Code Quality Report

ARC - Artificial Recognition of Cannabis

ENSE 400/477 2022/23

Gregory Sveinbjornson

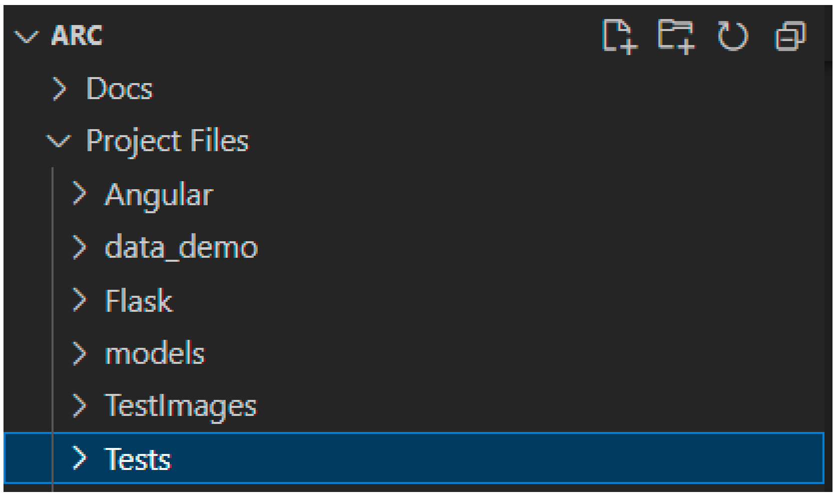
Feras Daghmoush

Introduction

The contents of this report includes the code quality analysis of our 4th year capstone project. As we worked with a corporate partner in BudSense, we wanted to ensure our code was up to their standards and could be used in industry. To ensure this, we went through our code with BudSense to make sure the quality was high enough to be used by them. This report will also include those responses.

Folder Structure

For our folder structure, we split up our code into different folders depending on what category the code file fits into. Here is our main overview:



```
ARC
  > Docs
  v Project Files
    > Angular
    > data_demo
    > Flask
    > models
    > TestImages
    > Tests
```

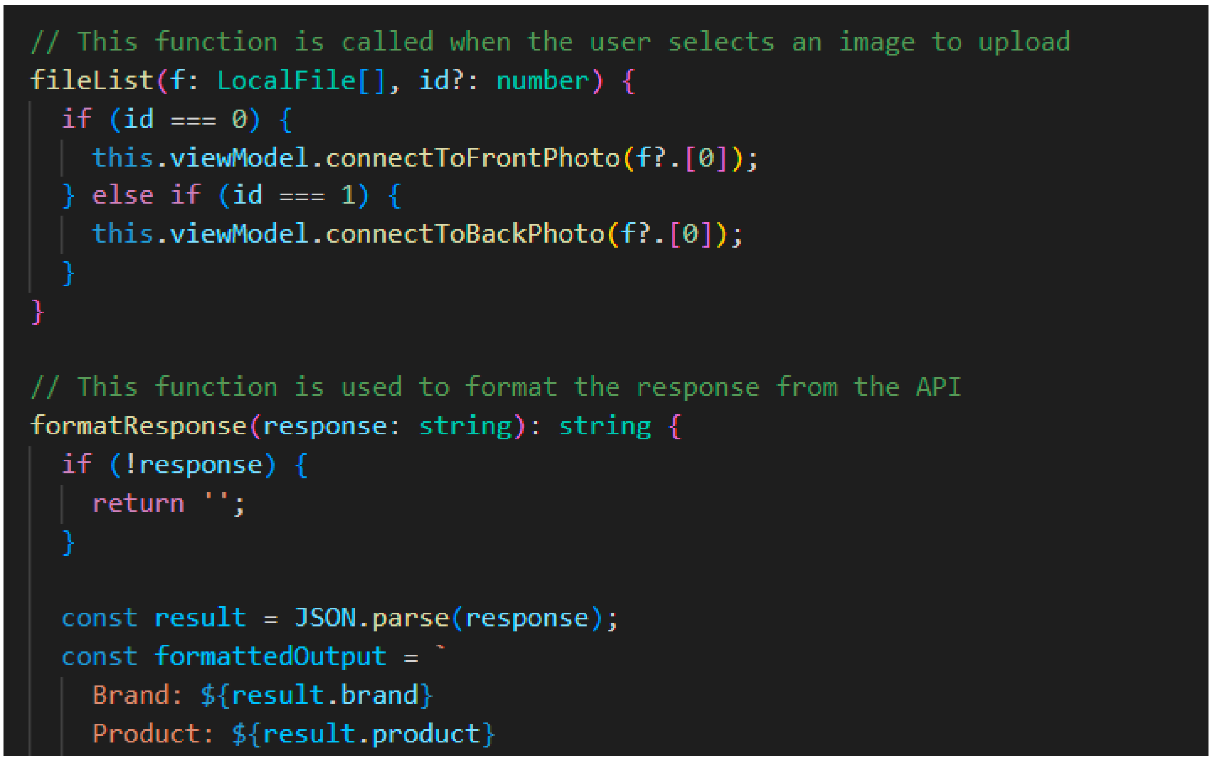
The folder ‘Angular’ holds all the files needed for our front end. The ‘data_demo’ folder is a small portion of our dataset designed to give anyone who looks at the code an idea of what our dataset looks like. The ‘Flask’ folder holds a copy of the code running on our Flask API hosted on an AWS EC2 instance. The folder ‘models’ holds versions 1 and 2 of our custom machine learning model. ‘TestImages’ holds our images used for testing the final project. Finally, ‘Tests’ contains all our testing files which are explained in detail in our Code Testing Report.

Readability

In attempting to make our code readable, we used three main strategies:

1. Using descriptive variable and function names
2. Ensuring related blocks of code are sectioned together
3. Using a consistent commenting style to describe difficult to understand functions.

Below is an example of descriptive function naming:

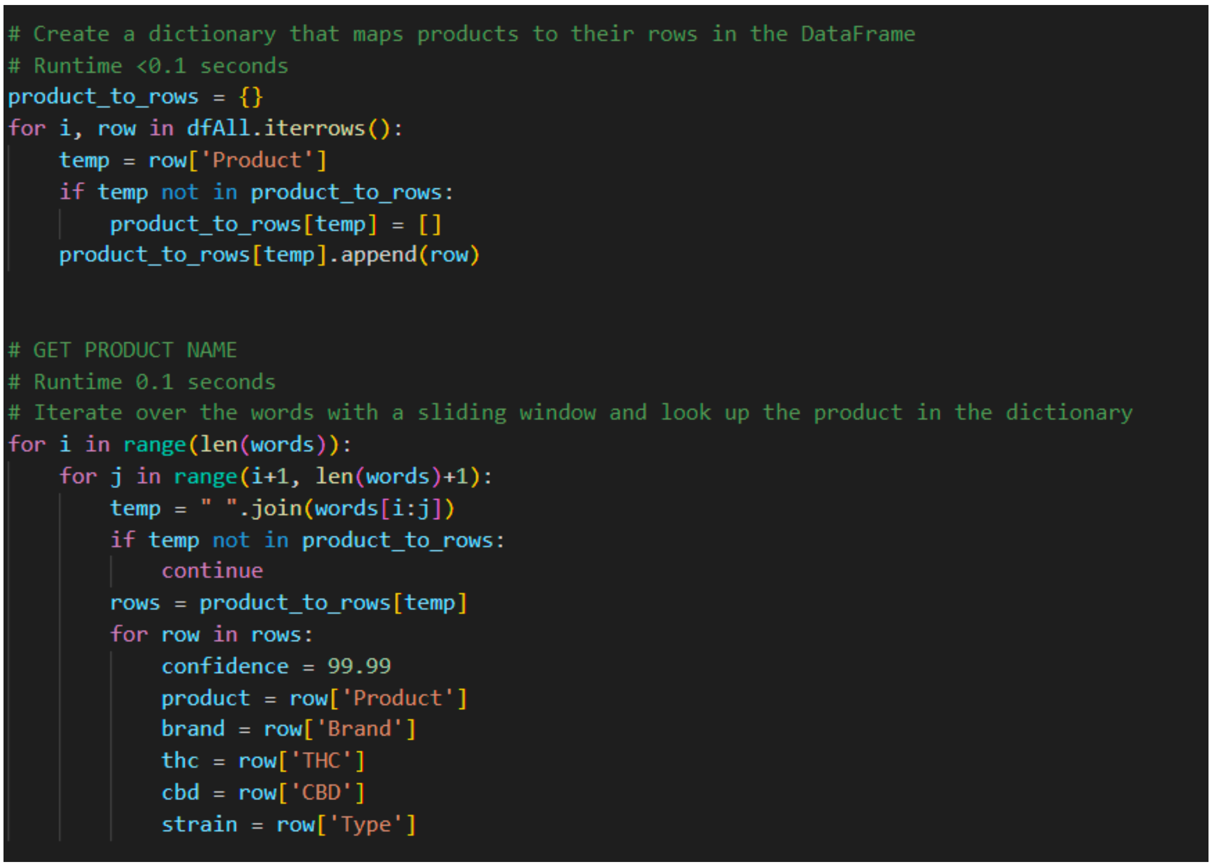


```
// This function is called when the user selects an image to upload
fileList(f: LocalFile[], id?: number) {
  if (id === 0) {
    this.viewModel.connectToFrontPhoto(f?.[0]);
  } else if (id === 1) {
    this.viewModel.connectToBackPhoto(f?.[0]);
  }
}

// This function is used to format the response from the API
formatResponse(response: string): string {
  if (!response) {
    return '';
  }

  const result = JSON.parse(response);
  const formattedOutput = `
    Brand: ${result.brand}
    Product: ${result.product}
```

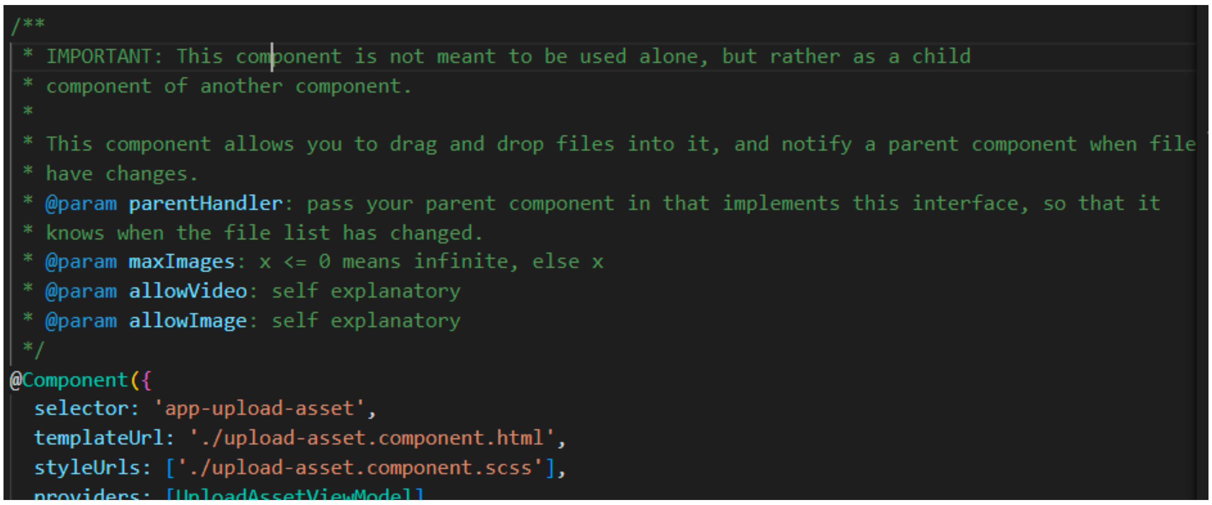
Below is an example of related blocks of code being section together:



```
# Create a dictionary that maps products to their rows in the DataFrame
# Runtime <0.1 seconds
product_to_rows = {}
for i, row in dfAll.iterrows():
    temp = row['Product']
    if temp not in product_to_rows:
        product_to_rows[temp] = []
    product_to_rows[temp].append(row)


# GET PRODUCT NAME
# Runtime 0.1 seconds
# Iterate over the words with a sliding window and look up the product in the dictionary
for i in range(len(words)):
    for j in range(i+1, len(words)+1):
        temp = " ".join(words[i:j])
        if temp not in product_to_rows:
            continue
        rows = product_to_rows[temp]
        for row in rows:
            confidence = 99.99
            product = row['Product']
            brand = row['Brand']
            thc = row['THC']
            cbd = row['CBD']
            strain = row['Type']
```

Below is an example of the commenting style used:



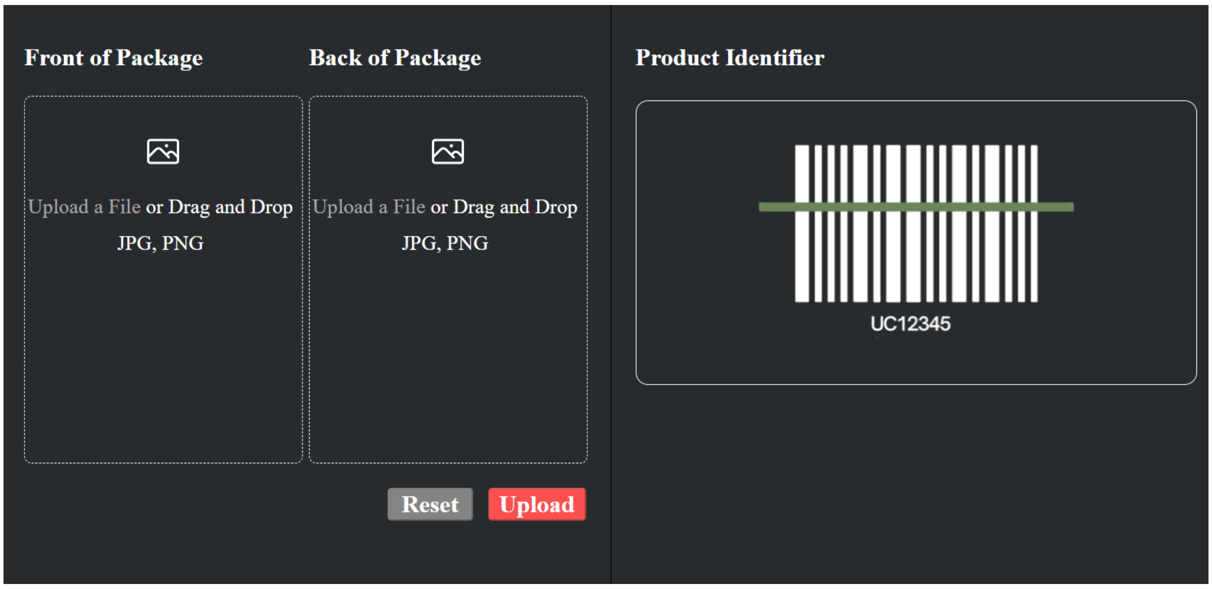
```
/**
 * IMPORTANT: This component is not meant to be used alone, but rather as a child
 * component of another component.
 *
 * This component allows you to drag and drop files into it, and notify a parent component when file
 * have changes.
 * @param parentHandler: pass your parent component in that implements this interface, so that it
 * knows when the file list has changed.
 * @param maxImages: x <= 0 means infinite, else x
 * @param allowVideo: self explanatory
 * @param allowImage: self explanatory
 */
@Component({
  selector: 'app-upload-asset',
  templateUrl: './upload-asset.component.html',
  styleUrls: ['./upload-asset.component.scss'],
  providers: [UploadAssetViewModel]
```

Portability

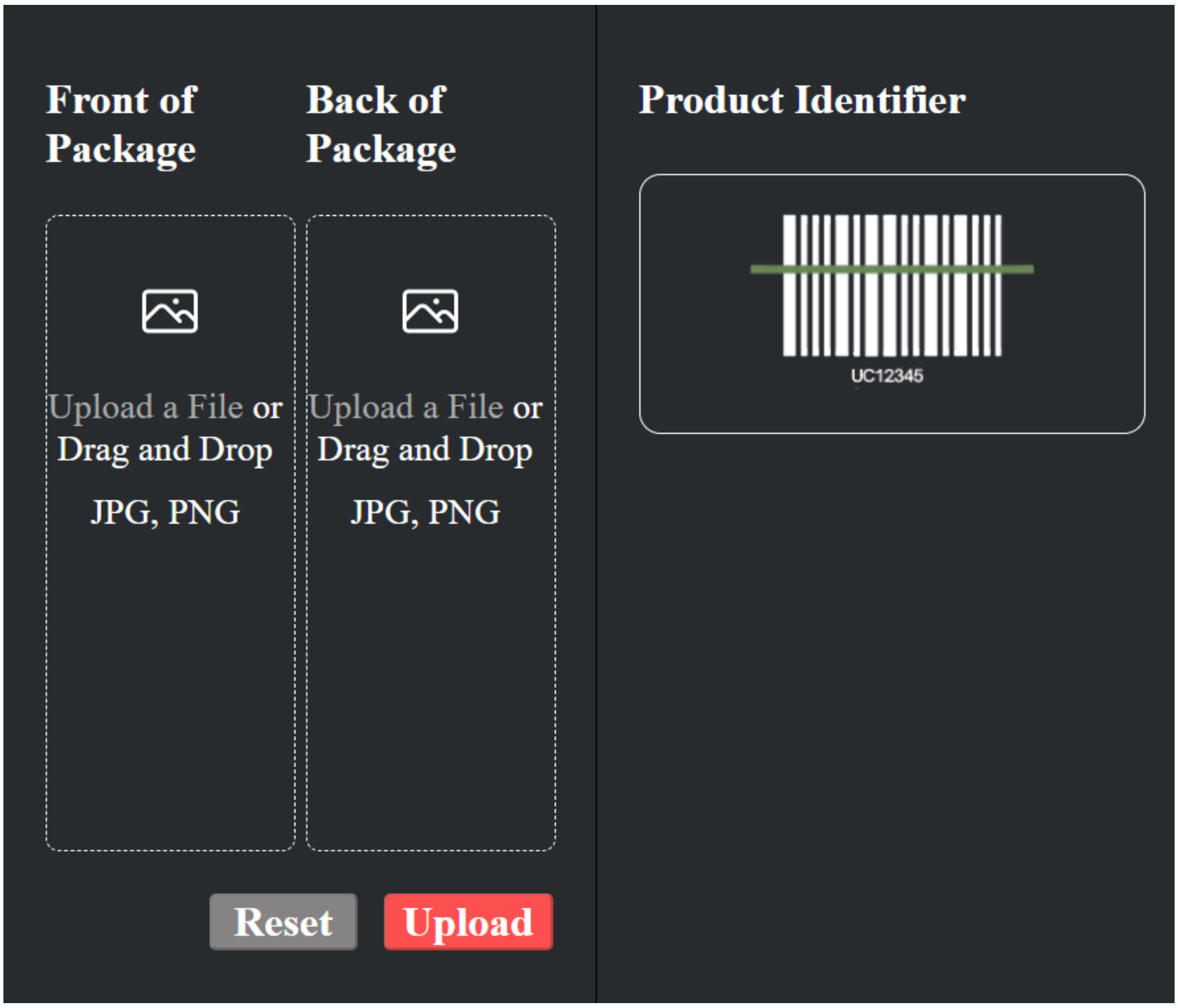
Our Angular code was made with an emphasis on portability as it has to be used on many different devices and browsers. This is why we leveraged our partnership with BudSense to create our Angular code, as their experience helped us create portable code that works well on any browser.

The styling of our page remains consistent on any screen size because of this, here are some examples.

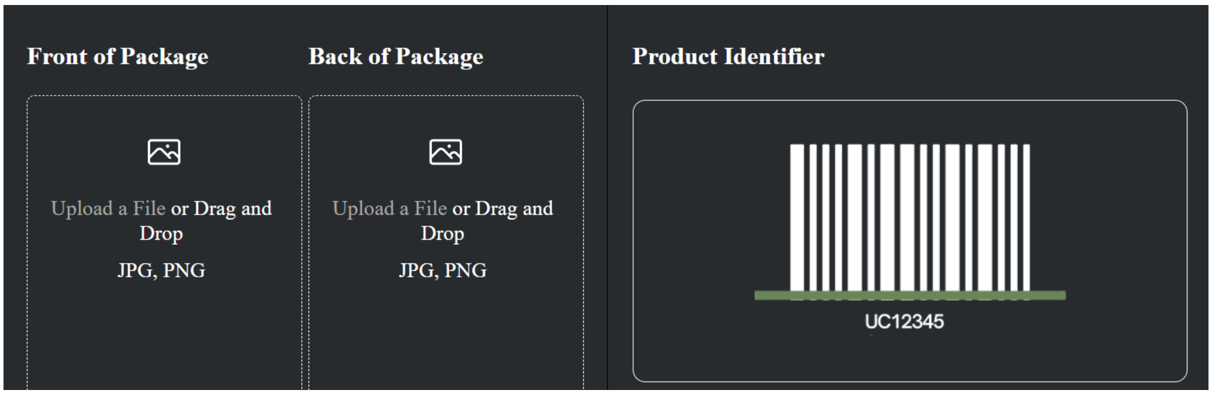
Regular:


Front of Package
Back of Package
Product Identifier
Upload a File or Drag and Drop
JPG, PNG
Upload a File or Drag and Drop
JPG, PNG
UC12345
Reset
Upload

Square:


Front of Package
Back of Package
Product Identifier
Upload a File or Drag and Drop
JPG, PNG
Upload a File or Drag and Drop
JPG, PNG
UC12345
Reset
Upload

Super wide:



Even though the screen sizes change drastically, the overall structure of the page doesn’t break.

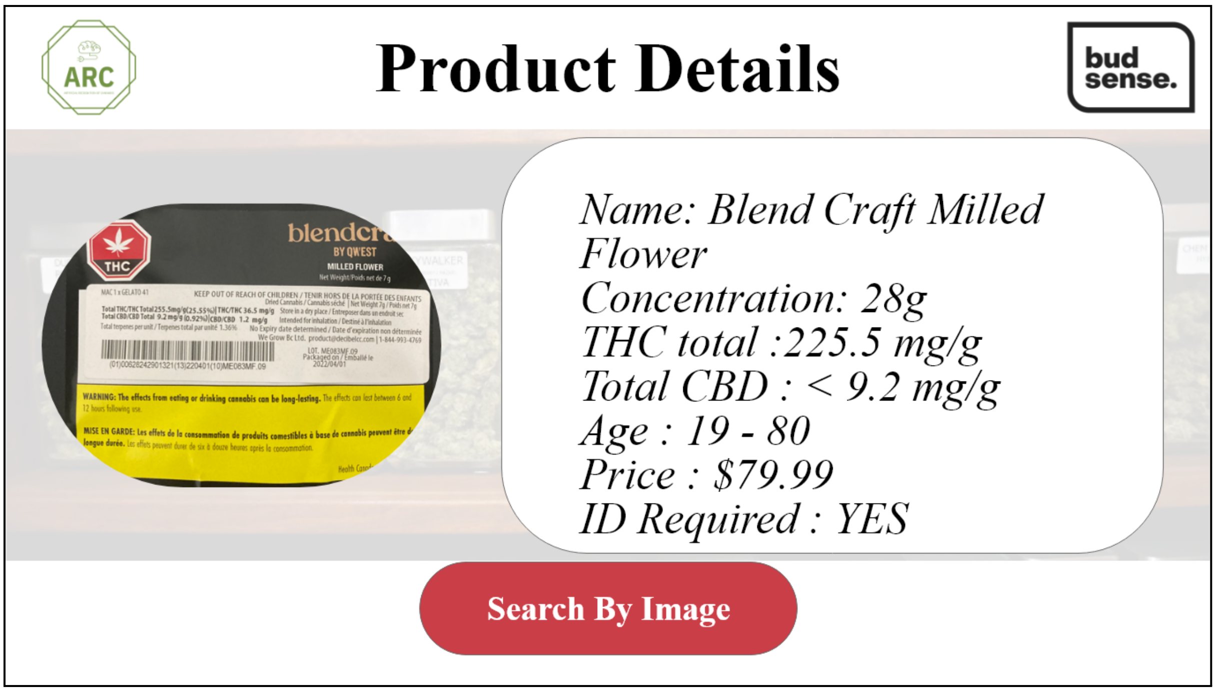
Security

Our security concerns were mostly limited to two spots in our program, sending the images to the API, and the service hosting the API itself. To ensure security when sending the images and getting a response, we used the HTTPS protocol to ensure encryption was used so no attackers could see what was being sent. To ensure security in our hosting service, we used a reputable service in AWS. We also set up a security group in AWS to ensure only legitimate actors can use our API.

Usability

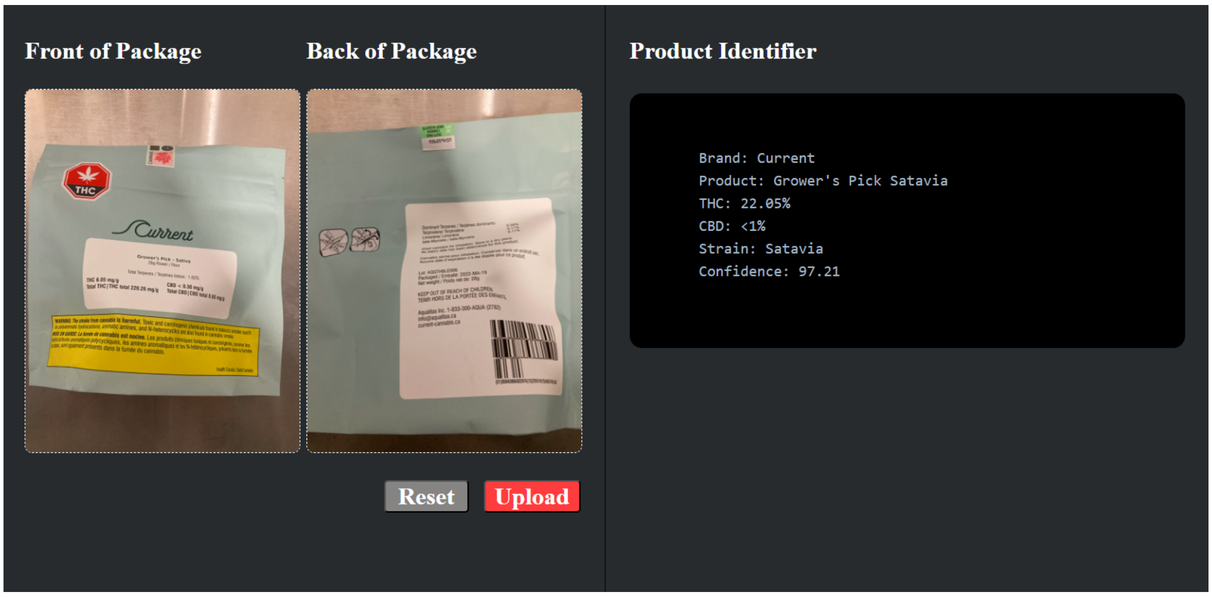
We wanted to ensure our project was quick and easy to use for our users, as employees would only have a use for our project if it increased efficiency. Because of this, the design philosophy of KISS - Keep it Simple, Stupid was implemented.

This change was made after feedback given to us on our High Fidelity Prototype, as many of the people we showed it to found it cluttered and confusing. Here is an example of one of the pages on our prototype, as there wre multiple:



Name: Blend Craft Milled Flower
Concentration: 28g
THC total :225.5 mg/g
Total CBD : < 9.2 mg/g
Age : 19 - 80
Price : $79.99
ID Required : YES

With the changes made from feedback given, our final user interface was much more simple and easy to read:



```
Brand: Current
Product: Grower's Pick Satavia
THC: 22.05%
CBD: <1%
Strain: Satavia
Confidence: 97.21
```

Maintainability

We believe our code is maintainable as we designed it to be readable for any programmer experienced in Angular and Flask development. In our experience modifying the code, it is difficult to get the whole system to break, and cannot be caused by changing one line of code. Features can be broken, but the overall project would be difficult to break on accident.

Conclusion

In conclusion, we believe our code is of good quality, which was echoed by BudSense. As they will be taking our code into their infrastructure, they believe it to be readable, portable, secure, usable, and maintainable enough to work with. We hope this continues to be true in our future work.