National College of Ireland

BSHCSD4

Software Development

Academic Year 2023 deferred from 2022

Micheal Butler

x18111106

x18111106@student.ncirl.ie

TechHelper

Technical Report

Contents

Executive Summary

The TechHelper app uses image-recognition software such as Teachable Machine and TensorFlow Lite to identify and classify everyday household technology items for users that are unfamiliar with them. The user can then add these items into a shopping list and search for nearby technology stores using the GoogleMaps API integrated in the app.

This application is directly tailored towards those who are not familiar with any household technology items such as the different types of USBs or an ethernet cable. As the app is for these types of people, ease-of-use is paramount as it is unlikely they will be able to navigate a complicated app if they cannot recognise common tech items.

As the industry is constantly evolving and new household technology is being introduced it is important that the items that the app can recognise can also be updated. This can easily be done by updating the TensorFlowLite model.

In summary, TechHelper uses innovative technology that can aid those in need, such as older people who are living alone, to replace faulty cables. As an example, the application would be a valuable investment for a grandparent living alone.

This report will cover the function, requirements, resources used and designing of said app.

1.0 Introduction

1.1. Background

The idea for this project came about from my awareness of the struggles that the older generations face when dealing with the ever evolving state of household and everyday technology. As I assist my grandfather with his everyday technology needs, I am aware that it can lead to unneeded stressful situations. This application seeks to help the user get the information they need as easily as possible to remedy this.

1.2. Aims

This project seeks to create a easy-to-use application that is accessible to any user that will allow them to recognise common electronic accessories via image classification, then save this classification to a list and show the user nearby stores that may have the item in stock

1.3. Technology

In this project I will be using numerous different technologies to achieve the stated project goals, it will be coded in Java.

- TensorFlowLite: TensorFlowLite is a machine learning library that I will be using to take the image provided and match them to the images to be matched against in the model.tflite file
- Teachable Machine: Teachable Machine is a free web-based tool that allows the user to create machine learning models to hold the images that will be matched against to classify a taken image.
- Google Maps API: I used the Google Maps API to allow the user to find technology related stores in their area.

- File Streaming: I used File Streaming to create a file for the list that the user saves so that the list persists over application end.
- Espresso: Used for testing the UI

1.4. Structure

This report will be broken up into:

- System - To document the requirements for the application
- Implementation - To detail the application code
- GUI - To detail the how the application looks
- Testing - To show the testing done on the application
- Evaluation
- Conclusions - Final thoughts on the app and how it can be improved

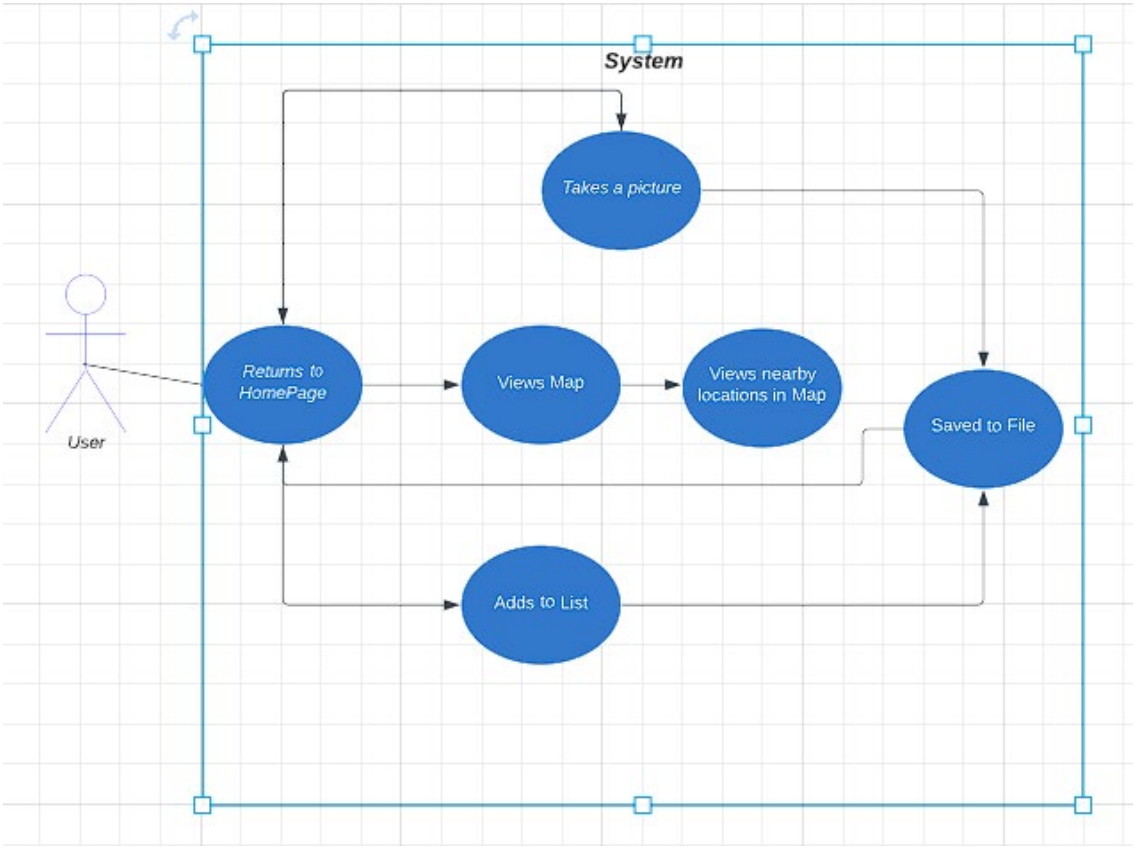
2.0 System

2.1. Requirements

2.1.1. Functional Requirements

- The app allows users to take an image and classify it and save it
- The app allows users save the data into file or delete it
- The app allows users to search for nearby shops based on their location

2.1.1.1. Use Case Diagram



2.1.1.2. Requirement 1: User classifies image

2.1.1.3. Description & Priority

The user enters the application and takes an image to be recognised and saved to file.
High Priority

2.1.1.4. Use Case

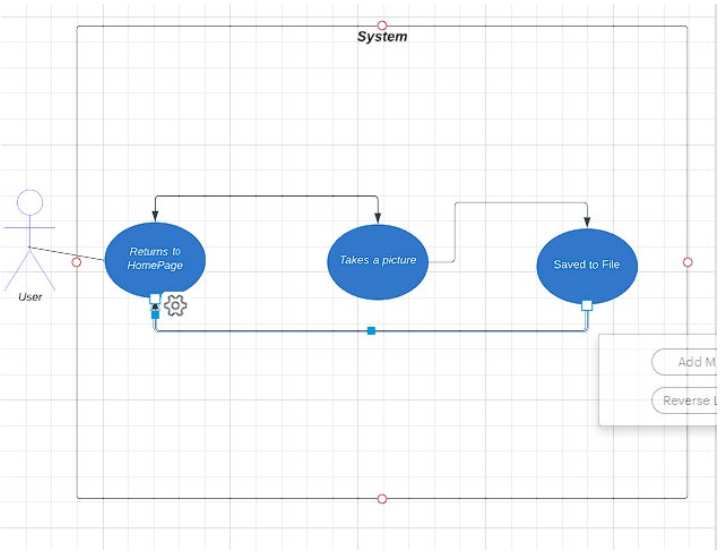
Scope

The scope of this use case is to define the process the user takes to get the image recognised

Description

This use case describes the path the user takes to fulfill this requirement

Use Case Diagram



Flow Description

Precondition

The user has entered the app and has entered the CameraActivity

Activation

This use case starts when an <User> enters the activity

Main flow

1. The system identifies the user has entered CameraActivity
2. The <Actor> clicks “Take a Picture”
3. The system opens the camera to take a picture
4. The <Actor> takes a picture and accepts it
5. The system displays the process results
6. The user clicks “Save to List”(see a1)
7. The system

Alternate flow

A1 : User Does Not Save Classification

1. The system displays the result
2. The <Actor> leaves the app

Exceptional flow

E1 : The User exits

3. The system displays the result
4. The <Actor> presses back
5. The system returns to the main menu

Termination

The system presents the the homepage again

Post condition

The system goes into a wait state

2.1.1.5. Requirement 2: User saves/deletes Data from list into file

2.1.1.6. Description & Priority

The user adds or removes information to the list and the data is saved to file on close.
High Priority

2.1.1.7. Use Case

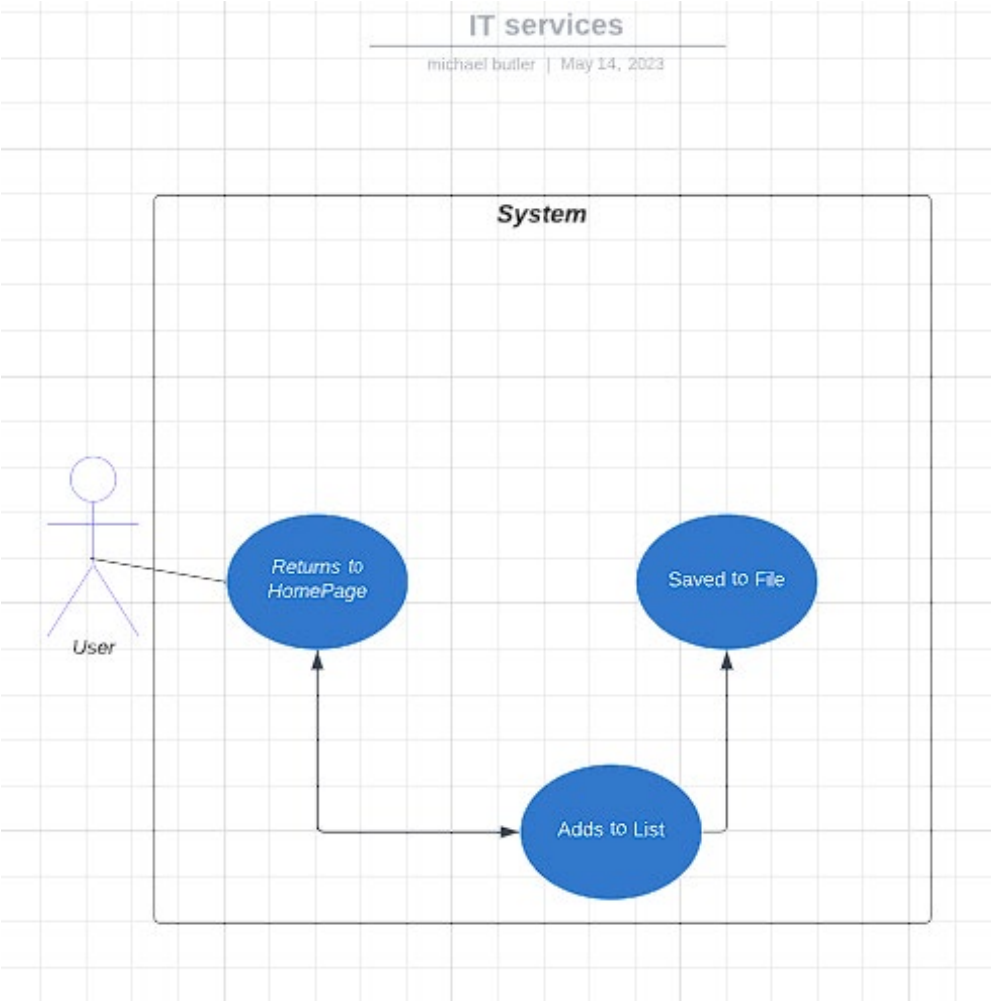
Scope

The scope of this use case is to define the process the user takes to effect the data in the file

Description

This use case describes the path the user takes to fulfill this requirement

Use Case Diagram



Flow

Precondition

The user has entered the app and has entered the ListActivity

Activation

This use case starts when an <User> enters the activity

Main flow

1. The system identifies the user has entered ListActivity
2. The <Actor> enters information into the textbox and clicks add(see A1)
3. The system adds item to the list
4. The <Actor> leaves the list
5. The system saves list to file

Alternate flow

A1 : User deletes a list entry

1. The system removes item from list
2. Rejoin main flow at 4

Exceptional flow

E1 : The User exits

1. The user exits with list unchages
2. The system saves unchanged list
3. The system returns to the main menu

Termination

The system presents the the homepage again

Post condition

The system goes into a wait state

2.1.1.8. Requirement 3: User searches for nearby stores

2.1.1.9. Description & Priority

The user uses Google Maps to search for nearby tech related stores. Medium Priority

2.1.1.10. Use Case

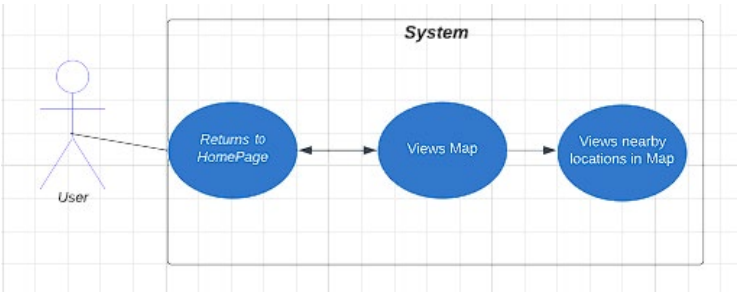
Scope

The scope of this use case is to define the process the user takes to find nearby tech related stores

Description

This use case describes the path the user takes to fulfill this requirement

Use Case Diagram



Flow Description

Precondition

The user has entered the app and has entered the MapsActivity

Activation

This use case starts when an <User> enters the activity

Main flow

1. The system identifies the user has entered MapsActivity
2. The <Actor> presses the search button(see A1)
3. The system displays all nearby tech related stores
4. The <Actor> leaves maps
5. The system returns to the main menu

Alternate flow

A1 : User does not press the search button

1. The system displays users current location
2. Rejoin main flow at 4

Termination

The system presents the the homepage again

Post condition

The system goes into a wait state

2.1.2. Data Requirements

The project requires outside data sources such as TeachableMachine Model and Google Maps API.

2.1.3. User Requirements

The user is required to own a phone, download the app and grant permissions for the camera and map functions.

2.1.4. Usability Requirements

Error Tolerance, Accessibility and User-Friendliness are the 3 main Usability requirements for this project, as it is designed for use by people generally unfamiliar with technology

2.2. Design & Architecture

Design and System Architecture for image recognition:

- Entry: The user first needs to enter this function from the MainMenu
- User Interface: For interaction with the service
- Input: The user takes a picture of the item with their camera that they would like classified
- Image Classification: Once the image has been taken image classification takes place using machine learning algorithms
- Output: The classified image is presented to the user via GUI with results
- Save result: The user can save the result of the classification to a file
- Data Access: FileStream to create and store data
- End: After saving the user is sent back to the MainMenu

Design and System Architecture for Maps

- Entry: User enters from main menu
- Map: Google Maps is called and shows location
- GoogleMapsAPI: Called to give programming access to the API
- Location Services: The app accesses the users location to supply the user with accurate information

- Security: API key is stored in the local.properties folder

System Architecture for List

- User Interface: Interacts with user and displays list
- App logic: processes data, creates arraylist
- Data Access: FileStream to create and store data

2.3. Implementation

Home Page/MainActivity

MainActivity contains functionality for navigating the app and is split up into various methods:

onCreate and onRestart: both of these methods call upon the functionActivity(). I have done this so that the listview would update if the user added new entries in the ListActivity and returned to the home page.

functionActivity: the functionActivity initializes all variables, sets up the listview and calls upon the loadContent() method.

loadContent: loadContent gets the application context and and finds the list.txt file in the app directory.

```
File path = getApplicationContext().getFilesDir();
File readFrom = new File(path,  child: "list.txt");
byte[] content = new byte[(int) readFrom.length()];
```

From there it converts the bytes content to a string, parses them correctly, adds the item to the arraylist and then sets the arraylist into the listview

```
stream = new FileInputStream(readFrom);
stream.read(content);

String s = new String(content);
s = s.substring(1, s.length() - 1);
String split[] = s.split( regex: ", ");
items = new ArrayList<>(Arrays.asList(split));
adapter = new PreviewViewAdapter( context: this, items);
listView.setAdapter(adapter);
```

PreviewViewAdapter

The preview adapter takes information and context from the MainActivity to create the layout of the entries to the listView.

ListActivity

ListActivity contains the same load function as well as the ability to add to the arraylist that can be then saved to the file in the onPause method. It uses adapters to populate the list

onCreate: the onCreate method sets up, initializes the variables and includes functionality to take the user text and add it to the listView. loadContent is then called to take information from the saved file and set it into the listView.

loadContent: loadContent functions the same as in the MainActivity.

onPause: the onPause method creates a “list.txt” file if it is not already created, it then prints the arraylist to this file to save the content on app close. I used onPause here so that if the user ever left the page its was updated, rather than have a save button

```
File path = getApplicationContext().getFilesDir();
try {
    FileOutputStream writer = new FileOutputStream(new File(path, child: "list.txt"));
    writer.write(items.toString().getBytes());
    writer.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

There are also add and remove methods to update the arraylist.

ListViewAdapter

The listViewAdapter is very similar to the previewViewAdapter, but it also contains a button to call on the remove item method in ListActivity.

Camera Activity

onCreate: initializes, on button press checks if user has permission to use camera else request.

```
picture.setOnClickListener(new View.OnClickListener() {
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    public void onClick(View view) {
        // Launch camera if we have permission
        if (checkSelfPermission(Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED) {
            Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            startActivityForResult(cameraIntent, requestCode: 1);
        } else {
            //Request camera permission if we don't have it.
            requestPermissions(new String[]{Manifest.permission.CAMERA}, requestCode: 100);
        }
    }
});
```

It also has a button to call the saveContent

loadContent: Same as load content in MainActivity and ListActivity

SaveContent: Same as in ListActivity, writes to file.

classifiyImage: The classifyImage creates a model and inputs for the function to reference itself against

```
Model model = Model.newInstance(getApplicationContext());

// Creates inputs for reference.
TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 224, 224, 3}, DataType.FLOAT32)
ByteBuffer byteBuffer = ByteBuffer.allocateDirect(4 * imageSize * imageSize * 3);
byteBuffer.order(ByteOrder.nativeOrder());
```

The model then processes that input and assigns it a confidence value.

```
// Runs model inference and gets result.
Model.Outputs outputs = model.process(inputFeature0);
TensorBuffer outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();

float[] confidences = outputFeature0.getFloatArray();
int maxPos = 0;
float maxConfidence = 0;
for (int i = 0; i < confidences.length; i++){
    if (confidences[i] > maxConfidence) {
        maxConfidence = confidences[i];
        maxPos = i;
    }
}
```

The method then gets the item assigned with the highest confidence and prints it to screen, displaying all of the other confidences below it.

onActivityResult: gets the image classified off camera and puts it into a thumbnail to display onto the screen

```
if (requestCode == 1 && resultCode == RESULT_OK) {

    Bitmap image = (Bitmap) data.getExtras().get("data");
    int dimension = Math.min(image.getWidth(), image.getHeight());
    image = ThumbnailUtils.extractThumbnail(image, dimension, dimension);
    imageView.setImageBitmap(image);

    image = Bitmap.createScaledBitmap(image, imageSize, imageSize,  filter: false);
    classifyImage(image);

}
```

MapsActivity:

onCreate: initializes and obtains map fragment.

A button to create a search parameter to send to FetchData to find nearby electronics_stores. GoogleMapsAPI is securely stored in local.properties so that it cannot be viewed.

```
//builds String to search on map
StringBuilder stringBuilder = new StringBuilder("https://maps.googleapis.com/maps/api/place/nearbysearch/json?");
stringBuilder.append("location="+lat+","+lng);
stringBuilder.append("&radius=5000");
stringBuilder.append("&type=electronics_store");
stringBuilder.append("&sensor=true");
stringBuilder.append("&key="+BuildConfig.GMP_KEY);

//send to fetchData, fetches data for nearby electronic store areas
String url = stringBuilder.toString();
Object dataFetch[] = new Object[2];
dataFetch[0] = mMap;
dataFetch[1] = url;

FetchData fetchData = new FetchData();
fetchData.execute(dataFetch);
```

onMapReady: Is called when the map is up and running and then calls to the getCurrentLocation method.

getCurrentLocation: this method first gets permission from the user to access their current location.

```
//premissions to find location
if(ActivityCompat.checkSelfPermission(
        context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
       && ActivityCompat.checkSelfPermission
       ( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED){

    ActivityCompat.requestPermissions
            ( activity: this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, Request_code);
    return;

}
```

The method then finds your current location and rechecks at intervals to ensure the map is accurate. It then places the user location into the map

```
//updates location
if(location != null){

    lat = location.getLatitude();
    lng = location.getLongitude();

    LatLng latLng = new LatLng(lat, lng);
    mMap.addMarker(new MarkerOptions().position(latLng).title("Current Location"));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng,  zoom: 15));

}
```

FetchData

FetchData is called from the button press in MapsActivity. It takes the String/Object that was built there and sends it to the DownloadUrl class to find the urlData.

```
try {
    googleMap = (GoogleMap) objects[0];
    url = (String) objects[1];
    DownloadUrl downloadUrl = new DownloadUrl();
    googleNearbyPlacesData = downloadUrl.retrieveUrl(url);
}
catch (IOException e){
    e.printStackTrace();
}
return googleNearbyPlacesData;
```

onPostExecute: takes the urlData from DownloadUrl and creates a JSONObject of it. It then loops through all the nearby locations and adds markers on those that are relevant. For this search, places with the `electronics_store` keyword are relevant as that is what is stated in the MapsActivity onClick method.

```
try {
    JSONObject jsonObject = new JSONObject(s);
    JSONArray jsonArray = jsonObject.getJSONArray( name: "results");

    //runs through nearby locations
    for (int i = 0; i < jsonArray.length(); i++){
        JSONObject jsonObject1 = jsonArray.getJSONObject(i);
        JSONObject getLocation = jsonObject1.getJSONObject("geometry").getJSONObject("location");

        String lat = getLocation.getString( name: "lat");
        String lng = getLocation.getString( name: "lng");

        JSONObject getName = jsonArray.getJSONObject(i);
        String name = getName.getString( name: "name");

        LatLng latLng = new LatLng(Double.parseDouble(lat), Double.parseDouble(lng));
        MarkerOptions markerOptions = new MarkerOptions();
        markerOptions.title(name);
        markerOptions.position(latLng);
        googleMap.addMarker(markerOptions);
        googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng,  zoom: 15));

    }
}
```

DownloadUrl

DownloadUrl takes the Object from FetchData that was created in onClick in MapsActivities and creates a http connection to search the Google Maps API for an output based on the parameters created. It then returns that data to FetchData to place markers on the search output locations.

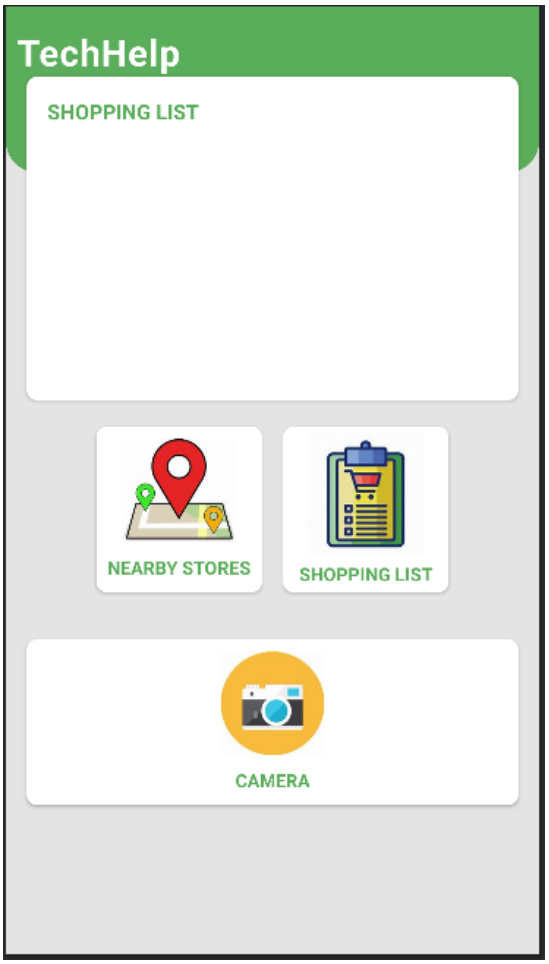
```
try {
    //creates http connection to connect to url
    URL getUrl = new URL(url);
    httpURLConnection = (HttpURLConnection) getUrl.openConnection();
    httpURLConnection.connect();
    inputStream = httpURLConnection.getInputStream();
    BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
    StringBuffer sb = new StringBuffer();
    String line = "";

    while ((line = bufferedReader.readLine())!=null){
        sb.append(line);
    }
    urlData = sb.toString();
    bufferedReader.close();
}
catch (Exception e){
    Log.d( tag: "Exception", e.toString());
}
finally {
    inputStream.close();
    httpURLConnection.disconnect();
}
return urlData;
```

2.4. Graphical User Interface (GUI)

The main goal for my GUI was for it to be easily recognisable and easy to use. Towards this goal I ensured that all actionable items such as buttons are large, with an image and clearly seen text.

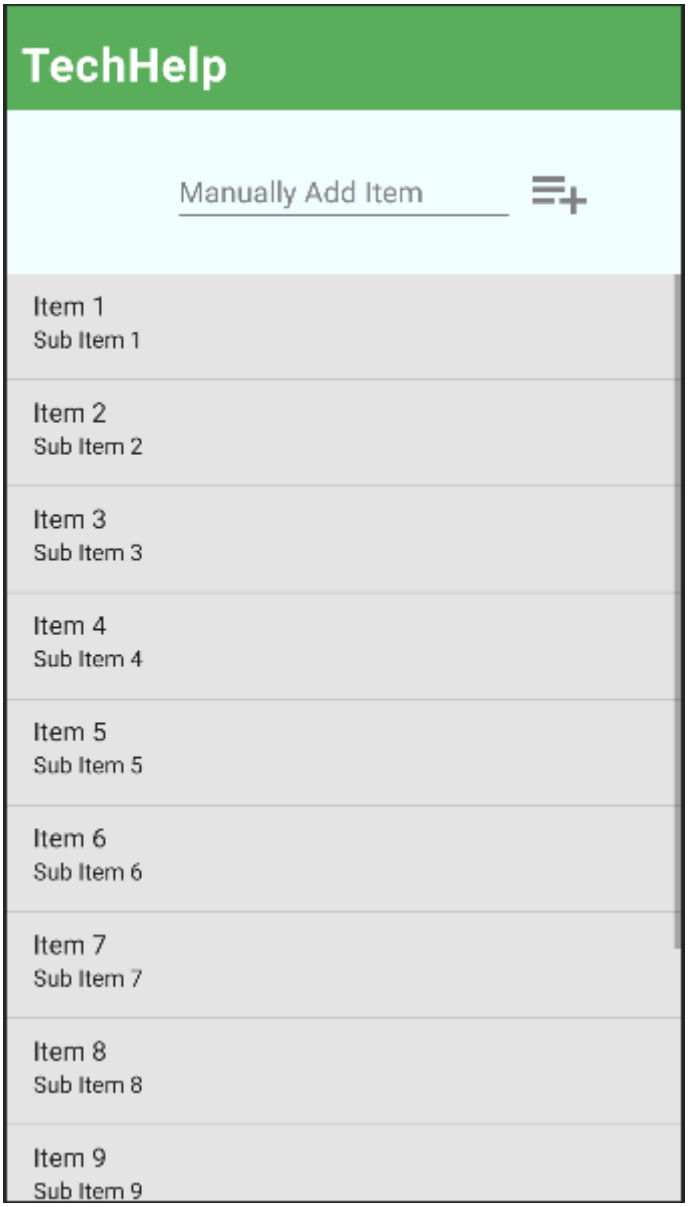
Home Screen



The Home Screen of the application as can be seen above includes a list view, and 3 clickable card views for navigation.

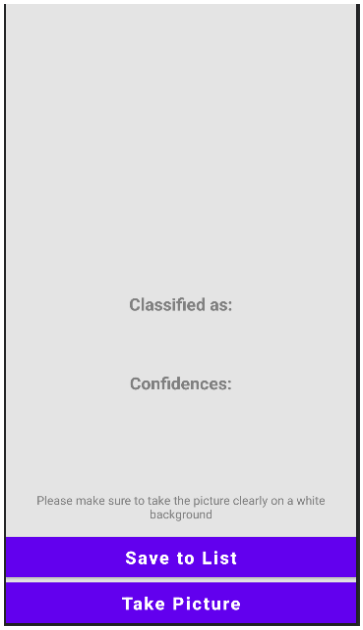
The listview shown in the home page is a preview of the shopping list “ListActivity”. The entries for the list are formatted by a preview adapter. The 3 clickable cardviews navigate to their stated activity as can be seen above.

Shopping List

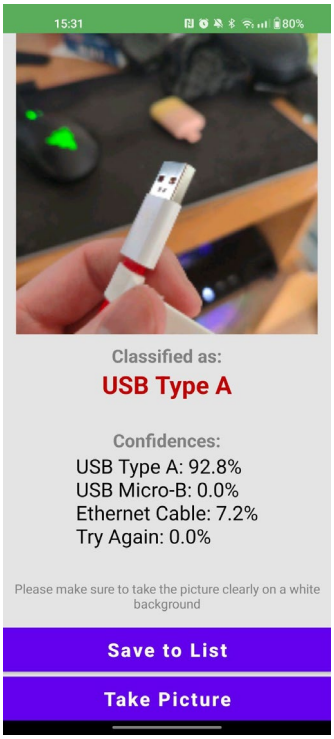


The Shopping List activity is used for the user to store items that they wish to save into a list. This can either be done by manually adding an item using the entry text field and the add button or by directly adding to it via saving the classification of the image taken from the camera. The list also uses an adapter to format the entries with a remove button for any item they no longer want on their list.

Camera Activity

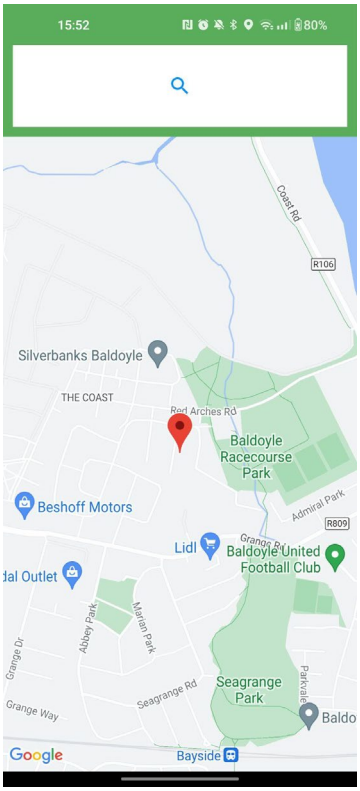


This is what the user will see when they click into the camera function. clicking onto the “Take Picture” button will open the user camera and once they take the picture the GUI will be populated as below.

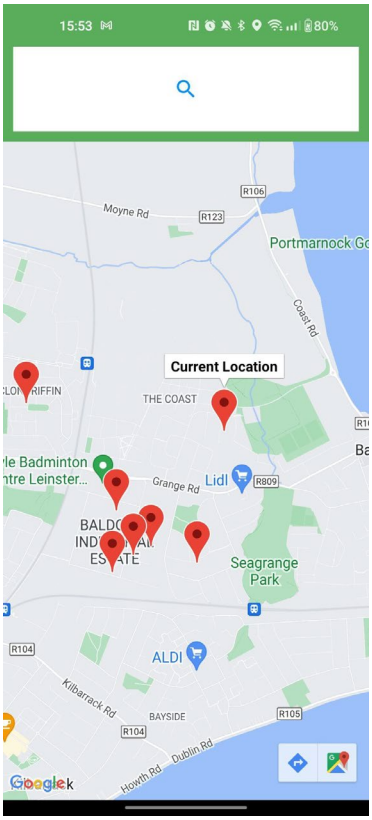


Once the image is taken the GUI will display the taken image, what the image is classified as and the different confidences that the function has in each of the results. The user can then press the “Save to List” button to save the classified item into the list and preview list.

Nearby Stores



Upon entering the Maps activity a map showing the users location will show itself. After clicking the search button at the top of the screen markers will be placed onto the map at the locations of nearby stores related to electronics as can be seen below.



2.5. Testing

For this application I created numerous tests to ensure the application runs smoothly. For my Unit and Instrumented tests I used JUNIT4 and for my UI Test I used espresso to test functionality.

Unit tests are for testing small snippets of functionality that do not require any outside resources to test. Instrumented testing requires a device or emulator to function as the tests require device resources. UI Tests can be run on devices but it is not necessary.

Unit Testing:

For my Unit Testing I created 3 tests related to the adding and saving of the List to file.

Test 1: Adding item to the arraylist.

```
@Test
public void addItem() {
    //tests add to arraylist true
    String s = "Test";
    items = new ArrayList<>();
    assert(items.add(s));
}
```

This test returns True as it is testing if you can add String “Test” to the ArrayList items.

Test 2: Removing item from the arraylist.

```
@Test
public void removeItem() {
    //test fails because test index is out of bounds, expected
    int pos = 0;
    items = new ArrayList<>();
    items.remove(pos);
    assertFalse(items.remove( o: true));
}
```

This test fails as there is nothing in the arraylist for the test to remove leading to an index out of bounds error.

Test 3: Testing splitting a String into an arraylist. This test helps in ensuring the output of loading the file holding the data successfully was entered into the arraylist

```
@Test
public void stringSplit() {
    //tested for multiple methods
    //tests to convert to string when reading from file
    //test failed successfully???? assert returns fail but Arraylists are the same output
    items = new ArrayList<>();
    items.add("reg");
    items.add("mick");
    items.add("john");
    test = new ArrayList<>();

    String raw = " reg, mick, john ";
    String s = new String(raw);

    s = s.substring(1, s.length() - 1);
    String split[] = s.split( regex: ", ");
    test = new ArrayList<>(Arrays.asList(split));

    assertSame(test, items);

}
```

Instrumented Test

Instrumented tests are tests that need to be run on a device or emulator to be tested. I created one instrumented test for this project that tests if the file holding data exists on the user's device.

```
@Test
public void existContent() {
    //test for file existing
    //test shows it exists
    @SuppressWarnings("deprecation") File isReal = new File(InstrumentationRegistry
            .getTargetContext().getFilesDir(), child: "list.txt");
    assert(isReal.exists());
}
```

Test returns true. This test is used alongside the unit tests to ensure that the user can store the arraylist to a file on the device so that data persists on app close.

UI Test

For my UI Test I used Espresso. UI Tests on android, test so that the user doesn't have any unexpected results when navigating through your application. I enacted 3 tests on my List Activity GUI.

Test 1/2: Tests the existence and location of the editTextBox in the List activity:

```
@Test
public void editBoxBesideImage(){
    //test to see if enter textfield is beside the add button
    onView(withHint( hintText: "Manually Add Item")).check(matches(isDisplayed()));
    onView(withHint( hintText: "Manually Add Item")).check(isCompletelyLeftOf(withId(R.id.add)));
}
```

This test succeeds as the test finds that the textbox exists and is completely to the left of the add button

Test 3: Tests that there is something being displayed inside of the listView

```
@Test
public void populatesList(){
    //checks if anything displayed in the listview
    onData(anything()).inAdapterView(withId(R.id.listView)).atPosition(0).check(matches(isDisplayed()));
}
```

This test succeeds as there is something in the file that is then displayed onto the listView, if the file was empty then there would be nothing for the listView to show and the test would fail.

2.6. Evaluation

This project can be evaluated by the accuracy at which it classifies images and the accuracy at which the map functions. The map functions have been found to be correct at all points tested.

After testing, the image classification for the Micro_USB has been shown to be untrustworthy, with it only recognizing the from a diagonal angle from above. This can be remedied by adding additional images that are taken from all angles to the tflite model.

3.0 Conclusions

After analysing the various features of the completed application it is clear that the app provides a valuable tool for its users. With features such as image-recognition, a list to add

their items and an area of interest map this app offers its user an easy-to-use platform for finding out what they need.

One of the key features of the app is its user-friendly GUI, which is designed specifically so that there is as little room for error as possible. This is especially important as the app is targeted towards those who are unfamiliar with technology and if the application was hard to use it would be a hard barrier to entry. Another notable feature of the app is the ability to directly save the recognised image classification to the items list rather than having to do it manually saving the user time and effort.

In terms of effectiveness the apps ability to correctly classify an item, allow users to see the confidences of each item to make an educated guess if the item is unclear, save this item to a list and show the user nearby technology stores on google maps ensures the user can get the items they need as easily as possible.

There are some limitations on the application such as the TensorFlowLite Model that holds the items that are recognised needing to be updated to include any new items that need to be recognised.

In conclusion, TechHelper is a valuable tool for users trying to find a replacement for that broken USB that there are “too many types of”. While there is room for improvement in terms of updating the application for new items, the app will be highly useful to those in its target demographic.

4.0 Further Development or Research

Further development of the app could include having the application automatically update the image classification model to stop the need to continuously update for new technology. This would be done by machine learning technology, using algorithms to find new common household tech items online.

Another way this could be done is to allow an admin user to classify images themselves and make it available to the main user. This could be an older person's primary care-giver to help them remember items around the house.

The app could also be expanded to allow the user to search the stock of nearby stores to see where the item classified is available.

At this stage the application could be monetised on a subscription or once off payment to a user admin. Alternatively ad-based monetisation could be implemented but I believe this would make using the application needlessly too complicated and be contrary to the intent of the service provided.

As the application would continue to grow it would be important to keep the users needs in mind

5.0 Appendices

5.1. Project Proposal

National College of Ireland

Project Proposal

HardwareHelp

08/11/2020

BHSC4

Software Development

<2020/2021>

Michael Butler

X18111106

X18111106@student.ncirl.ie

Contents

1.0 Objectives

The objective of this project is to create an Android application that will use the camera of the users phone to recognise the external ports of a device or a cable and inform the user of its name and details. Another feature that will take place after this step, if the user so wishes, is the app then linking the user to nearby stores that would supply said piece of hardware.

The app can be used to by anybody unfamiliar with IT to replace any parts they are unfamiliar and will be developed to be as easy to use as possible with as little clutter as possible in the apps UI.

The application will be developed in 3 steps:

Step 1: being to research the different types of ML Kits that are on the market ,such as FireBase, and decide which will be the best to use.

Step 2: will be to research different app UI styles and then to create my own that I feel matches my app the best.

Step 3: will be to implement the ML into the app and begin testing.

Once the core application has been completed, I can then proceed to add additional features. The additional features that can be implemented include:

- Internal PC part recognition, this can be used by people who are looking to build their own PC. After inputting the PC parts, it will recommend the user parts that function well with it. E.g. if the power supply of your is too low it will recommend an upgrade. This can be done either by taking a picture of a part or by inputting it manually.
- Linking the consumer to online stores such as Amazon for their product

The finished product of the application will be available on the Android Store.

2.0 Background

Managing hardware can be overwhelming for people who are not very IT inclined. This application is focused on reducing any confusion that people may have and working to get any issues resolved as fast as possible.

My decision for developing this application is made both around personal anecdotal experience and working on an IT Helpdesk.

As the resident IT student in my family, I am called upon for a number of different IT issues. The last issue that I helped with was replacing cabling for my grandfathers Wi-Fi. All went well but the main issue that he came across was that he purchased the incorrect wires for the devices he was using. This issue could have been avoided if he had some way to label each of these wires. I believe that with the use of an application such as the one I am developing this issue would have been avoided.

In my internship over 2020 I worked on an IT Helpdesk in Irish Life. During my time there I noticed there was a noticeable hit to the time/efficiency it took for an issue that was related to hardware, either the addition or replacement of new hardware such as an additional monitors or troubleshooting faulty equipment, to be resolved. The reason for this issue was that alongside the hardware that was being requested the client also had to request wires and connectors for these devices, there were a number of cases where the client already had these wires, thinking they were the wrong ones, without realising it or they didn't request the right ones. This either led to wasted resources, if they already had them, or wasted time, if they did not and the company would have to send more.

I would recommend this application to be integrated into processes such as requesting additional hardware for companies such as Irish Life. The app will be simple, easy to use and will take up very little additional time for occurrences such as these but will help to reduce the chances of error.

There are many different types of image recognition software on the market, the best example being Google Lens. Google Lens allows the user to take a picture of a wide variety of different objects or things such as text and then link the user to different articles on Google that are related to what was taken a picture of. After some testing, Google Lens doesn't seem able to recognise the individual ports on PCs and when taking pictures of IT related items as a whole it links to web-pages that do not apply to what we are looking for. The application that I look to develop will be more specialised on IT related issues rather than the broader scope of applications such as Google Lens. Researching similar software to this, leads me to the same conclusion.

3.0 Technical Approach

At the beginning, although I am only working in a one-man team, I will research some methodologies that I can use as a framework to help me work efficiently.

Firstly, I will need to research further into needs of my consumer base and based on my findings I can tailor the primary feature of the application. Once this has been done, I can then move onto researching the viability of additional features that run parallel to its main function such as the feature for assisting in the building of a desktop.

I will then have to research the different Machine Learning Kits that are available to me. At the moment the one I am leaning more towards is FireBase as I have prior experience with it in previous projects. Once I have done this, I will need to look into developing an UI for the application that prioritises ease of use. I will research this by downloading similar applications to the one I look to develop and both critique it myself and read the user reviews.

Once I have finished my research, I will begin the development of the UI in Android studio, making it as user friendly as possible and unit testing as I go along. Once I am confident in the UI, I will begin the implementation of machine learning into the application. I will then test the robustness of the application and once it is functioning smoothly, I will begin the implementations of additional features, repeating the process.

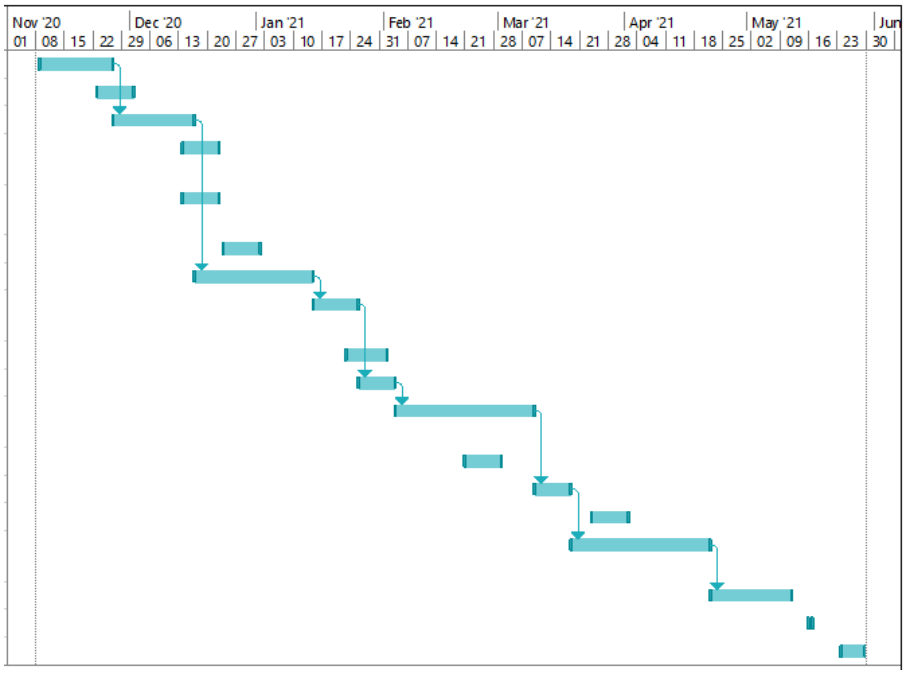
4.0 Special Resources Required

The special resources that I require for this project include:

- Android Studio
- FireBase Database and ML Kit

5.0 Project Plan

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		Research	14 days	Mon 09/11/2	Thu 26/11/2(	
2		Reflective Journal	7 days	Mon 23/11/2	Tue 01/12/2(	
3		Design UI	14 days	Fri 27/11/20	Wed 16/12/2	1
4		Documentation for Midpoint	7 days	Mon 14/12/20	Tue 22/12/20	
5		Mid Point Implementation	7 days	Mon 14/12/20	Tue 22/12/20	
6		Reflective Journal	7 days	Thu 24/12/2(	Fri 01/01/21	
7		Implement ML	21 days	Thu 17/12/2(	Thu 14/01/2	3
8		Finishing touches base project	7 days	Fri 15/01/21	Mon 25/01/21	7
9		Reflective Journal	7 days	Sat 23/01/21	Mon 01/02/2	
10		Testing	7 days	Tue 26/01/21	Wed 03/02/2	8
11		Implement Secondary features	24 days	Thu 04/02/21	Tue 09/03/21	10
12		Reflective Journal	7 days	Sun 21/02/21	Mon 01/03/2	
13		Testing	7 days	Wed 10/03/2	Thu 18/03/2	11
14		Reflective Journal	7 days	Wed 24/03/2	Thu 01/04/2	
15		Cleaning up and adding new features	24 days	Fri 19/03/21	Wed 21/04/21	13
16		Final Implementation	14 days	Thu 22/04/21	Tue 11/05/21	15
17		Video Presentation	1 day	Sun 16/05/21	Sun 16/05/21	
18		Project Showcase	6 days	Mon 24/05/2	Sat 29/05/21	



6.0 Technical Details

Implementation language and principal libraries

I will be coding using Java in Android Studios. Using FireBase ML Kit for image recognition, and Database to hold the images to match to through their API.

7.0 Evaluation

Testing

First, I will need to test each individual segment of my project before combining them using unit testing. I will likely do this by adding print markers throughout my code. eg: before the machine learning is implemented, I will need to fully test the robustness of the app itself and make sure all pages are functioning.

Once the unit testing has been complete I will move onto integration testing, where I will introduce features one by one into the application to make sure they run smoothly and once they all function correctly I will add them all in together and test the system as a whole.

Once this testing has been complete I will move onto more public acceptance testing, where I will go to my intended consumer base as well as others outside of it to get opinions on function and design critiques that can then be implemented at a later date.

Consumer Satisfaction

Once the project has been finished and released, how well it is received can be reviewed from either reviews on the app market and implementation of a additional function request system inside of the app itself, where the user can send critiques or reviews directly to the developer

1.1. Reflective Journals

Reflective Journal 1

For 01/11/2020

This month we got settled in back to college from our Summer break, there are some teething issues, but I have gotten back into the swing of things. We did our project pitch video for our final year project, where we submitted a short video detailing what we would like to do and the our supervisor would reply with what they thought of our idea and if we should change it. I accidently set my video to private when sending it in, so the review was a bit delayed although all of the other students I have asked got theirs back and I've yet to get mine. I decided to do mine on an android application that takes a picture and recognises what port or cable that it sees. This would be useful to anybody is not very IT inclined and can help them replace any faulty cables. It seems to be a unique idea and I cannot really find any other examples of it online.

I am confident that I will be able to deliver on my project idea but have yet to hear back from my supervisor about this, and my proposal is due next week.

We had another CA due for Mobile App Dev for the research and development of the frontend of our project for that class. This was a refresher for me as it has been a while since I have developed an android app. I am confidant that I will be able to deliver on my project idea.

Reflective Journal 2

Unfortunately, I have not gotten a lot done this month as there is a lot of extra work due in CAs than usual and I have to focus on them to get them in on time. Earlier on in the month I was able research further into how I wish to design the front-end of my project. I decided to use a darker colour theme to be easier on the eyes as some restaurants have lower lighting for atmosphere, I may at a later stage create a dark/bright mode switch so that the user can decide.

Reflective Journal 3

Michael Butler

X18111106@student.ncirl.ie

The month of December has been a very busy month for me, there have been a large number of assignments due for the end of the first semester. The main hand-in that I had was the mid-point submission for my Final Year Project. Due to me underestimating the size of the submission I was unable to completely finish the GUI of the prototype, but I was able to add enough functionality to properly showcase the services that the finished product will provide. While I was unable to showcase the style and design of the application, I was overall pleased with what I handed in.

Due to me not completing the GUI I was left a little bit behind in my Gantt chart plan, due to this I do need to catch up in this regard, that being said I have a number of CAs that are due at the beginning of January so I will be leaving it until after then.

I do think I will need to put more focus on this project in the future as it does give more credits than any other module.

Reflective Journal 4

X18111106 – Michael Butler

01/03/2021

For the month of February my goal was to implement the image recognition AI to the project. Unfortunately, I am a bit behind schedule and will begin this now. I plan to map type function so that the user can search for nearby stores. I plan to update the UI and style a bit as I am displeased with how they look.