizeImage = async (image) => {
    const resizedImgObject = await ImageManipulator.manipulateAsync(
      image.localUri || image.uri,
      [{ resize: { width: 500, height: 500 } }],
      { compress: 0, format: ImageManipulator.SaveFormat.JPEG }
    );
    // converting raw image to base64
    const base64 = await FileSystem.readAsStringAsync(resizedImgObject.uri, {
      encoding: "base64",
    });
    return base64;
  };
```

Figure 8(b) – Resize image / base64 encoding

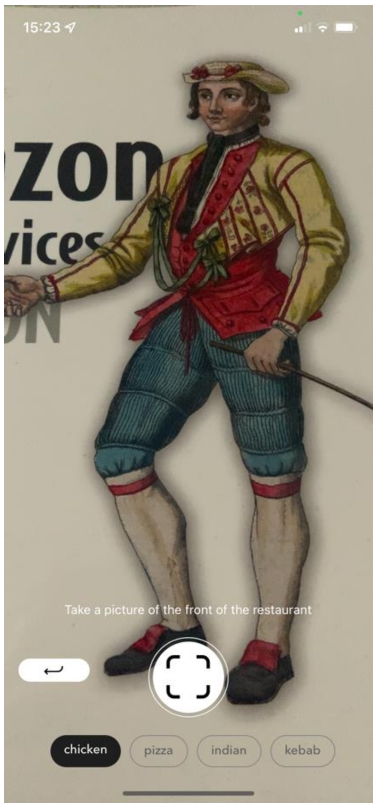
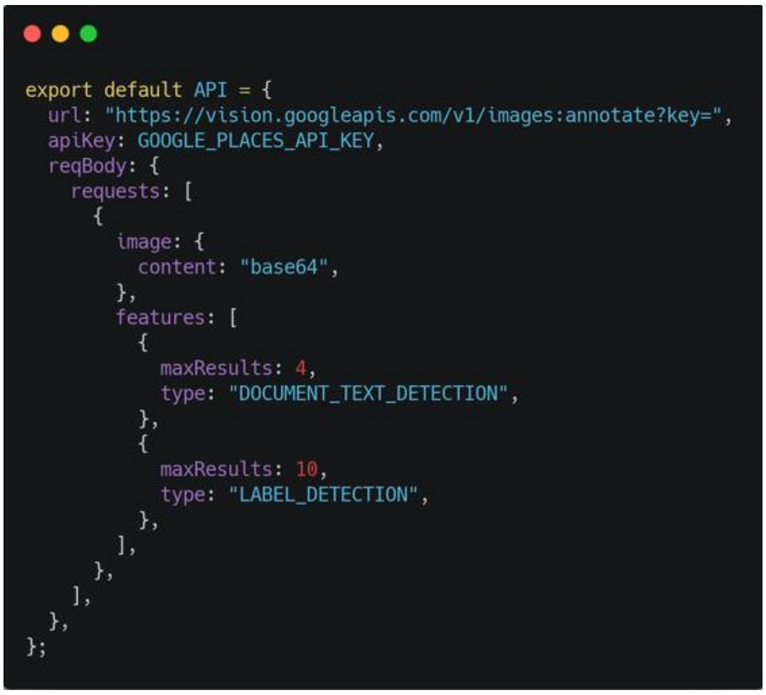


Figure 7 – Camera screen controls

Text/Label detection

Once image pre-processing is complete the final step is to make a POST request to the Vision AI with the image in the format shown in **Figure 9**, as my goal is to extract text from the image, one of the feature detections I needed to include in the request body is TEXT_DETECTION along with LABEL_DETECTION, the label detection feature will return labels detected within the image. One of the labels will include either a shop or building which will be used to ensure that the image is valid shown in **Figure 10**, and the user is not taking a picture of a random object, hence this provides accuracy to the image classification. Once the image classification is complete the API responds with an array of objects, a text annotation and label annotations, the label annotations objects contains a maximum of 10 labels that have been detected within the image, a bounding box coordinated for each object and their confidence score, these labels are used to ensure that a building has been detected in the image, see **Figure 10**. The same structure is also returned for the text annotations with each text block in the image. See **Figure 11**.

Once all text blocks are extracted. The next step is to match the text extracted with places near the user's current location as in most cases the user will be around the location of the place being scanned. Therefore, I only needed to match text blocks with the names of nearby places. To achieve this, I created a second react native hook to keep track of the user's current location, in **Figure 12** I have used Expo's location library which enables us to get details of user's current geographical coordinates including their latitude and longitude. Having these details helps fetching relevant places within the given coordinates as well as details on each place such the name, address, reviews, photos and other information. In this case the name field is what we need in order to compare the text extracted from the image with the names of places nearby and get the place with name that is found in the text from the image. In **Figure 13** the logic is to loop through the array of extracted text blocks and compare them with the array of nearby place names and return the matching place.



```
export default API = {
  url: "https://vision.googleapis.com/v1/images:annotate?key=",
  apiKey: GOOGLE_PLACES_API_KEY,
  reqBody: {
    requests: [
      {
        image: {
          content: "base64",
        },
        features: [
          {
            maxResults: 4,
            type: "DOCUMENT_TEXT_DETECTION",
          },
          {
            maxResults: 10,
            type: "LABEL_DETECTION",
          },
        ],
      },
    ],
  },
};
```

Figure 9 – Vision AI request body

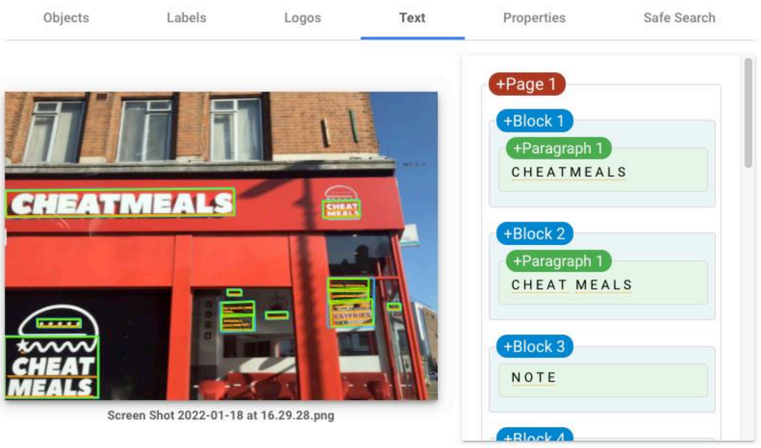
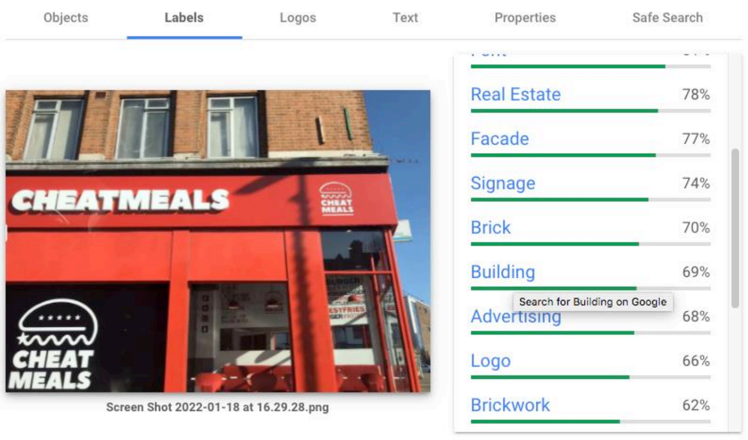
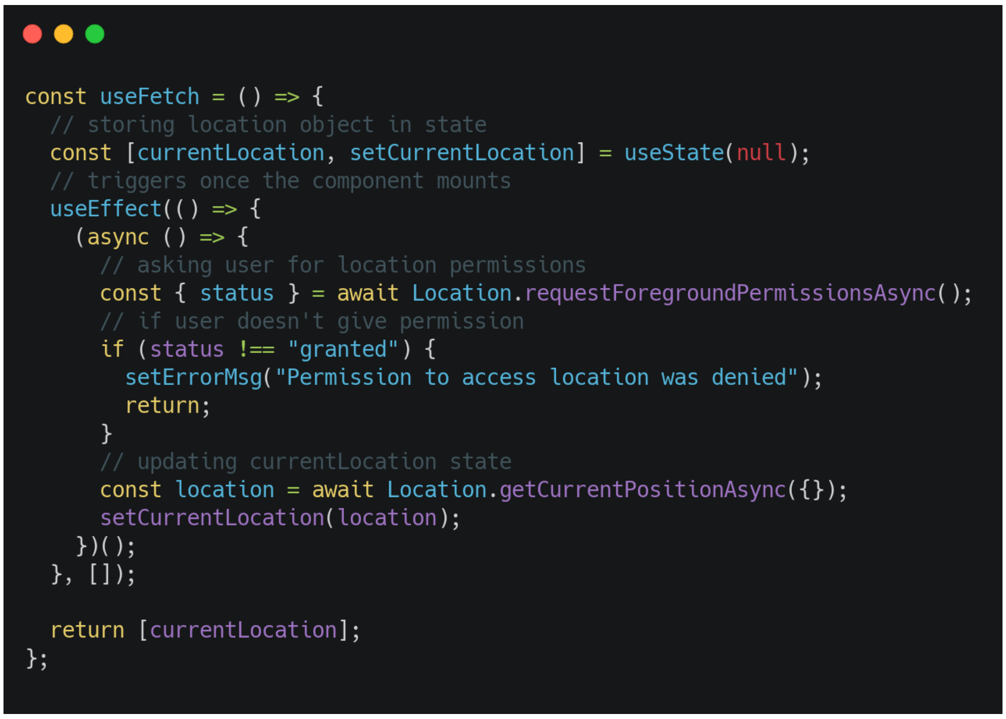


Figure 11 – Text detection from Vision AI


Objects
Labels
Logos
Text
Properties
Safe Search
Screen Shot 2022-01-18 at 16.29.28.png
Search for Building on Google

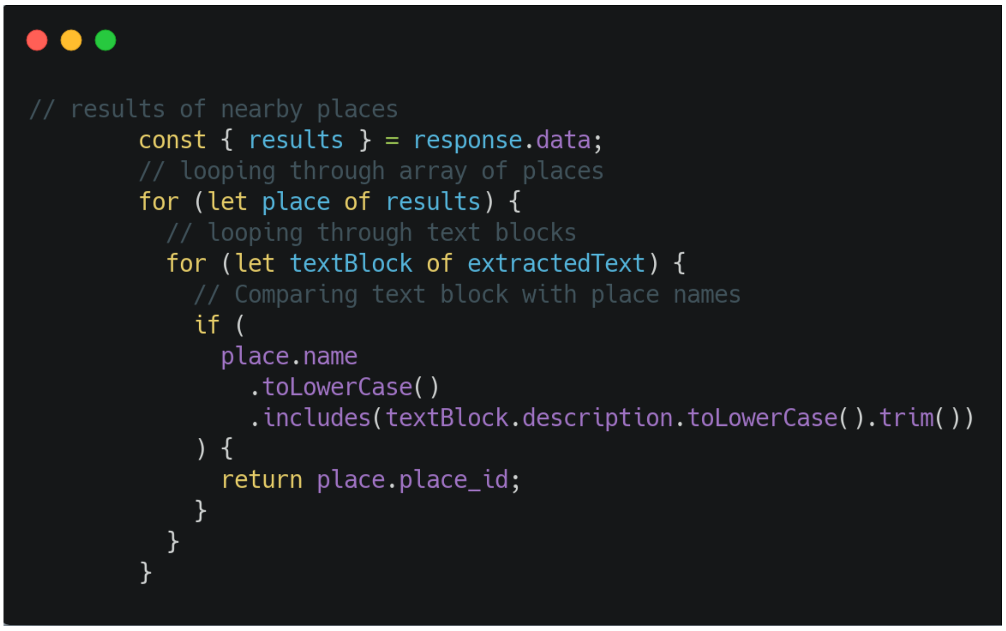
Real Estate	78%
Facade	77%
Signage	74%
Brick	70%
Building	69%
Advertising	68%
Logo	66%
Brickwork	62%

Figure 10 – Label detection from Vision AI



```
const useFetch = () => {
  // storing location object in state
  const [currentLocation, setCurrentLocation] = useState(null);
  // triggers once the component mounts
  useEffect(() => {
    (async () => {
      // asking user for location permissions
      const { status } = await Location.requestForegroundPermissionsAsync();
      // if user doesn't give permission
      if (status !== "granted") {
        setErrorMsg("Permission to access location was denied");
        return;
      }
      // updating currentLocation state
      const location = await Location.getCurrentPositionAsync({});
      setCurrentLocation(location);
    })();
  }, []);

  return [currentLocation];
};
```

Figure 12 – useLocation hook for user’s location



```
// results of nearby places
        const { results } = response.data;
        // looping through array of places
        for (let place of results) {
          // looping through text blocks
          for (let textBlock of extractedText) {
            // Comparing text block with place names
            if (
              place.name
                .toLowerCase()
                .includes(textBlock.description.toLowerCase().trim())
            ) {
              return place.place_id;
            }
          }
        }
```

Figure 13 – Matching extracted text blocks from images with place names

Displaying Results

Once a match is found for a given place the results screen is then shown to the user with details of the place. However, the issue I encountered in this stage is that the results screen is not a child component of the camera screen, meaning that both Camera and Results screen are parent components therefore data from camera screen cannot be passed directly to the results screen. This is where Redux plays a very valuable role. At a high level, redux is a state management tool, and it allows for easy access to data from any component within the application often referred to as a “store”. Most stores consist of three main tools, slices, reducers and selectors as illustrated in **Figure 14**.

There can only be a single store within a react application. A slice combines all logic used for each reducer. Reducers handle the logic of updating the value of a state given the action and its previous state (Kudiabor, 2020). With this logic we can then use redux to store the place data in the global store from the camera screen which can then be retrieved from the results screen from the global store by referencing the relevant state from the reducer. This data is then extracted and displayed to the user, however as not all functionality has been implemented yet only a few details are displayed as illustrated in **Figure 15**

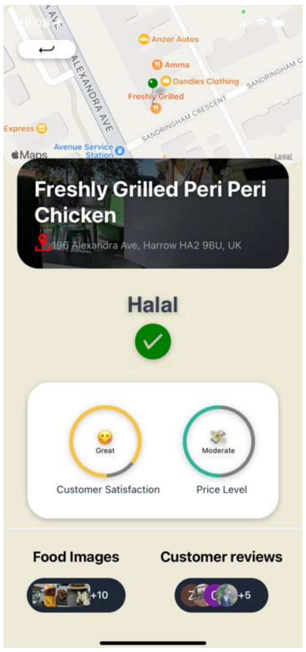
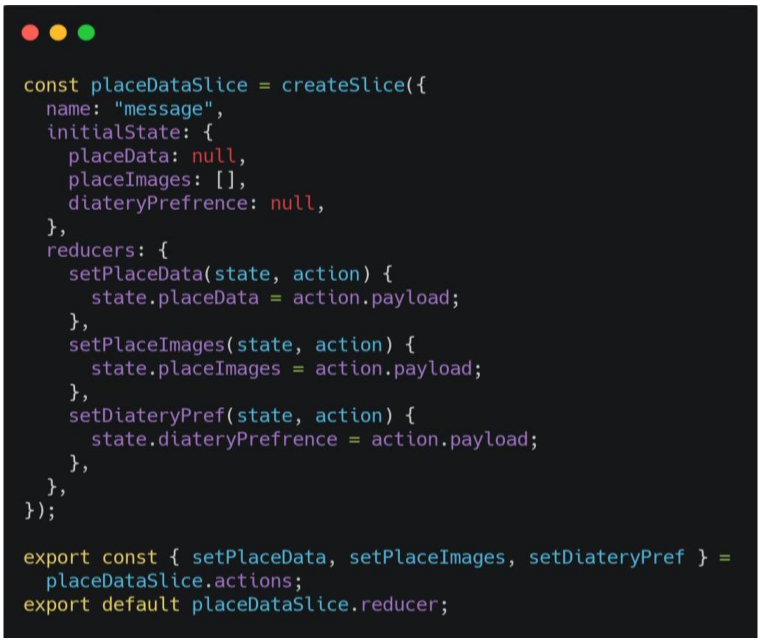


Figure 14 – Results screen showing place details (unfinished)



```
const placeDataSlice = createSlice({
  name: "message",
  initialState: {
    placeData: null,
    placeImages: [],
    diateryPrefrence: null,
  },
  reducers: {
    setPlaceData(state, action) {
      state.placeData = action.payload;
    },
    setPlaceImages(state, action) {
      state.placeImages = action.payload;
    },
    setDiateryPref(state, action) {
      state.diateryPrefrence = action.payload;
    },
  },
});

export const { setPlaceData, setPlaceImages, setDiateryPref } =
  placeDataSlice.actions;
export default placeDataSlice.reducer;
```

Figure 14 – Redux place data slice

2.8 Next steps

- Implement reviews screen
- Analyse reviews and display sentiment
- Implement Gallery screen for food image classification
- Improve accuracy of string-matching algorithm for place finding
- Improve scalability of app to be used with foreign text
- Measure performance of image classification
- Perform unit tests

2.9 Updated Project Timeline

DEADLINE	SUB-TASK	MILLSTONE	DELIVERABLE
~~WEEK 6~~	~~Pitch idea to supervisor~~ ~~Finalise project idea~~ ~~Finish proposal draft~~	~~Proposal Draft~~	~~Project Proposal~~
~~WEEK 7~~	~~Review draft with supervisor~~ ~~Improve upon feedback~~ ~~**Submit Proposal**~~		
~~WEEK 8~~	~~Gather Relevant Literatures~~ ~~Evaluate gathered literatures~~	~~Background~~	~~Literature Review~~
~~WEEK 9~~	~~Identify requirements specification~~ ~~Produce test plan~~ ~~Finalise system requirements~~	~~Requirements~~	
~~WEEK 10~~	~~Investigate similar systems~~ ~~Design sequence diagram~~ ~~Design database structure and tables~~ ~~Design UI in Figma~~	~~Analysis & Design~~	
~~WEEK 11~~	~~Setup React native with expo~~ ~~Setup GitHub repository~~ ~~Setup Google Cloud Services~~ ~~Implement App front-end~~ ~~Review Front-end with supervisor~~	~~Initial Implementation~~	
~~WEEK 12~~	~~Improve front-end upon feedback~~ ~~Review literature with supervisor~~ ~~Improve literature upon feedback~~ ~~**Submit Literature**~~		
WEEK 13	Continue working on front & back end Perform Unit tests & verifications Perform final build	Finished App	Final Report
WEEK 14	Complete app development		

WEEK 15	Perform UI/UX evaluation	System Evaluation
WEEK 16	Evaluate system performance	
WEEK 17	Work on final Report	Report Draft
WEEK 18	Complete final report	
WEEK 19	Review report draft with supervisor	
WEEK 20	Improve upon feedback	
WEEK 21	Improve upon feedback	
WEEK 22	Prepare Application for demo	Viva & Demonstration
WEEK 23	Prepare for Viva presentation	
WEEK 24	**Submit Final report**	

3.1 Relevant sources

- Link to GitHub repository [redacted]
- App can be viewed here [redacted]

Research Ethics Screening Form for Students

Middlesex University is concerned with protecting the rights, health, safety, dignity, and privacy of its research participants. It is also concerned with protecting the health, safety, rights, and academic freedom of its students and with safeguarding its own reputation for conducting high quality, ethical research.

This Research Ethics Screening Form will enable students to self-assess and determine whether the research requires ethical review and approval via the Middlesex Online Research Ethics (MORE) form before commencing the study. Supervisors must approve this form after consultation with students.

Student Name: [redacted]		Email: [redacted]
	Research project title: **Utilising Google's Vision AI for image classification**	
	Programme of study/module: **CST3990 Undergraduate Individual Project**	
Supervisor Name: [redacted]		Email: [redacted]

Please answer whether your research/study involves any of the following given below:		
1. HANIMALS or animal parts.	☐ Yes	[✓ No]
2. MCELL LINES (established and commercially available cells - biological research).	☐ Yes	[✓ No]
3. HCELL CULTURE (Primary: from animal/human cells- biological research).	☐ Yes	[✓ No]
4. HCLINICAL Audits or Assessments (e.g. in medical settings).	☐ Yes	[✓ No]
5. XCONFLICT of INTEREST or lack of IMPARTIALITY. If unsure see "Code of Practice for Research" (Sec 3.5) at: https://unihub.mdx.ac.uk/study/spotlights/types/research-at-middlesex/research-ethics	☐ Yes	[✓ No]
6. XDATA to be used that is not freely available (e.g. secondary data needing permission for access or use).	☐ Yes	[✓ No]
7. XDAMAGE (e.g., to precious artefacts or to the environment) or present a significant risk to society).	☐ Yes	[✓ No]
8. XEXTERNAL ORGANISATION – research carried out within an external organisation or your reseach is commissioned by a government (or government body).	☐ Yes	[✓ No]
9. MFIELDWORK (e.g biological research, ethnography studies).	☐ Yes	[✓ No]
10.HGENETICALLTY MODIFIED ORGANISMS (GMOs) (biological research).	☐ Yes	[✓ No]
11. HGENE THERAPY including DNA sequenced data (biological research).	☐ Yes	[✓ No]
12. MHUMAN PARTICIPANTS – ANONYMOUS Questionnaires (participants not identified or identifiable).	☐ Yes	[✓ No]

13. XHUMAN PARTICIPANTS – IDENTIFIABLE (participants are identified or can be identified): survey questionnaire/ INTERVIEWS / focus groups / experiments / observation studies.	☐ Yes	☒ No
14. HHUMAN TISSUE (e.g., human relevant material, e.g., blood, saliva, urine, breast milk, faecal material).	☐ Yes	☒ No
15.HILLEGAL/HARMFUL activities research (e.g., development of technology intended to be used in an illegal/harmful context or to breach security systems, searching the internet for information on highly sensitive topics such as child and extreme pornography, terrorism, use of the DARK WEB, research harmful to national security).	☐ Yes	☒ No
16. XPERMISSION is required to access premises or research participants.	☐ Yes	☒ No
17. XPERSONAL DATA PROCESSING (Any activity with data that can directly or indirectly identify a living person). For example data gathered from interviews, databases, digital devices such as mobile phones, social media or internet platforms or apps with or without individuals'/owners' knowledge or consent, and/or could lead to individuals/owners being IDENTIFIED or SPECIAL CATEGORY DATA (GDPR) or CRIMINAL OFFENCE DATA.	☐ Yes	☒ No
XPUBLIC WORKS DOCTORATES: Evidence of permission is required for use of works/artifacts (that are protected by Intellectual Property (IP) rights, e.g. copyright, design right) in a doctoral critical commentary when the IP in the work/artifact is jointly prepared/produced or is owned by another body	☐ Yes	☒ No
18.HRISK OF PHYSICAL OR PSYCHOLOGICAL HARM (e.g., TRAVEL to dangerous places in your own country or in a foreign country (see https://www.gov.uk/foreign-travel-advice), research with NGOs/humanitarian groups in conflict/dangerous zones, development of technology/agent/chemical that may be harmful to others, any other foreseeable dangerous risks).	☐ Yes	☒ No
19. XSECURITY CLEARANCE – required for research.	☐ Yes	☒ No
20. XSENSITIVE TOPICS (e.g., anything deeply personal and distressing, taboo, intrusive, stigmatising, sexual in nature, potentially dangerous, etc).	☐ Yes	☒ No

M – Minimal Risk; X – More than Minimal Risk. H – High Risk

If you have answered 'Yes' to ANY of the above questions, your application REQUIRES ethical review and approval using the MOREform **BEFORE commencing your research**. Please apply for approval using the MOREform (https://moreform.mdx.ac.uk/). Further guidance on making an application using the MOREform can be found at: www.tiny.cc/mdx-ethics.

If you have answered 'No' to ALL of the above questions, your application is Low Risk and you may NOT require ethical review and approval using the MOREform before commencing your research. Your research supervisor will confirm this below.

[redacted] [illegible]..... Date:10/01/2022......

To be completed by the supervisor:

Based on the details provided in the self-assesment form, I confirm that:	Insert Y or N
The study is Low Risk and *does not require* ethical review & approval using the MOREform	
The study *requires* ethical review and approval using the MOREform.	

[illegible]

[redacted].. Date: [redacted]

Reference

1. Bharatia, D., Ambawane, P. and Rane, P. (2019). Smart Electronic Stick for Visually Impaired using Android Application and Google's Cloud Vision. *2019 Global Conference for Advancement in Technology (GCAT)*. [online] Available at: https://ieeexplore.ieee.org/document/8978303 [Accessed 22 Jan. 2022].
2. Bougourzi, F., Contino, R., Distante, C. and Taleb-Ahmed, A. (2021a). CNR-IEMN: A Deep Learning Based Approach to Recognise Covid-19 from CT-Scan. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [online] Available at: https://ieeexplore.ieee.org/document/9414185 [Accessed 20 Jan. 2022].
3. Bougourzi, F., Contino, R., Distante, C. and Taleb-Ahmed, A. (2021b). CNR-IEMN: A Deep Learning Based Approach to Recognise Covid-19 from CT-Scan. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [online] Available at: https://ieeexplore.ieee.org/document/9414185 [Accessed 20 Jan. 2022].
4. Bugl, D. (2019). *Learn React Hooks: Build and refactor modern React.js applications using Hooks*. [online] *Google Books*. Packt Publishing Ltd. Available at: https://www.google.co.uk/books/edition/_/Tkm4DwAAQBAJ?hl=en&gbpv=1 [Accessed 21 Jan. 2022].
5. Darma Putra, I.K.G., Sri Asra, D.M., Dwiva Hardijaya, I.G.N., Surya Prabawa, I.G.G. and Satia Widiatmika, I.M.A. (2020). Medical vision: web and mobile medical image retrieval system based on google cloud vision. *International Journal of Electrical and Computer Engineering (IJECE)*, [online] 10(6), p.5974. Available at: https://core.ac.uk/display/333845350 [Accessed 24 Jan. 2022].
6. Fentaw, A.E. (2020). Cross platform mobile application development : a comparison study of React Native Vs Flutter. *jyx.jyu.fi*. [online] Available at: https://jyx.jyu.fi/handle/123456789/70969.
7. Hernandez, A., Ornelas, F., Hurtado, J. and Gonzalez, J. (2019). Face recognition in office environments with Google AIY Vision Kit. *2019 IEEE International Conference on Applied Science and Advanced Technology (iCASAT)*. [online] Available at: https://ieeexplore.ieee.org/document/9069514 [Accessed 20 Jan. 2022].
8. Hosseini, H., Xiao, B. and Poovendran, R. (2017). Google's Cloud Vision API is Not Robust to Noise. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. [online] Available at: https://ieeexplore.ieee.org/abstract/document/8260620 [Accessed 20 Jan. 2022].

9. Huang, T. (1996). *Computer Vision: Evolution and Promise*. [online] Available at: https://cds.cern.ch/record/400313/files/p21.pdf [Accessed 20 Jan. 2022].
10. Kudiabor, D.T. (2020). *State management with React-Redux*. [online] www.theseus.fi. Available at: https://www.theseus.fi/handle/10024/355184 [Accessed 21 Jan. 2022].
11. Markowitz, D. (2020). *I Built an AI Stylist Inspired by Social Media*. [online] Daleonai.com. Available at: https://daleonai.com/social-media-fashion-ai [Accessed 21 Jan. 2022].
12. Punia, R., Kumar, L., Mujahid, Mohd. and Rohilla, R. (2020). Computer Vision and Radiology for COVID-19 Detection. *2020 International Conference for Emerging Technology (INCET)*. [online] Available at: https://ieeexplore.ieee.org/abstract/document/9154088 [Accessed 20 Jan. 2022].
13. Rifiana Arief, A. Mutiara, Kusuma, T.M. and Hustinawaty (2018). *Automated Extraction of Large Scale Scanned Document Images using Google Vision OCR in Apache Hadoop Environment*. [online] undefined. Available at: https://www.semanticscholar.org/paper/Automated-Extraction-of-Large-Scale-Scanned-Images-Arief-Mutiara/4a82aa4b5b062f9e86bbc34262548d32d36323ef [Accessed 21 Jan. 2022].
14. Sai Aishwarya Edupuganti, Vijaya Durga Koganti, Cheekati Sri Lakshmi, Ravuri Naveen Kumar and Paruchuri, R. (2021). *Text and Speech Recognition for Visually Impaired People using Google Vision*. [online] undefined. Available at: https://www.semanticscholar.org/paper/Text-and-Speech-Recognition-for-Visually-Impaired-Edupuganti-Koganti/f6c835bd6543c98f8dfcf2e49fe44be42c4ab0c8 [Accessed 21 Jan. 2022].
15. Shalva Thakurdesai, Vira, S., Gouri Kanitkar and Save, J.K. (2021). *Smart Gallery using Google Vision*. [online] undefined. Available at: https://www.semanticscholar.org/paper/Smart-Gallery-using-Google-Vision-Thakurdesai-Vira/8dda9f6353b37755957459cbdf853b70db92ef15 [Accessed 21 Jan. 2022].
16. Shapovalov, V.B., Shapovalov, Y.B., Bilyk, Z.I., Megalinska, A.P. and Muzyka, I.O. (2019). The Google Lens analyzing quality: an analysis of the possibility to use in the educational process. *ds.knu.edu.ua*. [online] Available at: http://ds.knu.edu.ua/jspui/handle/123456789/2625.
17. Shinde, A., Tungar, M., Khairnar, P. and Gunjkar, J. (2017). Energy Efficient Business Card Recognition and Translation over Cloud Computing using Google Vision. *GRD Journal for Engineering*, [online] 2. Available at: https://grdjournals.com/uploads/article/GRDJE/V02/I04/0104/GRDJEV02I040104.pdf.
18. Sonka, M., Hlavac, V. and Boyle, R. (1993). *Image Processing, Analysis and Machine Vision*. [online] Boston, MA: Springer US. Available at: https://link.springer.com/book/10.1007/978-1-4899-3216-7 [Accessed 20 Jan. 2022].

19. Sugadev, M., Yogesh, Sanghamreddy, P.K. and Samineni, S.K. (2019). Rough Terrain Autonomous Vehicle Control Using Google Cloud Vision API. *2019 2nd International Conference on Power and Embedded Drive Control (ICPEDC)*. [online] Available at: https://ieeexplore.ieee.org/document/9036621 [Accessed 20 Jan. 2022].
20. Vaithiyanathan, D. and Muniraj, M. (2019). Cloud based Text extraction using Google Cloud Vison for Visually Impaired applications. *2019 11th International Conference on Advanced Computing (ICoAC)*. [online] Available at: https://ieeexplore.ieee.org/document/9087314 [Accessed 20 Jan. 2022].
21. Wäldchen, J. and Mäder, P. (2017). Plant Species Identification Using Computer Vision Techniques: A Systematic Literature Review. *Archives of Computational Methods in Engineering*, [online] 25(2), pp.507–543. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6003396/ [Accessed 20 Jan. 2022].