Malhar Bangdiwala (2019130005)
Rutvik Choudhari (2019130010)
Yashvi Mehta (2019130037)
Group Number:4

# MINI PROJECT-1 REPORT

## I. TITLE

The title of our mini project is**:**

Recipease: Food identifier and recipe recommendation app

## II. PROBLEM DEFINITION

Many people are fans of eating and trying new things. But often they do not realize that the item they are eating might contain allergens or it does not have the nutritional value they think it does. Further, sometimes the language barrier can prevent them from knowing the exact name of the dish they are eating.

Therefore, we have made an app on which you can just upload the picture of the food item, and it would provide you with its name, the allergens it contains, its recipe and its nutritional value. The app will also make recommendations and give warnings based on your preferences.

## III. Market Survey:

We did a market survey of 4 competitors of our product to get a better idea. Those four products are as follows:

- **Shazam:** Shazam is an app that identifies the song by listening to its audio. It is similar to what our app is trying to achieve, but for songs
  Ref: Shazam (2021) [Online]. Available: https://www.shazam.com/

- **Hebbars Kitchen:** Hebbars Kitchen is an app that has both written and video recipes for over 100 food items. It also provides the nutrient values of the dish. However, it does not provide any food recognition platform.
  Ref: Hebbars Kitchen (2020) [Online]. Available: https://hebbarskitchen.com

- **Calorie Mama AI:** Calorie Mama AI is an app that identifies food from its image and provides the number of calories it contains. However, this app does not provide any information about the recipe or the allergens that it may contain.
  Ref: Calorie Mama AI (2017) [Online]. Available: https://caloriemama.ai/

- **ViVino:** ViVino is an app that identifies wine from its image. This has the image classification component of our app but nothing else.
  Ref: ViVino (2021) [Online]. Available: https://www.vivino.com/

Based on these four products, we concluded our survey and decided to make an app whose primary objective is to identify food items and in addition to this, it would not only provide written and video recipes of various food items, but also its nutritional value and ingredients. Therefore, it is an amalgamation of all the above-mentioned apps. Further, the app would also have a recommendation system in accordance to the users' preferences and warn the user about the allergens and spice level of the food.

## IV. SCOPE OF WORK AND OBJECTIVES

### 1. Working idea

The exact idea of our app involves identifying food items based on uploaded images with the help of Machine Learning Models, warn about allergens if present and provide the nutritional value and written and video recipes. The app will also provide recommendations for new dishes based on the user's liking.

### 2.Boundaries and constraints

The app is constraint by the fact that 100% accuracy of image identification will not be present. The app will recognize a limited number of food items and it will not

be detecting the exact ingredients present in the food item. Currently it cannot recognize multiple dishes present in one single picture.

**3.Future Scope**

In the future, the app will also include features like identifying multiple dishes from a single image and exact ingredients from the dish. The database from which the selection of dish takes place will be expanded from the current 120 items. Increasing the accuracy of the ML models is also part of the future scope.

**4.Objectives**

The main objectives of the app are:

- To implement image classification from the provided data set.
- To warn users about the allergens that might be present in the product.
- To provide the users with the nutritional value of the food item that has been uploaded.
- To provide recipes to the user (written and video).
- To recommend new dishes to the user.

## V. Architecture/Block Diagram:

1. **Food classification**

   The user can either click an image or choose an image from his/her android device. This image is the input to further processing. It is sent to the Linux server hosted on the Azure Cloud Services over the HTTP protocol. This image is accepted by the flask app. Preprocessing is done on the image and then it is passed through the food-non food classifier (convolutional neural network) which detects whether or not there is any food item present in the image. If the image has a food item, then it is passed through another classifier which detects what food is present in the image. Based on the above steps, appropriate response is sent to the client in JSON format.

2. **Similar food recommendation**

   Every food item in the database has an array specifying what items are present in it. Based on this array, whenever a user views any item, the array of that item is sent to the flask app that is hosted on the Heroku server. On the server, using nearest neighbor clustering, items similar to the current

item are filtered and sent to the client in JSON format. These items are then displayed to the user.

3. **User specific recommendation**

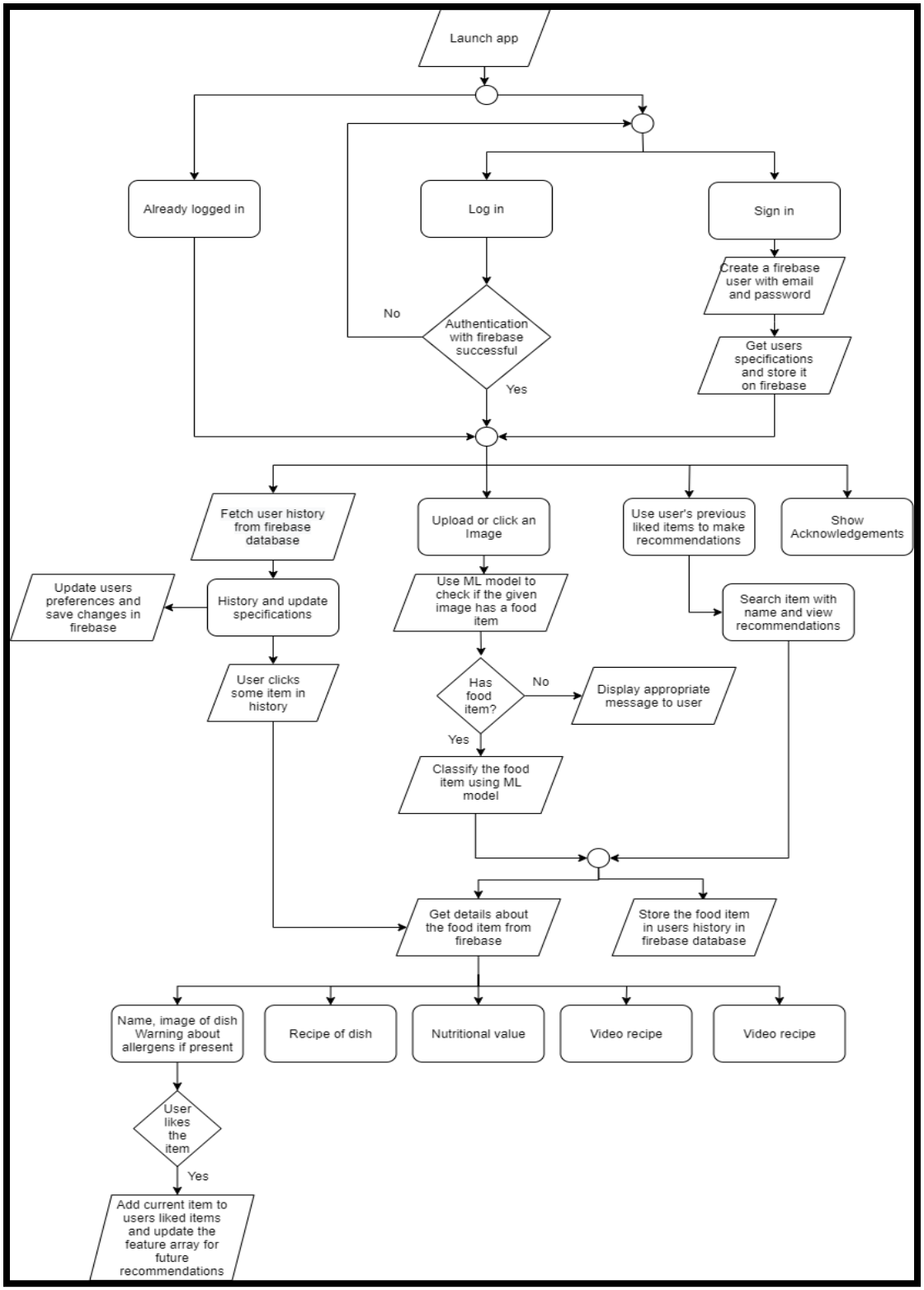
   As mentioned above, each item has an array specifying its ingredients, and each user has a similar array but it holds the weights of ingredients (initially zero). Whenever a user likes a particular item the weights of ingredients contained in that item are increased thus increasing priority. To get the user specific recommendations, this user array is sent to a flask app that is hosted on the Heroku platform. Nearest neighbor clustering is applied on the database and items similar to the ones that the user likes are fetched. These are then sent to the client in JSON format and displayed to the user.

4. **Block Diagram**

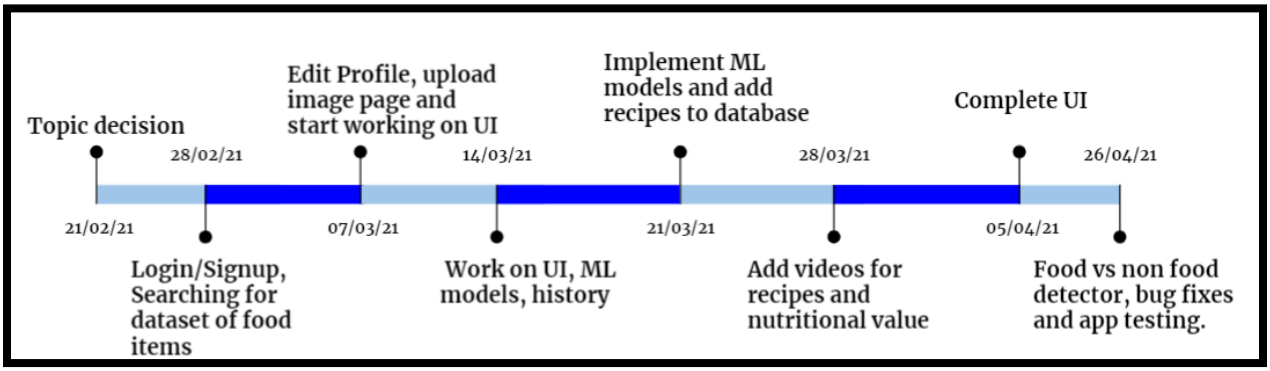
   The major modules of the app are as follows:

   1. ML model for food/non-food classification.
   2. ML model for dish classification.
   3. ML model for recommendations and similar items.
   4. Sign up/Log in pages
   5. Preferences page
   6. Upload page
   7. History page
   8. Search page
   9. Acknowledgement page
   10. Dish Information Module: Containing basic information, nutrient information, and recipes (written, YouTube video)
   11. Relevant alerts for the dish
   12. UI Module

   The exact flow of the app can be seen in the following block diagram:


Launch app
Already logged in
Log in
Sign in
Create a firebase user with email and password
No
Authentication with firebase successful
Get users specifications and store it on firebase
Yes
Fetch user history from firebase database
Upload or click an Image
Use user's previous liked items to make recommendations
Show Acknowledgements
Update users preferences and save changes in firebase
History and update specifications
Use ML model to check if the given image has a food item
Search item with name and view recommendations
User clicks some item in history
Has food item?
No
Display appropriate message to user
Yes
Classify the food item using ML model
Get details about the food item from firebase
Store the food item in users history in firebase database
Name, image of dish Warning about allergens if present
Recipe of dish
Nutritional value
Video recipe
Video recipe
User likes the item
Yes
Add current item to users liked items and update the feature array for future recommendations

## VI. PROJECT PLAN AND TIMELINE




**Phase 1:**

For phase 1 of our project, we completed the basic parts of the app (around 25% of the total work). This includes the login/signup page, basic UI, the food detection model, and setting up firebase for the user accounts.

Modules Completed: ML model for dish classification, Sign up/ Log in Pages, Upload Page, History Page (Partial), Preferences Page (Partial), Search Page (Partial).

**Phase 2:**

During the Phase 2, the major part of the app was completed (around 90% of the total work). We further worked on the UI, implemented the ML models, and completed the Dish Information Module. Different fragments to the app such as edit profile, history, and acknowledgements were completed during this period. The database for the dishes which includes their information, images, YouTube links for the videos etc. were also completed. All the alerts were also added.

Additional Modules Completed: ML model for recommendations and similar items, History Page, Preferences Page, Search Page, Acknowledgements Page, Dish Information Module, Relevant Alerts for the dish, UI Module (Partial).

**Phase 3:**

For the last phase, the feature of distinguishing non-food items from food items was added. The minor UI finalizations and bug fixes was completed.

Additional Modules Completed: UI Module, ML model for food/non-food classification.

## VII. IMPLEMENTATION DETAILS

### 1. Details of 100% Implementation Completed

- Signup/Login Page: The signup page includes the email, name and the password of the user. Users can login using their email and password. This module has been implemented using firebase.

- Preference Page: This page includes user's preferences of spice level and vegetarian, non-vegetarian or jain. Further, the user can add/remove any allergens in order to be warned if it is present in a dish.

- Upload page: This is the home page of the app. It is the page where the user can upload the image of the food item and the app will identify it for them. It has been implemented as a fragment. The food/nonfood classifier and the dish classification model has been implemented here.

- History page: The history page displays the history of the user, i.e., the items which the user has searched in the past. All the items are displayed using a list view and on clicking the item, the user is led to the dish information module. This page is also implemented as a fragment.

- Search Page: This page contains the recommendations for the user based on the dishes that he/she likes. The recommendation model has been implemented here. The items are displayed in a list view, similar to the one in the History Page. This page has also been implemented as a fragment. The search feature has also been used here by making use of an autocomplete text view.

- Acknowledgements Page: As the name suggests, this page acknowledges the three websites that we have used for all the information of the dishes. It has been implemented as a fragment. A list view is used and all the links redirect the user to the respective websites.

- Dish Home Page: This page is acts as the home page for the Dish Information Module. This module has been implemented as a separate

set of fragments. This page contains the name of the dish, its image and a like button.

- Recipe Page: The recipe fragment includes the ingredients of the food items, the cooking time as well as the steps. This has been implemented by using several recycler views.

- Nutrition Page: The nutritional value fragment specifies the overall nutritional value along with the quantity of each of the five food groups. This is done using recycles views, similar to the ones used in the recipe page.

- Video Page: In this fragment, a YouTube video for the given recipe is shown. This has been done using the YouTube API.

- Similar Food Page: In this fragment, items similar to the current item are displayed. The items are displayed in a list view, similar to the Recommendations Page. The recommendation model has been implemented here as well.

**2. Tech Stack**

The tech stack implemented in this app includes the following:

- Java: The entire Android app has been written in Java.
- Flask: Flask has been used for backend servers and RESTful APIs
- Firebase: Firebase has been used as the database for data storage and authentication of the user.
- Convolutional Neural Networks (Inception V3 Model, Google): CNN is used for all the machine learning models.
- Microsoft Azure: Azure Cloud Services is used for hosting image classification model.
- Heroku: Heroku is used for hosting the recommendation model.

**3.APIs**

The APIs that have been used for this app are as follows:

- TensorFlow: This has been used for the ML components.
- Sci-Kit Learn: This has also been used for the ML components.
- YouTube: This has been used to display the YouTube videos for the recipes.

## 4.Results

- **Food - Non-Food classifier**: The model has an accuracy of **97.35 %**
  - Total classes: 2
  - Total parameters: 113 M
  - Optimizer: RMSprop
  - Loss function: Categorical Cross entropy

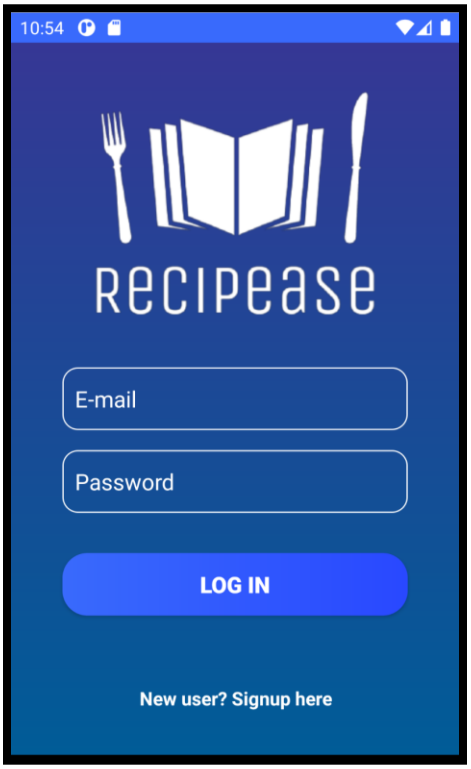
```
Epoch 1/10
410/410 - 2069s - loss: 0.2502 - accuracy: 0.9394 - val_loss: 0.1688 - val_accuracy: 0.9661
Epoch 2/10
410/410 - 2151s - loss: 0.1702 - accuracy: 0.9614 - val_loss: 0.1754 - val_accuracy: 0.9594
Epoch 3/10
410/410 - 2151s - loss: 0.1393 - accuracy: 0.9648 - val_loss: 0.1304 - val_accuracy: 0.9713
Epoch 4/10
410/410 - 2138s - loss: 0.1424 - accuracy: 0.9673 - val_loss: 0.1329 - val_accuracy: 0.9731
Epoch 5/10
410/410 - 2145s - loss: 0.1238 - accuracy: 0.9691 - val_loss: 0.1228 - val_accuracy: 0.9722
Epoch 6/10
410/410 - 2171s - loss: 0.1198 - accuracy: 0.9718 - val_loss: 0.1220 - val_accuracy: 0.9707
Epoch 7/10
410/410 - 2165s - loss: 0.1114 - accuracy: 0.9741 - val_loss: 0.1382 - val_accuracy: 0.9713
Epoch 8/10
410/410 - 2165s - loss: 0.1089 - accuracy: 0.9736 - val_loss: 0.1226 - val_accuracy: 0.9759
Epoch 9/10
410/410 - 2148s - loss: 0.1037 - accuracy: 0.9751 - val_loss: 0.1505 - val_accuracy: 0.9716
Epoch 10/10
410/410 - 2151s - loss: 0.1054 - accuracy: 0.9751 - val_loss: 0.1356 - val_accuracy: 0.9735
```

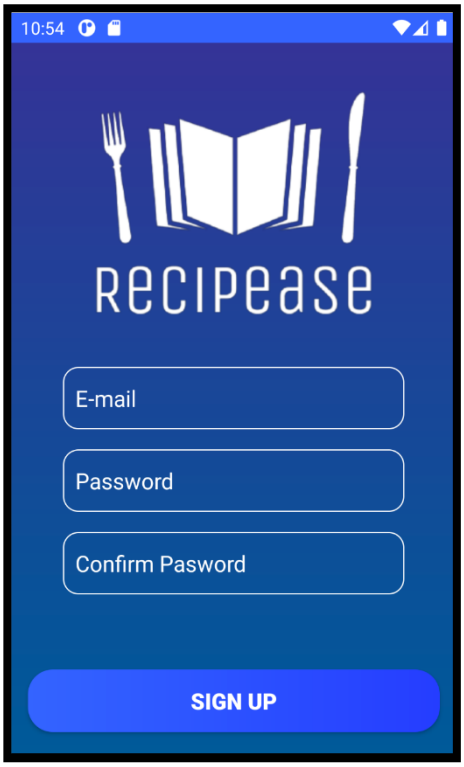
- **Food item classifier:** The model has an accuracy of **80.60 %**
  - Total classes: 40
  - Total parameters: 113 M
  - Optimizer: RMSprop
  - Loss function: Categorical Cross entropy

```
Epoch 1/10
/usr/local/lib/python3.7/dist-packages/PIL/TiffImagePlugin.py:788: UserWarning: Corrupt EXIF data.  Expecting to read 12 bytes but only got 10.
  warnings.warn(str(msg))
/usr/local/lib/python3.7/dist-packages/PIL/Image.py:932: UserWarning: Palette images with Transparency expressed in bytes should be converted to RGBA images
  "Palette images with Transparency expressed in bytes should be "
/usr/local/lib/python3.7/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning: Possibly corrupt EXIF data.  Expecting to read 12 bytes but only got 10. Skipping
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.7/dist-packages/PIL/TiffImagePlugin.py:788: UserWarning: Corrupt EXIF data.  Expecting to read 4 bytes but only got 0.
  warnings.warn(str(msg))
326/326 - 2221s - loss: 1.1329 - accuracy: 0.6793 - val_loss: 0.7237 - val_accuracy: 0.7862
Epoch 2/10
326/326 - 2208s - loss: 1.0022 - accuracy: 0.7145 - val_loss: 0.6959 - val_accuracy: 0.7964
Epoch 3/10
326/326 - 2217s - loss: 0.9550 - accuracy: 0.7227 - val_loss: 0.6824 - val_accuracy: 0.7975
Epoch 4/10
326/326 - 2212s - loss: 0.9412 - accuracy: 0.7267 - val_loss: 0.6749 - val_accuracy: 0.7988
Epoch 5/10
326/326 - 2213s - loss: 0.9354 - accuracy: 0.7267 - val_loss: 0.6693 - val_accuracy: 0.8005
Epoch 6/10
326/326 - 2200s - loss: 0.9148 - accuracy: 0.7299 - val_loss: 0.6664 - val_accuracy: 0.8016
Epoch 7/10
326/326 - 2200s - loss: 0.9179 - accuracy: 0.7325 - val_loss: 0.6628 - val_accuracy: 0.8043
Epoch 8/10
326/326 - 2180s - loss: 0.9140 - accuracy: 0.7335 - val_loss: 0.6592 - val_accuracy: 0.8057
Epoch 9/10
326/326 - 2192s - loss: 0.9040 - accuracy: 0.7399 - val_loss: 0.6559 - val_accuracy: 0.8060
Epoch 10/10
326/326 - 2162s - loss: 0.8952 - accuracy: 0.7408 - val_loss: 0.6542 - val_accuracy: 0.8064
```

**5. Module wise screenshots:**

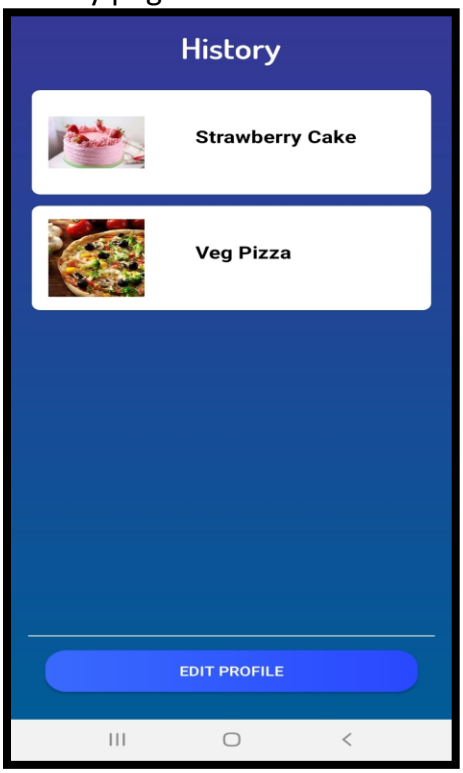
- Login/signup page:



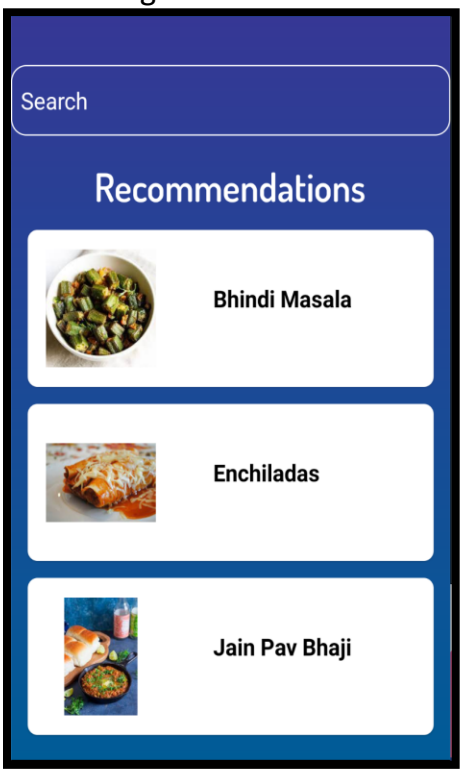

- Upload page:

- History page:

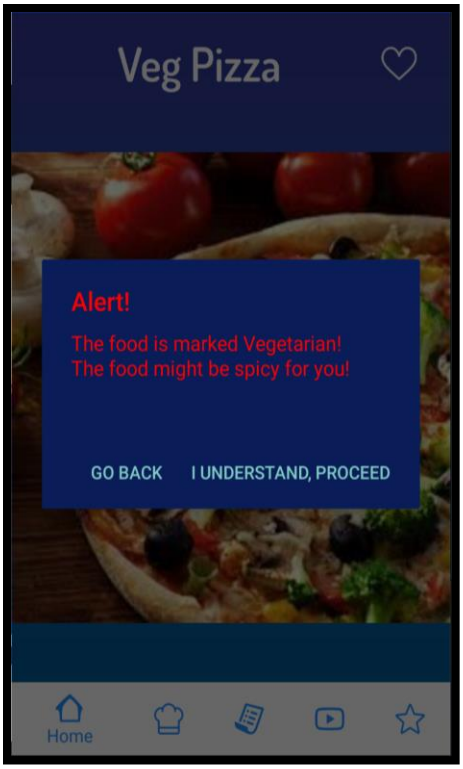



- Edit Profile Page

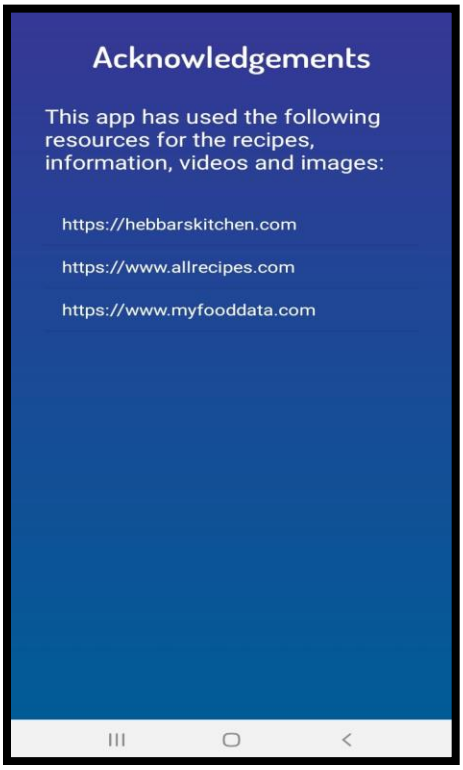
- Search Page:



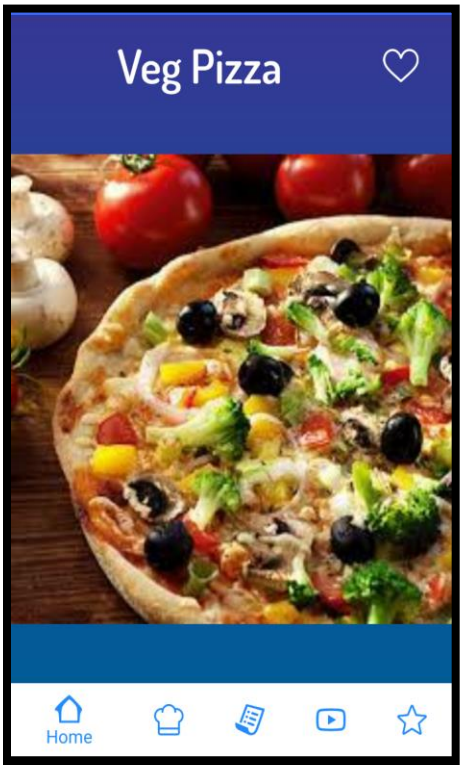

- Alerts

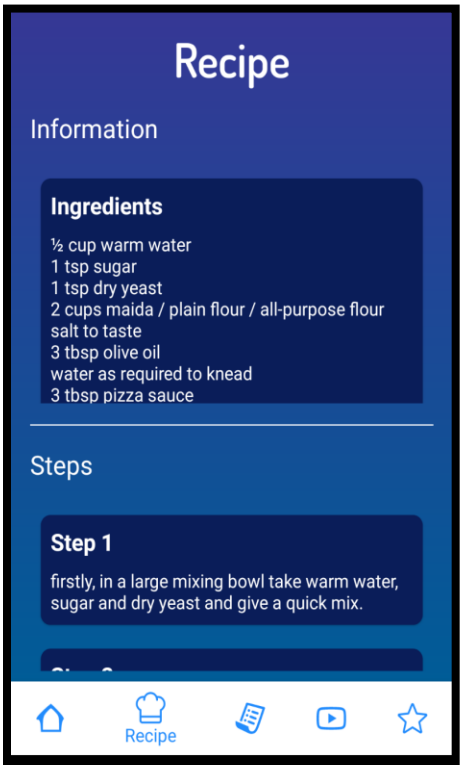


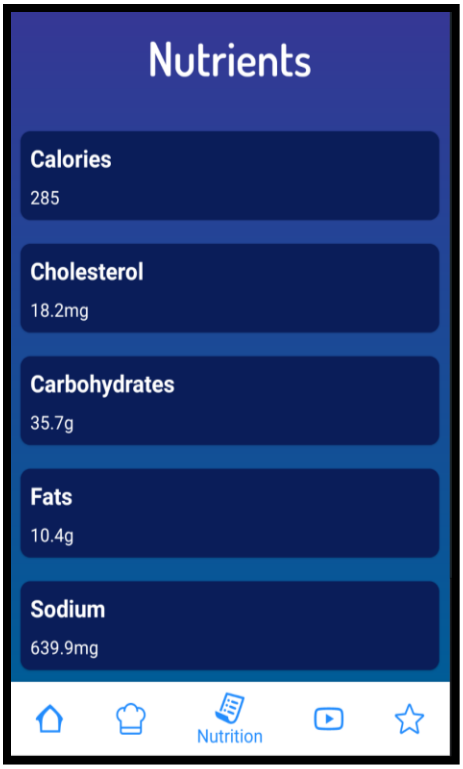
- Acknowledgements:

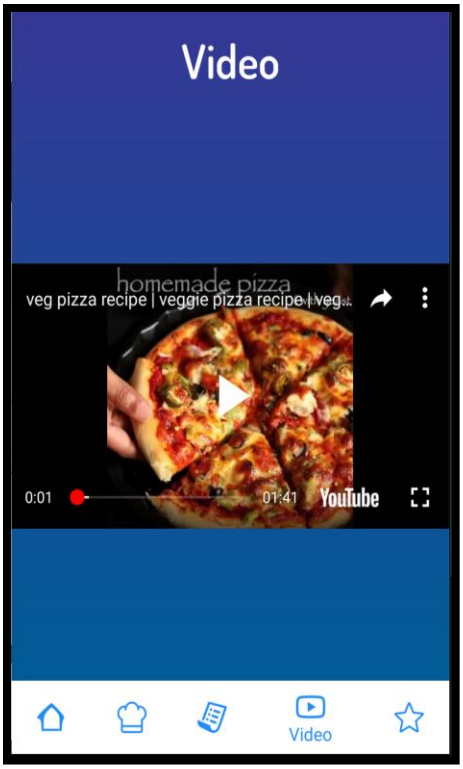


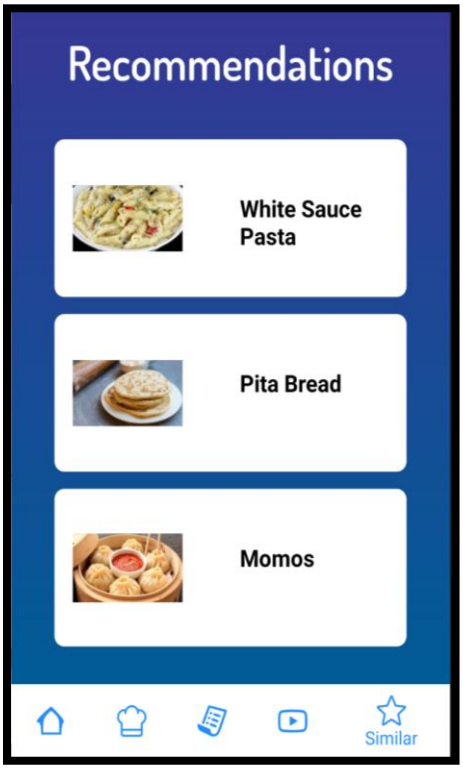

- Dish Information Module




Nutrients
Calories
285
Cholesterol
18.2mg
Carbohydrates
35.7g
Fats
10.4g
Sodium
639.9mg
Nutrition


Video
homemade pizza
veg pizza recipe | veggie pizza recipe | veg...
0:01
01:41
YouTube
Video


Recommendations
White Sauce Pasta
Pita Bread
Momos
Similar

## VIII. CONCLUSION AND FURTHER WORK

We have successfully implemented a food recognition app with various other features making it an all-in-one food app. All the objectives that were outlined in the beginning have been completed.

Through this project we gained hands on experience on making Android apps, image classification, creating REST API, hosting Flask apps on platforms like Heroku, Azure.

Future work would include increasing accuracy of the ML models and the number of food classes. Further we plan to identify multiple dishes from a single image and the exact ingredients from the dish.

## IX. REFERENCE RESEARCH PAPER/LINKS

[1] M. Jiang. “Food Image Classification with Convolutional Neural Networks”, Stanford CS230, Mar 2020

[2] D. Attokaren, I. Fernandes, A. Sriram, Y. Murthy, S. Koolagudi. “Food classification from images using convolutional neural networks”, in TENCON, Malaysia, 2017, pp.2801-2806

[3] Firebase (2021) Documentation [Online]. Available: https://firebase.google.com/docs

[4] Retrofit (2021) Documentation [Online]. Available: https://square.github.io/retrofit/2.x/retrofit/

[5] Glide (2021) Documentation [Online]. Available: https://bumptech.github.io/glide/

[6] Dr. Droid (2017) How to Implement YouTube Player Fragment in Android App [Online]. Available: https://www.androhub.com/implement-youtube-player-fragment-android-app/

All the information for the dishes has been taken from the following websites:

1. Hebbars Kitchen (2020) [Online]. Available: https://hebbarskitchen.com
2. All Recipes (2021) [Online]. Available: https://www.allrecipes.com
3. My Food Data (2021) [Online]. Available: https://www.myfooddata.com

The images used for training the models have been taken from the following:

1. Indian Food Classification Kaggle (2021) [Online]. Available: https://www.kaggle.com/l33tc0d3r/indian-food-classification
2. Food Images Kaggle (2021) [Online]. Available: https://www.kaggle.com/kmader/food41
3. Google Images (2021) [Online]. Available: https://images.google.com/
4. A Benchmark Dataset to Study the Representation of Food Images(2021) [Online]. Available: https://iplab.dmi.unict.it/UNICT-FD889/