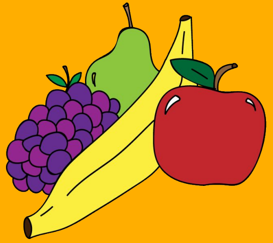


Picky Eater

A recipe search assistant Android application.

Team Tame Name:

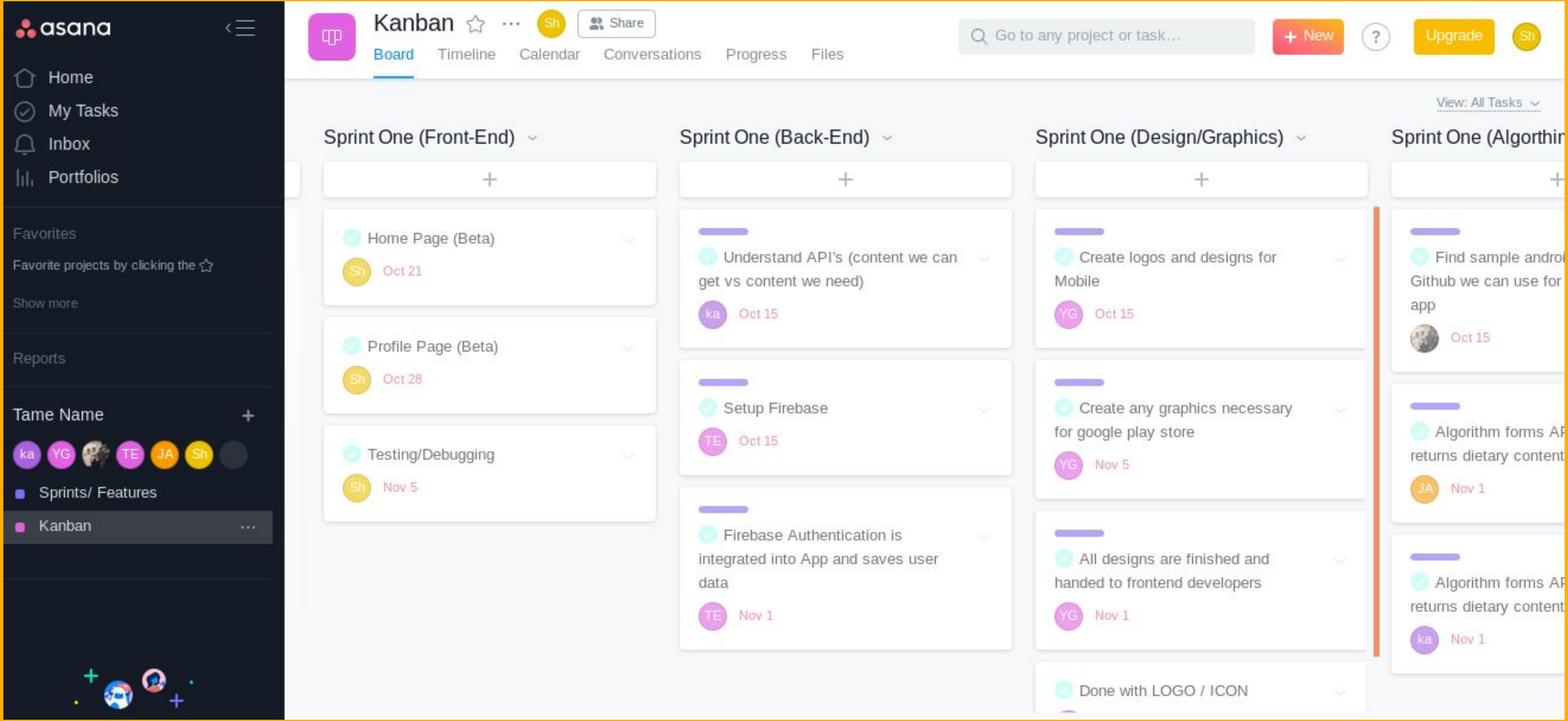
Josh Asmussen
Timothy Euken
Yiou Gao
David Gerstle
Shane Murphy
Kamen Shah

Tools Used

- **Android Studio: Development IDE** ★★★★☆
- **GitHub: Version Control** ★★★★★
- **Asana: Project Management** ★★★★☆
- **Slack: Team Communication Tool** ★★★★★
- **Adobe Illustrator/Photoshop: Graphic Design** ★★★★★
- **Google Firebase: Cloud Database** ★★★★★
- **Calabash: Testing Tool** ★★★★★

Project Management Methods Used

- Agile
- SCRUM meetings
- Peer Code Reviews
- Pair Programming
- Asana & Slack



Sprint One (Front-End)

- Home Page (Beta) — Sh Oct 21
- Profile Page (Beta) — Sh Oct 28
- Testing/Debugging — Sh Nov 5

Sprint One (Back-End)

- Understand API's (content we can get vs content we need) — ka Oct 15
- Setup Firebase — TE Oct 15
- Firebase Authentication is integrated into App and saves user data — TE Nov 1

Sprint One (Design/Graphics)

- Create logos and designs for Mobile — YG Oct 15
- Create any graphics necessary for google play store — YG Nov 5
- All designs are finished and handed to frontend developers — YG Nov 1
- Done with LOGO / ICON

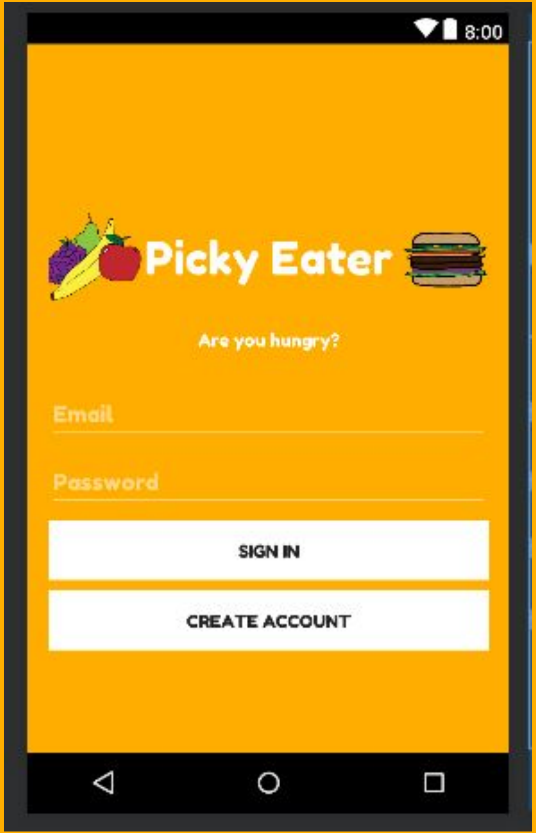
Sprint One (Algorithn

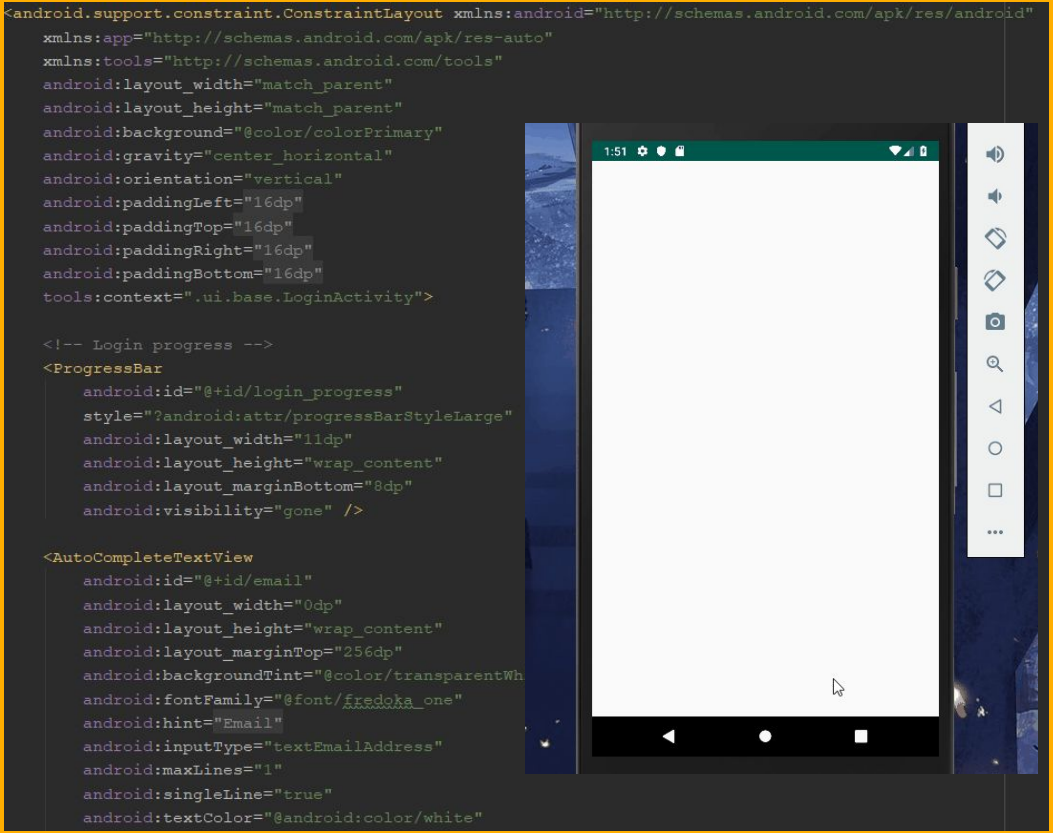
- Find sample androi Github we can use for app — Oct 15
- Algorithm forms AF returns dietary content — JA Nov 1
- Algorithm forms AF returns dietary content — ka Nov 1

Lessons Learned

- Time Management Priorities
- Communicate and confirm team meetings early
- Make use of open source tools
- Adobe XD should be used to create a mock UI of application
- If an error occurs in AS, check dependencies and gradle file

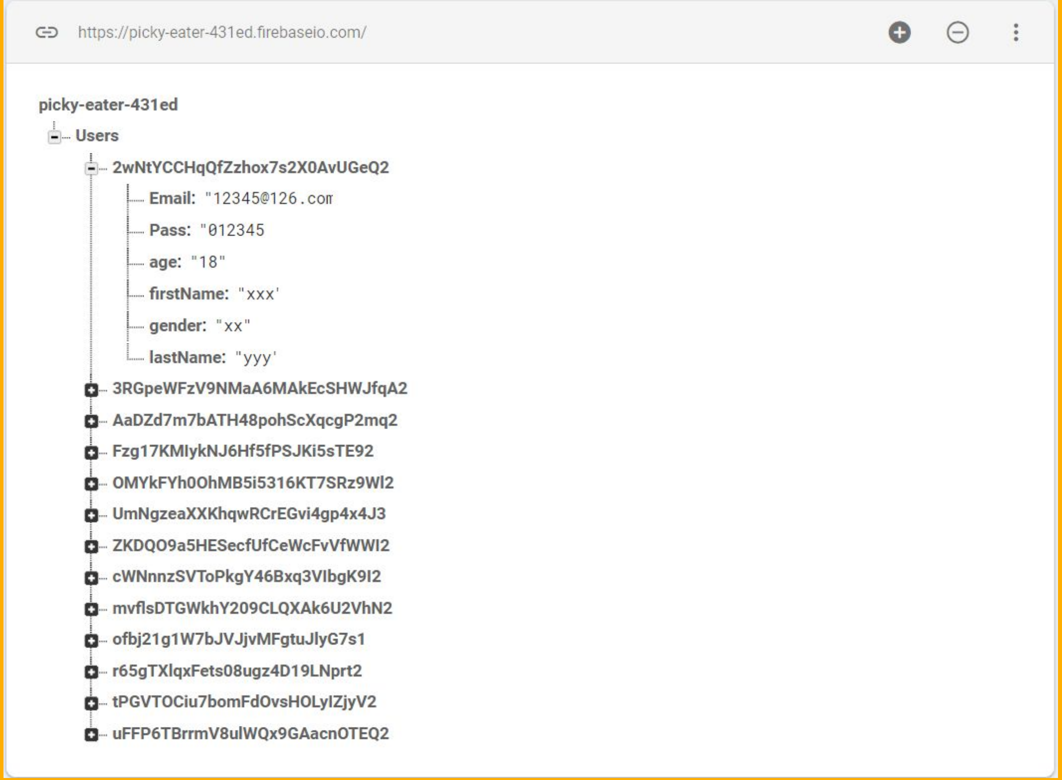
Frontend / Design





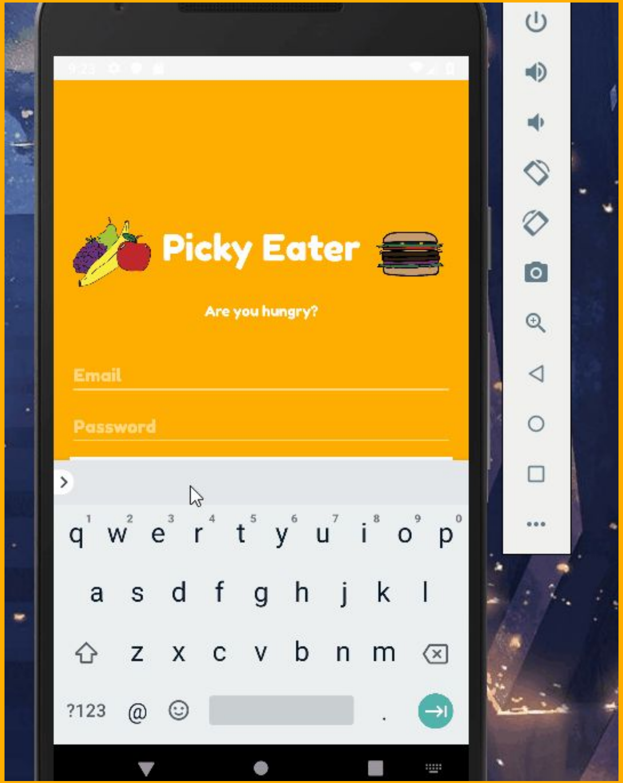
```
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimary"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingLeft="16dp"
    android:paddingTop="16dp"
    android:paddingRight="16dp"
    android:paddingBottom="16dp"
    tools:context=".ui.base.LoginActivity">

    <!-- Login progress -->
    <ProgressBar
        android:id="@+id/login_progress"
        style="?android:attr/progressBarStyleLarge"
        android:layout_width="11dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:visibility="gone" />

    <AutoCompleteTextView
        android:id="@+id/email"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="256dp"
        android:backgroundTint="@color/transparentWh
        android:fontFamily="@font/fredoka_one"
        android:hint="Email"
        android:inputType="textEmailAddress"
        android:maxLines="1"
        android:singleLine="true"
        android:textColor="@android:color/white"
```

Firebase



https://picky-eater-431ed.firebaseio.com/

```
picky-eater-431ed
 ⊟- Users
     ⊟- 2wNtYCCHqQfZzhox7s2X0AvUGeQ2
          ├- Email: "12345@126.com
          ├- Pass: "012345
          ├- age: "18"
          ├- firstName: "xxx'
          ├- gender: "xx"
          └- lastName: "yyy'
     ⊞- 3RGpeWFzV9NMaA6MAkEcSHWJfqA2
     ⊞- AaDZd7m7bATH48pohScXqcgP2mq2
     ⊞- Fzg17KMlykNJ6Hf5fPSJKi5sTE92
     ⊞- OMYkFYh0OhMB5i5316KT7SRz9Wl2
     ⊞- UmNgzeaXXKhqwRCrEGvi4gp4x4J3
     ⊞- ZKDQO9a5HESecfUfCeWcFvVfWWI2
     ⊞- cWNnnzSVToPkgY46Bxq3VlbgK9I2
     ⊞- mvflsDTGWkhY209CLQXAk6U2VhN2
     ⊞- ofbj21g1W7bJVJjvMFgtuJlyG7s1
     ⊞- r65gTXlqxFets08ugz4D19LNprt2
     ⊞- tPGVTOCiu7bomFdOvsHOLyIZjyV2
     ⊞- uFFP6TBrrmV8ulWQx9GAacnOTEQ2
```



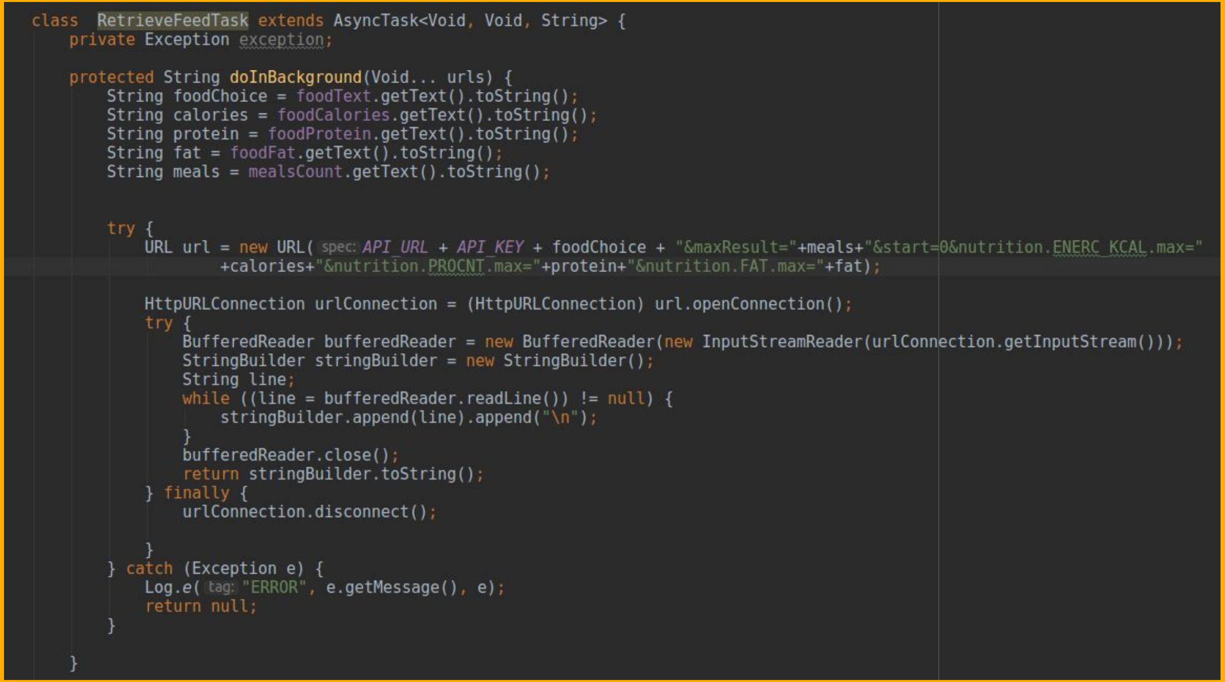
https://console.firebase.google.com/u/0/

API

Yummly™

https://developer.yummly.com/

API Call



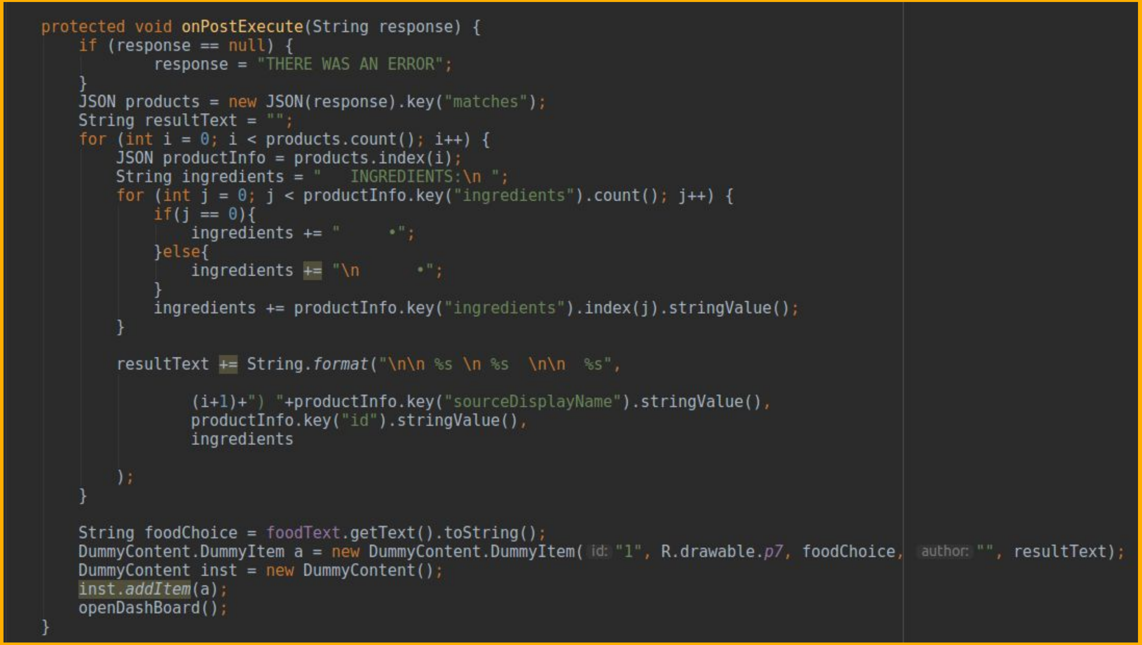
```
class  RetrieveFeedTask extends AsyncTask<Void, Void, String> {
    private Exception exception;

    protected String doInBackground(Void... urls) {
        String foodChoice = foodText.getText().toString();
        String calories = foodCalories.getText().toString();
        String protein = foodProtein.getText().toString();
        String fat = foodFat.getText().toString();
        String meals = mealsCount.getText().toString();


        try {
            URL url = new URL( spec: API_URL + API_KEY + foodChoice + "&maxResult="+meals+"&start=0&nutrition.ENERC_KCAL.max="
                    +calories+"&nutrition.PROCNT.max="+protein+"&nutrition.FAT.max="+fat);

            HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
            try {
                BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(urlConnection.getInputStream()));
                StringBuilder stringBuilder = new StringBuilder();
                String line;
                while ((line = bufferedReader.readLine()) != null) {
                    stringBuilder.append(line).append("\n");
                }
                bufferedReader.close();
                return stringBuilder.toString();
            } finally {
                urlConnection.disconnect();

            }
        } catch (Exception e) {
            Log.e( tag: "ERROR", e.getMessage(), e);
            return null;
        }

    }
```

Extracts user input and appends it into URL request

Makes URL connection

Stores JSON output in a string using StringBuilder

Parsing JSON Data



```java
protected void onPostExecute(String response) {
    if (response == null) {
            response = "THERE WAS AN ERROR";
    }
    JSON products = new JSON(response).key("matches");
    String resultText = "";
    for (int i = 0; i < products.count(); i++) {
        JSON productInfo = products.index(i);
        String ingredients = "   INGREDIENTS:\n ";
        for (int j = 0; j < productInfo.key("ingredients").count(); j++) {
            if(j == 0){
                ingredients += "     •";
            }else{
                ingredients += "\n      •";
            }
            ingredients += productInfo.key("ingredients").index(j).stringValue();
        }

        resultText += String.format("\n\n %s \n %s  \n\n  %s",

                (i+1)+") "+productInfo.key("sourceDisplayName").stringValue(),
                productInfo.key("id").stringValue(),
                ingredients

        );
    }

    String foodChoice = foodText.getText().toString();
    DummyContent.DummyItem a = new DummyContent.DummyItem( id: "1", R.drawable.p7, foodChoice,  author: "", resultText);
    DummyContent inst = new DummyContent();
    inst.addItem(a);
    openDashBoard();
}
```

JSON object is created with API call data

Another JSON object is created for each index of recipe

Data is formatted and displayed

Application Demonstration