COMP4651 Project Report, 2021 Spring, HKUST

OCR Auto-Documenting Application for Firebase Backend

LIU, Yat Long (20520953), CHENG, Ho Sing (20513950), and
YEUNG, Man Yin Michael (20603418)

Section 1 Abstract

It has been a hard job for individuals to read and input text data manually. In this project, we implemented an application using some cloud technologies, including an Optical Character Recognition (OCR) on texts printed on food products, and storage of the data in a Firebase backend. Firebase and Google Cloud Vision text recognition (OCR) are used in our implementation. The text on the food packages are finally converted to values stored in the app, and are visualized in the user interface. The implementation is considered complete for the first step and further improvements can be made in different areas to counter the respective limitations.

Section 2 Cloud Technologies

Section 2.1 Firebase

Firebase is implemented as the backend in this project. It is a development platform designed by Google for building secure and scalable mobile and web applications. We are using flutter to develop the mobile project. As such, we choose firebase as the backend because it integrates well with a flutter project. There are also several advantages of using Firebase. Firebase has a lot of powerful cloud products such as Cloud Firestore, Firebase Storage, Cloud Messaging and Firebase Authentication. Moreover, Firebase service is free of charge which is suitable for small scale projects.

We have used Cloud Firestore and Firebase Storage in this project:

Cloud Firestore is a flexible, scalable NoSQL database for mobile, web, and server development from Firebase and Google Cloud. Cloud Firestore is used to store the nutrition information of the food products as shown in Figure 2 below.

Firebase Storage is used for storing and serving user-generated content, such as photos or videos. Firebase Storage is used to store the food product pictures as shown in Figure 2 below.

Section 2.2 Google Cloud Vision Text Recognition (OCR)

Google Cloud Vision is implemented for easier data input of the application data, and is only used by staff (not customers). It provides image recognition features such as detecting objects, detecting faces and identifying popular places and product logos, and etc. With the help of Text Recognition in Google Cloud Vision (figure 1), the process of digitizing the food label is greatly shortened. It can recognize the texts in the image taken by the staff and convert the texts to Strings.

Figure 1

Text Recognition on an Image of a Vita Tea Drink

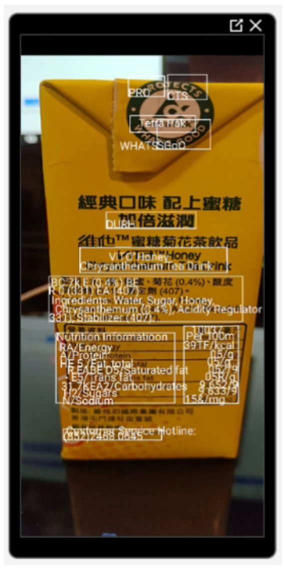
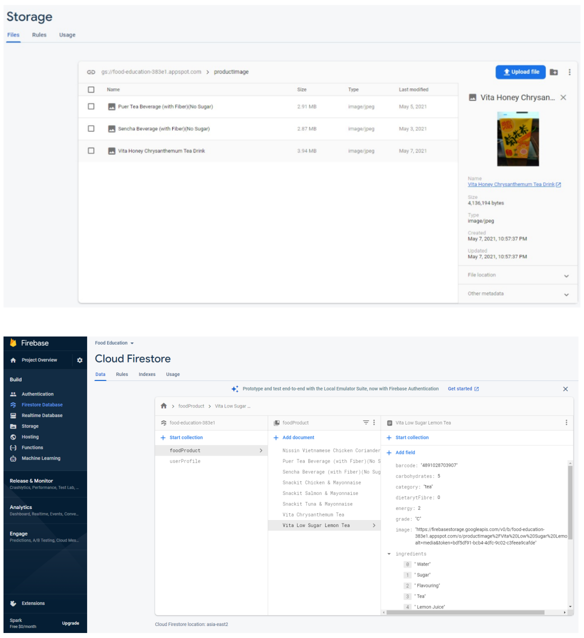


Figure 2

Information of a Vita Tea Drink Stored in Firebase



Storage

Files | Rules | Usage

gs://food-education-383e1.appspot.com > productImage

Name	Size	Type	Last modified
Puer Tea Beverage (with Fiber)(No Sugar)	2.91 MB	image/jpeg	May 5, 2021
Sencha Beverage (with Fiber)(No Sugar)	2.87 MB	image/jpeg	May 3, 2021
Vita Honey Chrysanthemum Tea Drink	3.94 MB	image/jpeg	May 7, 2021

Vita Honey Chrysan...

Name: Vita Honey Chrysanthemum Tea Drink
Size: 4,136,194 bytes
Type: image/jpeg
Created: May 7, 2021, 10:57:37 PM
Updated: May 7, 2021, 10:57:37 PM

Firebase — Food Education

Cloud Firestore

Data | Rules | Indexes | Usage

Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication Get started

foodProduct > Vita Low Sugar ...

food-education-383e1: foodProduct, userProfile

foodProduct: Nissin Vietnamese Chicken Coriander, Puer Tea Beverage (with Fiber)(No S, Sencha Beverage (with Fiber)(No Sug, Snackit Chicken & Mayonnaise, Snackit Salmon & Mayonnaise, Snackit Tuna & Mayonnaise, Vita Chrysanthemum Tea, Vita Low Sugar Lemon Tea

Vita Low Sugar Lemon Tea

```
barcode: "4891028703907"
carbohydrates: 5
category: "tea"
dietaryFibre: 0
energy: 2
grade: "C"
image: "https://firebasestorage.googleapis.com/v0/b/food-education-383e1.appspot.com/o/productImage%2FVita%20Low%20Sugar%20Lemo
alt=media&token=bdf5df91-bcb4-4dfc-9c02-c3feea9cafde"
ingredients
  0 "Water"
  1 "Sugar"
  2 "Flavouring"
  3 "Tea"
  4 "Lemon Juice"
```

Cloud Firestore location: asia-east2

Section 3 Algorithm

Section 3.1 Upload Data to Firebase

There are 5 majors steps to upload food product picture to Firebase Storage and food product data to Firestore Database, including:

1. Create a user defined class called “FoodProduct” which consists of all essential data needed to store in Firebase when the application starts. (Including name,barcode,nutrition information, ingredient list and image)
2. Fill data into the “FoodProduct” object by inputting in the app.
3. Upload the image of the food product and store it in the “productImage/” folder in Firebase Storage.
4. Get the downloadURL for the image and store in the “FoodProduct” object.
5. Upload the complete “FoodProduct” object to Firestore Database.

The detailed codes are shown in Appendix II. The detailed result are shown in Figure 2.

Section 3.2 Collect Texts in Ingredient List

There are 4 major steps to collect the ingredient list of a food product, including:

1. Capture multiple text paragraphs by the Google Cloud Vision OCR API and store in a list.
2. Read the list object to check if any text paragraph contains the keyword “Ingredient”.
3. If a paragraph contains the keyword ”Ingredient”, the words before “:” will be cut away by the substring function.
4. The words in the paragraph will then be split by “,” and stored into the “FoodProduct” object which was created when the app starts.

The detailed codes are shown in Appendix III.

Section 4 Usage of the Application

This application aims to facilitate and text data input processes. For example, this application can be used to help build another application for consumers to analyze different kinds of food products before purchases. Before the application can be fully available to the consumers, data of food products must be stored in the database in advance. As such, staff has to complete the data input process. However, it is very challenging and time-consuming for staff to type everything into the database. Viewing that, using text recognition for the text data will yield much faster progress. There are five majors steps for the staff to input all the data required using the implementation of this project, including:

1. Type in the Basic Information, e.g., Product Name, Category, Volume, etc.
2. Scan the Barcode for the Food Product
3. Read the Text on the Food Package using Google Cloud Vision text recognition (OCR)
4. Take a Picture of the Food Product Appearance
5. Upload the picture to Firebase Storage and upload other information to Firestore Database

Section 5 Limitations

- It is difficult or sometimes infeasible for the application to detect texts that are too small in size. This is a great issue because a lot of food products have very small texts on their packaging. To solve this issue, the OCR may have to change as Google Cloud Vision text recognition is not available to do so.
- Google Cloud Vision text recognition cannot detect tables. It is a great issue as the nutrition labels are normally in tabular form. In such cases, the app is not able to detect important figures like energy, fat, and protein.
- The result of the text recognition is sometimes incorrect. We suspect that unclear images (e.g. handshaking) and camera resolution may be the causes. This is a limitation as we do not have image clarity or camera resolution checkings.
- Some lists of ingredients may not be just simply separated by “,”, for example, some use other characters or spaces.

Section 6 Future Improvements & Developments

Section 6.1 Improving the Effectiveness of Data Input

For the OCR component, the Text Recognition from Google Cloud Vision can be replaced by the Amazon Textract to extend the text recognition to tables. For example, normally nutrition values are printed on the food products packages in a tabular form. Currently, Text Recognition from Google Cloud Vision cannot detect tables after several testings.

Section 6.2 Food Products Analysis by Consumers

As this app only provides the data input process. It does not directly provide help to a wide spectrum of audience or the general public. We propose that a Food Product Analysis app can be further developed in the future. In this app, the consumers can analyse food products and compare them with other similar products. As shown in the figure below, we can see that the consumers see how a particular food product performs in different aspects, and the rating and suggestions in general. (Desired UI can be found in Appendix IV)

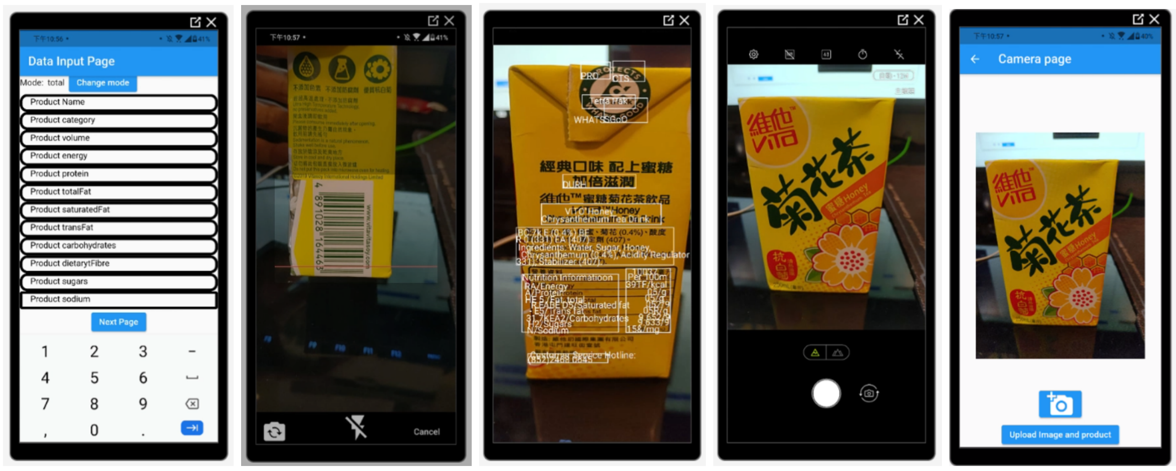
Section 7 Conclusion

The implementation of the application in this project can facilitate the process of text data input and make it much more efficient. Cloud technologies including Firebase and Google Cloud Vision text recognition (OCR) are also utilized to facilitate the implementation of the app. The performance of the final product is considered rather satisfying, but some future improvements and developments can still be carried out to enhance the user experience and significance of the app in various aspects.

Appendices

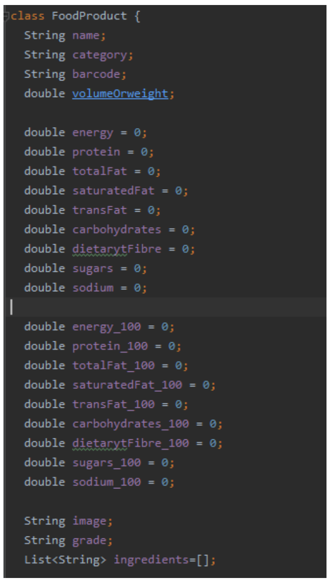
Appendix I

Figure 3 *Five Steps for the Staff to Input Food Products Data*

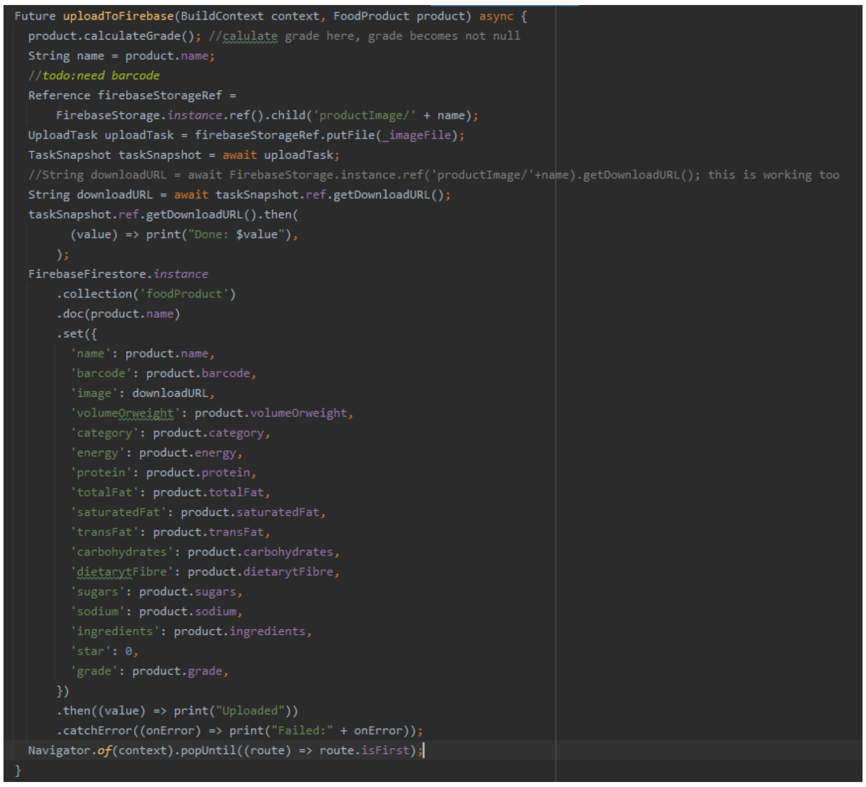


Appendix II

Figure 4 *Uploading Data to Firebase (in dart)*



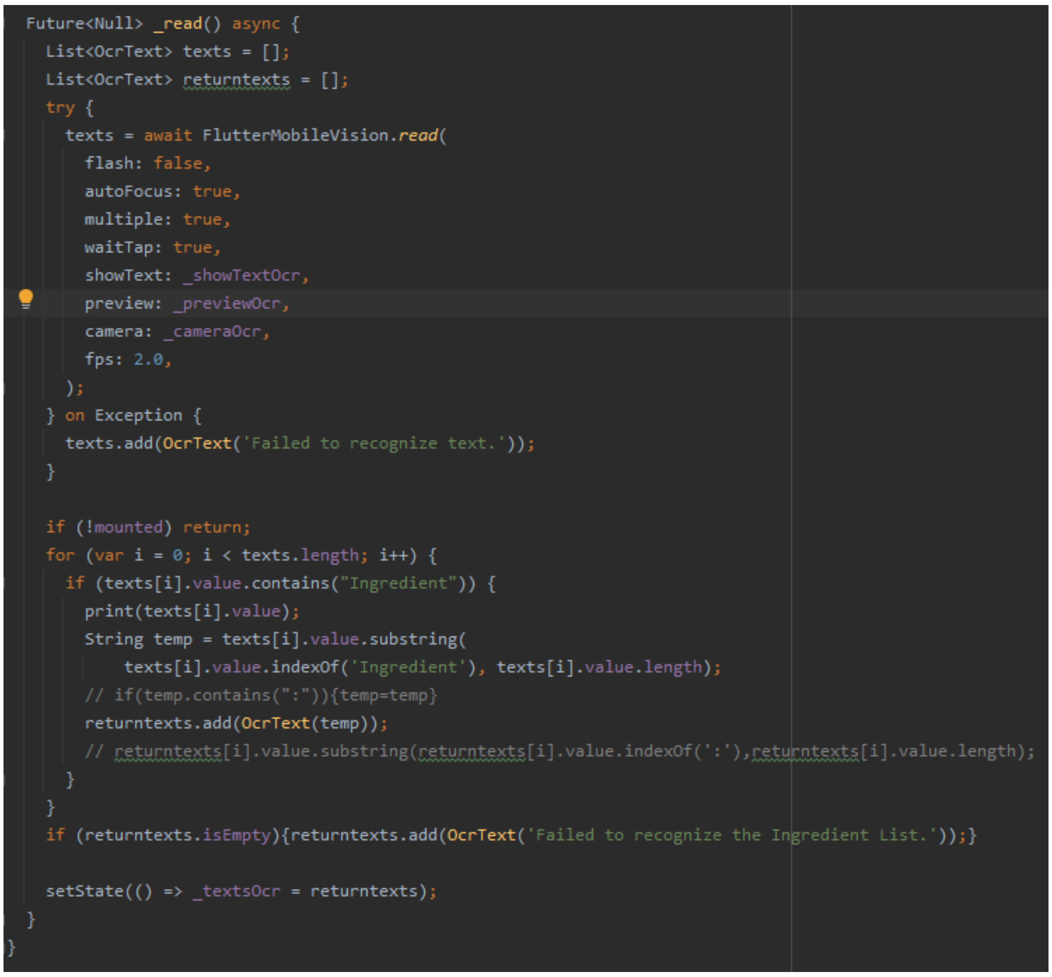
```dart
class FoodProduct {
  String name;
  String category;
  String barcode;
  double volumeOrweight;

  double energy = 0;
  double protein = 0;
  double totalFat = 0;
  double saturatedFat = 0;
  double transFat = 0;
  double carbohydrates = 0;
  double dietarytFibre = 0;
  double sugars = 0;
  double sodium = 0;

  double energy_100 = 0;
  double protein_100 = 0;
  double totalFat_100 = 0;
  double saturatedFat_100 = 0;
  double transFat_100 = 0;
  double carbohydrates_100 = 0;
  double dietarytFibre_100 = 0;
  double sugars_100 = 0;
  double sodium_100 = 0;

  String image;
  String grade;
  List<String> ingredients=[];
```



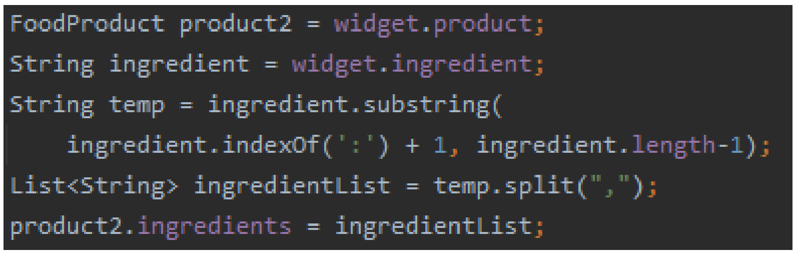
```dart
Future uploadToFirebase(BuildContext context, FoodProduct product) async {
  product.calculateGrade(); //calulate grade here, grade becomes not null
  String name = product.name;
  //todo:need barcode
  Reference firebaseStorageRef =
      FirebaseStorage.instance.ref().child('productImage/' + name);
  UploadTask uploadTask = firebaseStorageRef.putFile(_imageFile);
  TaskSnapshot taskSnapshot = await uploadTask;
  //String downloadURL = await FirebaseStorage.instance.ref('productImage/'+name).getDownloadURL(); this is working too
  String downloadURL = await taskSnapshot.ref.getDownloadURL();
  taskSnapshot.ref.getDownloadURL().then(
        (value) => print("Done: $value"),
      );
  FirebaseFirestore.instance
      .collection('foodProduct')
      .doc(product.name)
      .set({
        'name': product.name,
        'barcode': product.barcode,
        'image': downloadURL,
        'volumeOrweight': product.volumeOrweight,
        'category': product.category,
        'energy': product.energy,
        'protein': product.protein,
        'totalFat': product.totalFat,
        'saturatedFat': product.saturatedFat,
        'transFat': product.transFat,
        'carbohydrates': product.carbohydrates,
        'dietarytFibre': product.dietarytFibre,
        'sugars': product.sugars,
        'sodium': product.sodium,
        'ingredients': product.ingredients,
        'star': 0,
        'grade': product.grade,
      })
      .then((value) => print("Uploaded"))
      .catchError((onError) => print("Failed:" + onError));
  Navigator.of(context).popUntil((route) => route.isFirst);
}
```

Appendix III

Figure 5 Collecting Texts in Ingredients List (in dart)



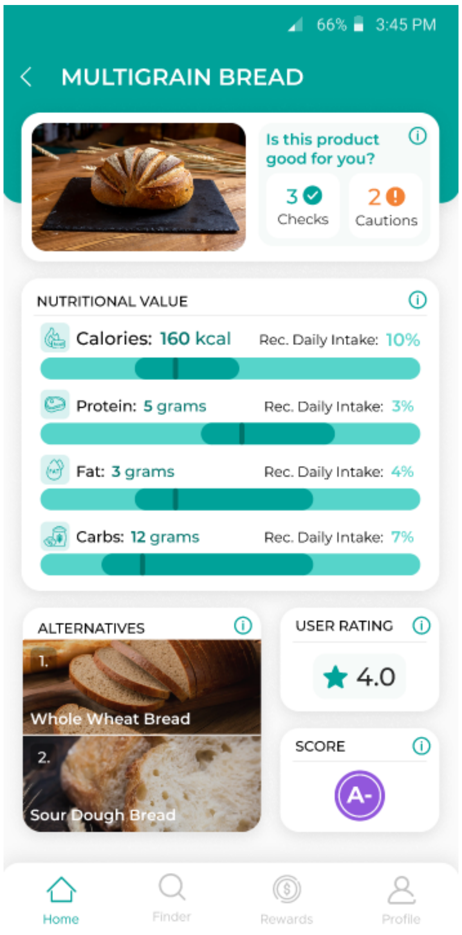
```dart
 Future<Null> _read() async {
   List<OcrText> texts = [];
   List<OcrText> returntexts = [];
   try {
     texts = await FlutterMobileVision.read(
       flash: false,
       autoFocus: true,
       multiple: true,
       waitTap: true,
       showText: _showTextOcr,
       preview: _previewOcr,
       camera: _cameraOcr,
       fps: 2.0,
     );
   } on Exception {
     texts.add(OcrText('Failed to recognize text.'));
   }

   if (!mounted) return;
   for (var i = 0; i < texts.length; i++) {
     if (texts[i].value.contains("Ingredient")) {
       print(texts[i].value);
       String temp = texts[i].value.substring(
           texts[i].value.indexOf('Ingredient'), texts[i].value.length);
       // if(temp.contains(":")){temp=temp}
       returntexts.add(OcrText(temp));
       // returntexts[i].value.substring(returntexts[i].value.indexOf(':'),returntexts[i].value.length);
     }
   }
   if (returntexts.isEmpty){returntexts.add(OcrText('Failed to recognize the Ingredient List.'));}

   setState(() => _textsOcr = returntexts);
 }
}
```



```dart
FoodProduct product2 = widget.product;
String ingredient = widget.ingredient;
String temp = ingredient.substring(
    ingredient.indexOf(':') + 1, ingredient.length-1);
List<String> ingredientList = temp.split(",");
product2.ingredients = ingredientList;
```

Appendix IV

Figure 6 An Expected UI of the App for Consumers



66% 3:45 PM

MULTIGRAIN BREAD

Is this product good for you?

3 Checks

2 Cautions

NUTRITIONAL VALUE

Calories: 160 kcal Rec. Daily Intake: 10%

Protein: 5 grams Rec. Daily Intake: 3%

Fat: 3 grams Rec. Daily Intake: 4%

Carbs: 12 grams Rec. Daily Intake: 7%

ALTERNATIVES

1. Whole Wheat Bread

2. Sour Dough Bread

USER RATING

4.0

SCORE

A-

END OF REPORT