National College of Ireland

Computing Final Year Project

BSCSD4

2023/2024

Taziwa Mushayabasa

20444686

X20444686

FloralScan

Technical Report

Contents

Executive Summary

This project aims to bridge the gap present between technology and the natural environment around us. This application is catered toward a younger demographic who take interest in the botanic life growing around them. The application aims to provide a user-friendly mobile platform for the identification of different plant life and exploration of various plant species, enhancing user engagement with the surroundings around them.

Alongside bringing higher awareness of the biodiverse life around the users of the application, this project also aims to educate the users about the plant life within their surroundings. To achieve this, the application will be utilizing several technologies to develop it and to allow it to function when complete. A prominent technology that will be highly relevant to the success of this application are the cloud services provided by Amazon Web Services. This report aims to clarify and dive deep into the production, technologies and use cases of this application.

1.0 Introduction

1.1. Background

I chose to undertake this project due to the growing potential found within integrating technology to assist with everyday learning in spaces that may be niche or under-explored at this moment of time.

Although technology is only ever-growing and advancing every day, there are many spaces that it can still be implemented and optimized within. For example, within the outside world for users using these technologies users prefer simplicity and accessibility especially when an application is used out of curiosity, convenience or education as seen with other learning applications such as Duolingo.

This application aims to bridge that area for botany in the current day and age. In addition to this, cloud services are also an ever-growing sector of technology that have become more prevalent as time continues. Having these can prove to be useful and overall provide an experience is simple to use toward users while upholding complex operations within an online back end.

Competitor applications such as PlantSnap and PictureThis exist with their own unique use cases.

Use Case

PlantSnap [28] is an identification application that specializes in identifying plant species. This application has the functionality to identify 90% of all known plant and tree species. PlantSnap is also a social media platform that enables individuals to share images and have discussions with other users of the application to share different findings, ask questions and build a community between each other.

PictureThis [29] is another platform with a high userbase of over 100 million, also serves as an identification application of floral life using artificial technology. This application differs from others as it also has the option to diagnose plant diseases and gives tips to users about the plant if they desire to care for one in the future. It also has the capability of different regional support via a language toggle and operating globally. The application also puts emphasis on having a user-friendly platform alongside being secure with the data used, reminding users that the pictures uploaded are processed but are never used externally

outside of this action. It also emphasizes that it must have permission to use the picture from the user before beginning the identification process.

Google Lens [30] is a platform developed by Google that provides a large selection of image recognition capabilities. These can range from buildings to plant life. Google Lens puts an emphasis on clear and focused photos in a well-lit room to improve identification results and accuracy.

Accuracy

These applications differ from one another as they contain varying levels of accuracy. PlantSnap reports an accuracy rate of 71.8% (Pankau, 2022; Schmidt et al., 2022) while PictureThis achieves an accuracy rate of 97.3%. Google Lens correctly identifies plant species at 92.6%.

These existing applications and accuracy metrics serve as both a guideline and a foundation for my application during development.

1.2. Aims

This project aims to produce a product that can provide an accessible tool that can promote both botanical awareness and education to users who desire this through a convenient mobile application. Through the creation of the application, engagement of the natural environmental around us is looked toward to increase due to the heightened awareness and curiosity brought by it.

By providing an application that provides an easy to access user interface that can effectively identify these different plants, it aims to appease to a younger audience on these topics while being accessible, convenient and a simple procedure to use.

Additionally, this application will provide a search functionality that allows users to explore the results of their search query from the applications own database. This feature allows users to have an in-app search feature that furthers the user-friendliness of the application.

These elements combined should provide an application that serves as both a tool for plant identification and a platform for users to be able to learn more about plants if they wish to.

1.3. Technology

To develop this application, several technologies were identified and set for use for effective functionality. This application will be utilising various AWS cloud services alongside the use of the Java programming language and XML. AWS provides specific services that allows the bulk of the applications back-end operations to be hosted within the cloud environment, allowing reduced load and faster run-time.

For this application AWS S3 [9] will be used. AWS S3 is a storage system that allows different objects to be inserted into there. Objects can vary from images to videos to text files. These objects are secure within the AWS S3 bucket as the S3 bucket allows heavy configuration to restrict access, keep objects within the bucket secure and to specifically control who can view the items within there. Additionally, the bucket can be made private and secure to everybody outside of the root user and can allow other users only given access depending on the configuration of the bucket. Using AWS S3, user requests via images will be stored within an S3 bucket made specifically for the application and will be the starting point within the AWS pipeline in order for the plant analysis operation to begin. AWS S3 also provided as a service that allowed training of plant images via AWS Rekognition. Using a separate S3 bucket, AWS Rekognition will grab from this storage system and use the datasets given to it in order to train it and make it efficient at analysis. AWS IAM is also a core structure of this application.

AWS IAM [11] is an AWS service that allows users to be created to be given credentials for the application to access and use the bucket. These IAM users are personally configured and are applied with the principle of least privilege to ensure that the permissions given are both secure and allow no room for any malicious actors to act. These permissions are set and cannot be changed internally, such as the administrator user on a file system, making this secure. Without the IAM user, the application would have no ways of accessing the AWS cloud services configured safely and securely, adding risks that cannot be allowed to happen. Utilising AWS IAM, the application inherently uses the IAM profile constructed for it in order for users to submit requests from within the application. Permissions are set in order for both the AWS configurations alongside user data to be kept secure in this process.

AWS Lambda [7] is a service within AWS that provides both codes to be implemented, code to invoke interactions between services in the AWS pipeline and computing power offered.
AWS Lambda will be important as it provides a trigger for the application to continue its workflow and invoke other applications. AWS SNS is similar in nature, but also invokes services alongside AWS Lambda.

AWS Rekognition [10] is a system that provides image analysis, the main premise that holds this application. Utilising AWS Custom labels from AWS Rekognition, plant images specifically will be trained onto a model in order for it to be used in the analysis process for the application.

AWS API Gateway [12] will be used as an endpoint in order to relay the information toward the application once a successful process of all the previous services within the application back-end have been completed.

Within the mobile application backend, the main languages used to produce the interface and functions in order to store, upload, and traverse the pages of the application will be made with Java and XML. Java provides effective as a language to call different functions and produce many of the different functions needed by traversing the application and invoking information to be sent toward AWS. XML is utilised in conjunction with the inherit Android Studio GUI editor to properly design, style and edit the layout of the application. These AWS services combined with production via Android Studio will ultimately achieve what I have set out to achieve with this application and its technologies. AndroidStudio [23] is the official IDE for mobile development, which I was why I opted to use it over other IDEs available for the development of this project.

1.4. Structure

This document is structured in a format that provides concise titles to identify each section by. These titles are the Introduction, the System, the Testing, the Further Development & Research and the Conclusion. If needed, refer to the table of contents on pg. 2/3 of this report.

These sections provide the projects overall development cycle alongside what the complete product has achieved through this process. Sections are divided to delve deeper into the specifics of the application and the different aspects of production, use cases, testing and reflections thus far.

These sections will also dive deep into the decision-making process for these different requirements and methodologies onboarded for this application.

2.0 System

2.1. Requirements

The requirements of this application are as follows:

- To provide users an accessible, easy to use interface.
- To allow users to securely provide their gallery images of plant images to be identified.
- To allow users to alternatively use the in-app camera function to take a picture of the plant life to be identified.
- To be able to view the plant information returned by the application after image analysis has been conducted and finished.

As this application is designed with ease of use in mind, all system functions should be able to be adapted to in a minimum amount of time as that is how the design principles of the application are. Training of use of the application should be under 15 minutes ideally. As this application is also marketed toward a younger audience it is imperative to keep these functions simple as they are effective. The average number of errors experienced by users should not exceed over three per day. Reliability, user-friendly navigation, and the reliability of the application's core functions are a must.

2.1.1. Functional Requirements

The functional requirements of this application are listed below to provide awareness to what the system must prioritize to accomplish.

AWS Functionality: The application must be able to connect seamlessly with the Amazon Web Services set up to proceed with the AWS pipeline. This includes permissions to submit images to the s3 bucket alongside access to retrieve information from the API gateway. Successful operations of this allow the application to function as expected.

Plant Detail Display: The mobile application can successfully display of plant details alongside the plant name after receiving a request from a user. These requests would be uploading from their camera or image gallery and receiving relevant plant information within the application in return,

Search Functionality: Thie mobile application can successfully display a list of plants relevant to the search query the user has entered. If the query is irrelevant or misspelled, the application will return “No search terms found”. When successful matches are found, user can be able to select these different plants to have a pop-up description of the plant name and what the plant is appear.

User Interface: The user interface should be aesthetically pleasing and easy to use. Users can easily navigate the application without issue alongside being able to identify what page they are on through the icons and titles witnessed through the navigation bar. Buttons in order to trigger different functionalities are clearly labelled and straightforward for users to be able to use. User Interface is expected to not crash or act in an unknown manner. I.e pages not loading and being force pushed to another to compensate.

2.1.1.1. Use Case Diagram

Figure 1: FloralScan Use Case Diagram

2.1.1.2. Requirement 1 User wants to upload plant image.

2.1.1.3. Description & Priority

This use case is the process that a user would proceed through when they want to use the applications “Upload” feature. In this use case, the user is uploading an image of plant life with the expectation of being returned information of the plant life they submitted from the application.

2.1.1.4. Use Case

User uploads an image.

Scope

The scope of this use case is for a user to upload an image from their gallery of a plant to be analysed and returned to them via the AWS cloud pipeline and AWS Rekognition.

Description

This use case describes and showcases the process by which a user uploads an image and invokes the AWS pipeline in order for their image to be analysed.

Use Case Diagram

The diagram highlights three actors, these being the User, the Mobile Application and AWS. This use case encompasses the actions of uploading from the application via the “Upload” button.

Flow Description

Precondition

Application is open in an idle state, waiting for user input.

Activation

This use case starts when the User uploads an image via either the “Use Camera” or “Upload file” option within the application.

Main flow

1. The system identifies the upload button selected and prompts the user for an upload.
2. The user uploads the image or takes a picture of one.
3. The system uploads the image to AWS S3 and invokes the AWS pipeline.
4. The user is supplied with a loading screen while this process is operating.

5. The system provides the information back to the application with the information.

Alternate flow

A1: Upload Reselect

1. The user decides to upload a different image or has selected the wrong one.
2. The user selects “no” at the confirmation prompt.
3. The use case continues at position 1.

Exceptional flow

E1: Upload Error

4. The system runs into an unexpected error or network interruption.
5. The user is notified of the error and is asked to try upload again.

The use case continues at position 1 of the main flow.

Termination

The system presents the page with the plant and information at completion.

Post condition

The system goes into a wait state, ready for the user to initiate action.

List further functional requirements here, using the same structure as for Requirement1.

2.1.2. Data Requirements

The data requirements of this use case is the image from the users gallery that they have chosen, as it will be granting the application to use it to be analysed.

2.1.3. User Requirements

User requirements of this use case is the user granting the application to use the image selected. Another requirement is the image being of a plant and not of another object. Additionally, for this use case users should seek to use a clear image, if possible.

2.1.4. Environmental Requirements

The environmental requirements of this use case would be for the user to be somewhere which they are connected to the internet. Additionally, the image uploaded should be one in a bright environment of a singular plant or focused upon one plant to increase the chances of the correct information being returned.

2.1.4.1. Description & Priority

User has opened the search page and is going to perform a search using the applications search functionality to learn more about a specific plant.

This is a high priority functionality as it correlates with one of the core objectives of the overall application. This being the delivery of detailed information of plant life.

2.1.4.2. Use Case

User is going to use the search functionality.

Scope

The scope of this use case is for the user to successfully use the search functionality to display the plant they have entered.

Description

This use case describes the process which a user would go through in order to query for the plant name through using the search feature. Multiple descriptions and relevant plant images will display.

Use Case Diagram

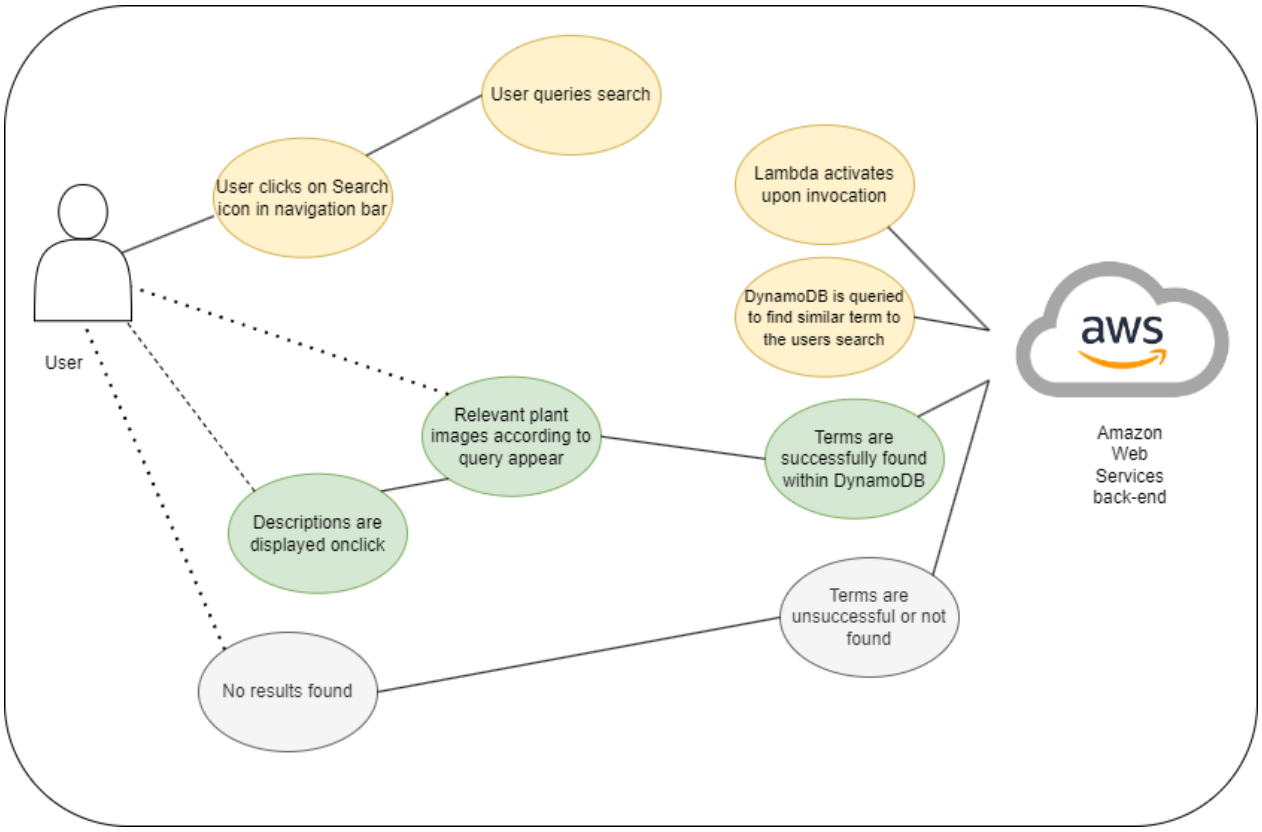


Figure 2: FloralScan Use Case Diagram

Flow Description

Precondition

The application is open, and the user is on the Search page after navigating to it using the navigation bar.

Activation

This use case begins when the user taps on the search icon on the Search page.

Main flow

6. The user queries the search functionality by typing in a plant name.
7. The user submits the plant name into the search functionality.
8. The application sends this request to the AWS back-end.
9. AWS begins using DynamoDB to search for matching plant names.
10. Relevant plant names, images and descriptions are returned.
11.

Alternate flow

A1: DynamoDB cannot successfully find any plant names that match or are similar to the users search query.

6. The user submitted an entry that cannot be found within the database.

7. The application returns no search results found.
8. No plant images, descriptions or images appear.

Exceptional flow

E1: User clicks on a search result.

9. The system has successfully loaded search results that match or are like the user's query.
10. The user selects one of these search results.
11. A pop of description alongside the plant name appears for the user to be able to read.

Termination

This use case will terminate when the user finds their desired plant, de-selects the description pop up, leaves the page or restarts the use case by searching another query in the search functionality.

Post condition

The application will remain on the search page, until the user selects another action within the application.

List further functional requirements here, using the same structure as for Requirement1.

2.1.5. Data Requirements

User query is sent as data toward the search functionality, must be compliant with this.

2.1.6. User Requirements

User is required to query the plant name with no irregular letters in order to increase chances of a successful match from within the database.

2.1.7. Environmental Requirements

User should be in range of an internet connection.

2.1.8. Usability Requirements

User should type with normal text characters in order to properly query the functionality.

2.2. Design & Architecture

This application is made to be usable to all audiences but particularly toward a younger demographic. With an accessible and simple interface, this application is made to be engaging and approachable for all users of different age demographics.

The system architecture of this application follows the use of Amazon Web Services to uphold a functioning and unique process for the applications requests. This architecture allows users to easily use the mobile application front-end without any exposure or confusion with the backend operations of the core features of the application.

Displayed below is the AWS pipeline for events that occur when users submit a request.

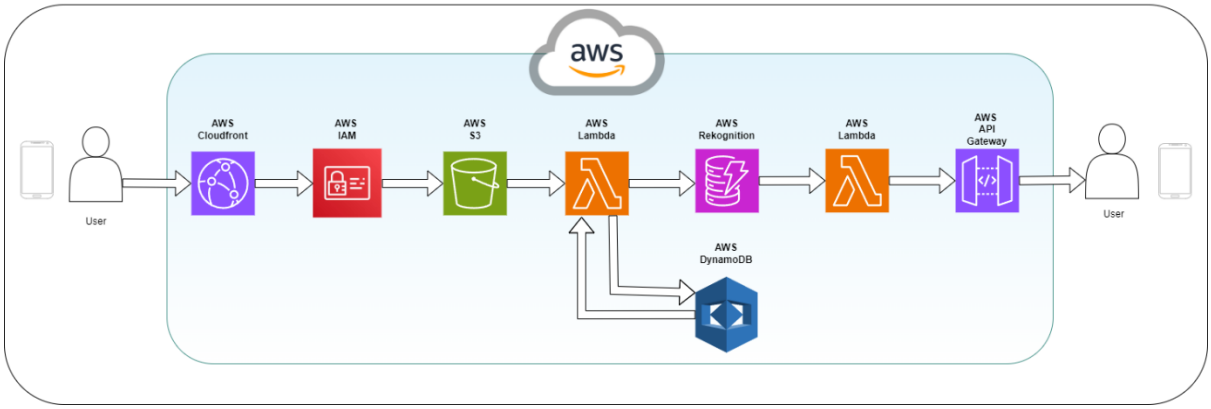


Figure 3: FloralScan AWS Cloud Back-end Pipeline

The AWS architecture overview is as follows.

AWS Cloudfront [14] is a content delivery network that allows the acceleration of data to be delivered to users of the application.

AWS IAM [11] is used to manage access control to different AWS services. By applying principle of least privilege, security is able to be tightened and ensure a secure interface of services within the back end. By creating a central IAM user with the application in mind, it assumes permissions to the resources it needs whilst having parameters instilled onto it, preventing misuse of this resources if bad actors were to access it.

AWS S3 [9] is an object storage system that can be used for a multitude functions. By integrating it with other services such as Lambda, S3 can become a storage system that allows the objects within it to be used within functions, as an output area to store objects or to act as an area for datasets to be trained for example.

AWS Lambda [7] is a serverless compute service that runs code without the need for provision or server management. This service is purely event driven, which allows it to be useful to handle code execution when users make requests. AWS Lambda has the

advantage of being able to be invocated through events of other services too. In the context of the FloralScan application, AWS Lambda can allow itself to be invoked with code execution through setup with the AWS S3 bucket used to receive user images.

AWS Rekognition [10] is an image recognition service that can analyse and detect differences within the plants from the images uploaded from users. Through training this system with plant datasets, it will be able to detect different plants from one another. When AWS Rekognition is trained, a “Model Performance” score is given to give context on how well the model can identify different plant life. This can also be an indicator of needing to re-train data with larger, more diverse datasets.

AWS DynamoDB [8] is a database system within the AWS cloud that allows the storage of the plant data through values such as their names, descriptions, and attributed image.

AWS API Gateway [12] is an AWS fully managed service that allows the use of an endpoint for the application to be left open, allowing easy retrieval of the data needed.

When used in conjunction with each other, these services provide a back end that allows the application to quickly process the requests made by users and provide them the plant details expected. A sequence diagram of this operation can be seen by Figure 4.

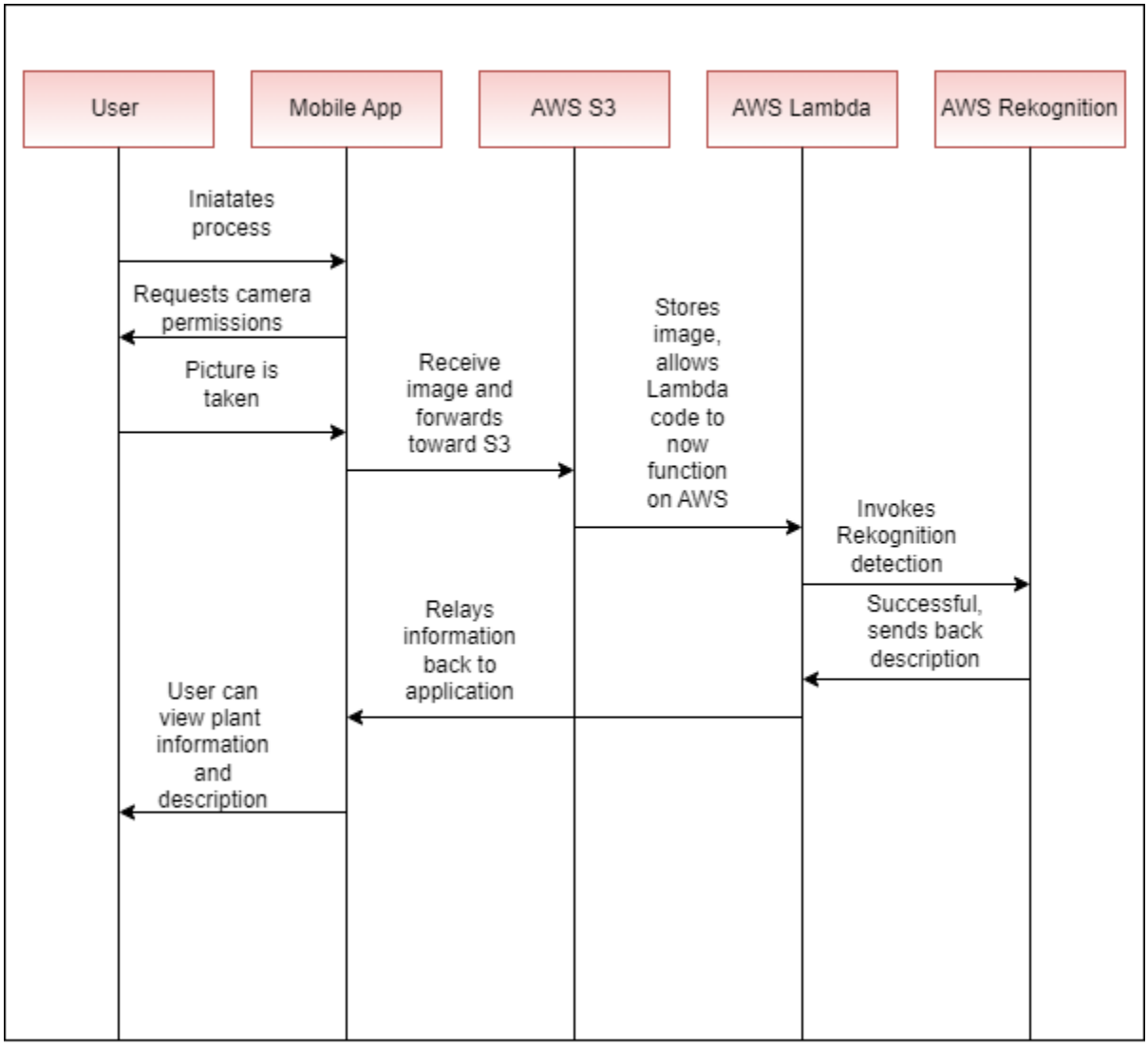


Figure 4: Sequence Diagram.

2.3. Implementation

This section will cover the different components used to build the application. The FloralScan application is built using object orientated design to provide a functional, smooth environment for the userbase to navigate through. A compilation of these components can be found through the upcoming sections, starting with the classes used for this application.

A class diagram of these classes can be seen with Figure 5.

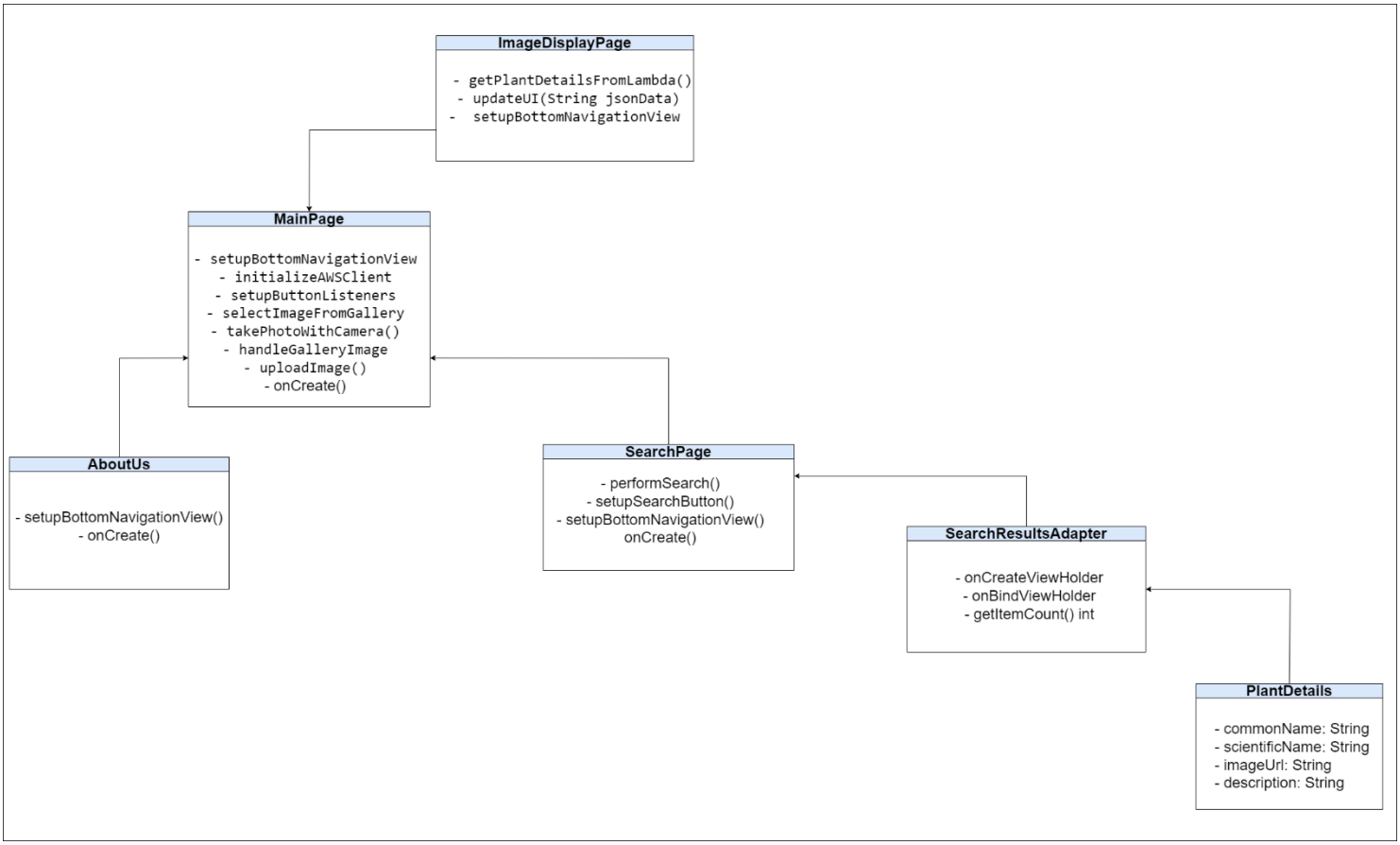


Figure 5 Class Diagram

2.3.2 MainPage Class

The MainPage class is the main class of the application. It is the java class that users of the application will see when first opening the application. This class gives a multitude of options for the user to select to do. Two buttons are apparent to the user within this class. These buttons are the “Upload” button and the “Camera” button, both supplied by icons from the drawable folder within AndroidStudio. These buttons are essential for functionality as they allow the user to access the main functionality of the application, this being the request to process images to be analysed through the AWS back-end.

Code Samples 1, 2 and 3 displays the functions that let the buttons be listeners. Upon interaction, one of the two functions will trigger depending on the button selected.
Code Sample 1 is the setup method for these buttons. Code Samples 2 & 3 are two different

functions depending on the button pressed. The design pattern used for these buttons are command patterns [24]

Using this design pattern is advantageous as it is reusable alongside supporting decoupling. This means that it'll execute the command when interacted along while not needing to know the details of the operation.

```
1.      private void setupButtonListeners() {
2.          binding.buttonUploadFile.setOnClickListener(v -> selectImageFromGallery());
3.          binding.buttonUseCamera.setOnClickListener(v -> takePhotoWithCamera());
4.      }
5.
```

Code Sample 1: MainPage Button Interactions

```
1.      private void selectImageFromGallery() {
2.          Intent pickPhoto = new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
3.          startActivityForResult(pickPhoto, REQUEST_IMAGE_UPLOAD);
4.      }
5.
```

Code Sample 2: MainPage Image Button

```
1.      private void takePhotoWithCamera() {
2.          if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED) {
3.              ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA},
MY_PERMISSIONS_REQUEST_CAMERA);
4.          } else {
5.              dispatchTakePictureIntent();
6.          }
7.      }
8.
```

Code Sample 3: MainPage Camera Button

After these buttons are utilized, users are then moved toward the ImageDisplay class in order to witness the information supplied from the AWS back-end in response.

2.3.3 ImageDisplay Class

The ImageDisplay class is responsible for supplying the information back to the user after they have sent their request from the MainPage class. Within the ImageDisplay class, the user will be given the plant name, description and an image supplied by the database. This class, along with MainPage, communicate with the Amazon Web Services back-end in order to supply the correct information based on the image uploaded to the AWS pipeline. This class utilizes different elements to display the information in the correct way.

Code Sample 4 shows the function inside the ImageDisplay class to return the information to the application, after being successfully processed within the AWS back-end.

```
 1.     private void getPlantDetailsFromLambda() {
 2.         OkHttpClient client = new OkHttpClient();
 3.         String lambdaEndpoint = "https://utf5cdwynh.execute-api.eu-west-
1.amazonaws.com/PlantDefaultStage/flowers"; // API Gateway endpoint
 4.
 5.         Request request = new Request.Builder()
 6.                 .url(lambdaEndpoint)
 7.                 .build();
 8.
 9.         client.newCall(request).enqueue(new Callback() {
10.             @Override
11.             public void onFailure(Call call, IOException e) {
12.                 e.printStackTrace();
13.                 Log.e("LambdaAPI", "Network request failed: " + e.getMessage());
14.                 // Handle failure
15.             }
16.
```

Code Sample 4: Receiving information to display plant details

2.3.4 SearchPage Class

SearchPage is the class that enables users to be able to query searches within the application in order to display and learn more about specific plants that they would want to see. This class communicates with AWS also to fetch, receive, and display the search results and their corresponding values.

Code Sample 5 shows this operation in action via the query validation function.

```
1.     public String validateQuery(String query) {
2.         if (query == null || query.trim().isEmpty()) {
3.             return "Please enter a search query.";
4.         }
5.         return "Valid";
6.     }
```

Code Sample 5: Search Query

An additional feature of this functionality is the ability for users to be able to click one of the plants displayed in the list returned and be given a pop-up description if the user selects one of the plants given back by the list.

2.3.5 SearchResultsAdapter Class

SearchResultsAdapter is the class that supports SearchActivity to be able to list the search results. The code snippet provided below shows this functionality.

```
    public static class ViewHolder extends RecyclerView.ViewHolder {
         TextView nameTextView;
         TextView sciNameTextView;
         ImageView imageView;

         public ViewHolder(View itemView) {
             super(itemView);
             nameTextView = itemView.findViewById(R.id.textview_plant_name);
             sciNameTextView = itemView.findViewById(R.id.textview_plant_sci_name);
             imageView = itemView.findViewById(R.id.imageview_plant);
         }
     }

```

Code Sample 6: Search Function to return information from query.

2.3.6 PlantDetails Class

PlantDetails is a child class that exists to hold the values of the data returned. These values would be information such as the plant name, plant description and image. Code Sample 7 shows this code as such.

```
private String commonName;
private String scientificName;
private String imageUrl;

private String description;

public PlantDetails(String commonName, String scientificName, String imageUrl, String
description) {
    this.commonName = commonName;
    this.scientificName = scientificName;
    this.imageUrl = imageUrl;
    this.description = description;
```

Code Sample 7: Function that gets plant details.

2.3.7. Amazon Web Services Configuration

Within AWS many different configurations and setup were required in order for AWS to properly have a pipeline to act as the cloud back-end for this application. A visual display of this pipeline can be seen within Figure 3, inside the Design & Architecture section of this report.

2.3.7.2 AWS Identity and Access Management

AWS Identity and Access Management often referred to as IAM is a service that allows users to be able to be created, alongside corresponding roles and policies in order to properly

enact the Principle of Least Privileged to these users. The FloralScan application when accessing the AWS backend assumes the role of an IAM “user” in order to properly access the AWS resources. In order for it to interact with the services setup, a dedicated role alongside policies were created for it.

Inside Code Sample 8 the policy JSON code configured for it can be seen. Certain aspects within this policy have been adjusted for the report in order to not expose the certain elements of information for the services mentioned.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::floralscaninput/*",
                "arn:aws:s3:::floralscaninput"
            ]
        },
        {
            "Sid": "Statement2",
            "Effect": "Allow",
            "Action": [
                "rekognition:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Statement3",
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:DescribeTable"
            ],
            "Resource": [
                "arn:aws:dynamodb:[Database-ARN]:table/FloralScanDatabase"
            ]
        }
    ]
}

```

Code Sample 8: IAM JSON policy

2.3.7.3 AWS S3

AWS S3 was implemented through being the storage system for the user image requests, alongside being the central service used to train the AWS Rekognition image recognition function. AWS S3 was

be used to hold the dataset AWS Rekognition requires to be able to train the model in order to have the image analysis functionality working.

The Lambda function was utilised in order to invocate code whenever a user request was received from the S3 bucket. This image would be processed toward AWS Lambda, and AWS Lambda would invoke other services through code within the pipeline until the operation has successfully operated, before passing the result toward the AWS API Gateway.

Code Sample 9 displays the code inside the Lambda function that instructs the database to be invoked for searching via the plant name.

```
common_name = labels[0]['Name']

        response = dynamodb.query(  # Plant details from db
            TableName='FloralScanDatabase',
            IndexName='common_name-index',
            KeyConditionExpression='common_name = :common_name',
            ExpressionAttributeValues={':common_name': {'S': common_name}}

```

Code Sample 9 Lambda DB Query

Snippet of code that instructs the image entered into the S3 bucket that invokes the Lambda function to be processed via AWS Rekognition.

```
    PROJECT_VERSION_ARN = 'arn:aws:rekognition:[Project-
Arn]:project/FloralScanProject/version/FloralScanProject.2024-02-11T18.44.28/1707677063823'

    try:
        # Analyze image via Rekog
        rekognition_response = rekognition_client.detect_custom_labels(
            ProjectVersionArn=PROJECT_VERSION_ARN,
            Image={
                'S3Object': {
                    'Bucket': 'floralscaninput',
                    'Name': 'PhotoInput/gallery_image.jpg'
                }
            },
```

Code Sample 10 AWS Rekognition invoked within the Lambda function

2.3.7.4 AWS Rekognition

AWS Rekognition was implemented through the Lambda code, alongside configuring the service through Custom Labels in order to have a model that can analyse different features of different plants and output a result. Through amassing training data under the license of Public Domain and free to use source such as the 102 Oxford Dataset [25] and the “Flower Image Dataset” [26] a model for this application was trained and is used within Lambda for the function to properly operate. Implementation of it to be invoked was done through using the specifics models resources inside the Lambda function code. Refer to Code Sample 10.

2.3.7.5 AWS DynamoDB

AWS DynamoDB was used as the database system for the AWS cloud pipeline back-end. By keeping the database native to AWS, this makes integration of it with the rest of the services easier and much more coherent. The database was trained via the Perenual API [13] through a script that allowed the name, description, and image values to be saved within the DynamoDB. This use of DynamoDB, rather than grabbing directly from the Perenual API means in the event the API faces a downtime outage, the database is still readily available to be used for the application. By setting the partition key within the database as the common_name this allows the previously mentioned Lambda function to properly query the names returned by the AWS Rekognition analysis and find the nearest match within the database.

2.3.7.6 AWS API Gateway

Once the entire process is finished, the AWS API Gateway is available as an endpoint for the application to receive these values and display them as needed. This endpoint is provided to the application code within AndroidStudio in order for this to be possible. Following this, styling is used in order make it presentable.

Code Sample 11 displays the API Gateway endpoint to display information within application, adjusted not to expose sensitive information.

```
1. private void getPlantDetailsFromLambda() {
    OkHttpClient client = new OkHttpClient();
    String lambdaEndpoint = "https://[API-Gateway].execute-api.eu-west-
1.amazonaws.com/PlantDefaultStage/flowers"; // API Gateway endpoint

    Request request = new Request.Builder()
            .url(lambdaEndpoint)
2.
```

Code Sample 11 AWS API Gateway Endpoint

2.4. Graphical User Interface (GUI)

This section of the report will showcase the various screens that users will see whilst using the FloralScan application.

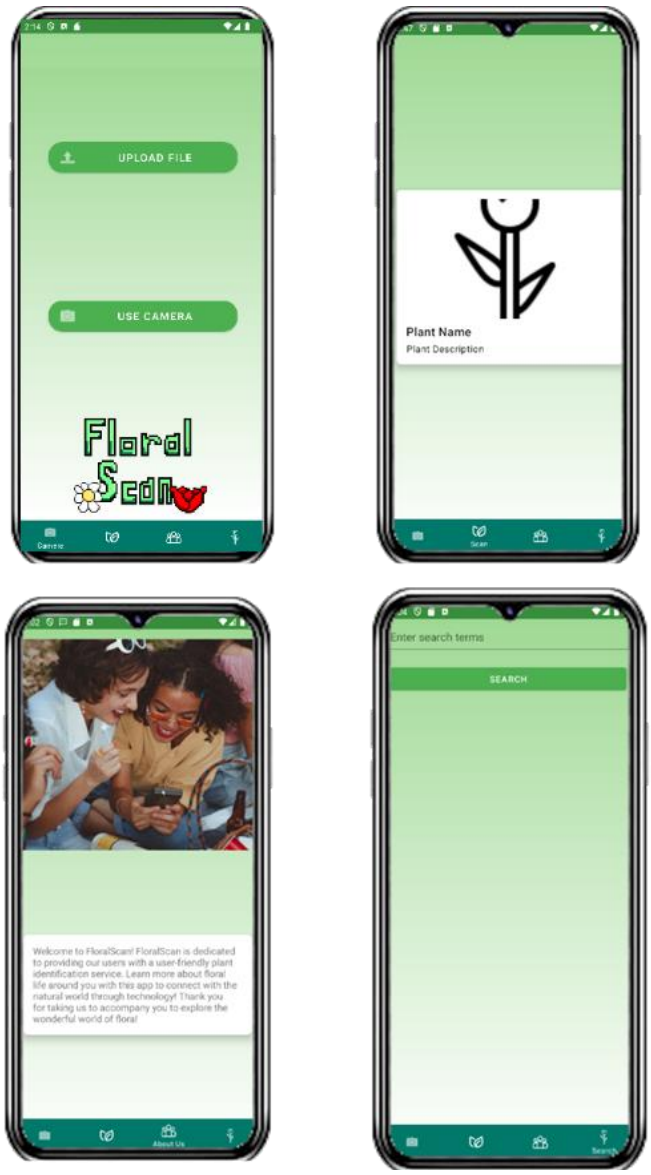


Figure 6 FloralScan GUI Pages. Top Left: Main Page. Top Right: ImageDisplay. Bottom Left: AboutUs. Bottom Right: SearchPage.

For users who have just opened the application from a fresh session, the flow will start them onto the main page.

From here, users have a multitude of options depending on what actions they want to proceed to do.

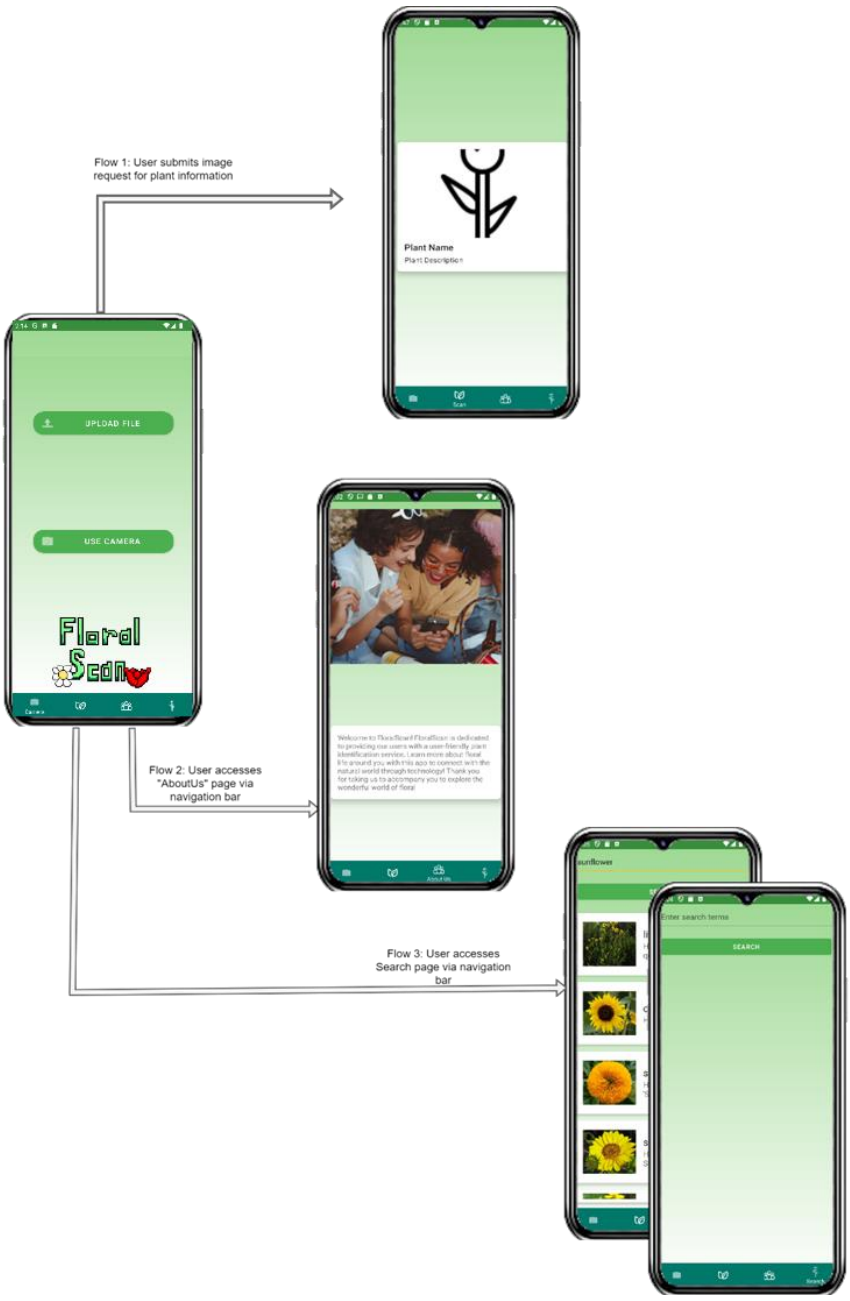


Figure 7:FloralScan Flow
Flow 1: User submits image request to for plant information
Flow 2: User accesses "AboutUs" page via navigation bar
Flow 3: User Accesses Search Page Via Navigation Bar

Users have the choice to either start the process immediately by uploading an image from their camera of a plant they have taken a picture of. They also have the option to upload an image of a plant from their image gallery if they would prefer that. The application will prompt for permission for the user to do either of these actions which they can decline if they do not feel as if they want to proceed with the action.

If accepted, the flow will lead the user to the ImageDisplay page that will provide the output from the AWS back-end operations.

Alternatively, users can navigate to the Search page via the navigation bar in order to search for specific plants they may want more information upon. Successful matches to the users search query will display a list of the nearest-matching definitions found within the applications DynamoDB database. When one of these search results are clicked upon, a description pop-up appears for the user to read the information supplied of the plants description.

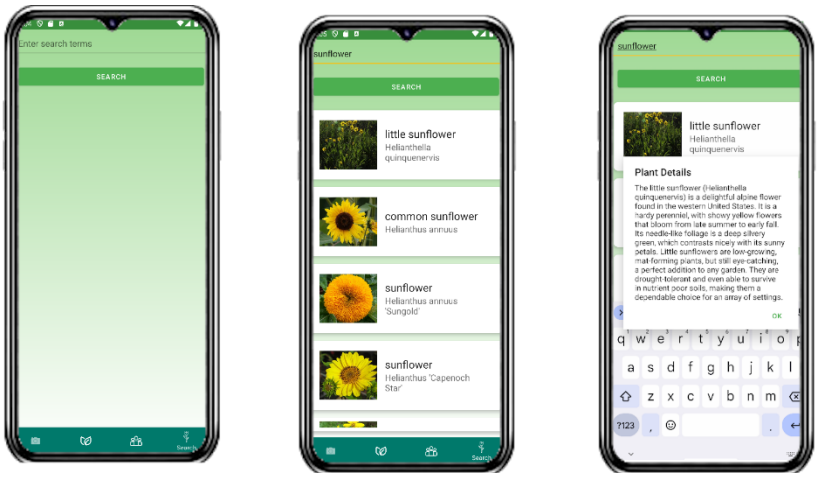


Figure 8: FloralScan GUI
About Us page with list & description feature

The final page of the GUI of this application is the AboutUs page of the application. This page of the application has an auto-scrolling preview wheel of images alongside a description of the application for new users to become acquainted.

2.5. Testing & Evaluation

Tests for the FloralScan application have been executed through different software available to perform Unit testing, Integration testing, Performance Testing and SMART testing and branch coverage testing. Through these tests an evaluation was able to be conducted on the system overall.

Integration Tests

I have performed integration tests for the image upload functionality invocation via the Lambda function in order to ensure the S3 bucket is properly receiving and storing the image request from the user. Integration testing [18] is the phase which these different modules and services are tested as a combined unit in order to ensure proper functionality where expected.

AWS Lambda provides a "Test" feature within their software which allows me to test both the code and the interactions of different AWS services between each other to ensure everything is functioning as expected. The unit test used here allows me to validate that the Lambda function reacts properly to the image upload event from users within the application.

```
{
  "Records": [
    {
      "eventSource": "aws:s3",
      "awsRegion": "eu-west-1",
      "eventName": "ObjectCreated:Put",
      "s3": {
        "bucket": {
          "name": "floralscaninput",
          "arn": "arn:aws:s3:::floralscaninput"
        },
        "object": {
          "key": "PhotoInput/gallery_image.jpg"
        }
      }
    }
  ]
}

```

Code Sample 12 Integration Test Setup

The output of this test is as displayed below:

```
{
  "statusCode": 200,
  "body": "{\"common_name\": {\"S\": \"sunflower\"}, \"description\": {\"S\": \"The sunflower
(Helianthus divaricatus) is a beautiful and hardy perennial flowering plant. This impressive
bloom is known for its bright yellow flowers and typically grows to heights of up to three feet
tall. Its large paddle-shaped petals, decorated with bright yellow center disks, bring joy and
```

```
beauty to any flower garden. Sunflowers are incredibly easy to grow and require minimal effort
to thrive. The flowers are attractive to bees and other pollinators and provide years of
stunning beauty. Sunflowers are an incredible addition to any garden, and they make a delightful
border or accent plant.\"}, \"image_url\": {\"S\":
\"https://perenual.com/storage/species_image/3387_helianthus_divaricatus/regular/51190351654_20a
bbf1e57_b.jpg\"}, \"id\": {\"N\": \"3387\"}}"
 }
```

```
1. START RequestId: 8c2c81e9-221a-49d3-aefa-ef7e175ff1c3 Version: $LATEST
2. END RequestId: 8c2c81e9-221a-49d3-aefa-ef7e175ff1c3
3. REPORT RequestId: 8c2c81e9-221a-49d3-aefa-ef7e175ff1c3  Duration: 685.91 ms Billed Duration:
686 ms    Memory Size: 128 MB Max Memory Used: 81 MB
4.
```

Code Sample 13 Integration Test Results

This code is validated to be correct through the output through different things displayed within the log. The output showing the plant name, description and image values are indicative that they have been properly fetched within the DynamoDB function from the piece of code that looks to get these values.

```
response = dynamodb.query(  # Plant details from db
2.              TableName='FloralScanDatabase',
3.              IndexName='common_name-index',
4.              KeyConditionExpression='common_name = :common_name',
5.              ExpressionAttributeValues={':common_name': {'S': common_name}}
6.          )
7.
```

Code Sample 14 DynamoDB Query

The label of the sunflower plant is a successful reading from AWS Rekognition also, as before this within the Lambda function AWS Rekognition is instructed to analyse the plant received and return the label name in order for DynamoDB to be invoked and search for the details of the string if available.

```
1.          rekognition_response = rekognition_client.detect_custom_labels(
2.             ProjectVersionArn=PROJECT_VERSION_ARN,
3.             Image={
4.                 'S3Object': {
5.                     'Bucket': 'floralscaninput',
6.                     'Name': 'PhotoInput/gallery_image.jpg'
7.                 }
8.             },
9.
```

Code Sample 15 Rekognition Custom Label Invocation

Unit Testing

Unit tests [17] for the application have been carried out within AndroidStudio. A unit test is a software testing method that isolates a piece of code within a system in order to view how this piece of code performs, and if there re any errors that may come up from different test cases of this code.

An example of unit testing within my application consists of the Search functionality within the SearchPage class. Within this class, many different operations are active but in particular the search button is important to test as that is the functionality that allows the rest of the functionalities within that class to operate.

```
testSearch (new String[]{"rose", "lily", "daisy", ""});

private void testSearch(String[] queries) {
    for (String query : queries) {
        testHardcodedSearch(query);
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

private void testHardcodedSearch(String query) {
    runOnUiThread(() -> {
        searchInput.setText(query);
        searchButton.performClick(); //clicks
    });
}
```

Code Sample 16 Unit Tests

In this setup, the instance of the search bar will enter the hardcoded queries of the different plant names selected from the array and iterate through them. After the entries the program simulates the search button click in order to initiate the search function. Each query is expected to interact with the backend as expected and in the scenario an error occurs, error handling is implemented to see what is causing the error.

Through these tests I am able to examine the applications behaviour to different scenarios and how the features may interact with them.

SMART Testing

Smart Testing [15] was performed on the FloralScan application in order to verify that the application met all of the SMART requirements. SMART testing is an acronym for Specific, Measurable, Achievable, Relevant and Time-Bound.

The **Specific** requirement was achieved locally through individually testing the features of the application through both the emulator provided by AndroidStudio [23] and my own Samsung mobile phone. Every feature present was tested, alongside elements loaded through different interactions of the application, for example the definitions

fetched and displayed by from the search functionality and the image scrollwheel that is present within the AboutUs class. The objective of this was to ensure every feature met its requirement relevant to its use case.

The **Measurable** requirement was measured through using the Profiler tool in AndroidStudio in order to analyse and properly gauge the performance of the application. This test measured the CPU and memory usage from each class in the application from my Samsung S10 mobile phone.

MainPage

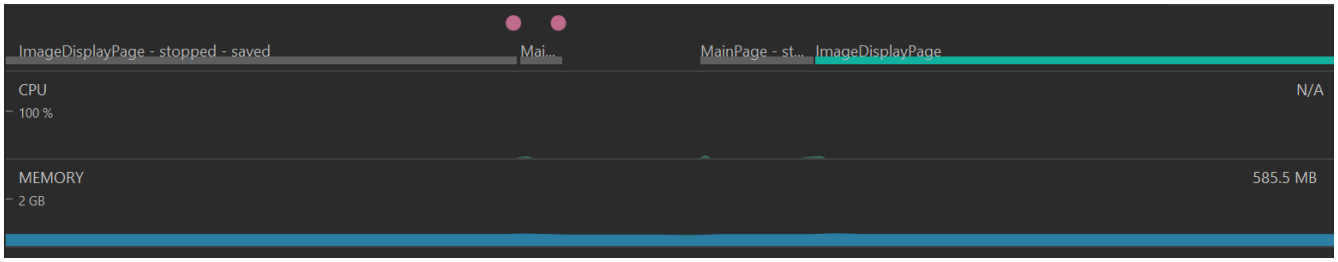


Figure 9: MainPage Performance Test
CPU & Memory

ImageDisplayPage

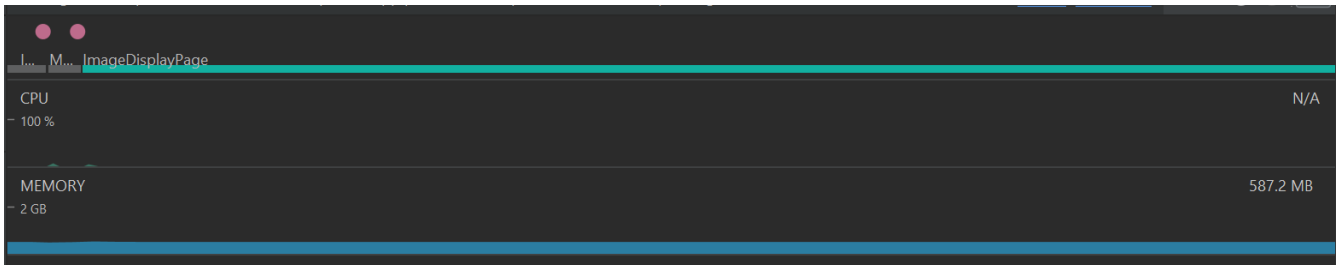


Figure 10: ImageDisplay Performance Test
CPU & Memory

AboutUs page

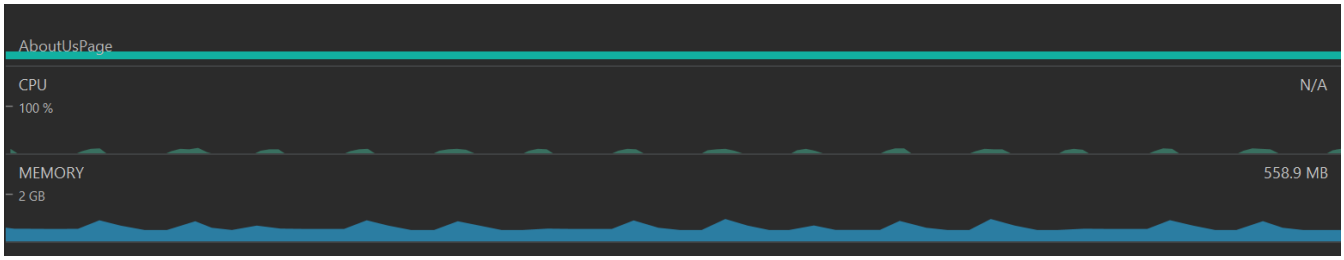


Figure 11: AboutUs Page
CPU & Memory

SearchPage

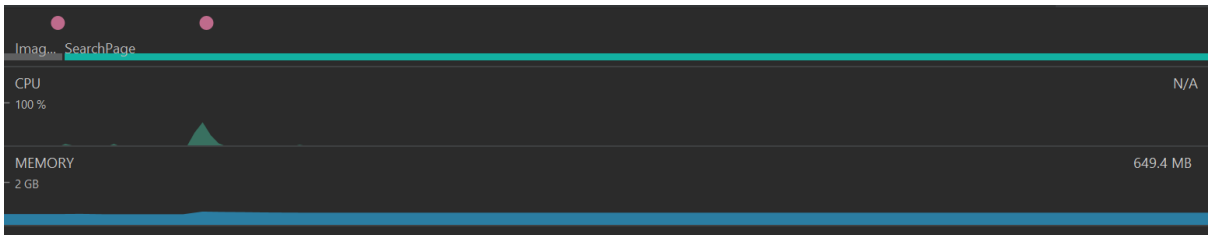


Figure 12: SearhPage Performance Test
CPU & Memory

The **Achievable** requirement was attained through being able to test the application and its features during the sprint periods of development for the application. This made it easier to both see new features as they were added and also to fix them as they occurred.

The **Relevant** requirement was achieved through verifying that the application successfully was able to complete the main operation it was designed for alongside fulfil its several use cases.

The **Time-Bound** requirement was met through manually testing the application over the course of the development period of the application. This made it easier to verify the features that were functioning and the features that were fully complete before beginning development on the next feature or design element of the application confidently. [15]

Branch Coverage Testing

Table 1: Branch Coverage Testing

Branch Coverage Testing	Test Case	Code Coverage	Total Number of Statements
Main Page	1	25%	4
ImageDisplayPage	2	50%	8
AboutUs	3	75%	12
SearchPage	4	100%	16

Branch Coverage Testing [16] is a testing method used to evaluate the effiency and functionality of the navigation bar code between each page of the application where applicable. Each test case of this testing method was done via the navigation bar between each page of the application. Every page was tested from each other to verify that it properly functioned for each pathway of the code.

Inside the Code Sample 17 from the MainPage, the navigation bar is built to be able to successfully interact with other pages depending on if the MainPage is open, and if the MainPage needs to transition to these pages without mixing them up or giving the incorrect page.

```
 1.  1. if (item.getItemId() == R.id.navigation_home) {
 2.  2.                 // MainPage already displayed
 3.  3.                 return true;
 4.  4.             } else if (item.getItemId() == R.id.navigation_dashboard) {
 5.  5.                 intent = new Intent(this, ImageDisplayPage.class);
 6.  6.             } else if (item.getItemId() == R.id.navigation_notifications) {
 7.  7.                 intent = new Intent(this, AboutUsPage.class);
 8.  8.             } else if (item.getItemId() == R.id.navigation_search) {
 9.  9.                 intent = new Intent(this, SearchPage.class);
10. 10.             }
11. 11.
12.
```

Code Sample 17 Navigation Bar Logic

This differs from the code inside the ImageDisplayClass, shown by Code Sample 18.

```
 1.      if (item.getItemId() == R.id.navigation_home) {
 2.                 intent = new Intent(this, MainPage.class);
 3.             } else if (item.getItemId() == R.id.navigation_dashboard) {
 4.                 // ImageDisplay is already open
 5.                 return true;
 6.             } else if (item.getItemId() == R.id.navigation_notifications) {
 7.                 intent = new Intent(this, AboutUsPage.class);
 8.             } else if (item.getItemId() == R.id.navigation_search) {
 9.                 intent = new Intent(this, SearchPage.class);
10.             }
11.
```

Code Sample 18 Navigation Bar Logic via ImageDisplay Class

The Branch Coverage testing was used to test and validate that these operations functioned correctly. As each page had 4 testing paths through the navigation bar, each test case had 4 statements to be tested. This being the page that the application is currently on, and the navigation of the remaining three pages.

For example, testing within the MainPage would include selecting the navigation bar icon that would bring a user toward the MainPage, and then to the ImageDisplay, followed by

the AboutUs page and finally the SearchPage. The MainPage is tested in this scenario to ensure that the active page does not bring the user to any other page, and instead is ensured that it stays on the correct page.

User Acceptance Testing

Acceptance testing is a testing method used to evaluate if the system meets the user requirements and the use case scenarios. This testing method is useful as it is the testing method that helps test the application before releasing it to production. For this application, the tests were used to test the core functionality and adjust these features as needed before finishing the project.

Within Table 2, these tests can be seen.

Table 2 User Acceptance Testing

ID	Test Case	Pass/Fail	Outcome Description	Date Tested
1	Correct plant returned when identifying picture of a dandelion	Pass	Information of “Dandelion” is returned.	09/05/2024
2	Correct plant returned when identifying picture of a daisy.	Fail	Information of “African Daisy” is returned. To subside this, adjust labels between “African Daisy” and “English Daisy” within AWS Rekognition.	09/05/2024
3	Correct plant returned when identifying gallery image of rose	Pass	When uploading picture of rose from image gallery, the correct information is returned.	09/05/2024
4	Correct plant returned when	Pass	Information of sunflower	09/05/2024

	identifying gallery image of a sunflower		uploaded from the image gallery, the correct information is returned	
5	Using the search query functionality, search query of “lily” returns list of plants relevant to this search, prioritising the tulips at the top of the list.	Pass	Information of “lily” plant is returned alongside relevant searches. This includes the white “Lily of the Nile”, “African Lily” and the blue “Lily of the Nile	09/05/2024
6	Entering a blank query into the search box does not search anything, prompt for user to search appears.	Pass	Functionality acts as expected, nothing is searched when this happens and instead prompts the user to search for a plant name	09/05/2024

3.0 Conclusions

The FloralScan application has been developed and designed to provide a platform for the younger audience to be able to help them discover and learn about the plant life around them. This application has been designed in a manner to allow it to be easy to use from a user experience standpoint without many complications behind how to use it. This simplistic and user-friendly interface makes it alluring for the userbase of this application.

This application holds its advantages through this design philosophy, alongside providing a complex functionality presented in a straightforward way that users do not need to burden themselves learning to have it properly function. By offering different upload methods through using their mobile phone camera or an image directly from their gallery, they are given the choice and flexibility on how they want to use the application. The application,

providing an in-house search functionality is another advantage to it as it allows the user to query a quick search without needing to leave the application in order to do this. It provides an extra element of user engagement with the application.

This application comes with its own unique limitations and flaws, however. For example, as the analysis relies upon the quality of the image submitted by the user, the image analysis function relies upon the user to submit images that are of high quality, or clear. Additionally, the application relies on the user to submit a focused image of the plant image to have a higher chance of the correct plant details returned to them. Adding to this the application does not accept MP4 video formats, resulting in the plant details returned being empty, displayed with the placeholder text of the plant name, image, and description.

Addressing both the advantage of this application allows a thorough evaluation and conclusion of the project. The application has its faults; however, it is a useful tool for the users who will use it to broaden their learning of plant life around them in their everyday lives.

4.0 Further Development or Research

The FloralScan application prides itself on having effective plant recognition relevant to the userbase through allowing them to scan plants around them in their own personal lives at the ease of their mobile phone. The application also prides itself on the speed that this can be accomplished with alongside the search functionality to further the userbases learning experience of floral life around them.

Further development of this application if given the time would add functionalities that would increase both user engagement and user learning. These developments would come in the form of centralized discussion boards that would allow users to connect with each-other and be able to exchange tips, information, and their own findings of floral life.

Achieving this would require the use of a user account creation system through AWS Cognito. Through implementation of this through AWS Cognito and the AWS SDK, users would be able to create their accounts through different methods. This can be through their own Google account if agreed to, or with a fresh account native to the application with their own email address. To uphold a server to constantly have this discussion feature possible, AWS EC2 would be used to host this space in order to have resources actively ready for new messages. AWS EC2 is suitable in this scenario for additional reasons such as its ability to be scalable during higher workload periods where resources are prone to be stressed out.

In addition to this, users with accounts can contribute to the application as a whole by helping to contribute plant images that be missing, incomplete or of low quality within the

database. This would be seen from the Search page. An administrator user will be queried these different images to see if they are appropriate for the plant description and name in order to properly ensure the correct plant is being placed into the database.

To increase user engagement, a functionality that gives users randomized daily “quests” or “missions” could be incorporated, encouraging users to go out into their local environment and complete these missions within a weekly timeframe. These elements would include objectives such as “Scan three Roses using the camera” and “take a picture of a Dandelion” for example. These concepts, though simple, can make a large difference toward the user experience whilst using the FloralScan application.

An alternative training data method would be utilised in a future development version of the FloralScan application if possible. AWS Rekognition works efficiently alongside the other AWS services and was easier on the development aspect, however Custom Labels having a restriction of 250 labels max heavily brought down the number of plants used within the dataset. In a future update of this application, an alternative tool would be used in order to encompass a larger number of plants to be able to be analysed. Ideally, using a software such as OpenCV [27] and then hosting it within an EC2 instance in AWS would still maintain the cloud aspect of this project whilst working around the restriction.

5.0 Project Poster

SOFTWARE DEVELOPMENT

National College of Ireland

FloralScan

By Taziwa Mushayabasa

Overview

Experience the ease of plant information supplied with the snap of a camera with FloralScan! This application is an innovative mobile application used to capture plant life through pictures to identify them instantly. Simply snap a photo and allow the application to scan it to be returned detailed information of the plant!

Motivation

The motivation behind the development of this application was to further integrate the outside world with the growing technologies of today.

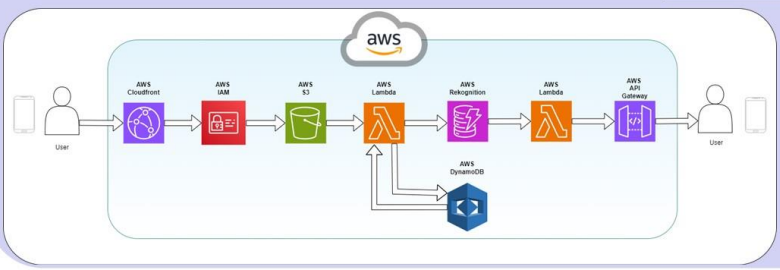
Those who possess a passion for botany who are looking for an interactive entry point will appreciate this application.

How It Works

This is a mobile application that utilizes the users' camera and Amazon Web Services as an online back-end system in order to process the operations.

By using the application interface and taking a picture, the application will return the desired information.

Technologies Used



Python

XML

JSON

Java

Figure 13 Project Poster

6.0 References

[1] Ryan Pankau (2022) “How accurate are photo-based plant identification apps? “Available at: https://extension.illinois.edu/blogs/garden-scoop/2022-01-21-how-accurate-are-photo-based-plant-identification-apps” [Accessed at 25/03/2024]

[2] Glority LLC Limited (2024) PictureThis Available at: https://www.picturethisai.com/ [Accessed at 25/03/2024]

[3] 2024 PlantSnap Inc (2024) PlantSnap Available at: https://www.plantsnap.com/

[4] Nalani Straight (2023) “How To Identify Plants With Google Lens Available at: https://robots.net/tech/how-to-identify-plants-with-google-lens/

[5] 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. *Educational Dimension* [Online], 1, pp.219–234. Available from: https://doi.org/10.31812/educdim.v53i1.3844 [Accessed 25 April 2024].

[6]Ryan J Schmidt, Brianna M Casario, Pamela C Zipse and Jason C Grabosky (2022) “An analysis of the accuracy of photo-based plant identification applications on 55 tree species” Available at:

https://scholarship.libraries.rutgers.edu/esploro/outputs/991031655349704646 [Accessed at 05 May 2024]

[7] Amazon Web Services, Inc (2024) “Amazon Lambda” Available at: https://aws.amazon.com/lambda/ [Accessed at 05 May 2024]

[8]Amazon Web Services, Inc (2024) “Amazon DynamoDB” Available at: https://aws.amazon.com/dynamodb/ [Accessed at 05 May 2024]

[9]Amazon Web Services, Inc (2024) “Amazon S3” Available at: https://aws.amazon.com/s3/ [Accessed at 05 May 2024]

[10] Amazon Web Services, Inc (2024) “Amazon Rekognition” Available at: https://aws.amazon.com/rekognition/ [Accessed at 05 May 2024]

[11] Amazon Web Services, Inc (2024) “AWS Identity and Access Management Documentation” Available at: https://docs.aws.amazon.com/iam/ [Accessed at 05 May 2024]

[12] Amazon Web Services, Inc (2024) “Amazon API Gateway” Available at: https://aws.amazon.com/api-gateway/ [Accessed at 05 May 2024]

[13] Perenual (2024) “Plant API Documentation” Available at: https://perenual.com/docs/api [Accessed 05 May 2024]

[14] Amazon Web Services, Inc (2024) “Amazon Cloudfront” Available at: https://aws.amazon.com/cloudfront/ Available at: [Accessed 05 May 2024]

[15] Vicky Di Ciacca (2024) “Get Smart About Writing Requirements” Available at: https://www.be-positive.co.uk/blog/get-smart-about-writing-requirements-2/ [Accessed 05 May 2024]

[16] Carlos Schults (2021) “What is Branch Coverage and What Does It Really Tell You?” Available at: https://linearb.io/blog/what-is-branch-coverage [Accessed 05 May 2024]

[17] Amazon Web Services, Inc (2024) “ What is Unit Testing?” Available at: https://aws.amazon.com/what-is/unit-testing/#:~:text=Unit%20testing%20is%20the%20process,test%20for%20each%20code%20unit. [Accessed 05 May 2024]

[18] Katalon (2024) “What is Integration Testing? Definition, How-to, Examples” Available at: https://katalon.com/resources-center/blog/integration-testing [Accessed 05 May 2024]

[19] fauxels (2019) “Man Taking a Photo of a Flower” Photo by fauxels from Pexels: https://www.pexels.com/photo/man-taking-a-photo-of-flower-3228769/ [Accessed 05 May 2024]

[20] Alina Viana Prado (2019) “Shallow Focus Photo of Woman Using Game boy” Available at: Photo by Aline Viana Prado from Pexels: https://www.pexels.com/photo/shallow-focus-photo-of-woman-using-game-boy-3491940/ [Accessed 05 May 2024]

[21] Keira Burton (2020) “Multiracial positive male and female students using smartphones in city park” Available at: Photo by Keira Burton from Pexels: https://www.pexels.com/photo/multiracial-positive-male-and-female-students-using-smartphones-in-city-park-6146931/ [Accessed 05 May 2024]

[22] dotPDN LLC (2023) Paint.NET Available at: https://www.getpaint.net/ (Accessed 09 May 2024).

[23] Android Studio (2023) Available at: https://developer.android.com/studio (Accessed 09 May 2024).

[24] GeeksforGeeks (2024) Available at: https://www.geeksforgeeks.org/command-pattern/ (Accessed 09 May 2024)

[25] Nilsback, M.-E. and Zisserman, A. (2008) 'Oxford 102 Flower Dataset', Available at: http://www.robots.ox.ac.uk/~vgg/data/flowers/102/ (Accessed: 09 May 2024)

[26] Aksha05 (2020). 'Flower Image Dataset' Available at: https://www.kaggle.com/datasets/aksha05/flower-image-dataset (Accessed: 09 May 2024)

[27] Bradski, G. and Kaehler, A. (2024). “OpenCV”. Available at: https://opencv.org/ (Accessed: 09/05/2024)

[28] 2024 PlantSnap Inc (2024) PlantSnap Available at: https://www.plantsnap.com/ (Accessed: 09/05/2024)

[29] Glority LLC Limited (2024) PictureThis Available at: https://www.picturethisai.com/ (Accessed: 10/05/2024)

[30] 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. *Educational Dimension* (Online), 1, pp.219–234. Available from: https://doi.org/10.31812/educdim.v53i1.3844 (Accessed: 10/05/2024)

7.0 Appendices

7.1. Project Proposal

Below, the project proposal can be viewed for further information about the application and the functions supplied with it.

Contents

- ## Objectives

This application addresses the challenge of accurately for individuals who lack knowledge about botanical life. This application would provide an interactive educational outlet for individuals who wish to learn or observe the different plant species both inside their local environment and toward broader plant life outside of it.

This project prioritizes the goal of creating a convenient, user-friendly environment that allow users of the application to identify the plant life around them through the ease of their mobile phone camera. If users do not have the mobile data, they have the option to still use the application later via an image uploader feature when they are connected to an internet network. Alongside this, the application aims to be accurate in the information returned to the user when a plant is scanned in. To keep user retention and overall user happiness, this process aims to also be real time and fast in response time to allow a smooth user experience.

This project prioritizes the goal of creating a convenient, user-friendly environment that allow users of the application to identify the plant life around them through the ease of their mobile phone camera.

Background

I chose to undertake this project because of the growing potential found within merging technology and machine learning into spaces that do not incorporate them yet, or where they are currently under-explored. Through creating this project, the potential of machine learning being used in these areas may be shown to be slightly more potent and given more time invested upon for where it may be able to be implemented next. In addition to this, cloud services at this current date are only growing larger in value and mixed with machine learning in these spaces could provide to be a valuable tool. AWS provides a multitude of cloud services that can be utilized for this purpose. As of writing this report, AWS contains over 200 cloud services available for users to utilize. AWS provides high performing, cost-effective, scalable infrastructure for users and businesses' alike to implement them into their software if they wish. AWS' machine learning service that I will be utilizing will be Amazon Rekognition. Amazon Rekognition is a machine learning service that provides scalable image and video analysis. Using deep machine learning techniques, it is suited for detecting objects within images and videos.

The target audience for this application is toward younger demographics who may take up botany as a hobby, are outside often or travel. An additional reason for the creation of this project is both because of the increasing use of mobile phones and the slow creep problem today of global warming. By encouraging users to use their mobile phones to witness the life present all around us, it may increase efforts to preserve it and contribute to a healthier environment. The ease and smoothness aimed for this project hopes to make this process easy and convenient for users of the application.

State of the Art

Similar applications and websites that exist for machine learning plant applications are Google Sense, Wikipedia and iNaturalist.

Google Lens is the furthest development of the available applications mentioned, as it also used machine learning to allow users of the application to have a smooth experience while using it. Google Lens is effective as it provides information in real time with a fast respond rate of a second. However, Google Lens falters in the area where it would require an internet connection to function alongside accuracy issues if the object captured is not commonly photographed.

iNaturalist is similar in the sense that it is a community driven board and requires users to share information and inform one another. Although this provides a strong sense of community, responses may take longer than an average user or professional would like due to iNaturalist not having the capacity to provide real time data.

The application in development is aimed to provide a convenient method to be able to identify these different plants through the ease of the mobile phone. Although the application response time may not be within the speeds of Lens, it will still provide useful by returning the desired information in real time. Returning the information in real time means the data will be analysed as soon as it is produced. A realistic response time for this application would be within the 5 to 10 second range. By simplifying the process of identification and education about floral life, this app encourages young people and old alike to explore the world around them, increasing the interest in botany accessible to a larger array of people.

• Technical Approach

To develop this application, I will be incorporating an agile methodology approach through sprints. Each sprint will have different goals to meet to ensure development is going on schedule for the final production. This process allows for both flexible and adaptable development of the application alongside ensuring that there can be useful feedback and learning from each phase that can be used into the next.

The first few sprints will include focus upon the functionalities of the application. This will be the AWS pipeline of setting up the S3 storage bucket, the AWS Rekognition software and the IAM user policies as an example. Alongside this, the basic user interface will be developed and set up within Android Studio. Shorter sprints will aim toward refining these features and making the mobile interface enhanced to improve user experience. The later sprints will incorporate focus toward polishing the application alongside making it fully functional to fulfil its purpose. This will include optimization of performance via AWS Cloudfront incorporation, deployment preparation and extensive testing in order to validate it is ready for potential users. Doing this ensures the application is prepared to be released while upkeeping the user needs, expectations, and experience.

This agile methodology will allow development of the application to be structured with a plan to follow, but also allows it to be flexible for changes in schedule or if re-evaluation is needed. This will help as it will be combined with the meetings to allow an overall polished application in the final stages of development. *To develop this application, I will be

incorporating an agile methodology approach through sprints. Each sprint will have different goals to meet to ensure development is going on schedule for the final production.

• Technical Details

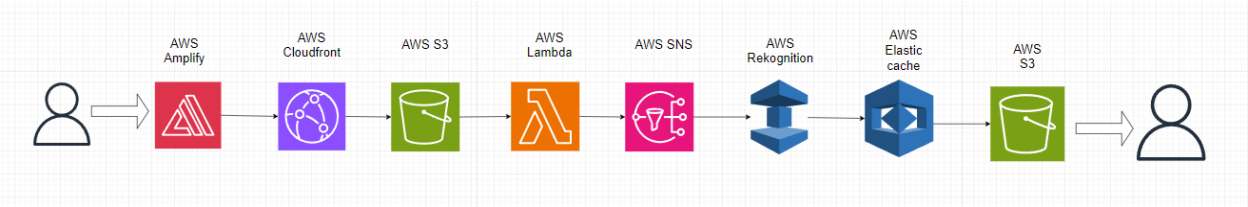


Figure 14: Initial AWS Pipeline

Undoubtedly the Cloud services used through AWS will be the most important aspect of the application functioning, otherwise the application workflow would be incomplete and would require different individual services to work alongside each other. This may cause issue if some of these services are incompatible with each-other and cause additional work upon the project. By using the Cloud services provided by AWS, an application workflow can be achieved easily and comfortably. AWS also provides many services that will be useful for the application to have. This includes core services such as a storage system, a hosting service, and services to hold code if needed, alongside services to help customer enjoyment of the application. Examples of these services are Global Accelerator and Cloudfront.

The implementation languages of the application will be centred around XML. Java will code functions needed inside the application and XML will be able to create the pages that the users will interact with. Android Studio allows an internal Android simulator that allows creation of the user interface while utilizing XML. This will help the application give a complete, polished look.

• Special Resources Required

This project will require several different special resources to be able to be completed. The following tools will be needed during development to be able to test and complete the application.

Android Studio is a development environment that will be needed for development of the application as it is mobile based. By using this IDE, access to coding, testing, and debugging

will be made easier for mobile interfaces. Alongside this, Android Studio provides the opportunity to emulate and test the user interface live through both a virtual phone displayed within the IDE and a physical mobile design. Android Studio allows an internal IDE editor that allows the aesthetic of the application for mobile to be built. Alongside this, Android Studio specializes in the XML language to code within the application. Through constant testing and tweaking of the application, will allow it to become a polished product once it leaves the prototype stage and becomes fully complete.

The use of AWS is the additional special feature needed to complete this product. By having access to the cloud services AWS provides the main objectives of the application can be achieved, alongside being able to achieve a seamless user experience through using the Cloud services provided by AWS.

- Project Plan

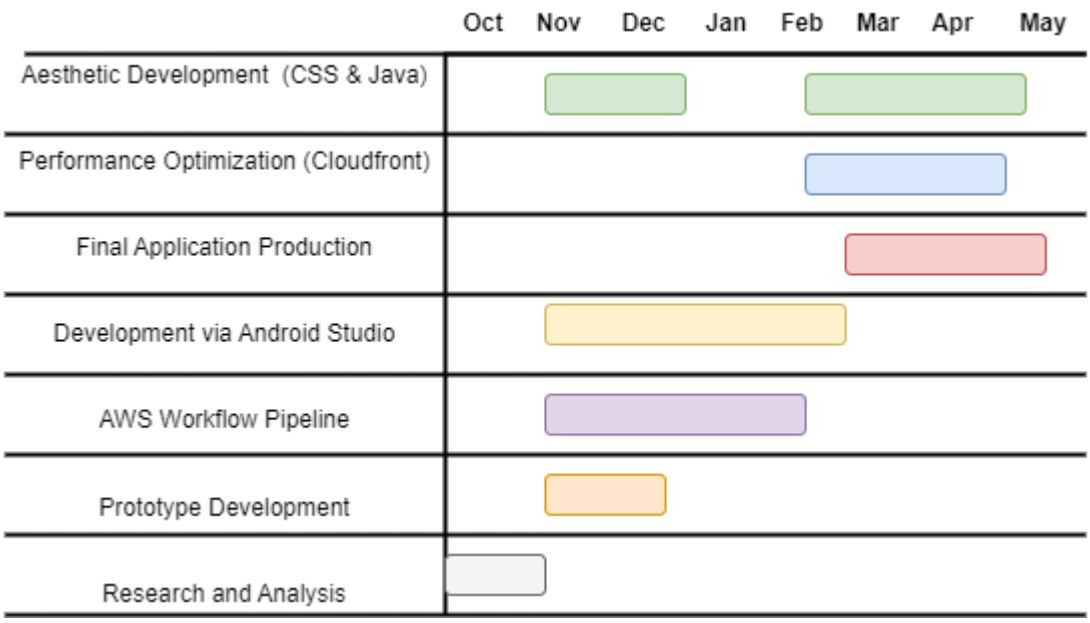


Figure 15: Project Plan

The project plan for this application will be as follows during the duration of the development of this project. For the starting weeks of the project extended research and analysis will be executed. During this period, focus towards key areas of development of the project such as wireframing and application workflow among the Cloud services will be done. Ensuring a successful Cloud application workflow in concept alongside how it will interact with the code within the chosen editor, Android Studio, will be essential to the success of the development of the project. Additionally, a general application workflow will be developed to witness the process of how a user may use the application, and the different possibilities of what may happen depending on different circumstances.

Following this section of the project, the prototype of the final product will now begin production. This will include the development and use of the application via Android Studio, AWS cloud services and different Api's that can be used to create the prototype. Android Studio is the choice of use due to its capabilities and versatile nature being about to be used for effective coding, alongside testing, debugging, and previewing the application within a virtual android UI. As it is the most feasible and recommended choice for android application development, it will be used for the development of the prototype application.

AWS and the cloud services provided by them are a crucial part of this applications development and success. As the cloud services provide both a reliable and secure environment, proper focus toward manufacturing the cloud application workflow will be important to ensuring the project's success. For these reasons, there will be dedicated time toward the development of how the Cloud services within AWS will interact with each other. Ideally, 1-2 weeks will be given to developing this workflow and connecting them to the application. Once the Cloud service workflow works adequately between each other the next objective for the prototype will be to connect it to the code used within Android Studio to achieve a working prototype. Adjustments may be needed to be made depending on the circumstances of the compatibility for the prototype.

After the prototype is developed the project plan will move toward the final application production phase. This phase will be critical as it aims to ensure the application components not only work efficiently between each other, but also flawlessly to a level that produces customer satisfaction. One key component of this will be the overall speed of the final application product. To achieve effective application speeds, AWS Cloud services such as Cloudfront will be used to make this a reality. Cloudfront is a service specifically made by AWS to be able to accel erate data transmission for users to be able to have faster results overall, regardless of where they are situated globally. Additionally, putting emphasis on the overall aesthetic of the application will be an important component of the final phase development that will be given time to as this is also to appease customer satisfaction. An application that can both be functionally and visually pleasing is the most important final deliverable.

8.0 Testing

Testing the system of the application will come through several testing methods. The testing methods used will be to ensure that proper functionality of the application is present throughout the difference areas within it. Different testing methods will be used in different areas of the application in this regard. The technical testing of the application will be through testing methods' such as End to End tests and performance tests. End to End tests is important as they ensure the entire system is functional and the workflow properly works when needed by a user. By using this testing method, tests will be executed via doing a full walkthrough of the application and the several pages available. Additionally, testing the identification system via a wide range of diverse plant images, alongside different angles of the same plant will assist with the recognition accuracy and the overall functionality of the application.

Performance tests will be achieved through using TestRail to assess the speed of the application. Speed within the application will be important as an application that processes requests at a slower rate may reduce customer satisfaction alongside reducing the desire for a user to want to use the application. This performance test will be done through measuring how the system can identify plants while handling several requests at once.

The overall application's functionalities such as the upload feature, will be tested in to ensure that the application has its preferred performance, user interface and processing for plant images. By testing this feature, it will be able to ensure the application is both intuitive and accessible for users, alongside providing the purpose it is intended to provide.

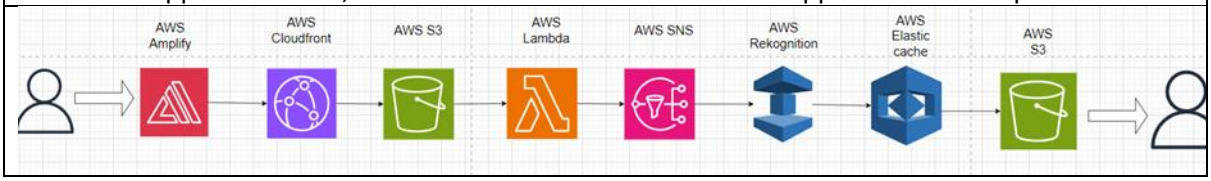
8.1. Reflective Journals

Month: October

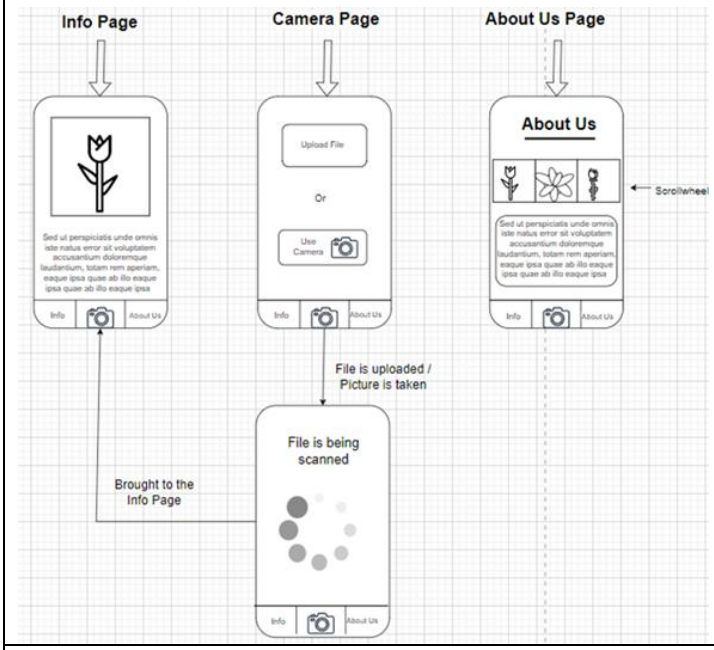
Student Name	Taziwa Mushayabasa
Student Number	20444686
Course	BSHCSD4
Supervisor	William Clifford

What?

For the month of October, progress on the project has been developed through wireframing, concepts and finalizations on the final application idea before starting development via code. The application was decided through both the convenience of mobile phones today and the importance of conservation of nature through awareness. By encouraging users of the application to learn more of the natural world around them, it will inversely help in the current fight against global warming. This application also can be used effectively by content creators and botanic enthusiasts alike who enjoy learning about the plant life around them. This application utilizes the mobile phone camera, allowing users to use it within the application being produced to receive and learn about plant life around them within real time. To achieve this, AWS cloud services will be used to be able to deliver the real-time functionality the application aims to provide. Utilizing my own previous experience with AWS cloud services, I have created a cloud application workflow alongside a wireframe to outline the services that will be used, alongside how they will be delivered to the user of the application. This information will be vital as development begins in the following month. Development toward the user interface of the application will be executed within Android Studio. Android Studio presents itself as a new editor I can challenge myself to create this application with, as I have not used this IDE to create applications in the past.



So What?



Now What?

During the month of November, focus on the development of the prototype application will begin. During this period, different components such as the AWS cloud workflow and the prototype layout within Android Studio will begin development.

Student Signature	

Month: November

Student Name	Taziwa Mushayabasa
Student Number	X20444686
Course	BSCHSD4
Supervisor	William Clifford

What?

Over the duration of this month, development upon the project has been expanded upon through the additions of supervisor meetings and the beginning stages of implementation. By being able to attend supervisor meetings regularly, I have been able to properly structure my development of the application, alongside articulate the concept and the ideas from the project into several diagrams. These diagrams will help those who are not familiar with my application from a software development standpoint to be aligned with the back-end structure that will be implemented into this application, and how they interact with one other. This is done through a multitude of charts such as a project wireframe to showcase the interface and how it can be interacted with for a user who is using the application. Additionally, an AWS pipeline diagram of the backend tools that will be implemented into the application was produced. A Gantt chart was produced to outline the structure of development moving forward for the development process of the application. User Stories for the application were developed to target key features that a user may look toward if they were use it. Following the theoretical and planning phase of the project, focus was shifted toward the development of the applications interface via Android Studio and the back end via AWS through the many services it provides. This includes configuration of the S3 bucket, Lambda code and Amazon Simple Notification Service.

So What?

Successes within this process arose from configuration of the AWS S3 bucket, and linking newly built AWS SNS and AWS Lambda, connected. This success did not come automatically, as further troubleshooting was conducted upon permission policies with AWS S3. Thankfully AWS provides AWS IAM In order to supply these different permissions and roles depending on how they are configured. Small successes have also been achieved through beginner development of Android Studio such as. The challenge of Android Studio remains as it utilizes XML mainly rather than HTML and CSS. Android Studio has an innate GUI for the mobile interface for designs, however they still require a level of expertise in order to fully maximise the capability of it, alongside the code operating correctly.

Now What?

For the remainder of the month of December, progress toward development of the prototype application will follow suit to the sprints utilized for the development of this application. By the end of the month of December, the application should have a basic interface that holds functionality by being able to upload images toward the S3 bucket. The interface will be within basic form as it will be in a prototype state. Focus on the functionality of the application will be focused upon. Ideally the image uploaded will be able to trigger the AWS pipeline to begin its order of operations. However, as it will be the prototype it may not return the analysed image back to the interface. To address current challenges of the application, planning will be key, and diagrams will be adjusted accordingly to stay within the proposed development cycle. Challenges will also be brought up with supervisor with the objective of finding a conclusion that is both suitable and effective.

Student Signature	

Month: December

Student Name	Taziwa Mushayabasa
Student Number	20444686
Course	BSHCSD4
Supervisor	William Clifford

What?

Over the course of the month of December progression for this project has been furthered at a consistent and stable rate. After strong application planning and theoretical development via wireframes, application workflow diagrams, use case diagrams and sequence diagrams the prototype had begun to be developed. The prototype reached its completion via its envisioned completion date via the agile methodology previously used before to plan out development of the project. The prototype consists of several different features. First

and foremost, the prototype had XML and Java code produced to have a stable environment to be shown within a mobile interface. Including this, both the main page for the prototype and the display page were created and made functional. For these pages to have their functionality, AWS needed to be configured and set up in order to be functional for the prototype standards. Creation of the S3 bucket and the corresponding access policies were created in order to be able to hold the images temporarily. IAM users for the application were created in order to securely upload into the bucket alongside implementing role policies in order to make the IAM user only able to do what is needed and restricted from anything else. Alongside this, access keys were created in order for the application to be able to access these images to display within the prototype.

So What?

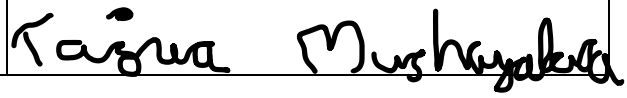
Consider what that meant for your project progress. What were your successes? What challenges still remain?

Successes included the successful styling for the prototype via Android Studio, configuration of the S3 bucket and the corresponding access policies, IAM user creation with least privilege, success via functionality within Android Studio alongside implementation of features such as switching via the navigation bar and implementing two different functionalities for uploading images via camera or image gallery. Another big success for this month was the connection between AWS and the prototype functioning as expected, showing that the application is now utilizing cloud services.

Now What?

Outstanding challenges currently include image detection via descriptions. This challenge will be addressed through researching free-tier API links that provide a simple, but effective description of certain definitions and plant names. Ensuring this is a free tier service is a must, as it will not fall under fair use if it is not free-tier and free to use for users and applications. Furthermore, an outstanding challenge is now complete

development of the application, including proper styling, efficient service time, customer satisfaction and ease of use. These different development goals alongside ensuring the AWS workflow is working as expected will be addressed through following the agile methodology planned in order to complete this application in the time expected, and to focus on the different aspects of the application as such.

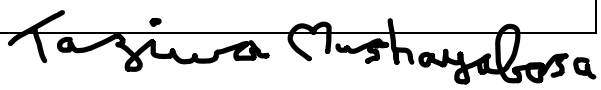
Student Signature	

Month: January

Student Name	Taziwa Mushayabasa
Student Number	20444686
Course	BSH Computing
Supervisor	William Clifford

What? In my project this month I'm preparing and learning the remaining technologies in AWS. I focused on the implemented AWS technologies, with a focus ensured onto AWS SNS. With AWS SNS I will be able to enable an automatic system that can enable functionality with both AWS Lambda and AWS Rekognition. Having this enabled will allow me to have my services decoupled. AWS Lambda has been successfully added as an endpoint for the S3 bucket, meaning Lambda will be invoked when an image lands into the S3 bucket. The complete development of AWS SNS will allow AWS Rekognition to be able perform its operations and begin analysis to be able to be returned to the user in the application.
So What? My successes in this would be the completed development of the AWS Lambda function in order to invoke Rekognition to be able to ideally start analysis of images. The challenges that remain are feeding Rekognition the data to be able to now properly analyse the floral objects put into the AWS S3 bucket that will be passed onto it to recognize and continuing development of the Amazon

Simple Notification Service to be able to properly set up these endpoints and keeping these services decoupled.
Now What? To address these outstanding challenges, I will access the official AWS forms to find assistance and guidance to properly carry these operations out. These forums are provided by AWS in order to provide rich technical support, advice and practices in order for developers using these services to be able to have guidance on how to properly manage, build and potentially troubleshoot common issues that they may be facing. With these resources in mind, I will be able to develop the AWS SNS component in my project successfully.

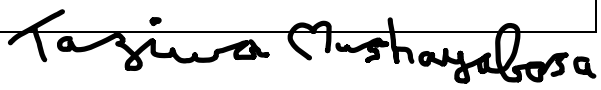
Student Signature	

Month: February

Student Name	Taziwa Mushayabasa
Student Number	X20444686
Course	BSHCSD4
Supervisor	William Clifford

What? Within this month of development for the application multiple back-end processes have become closer to completion. The images compiled into multiple different folders with the labels of the plant species had been uploaded into the S3 bucket that will be used to train the AWS Rekognition custom label had been complete. The AWS Rekognition custom label uses a transfer learning process in order to properly and efficiently analyse and detect these images. Initially the labels rose, sunflower, tulips, dandelions, magnolias and lily's were uploaded in order to test if the API would recognize these functions. After several trial and error through Lambda code manipulation it had successfully functioned.
So What?

For the project this is a massive achievement as one of the core functionalities has been confirmed to work as planned. Future challenges that remain now reside within proper documentation of the testing used to achieve this, alongside furthering the CSS design of the application within Android Studio and returning the details grabbed by the API with this CSS incorporated. The documentation process will also include the transparency of the application and the data used.
Now What? To address the outstanding challenges, I must properly adhere to the wireframes for the CSS architecture. By properly following the established wireframes for the CSS architecture it will guide the development of the final production of the design of the application interface but also to ensure that consistency is followed within what had been established for the design alongside usability. Following this, I must validate the testing methods I had used in order to ensure they both make sense but also, they are understandable to an outside perspective. This will require a thorough review of the processes recorded, alongside the testing methods available now in order to ensure testing protocols are followed and are effective for an ideal user-friendly experience.

Student Signature	

Month: March

Student Name	Taziwa Mushayabasa
Student Number	20444686
Course	BSHCSD4
Supervisor	William Clifford

What?

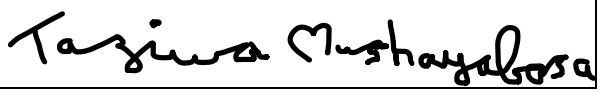
Within this month of development for the application the completion of the main functions of the project has come to near completion. This pertains to the functions concerning the AWS back-end with the flower recognition and the complete pipeline to have this function properly operate. Final amendments of this includes returning this data to the mobile application-client side properly alongside implementing the design UI in order to adhere to a user-friendly environment. In addition to this, further documentation and testing of the application has been done during this phase of the application to further stabilize and validate the functions and facts that currently surround it.

So What?

For the project this is a massive achievement as it means it is entering the final few phases of development before it is ready to be released into a production environment for use for those who may see fit. This is also a big achievement as functions work properly and alongside this the API is works as expected, returning the proper information in conjunction with the image detection software, AWS Rekognition. For both of these to function properly, it confirms that the previous back-end engineering in the AWS cloud services pipeline works successfully also. This includes configuration of the S3 bucket, software code & environmental variables within AWS Lambda to name a few.

Now What?

The application will be entering the final phase of development, polishing these elements, and validating the research done among this in order to confirm it successful for a production environment where users who are interested of the application can utilize it with minimum issue.

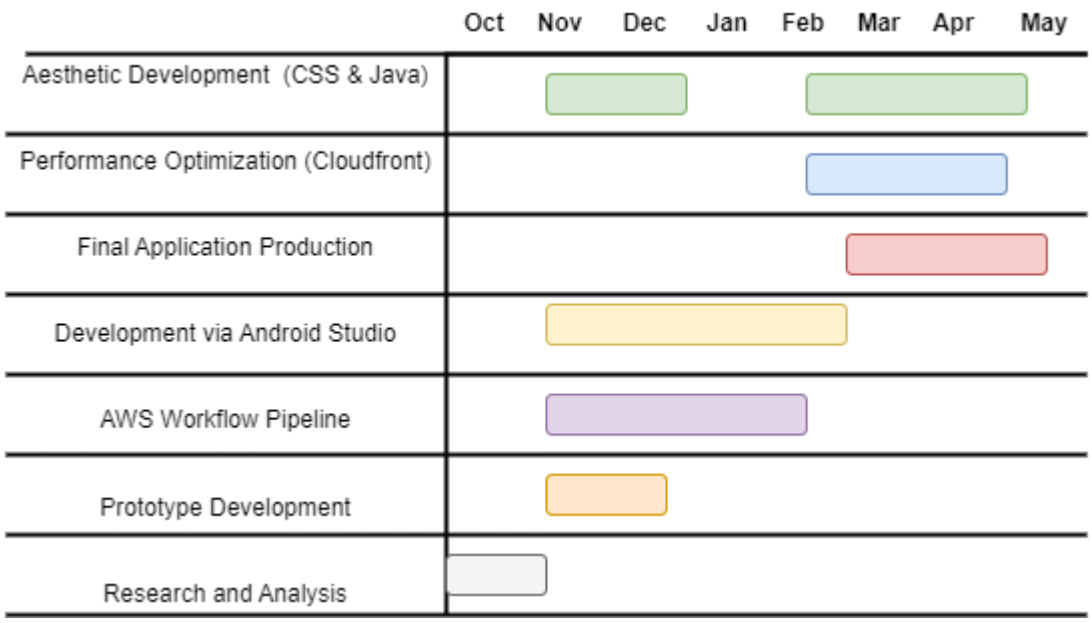
Student Signature	

Month: April

Student Name	Taziwa Mushayabasa
Student Number	20444686
Course	BSHCSD4
Supervisor	William Clifford

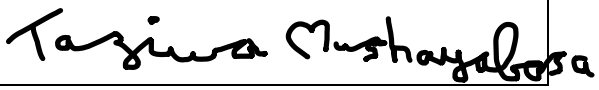
What?

Within this month of project development, the application has reached a status of completion. This coincides with the projected development timeline developed during the initial stages of the project. Expected development of the application at latest would extend into May but thankfully through proper timekeeping and management over the course of the college semesters this has been adverted. This chart was made during the start of the project plan and progression in order to have a timeline to abide to during the applications development cycle. The application possesses the functional requirements it sought after whilst also incorporating an additional feature in the form of a “search” bar.



So What?

This completion of the applications development cycle has allowed time during this month to allow for proper testing of the application through different testing methods such as unit testing, integration testing and usability testing. The completion of the application as a whole has been a relieving experience and is satisfying to witness the both the local environment through the IDE and my own personal mobile phone to be able to work seamlessly with the AWS back-end environment for the applications purpose. This alongside the user interface layout produced and the many layers behind development of them has made this feel like a large

accomplishment overall for my project, and my own skills as a developer who looks to specialise in working with cloud services.	
Now What? With the completion of this application, I can confidently rest and stay assured of my work. This application will be the forefront of what I display within my portfolio in the future when looking for employment.	
Student Signature	

8.2. Other materials used.

To create the flower visual assets for the application a program named Paint.net[22] was used to develop them. These assets were made by me by utilizing layers in order to be able to rollback progress if mistakes occur, or if the design needs slight tweaking without erasing major parts of progress in the process.

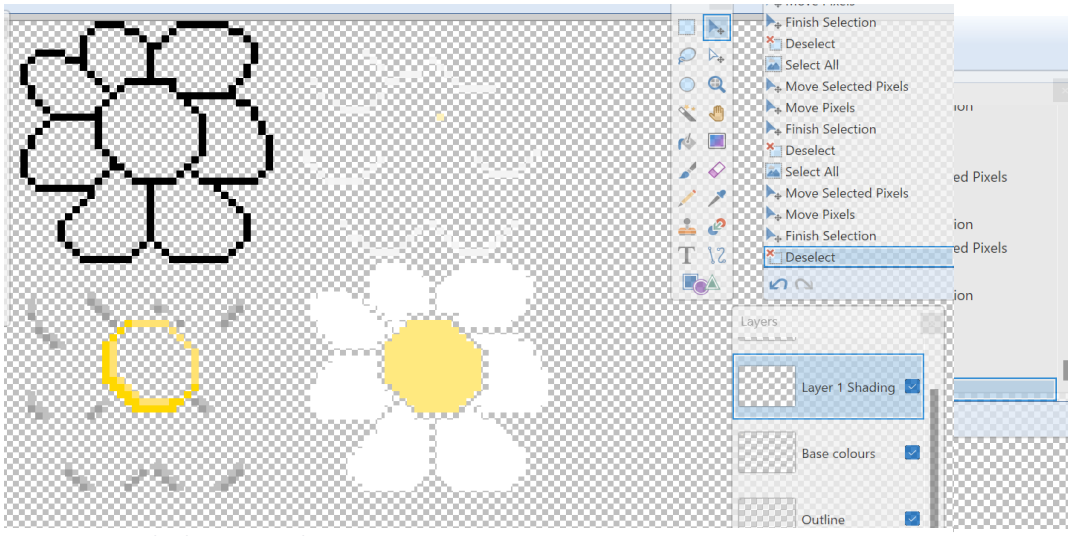


Figure 17: Aesthetic Design Element: Rose

The images shown below show these assets made within the layers, separate, to see the development process of these aesthetic assets.

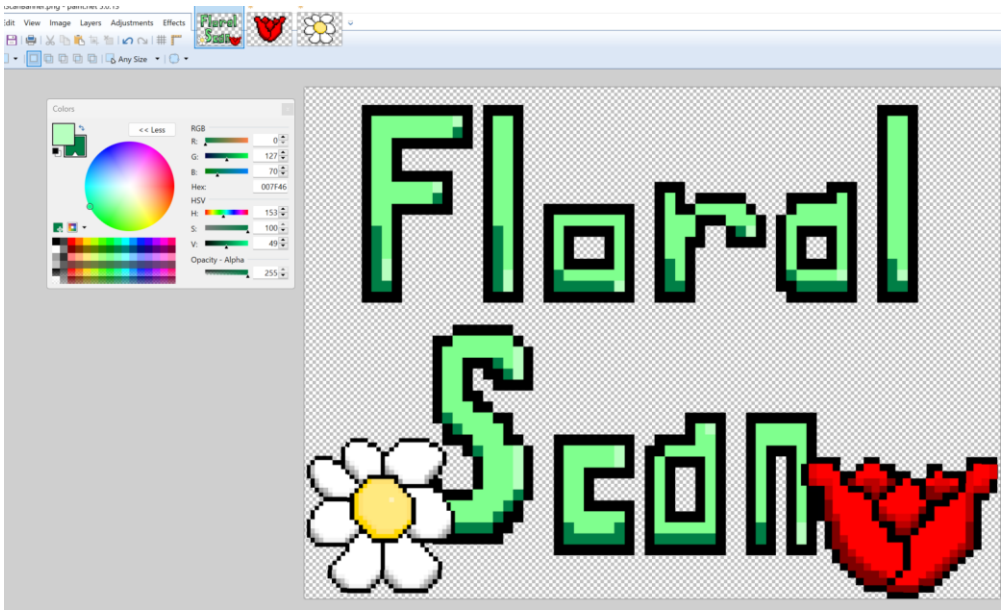


Figure 18: Aesthetic Design Element: Title

For use of the images within the scrollwheel functionality that is present within the AboutUs Page, free-to-use stock images from Pexels [19] [20] [21] were used in order to bring a lively feel to the application. The images selected fit the context of the application, alongside ensuring being free to use for people who may consider implementing them in their applications.