

# Filler README



Filler

<b>Introduction</b>	<b>5</b>
Team Name	5
Proposed Level of Achievement	5
Motivation	5
Aim	5
User Stories	5
Project scope	6
<b>Features</b>	<b>7</b>
Feature 1: Sorting	7
Process	7
Feature 2: Helpful tips	11
Feature 3: History and stats	14
Feature 4: News page	16
Feature 5: Virtual pet	18
Feature 6: Reminders	20
Tech Stack	21
<b>Software Engineering Principles</b>	<b>22</b>
Single Responsibility Principle (SRP)	22
Open-Closed Principle (OCP)	22
Liskov Substitution Principle (LSP)	22
Interface Segregation Principle (ISP)	22
Dependency Inversion Principle (DIP)	23
Keep it Simple, Stupid Principle (KISS)	23
Modularity	23
Version Control	24
<b>Design</b>	<b>25</b>
Pages	25
Navigation Diagram [previous]	28
Navigation Diagram [current]	29
<b>Testing</b>	<b>30</b>
User Testing	30
Timeline	32
<b>Limitations</b>	<b>33</b>
Limited Supabase storage, OpenAI and News API queries	33
Heavy reliance on users for data inputs	34
<b>Challenges</b>	<b>34</b>
Using Expo	34

Incompatibility of packages	35
Designing a user-friendly interface	35
<b>Other Documents</b>	<b>36</b>
Install Instructions	36
Technical proof of concept	38
Project log	38
Feedback form	38
<b>Acknowledgements</b>	<b>39</b>
OpenAI API	39
NewsAPI	39
Flaticon	39
Trash Dataset	39

Doesnt Matter proudly present to you...

# Filler

Want to start living sustainably but don't know how?

## SWE Practices

KISS: prioritise simplicity and ease of use.

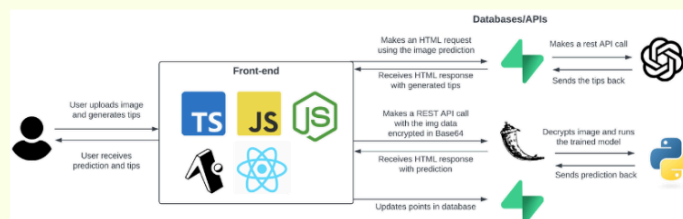
Modularity: components are not dependent on each other, preventing bugs from one component affecting others parts.

Version control: Github branching for version control, issues for bug reporting, and Milestones for tracking progress of features.

## Why Us?

Filler is an app that aims to inculcate sustainable recycling habits by identifying trash, providing recycling tips and gamifying the process to make it fun and achievable!

## How Our App Works



Navigation flow

## Features

### Sorting

Users can take pictures of the trash. Our app then identifies what category the trash belongs to so that users can make the correct choice when recycling.

### History & Stats

View one's recycling history. Provide information on user's last activity.

### Helpful Tips

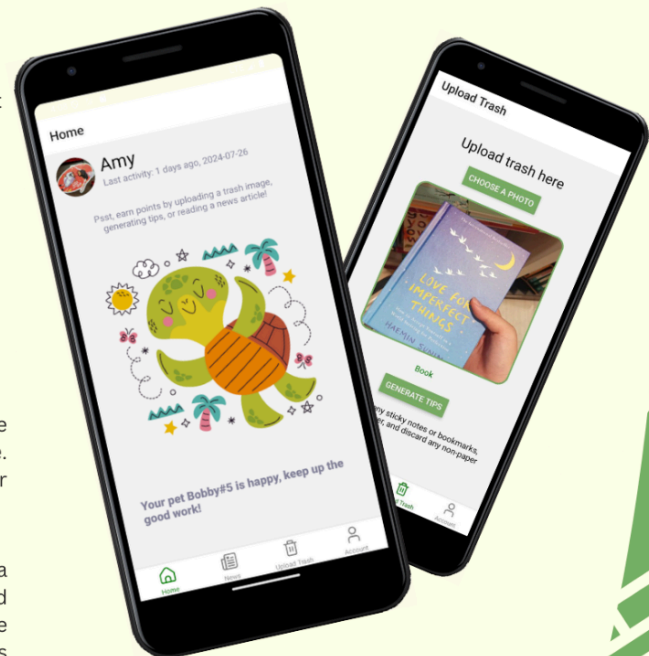
Generate tips for user on how to clean the trash to make it fit for recycling. More often than not, trash that is not properly cleaned can contaminate the whole batch of trash.

### News Updates

Blogs and articles about sustainable living can be obtained at one's fingertips via a curated news page. We believe that an all in one app will make it easier to users to engage in sustainability related topics.

### Virtual Pet

Users can take care of their virtual pet through a point system. Points can be earned when users read an article, scan a trash image, or generate tips. The pet's state will change based on number of points the user earns.



ML Model



We trained our CNN model via TensorFlow. By adjusting the model layers, we improve our model accuracy. Due to large datasets being used, we made use of Google Colab.

Future improvements: recognise wider sets of trash to provide for more specific tip generation.

Data collection: we used publicly available datasets on Github to obtain images grouped based on categories.



Scan the QR code to get on board now!

By He Xinghua & Nguyen Anh Duc

# Introduction

## Team Name

Does(n't) matter

## Proposed Level of Achievement

Apollo 11 (Current level: Gemini)

## Motivation

As residents of the Ridge View Residential College, a college in nature that has a strong emphasis on sustainability, we feel a strong sense of responsibility to play our part in saving the environment. Equipped with the interdisciplinary knowledge of both Computer Science and environmental sustainability, we feel inspired to make use of our knowledge and contribute meaningfully to society by designing an application that everyone can use.

## Aim

We aim to **inculcate good recycling habits** amongst Singaporeans and **educate** users on some tangible steps they can take to live more sustainably. By making the process of recycling **fun** and engaging through our gamification process, we hope that our application can **reach a wider audience** with **greater retention**.

## User Stories

1. As a person who often gets confused about which bin to throw the trash in, I want to be able to recycle correctly.
2. As a person who cares about my individual recycling habits, I want to ensure that the trash I recycle is first cleaned off appropriately such that it does not end up in landfills or spoil the whole batch of recyclable trash.

3. As a person who cares about my society, I want a dedicated platform that contains sufficient information on sustainability and how I can play a part in it so that I have the information at my fingertips.
4. As a person who cares about the earth, I do not want my eco-friendly habits to be a tedious task. I want my pro-environment lifestyle to be fun and interesting so that I can sustain it for a long period of time.

## **Project scope**

Filler is a mobile application designed to promote recycling through image recognition, providing sorting instructions, sustainability tips, and a gamified experience to encourage eco-friendly habits.

Filler aims to transform waste management and recycling behaviors through a user-friendly mobile application that leverages advanced image recognition technology to accurately identify various types of waste materials. Upon identifying the items, the app provides detailed instructions on how to properly clean and sort the trash.

In addition to functional sorting guidance, Filler offers daily sustainability news articles to educate users on broader eco-friendly practices, such as energy conservation, water usage reduction, and choosing sustainable products. The news articles will also keep users interested to find out more about sustainability in general, hopefully instilling in them the willingness to partake in or initiate bigger and more meaningful projects in their community.

The app also includes a unique gamification feature where users can nurture a virtual pet turtle that thrives as users increase their recycling efforts, making the process engaging and rewarding.

This project seeks to engage users consistently, track their progress, and build a culture of reducing environmental impact through practical actions in their daily lives.

# Features

Filler's features are outlined in the following sections, organised by the following tags:

**[Proposed]** - features for Minimum Viable Product (MVP) by Splashdown

**[Current Progress]** - elaboration on current progress of specific feature

## Feature 1: Sorting

### Process

#### Machine Learning

We used Google Collab to first train a small set of trash. Through the Keras workflow, we were able to streamline the process for the otherwise complicated Convolutional Neural Network model that we are using.

#### First model:

We trained the model on a 'plastic' subfolder from the data set by [Nikhil Venkat Kumsetty](#) with 1859 photos in 5 classes of trash.

```
['cigarette butt',  
 'plastic cups',  
 'plastic bottles',  
 'plastic bags',  
 'plastic containers']
```

We first loaded the data from the trash dataset uploaded to Google Drive. Data cleaning was performed to ensure that the image size is not too small (<10kb) and that the file can be opened and read by OpenCV.

We scaled the data so that the RGB values are in the range of 0 to 1, which is in a form the model can recognise.

We split the data into 3 categories: training (~70%), validation (~20%) and testing (~10%).

We then added the layers as shown below:

```

#add the different layers

#first layer is convolution layer, 16 filters: scan and condenses img
#(3x3) blobs
#1: move 1 pixel by pixel
#activation='relu' only preserve +ve inputs and remove -ve inputs
model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(256,256,3)))

#Take max value from 1st layer and return it
model.add(MaxPooling2D(pool_size = 2, strides = 2))

model.add(Conv2D(32, (3,3), 1, activation='relu'))
model.add(MaxPooling2D())

model.add(Conv2D(16, (3,3), 1, activation='relu'))
model.add(MaxPooling2D(2,2))

#flattens to single value
model.add(Flatten())

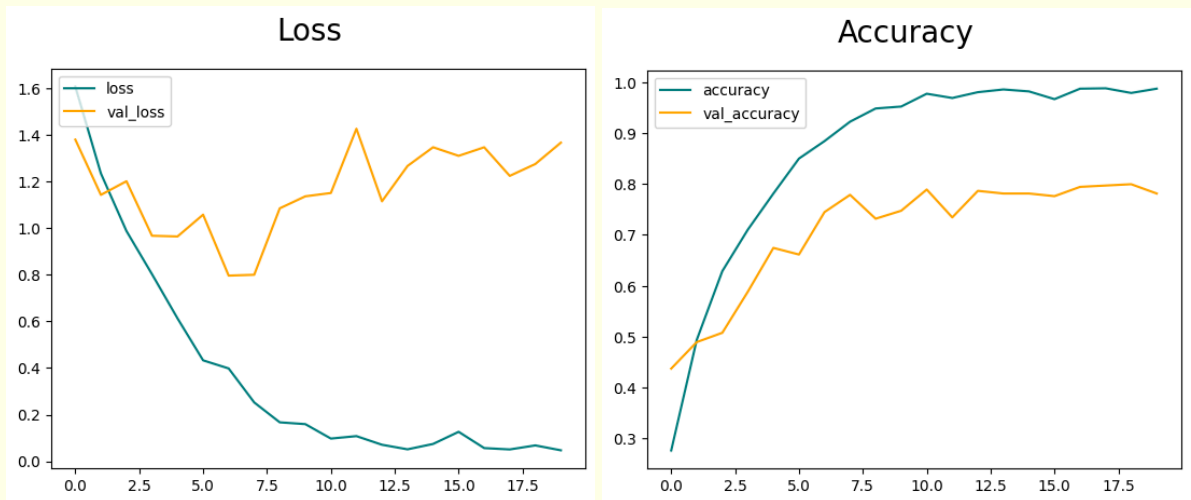
model.add(Dense(256, activation='relu'))
#clasify into the 5 diff categories
#activation='softmax' output format is in probability dist of [0.1, 0.5, 0.4]
#shows the probability of each
model.add(Dense(5, activation='softmax'))

```

The model is then compiled. Instead of producing a binary value, the final layer is a SoftMax layer where the final result is an array containing n probabilities, with n being the number of trash categories and the index with the highest probability being the model's prediction for the most likely category.

We plotted the Losses and Accuracy of our model, which provides us useful feedback.





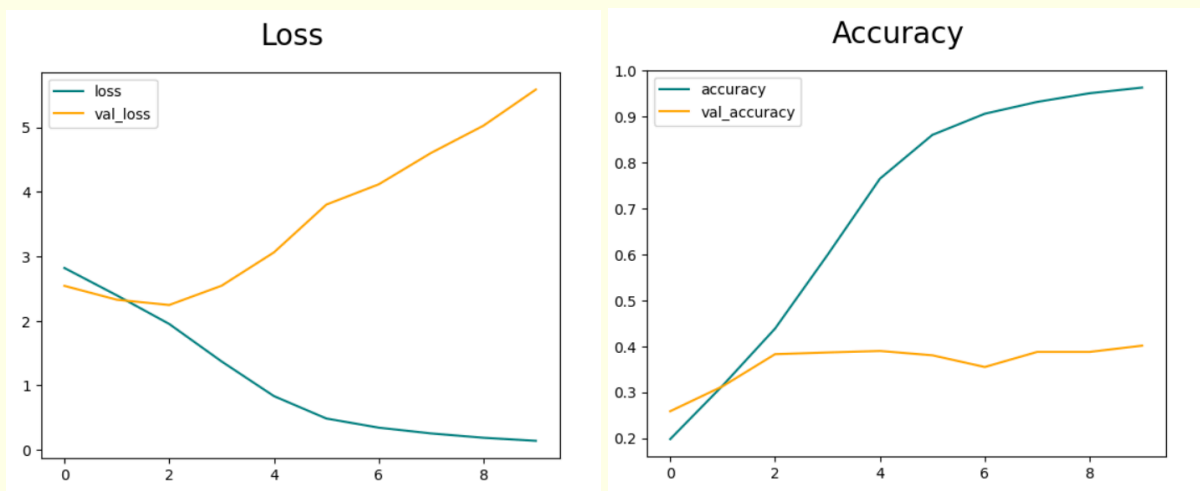
We then saved the trained model.

### **Second model:**

We then upsampled it by using all the data in the dataset with 12075 photos in 25 classes shown below.

```
['cigarette butt',  
'plastic cups',  
'plastic bottles',  
'plastic bags',  
'plastic containers',  
'glass',  
'cardboard',  
'tetra pak',  
'paper',  
'paper cups',  
'news paper',  
'beverage cans',  
'construction scrap',  
'metal objects',  
'metal containers',  
'spray cans',  
'gloves',  
'masks',  
'medicines',  
'syringe',  
'electrical cables',  
'electronic chips',  
'laptops',  
'smartphones',  
'small appliances']
```

However, using the same combination of layers above gives a relatively worse val\_accuracy.



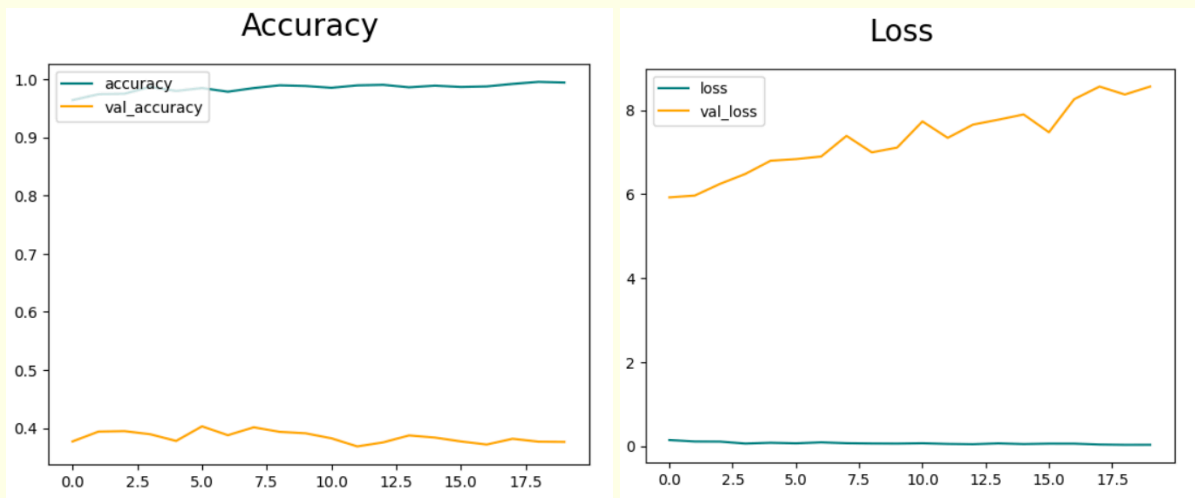
So we tried other ways such as batch normalization, normalizing every batch in between each layer, or adding more layers to the model training.

```
##BatchNormalization
model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(256,256,3)))
for i in range(16):
    model.add(BatchNormalization())
    model.add(Conv2D(4, (3,3), 1, activation='relu', padding='same'))
    model.add(Dropout(0.1))
    model.add(Conv2D(4, (3,3), 1, activation='relu', padding='same'))

model.add(Conv2D(16, (3,3), 1, activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(25, activation='softmax'))

model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(256,256,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3), 1, activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), 1, activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation="relu"))
model.add(Dense(25, activation="softmax"))
```

However, they did not improve the accuracy, even when they were more time consuming to train the model.

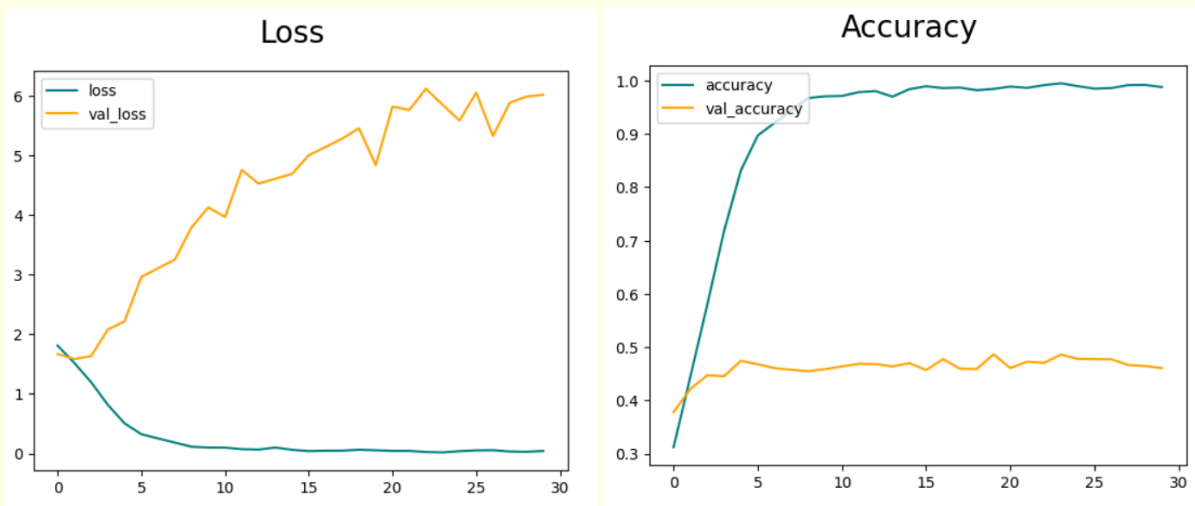


Third model:

We scaled down the categories into 8 with the same data set.

```
['medical waste',  
'plastic waste',  
'glass',  
'e-waste',  
'cardboard',  
'cigarette butt',  
'metal waste',  
'paper waste']
```

This gave a slightly better accuracy of around 0.5 so we settled for it.



## **Backend server**

We used Flask to set up a backend server that is listening to HTTP requests. The server receives a POST request that has the image encoded in base64, which is then converted to readable format, fed into the model, and result generated.

## **Fetch requests**

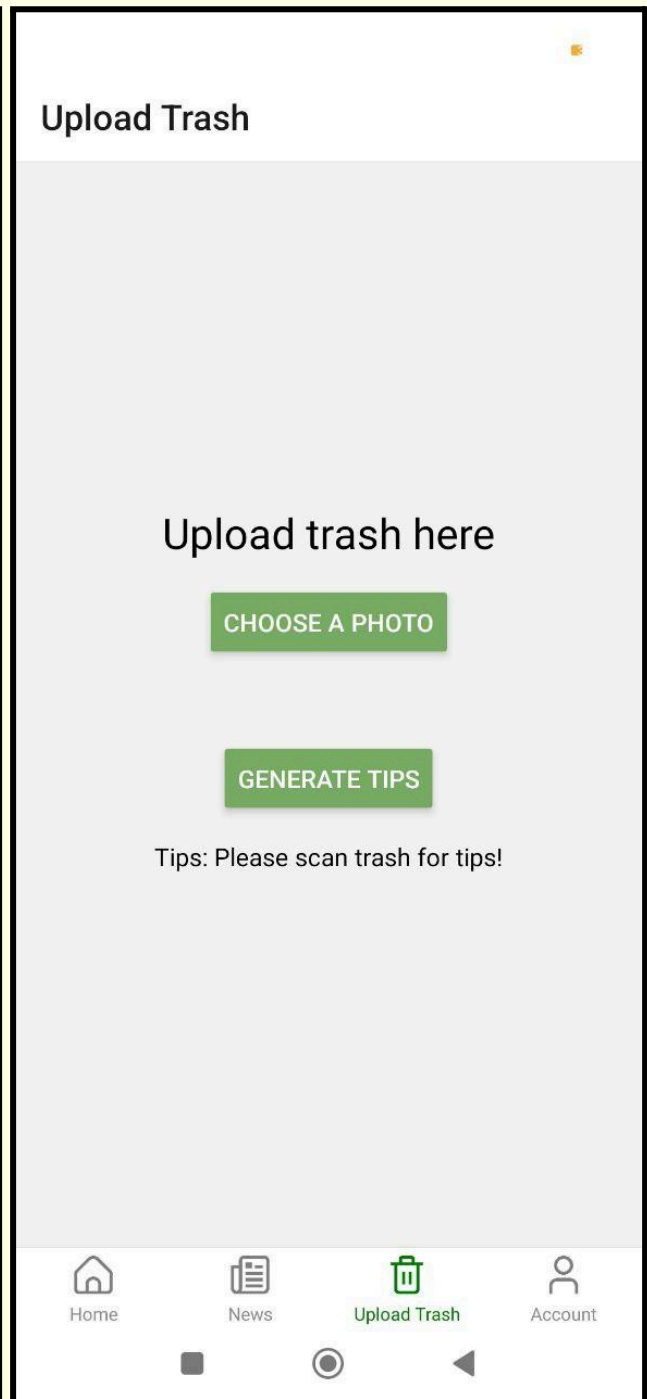
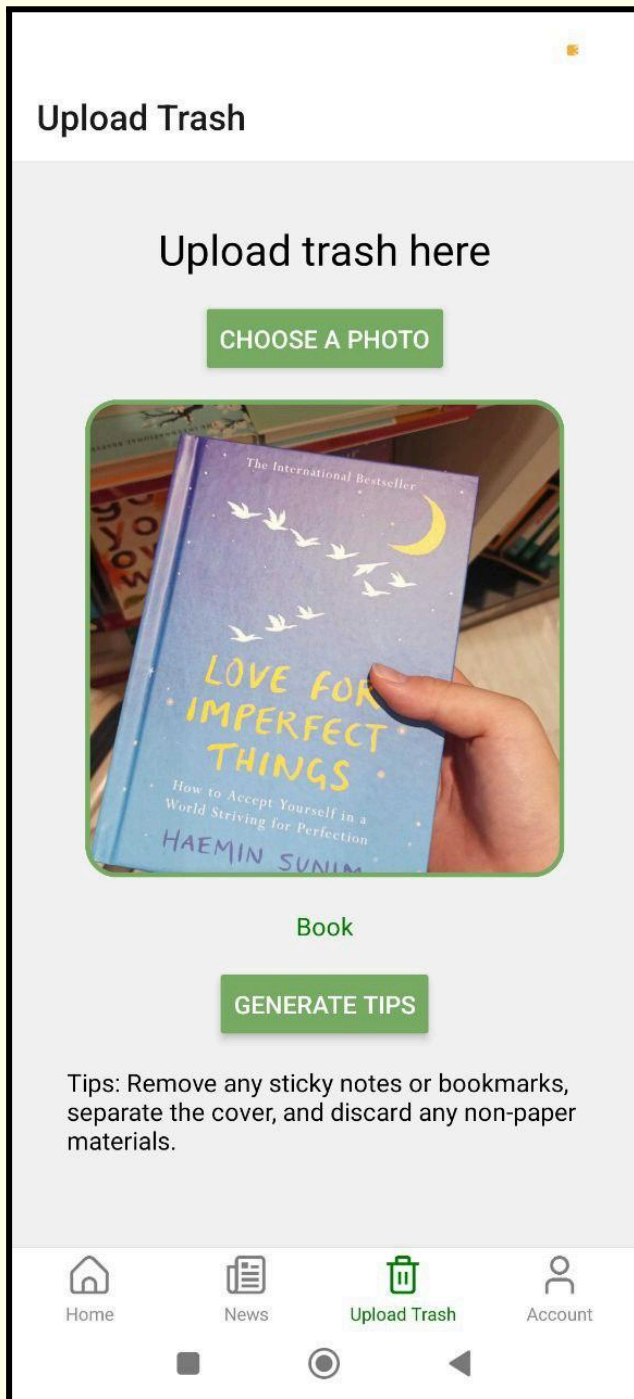
Client sends POST request, and awaits for the result generated from the server side.

### **[Proposed]**

Users can take pictures of the trash and the trash can. Using image recognition, the software will analyse the image, recognise the trash and identify it. Users will also be able to take pictures of trash cans in the future, and our app will inform users which trash can to throw the item into.

### **[Current Progress]**

Users can upload the pictures of the trash for detections of the trash type. Currently only can categorise images into 20 categories of plastic waste accurately. We are still training our model to recognise a wider array of trash. We will be doing this via fine tuning the model, adding more layers to the CNN model we are using as illustrated above.



Left: Click "**Choose a photo**" to select a photo and upload the image

Right: Detection result generated

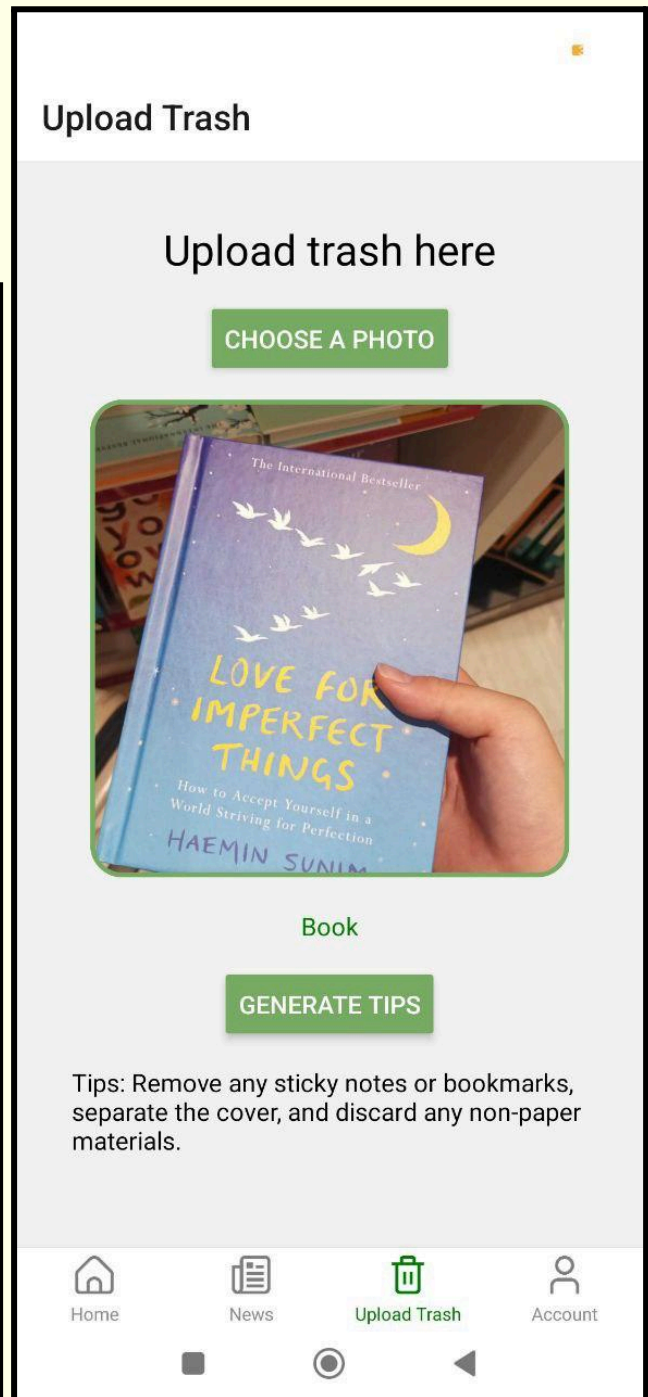
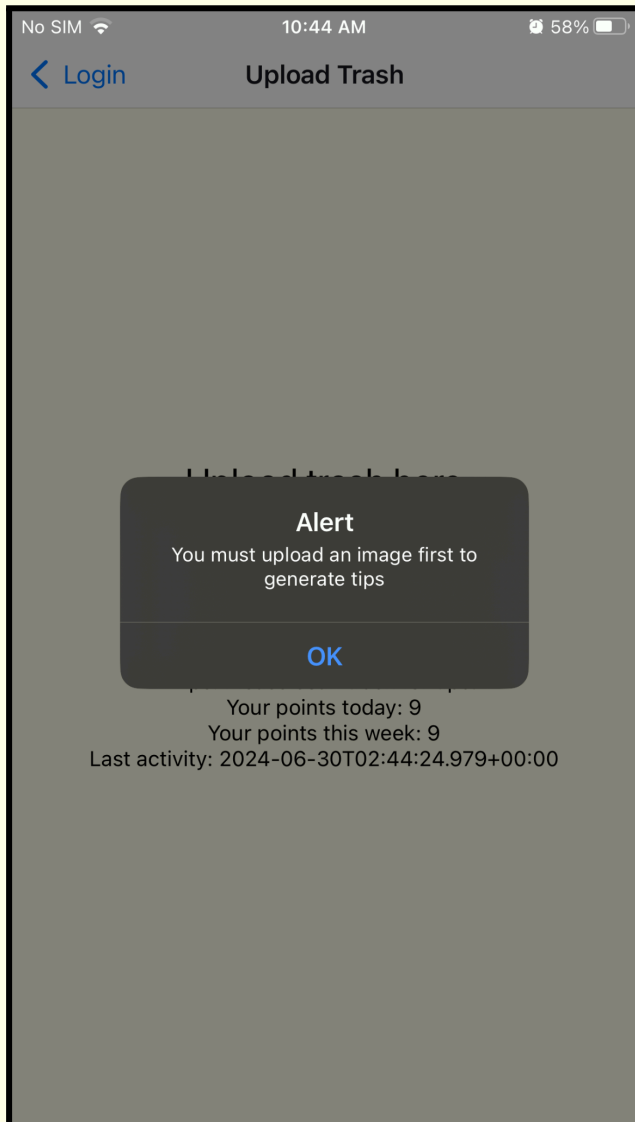
## **Feature 2: Helpful tips**

### **[Proposed]**

After scanning the image, based on the trash category, the app will generate information for the user on what to look out for when recycling the trash: (e.g.) How to clean the trash so that it can be fit for recycling. Our application will also inform users when some items are not recyclable.

### **[Current Progress]**

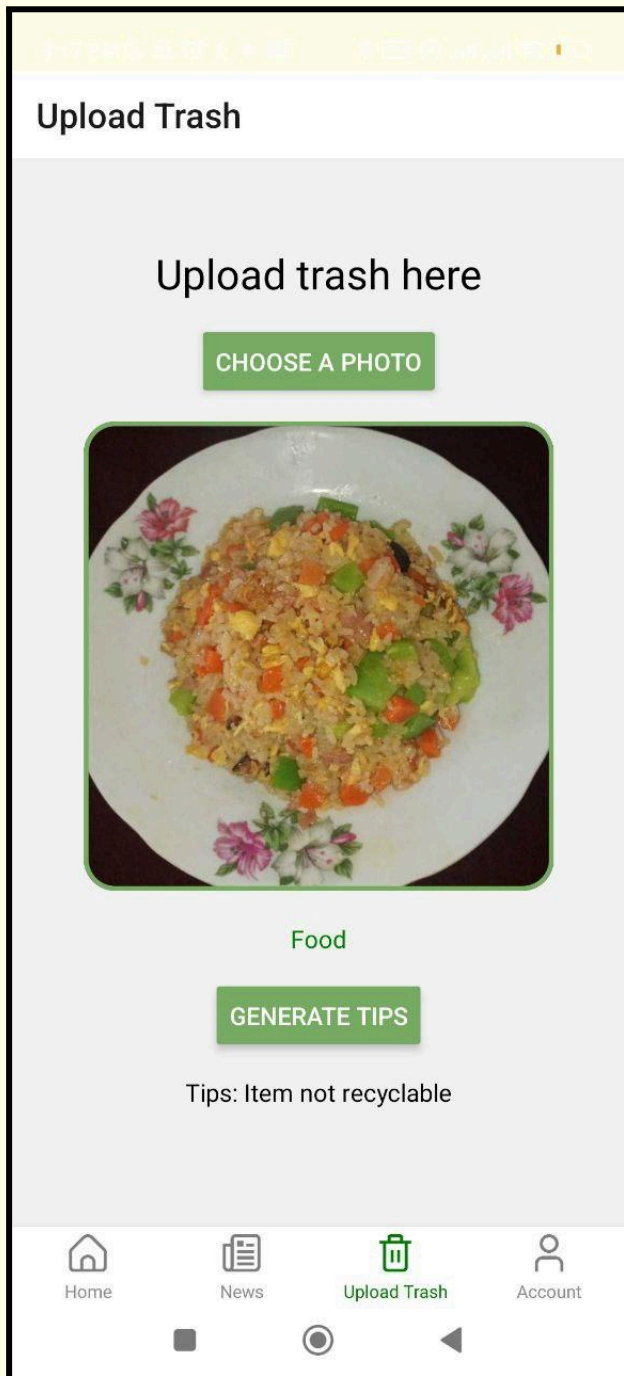
Using OpenAI API, the users are able to key in their trash and will be received with information on whether the trash is recyclable and how to recycle the trash if they are recyclable. If users have not uploaded any trash images, an alert will popup, preventing the user from generating tips.



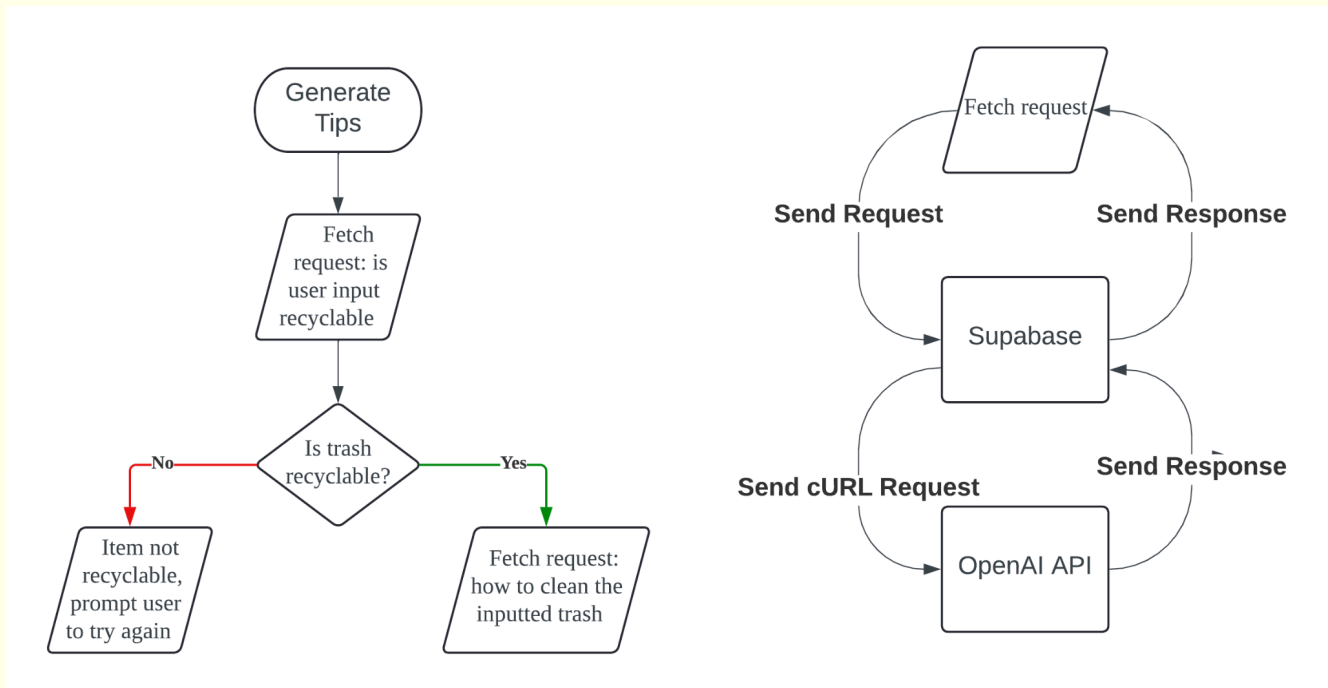
Left: No trash uploaded yet, alert pops up to notify users to upload image first.

Right: Upload Trash screen where users can click on "**Generate tips**" to generate tips after the uploaded trash has been scanned.





Inputted trash cannot be recycled, no tip generated



When pressing the Generate Tips button, the app sends a fetch request to Supabase, a backend server that then sends a cURL request to OpenAI API. The app request is sent via a custom generate\_tips supabase function, in which the function takes in a description parameter. We first send the request using the description "Is {generated image prediction} generally recyclable?" We limit our response to OpenAI to only Yes or No. If No, prompt the user to try again. If Yes, we then send another request using the description "How to clean {generated image prediction} such that it is fit for recycling?". The response is then sent back from OpenAI to Supabase, then back to the application in a JSON object format.

Unfortunately, OpenAI is not very accurate in terms of answering the first question on whether an item is recyclable. User testing revealed that certain obviously non-recyclable organic matter such as food and animals are fit for recycling.

In the future, we are planning to compile our own database of common recyclable items and unrecyclable ones, instead of asking OpenAI. Generally, the tips provided once the item is indeed recyclable is accurate.

## **Feature 3: History and stats**

### **[Proposed]**


User login to save user data so that users can track their recycling history. Users can view one's recycling history. The app will provide statistics and data analysis on the impact of user's recycling habits on the environment.

### **[Current Progress]**

The app has a basic login page. Users can log in and log out. Once logged in, they can update personal details and their profile picture.

Users are able to see their weekly points (reset on Sundays) and daily points (reset at midnight), as well as their last activity. The points are refreshed approximately every 10 seconds with a `setInterval` function that takes in the `updatePoints` function as a parameter.

Welcome to Filler!  
Ready to live more sustainably?




**Email**  
✉ email@address.com

**Password**  
🔒 Password

**Sign in**

Don't have an account yet? [Sign up](#)

**Account**



**UPLOAD**

**Email**  
1@163.com

**Username**  
Bob

**Pet Name**  
Bob#2

Last activity: Today, 2024-07-28  
Daily points: 5  
Weekly points: 5

**Update**

Bob#2 is sorry to see you go... [Sign out](#)

Home News Upload Trash **Account**

Left: login page where users can either Login or Sign up

Right: Account page. Users can click "**Upload**" button to upload a profile picture and "**Update**" to change their username and Pet Name. Click "**Sign Out**" to log out of the account.



Home page where users can see their last activity.

## Feature 4: News page

### [Proposed]

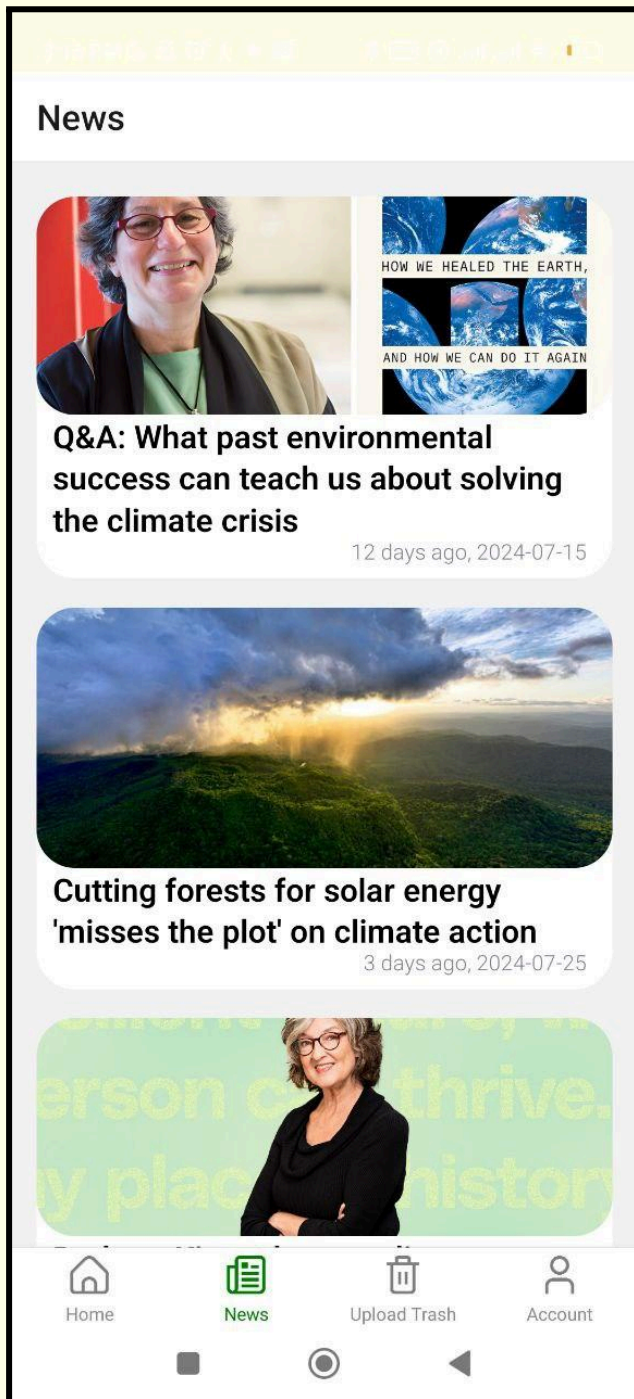
Curate a news page where users can receive tips, blogs or articles on how to live more sustainably and get updates on the latest environment related news. We intend to implement this feature via web-scraping and filtering only environment related news.

### [Current Progress]

Basic news page where users can select sustainability related news and will be redirected to the external news URL. We did this via making a REST API call to supabase, via a custom function, `generate_news`, where supabase will make a cURL request to News API. News API then returns a JSON object where the data contains an array of individual news articles. Only the array is sent back to the application, where we first filter out articles that are removed. Then, we obtain the image url, title, and date to be displayed on the NewsCard component. Each article is rendered as an individual NewsCard in a flatlist inside the NewsScreen component.

Once clicked, the NewsCard component is responsible for updating the user points and redirecting the user to the news article in their preferred browser.

Currently, our search terms are "climate change" and "sustainability", where articles must contain either keywords. However, the search results could be further refined as some of the articles displayed are not relevant. Furthermore, we did not filter for the period, where some articles may be out of date.



Left: News page, clicking on the news will redirect users to the news URL.  
Right: Redirected URL, opened in user's default browser.

## Feature 5: Virtual pet

### **[Proposed]**

Gamify the system by having a virtual pet that one can take care of through earning points via recycling trash or reading articles. We want to make use of the user's recycling history and change the pet's state based on the points gained. Below a certain point, the virtual pet will be sad, while above a certain point, the virtual pet will be happy.

We hope that the user will be motivated to keep their virtual pet happy, thereby encouraging daily environmentally friendly behaviour.

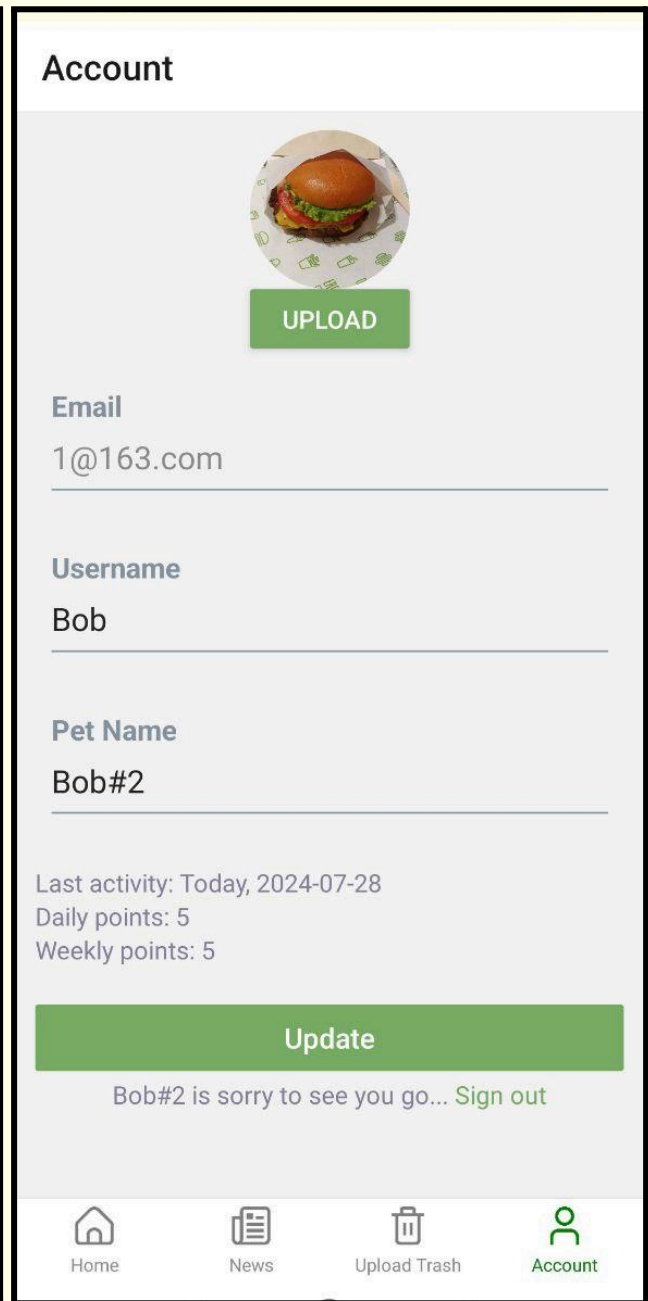
### **[Current Progress]**

The virtual pet can be seen in the homescreen. The point system resets daily and users have to earn >0 points by either reading an article, generating tips, or scanning trash images. Each of these actions increase points by one, thereby fulfilling the daily quota. At 0 point, the default state of the pet will be sad. However, upon earning 1 point, the pet will be happy. Both the virtual pet image and the text below the pet indicating its state will be updated.

The homepage constantly refreshes via the setInterval function, where the parameters is the getProfile function to get point data from Supabase database. The page refreshes every 15000s ms. The image of the virtual pet is randomised. There are 5 images to be selected and the virtualPetState function is called and returns which image to display.

In the future, we want to animate the virtual pet, and give it other interactive elements such as pressing the image will display speech bubbles. We also wish to include other point systems such as reading news articles for hunger and scanning images for happiness. There should also be a status bar to show the virtual pet's state.





Left: Virtual pet in home page where users can see the current state of the pet in the text below the pet.

Right: Account screen where users can see how many points they earn that week and day.

## **Feature 6: Reminders**

### **[Proposed]**

Send notifications for users if they have not earned any points today. The reminder will be sent at 8 am daily and resent at 8 pm if there are no activities from the user during this time.

### **[Current Progress]**

As of now, we have yet to implement this additional feature. We will be implementing this after Milestone 3. We are still determining the best time interval to send a notification to users.

# Tech Stack

## **Frontend**

React Native, Expo, Javascript, Typescript

## **Backend**

Node.js, Supabase, Flask, Python

## **Database**

Supabase

## **Image Recognition**

TensorFlow, Google Collab

## **Authentication**

Supabase Authentication

## **Game Character**

React Native, Expo

## **Version Control**

Git with GitHub

# Software Engineering Principles

## Single Responsibility Principle (SRP)

*If a class has only one responsibility, it needs to change only when there is a change to that responsibility.*

The Turtle module, which is part of the Virtual Pet Screen component has only one responsibility and is only used in that screen. Thus, it is only responsible for one thing and will not be affected by other modules in the same screen. Similarly, the NewsCard component is used for fetching data from NewsAPI and rendering them has only this responsibility. Thus, following this principle allows for easier debugging as bugs that arise from the NewsScreen can be traced back to the NewsCard.

## Open-Closed Principle (OCP)

*A module should be open for extension but closed for modification. That is, modules should be written so that they can be extended, without requiring them to be modified.*

## Liskov Substitution Principle (LSP)

*Derived classes must be substitutable for their base classes.*

Following what we have learnt in CS2030s, we applied the LSP in this project. All children classes can be substituted by their parent class, allowing for backwards compatibility. In case there are changes to the current codebase, the substitutable functions ensure that code does not break.

## Interface Segregation Principle (ISP)

*No client should be forced to depend on methods it does not use.*

While cleaning up the project, we have identified many occurrences where there are cyclical components where both classes depend on each other, when only one class needs to depend on the other. When implementing the bottom tab navigation, we ran into the issue of the Tab bar rendering the Homescreen\_login yet the Homescreen\_login

imports the component Tab bar. We violated this principle and this resulted in warnings from the compiler and our application crashing.

## **Dependency Inversion Principle (DIP)**

*High-level modules should not depend on low-level modules. Both should depend on abstractions.*

*Abstractions should not depend on details. Details should depend on abstractions.*

With this principle, any change to a class will have lesser impact on other classes that are related. This allows for greater flexibility when adding new classes or changing existing ones.

## **Keep it Simple, Stupid Principle (KISS)**

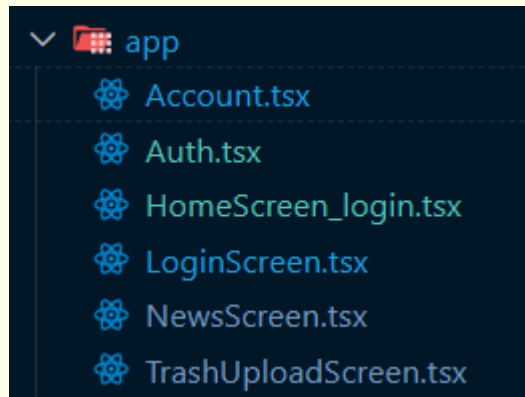
*Prioritise readability and simplicity over complexity.*

All of the features are very intuitive and easy to understand. We design our application around the existing mainstream application so that users do not experience a drastic shift when using ours. In the future, we will prioritise using layouts similar to mainstream apps, with perhaps a user guide to make our app easier for onboarding and longevity. We use the bottom tab where one can head to the Account page, the News page, and the Upload Trash page.

## **Modularity**

*Breaking the problem down into individual smaller components.*

All of the components are not dependent on each other, this is to prevent bugs from one part of the component affecting all other parts. In our design, every single screen is its separate component, with specific functions created for the particular screen. This ensures that components of individual screens can render properly. Each of our screens has a different component.



The App.js file is our application's entry point, where it renders the LoginScreen. The LoginScreen then, will render the HomeScreen\_login if there are existing sessions and Auth screen if the user has not logged in previously. From the HomeScreen\_login, users can navigate to other pages via the bottom tab: Account, NewsScreen, and TrashUploadScreen.

## Version Control

### Github

- Branching

Each of us work on different branches, and never on the same branch. We split the branches into 3 categories, each responsible for different features of the app: detection, testing-branch and unit-testing. The detection for implementing and testing the tensorflow model. Testing-branch for all other main features and UI design. Test/unit-test for performing system testing.

Active branches					
Branch	Updated	Check status	Behind	Ahead	Pull request
test-branch	yesterday	0 / 1	0	1	...
test	5 days ago	0 / 1	1	0	...
unit-test	last week	0 / 1	5	1	...
detection	last week	0 / 1	20	3	...

- Github milestones

We set milestones every 2 weeks, ranking each issue based on priority with top, medium and low priority. We will then work on the ones with top priority, aiming to minimally close the top priority issues by the 2 week deadline.

<div> <div>3 Open</div> <div>0 Closed</div> </div> <div>Sort</div>	
<div>DM_M3</div> <div> <div>Due by July 14, 2024</div> <div>Last updated 1 day ago</div> </div> <div> <div>Refining virtual pet</div> <div>Make the UI prettier: News Page, Bottom Navigation Tabs</div> </div>	<div> <div>16% complete</div> <div>5 open</div> <div>1 closed</div> </div> <div> <div>Edit</div> <div>Close</div> <div>Delete</div> </div>
<div>DM_M1</div> <div> <div>Past due by 19 days</div> <div>Last updated 1 day ago</div> </div> <div>Aim is to finish history &amp; stats feature, and get the image detection to work</div>	<div> <div>100% complete</div> <div>0 open</div> <div>2 closed</div> </div> <div> <div>Edit</div> <div>Close</div> <div>Delete</div> </div>
<div>DM_M2</div> <div> <div>Past due by 12 days</div> <div>Last updated 9 days ago</div> </div> <div>Aim to implement news page and improve UI.</div>	<div> <div>60% complete</div> <div>2 open</div> <div>3 closed</div> </div> <div> <div>Edit</div> <div>Close</div> <div>Delete</div> </div>

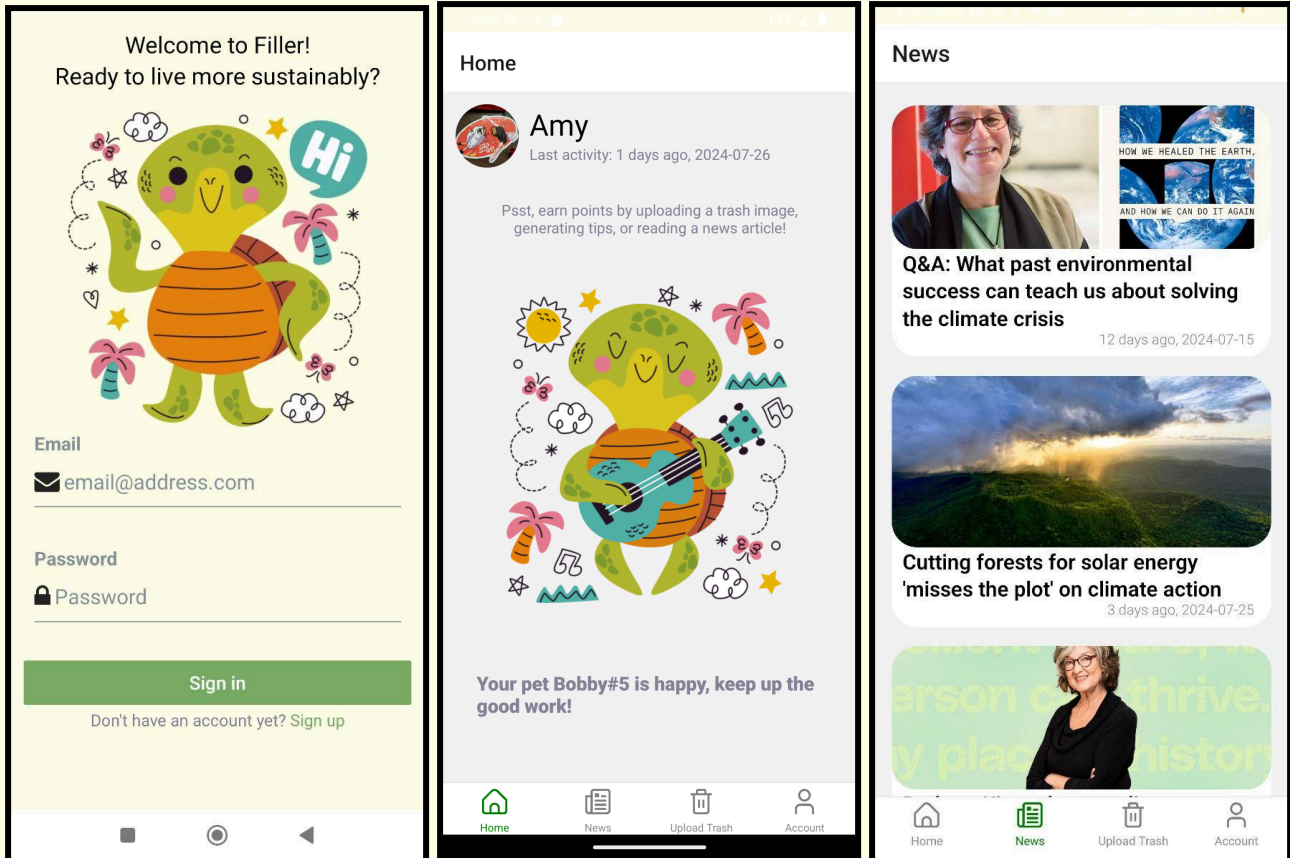
## - Github issues

Under the milestones, we have identified feature improvements and new features to be implemented. We have also identified bugs along the way and use tags to label them. All of the issues have their respective assignees, where certain tasks are allocated to those who are more proficient with the technologies involved. Duc often handles the ML component while Xinghua handles Front-end and Backend.

# Design

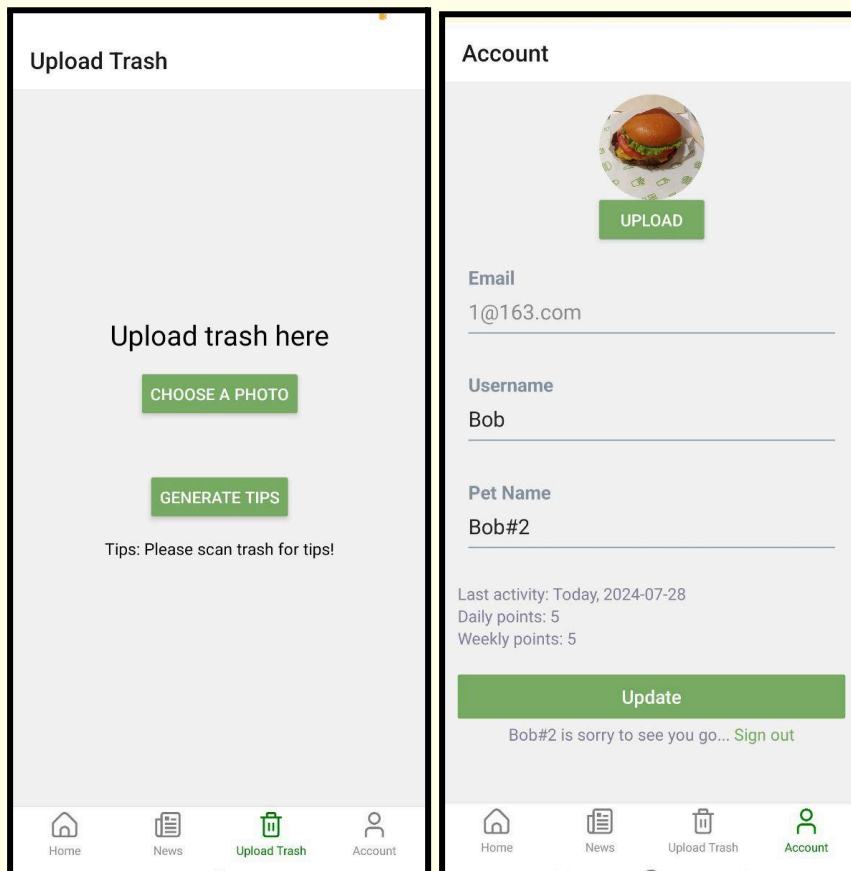
## Pages

There are 5 Screens in total, the Login Page, Home Page, News Page, Upload Trash Page, and the Account Page.



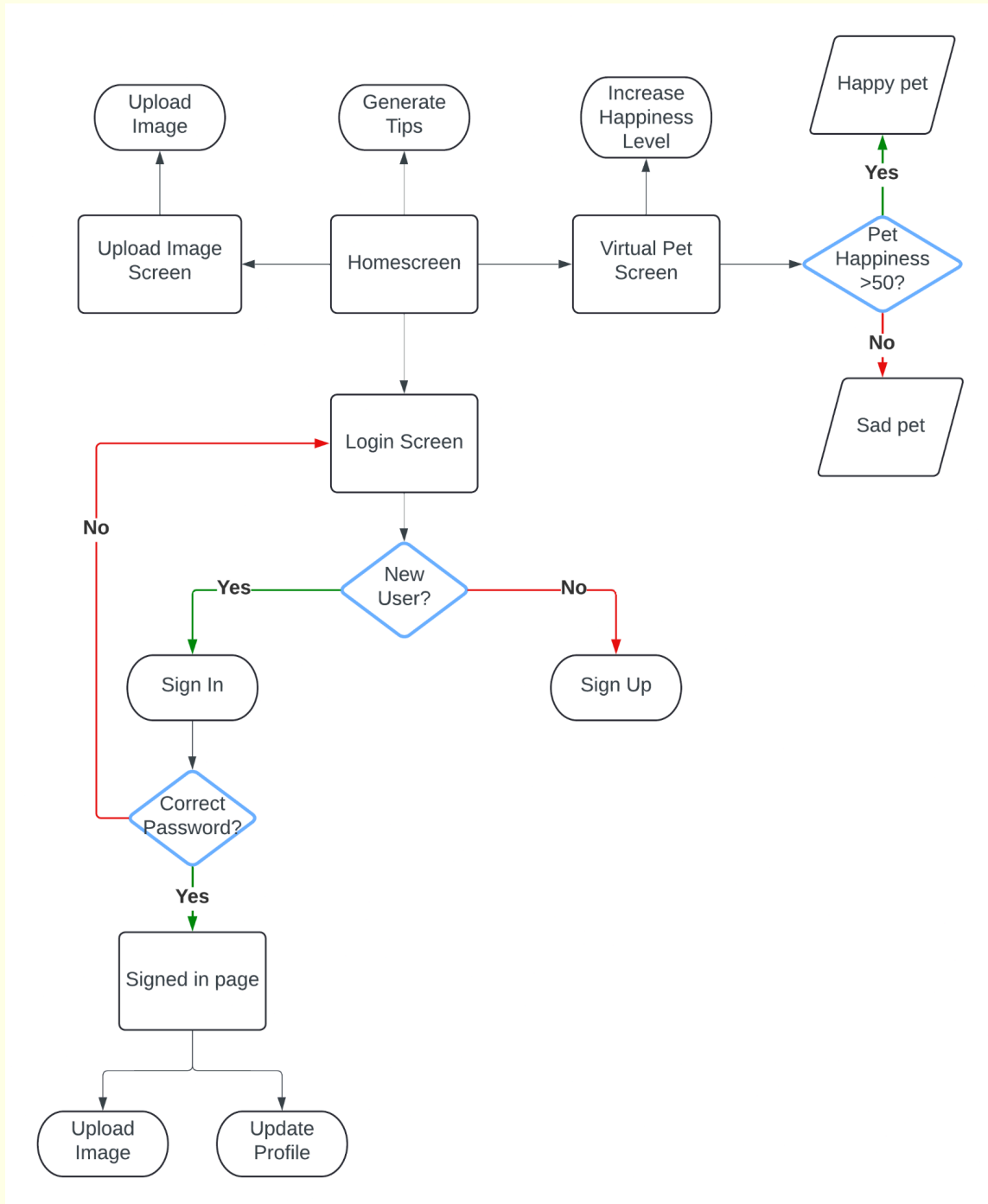
From left to right: Login Page, Home Page, News Page





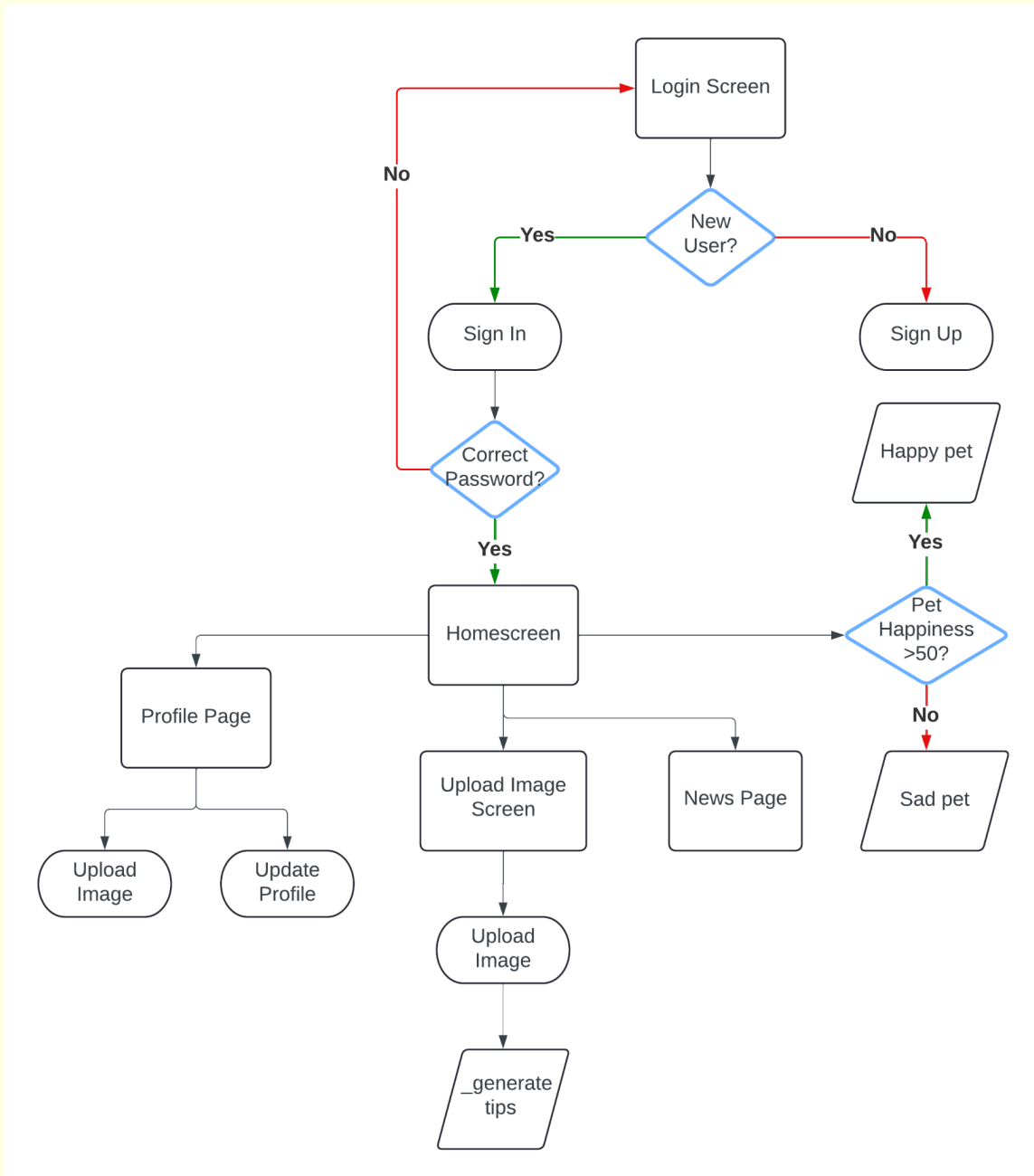
From Left to Right: Upload Trash Page, Account Page

## Navigation Diagram [previous]



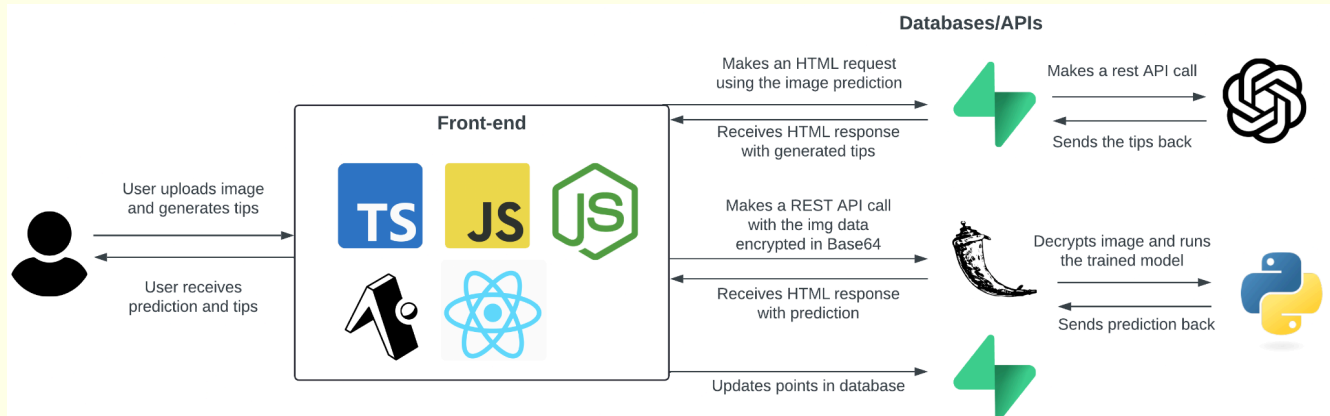
Our previous iteration of the user navigation diagram was implemented in Milestone 2. Clearly, there are some very awkward layouts such as the Login Screen being on a separate screen and not leading anywhere meaningful. The virtual pet's happiness is also not based on the user's points but on a completely unrelated point system. Because previously, we have yet to implement the point system and figured out how to sync data across the screens to the data in the database.

## Navigation Diagram [current]



Our current iteration of the user navigation diagram is more intuitive, implemented in milestone 3. The Login page rightfully leads to a homepage, where one can navigate to News Page, Upload Trash page, and Account page.

## How Our Application Works





# Testing


## Unit testing


### Test Setup and Execution

- Test Files: Multiple test files on individual components such as ``GenerateTips.test.js``, ``HomeScreen.test.js``, ``NewsCard.test.js``, and ``Turtle.test.js`` were created, following our SWE principle. Each file contains test cases specific to the component it tests.

 `GenerateTips.test.js`

 `HomeScreen.test.js`

 `NewsCard.test.js`

 `Turtle.test.js`

- Test Framework: The project uses Jest as the testing framework. This can be seen from the use of functions like ``it``, ``render``, and assertions provided by Jest.
- Test Scenarios: The tests cover various aspects like rendering components with provided data, handling user interactions, and checking the functionality of methods within components. For instance, tests check if the component renders correctly, handles button presses, and interacts with the UI as expected.

✔ Test run at 7/28/2024, 8:04:21 PM

✔ should return tips

✔ renders correctly

✔ handles press event correctly

✔ renders correctly

✔ handles input and button press

### Test Results and Coverage

- Results: The test run was successful, with all test suites and tests passing. The results show that there were 4 test suites, each corresponding to a different component, and a total of 5 tests were run, all of which passed.

```

Test Suites: 4 passed, 4 total
Tests:      5 passed, 5 total
Snapshots:  1 passed, 1 total
Time:       17.769 s
Ran all test suites related to changed files.

```

- Snapshot Testing: There is evidence of snapshot testing, indicated by the passing snapshot, which helps in ensuring that the UI does not change unexpectedly.

- Code Coverage: The code coverage report provides a detailed breakdown of the coverage metrics, including statements, branches, functions, and lines. For example, `HomeScreen.js` shows 70% statement coverage and 66.66% line coverage. The coverage report helps identify which parts of the codebase are well-tested and which need additional tests.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	36.98	25	15.78	38.57	
Filler	0	100	0	0	
App.js	0	100	0	0	15-18
Filler/__mock__/@react-native-async-storage	0	100	0	0	
async-storage.js	0	100	0	0	5-7
Filler/app	37.83	25	22.22	37.83	
GameScreen.js	0	100	0	0	6
GenerateTipsScreen.js	0	0	0	0	6-46
HomeScreen.js	70	50	33.33	70	28-31,38-41
Filler/components	53.33	50	14.28	66.66	
Turtle.js	53.33	50	14.28	66.66	12-16,20
Filler/components/utills	7.69	0	0	7.69	
GenerateTips.js	7.69	0	0	7.69	5-24
Filler/constants	100	100	100	100	
NewsCard.style.js	100	100	100	100	
theme.js	100	100	100	100	

## Uncovered Areas and Future Work

- Uncovered Lines: The coverage report indicates specific lines in the code that were not covered by the tests. For example, `Turtle.js` has some uncovered lines, as indicated by the line numbers provided in the coverage report.
- Future Improvements: We will focus on increasing coverage, particularly in areas with low percentages. Writing additional tests for uncovered lines and branches can help ensure the robustness and reliability of the application.

Overall, the unit testing process demonstrates a comprehensive approach to validating the functionality and stability of the application. The use of Jest, coupled with the generation of detailed coverage reports, ensures that the application components work as expected and allows for the identification of areas needing further testing.

## User Testing

Size: 5 to 10 people

Observations:

We found out that through user testing, our application is not intuitive to use. Thus, we added labels on buttons and mimicked existing application's design. Furthermore, users reported being unable to click some of the buttons as it went out of the screens. Thus, we had to think of ways to ensure that all components are rendered correctly on all devices.

The bugs we found are the following:

1. Users are unable to Sign Up as the "**Sign Up**" button overflows to the bottom of the screen. Thus, they are stuck at the Login page and unable to proceed. We rectified the problem by shrinking the layout and obtaining another round of user feedback.
2. User data is not updated across screens unless they sign out then sign back in. This occurs because the screens are only rendered once and the data is never updated since. We rectified this by having a setInterval function to fetch the latest data from the backend. While this process is not smooth, as we do not want the application to crash from refreshing too frequently, the lag time is approximately 10 seconds between each refresh.
3. Profile in the Home Page does not get updated at all. We discovered that the parameters have typos. Thus, we rectified the problems by fixing the typos.
4. Virtual pet screen causes application to crash. This issue only occurs on android, where we implemented a conditional statement within the <Image> Prop. Once we remove the image, the virtual pet works again. However, that was a temporary fix. To rectify this, we put the conditional statement out of the Image object.
5. Generated tips have low accuracy and sometimes give nonsensical answers. Unfortunately, this is an issue that we are still working on. Users reported that there are inconsistencies with the same image, as it is sometimes identified correctly and sometimes not.
6. Generated tips lags at times. We are unfortunately unable to have a satisfactory fix to this bug as we are limited by the OpenAI API's 3 requests/min restriction.

# Timeline

1. Milestone 1 - Technical proof of concept (i.e., a minimal working system with both the frontend and the backend integrated for a very simple feature)
  - a. Feature Login System done to level that user can register, login and logout.
  - b. Feature photo recognition done to level that system can recognise a limited set of items (2-3 types of trash)
  - c. Feature Frontend done to level that basic screen can allow log in, image capture and displaying results
  - d. Feature Backend done to level to store information of the account and image recognition
  - e. Minimum UI/UX
2. Milestone 2 - Prototype (i.e., a working system with the core features)
  - a. Feature Login System done to level that is enhanced for improved security and error handling.
  - b. Feature photo recognition done to level that system can recognise a broader set of items (10-15 types of trash)
  - c. Feature Instruction is added as a pop up screen after photo recognition to suggest tips on what to do with the trash.
  - d. Feature gamification done to the level that introduce the pet turtle character and basic interaction with user
  - e. Feature Frontend done to level that improve UI/UX and integration with the new game character
  - f. Feature Backend done to level to improve and integrate with the new feature
  - g. User testing
3. Milestone 3 - Extended system (i.e., a working system with both the core + extension features)
  - a. Feature Login System and Fronted done to a level that is complete with presentable, user-friendly UI/UX and gamification character.
  - b. Feature photo recognition done to a level that is high accuracy with a wide range of recognisable items.
  - c. Feature Instruction is improved with more detailed instructions.
  - d. Feature News page is added
  - e. User testing



## **Limitations**

### **Limited Supabase storage, OpenAI and News API queries**

Supabase has a limit of 2 active projects and the free tier automatically pauses any projects that are inactive for 1 week. Furthermore, it only allows 50000 monthly users, which is not ideal for turning this project into a large scale one.

We used OpenAI API's cheapest model at the time of the start of development, GPT-3.5 Turbo, which costs \$3.00 / 1M input tokens. Input tokens are calculated based on length of input while output tokens are calculated based on length of response. Thus, we have to be very careful to prompt it in the most efficient way possible, with the least amount of characters and limit its response length. While the most advanced model, GPT-4o, which has been proven to drastically improve response accuracy, costs \$5.00 / 1M input tokens, the cost can increase significantly if the user base grows. Furthermore, using OpenAI's free tier also limits the number of requests sent per minute to only 3. This has caused some problems where users will need to spam the generate tips button as it appears to not be functioning.

The News API also has limitations on how many requests can be made. Only 100 requests can be made a day, and no extra requests are allowed once the quota is used up. This meant that articles may not even load should there be a large user base. A workaround for this is to fetch all the articles that day and store them in the database for all users to fetch, where only 1 fetch request to News API is needed daily. Another limitation is that the news articles have a 24 hour delay. Hence, users will not be able to get real-time news headlines.

### **Heavy reliance on users for data inputs**

As a tracker app, there is reliance on users for data inputs. The points can be easily obtained by clicking on the news article and clicking away or scanning images without actually recycling the item. This defeats the gamification system as users are not actually cultivating good habits but simply pressing buttons to earn points.

A potential solution to this would be to allow for video image detection where users must scan the entire process of disposing the trash.

# Challenges

## Using Expo

While Expo provides great benefits such as obscuring the React Native complexities and the ability to write code that can be used for both android and ios, there are numerous challenges that arose.

Firstly, due to package incompatibility, building Expo applications was extremely difficult. Not only does the build process take very long, during the build process, some of the files can get overwritten. Furthermore, free Expo users are limited to 30 builds monthly, and the builds can take even longer as they are put in the free tier queue. Should there be a compilation error, it can take up to 20 minutes for the process to fail and display the error message. At times, one has to read through hundreds of lines of messages to figure out what went wrong. On one occasion, we had to remove every component and add them back one by one to figure out where the bug occurs. This process is extremely time consuming and inefficient.

Secondly, while Expo allows for both android and ios application development, we ended up only with an android .apk file. This is because the build feature only allows users to publish android applications if they are on windows. Thus, while our application should also be available for ios based on local testing, we are ultimately unable to publish an ios version.

Lastly, there was difficulty passing data across screens. When we tried to pass data using route parameters by passing extra objects via the navigation.navigate function, the application failed to build. We then tried other workarounds such as AsyncStorage, localStorage, and Contexts. None of which worked smoothly because it either fails to compile or the data simply does not get updated. In the end, we still have to learn the intricacies of React Native to find the suitable packages to use for our application.

In the future, we are thinking of migrating away from Expo and developing the application only on Android Studio to avoid such issues.

## Incompatibility of packages

One of the main challenges faced was the incompatibility of packages with the Expo workspace. Most of the Node.js packages are either out of sync with expo, or are just incompatible. There were a lot of issues regarding dependencies especially when we were trying to set up the TensorflowJS. There were many unresolved dependency version conflicts that rendered TensorflowJS deprecated and unusable. We tried various other solutions such as Tensorflow Lite, which was supposed to be compatible with mobile devices. We tried loading an existing model, yet that did not work. We keep facing similar issues. We eventually had to switch to training the ML model on Google Collab and creating a backend Flask server that runs the ML model using Python. Similarly, there are packages that we had to find alternatives to because while they are compatible with React Native, it will not work with Expo.

## **Designing a user-friendly interface**

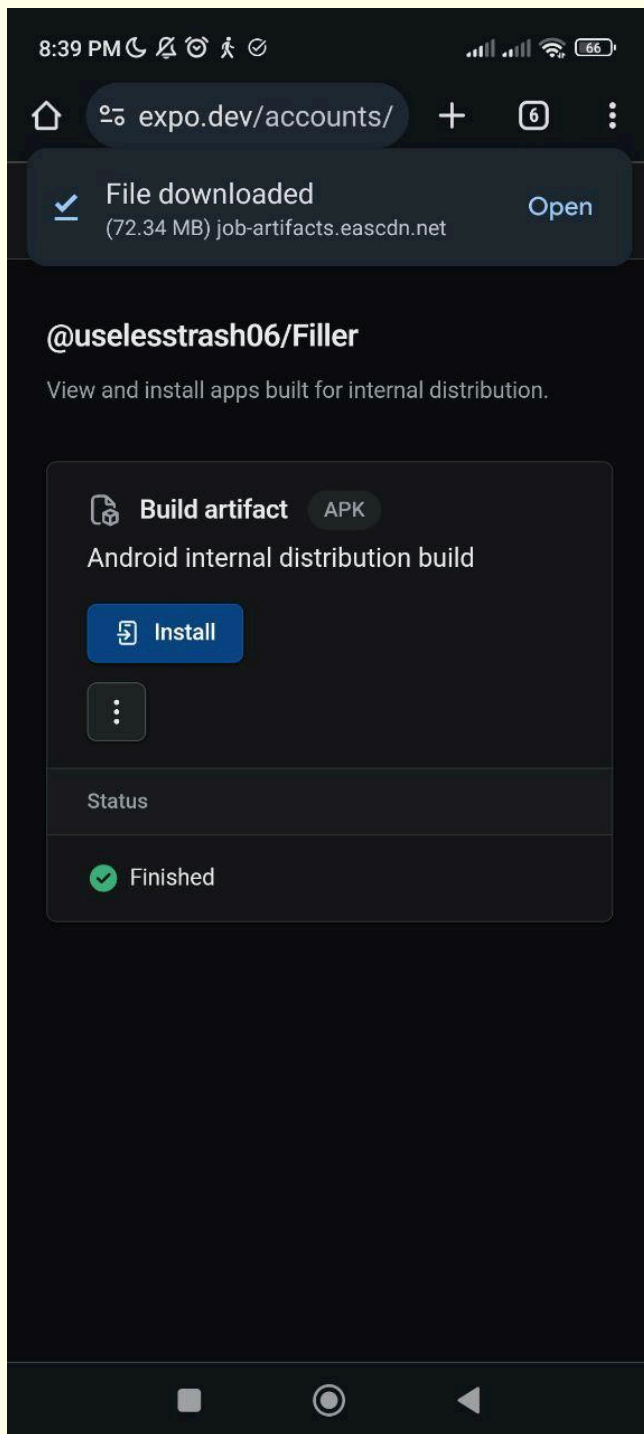
One of the issues we faced when we released the application is that users reported not knowing what to do with the application. Instructions such as how to earn points and what each button does is unclear. Thus, we had to find a workaround where we applied the KISS principle to keep our UI as simple and as intuitive as possible, mimicking existing mainstream applications such as Instagram and DuoLingo. Via multiple rounds of user feedback, we eventually settled with our current UI design.

# Other Documents

## Installation Instructions

### # For Android Phone

1. Click the [link](#)
2. Click "**install**"
3. Click on "**yes**" or "**download anyways**" for all the warnings in the popups
4. In downloads, find the file you just installed and open the file to start the installation process of the .apk file



## # For Android Emulator

1. Head to your preferred browser and paste this [link](#)
2. Follow the installation step as per the Android Phone

## # Alternative installation

1. Download this .apk file at this [link](#)

## Project log

 [Orbital\\_Logs](#)

## Feedback form

<https://forms.gle/Hjr3kzRNXBbZ4Fuh8>

# Acknowledgements

## OpenAI API

Handles the generate tips segment

URL: <https://openai.com/index/openai-api/>

## NewsAPI

Fetches news from external sites

URL: <https://newsapi.org/>

## Flaticon

Turtle stickers created by Stickers - Flaticon

URL: <https://www.flaticon.com/free-stickers/turtle>

## Trash Dataset

N. V. Kumsetty, A. Bhat Nekkare, S. K. S. and A. Kumar M., "TrashBox: Trash Detection and Classification using Quantum Transfer Learning," 2022 31st Conference of Open Innovations Association (FRUCT), 2022, pp. 125-130, doi: 10.23919/FRUCT54823.2022.9770922.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9770922&isnumber=9770880>  
[https://github.com/nikhilvenkatkumsetty/TrashBox/tree/main/TrashBox\\_train\\_dataset\\_subfolders](https://github.com/nikhilvenkatkumsetty/TrashBox/tree/main/TrashBox_train_dataset_subfolders)