

# Project Title : Nutrients Detection

**CP301 Development Engineering Project**  
**Department of Computer Science & Engineering**  
**Indian Institute of Technology Ropar**

Team number - 18

Team members - Divyanshu (2017csb1074)  
Adarsh Kumar (2017csb1064)

Project mentors - Dr. Ramanathan  
Dr. Puneet Goyal



# Previously chosen projects

Initially, we selected **Garbage Management** as our project which would help the authorities quickly know which area is polluted and needs immediate attention. That project was dropped because some other team was working on a similar project.

We also suggested some other projects like **Air Water Turbidity Measurement** and **Petrol/Diesel Refill Reminder and Suggestion** apps, but we could not find suitable mentors for these.

Finally, just before mid semester, Dr. Ramanathan suggested us to work on Nutrients Detection.

# Problem description

In today's busy world, people ignore to take proper diet and often lack essential nutrients in their body due to same routine intake. With one sixth of the global population residing in India, **one third of about two billion people suffering from vitamin and micronutrient deficit are in India** ([source](#)). Not only poverty, but a busy lifestyle is also an important factor in this problem.

Thanks to Dr. Ramanathan for suggesting this problem.

# Problem description (continued)

Micronutrients (vitamins and minerals) are required in small quantities and responsible for vital functions of the human body. Logically, they should be addressed easily and on a priority basis. The facts are, however, contrary. As a result, micronutrient malnutrition has been a persistent problem in India, and as the recent data suggest, some forms of micronutrient malnutrition are reaching their peak in the present century ([source](#)).

# Possible Solutions and Constraints

The problem of listing nutrients in a meal is very challenging on its own since there might be variations in how one makes a dish. So instead of detecting exactly how a dish was prepared we assume that the dish was prepared the most common way. We **detect the food** first (food classification), then provide nutrients in the most common way the dish is prepared.

For example, if you search nutritional information of samosa, the values you get will show the values from the most common way a samosa can be prepared.

There already exists a similar app ([link](#)) but it exists in the context of American food. As per our investigations, we are the first ones to offer a solution for the Indian food.

# Proposed Solution

Our solution tries to **estimate the nutrient intake of a person by recognising the food** image that he takes on the mobile app which will indicate what he lacks in his diet. This will help maintain a healthy life and in some cases may even avoid deficiency diseases or obesity.

We created a dataset of 29 classes (indian food like butter chicken, samosa, idli, masala dosa, etc) with 500 images per class (downloaded via `bing_image_downloader` library). All sources have been noted.

Thanks to Dr. Puneet for suggesting to create a new dataset specialized for indian food.

# Proposed Solution (continued)

**Quantity is not estimated**, infact different **standard consumption sizes are provided** as a response, and the user can select how much did he consume.

The classification problem was trained on various top performing CNN architectures - **Xception, VGG16, ResNet50V2** and **MobileNetV2** - with few extra layers at the top of each architecture. Each of the architectures was evaluated using **5-fold cross validation** whose results are provided in further slides. Transfer learning was used for each of the models pretrained on the imagenet dataset.

Thanks to Dr. Puneet for suggesting k-fold cross validation for model evaluation

# Proposed Solution (continued)

The value of hyperparameters were chosen for each model by experimentation on a simple 80-20 train-validation split. The hyperparameters that worked the best were chosen for the 5-fold cross validation training. VGG16 required a lot of computing resources, so training was stopped earlier.

**MobileNetV2** showed the best results with an **average accuracy of 93.0100691318512%** across the 5 folds. Out of the 5 trained models of MobileNetV2, the third one was finally selected for the application due to its high test accuracy.



# Technologies used

- Django and Django Rest API for creating nutrients detection API
- Keras and Tensorflow for image classification
- Flutter for app development

Please visit this [link](#) and ask for access. This google folder contains all the source code and other files.

# Xception (test scores)

Set	Loss	Accuracy
1	0.29217585921287537	91.75577759742737%
2	0.2618049085140228	91.54881238937378%
3	0.24284081161022186	92.7511215209961%
4	0.2937113344669342	90.89027047157288%
5	0.29885581135749817	91.65229201316833%

**Average Accuracy : 91.71965479850769% +- 0.5978012531895914%**

# VGG16 (test scores)

Set	Loss	Accuracy
1	0.5553471446037292	82.68368244171143%
2	0.5747872591018677	82.47671723365784%
3	0.5512798428535461	82.36106038093567%
4	0.5539462566375732	83.160799741745%
5	0.5518713593482971	81.92480206489563%

**Average Accuracy : 82.52141237258911% +- 0.40467321608588463%**

# ResNet50V2 (test scores)

Set	Loss	Accuracy
1	0.24992984533309937	92.48016476631165%
2	0.2994503676891327	91.51431322097778%
3	0.3022822141647339	91.37038588523865%
4	0.3201294243335724	90.89027047157288%
5	0.29601043462753296	90.92790484428406%

**Average Accuracy : 91.436607837677% +- 0.5756210194161737%**

# MobileNetV2 (test scores)

Set	Loss	Accuracy
1	0.25249364972114563	91.65229201316833%
2	0.2428516298532486	92.72162914276123%
3	0.20994901657104492	93.82119178771973%
4	0.2543746829032898	93.30573081970215%
5	0.2365763783454895	93.54950189590454%

**Average Accuracy : 93.0100691318512% +- 0.7698155408648589%**

# Nutritional Values

All nutritional values of food items were taken from a reputed web source - <https://www.fitbit.com/foods> and each of the link was noted down (will be provided in the source code).

Fitbit, Inc. is an American company headquartered in San Francisco, California. Its products are activity trackers, smartwatches, wireless-enabled wearable technology devices that measure data such as the number of steps walked, heart rate, quality of sleep, steps climbed, and other personal metrics involved in fitness ([wiki](#)).

After copying the nutritional values from the table given in the website, it was converted to json string so that it can be used directly in our django application.

# Challenges faced

- Lot of computation required, even google colab has many limitations so had to use different accounts
- Slow internet speed and large size of dataset (5.38 GB)
- Initially used food-101 dataset which required 10 GB downloading, but finally decided to make own dataset for Indian food
- Had to learn concepts of deep learning and other related concepts like k-fold cross validation

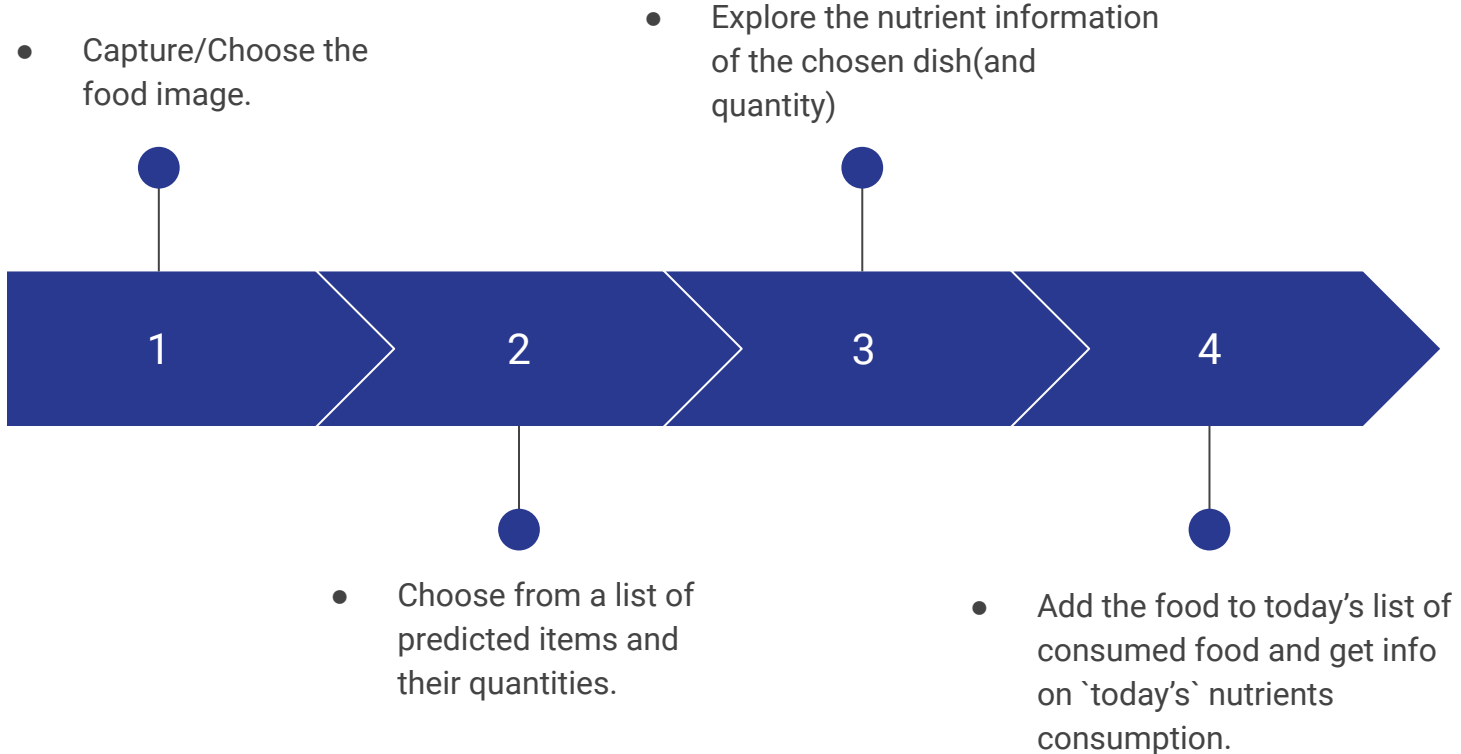
# Mobile Application

- **ANDROID**
- **iOS**

**App Size: <6 Mb**

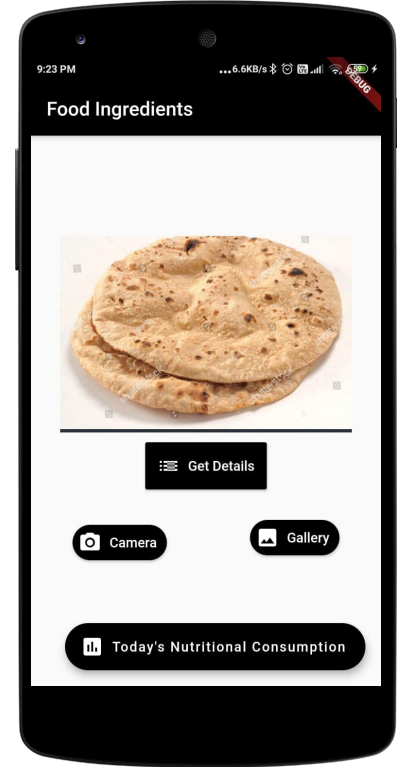
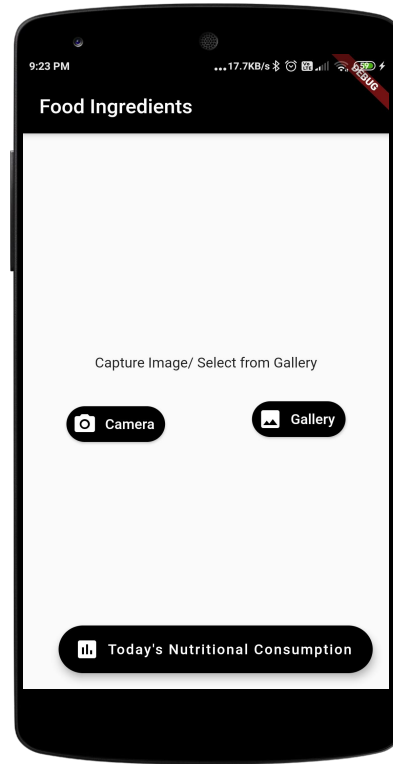


# Using The Application



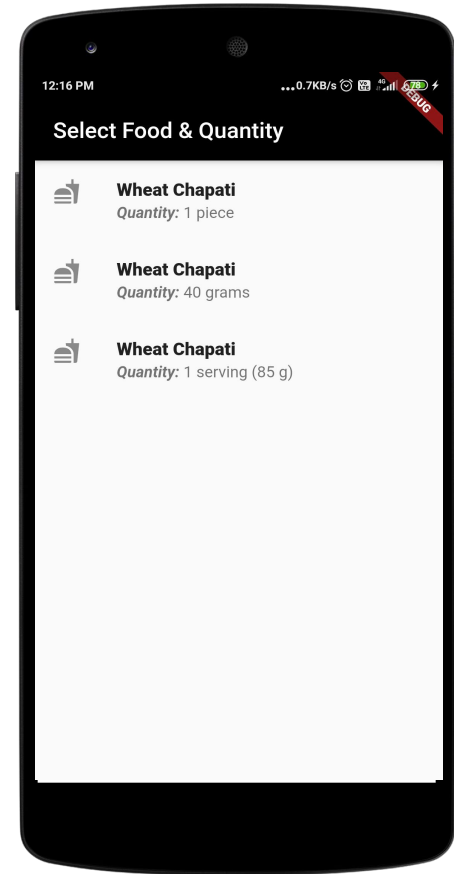
# User Interaction & User Experience- 1

- User gets two options to upload the food image:
  - Capture image at the moment.
  - Pick image from the gallery.
- After image selection, user clicks on the 'Get Details' button.



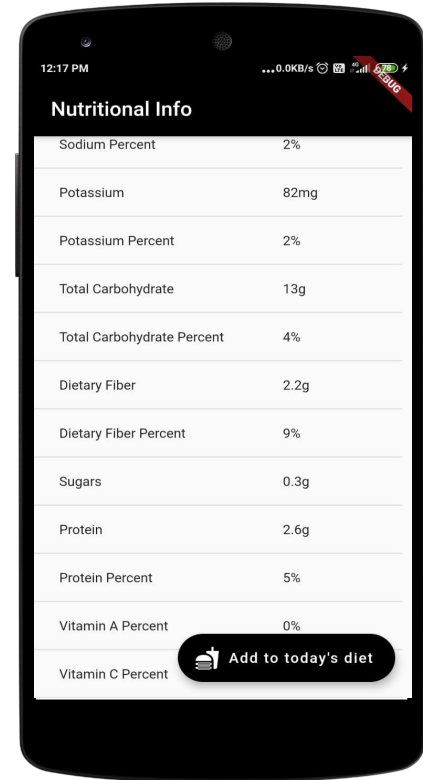
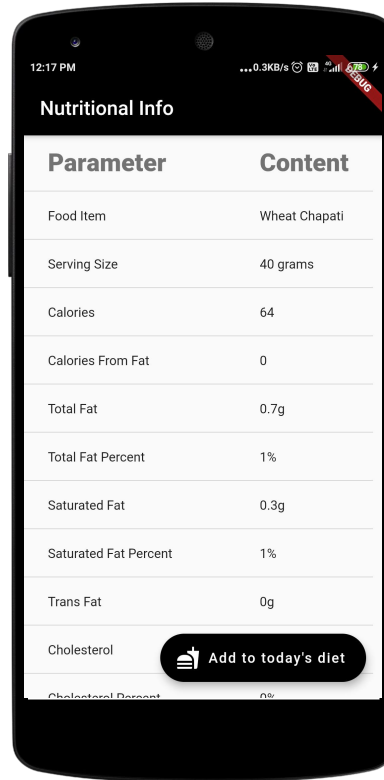
# User Interaction & User Experience- 2

- On this screen, we provide a list of the best of our predictions, along with some standard consumption units.
- The user is supposed to tap on the item with closest consumption size.



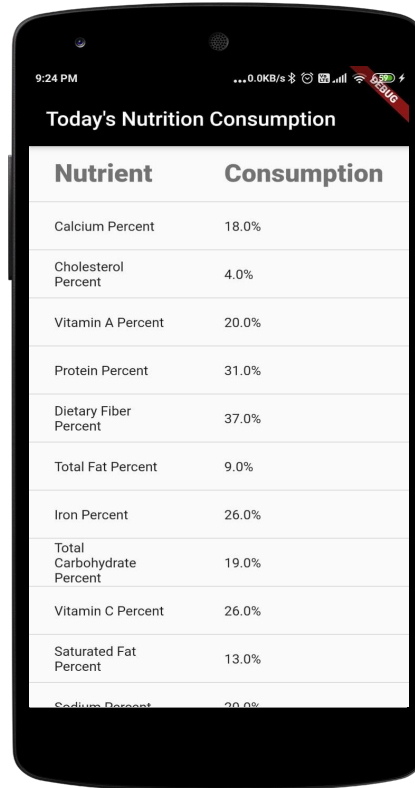
# User Interaction & User Experience- 3

- This screen lists all the nutrient information contained in that particular diet.
- The user can go back and select some other dish/dish quantity from here.
- Upon being satisfied with the selection, the user can add the selection to today's diet by pressing the button at the bottom.



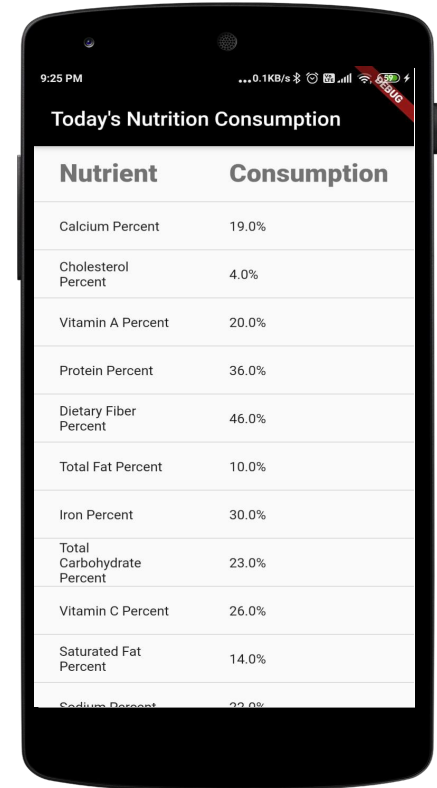
# User Interaction & User Experience- 4

- On this screen, the user gets an idea about:
  - how much nutrients has been consumed by him for the day
  - How much more, from each of the nutrient, has he to consume for the day.
- The stored information refreshes back to 0 every day.



9:24 PM ...0.0KB/s 5G

Nutrient	Consumption
Calcium Percent	18.0%
Cholesterol Percent	4.0%
Vitamin A Percent	20.0%
Protein Percent	31.0%
Dietary Fiber Percent	37.0%
Total Fat Percent	9.0%
Iron Percent	26.0%
Total Carbohydrate Percent	19.0%
Vitamin C Percent	26.0%
Saturated Fat Percent	13.0%
Sodium Percent	22.0%



9:25 PM ...0.1KB/s 5G

Nutrient	Consumption
Calcium Percent	19.0%
Cholesterol Percent	4.0%
Vitamin A Percent	20.0%
Protein Percent	36.0%
Dietary Fiber Percent	46.0%
Total Fat Percent	10.0%
Iron Percent	30.0%
Total Carbohydrate Percent	23.0%
Vitamin C Percent	26.0%
Saturated Fat Percent	14.0%
Sodium Percent	22.0%

# Challenges Faced

## Scrollable for Tabular Data

- The tabular data (the one showing parameters and content) was hard to be viewed in scrollable form.

## Store Data

- Storing data without getting the user into complexities such as login (JWT or Google Login) seemed difficult at first.
  - Working with `Futures` (async...await) to get the local device storage instance was a problem as most of the widgets don't support asynchronous data loading.
-

# Challenges Faced ...contd

## String Manipulation

- Most of the data was in string form with 'mg', 'g', 'grams', etc appended to it. Tried to extract just the numbers from generic data for calculation(addition, etc) using regex. Achieved some success but still needs improvement for better displaying of data.

---

Thank you