# SmolKat: A Smart Kitchen Assistant

Authors: Elena Gong, Nanxi Li, Yang Yue: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—SmolKat is a recipe recommendation system that is capable of suggesting recipes that can be made with food items available and satisfy the requirements of the user. We incorporated computer vision and speech recognition technology into common recipe recommenders. Our prototype can scan a surface with up to 10 food items and provide up to 30 recipes that only need the food items available. Additionally, instead of by typing and scrolling, SmolKat can interact with the user through audio. SmolKat can listen for user requirements and provide step by step cooking instructions. SmolKat's response latency is less than 1 second.**

***Index Terms*—Computer Vision, Feature Extraction, Food Ingredients, Recipe Recommendation, Speech Recognition.**

## 1 INTRODUCTION

SmolKat is motivated by the concept of reducing food waste through high refrigerator throughput. We've noticed that oftentimes food waste is unintentional, and often caused by spoiling ingredients that a user cannot figure out how to use or ingredients that are hidden. This food waste contributes to higher weekly shopping costs and frustration.
SmolKat is a recipe recommendation system that helps reduce food waste by directing users to ingredients and providing useful recipes. We aim to help those who struggle with finishing the food in their fridge because they either constantly forget about the food at the back of their fridge or have a difficult time using all the food they have because they don't know an appropriate recipe. Currently there are smart fridges that have integrated computers to track items and generate shopping lists, but SmolKat strives to be more than simply an item tracker by providing recipes.

Current recipe recommendation systems only suggest recipes based on the preferences of the users, resulting in recipes that require material that's not available. SmolKat solves this by scanning inside the fridge or cabinet, and automatically identifies what foods are available, and then recommend recipes based on user's requirements. SmolKat must be able to scan a surface with up to 10 items, take in user vocal inputs for recipe preference, recommend up to 30 recipes and provide step-by-step instruction. SmolKat will respond to user vocal input within 1 second at 90% accuracy. Additionally, SmolKat will be able to update the database from image recognition within 5 seconds with an object identification accuracy of 95% to ensure user happiness.

## 2 DESIGN REQUIREMENTS

### 2.1 Image Recognition

The main requirements for the image recognition is that the items on the layer needed to be captured, classified and counted correctly. Photos will be taken when the user opens the fridge. The photos will be cropped into pieces along the led grids, and each cropped image will become the input to the image recognition API, Google Vision to get a classification result. To test our model, for which we require 95% accuracy, we will setup a software testbench which will use pre-generated data using the specific camera that we will be using. We chose this accuracy number since recipes require multiple ingredients, so mislabeling an object may lead to significant errors in recipe recommendation.

### 2.2 Speech Recognition

The main requirement for speech recognition is to correctly parse the user's audio input in a manner that the system can respond to. This involves turning the user's audio into text before matching against templates for the request and user preferences. The audio will be stored in real-time before being processed via NVIDIA's open-source NeMo Library. We require 90% accuracy on our testbench, which will be run with prelabeled audio files collected using the specific microphone model that we selected.

### 2.3 Recipe Recommendation

We assume that the user's pantry has ample shelf stable food items.

The first requirement is variety of recipes. The recipes recommended should cover most of the food items available in the least amount of recipes, so that every item in the fridge is possible to be used. For example, if we offer two recipes of noodles that will both only use tomato and eggs, then we should only keep one of them. Another aspect of variety is that we want to make sure that if the user does have a lot of stock of some items, we still expect that users will not get same recipes continuously. We will check for these cases with a software testbench, with success being the user never receives two similar recipes in a row.

After reading peer reviews from the presentation, we decided to add a requirement to keep track of the freshness of food in the fridge. If we have a box of eggs 10 days ago and a piece of chicken breast that was added to the fridge yesterday, then we should choose some recipes that will use eggs (and chicken) rather than only chicken. This will help reduce the amount of spoilage in the fridge.

# 3 ARCHITECTURE OVERVIEW

Our system architecture consists of multiple components, all of which sit on the NVIDIA Jetson Nano hardware system. This allows us to process everything in real-time onboard without incurring the latency and power costs of having to access the cloud. The individual components of our system include Image Input processing, Audio translation and parsing, Recipe Recommendation and Storage, LED grid and guidance system, and the Text to Speech Synthesis.

## 3.1 Image Input Processing

The image input processing is where we do our Object Recognition to detect what items are in the refrigerator. At regular intervals, we will capture an image of the refrigerator, run object recognition, and output a list of items recognized in the refrigerator to the next component down the path, which is the Recipe Recommendation and Storage.

## 3.2 Audio Translation and Parsing

The input audio from the user should contain commands in a format such as "SmolKat, recommend me a vegan recipe", where "SmolKat" is the trigger word, and ""vegan" is the user preference. This module will take the input audio when the trigger word is used, and will first convert the audio to text, before matching with a standard template to extract user preferences. It will output a command to the Recipe Recommendation and Storage unit with the parsed out user preferences.

## 3.3 Recipe Recommendation and Storage

The Recipe Recommendation system contains the Recipe database, which holds the preexisting recipes – this will be implemented as a JSON file since the number of recipes are small. The Storage also contains the list of available items, tagged with their entry time and location, which guides the recipe recommendation system in selecting a specific recipe. This system outputs to the Text to Speech Synthesis unit, which recipe was selected, and also outputs to the LED grid and guidance system which locations to highlight.

## 3.4 LED Guidance System

The LED subsystem serves a simple purpose, which is to drive the LED grid correctly given a specified location. The LEDs are laid out in a 3x3 grid across our testing system, and will act as the guide to the user.

## 3.5 Text to Speech Synthesis

The Text to Speech Synthesis is how the system interacts with the user. It converts the input recipe into audio and can also perform tangential functions such as emailing the user a copy of the recipe.

## 3.6 Sub-Component Flows

Overall, both the Image Input Processing and Audio Translation and Parsing sub-components act as intermediaries between the user and the Recipe Recommendation system. They take in user provided requests, and images collected from sensors, and convert them into software structs usable by the Recipe Recommendation system.
The Recipe Recommendation system provides the brains of our system, storing both potential recipes and also the collected items and their labels (timestamps, quantity, etc.). This sends the selected recipe to the system's output subcomponents, namely the LED grid and Text to Speech Synthesis, which then output the request to the user. A full figure our of systems is displayed as Fig 4 in Appendix.

# 4 DESIGN TRADE STUDIES

## 4.1 Image Recognition

There are many popular image recognition APIs available on the market, and we have candidates of AWS Rekognition, Google Vision API, IBM Watson, Microsoft Azure. We compared these APIs based on the accuracy and false positives of recognition results. From a previous study[1], the Google Vision Image Recognition Engine's Accuracy was the highest among the candidates, which reached 81.7% on 2000 images of four categories, Charts, Landscapes, People, Products. This API also has the lowest false positives, which means that it is unlikely that we will offer some recipes that includes some ingredients that the user does not have in the fridge.

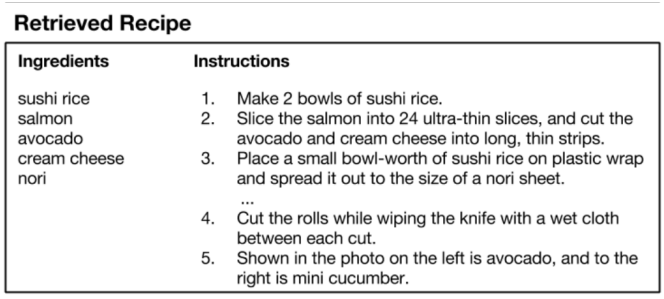
## 4.2 Speech Recognition

There are various speech recognition tools on the market, with all major cloud providers (Google, Microsoft Azure, AWS) offering some sort of speech to text service. However, one issue we ran into was the need to be offline, using pre-trained neural nets, that could also make use of the hardware that we had selected (NVIDIA Jetson Nano). This meant that we were limited in options, and ended up testing out the NVIDIA open source NeMo library, which has been accurate on the sample audio that we have piped in. Additionally, this library can take advantage of the onboard GPU.

## 4.3 Recipe Recommendation

For the cooking recipes, we chose Recipe1M+, a new large-scale, structured corpus of over one million cooking recipes[2]. This dataset is appropriate for our project because the recipes are clearly formatted and we can filter then by the ingredients they need easily; also, we have millions of options from this recipe, which will help us build

our choices of 30 recipes to meet all requirements. A sample of recipe is shown at the end of this section.
Since we want to store the information including the timestamp of the items and the latest served recipe, we will be using SQLite database to store the data. In previous section, we will use Google Vision API in Python, and Python also has a SQLite Library that helps us store the information in relational tables. Also, since we are not working on cloud deployment, we will not consider other complex databases, like MySQL, PostgreSQL, etc.



**Retrieved Recipe**

| Ingredients | Instructions |
|---|---|
| sushi rice<br>salmon<br>avocado<br>cream cheese<br>nori | 1. Make 2 bowls of sushi rice.<br>2. Slice the salmon into 24 ultra-thin slices, and cut the avocado and cream cheese into long, thin strips.<br>3. Place a small bowl-worth of sushi rice on plastic wrap and spread it out to the size of a nori sheet.<br>...<br>4. Cut the rolls while wiping the knife with a wet cloth between each cut.<br>5. Shown in the photo on the left is avocado, and to the right is mini cucumber. |

Figure 1: Sample of Recipe

### 4.4 Instruction and Guidance

We want to provide guidance and instructions for those who want not only just a recipe, but also the guidance on how to make the dish from the recipe. We will provide LED guidance so that the user can see where the items they need are instead of spending a long time searching for each of the ingredients in the fridge. We considered using single sparse LEDs to point out where the ingredients are. Although this is easier to implement, this methods needs many connections via the GPIO pins since each LED light needs its own control. This implementation is not scalable, because we have limited amount of GPIO pins on the NVIDIA Jetson Nano. Instead, We will use a single LED strip that has serially connected LEDs to make a LED grid that highlights where the items are, so that the only connection it needs with NVIDIA Jetson Nano is one GPIO pin. This will result in a significantly more difficult LED light up algorithm, but it allows us to have a dense LED arrangement, so that the user can still see the light even when the surface is relatively densely packed. We also want to provide instructions of the steps of the recipe. We considered implementing it in 3 ways: sending an email with recipe details to the user, delivering the entire recipe vocally in one go, and delivering the recipe vocally step by step. We decided to move on with the third implementation because this is the most intuitive way to follow a recipe in the kitchen. This method does not require the user to check their phone while cooking and also does not require the user to remember an entire recipe before they start cooking.

## 5 SYSTEM DESCRIPTION

### 5.1 Hardware Peripherals

The hardware peripherals we need include the camera to take pictures of the food items, the microphone and speaker for interacting with the user vocally, the LED grid that directs user to the ingredients they need for the recipe, and the NVIDIA Jetson Nano as the processor that process all inputs and outputs.

The camera, microphone and speaker are off-the-shelf, since our product do not have a specific requirement for these components. The NVIDIA Jetson Nano developer kit was chosen since it is light weight, powerful, and has multiple USB ports to connect the peripherals. The camera, microphone and speaker all use USB connection to NVDIA Jetson Nano. The LED grid will be made out of LED strips that are connected serially. This will be a 3x3 grid attached onto a 12 inchx12 inch acrylic board. It will be connected to NVIDIA Jetson Nano via the GPIO pins.

### 5.2 Image Recognition

For the Image Recognition, we decided to use package Google Vision API. Therefore, our focus for this part will be image processing and managing the input/output.

We use Camera IMX219-160 to take the photos for items on the layer. The reason we chose this camera is that it is compatible with NVIDIA Jetson Nano, and has a high resolution ($3280 \times 2464$). Since we want to use one camera to cover an entire layer with led grids, the wide angel of view (160 degrees) that this camera offers is also ideal for our purpose.

The photos will be taken by the camera every time the user opens the fridge. Since we will use acrylic board in our product to simulate the inside of a fridge, photos will be taken every time there are items changed on the board and will be controlled by us. Since there are 3x3 grids on the board, with each type of item in one grid, the image will be cropped into 9 pieces accordingly. The cropped images will be the input to the API, and API will return a response object with attributes including the classification of the input image. Since this attribute will has the format of a list, with the first index with highest probability(confidence) for that class, we will only take the first element in the list as the recognition result. After getting the result, we will match it to our list of items to see if this word is in the list. If it is found in the list, the result will be returned as it is; if the result is not found in the list, it indicates that whether this grid is empty or the result is wrong. In this case, NULL will be returned and this result will not be used for recipe recommendation because it does not include useful information. Since we expect high accuracy of image recognition, for most of the cases, NULL just means the grid is empty. We will set our list of items to be 9 items for now, since the grid is designed in 3x3, and the following is a table of candidate items. These items are selected based on the recognition results of Google Vision

API on images of these items available online.

| Type | Ingredient Candidates |
|---|---|
| Fish | Salmon, Tuna, Shrimp, Squid |
| Meat | Beef, Chicken, Sausage |
| Vegetables | Mushroom, Potato, Eggplant, Carrot, Tomato, Cucumber, Green onion, Onion, Broccoli |
| Fruit | Oranges, Apples, Bananas, Strawberries, Pineapple |

Figure 2: Table of Candidate items

### 5.3 Speech Recognition

For Speech Recognition, we decided to go with NVIDIA NeMo, by using a pre-trained neural network to handle audio to text. The focus of this part will be focusing on the technical implementation details and hardware.
For the microphone, we selected KISEER USB Mini, because it is directly compatible with the Jetson Nano. Additionally, we also selected this due to its price point, since we care significantly about the cost of our system.
Our system first starts by sampling at fixed intervals for noise. When it hears noise, it will record the incoming audio into a wav file format. Then it will use the neural net to convert the wav file into text output. From this, we will check if the user had given a command or it was background or conversational noise. If the user had given a valid command, we then try to match the text into a command format and parse out the user requirements. It will send this packet to the Recipe Recommendation System. Overall this system is stateless, and simply serves to translate the audio into a recognizable form with high accuracy.

### 5.4 Recipe Recommendation System

For recipe recommendations, one of the main tasks is find the appropriate recipe from the Recipe1M+ dataset. We will filter the recipes and only keep the recipes that include our list of items from previous section, and order them by the number of items on the list they use in the recipes. The next step will be excluding the ones that require other perishables that are not on our list. We will have a list of perishables that exclude the items we choose to detect, and use that to filter out the related recipes. Then from the selected recipes, we will choose 30 recipes that differ in cooking style and combination of ingredients as much as possible. With the 30 recipes decided, we will manually tag these recipes with the requirements we plan to get from the user, including ""vegan", ""non-dairy", ""dessert", ""main dish", etc. These tags will help us filter the recipes when user want to specify their needs.

The next task will be the main pipeline of the recommendation system: given the input from users, return a list of appropriate recipes. The input to the recommend system will be list of requirements and ingredients, in the format of [Food: item1, item2,...; Requirement: requirement1, requirement2, ...]. The system will use these keywords to filter the recipe lists and find a list of recipe that match all keywords. If no recipe is found, then it will return NULL. If recipes are found, they will be compared with the latest served recipe stored in the database. If that recipe is found in the list, it will be removed so that the user will not get the same recipe for lunch and for dinner on the same day. The last step will be reordering the recipes based on the timestamps of ingredients. The recipes that use older ingredients will be put at the front of the list. The reordered list will be the final output of the recommendation system and served to user.

### 5.5 User Interface

The entire process starts when the user open the fridge and put in the ingredients he/she just bought from supermarket. The camera will take a picture of the grid and the images will be cropped and classified. And the user will go back to the work until getting hungry and decide to prepare for a meal. The user will speak a sentence with the needs for this meal, in the format of ""I need a recipe for **3** people, **vegan**." Once the user says a sentence, the system will convert the speech to text and extract the important needs from the sentence. These needs, along with the ingredients stored in the database, will be sent to the recommendation part and get a output of a list of recipes. Then the recipe details will be delivered by speech to the user, and the LED grids corresponding to the ingredients needed for the recipe will light up to help user locate the goods they need for the wonderful meal.

Our user experience flow is shown in the following graph:

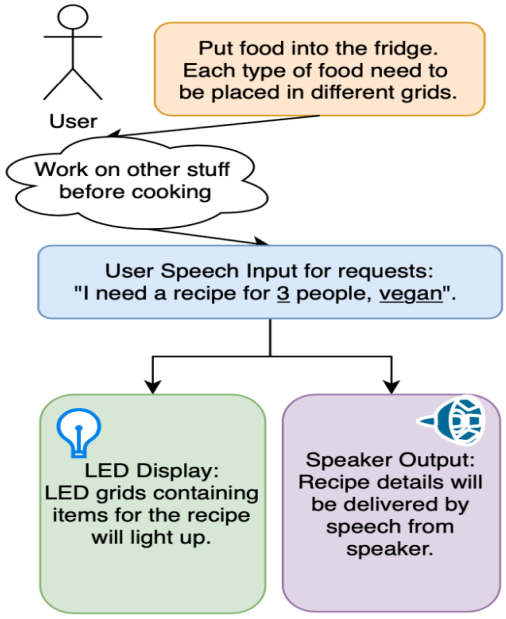



Figure 3: User Interface

An example of the user experience that a user will have with the system will be similar as the Fig 5 in Appendix.

# 6 TEST METRICS AND VALIDATION

## 6.1 Image Recognition

We care about the recognition accuracy, because we will recommend the recipes based on what ingredients we have. Therefore, we will test the image recognition accuracy by building a test dataset consists of photos that we take on the board we made, and also photos that we take in the real fridge. In this way, we will be able to evaluate the accuracy in accord with the environment that we are actually working with, and we can also measure how good it will be after transferring our product to a real fridge. The accuracy will be 90%, since we know from the Google Vision reports that the accuracy of classification on a four-class dataset reached 82%. In our system, we will only have 9 items, and we will also have clean background and no overlapping of items, which is more limited than the dataset used for that measurement, so we expect to get higher accuracy than the one in report.

## 6.2 Speech Recognition

We require a specific accuracy on speech recognition because we do not want the user to have to repeat commands, which would degrade the user experience. To measure test accuracy, we will first need to make sure we are testing for the appropriate hardware, since different microphones could have different ranges and capture different background noises. Thus we will make a dataset of audio input of real and fake commands, with varying user preferences using our selected hardware. Then our software testbench will run through the samples and check their respective outputs for our implementation, and make sure we can accurately separate the real commands from false commands with 90% accuracy.

## 6.3 Recommendation System

We want to test if the recipe output is corresponding to the input ingredient correctly. Therefore, we will build our own test cases, for each case it will be a list of requirements and ingredients, in the format of [Food: item1, item2,...;Requirement: requirement1, requirement2, ...], and we will see if the output recipe meets all these requirements. We want to make sure that the output is one hundred percent meet all requirements and only using fresh goods that are listed in the input.

## 6.4 LED Guidance System

We want to test if the correct LED grid will light up according to the selected recipe and the food arrangement. We want this LED highlighting to be correct for more than 95% of the time. We will be testing this by using 3 specific recipes and random arrangement of the food. A correct highlighting means that the grid highlighted need to contain (part of) the food item needed for the recipe.

## 6.5 System Latency

We care about the latency of entire system because we don't want the user to wait for their recipes and we hope that users will have smooth experience. The "entire system" we measure will be the time start from user finish the sentence of request to list of the recipe(s) is returned as an output. We are not counting the step-by-step speech instructions because that depends on the user's pace of cooking. We expect the latency to be less than 1 second, because the image recognition work will start as soon as the user close the fridge after putting newly bought ingredients. For the speech recognition part, the reports from NVIDIA indicates that the speech to text on Jetson Nano usually take milliseconds. For the recommendation system, the computation is not complicated(filtering, ordering, etc.)thus the latency can be seen as instant. Considering the data i/o across different parts of the system, we aim for 1 second for our project. The test method will be recording time from end of user input to start of system output. If the speed cannot be achieved, we will try to improve by code optimizations and train the models with different datasets to reach for the best results.

# 7 PROJECT MANAGEMENT

## 7.1 Schedule

Figure 6 below is the Gantt chart of our project. It includes a schedule with the division of tasks per person and over the whole team for the semester. Compared to the old version of schedule displayed in our proposal presentation, the schedule has been slightly changed on the general timeline that most tasks are moved behind one week due to the delayed arrival of materials, but we will be able to complete all tasks in time following this schedule as the initial version had extra time for final improvement.

## 7.2 Team Member Responsibilities

Elena will work on the recipe recommendation, feature extraction of recipes and support image recognition on the data processing. Nanxi is in charge of the Hardware and Integration, including building LED grids and integrating code of different parts on the board. Yang is responsible for the speech processing, and will focus on speech-to-text and text-to-speech functionalities.

Everyone will test their own parts, and everyone will work together on testing the entire system and improve based on the evaluation as a team.

## 7.3 Budget

The Fig 7 below is our bills of materials and budget. In general, we spent our budget on hardware components and raw materials to build the clear board with LED grids. In order to allow teammates work separately on our own tasks, we bought two pieces for each component. The additional set of components will not be wasted; we might use them for a second layer on the shelf to better imitate the situation of a real fridge. We have enough budget left for additional functionalities like WiFi module for an email copy of recipe to users.

## 7.4 Risk Management

### 7.4.1 Recipe Recommendation System

There are a few different potential risks for the recipe recommendation. The first risk is that with the selected items, it might be difficult to find appropriate recipes under the limited number of items. In order to handle this risk, we decided to assume that the user will have sufficient storage in pantry and thus we will have unlimited dry ingredients for the recipes. In addition, we believe that our product can also be applied to pantry, as the only change will be from recognizing perishables to cans/bottle items. Due to the limit time we will set this consideration as a stretch goal and hold the assumption for the convenience of implementation.

Another risk is that the system may recommend users same recipe continuously if they have a lot of stock of same items in the fridge. To handle this risk, the system will keep track of the latest recipe that was offered and accepted by the user. Since we expect to provide a list of available recipes to users, we will only offer the latest recipe again if there are no other qualified options; otherwise we will exclude this recipe and serve the rest.

### 7.4.2 Image Recognition

The Google Vision API that we decided to use is a pretrained model with high accuracy compared to other popular computer vision APIs that focus on classification and recognition. However, since we are focusing on limited items, we cannot guarantee that the API will perform very well with our limitations and our data. To handle this risk, we will take a set of photos with items on the board we build, crop these photos by ourselves into images for each item, and add these cropped photos to dataset to train the models. In this way, we will be able to see how our environment influence the image recognition. We will evaluate the accuracy to make sure that it still achieves about 80%.

### 7.4.3 Speech Recognition

The NeMo Speech Recognition model is pretrained and should be very accurate in terms of speech to text, since we are not doing special functions such as in Image Recognition. In the instance that the model is triggering too frequently, we will select a different trigger word, that is more unique than “SmolKat”, in a similar manner to how Amazon Alexa functions, by triggering on the hard X sound in “Alexa”, which is generally uncommon in the English vocabulary. At the worst case, if we cannot meet our accuracy, we can simply build in a query system, which will ask the user if they meant a specific request. The positive feedback from the user should fix any internal technical errors.

### 7.4.4 LED Guidance System

The risk of the LED Guidance System lays in recognizing that for each item, which grid do they belong to. It is easy when the items are perfectly aligned within each grid. However, if there is an item that spans multiple grids, we need the algorithm to decide which grid should be lightened up to highlight that food item. To handle this, we will have a deterministic algorithm to decide the grid to light up according to food arrangement. We will also focus on the accuracy when testing the LED light up algorithm.

# 8 RELATED WORK

There are many data-driven recipe recommendation system related works/papers available, and most of them are web application based or mobile applications. For example, one of the related work is a a cooking recipe recommendation system which runs on a consumer smartphone as an interactive mobile application [3]. The proposed system employs real-time visual object recognition of food ingredients, and recommends cooking recipes related to the recognized food ingredients. Because of visual recognition, by only pointing a built-in camera on a smartphone to food ingredients, a user can get to know a related cooking recipes instantly. The objective of the proposed system is to assist people who cook to decide a cooking recipe at grocery stores or at a kitchen. In the current implementation, the system can recognize 30 kinds of food ingredient in 0.15 seconds, and it has achieved the 83.93% recognition rate within the top six candidates.

Our strength lies in the part that we decided to keep everything on the board, without any requirement for WiFi connection. For these applications, the users will be at a loss if they do not have WiFi connection when they want to cook.

Also, our project allows user to add additional constraints on the recipe based on their personalized needs, like vegan or non-dairy. This makes our project a good option for people who need special accommodations for their diets.

In terms of convenience of usage for users, we are advantageous on the point that the camera will be able to collect the info of all items on the layer in one shot. Our system will crop the images into pieces and do the detection for

each picture. However, using the smartphone camera may require user to take photos separately for each item which makes the process more complex.

# References

[1] Eric Enge. *Image Recognition Accuracy Study*. 2019. URL: https://www.perficient.com/insights/research-hub/image-recognition-accuracy-study (visited on 02/20/2021).

[2] Javier Marin et al. *Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images*. 2019. arXiv: 1810.06553 [cs.CV].

[3] Keiji Yanai, Takuma Maruyama, and Yoshiyuki Kawano. "A Cooking Recipe Recommendation System with Visual Recognition of Food Ingredients". In: *International Journal of Interactive Mobile Technologies (iJIM)* 8.2 (2014), pp. 28–34. ISSN: 1865-7923. URL: https://online-journals.org/index.php/i-jim/article/view/3623.

## Appendix A

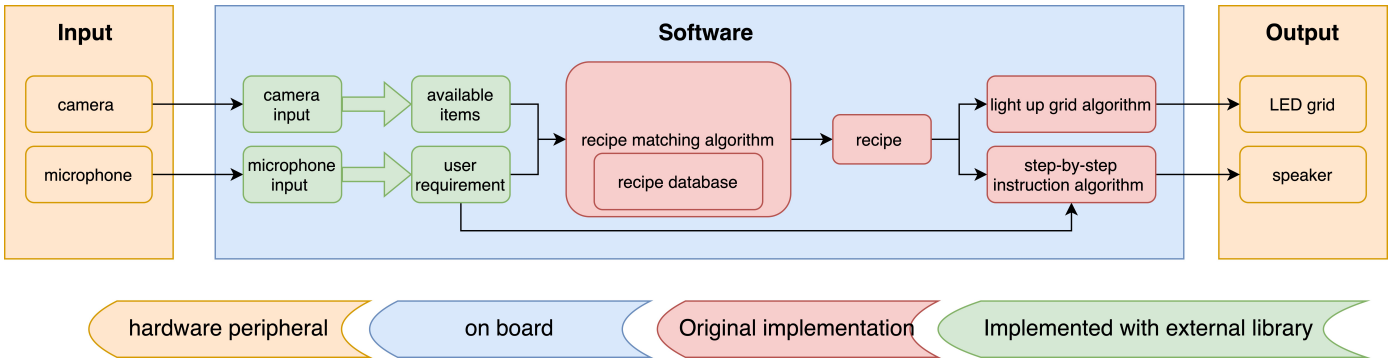



Figure 4: System Architecture

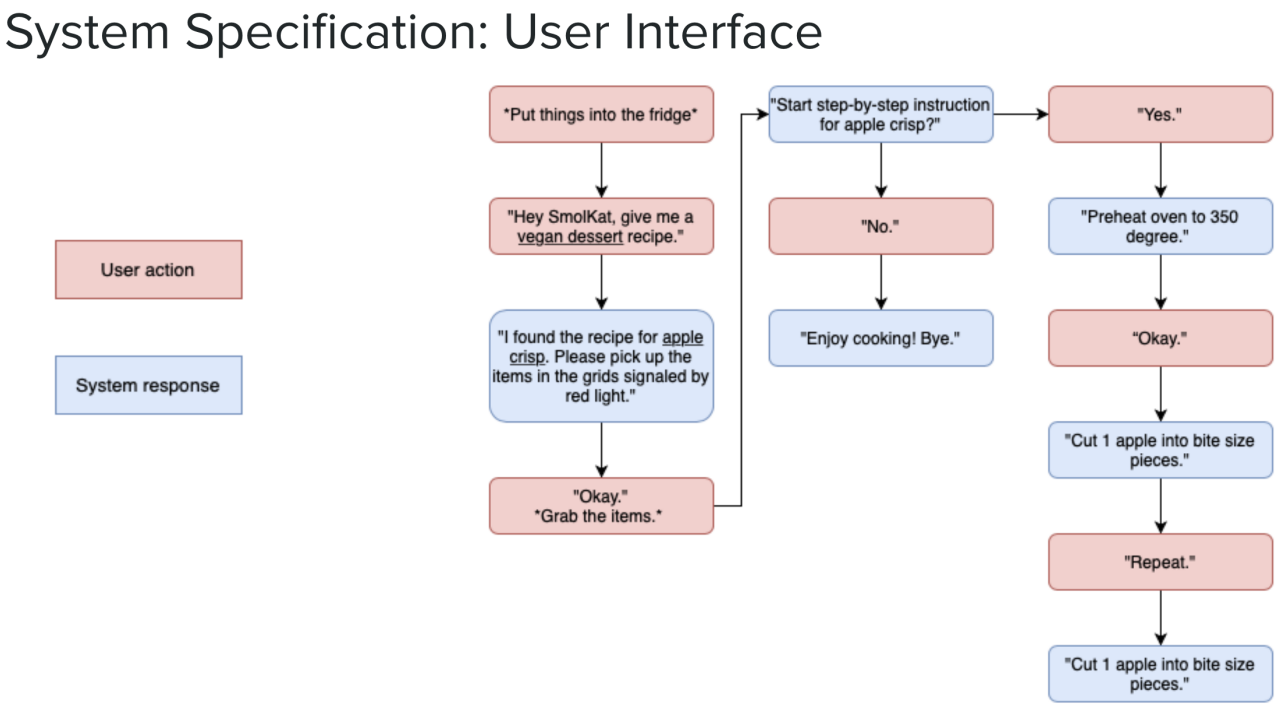



Figure 5: User Flow

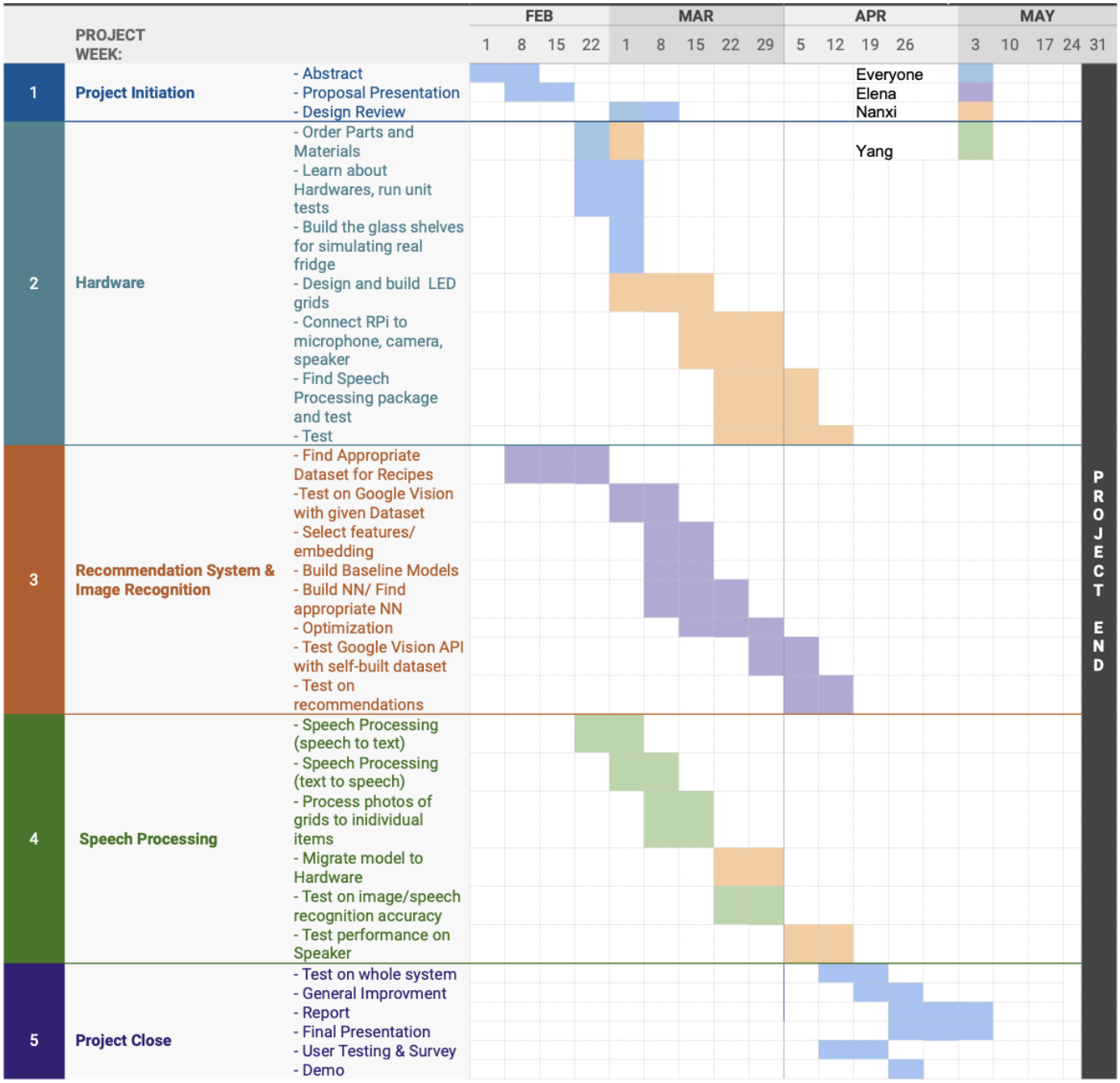



Figure 6: Gantt Chart

| Component | Quantity | Description | Cost | Total Cost |
|---|---|---|---|---|
| Jetson Nano | 2 | NVIDIA Jetson Nano Developer Kit (945-13450-0000-100) | 89.99 | 179.98 |
| Camera | 2 | NVIDIA Jetson Nano Camera IMX219-160 | 29.6 | 59.2 |
| SD card | 1 | 32 GB x 2 pack | 16 | 16 |
| Microphone | 1 | KISEER 2 Pcs USB 2.0 Mini Microphone, 2 pack | 8.99 | 8.99 |
| Speaker | 2 | HONKYOB USB Mini Speaker Computer Speaker | 11.99 | 23.98 |
| LED grid | 1 | ALITOVE WS2812B Addressable RGB LED Strip, 150 LEDS | 20 | 20 |
| Acrylic Board | 1 | SimbaLux Acrylic Sheet Clear Cast Plexiglass 12” x 12” x 2 pack | 17 | 17 |
| Acrylic Case for Jetson Nano | 2 | Makeronics Acrylic Case/Enclosure (Clear Transparent) for Jetson Nano with Camera Case | 17.5 | 35 |
| 5V2A power supply | 2 | Jetson Nano compatible power supply, borrowed from inventory: EDE0164,EDE0165 | 0 | 0 |
| | | | | 360.15 |
| Downloaded Tools | | | | |
| Nemo Toolkit on Jetson Nano | | Speech-to-text package | | |
| Google Vision API | | Image Recognition API | | |

Figure 7: Bills of Materials