American University of Sharjah

Department of Computer Science and Engineering

Smart Kitchen: Food Management System

This Project is submitted to the Department of Computer Science and Engineering at the College of Engineering in partial fulfillment of the requirements for the degree of Bachelor of Science Degree in Computer Science and Engineering

Presented By:

Farzaan Siddiqui - b00093566
Adithya Sankar - b00089172
Kareem Ahmed - b00088375
Mohammad Arfan Ameen - b00093780

Advised By:

Dr. Michel Pasquier

20th April, 2025

Table of Contents

Abstract

The Smart Kitchen project tackles the critical issue of food waste, a major contributor to environmental and financial challenges, by integrating IoT, AI, and mobile technologies for sustainable kitchen management. The system optimizes food inventory tracking, minimizes waste, and streamlines meal planning through sensors, cameras, and machine learning algorithms to monitor storage conditions, track expiration dates, and suggest recipes based on ingredients and user preferences.

A central processing unit coordinates data from IoT devices such as load sensors, gas detectors, and humidity monitors, ensuring real-time tracking of food quantities and conditions. A mobile app offers inventory updates, personalized meal plans, and automated shopping lists, enhancing convenience and reducing household costs while promoting sustainable consumption.

Key benefits include reduced food waste, alignment with UN Sustainable Development Goals, and efficient grocery management. Challenges such as privacy concerns, device compatibility, and kitchen integration are addressed through design refinements and user feedback.

1.Introduction

In recent years, the rise of smart home technologies has transformed the way we interact with our living spaces, offering increased convenience, efficiency, and sustainability. One area of focus within this field is the integration of intelligent systems to manage household resources, such as food and energy, more effectively. With growing concerns about environmental sustainability and the economic impact of wasted resources, there is a push to develop smart solutions that can automate everyday tasks while minimizing waste. The convergence of IoT (Internet of Things), AI (Artificial Intelligence), and mobile technology is enabling the creation of innovative systems designed to simplify daily routines, such as inventory management, meal planning, and shopping assistance, making households more efficient and eco-friendly. This project fits into this broader trend by addressing the issue of food waste through an intelligent kitchen management system.

1.1 Motivation and Background

In modern households, food waste is a significant problem that contributes to environmental degradation and significant economic loss. According to the Food and Agriculture Organization (FAO)[1], approximately one-third of all food produced globally is wasted, this issue arises from various factors such as over-purchasing, improper storage, and lack of awareness regarding food expiration dates. Additionally, managing grocery stock and deciding what to buy is typically a very time-consuming and laborious task. Many individuals and families struggle to keep track of what they have in their pantries and refrigerators, leading to unnecessary purchases and wasted food in the long run. Moreover, a lack of easy access to meal planning and inventory tracking tools amplifies the problem. By addressing these challenges, our project seeks to streamline the kitchen inventory management process, helping people make more sustainable choices while minimizing food wastage.

1.2 Project Description

This project aims to develop a system that effectively monitors and manages food inventory within the kitchen to reduce waste. We aim to automate the process to a significant extent while keeping costs low and providing the maximum functionality to aid the user in food management.

The proposed solution is to develop an Intelligent IOT kitchen that can detect food kept in the kitchen and make it convenient for the users to keep track of food items in their kitchen helping them with various purposes. The system is designed with a 360 camera that is capable of identifying items and their specific locations in the kitchen. Additionally, the system monitors the items and provides real-time alerts to the user if an item is near the expiration date and suggests recommendations on the recipes in which they can be used. The system becomes more efficient as the user uses it frequently. This allows the system to learn and understand the user's preferences for recipes and the types of food it can prepare. This system will also use a weighing scale/pressure pad to help automatically calculate the weights of food items for accurate tracking of objects while reducing the amount of manual input from the user. In addition, we will also implement a meal planning system that can allow users to plan their meals, and a smart shopping assistance system that will cross-check the ingredients that are available with the planned meal and notify the user of missing ingredients.

1.3 Design Objectives

- **Inventory Tracking:** It will monitor the food items in the kitchen using sensors and cameras in each cupboard/cabinet, fridge, etc. It will also have a weighing scale/pressure pad to weigh items. Will also keep track of the minimum quantity before requiring a restock of the item through a reminder.

- **Expiration Management:** The system will track and monitor the freshness of food items and their expiry dates, and give reminders for food that is about to expire.
- **Meal Planning and Recipe Recommendation:** The system will offer personalized meal planning by suggesting recipes based on the ingredients available in the user's inventory. It will analyze the stored food items and propose meal options, helping users create balanced meals while minimizing waste. The system will also recommend the user recipes which may be to the liking of the user, and the user will also be able to enter their recipes to plan their meals accordingly.
- **Smart Shopping Assistance:** With the help of inventory tracking, the system will generate a shopping list based on the analysis of items present in your inventory and those absent from their meal plan to create a shopping list.
- **Mobile Application:** The system will be connected to a mobile application to allow the user easy access to the information in the system, including notifications and reminders.

1.4 Limitations

Despite the system's promising capabilities, several limitations were identified during both the design and implementation phases. These constraints span technical, logistical, and user adoption challenges, which must be addressed to ensure the system's scalability, efficiency, and user acceptance.

1.4.1 Privacy and User Acceptance

One of the primary limitations relates to user perception of privacy. The integration of cameras and continuous monitoring through sensors may be seen as intrusive by users, potentially deterring adoption of the system into homes. Concerns over data security,

surveillance, and constant monitoring must be addressed through transparent privacy policies, user-controlled settings, and secure data handling mechanisms.

1.4.2 Physical Installation Constraints

The current system design often requires physical modifications to kitchen spaces, such as drilling and reconfiguring storage layouts. This can pose a significant barrier for users, especially those in rented or already optimized spaces. Additionally, the installation of pressure sensors and wiring in certain cupboards or containers may not be feasible due to spatial limitations, further complicating deployment.

1.4.3 Compatibility with Existing Smart Appliances

In homes with existing smart devices—such as smart fridges or stoves—compatibility may become an issue. The lack of integration or communication between different systems could lead to redundancy or inefficiency, undermining the value of the proposed solution. Furthermore, as an already full-fledge system the proprietary technology differ, furthering the notion of incompatibility.

1.4.4 Hardware Power Limitations

During implementation, a key issue encountered was the limited power delivery capacity of the Arduino setup. Each Arduino could only support one set of sensors due to power constraints, significantly increasing hardware costs. This limitation may stem from low wattage output from USB power cables or power losses through jumper cables and breadboards, indicating a need for better power management strategies.

1.4.5 Raspberry Pi Performance Constraints

The Raspberry Pi 4B, used as the central processing unit, lacks the computational power required to run larger object detection models efficiently. This limits the sophistication of real-time image recognition tasks and presents a bottleneck to system scalability. Alternative edge devices such as the newer model Raspberry Pi5 and the AiHat extension or cloud-based processing solutions may be considered in the future.

1.4.6 Manual Image Tagging for Model Training

A critical software limitation involved the absence of a reliable auto-tagging tool for training the object detection model. The team had to manually label images, a process that was both time-consuming and resource-intensive. Incorporating semi-automated labeling tools from Roboflow, Anaconda or exploring transfer learning with well-annotated datasets could mitigate this issue.

1.4.7 Camera-Based Object Recognition and Weight Estimation Accuracy

The current system’s ability to accurately detect objects and estimate weights using visual and sensor data is still limited by both algorithmic performance and environmental conditions. Lighting, occlusions, and object variation introduce potential inaccuracies. Time and budget constraints further limit the scope for refining detection accuracy.

1.5 Organization Of The Report

This project report starts with a Literature Review that examines current research and existing products to identify gaps relevant to the project. It moves into the System Specification, detailing the functional and non-functional requirements that describe the system's behavior, performance, and optional elements. The Technical Approach and Design Alternatives

section describes the proposed solution in detail, and includes various design diagrams while also considering alternative designs and resource needs. The Project Management Plan discusses how the project will be managed, focusing on timelines, responsibilities, and resource allocation. The implementation then explores the necessary hardware, software, and system integration. Following that, the Validation, Verification, and Performance Analysis Plan outlines how the system's functionality and performance will be tested and assessed. The broader Global, Economic, and Societal Impact of the project is analyzed to understand its implications beyond the technical scope. The report also includes relevant Standards that the project adheres to, and ends with a Conclusion that summarizes the key findings and proposes areas for future work.

1. Literature Review

Food management and the reduction of food waste are critical areas of focus, resulting in the development of numerous systems and projects aimed at assisting individuals in better managing their food and groceries. This literature review examines 47 research papers to investigate the various systems implemented, methodologies applied, and the hardware and software utilized in developing Smart Kitchen Systems. The literature review provides a comprehensive analysis of the following key areas: existing products in the market (e.g., Bottom Up Smart Kitchen, Samsung Smart Fridge, Siemens Smart Fridge), approaches to object recognition (e.g., RFID, Image Analysis and Smart Vision, NFC stickers), techniques for object quantity measurement (e.g., load/weight sensors, level measurement), database implementation methods (e.g., Google Firebase, MongoDB, MySQL), and the types of sensors used for specific functions (e.g., temperature sensors, humidity sensors, gas sensors, PIR sensors). Additionally, the review explores other supporting systems designed to enhance the accessibility and convenience of smart kitchen solutions for users.

2.1 Existing Products

2.1.2 Samsung Smart Fridge

The Samsung Smart Fridge [3], part of the company's Family Hub lineup, seamlessly integrates advanced technology with everyday convenience, offering a range of features designed to meet the needs of modern households. Designed to enhance food management, entertainment, and connectivity, the Samsung Smart Fridge serves as a multifunctional hub for any kitchen.

Key features include the control panel that allows users to browse the web, leave digital notes, and serve as a central command center for managing other smart devices in the home. Additionally, the fridge is equipped with internal cameras, enabling users to remotely check

the contents of their refrigerator via their smartphones without physically opening the door. This feature provides convenient access to inventory information, helping users avoid purchasing duplicate items or missing essentials.

The fridge also boasts an integrated food management system and voice assistant, which assist users in tracking expiration dates, creating shopping lists, planning meals, and suggesting recipes based on available ingredients. The voice assistant enables hands-free control of these systems, adding a layer of convenience and efficiency to kitchen management.

Despite its innovative features, the Samsung Smart Fridge does come with several drawbacks. Non-accurate quantity tracking, lack of user convenience, lack of modularity, and the user interest. The high cost makes it a significant investment, often placing it beyond the reach of many consumers. Additionally, the complexity of the technology may be overwhelming for less tech-savvy users, and the maintenance and software updates required to keep the system running smoothly can incur additional costs. Privacy concerns also arise from the use of internal cameras and continuous internet connectivity. Furthermore, the reliability of the smart features is highly dependent on stable internet access, which can affect performance if the connection is interrupted or weak.

2.1.3 Siemens Smart Fridge

The Siemens Smart Fridge [4], while similar to the Samsung Smart Fridge, is very limited in its “smart” applications. It allows users to connect and open the door of the fridge using voice commands and also see an image of the items inside the fridge using the internal cameras on the user's smartphone. However, it has most of the disadvantages of the Samsung Smart Fridge, mainly the high price point, security concerns, and the reliability of the smart features. Other than mentioned above, there is no real automation involved in the Fridge.

2.2 Object Identification

2.2.1 RFID

RFID technology plays a crucial role in enhancing smart kitchen systems by enabling efficient inventory management and user interaction. By employing RFID tags, systems can automatically identify and track kitchen items, monitor stock levels, and generate shopping lists to replenish the inventory[5][6]. Research Indicates that RFID can alert users about expired items and suggest recipes based on available ingredients, contributing to a more organized kitchen environment [7]. RFID technology will facilitate automated inventory tracking and restocking processes. However, due to the need for the user to manually scan each and every item individually adds a significant drawback to user experience.

2.2.2 Image Analysis And Smart Vision

Image analysis, particularly through the use of cameras and machine learning, is vital in enhancing the functionality of smart kitchen systems such as the IoT-based smart refrigerator. This particular system allows users to monitor and interact with their fridges in real time, utilizing image analysis for effective object recognition and tracking of items stored inside [8]. Coupled with temperature and humidity sensors, the refrigerator provides critical information alerts and notifications, ensuring optimal food storage conditions. Integrating image analysis into our smart kitchen project will enable automated inventory management and assist in recognizing items.

Smart vision, powered by artificial intelligence, is essential for enhancing the functionality of smart refrigerators by enabling the detection of quality, quantity, and type of fruits and vegetables. As noted by Luo et al.[9] this system utilizes computer vision to recognize, classify, and quantify produce stored within the refrigerator. Integrating smart vision

capabilities into our project will significantly improve user experience by providing real-time updates and insights and fostering a deeper understanding of food quality, enriching the overall smart kitchen environment.

2.2.3 NFC stickers

NFC stickers serve as a pivotal element in enhancing smart kitchen systems, exemplified by the pocket pantry storage solution As discussed by Kim et al. [10], these stickers enable seamless tracking of food items through mobile applications, allowing users to easily monitor their pantry inventory. The integration of NFC technology streamlines interactions with the system, simplifying the process of locating and managing food supplies. Similarly to the RFID tags, due to the need for the user to manually scan each and every item using a NFC reader individually adds a significant drawback to user experience. The system required to create these stickers and manage them is also an extra burden on the user.

2.3 Quantity Measurement

2.3.1 Load/Weight sensors

We found systems that incorporated a variety of sensors to enhance their functionality, with load sensors playing a particularly crucial role in evaluating the weight of objects [11]. These load sensors are essential for monitoring the weight of stored items, which is fundamental to tracking inventory levels in real time [12]. The system begins by recording the initial weight of each object [13], which serves as the baseline or full volume [14]. As items are used or removed, the load sensors detect weight changes, allowing the system to calculate the amount consumed.

By combining the weight data and visual information [15], the system can provide a more accurate estimation of the remaining amount of each object [16]. This integrated approach enhances the system's ability to manage and track inventory efficiently, offering a real-time view of available resources, which is especially useful for smart kitchens or automated inventory systems[17].

2.3.2 Level measurement

Building on the idea presented in section 2.3.1, some systems have a camera within the cupboard that captures images of the objects stored and the quantity of products available. This camera plays a crucial role in enhancing inventory management by providing a visual representation of the contents. The captured images are processed using MATLAB R2019a, a software known for its robust image-processing capabilities [18] discussed in *2.2.2*.

Through image processing techniques, the system analyzes the captured images to determine the level of each product and its current position within the cupboard [19]. This analysis allows for more accurate tracking of inventory, as the system can identify which items are present and their respective quantities. By combining the data from the load sensors with the visual information obtained from the camera, the system creates a comprehensive overview of the stored items [20]. This multifaceted approach not only improves the accuracy of inventory management but also facilitates better organization and accessibility of products, making it particularly beneficial for users in smart kitchen environments [21][22]. However, this approach is very hard to implement on a large scale, and is also not as good as using weight instead.

2.4 Sensors

2.4.1 Temperature Sensor

Temperature sensors are crucial components that accurately determine temperature. Ashwathan R. et al [23] introduced a system that used sensors to detect the temperature inside a refrigerator, ensuring consistent temperature for food storage. In another study conducted by Sowndarya Palanisamy et al.[24] used a DHT11 sensor to measure the temperature of containers with great precision, which is crucial for preventing food spoilage. Additionally, Shreya Gupta et al.[25] employed thermistors for temperature measurement in their research. Furthermore, Neethu Nadar et al.[26] utilized an LM35 series sensor for temperature and a DHT22 humidity sensor to detect cooling defects or lower temperatures in the fridge, making it possible to store foods requiring specific temperature conditions.

2.4.2 Humidity Sensor

When it comes to food preservation, humidity is a crucial parameter. Sowndarya et al. [24] conducted a study in which they used a sensor to measure the electrical resistance between two electrodes for accurately detecting and monitoring humidity levels. Additionally, it utilizes a moisture-absorbing substrate to enhance the precision of their humidity measurements. In a separate research conducted by Ashwath R et al [23], they demonstrated the use of a Node NodeMCU controller to control and power a humidity sensor. Moreover, Devanath B et al [27] demonstrated in the paper that it showcases the implementation of humidity sensors in the food industry to monitor and maintain conditions for ensuring the safety and quality of food products.

2.4.3 Gas sensor

Gas sensors help in resolving issues by detecting and preventing gas leaks, which can potentially lead to kitchen fires. Additionally, gas sensors can also monitor the concentration of specific gasses, thereby helping to prevent food spoilage. For example, Shah et al. [28] used a system equipped with an MQ-135 gas sensor to detect LPG leaks. Additionally, research by Shah et al. [28] also used the MQ-135 gas sensor, which can detect combustible gasses such as LPG, CH4, CO, and propane. In another study conducted by Simran Goelet et al., [29] the Flying Fish MQ3 Gas sensor was employed to detect the presence of ethylene. Ethylene is known to induce fruit ripening, and an excessive amount can result in fruit spoilage. This is particularly significant as the spoilage of one fruit can lead to the spoilage of others due to the spread of ethylene.

2.4.4 Other sensors

Various types of sensors are used in different research papers, including passive infrared (PIR) and infrared flame sensors, which are used for detecting human activity and fire incidents. For instance, a study by Shah et al [28] used PIR and infrared flame sensors to identify kitchen fires. Additionally, Nasser et al.'s [30] paper introduces Light Dependent Resistors (LDR) to monitor containers containing liquids.

2.5 Shopping Cart System

Numerous studies have implemented automated shopping list generation by comparing the current inventory with the user’s predefined list of stocked items, ensuring that quantities remain above specified minimum thresholds [5][6][31]. The generated list is then transmitted to the user, either via a mobile application or a web-based platform, for review and further action.

2.6 Database Implementation

2.6.1 Google Firebase

Google Firebase in the context of smart kitchens, provides a good backend solution. In a study conducted by Abrar et al., sensor [32] and microcontroller data were uploaded to Firebase and integrated with the app. Firebase is known for its strong computational capacity and efficient handling of data transfer, which saves time. Another paper by Ashwatan et al [23] discusses using Firebase as a real-time database for storing sensor-collected data. Another paper [33] discusses the use of Firebase to store images which was used for image processing techniques to identify vegetables. The only issue found is that there are a limited number of reads and writes that can be done to the database, which can be solved by buying a subscription for the service.

2.6.2 MongoDB

MongoDB is a NoSQL database that allows for dynamic schemas, enabling developers to store unstructured or semi-structured data without requiring predefined schemas. It also supports the distribution of data across multiple servers, which helps improve performance and enables the database to handle high throughput applications with large datasets efficiently. For example, [34] uses MongoDB connected to a Raspberry Pi and a fingerprint reader, in order to save fingerprints and do authentication of people. Drawbacks with MongoDB is that the data that can be uploaded is very limited unless a cluster is bought.

2.6.3 MySQL

MySQL can be used as a central database for managing data from various sensors and microcontrollers in a smart IoT kitchen. It can store information on inventory levels and

expiration dates, allowing for real-time updates and notifications. It could be integrated with mobile apps, enabling users to easily manage their kitchens. For example, in a system proposed by Chatterjee [35], MySQL was used to maintain the database. Moreover, another study by Sivakumar et al. [36] also used MySQL to manage system data efficiently. However, we would need a proper server and setup for MySQL to run, which can be time consuming and resource intensive.

2.7 Applications

Applications and interfaces can help users control the system more efficiently and provide real-time data updates. Chatterjee et al.[35] proposed a system that utilizes a mobile app developed for Android. It will send alerts to the user through this app. Additionally, Sandeep et al.[37] presented an article that describes an Android application created using Java in Eclipse. This app allows users to access kitchen parameters such as containers and gas leakage, which are displayed, and regular notifications are sent to the user as they change. Another article by Rezwan et al. [38] discusses an app with a website and mobile app for ordering groceries with cash on delivery. Users can create monthly grocery lists, receive alerts when items are running low, and track their orders through this website. Furthermore, Kim et al. [39] proposed a system that uses a mobile app allowing users to manage pantry supplies. The app uses speech recognition to control the system that shows the requested item. It is built with Google's Flutter, which works on both Android and iOS.

2.8 Recipes recommendation

Smart kitchen systems are increasingly integrating personalized recipe and recommendation features to enhance user experience and promote efficient use of available ingredients. These functionalities leverage data on current inventory, dietary preferences, and cooking habits to suggest meal ideas tailored to the user's needs. By utilizing machine learning algorithms and

real-time inventory tracking, many systems can suggest recipes based on items already present in the kitchen, reducing food waste and minimizing additional grocery purchases. For instance, ChefAI.IN demonstrates the potential of AI algorithms in generating culturally specific recipes, such as Indian cuisine, by analyzing ingredients on hand and providing personalized suggestions [40]. Similarly, a system named AutoChef explores the automated generation of cooking recipes, allowing users to receive recommendations that align with both preferences and availability [41]. These recommendation systems would be required to run on a cloud or external service instead of running locally. We will need to adapt these systems to be able to run on our system.

2.9 Expiration Date Management

The tracking and estimation of expiration dates is a necessary component of any inventory system as it consistently ensures that no spoilage occurs to the items and to allow the user to be notified for their restocking. However, as we researched more extensively on the implementation, we are yet to see this feature documented properly.

2.10 Conclusion

The literature review provides a comprehensive analysis of existing Smart Kitchen Systems, focusing on the integration of various technologies to enhance food management, reduce waste, and improve user convenience. The review covered numerous research papers that explored diverse approaches to system implementation, object recognition, object quantity measurement, database integration, and the use of various sensors in smart kitchens.

The findings reveal that the implementation of smart kitchen systems involves a combination of advanced hardware and software technologies, such as image analysis, smart vision, and active weight monitoring for effective object recognition. These technologies contribute

significantly to optimizing inventory management, ensuring accurate tracking of food items, and minimizing waste. Weight and level measurement sensors further aid in the precise monitoring of food quantities, enhancing the accuracy and efficiency of smart kitchen solutions.

Database integration, particularly using platforms such as Google Firebase, plays a crucial role in maintaining real-time data, enabling users to manage their kitchens remotely through mobile applications. The review also highlights the importance of environmental sensors, including temperature, humidity, and gas sensors, which are essential for maintaining the optimal storage conditions needed to prevent spoilage and enhance food safety.

Despite the innovative advancements in smart kitchen systems, several limitations and challenges were identified, such as the high cost of advanced appliances, privacy concerns associated with camera and sensor use, and the dependence on internet connectivity for optimal performance. Addressing the aforementioned issues and working on reducing the existing limitations will be vital for the widespread adoption and success of these systems in the consumer market.

Our smart IoT kitchen system will use a variety of technologies to enhance user experience and efficiency. We will utilize image analysis for inventory tracking and supply management. Load sensors will monitor the weight of kitchen items, while gas sensors will detect ethylene gas to assess the ripeness of fruits and vegetables. Additionally, we will implement a shopping cart system for grocery management. Moreover, integrating artificial intelligence into our system will suggest cooking recipes based on available ingredients hence simplifying meal preparation for users. To further optimize the system, we will incorporate passive infrared sensors, as well as temperature and humidity sensors to monitor the kitchen environment. Our development will focus on creating a user-friendly mobile application for

Android using Java. A 360-degree camera will enhance image analysis and will improve inventory tracking. By combining these innovative technologies, we aim to create an efficient and intuitive smart IoT kitchen that transforms cooking and grocery management into a more enjoyable experience.

2. System Specification:

3.1 Functional Requirements:

F.R. 1: The system shall monitor and measure food items using sensors, cameras, and scales in kitchen storage areas.

F.R. 2: The system will allow users to input and modify details:

F.R. 2.1: The system shall allow users to input and modify item information manually.

F.R. 2.2: The system shall allow users to add/remove items directly into a shopping list.

F.R. 2.3: The system shall allow users to save and use custom recipes.

F.R. 2.4: The system shall allow users to set and change notification preferences.

F.R. 2.5: The system shall allow users to input/modify expiration and expiration reminder dates.

F.R. 3: The system will allow users to view information and receive alerts:

F.R. 3.1: The system shall allow users to view the details of each item.

F.R. 3.2: The system shall allow users to view detailed reports of their inventory.

F.R. 3.3: The system shall send users an alert when items near/have reached their expiration date.

F.R. 3.4: The system shall provide users with a historical log of inventory changes.

F.R. 3.6: The system shall provide users with a log of expired/removed items

F.R. 3.7: The system shall generate and provide users with a shopping list of required/missing items for meal preparation.

F.R. 3.8: The system shall allow users to view previous shopping lists entries.

F.R. 3.9: The system shall alert users when items are unable to be identified or incorrectly measured.

F.R. 4: The system shall store and log the necessary information:

F.R. 4.1: The system shall store item name, quantity, expiration date, freshness, and location.

F.R. 4.2: The system shall store user information and preferences regarding notifications, recipes, and meal preparations.

F.R. 4.3: The system shall log all items added, moved, or used including when and quantity.

F.R. 4.4: The system shall log all items that expired or were removed after the expiration date.

F.R. 4.5: The system shall store where each item is located.

F.R. 4.6: The system shall store recipes specified or entered by the user.

F.R. 5: The system shall give users recommendations:

F.R. 5.1: The system shall recommend recipes to the user based on preferences, existing inventory, and past meals.

F.R. 5.2: The system shall recommend recipes to the user based on items nearing expiry.

F.R. 5.3: The system shall recommend how, where and what temperature items should be stored in.

F.R. 5.4: The system shall recommend quantities of items to order based on expiration logs, and meal plans.

F.R. 6: The system shall do the necessary computations:

F.R. 6.1: The system shall compute the change in weight on a pressure pad/weighing scale to measure the weight of an object.

F.R. 6.2: The system shall compare stored item quantity and weight to generate alerts

F.R. 6.3: The system shall compare stored item expiration dates and user preferences on alert timing to generate alerts.

F.R. 6.4: The system shall compare items required for the current meal plan and items in inventory to generate a shopping list.

F.R. 6.5: The system shall compute the optimal quantities of each item.

F.R. 7: The system shall allow users to make any inputs or receive any reports, lists, or notifications, through an interface or a mobile application.

3.2. Nonfunctional requirements

N.F.R. 1: Performance

N.F.R. 1.1: Response Time: The system must process user commands and interactions with a response time of no more than 2 seconds to ensure a seamless user experience. This is critical for real-time inventory monitoring, notifications, and updates.

N.F.R. 1.2: Multi-User Handling: The system shall support concurrent usage by multiple users without performance degradation. It should handle at least 100 simultaneous users effectively for household and multi-tenant applications.

N.F.R. 1.3: Real-Time Data Updates: The system shall provide real-time updates and notifications for monitored appliances, inventory changes, and kitchen conditions. Users should receive alerts immediately when thresholds or conditions are met.

N.F.R. 2: Security

N.F.R. 2.1: Data Protection: User data, including login credentials, inventory data, meal plans, and shopping lists, must be encrypted both at rest and during transmission using industry-standard encryption protocols (e.g., AES-256 for data at rest, TLS 1.3 for data in transit).

N.F.R. 2.2: Authentication and Access Control: The system shall include secure authentication mechanisms, such as multi-factor authentication (MFA), to verify user identity and prevent unauthorized access.

N.F.R. 2.3: Regulatory Compliance: The system must comply with data protection laws and regulations relevant to the target market (e.g., GDPR, CCPA) to safeguard user privacy and ensure that personal data is accessible only by authorized users.

N.F.R. 3: Device Compatibility

N.F.R. 3.1: IoT Protocol Support: The system must support various IoT communication protocols (e.g., MQTT, HTTP, CoAP) to ensure compatibility with a wide range of smart kitchen appliances from different manufacturers.

N.F.R. 3.2: Cross-Platform Accessibility: The system shall be accessible through both iOS and Android mobile applications, as well as major web browsers (e.g., Chrome, Safari, Firefox, Edge) for ease of use across devices.

N.F.R. 3.3: Hardware and OS Support: The mobile application and web platform must function seamlessly on common operating systems, supporting devices released within the past 5 years to maintain broad usability.

N.F.R. 4: Maintenance

N.F.R. 4.1: Modular Architecture: The system must have a modular design that allows for easy updates and maintenance of individual components (e.g., inventory tracking, meal planning, notification modules) without affecting the entire system's functionality.

N.F.R. 4.2: Remote Updates: The system shall support over-the-air (OTA) updates for firmware and software patches to maintain security, add new features, and fix bugs with minimal user intervention.

N.F.R. 4.3: Diagnostic Tools: Built-in diagnostic tools must be available for developers and users to troubleshoot issues quickly and efficiently, identifying and isolating malfunctioning components.

N.F.R. 5: Reliability

N.F.R. 5.1: Resilience and Failover: The system must be designed for resilience. If a component fails (e.g., a sensor, network connection, or cloud service), other parts of the system should continue to function. Critical functions such as inventory updates and appliance control must have local fallback options to ensure ongoing operations.

N.F.R. 5.2: Data Backup and Recovery: The system shall perform automatic, scheduled backups of all critical data, including user preferences, device settings, and usage history. Restoration capabilities should be available, allowing data recovery within 2 minutes in the event of data loss or system failure.

N.F.R. 5.3: Load Management: The system must manage load distribution effectively to prevent server overloads during peak times. Load balancing mechanisms must ensure a consistent performance experience, even under high user demand.

N.F.R. 6: Usability and User Experience

N.F.R. 6.1: User Interface (UI): The system must offer an intuitive and user-friendly interface that accommodates users of different technical expertise levels. The design must prioritize ease of navigation, clear categorization, and informative feedback during user interactions.

N.F.R. 6.2: Accessibility: The system shall include multi-language support and accessibility features such as screen reader compatibility, adjustable text sizes, and high-contrast mode to cater to users with disabilities.

N.F.R. 6.3: Customization: Users must be able to customize notification settings, including types of alerts, delivery channels (e.g., SMS, push notifications), and timing to suit individual preferences.

N.F.R. 7: Scalability

N.F.R. 7.1: Scalable Architecture: The system must be scalable to accommodate future growth, including the addition of new kitchen appliances, users, and data sources without needing significant architectural changes.

N.F.R. 7.2: Horizontal and Vertical Scaling: The system should support horizontal scaling (adding more machines) and vertical scaling (upgrading existing machine capabilities) to meet increased data processing and user interaction demands.

N.F.R. 8: Data Integrity and Accuracy

N.F.R. 8.1: Data Synchronization: The system must ensure that data across all devices and platforms remains synchronized, preventing data discrepancies and ensuring that users view consistent information.

N.F.R. 8.2: Redundant Data Checks: The system shall perform redundant checks to verify the accuracy of input data from sensors and user interactions, ensuring that inventory and expiration updates reflect true conditions.

Pseudo Requirements (Optional Constraints):

- **Integration with Legacy Systems**: If required by specific users or clients, the system shall interface with pre-existing kitchen automation systems or data repositories, potentially written in older languages or with outdated data structures.
- **Implementation Language**: The primary development language for core modules must align with client preferences (e.g., Python for its IoT libraries, JavaScript for frontend and web integration).

3.3 Use Cases

3.3.1 Use Case Diagram

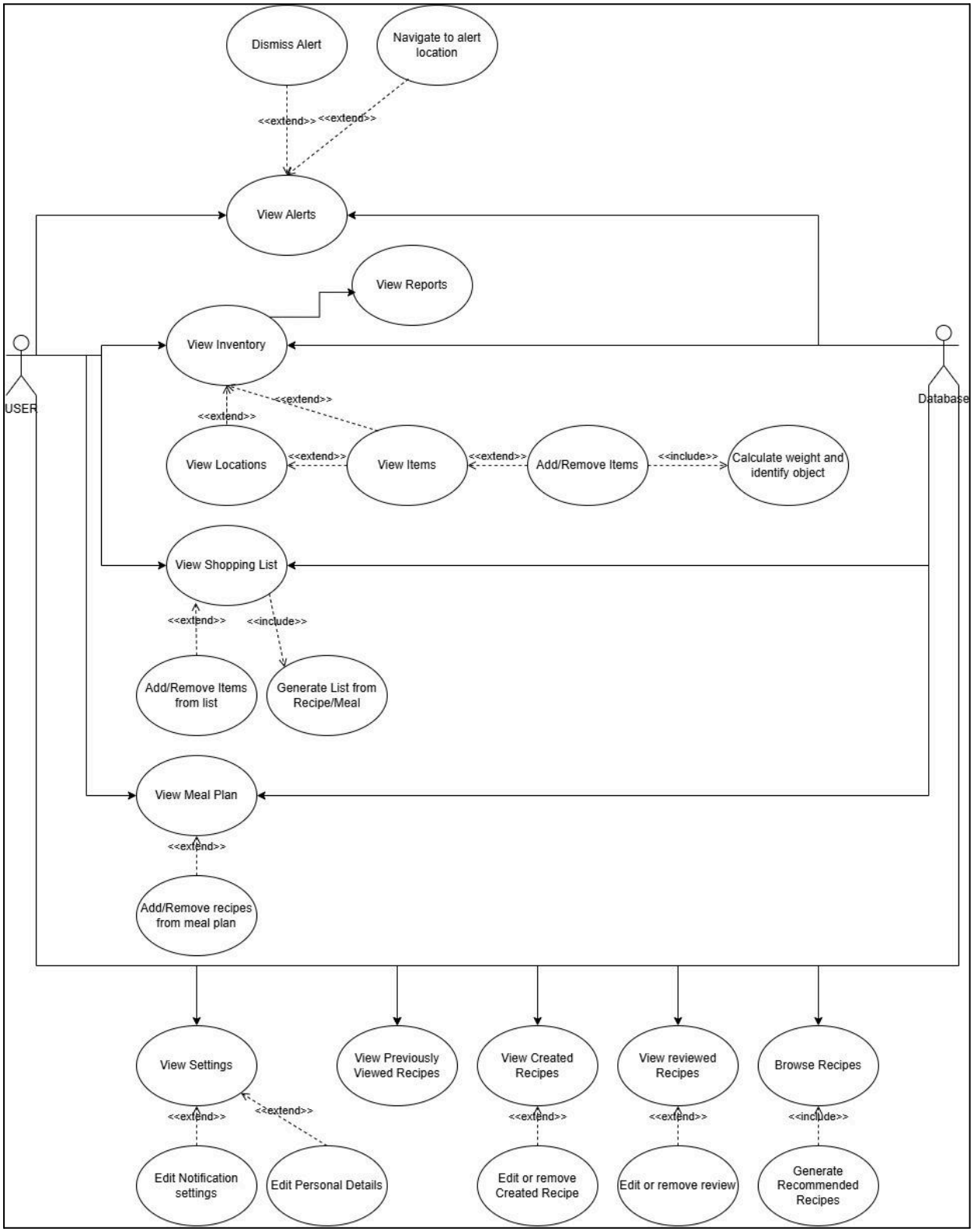


Fig. 1: Use Case Diagram

3.3.2 Use Case Tables

Use Case ID:	FR1.1
Use Case Name:	Visual Inventory Detection
Created By:	Kareem Ahmed
Actors:	User, Database
Description:	The system uses cameras to visually detect and identify items in the kitchen, updating the inventory system.
Preconditions:	Cameras must be installed and configured. The Central Processing System must be connected to the cameras and Firebase.
Postconditions:	The detected items are logged in the inventory database. The inventory list is updated and displayed on the user interface.
Normal Course:	- The interior camera captures images of the cupboard and fridge areas. - The Central Processing System processes the images using computer

	vision algorithms. - Detected items are identified and matched with the database entries. - The database is updated with the items' quantities and storage locations. - The user sees updated inventory details on the interface.
Alternative Courses:	If an item is not recognized, the system prompts the user to manually input it through the mobile app.
Exceptions:	If the camera fails to detect or capture an image, the system will notify the user to check the camera placement or configuration.
Includes:	Object Detection, Inventory Update
Assumptions (if any):	The cameras are positioned correctly for effective item tracking.

Table. 1: Inventory Detection via Cameras

Use Case ID:	FR1.2
Use Case Name:	Weight-Based Inventory Tracking

Created By:	Mohammad Arfan Ameen
Actors:	User, Database
Description:	Weight sensors connected to Arduino devices are used to measure and update inventory quantities.
Preconditions:	Weight sensors must be installed and connected to the Arduino boards, which interface with the database.
Postconditions:	Updated weight data is stored, reflecting the current quantity of items.
Normal Course:	- Weight sensors detect changes in item weights. - Arduino boards process the data and send it to the Firebase database. - The system updates the quantity of the item based on the weight measurement.
Alternative Courses:	If the weight data is inconsistent, the system notifies the user for manual verification.
Exceptions:	If a sensor malfunctions, the Arduino board logs the error and triggers a system alert.

Includes:	Data transmission from Arduino, user notification for discrepancies
Assumptions (if any):	Weight sensors are calibrated and functioning properly.

Table. 2: Weight-Based Inventory Updates

Use Case ID:	FR1.3
Use Case Name:	Environmental Monitoring of Storage Conditions
Created By:	Farzaan Siddiqui
Actors:	User, Database
Description:	Monitoring temperature and humidity levels in storage areas to ensure proper food storage conditions.
Preconditions:	Temperature and humidity sensors must be set up and connected to the Arduino devices.
Postconditions:	The system logs the temperature and humidity data in the database, providing the user with current storage condition details.

Normal Course:	- Temperature and humidity sensors measure the environmental conditions. - Data is processed by the connected Arduino and sent to the Firebase database. - The system displays current temperature and humidity levels on the mobile app.
Alternative Courses:	The system alerts the user if temperature or humidity exceeds set thresholds
Exceptions:	If sensor readings are unavailable, the system retries data collection and logs the failure if it persists.
Includes:	User notifications for threshold breaches
Assumptions (if any):	Sensors are positioned accurately and are operational.

Table. 3: Temperature and Humidity Monitoring

Use Case ID:	FR2.1
Use Case Name:	Expiration Management - Expiry Date Input
Created By:	Adithya Sankar

Actors:	User, Database
Description:	The system will allow the user to manually input expiry dates for food items stored in the system, ensuring items approaching expiry are properly monitored.
Preconditions:	The Central Processing System must be functional. The user must have valid access to the mobile app.
Postconditions:	Expiry dates are stored in the database. Users are notified when an item is approaching its expiry date.
Normal Course:	- The user inputs the expiry date for a new item either via the mobile app. - The Central Processing System processes and sends the data to the database. - The database updates the item's details, including its expiry date. - The system calculates the time remaining until expiration. - The user receives notifications based on the set threshold before the expiry date.

Alternative Courses:	If the user fails to input a date, the system will remind them to do so within the application.
Exceptions:	If communication fails betweenmThe Central Processing System, the user receives a notification to reattempt input.
Includes:	Database Update, Notification System
Assumptions (if any):	The Firebase database is accessible and operational.

Table. 4: Expiry Date Input

Use Case ID:	FR2.2
Use Case Name:	Expiry Date Notification Alert
Created By:	Kareem Ahmed
Actors:	User, Database
Description:	The system will send reminders for items approaching their expiration dates based on user-defined thresholds.
Preconditions:	Expiration dates must be properly input and synced with the cloud database.

Postconditions:	Users receive timely reminders, and the system tracks expired items.
Normal Course:	- The system checks the expiration dates for items in the inventory at regular intervals (daily/weekly). - If an item's expiration date is within a user-defined threshold, a reminder is triggered. - The mobile app sends notifications to the user's device. - The user receives an alert and can plan their usage accordingly.
Alternative Courses:	If the user has disabled notifications, no reminder is sent.
Exceptions:	If there is a failure in the notification service, the system logs the issue and retries sending the alert.
Includes:	Notification system, date checking
Assumptions (if any):	Users have set expiration threshold preferences in the app.

Table. 5: Expiry Reminder

Use Case ID:	FR2.3

Use Case Name:	Expiration Log
Created By:	Mohammed Arfan Ameen
Actors:	User, Database
Description:	This use case tracks and logs expired items, allowing users to view a history of expired food items and gain insights into food wastage patterns.
Preconditions:	The expiration dates for items are stored in the system. The system tracks all food item removals or disposals after expiration.
Postconditions:	An updated log of expired items is stored in the system. Historical data regarding expired items is available for user review.
Normal Course:	- As food items approach expiration, the system tracks them for removal or disposal. - When an item expires, it is flagged in the inventory and logged as expired in the system.

	- The user can view a detailed report of expired items, including dates and usage patterns, via the app or interface.
Alternative Courses:	If items are not removed by the user after expiration, the system sends a reminder for manual removal.
Exceptions:	If an item is incorrectly marked as expired, the system flags the data for review.
Includes:	Data storage for expired items. Historical data analytics for expired item patterns.
Assumptions (if any):	The system properly tracks item removals and updates the expiration log in real-time. Users can view historical logs of expired items via their interface.

Table. 6: Expiration Log

Use Case ID:	FR3
Use Case Name:	Suggest Recipes Based on Available Ingredients

Created By:	Farzaan Siddiqui
Actors:	User, Database, Meal Planner Algorithm
Description:	Based on the available ingredients detected in the inventory, the system will suggest recipes to the user that match the available items.
Preconditions:	The inventory tracking system must be functional and updated in real-time with current ingredient quantities. The user must have provided dietary preferences and restrictions.
Postconditions:	A list of suggested recipes appears on the user interface. The user selects a recipe to proceed with.
Normal Course:	- The system retrieves the current inventory data from the database. - The meal planner algorithm analyzes the available ingredients. - Based on the ingredients available, the system generates a list of recipes. - The list of recipes is displayed on the user interface for selection. - The user selects a recipe, which is then added to their meal plan.

Alternative Courses:	The system will suggest substitutions or alternative recipes if there are not enough ingredients for a meal.
Exceptions:	If the inventory data is outdated, the system will prompt the user to update the information.
Includes:	Inventory Analysis, Recipe Generation
Assumptions (if any):	The recipe database contains sufficient options that meet dietary needs. Inventory data is correctly synced with the Firebase database.

Table. 7: Meal Planning Recommendations

Use Case ID:	FR4
Use Case Name:	Generate Shopping List Based on Missing Items
Created By:	Adithya Sankar
Actors:	User, Database
Description:	The system generates a shopping list based on the analysis of the user's meal plan and current inventory, highlighting missing or low-stock items.

Preconditions:	The meal planning system must be operational. The inventory tracking system must have accurate data.
Postconditions:	A shopping list is generated, displaying missing items. The user may confirm or edit the list.
Normal Course:	- The system analyzes the user's meal plan and compares it with the available inventory. - Missing or low-stock items are identified and added to the shopping list. - The user is presented with a shopping list on mobile app. - The user can manually add or remove items from the list. - The user may choose to purchase directly through the app or store preferences for future shopping.
Alternative Courses:	If the user prefers to shop manually, they can disable automatic list generation.
Exceptions:	If the shopping list algorithm encounters an

	error (e.g., missing inventory data), the system will notify the user and allow manual input.
Includes:	Inventory and Meal Plan Analysis, Shopping List Creation
Assumptions (if any):	The meal plan data is up-to-date and complete.

Table. 8: Smart Shopping Assistant

3. Technical Approach and Design Alternatives

4.1 System Overview

As discussed in Section 1, food waste is a consistent problem in households, leading to environmental, economic, and social costs. A significant amount of food is wasted due to over-purchasing, improper storage, and a lack of awareness and tracking of expiration dates. Current systems and implementations lack easy-to-use, integrated features for inventory tracking, expiration monitoring, meal planning, etc, in a single solution. We aim to solve these issues by developing an intelligent kitchen management system that leverages IoT and AI to streamline food inventory management, reduce waste, and support users in meal planning and grocery shopping. This system includes the following main modules:

- **Inventory Tracking**: Sensors, cameras, and weighing scales monitor food items in the kitchen to track quantity, weight, and location. The system will notify users when items reach a minimum threshold, prompting restock reminders.
- **Expiration Management**: The system will track food freshness and rough expiration dates, sending reminders for items nearing expiration to minimize spoilage.
- **Meal Planning and Recommendation System**: Based on available ingredients, the system will suggest personalized recipes and meal plans. Users can input their recipes, and the system will learn preferences over time to offer more tailored suggestions.
- **Smart Shopping Assistance**: The system will create a shopping list based on meal plan requirements and inventory analysis
- **Mobile Application**: A mobile app will connect users to the system, providing notifications, reminders, and easy access to inventory and meal planning information on the go.

This solution aims to provide users with an integrated, user-friendly experience that simplifies kitchen management and supports sustainable, efficient grocery shopping while reducing food waste and cost.

4.2 Design of the Solution

The proposed solution has each location identified, with sensors connected to an Arduino which communicates with a main Raspberry Pi. This Raspberry Pi does the main computations and communication with the database and users and has two cameras connected which are RPi camera module v3 and OV3660 Camera for object detection Module. Each Arduino has sensors such as Humidity/temperature sensors, gas sensors and load cells connected to amplifiers The overall kitchen is monitored by a 360-degree camera which identifies and tracks items and where items are stored. The whole system is also connected to a local LCD Monitor and accessible to the user through a mobile application for ease of access.

4.2.1 System Overview

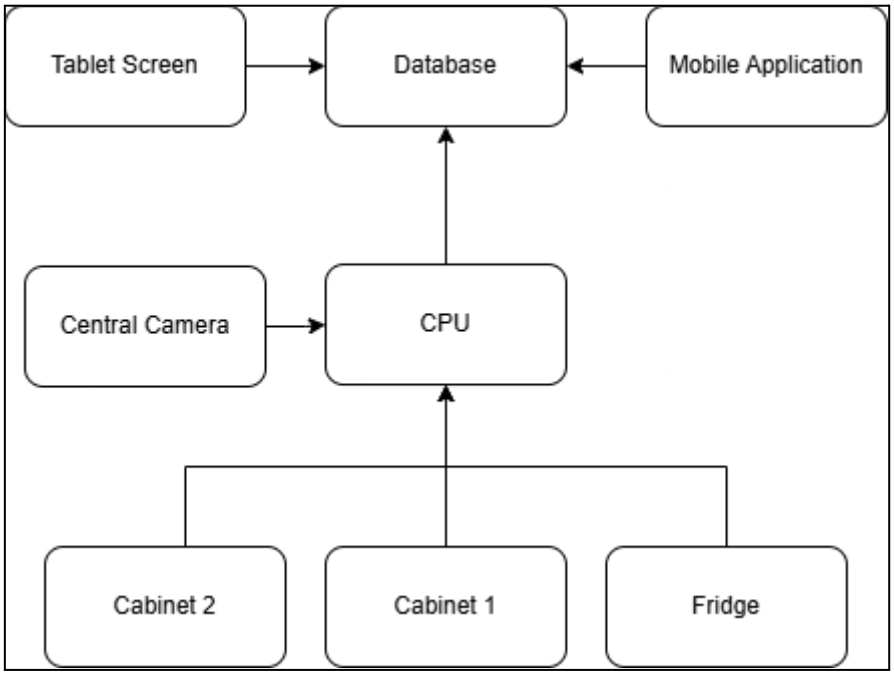


Fig. 2 Overall System Architecture

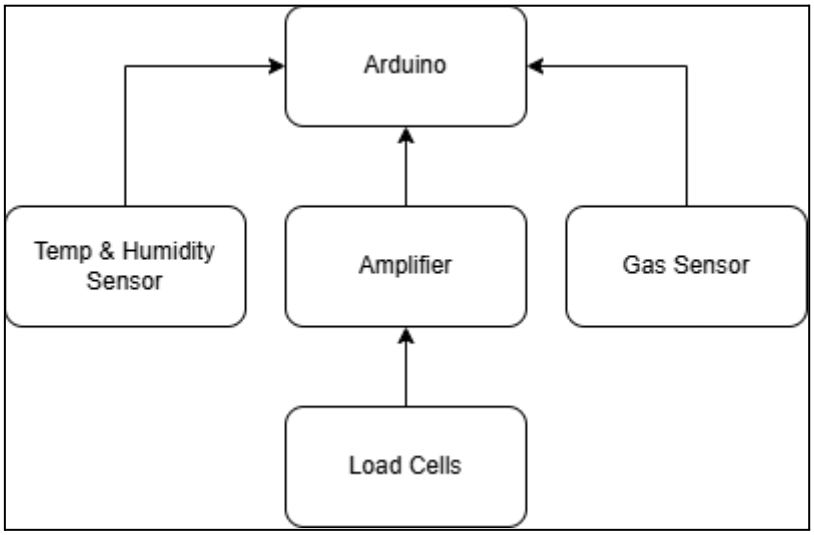


Fig. 3: Internal Cabinet Block Diagram

This block diagram in Fig.2 represents the proposed system across multiple kitchen storage areas, such as cabinets and a fridge. At the core of the system is a Raspberry Pi that serves as the central processing unit, coordinating with multiple Arduino microcontrollers (Arduino 1, 2, 3, and 4) assigned to different storage areas. Each Arduino setup is equipped with various sensors to track specific environmental conditions and item characteristics.

For example, each cabinet and the fridge are outfitted with temperature and humidity sensors, HX711 amplifiers to monitor conditions, gas sensors for detecting spoilage gasses, and load cells to measure the weight of food items, ensuring accurate inventory tracking. The system also includes an RPI camera module v3 and OV3660 camera module to recognize and identify items stored in each area. The Raspberry Pi aggregates data from all sensors and communicates with a weight scale, camera, LCD monitor, and mobile device to provide real-time feedback, reminders, and updates on inventory status.

4.2.2 System Workflow

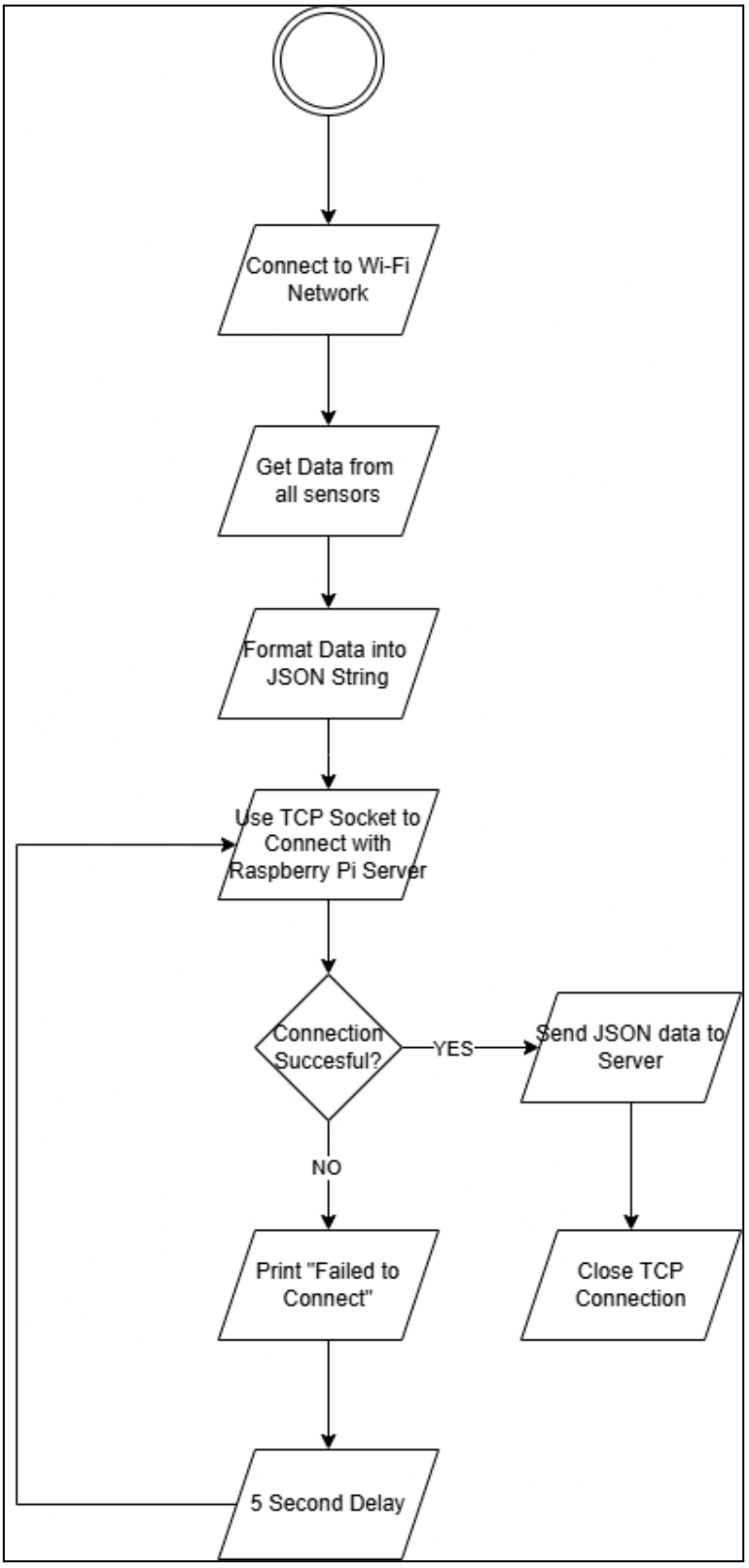


Fig. 4: System Initialization

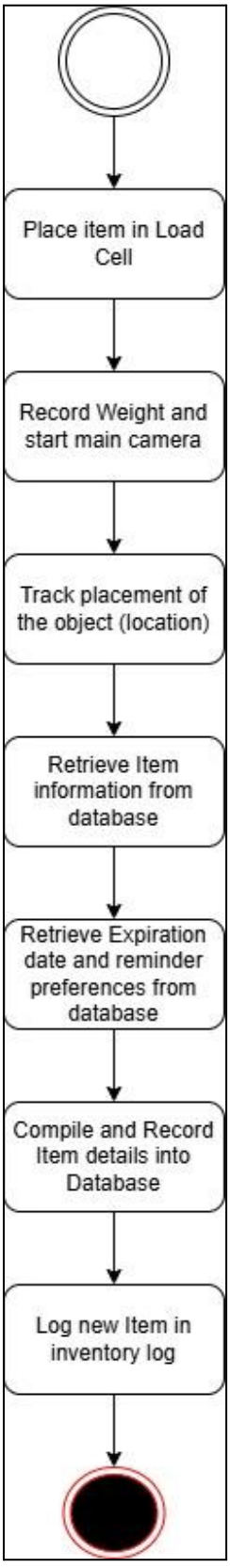


Fig. 5: Entry of Items

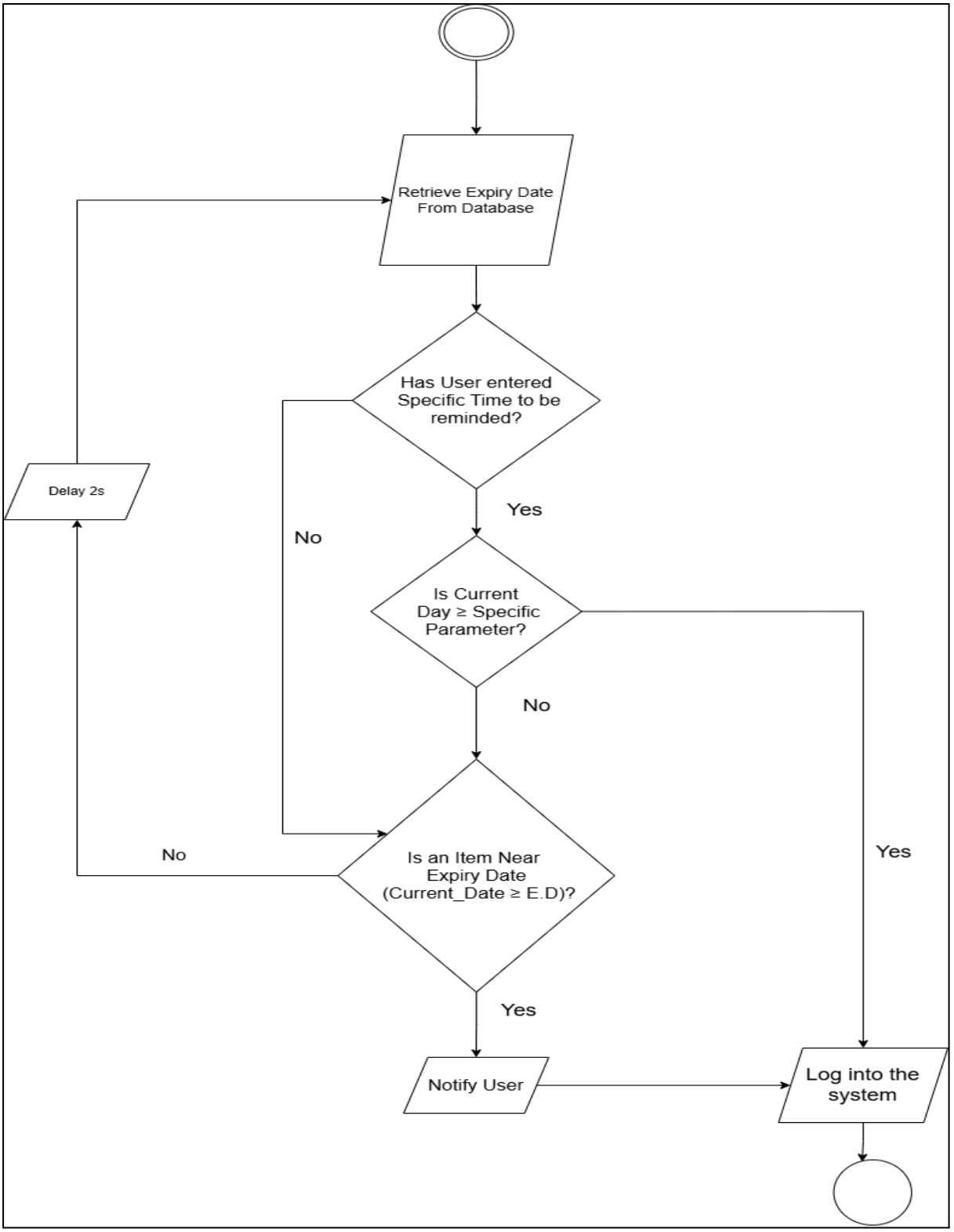


Fig. 6: Expiry Date Check of Items

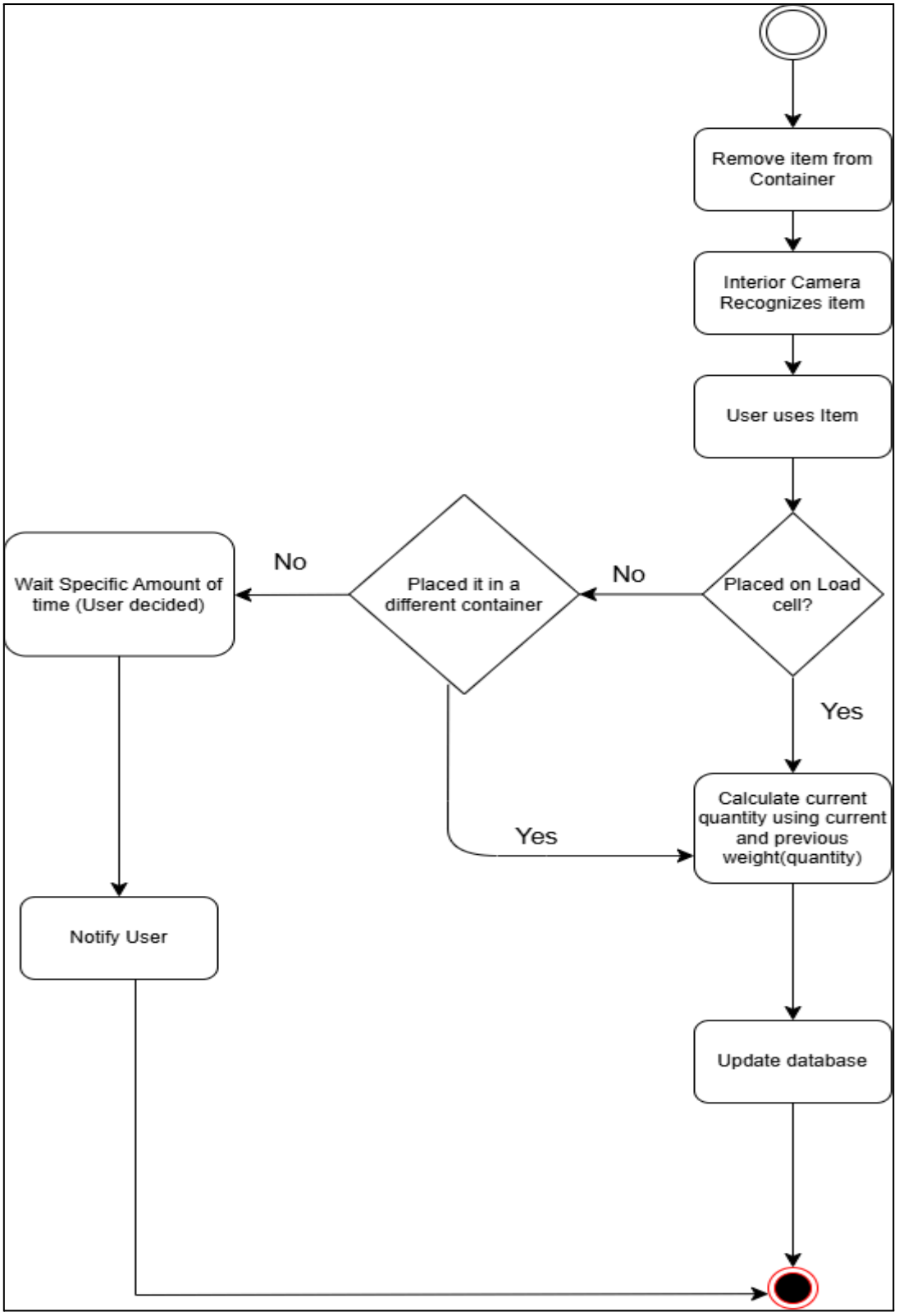


Fig. 7: Usage of Stored Items

This workflow diagrams (Fig. 5 and Fig. 6) represents the flow of instructions in the system. The primary input for the system would be the items. When the user arrives home the system starts mapping out the locations of the items through the following processes. The user first puts the items individually in the designated areas (the cabinets or the fridge). The main camera module tracks the user with the current item at hand and notifies the system of the location chosen; furthermore, the fridge/cupboard camera makes sure that the information is correct. The item's weight is then tracked through both the main scale where the user initially weighed the item and through the load cells present in whichever designated location. To keep track of the percentage of use and/or the item's quantity available.

Using the predesignated database of said items, the expiry date will be set. All the stated information can be accessed and modified by the user (weight, expiry date, type of item) to ensure correctness. When an item nears the expiry date or stock provision, the user will receive a notification of the item, location, and expiry date. The user will then be given at the beginning of the device's installation a survey to input their food preferences, allergies, and cuisine inclination. Based on the assessment, an intelligent recommendation system developed by the team will generate and/or recommend a list of recipes based on available food products in the kitchen; allowing the user to access it via the application.

4.2.3 Additional Flow-Charts

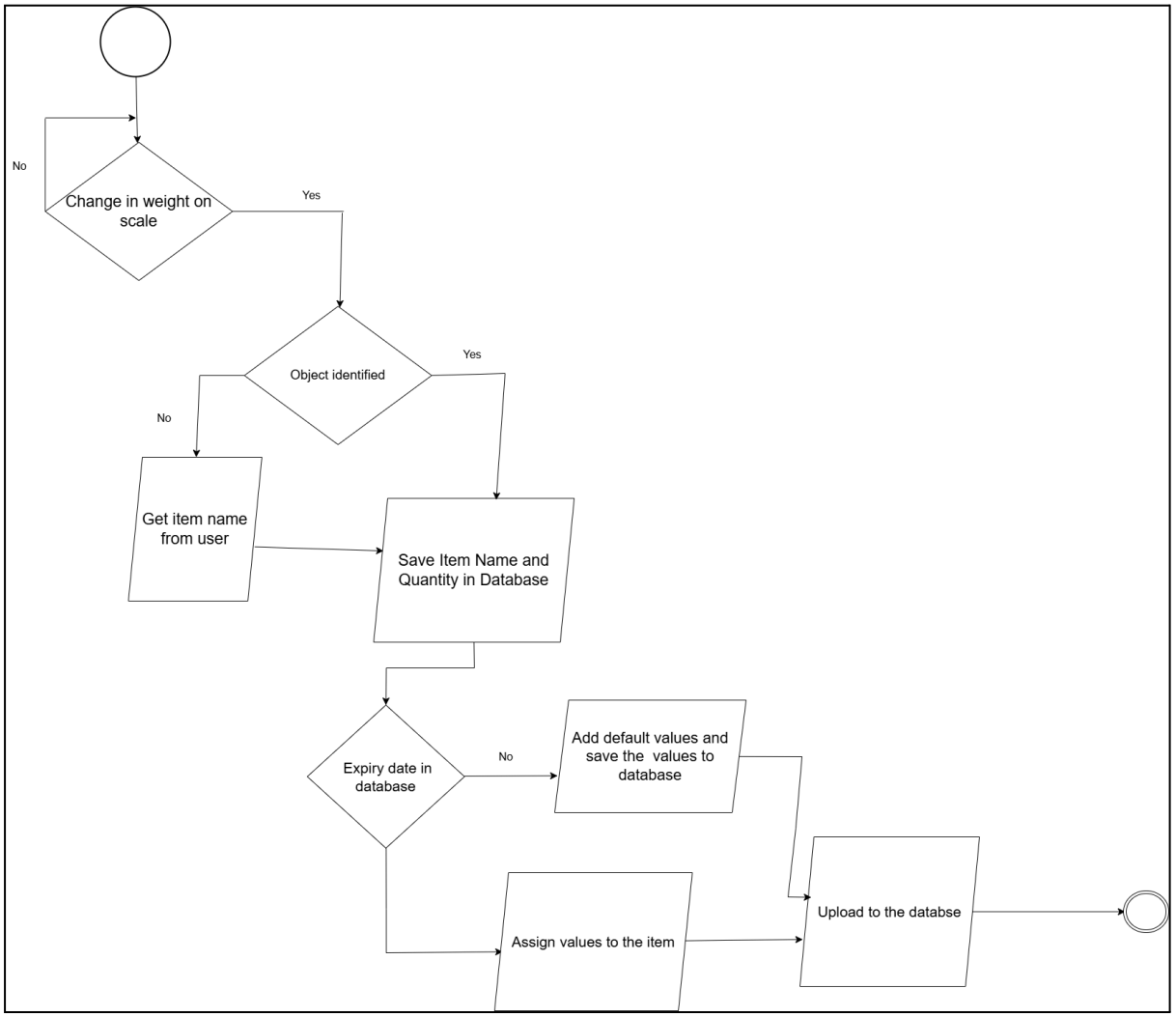


Fig. 8: Flow Chart for Item Use

The above figure (Fig.7) represents the flow of events when an item is placed into the fridge or cupboard in the kitchen. It begins by detecting a change in weight on the pressure pads to start the whole process. The item is then identified and any invalid/incomplete data is input into the system by the user. The system also prompts the user for a restock quantity (if the item doesn't exist in the database already) and adds all the details into the database. The system then goes back to idle until a weight change is detected.

4.3 Alternative Designs

Various modules of our system have been previously implemented in different forms, giving us a broad foundation of alternative design ideas to build upon. This range of possibilities has been further expanded by numerous ideas brainstormed by our team.

4.3.1 Object Identification

Instead of the RFID and NFC stickers used by [5][6][7] and [10], it was decided to use a camera-based object identification method, similar to the implementation done by [8] and [9], as it met our requirements for minimizing user inputs and going for automation. RFID and NFC stickers would require a whole reading and sticker management system, as well as having the user individually tag the item accordingly, which goes against our goal of automating kitchen management.

The implementation of the main camera was first decided to be one central 360-degree camera that handled object identification, tracking, and estimation of measurements placed in the center of the kitchen. However, there were certain limitations to this limitation, as in certain situations this camera may not be able to correctly identify and quantify the objects. For example, if an object is hidden from the camera by the user's body or an object, or when the camera is unable to identify the object at all.

The final implementation which was decided upon was having 1 main 360-degree camera with smaller cameras in each storage area. This implementation was decided as it minimizes the cost and increases both reliability and redundancy in case of any errors.

4.3.2 Quantity Measurement

The two main methods for quantity measurement discussed were Load/Weight sensors and level measurement. However, the level measurement was inefficient for our needs, especially

in the implementations mentioned in [18] and [19]. This was because the implementation required a stick with sensors or markings in order to be used. We decided to implement load sensors and weighing scales for a more accurate and reliable measurement of quantities, as this allows us to store even liquids in the form of weight, instead of having a separate measurement system for it.

In terms of the exact implementation, we had discussed multiple layouts for the load cells and weighing scales. The first implementation was having a central weighing scale that the user had to place items on for the system to identify and receive their weight, however, we found that such a method was inconvenient for the user as it added an extra step when preparing food or adding food to the system (the user would have to place items on the scale after using for the system to take the new value).

The possible alternatives we found for the first implementation were either to have multiple weighing stations in the form of a large pad for ease of weighing items or to have smaller load cells in each cabinet. It was eventually decided that we would have one main weighing station and smaller load cells in each cabinet, this allowed minimal extra steps from the user, while allowing the user to use the weighing scale as needed.

4.3.3 Database Implementation

The database is the backbone of the entire system and it is required to be secure while allowing easy access for the modules to make any modifications in the data. Upon reviewing the different implementations used in [32][23][33][34][35] and [36], i.e. Google Firebase, MongoDB, and MySQL, we decided to use Google Firebase due to how easy it is to use and manage, as well as its sufficient security. Google Firebase also comes with very handy services like Google Authentication, which allows users to directly login using their google accounts instead of having our application manage the usernames and passwords.

4. Implementation

5.1. System Overview

The core computational unit is the Raspberry Pi 5 with AI hat, the central processing hub, interfacing with two Arduino Giga R1's microcontroller responsible for sensor data acquisition. The system is also equipped with advanced image processing capabilities through the RPI Camera running Yolov11s and real-time motion detection via a Eufy 360-tilt camera. Data from all components are transmitted to a Firebase server for centralized storage and further analysis.

The system comprises two storage compartments designed for efficient monitoring:

- Two Cupboards: Each containing one full rectifier load cell, one SHT30 sensor, one MQ-3 sensor, and one RPI Camera.

5.2. System Components

5.2.1. Central Processing Unit

The Raspberry Pi 4B functions as the main data processor, responsible for aggregating sensor readings and transmitting processed data to the Firebase server. It communicates with the Arduino Giga R1 via a serial connection at a baud rate of 115200.

5.2.2. Microcontroller and Sensor Integration

The Arduino Giga R1 acts as the primary microcontroller, interfacing with the following sensors:

- **SHT30 Temperature and Humidity Sensor:** An analog sensor that monitors environmental conditions with an update interval of 0.5s ±0.1s. Connected to a 5V VCC, ground, two PWM digital input pins, and a CLK synchronization line, and has a power consumption of 7.5-10mW when active (2mW when idle).

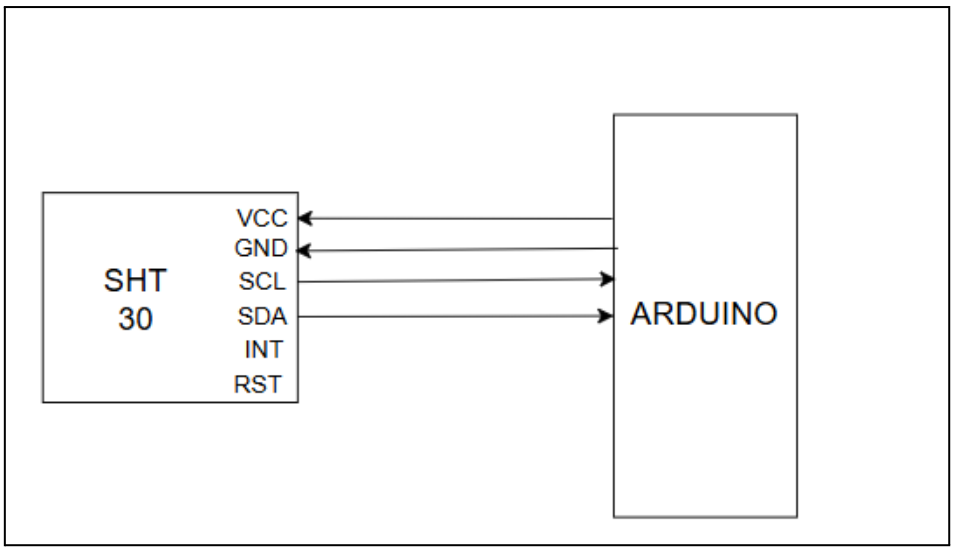


Fig. 9: SHT30 Pin Diagram

- **HX711 Amplifier with Full-Bridge Rectifier Load Cell:** An analog sensor that measures weight after calibration, arranged in a rectangular cross-formation. Connected through the CLK, two digital input ports, and powered via a 5V VCC and ground connection. Has an accuracy of 1g with a threshold of acceptance being 10g. Has a power consumption of 32.5mW when active (15mW when idle).

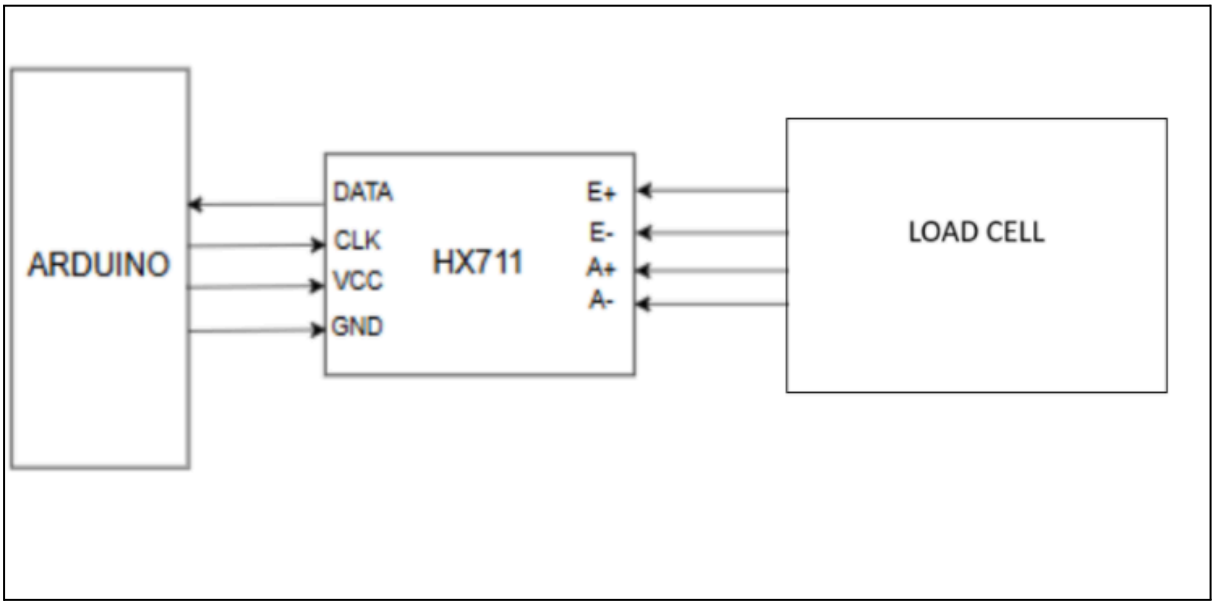


Fig. 10: HX711 & Load Cell Pin Diagram

- **MQ-3 Alcohol Sensor:** An analog sensor that detects the presence of methanol to assess food spoilage, particularly in fruits and vegetables. Connected via a 5V VCC,

ground, one digital input pin, and one analog input pin. With an accuracy rate of 5% with our acceptance threshold being 10%. With an average power consumption of 7.5mW when active (2.5mW when idle).

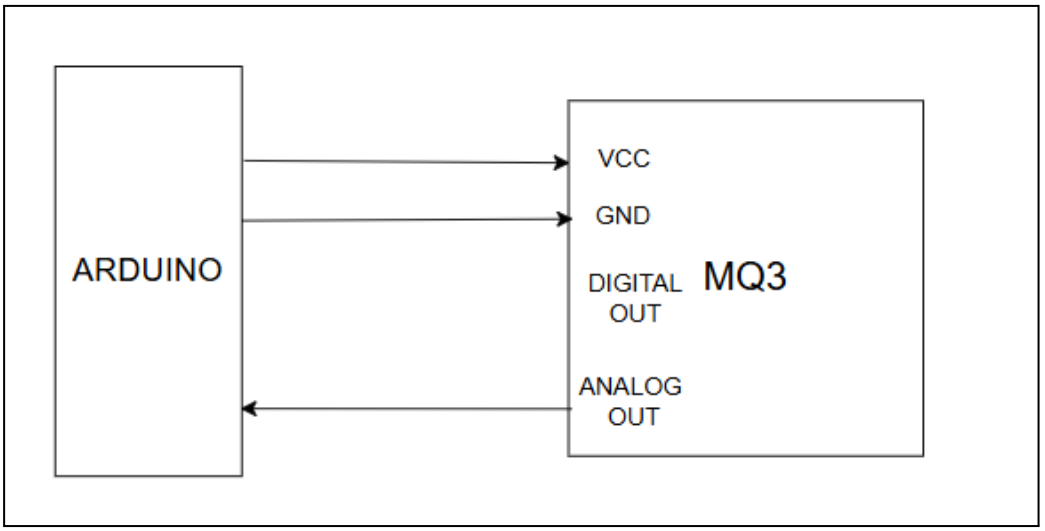


Fig. 11: MQ-3 Pin Diagram

- **MQ-214 METHANE SENSOR:** An analog sensor that its primary directive is to detect the presence of Methane to assess food spoilage, particularly meats, chicken and fish. Connected via a 5V VCC, ground, one 10kOhm resistor, and one analog input pin. With an accuracy rate of 5% with our acceptance threshold bein1g 10%. Has an average power consumption of 8.5mW when active (3mW when idle).

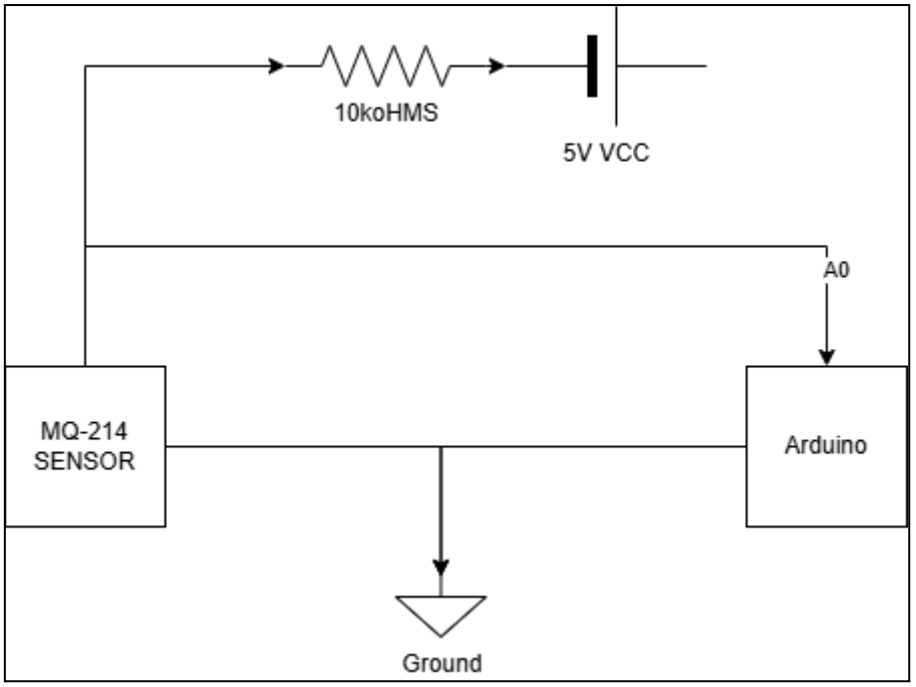


Fig. 12: MQ-214 Circuit Diagram

- **Eufy 360-Tilt Camera:** A digital camera that facilitates system activation through motion detection and assists with weight measurement.. It will be connected to the system through WiFi to the Raspberry Pi. Has an average power consumption of 5W when active (2.5W when idle).

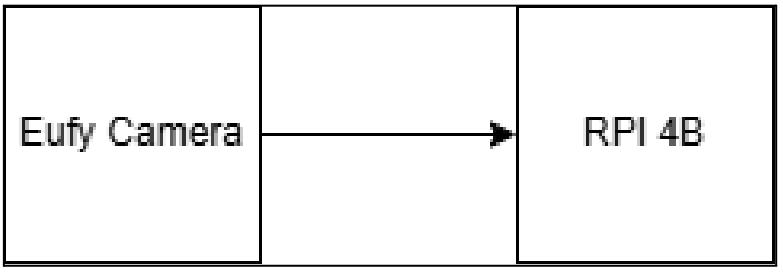


Fig. 13: Eufy Camera Connection

- **RPI Camera Module 3**: This camera module is used for real-time object detection by running a YOLO11 model directly on the Raspberry Pi. It connects through the CSI port, ensuring high-speed data transfer and efficient processing. Has an average power consumption of 450mW when active (and an average of 175mW when idle).

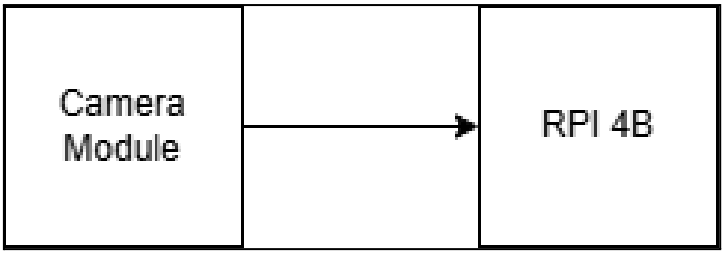


Fig. 14: Camera Module Connection Diagram

- **OV3660 Camera Module**: A camera module used for Object detection tasks, which is processed using a YOLO11 model. It is also connected to the Raspberry Pi via the USB port. Has an average power consumption of 650mW when active (325mW when idle).

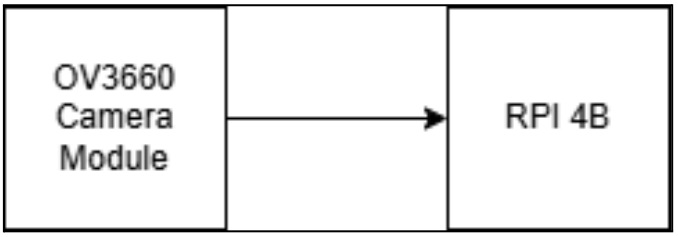


Fig. 15: Camera Module Connection Diagram

5.3. System Initialization & Configuration

5.3.1. Raspberry Pi Interfacing

The Raspberry Pi 4B directly interfaces with the following components over Wi-Fi:

- Eufy Camera: Configured via RTSP for real-time video streaming, facilitating motion detection-based system quantity measurement.
- Arduino: This device represents distinct locations in the kitchen. It has its sensors and feeds data to the Raspberry Pi for processing.

- Data Transmission: A dedicated Python script is deployed on the Raspberry Pi to collect and process object data from the RPI Camera lens and the Arduinos. This data is transmitted to the Firebase server for storage and syncing with the Flutter application.

5.3.2 Hardware Layer and Edge Processing

The Arduino will read the data from the sensors and send it to Raspberry Pi, which then uses a Python script to process the sensor data, including our generated information from our Yolov11s model for food identification (if needed), and send the formatted data to Firebase every 3 ± 2 seconds.

5.4. Software Architecture Design:

The Smart Kitchen requires modular and scalable architecture that ensures efficient real-time data handling from sensors, seamless communication between the Raspberry Pi and Firebase, and a clean Flutter app to display the data stored on Firebase.

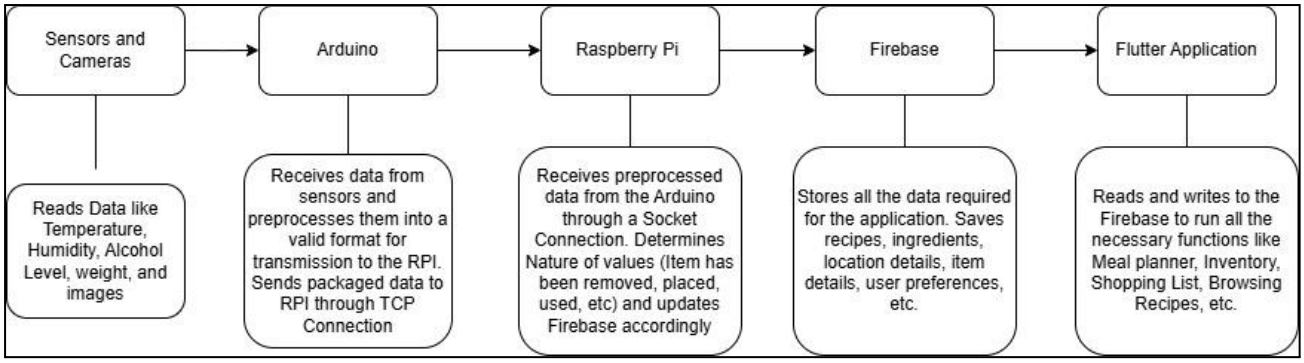


Fig. 16: Data Flow in System

Components:

- **Sensors (Load Cells, Humidity and Temperature, Alcohol, Methane, and Cameras):** Identify, Measure and Track food data. Sends collected data to Arduino.

- **Arduino:** Collects sensor data and sends it to Raspberry Pi. Used to increase modularity and help organization.
- **Raspberry Pi:** Processes data from the Arduino and uploads to Firebase using a python script.
- **Firebase (Backend):** Stores inventory data uploaded from the Raspberry Pi and syncs it with the Flutter App.
- **Flutter App:** Fetches real-time data and provides user interaction, through both laptop and mobile devices.

Layer	**Technology**	**Purpose**
Hardware Layer (Code)	Arduino + Sensors	Collect raw sensor data
Edge Processing	Raspberry Pi (Python)	Processes data and uploads to Firebase
Backend	Firebase	Store and manage food inventory, recipes, shopping lists, etc.
Frontend	Flutter	Display inventory, recipes, meal plans, shopping lists, etc.

Table. 9: Software Layers

5.5 Hardware Layer (Code):

The hardware layer consists of the following components which have their own code to read and send the data to the arduino periodically:

- **SHT30 (Humidity and Temperature) Sensor:**

Arduino code which periodically reads and prints the Humidity and Temperature in the storage location (cabinet, fridge, etc) the sensor is in. Flow chart of the code can be seen in fig. 6. The code first initializes the sensor before reading the Temperature and Humidity values. It also sends any error that occurs when initializing the sensor to the arduino as well.

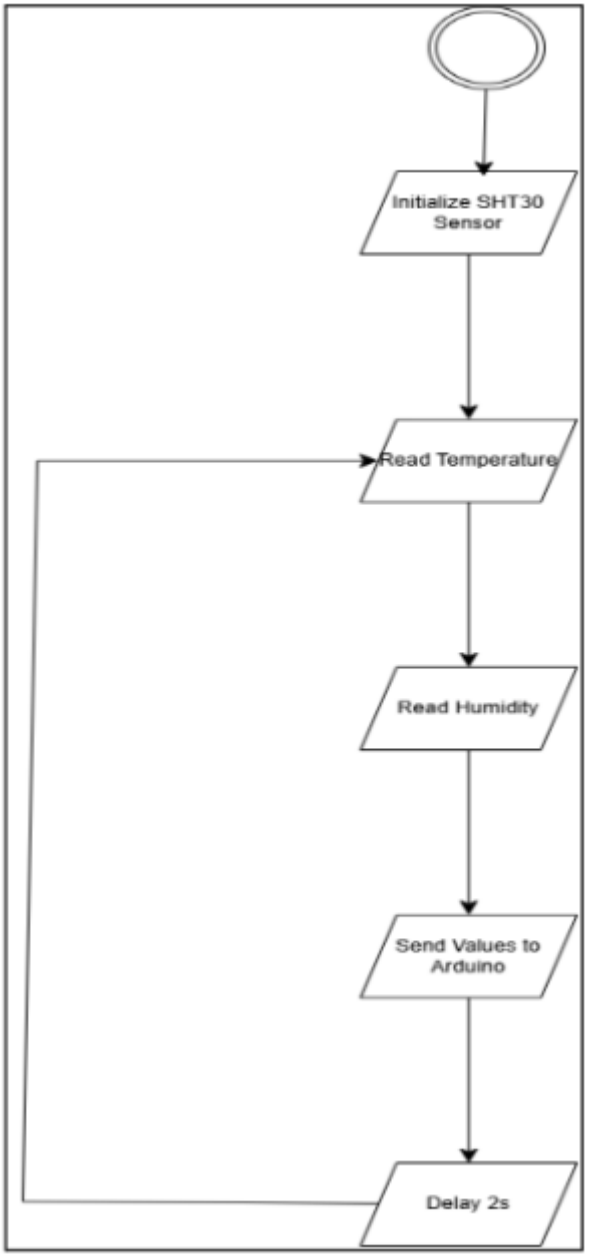


Fig 17: Code Flow Chart for SHT30 Sensor

- **MQ - 3 (Gas) Sensor:**

The MQ - 3 Sensor reads the alcohol and ethanol levels in the storage location and sends the data to the connected arduino. The Flow Chart for the code can be seen in fig. 7. The code initializes the sensor and reads the Analog and Digitals values before sending them. The Analog value is the alcohol level in the surroundings and the digital value is a flag that indicates detection of spoilage. This code will send this information to the Arduino.

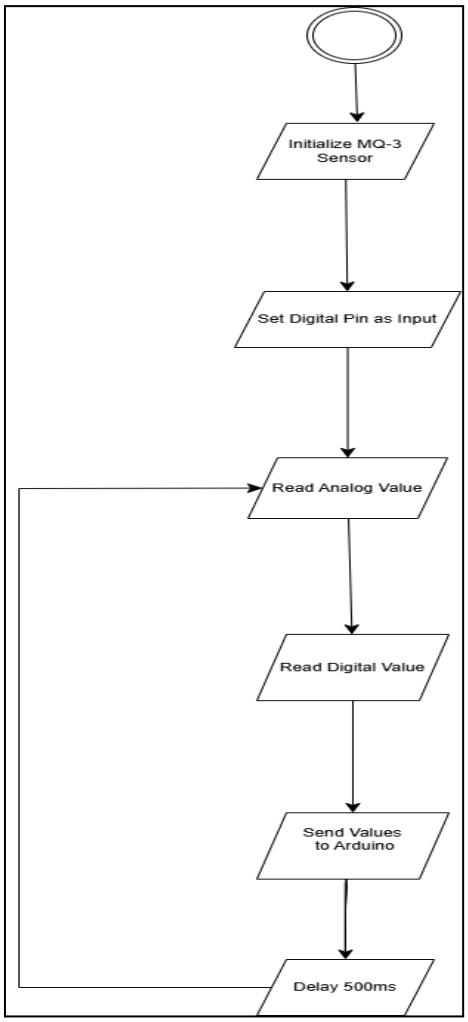


Fig. 18: MQ-3 Sensor Flow Chart

- **HX711 Amplifier and Load (weight) Cell:**

The Load cell and HX711 Amplifier helps to read and store the weight of objects placed on it. The fig 8 below, shows a sample of how it will work, and sends information to the Arduino. The current code initializes the sensor and goes through a calibration process to

ensure accurate readings on the scale. After this, it runs a loop which reads and sends the weight on the scale to the Arduino.

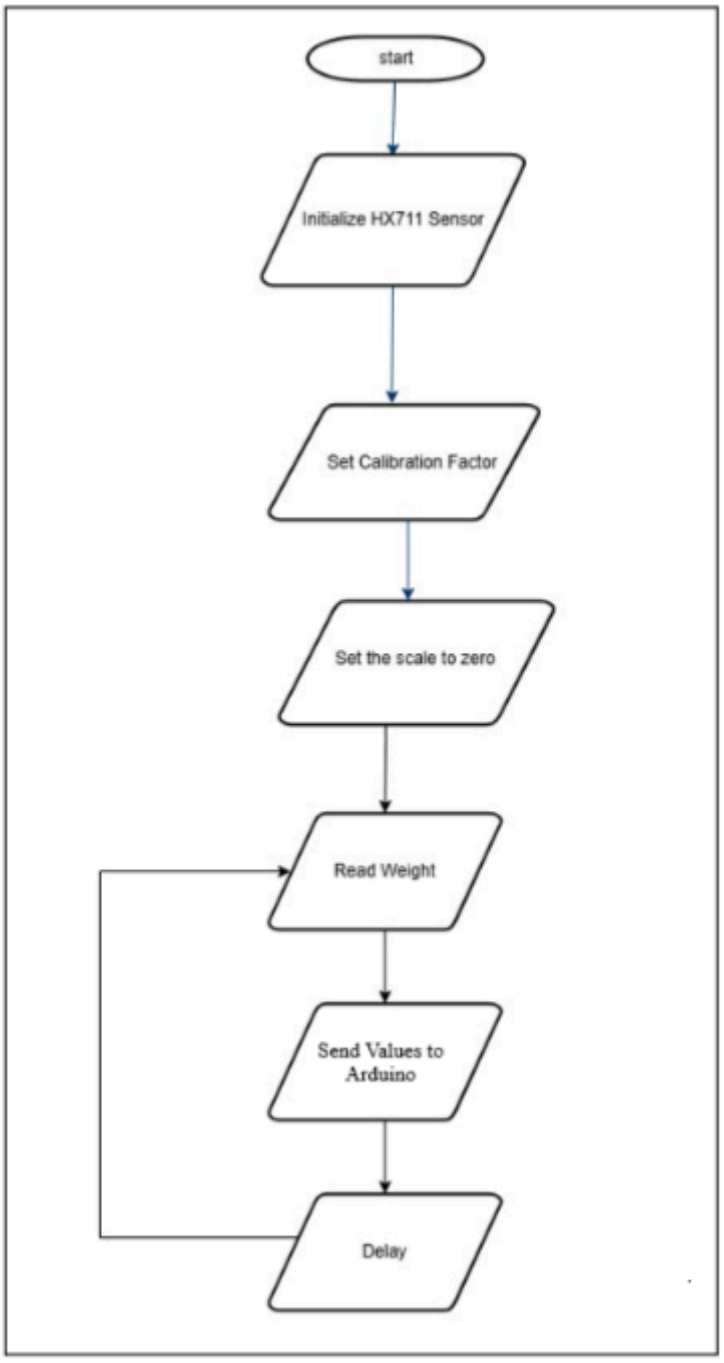


Fig 19: HX711 Flow Chart

- **RPI Cameras (USB and Module 3 Object Identification and Tracking):**

The Cameras are used to identify the objects in the storage location and send this information to the Arduino. Fig. 9 below shows the flow chart of the code used to identify the items via the collected dataset used to train the Yolov11 model.

5.6 Edge Processing:

The data sent by the Arduino will be processed in the Raspberry Pi and updated in the Firebase database. Fig.18 below shows a sample of how the code will work. When data is received from the Arduino, the Raspberry Pi will run a python script that will package and send the data to the database and ensure the data was sent. If the data fails to send more than three times then it will be considered a transmission and the system will continue to the next set of data instead.

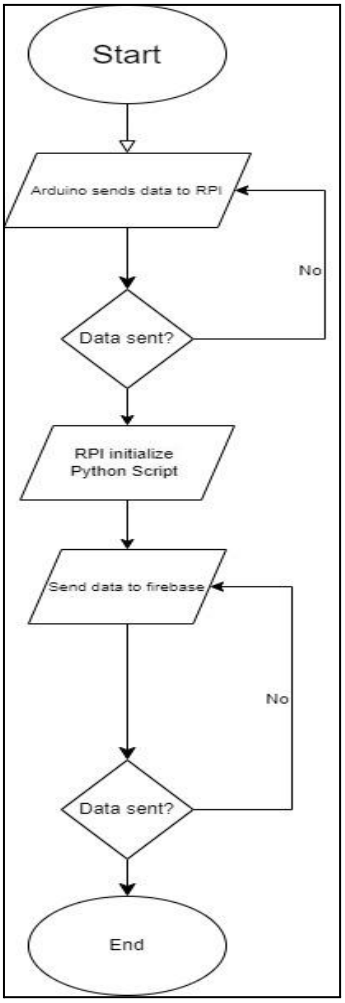


Fig. 20: Raspberry Pi Upload Flow Chart

5.7. Backend Layer (Firebase):

Firebase is a NoSQL document-based database, so we will use collections and subcollections for structured data storage. Each user will have personalized data, which is the main objective for the design in Fig. 19.

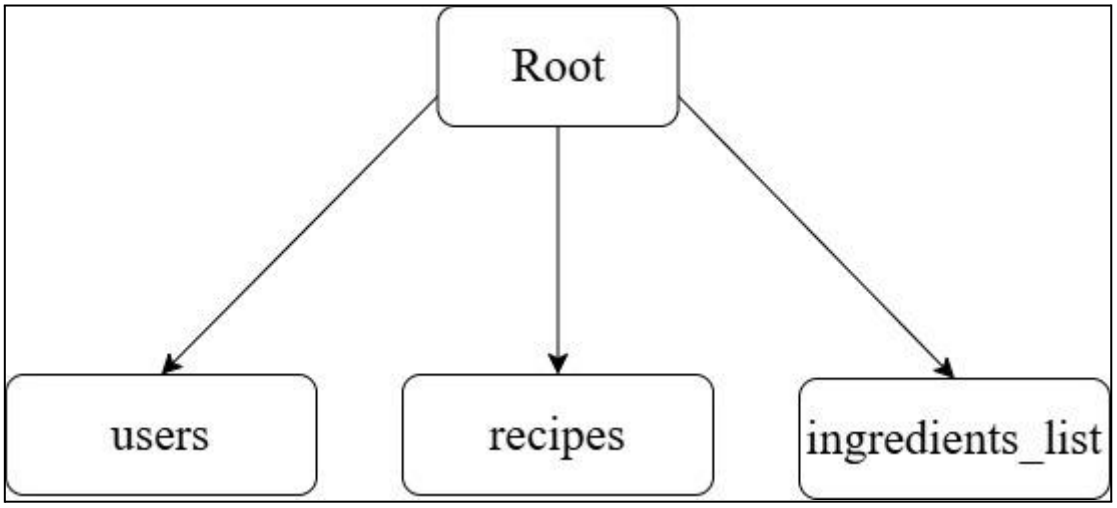


Fig. 21: Overall Database Layout

In the figure above, the users collection is responsible for storing all user-specific data and preferences. In contrast, the recipes and ingredients_list collections manage system-wide information, encompassing all available recipes and ingredients within the system.

The ingredients_list collection, as depicted in Figure 20, holds comprehensive nutritional data for each ingredient. This includes values for calories, carbohydrates, cholesterol, fat, fiber, protein, sodium, and sugar, all standardized per 100 grams. Each ingredient entry is uniquely identified by an ingredientId, ensuring consistent referencing across the system.

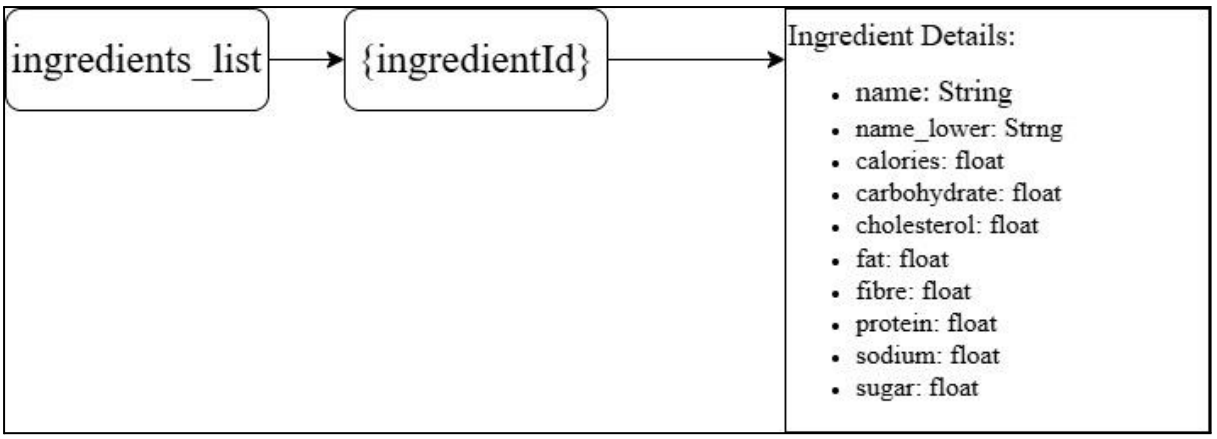


Fig. 22: Ingredient_list Collection Layout

The recipes collection contains comprehensive information for each recipe, including its ingredients and associated user reviews. As illustrated in Figure 21, each recipe document stores detailed metadata such as nutritional information (e.g., calories, fat, protein), number of servings, yield, preparation and cooking instructions, total preparation time, images, and expiration duration (in days). It also includes the number of reviews and the aggregated rating score. Additionally, the recipe stores a list of its ingredients along with their corresponding quantities in grams.

User reviews are stored within each recipe using the user's unique ID as the document identifier for the review. This structure ensures that each user can submit only one review per recipe, while still allowing them to update or modify their review at any time.

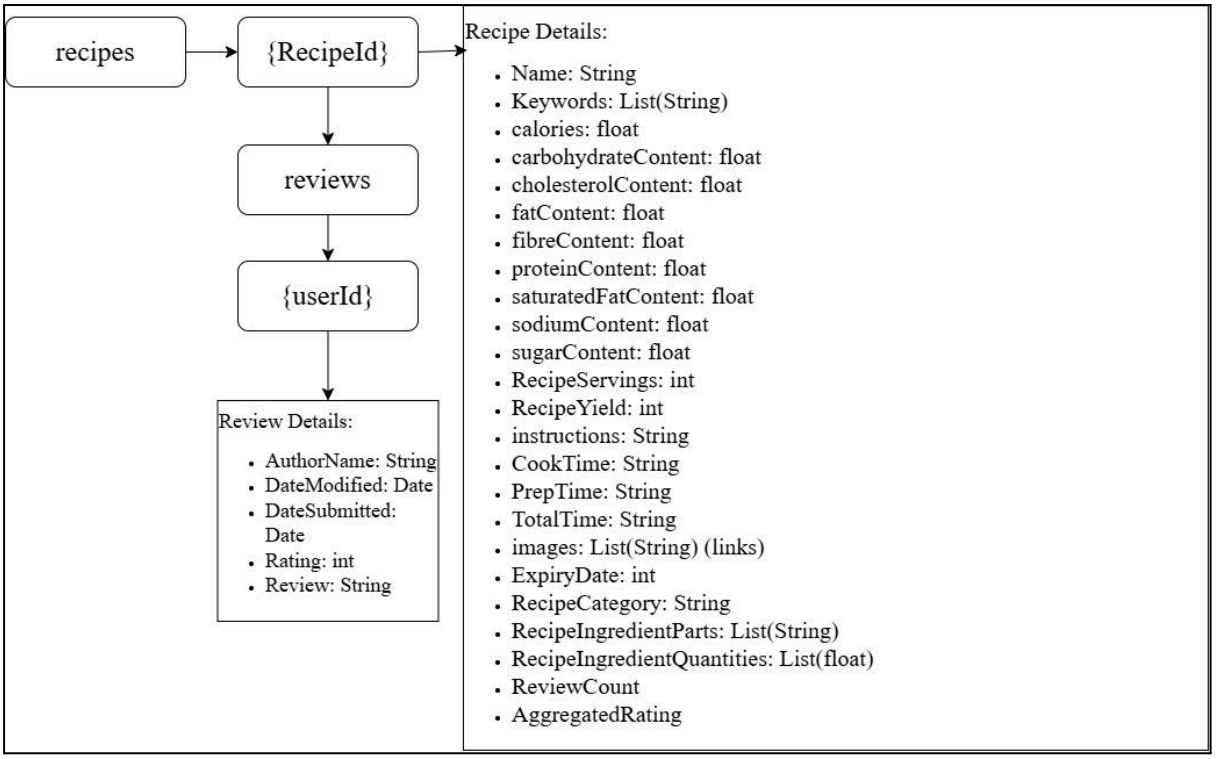


Fig. 23: Recipe Collection Layout

The users collection contains all user-centric data. As illustrated in Figure 22, each user is represented by a unique userId, which is generated via Google Authentication, ensuring a consistent and secure identifier linked to the user's account.

Each user document branches into six dedicated subcollections, allowing for clean segmentation of user data:

- **inventory**: Stores detailed information about the user's items.
- **locations**: Contains data about various storage locations associated with the user.
- **alerts**: Holds user-specific alerts and notifications.
- **meals**: Maintains data relevant to the meal planning functionality.
- **preferences**: Stores the user's personal settings and dietary preferences.
- **recipes**: Keeps track of recipe IDs that the user interacts with or saves.

This structure supports a modular, scalable design, enabling efficient data retrieval and personalized experiences within the application.

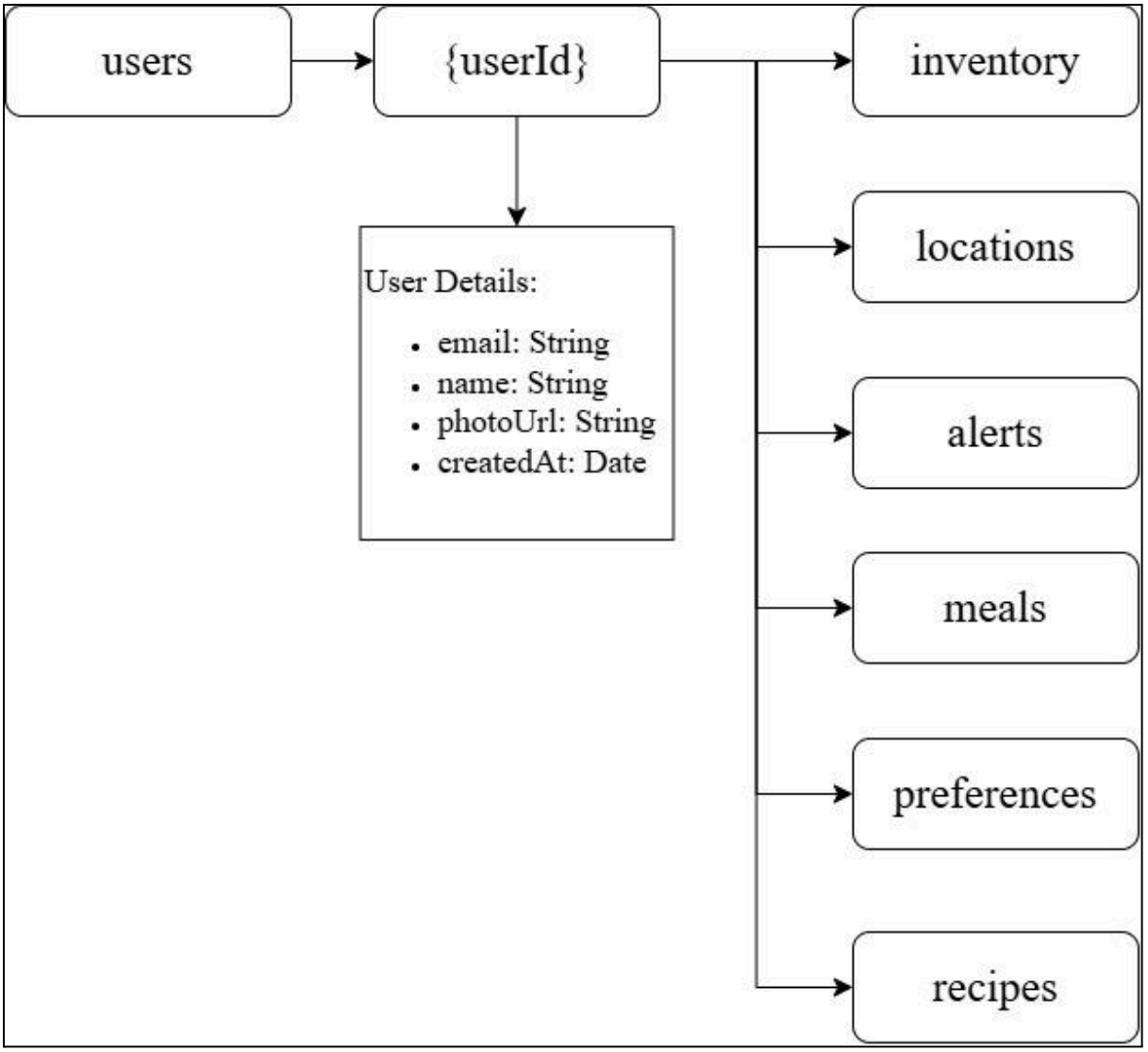


Fig. 24: Users Collection Layout

The system architecture begins with the inventory collection, where each item is stored using a unique identifier in the format {itemId}_{locationId}. This naming convention enables the system to differentiate between instances of the same item stored in different locations, treating each as a distinct entry. For instance, if apples are stored in two separate locations, they are recorded as two individual inventory entries.

Each item document contains detailed information, including the item name, weight, category (either meal or ingredient), a flag indicating whether the expiry date is estimated, the storage location, and a reminder date that determines when a notification should be sent regarding the item's expiration.

The locations collection stores data for each unique locationId, capturing a comprehensive set of attributes such as the location name, alcohol level, humidity, temperature, current weight, category, and a live image feed (if available). This collection essentially aggregates all sensor data relevant to each storage location, supporting real-time monitoring and analysis.

In parallel, the alerts collection manages system-generated alerts. Each alert entry includes a subject, message, alert type (e.g., item, location, or expiry), urgency level (ranging from default to high), the timestamp of creation, and a flag indicating whether the alert has been read which is used primarily for user interface tracking. An optional id field may also be present to associate the alert with a specific item or location, depending on the context.

An overview of these data structures and their interrelationships is presented in Figure 23.

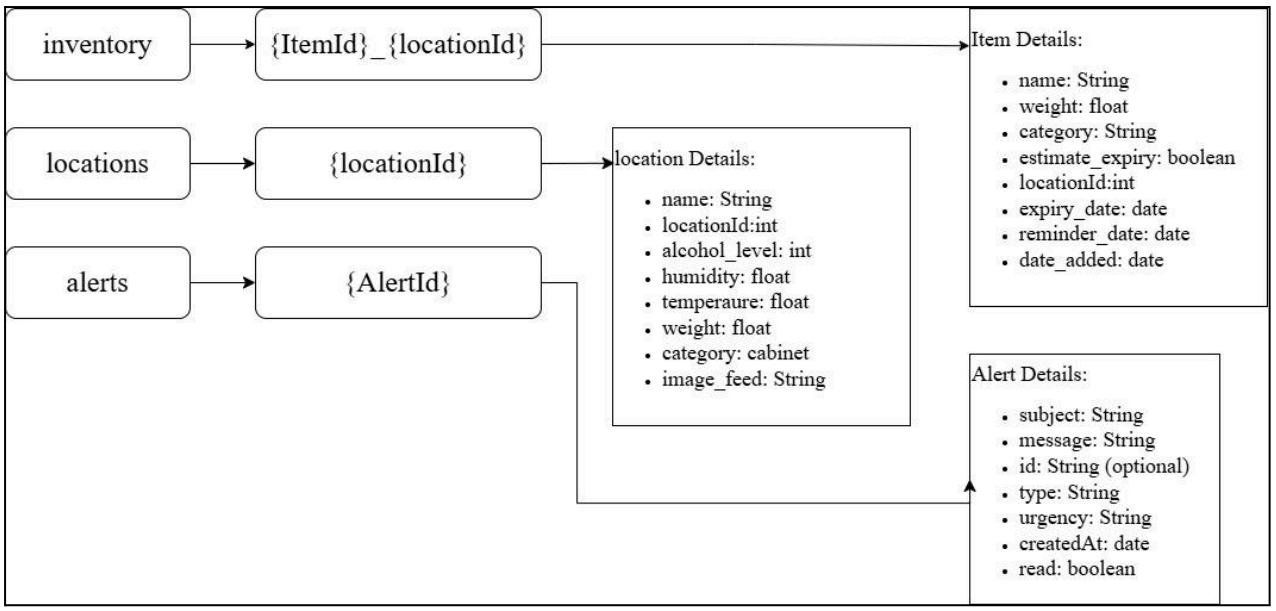


Fig. 25: Users Collection Layout

Each userId also maintains a meals collection, which contains entries for each day of the upcoming week. Each daily entry holds a structured list of meals (e.g., breakfast, lunch, dinner), with each meal referencing a list of recipeIds. This design allows the application to efficiently retrieve and display the corresponding recipe details when planning or preparing meals.

The preferences collection enables users to personalize their experience by setting custom expiry durations for specific items and specifying how far in advance they wish to be reminded about upcoming expirations. This ensures timely notifications that align with individual user preferences.

The recipes subcollection under each user document supports the recipe-related features of the application. Instead of duplicating full recipe data, this collection stores only the recipeIds, minimizing redundancy and optimizing storage. Recipes can be organized into different subcollections such as saved, reviewed, viewed, and created. The saved collection allows users to categorize and organize their favorite recipes, while the reviewed and viewed collections track user interactions for improved personalization. The created collection stores

recipes authored by the user, enabling them to edit or manage their own contributions to the system.

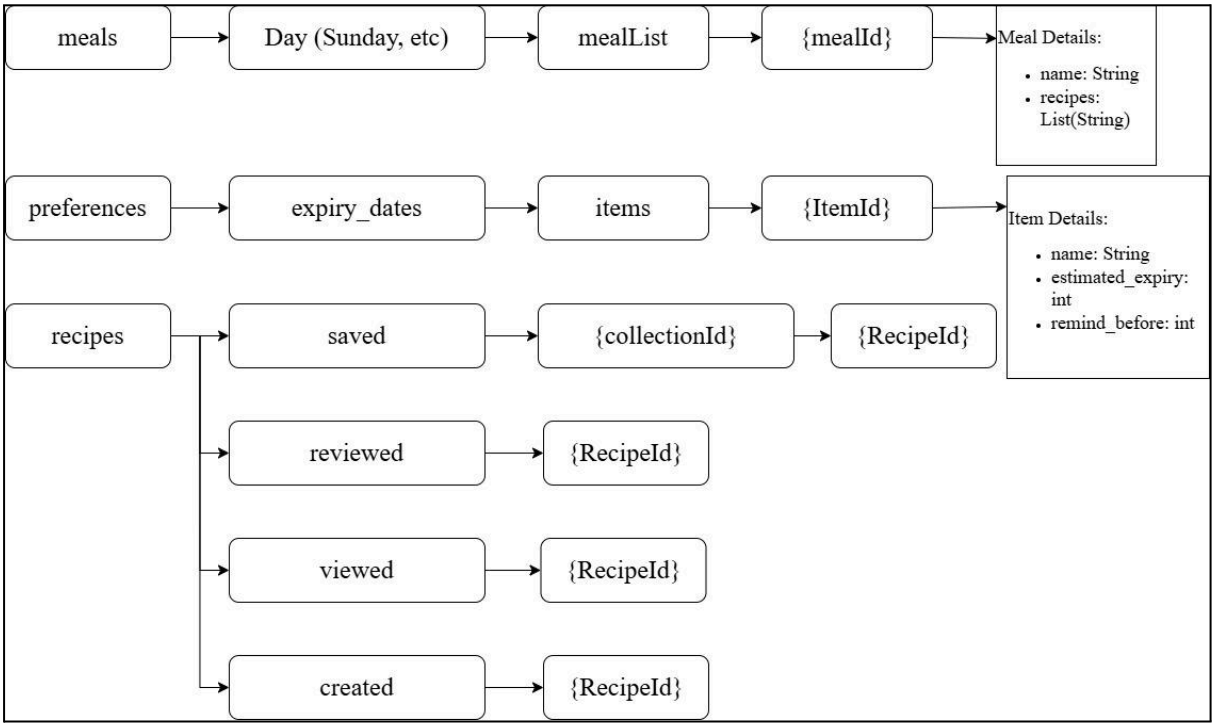


Fig. 26: Users Collection Detailed Layout 2

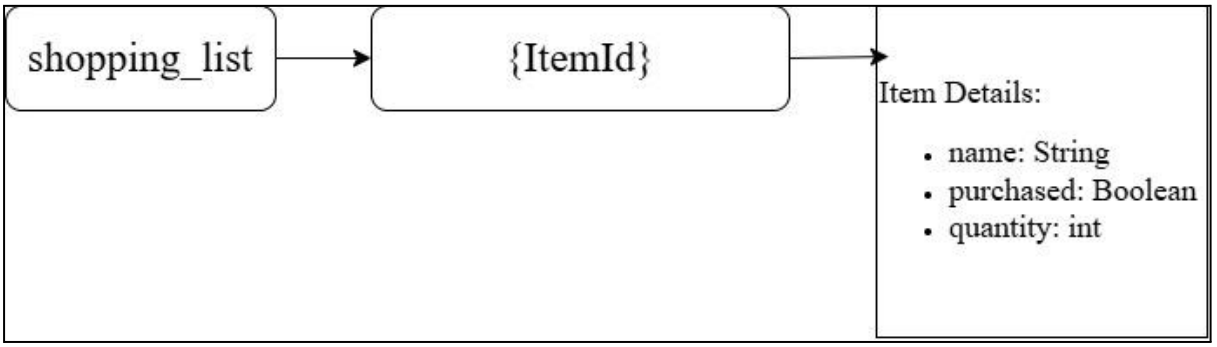


Fig. 27: Users Collection Detailed Layout 3

5.8 Frontend Layer (Flutter):

The application will fetch real-time data from Firebase and display the information to the user in a convenient and timely manner. It will also incorporate push notifications to cover various aspects, including inventory updates, recipe recommendations, and real time system alerts. Users will be notified when food items are nearing expiration, or have expired.

The front end is developed using Flutter and consists of a mobile application for individual users, which also provides information while offline.

The mobile application enables users to track their kitchen inventory, receive real-time updates, and browse or filter food items.

Both ends will follow the MVVM (Model-View-ViewModel(Monitor)) pattern to ensure a well structured, maintainable, and scalable application. Firebase will be used for real-time data fetching, enabling seamless synchronization across devices.

Fig. 28 shows the pages that the user can access in the application after logging in. Although certain routes such as returning to the home page from any page have been emitted, main pages will allow you to navigate to other main pages through the navigation bar.

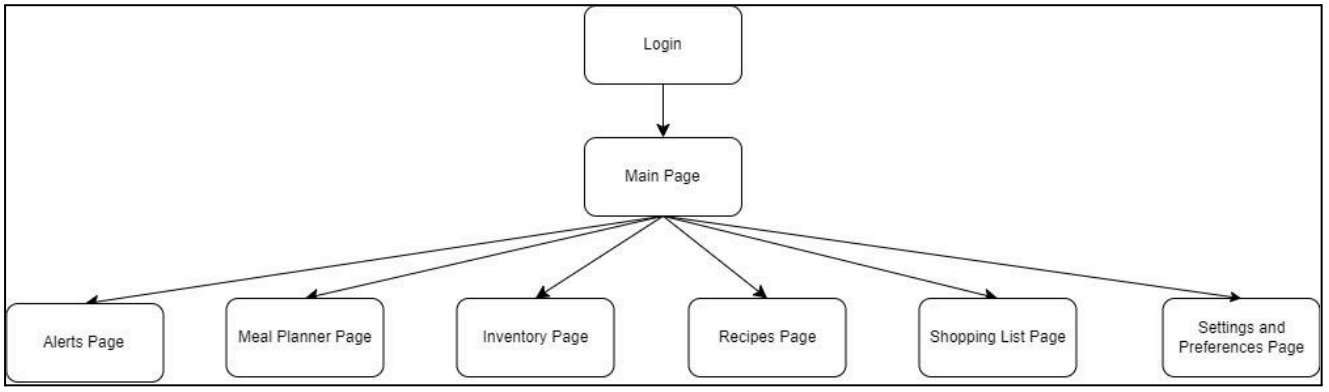
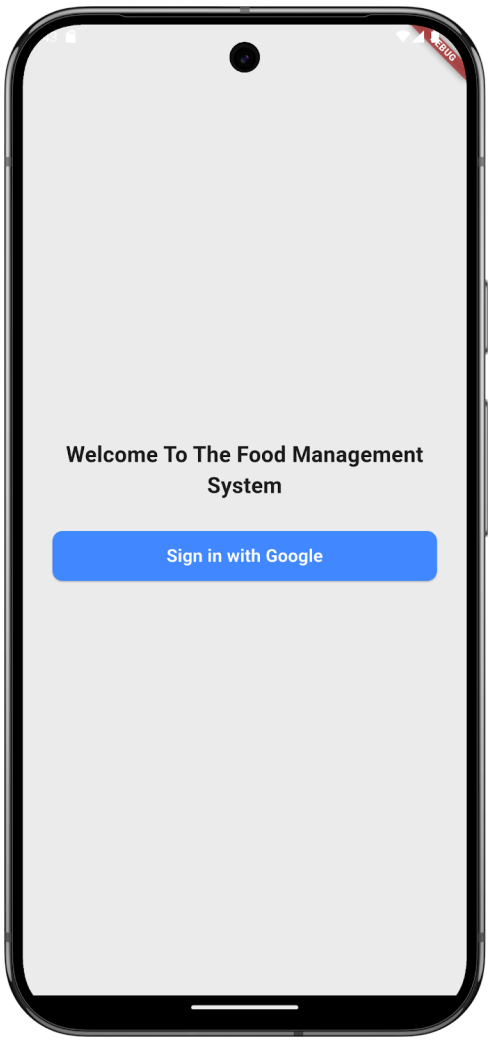


Fig. 28: Overall Frontend Pages/Flow



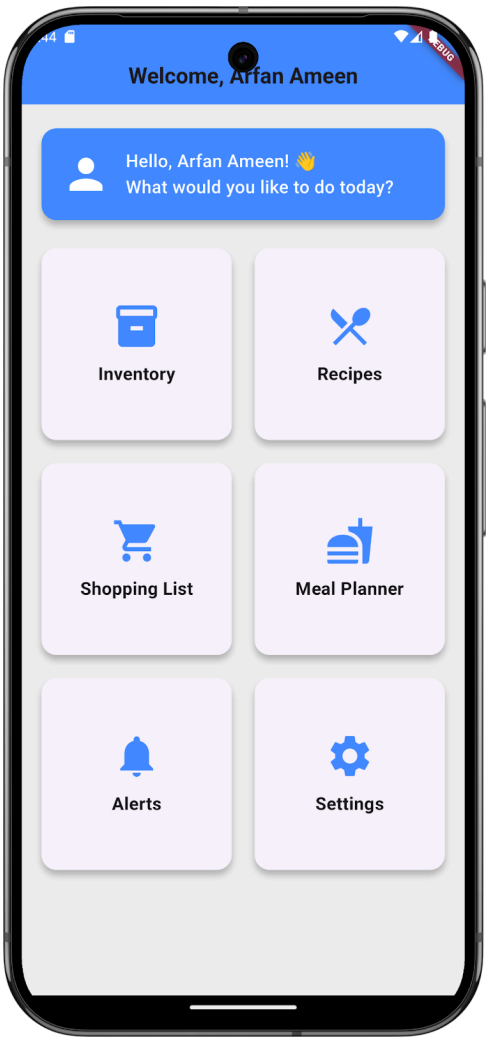


Fig. 29: Sign-in Page **Fig. 30: Main Page**

The approximate functionality of each page can be found in the following figures:

- Alerts Page: Allows the user to view or dismiss alerts, as well as add any reminders based on the alert. For example, setting a reminder to add milk to the shopping list as it is expiring soon.

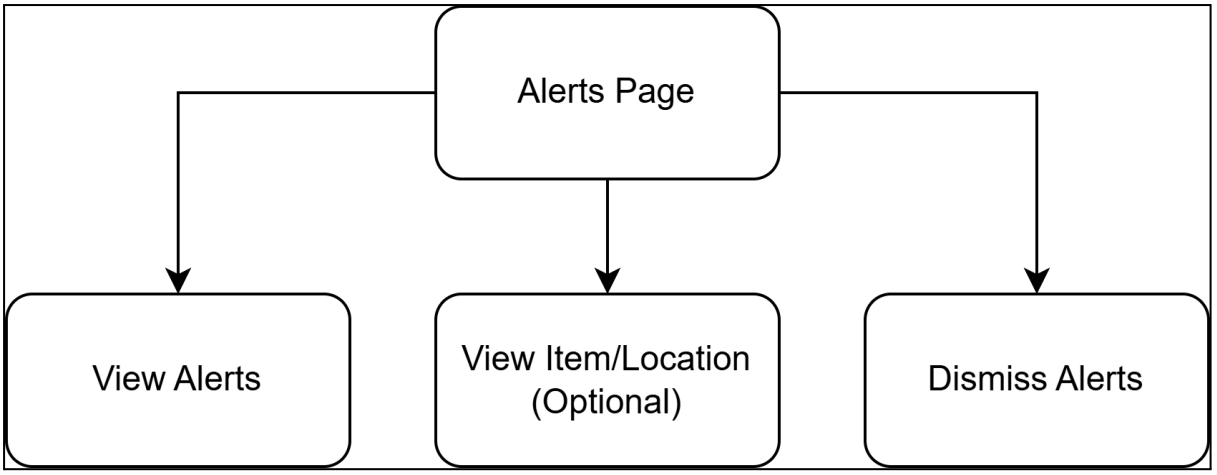


Fig. 31: Alerts Page Functionality

As you can see, the Alerts page in Fig. 32 shown with sample data is colour coded with the urgency of the notification, and highlights if unread. It also displays a short part of the message and the subject of the alert to give the user an idea of the notification. Upon clicking the alert, it takes the user to the detailed alerts page, which shows the whole message and other parts of the alert. There are also buttons to take the user to the item/location, or delete the alert.

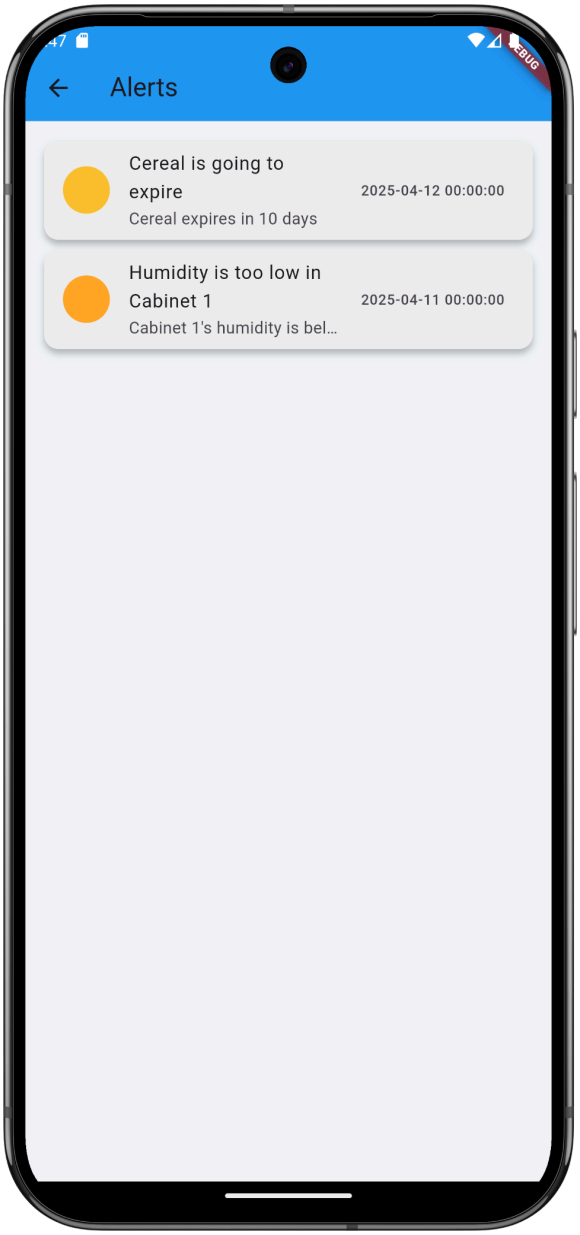


Fig. 32: Alerts Page

Fig.33: Detailed Alerts Page

- Meal Planner Page: Allows the user to add or remove saved recipes to the weekly meal plans. It will also allow the user to browse online recipes which will be recommended to them.

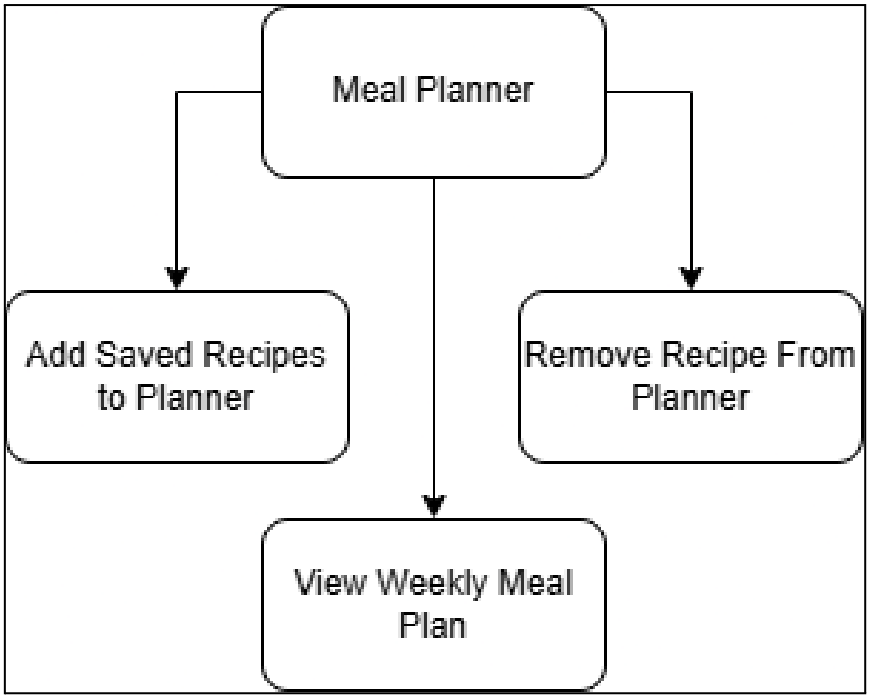


Fig. 34: Meal Planner Page Functionality

As seen in Fig. 34 the meal planner page displays the days for the next week. This allows the user to plan their next week. Upon clicking a day, they are able to make the meals which go into that day, For example, Breakfast, Lunch, Dinner, snacks, etc. Clicking a meal takes them to a detailed meal page as seen in Fig. 35, which allows the user to add recipes to that meal from their saved recipes.

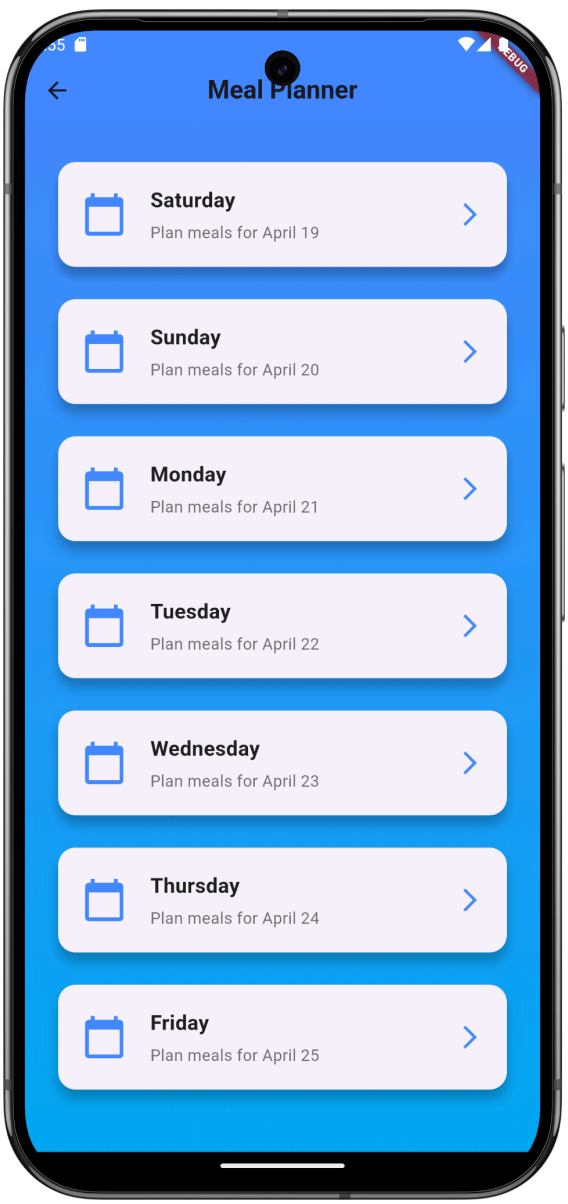


Fig. 35: Meal Planner Page

Fig.36: Meal List Page

- Recipes Page: Allows the user to browse recipes recommended to them by the recommendation system. It also allows the user to save and rate the recipes, as well as add their own recipes to the system.

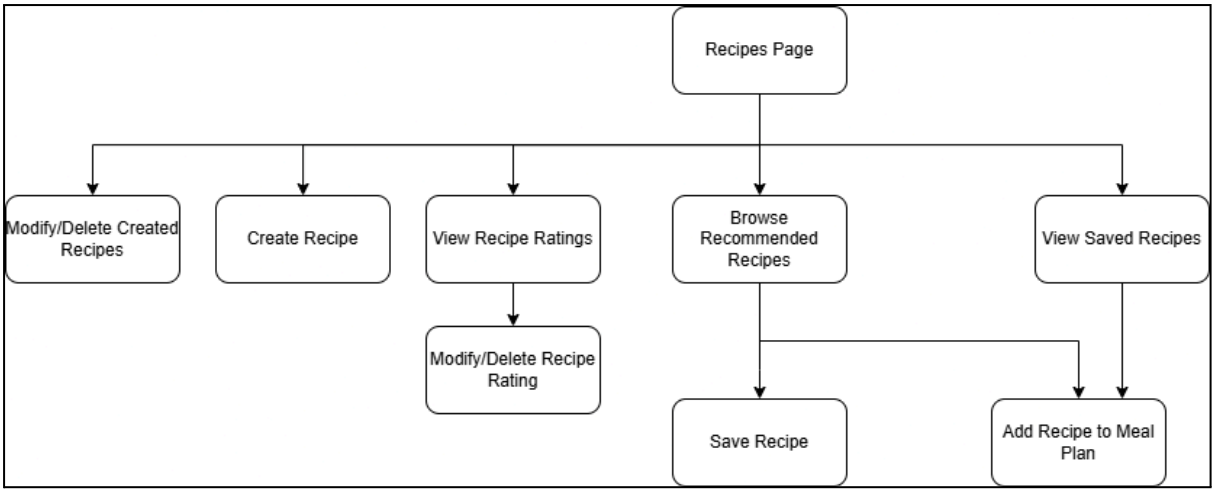


Fig. 37: Recipes Page Functionality

The Recipes Page allows you to view your saved recipes, explore recommended recipes, rate them, and add your own creations to the database. You can create new recipes by searching through our database for available ingredients and their nutritional information, then incorporating them into your custom recipes. Users can also modify or delete their created recipes, view and update recipe ratings, save recommended recipes, and conveniently add any saved or recommended recipe directly to a meal plan. A detailed view can be seen below in Fig. 38 and Fig. 39.

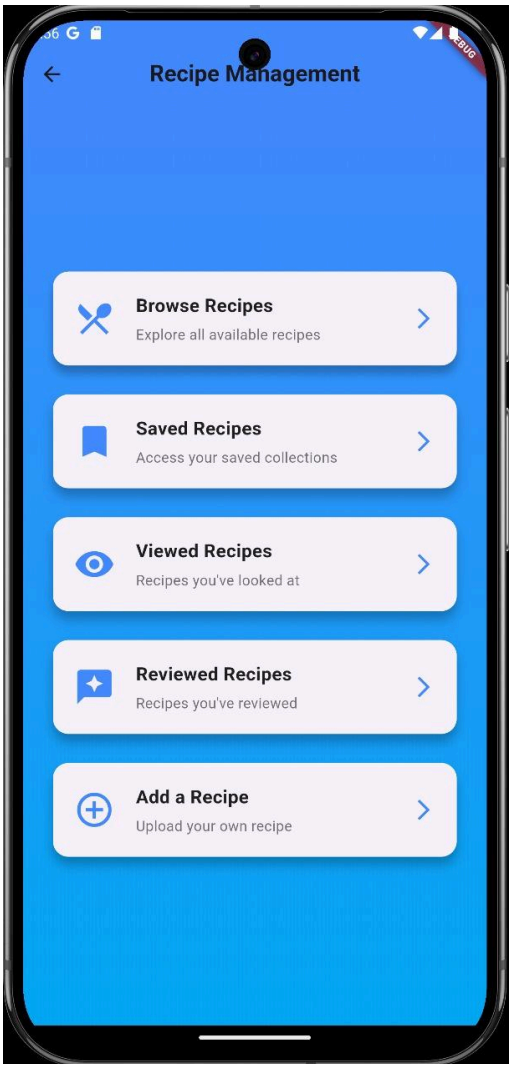


Fig. 38: Recipes Page

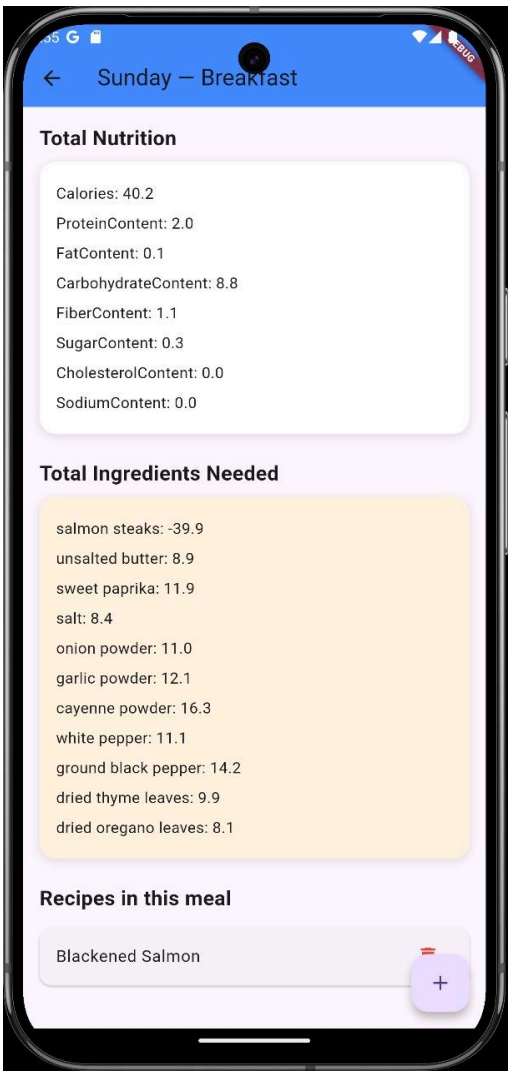


Fig. 39: Adding Recipe Page

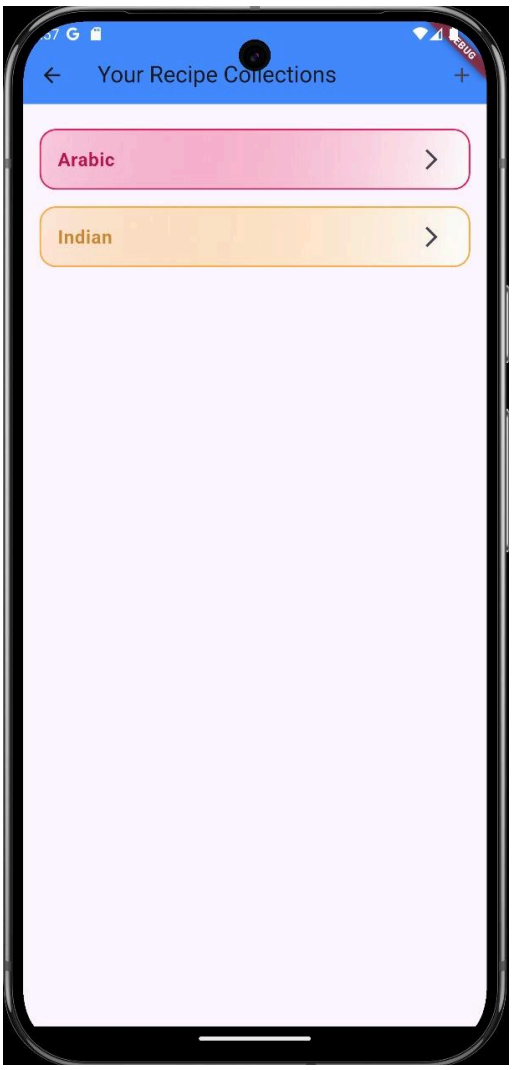


Fig. 40: Saved Recipes (Collections) **Fig.41: Recipe Details Page**

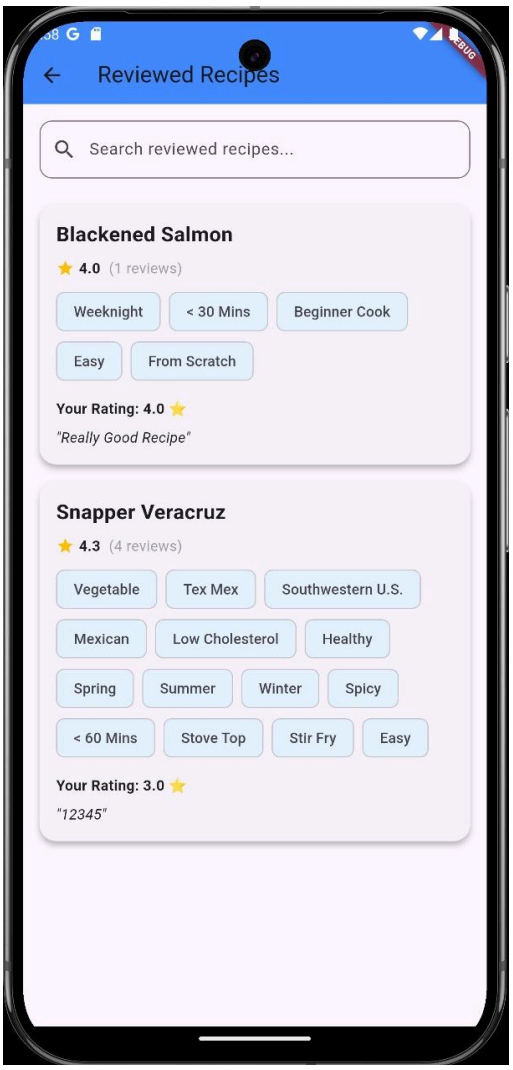


Fig. 42: Reviewed Recipes Page

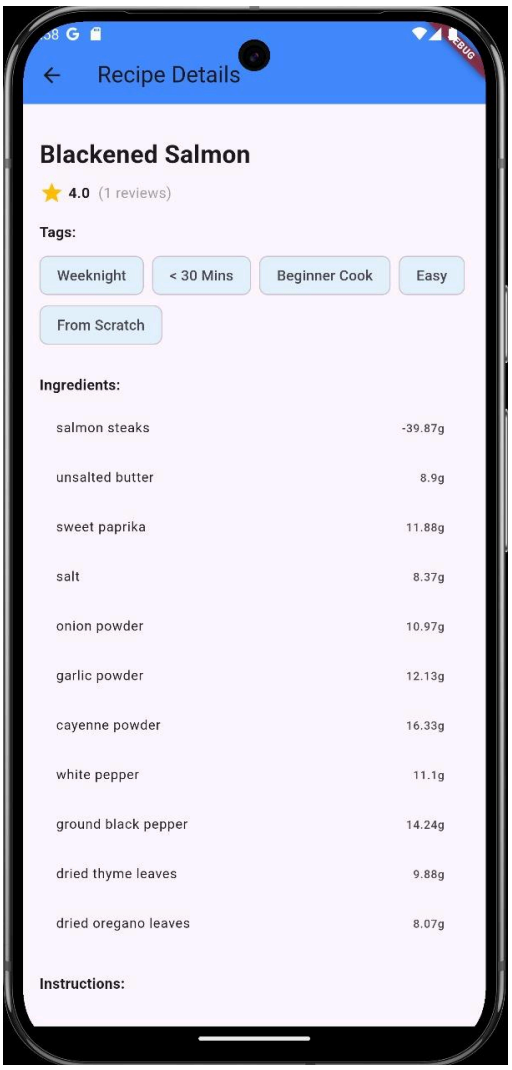


Fig.43: Recipe Details Page

- The Inventory Page allows the user to view and manage the overall inventory in the kitchen, including organizing items based on their storage locations. Users can access detailed information about each storage location and view specific details for every item. Through the same page, users can also access comprehensive reports such as the Expiration Log, which highlights items nearing expiration; the User Interaction Log, which records all user actions such as moving, using, or adding items to the shopping list; and the Item Log, which tracks items added to or removed from the kitchen. Each log records the associated timestamps of these events. Additionally, users can directly

view a consolidated list of all items in inventory from this page, ensuring efficient tracking and management. A visual representation is provided in Fig. 44.

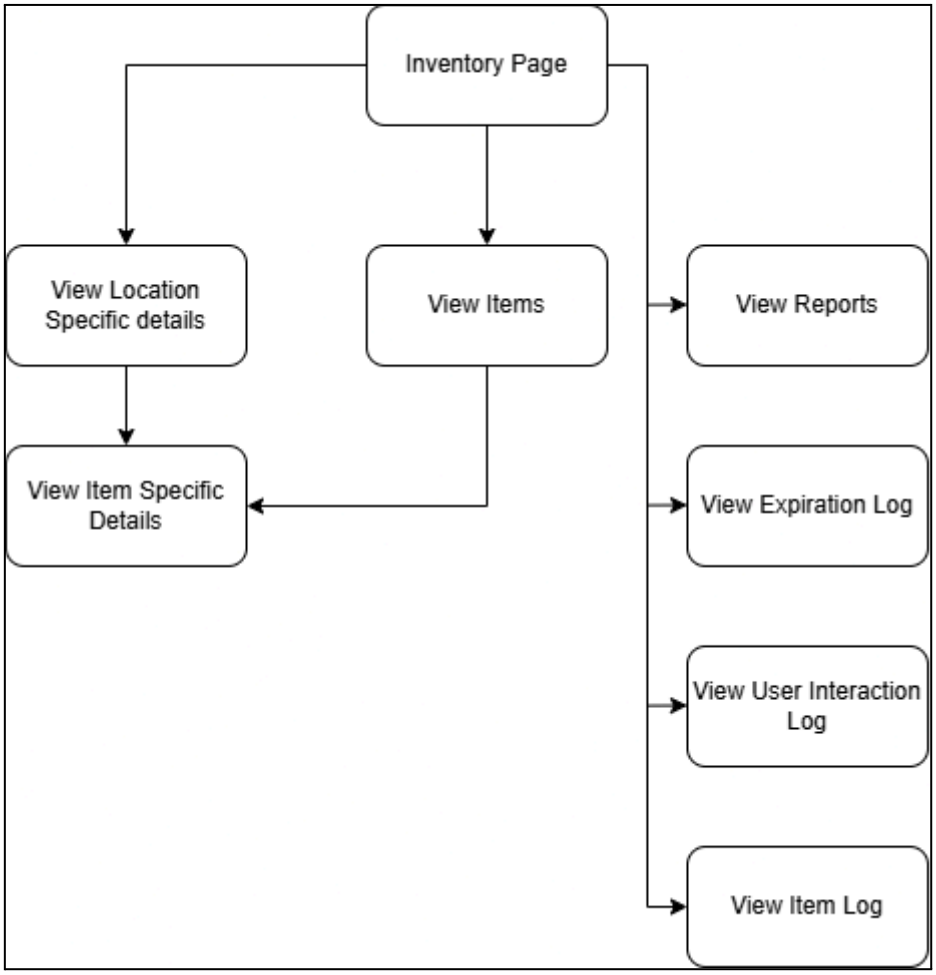
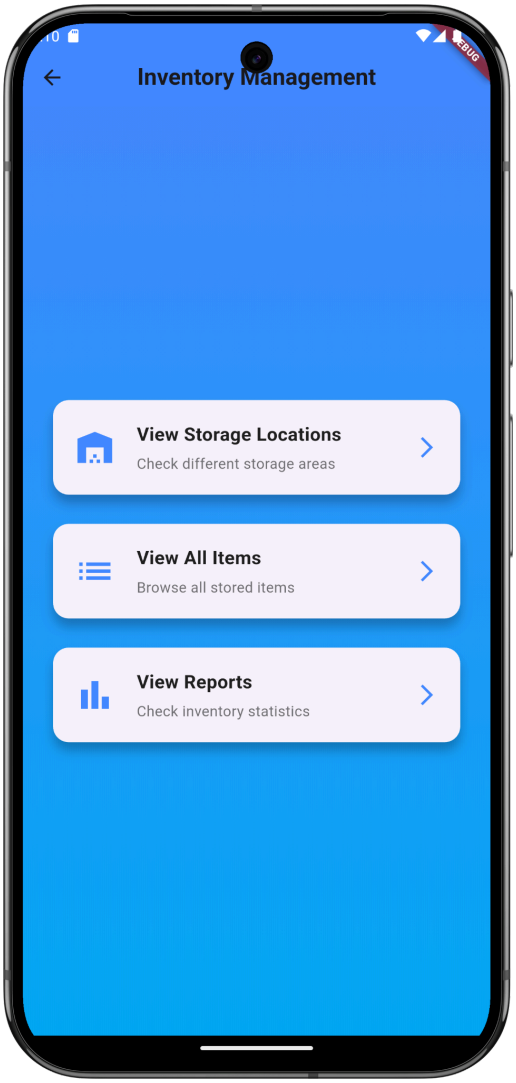


Fig. 44: Inventory Page Functionality

As you can see in Fig. 40, the inventory page allows you to view storage locations, All items, or reports, expiration log, user interaction log and item log.. In Fig. 41, you can see how the details of the location and the items are displayed in each storage location.



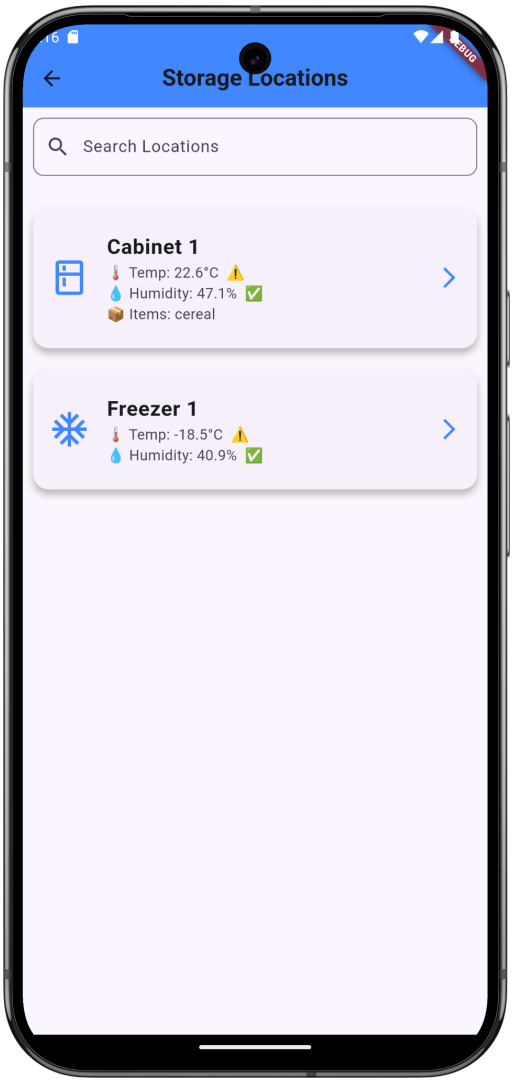


Fig. 45: Inventory Page **Fig.46: Storage Locations Page**

The location details page shown in Fig. 47 displays the information of the location including temperature, humidity, etc. It also displays warnings for any incorrect temperature and humidity conditions in that location based on its category. Clicking on an item opens an item detail page, which displays the weight of the item, when the item was added, and when the item will expire as seen in Fig. 48. Further pages about item expiry and expiration log information can be seen in Fig. 50

.

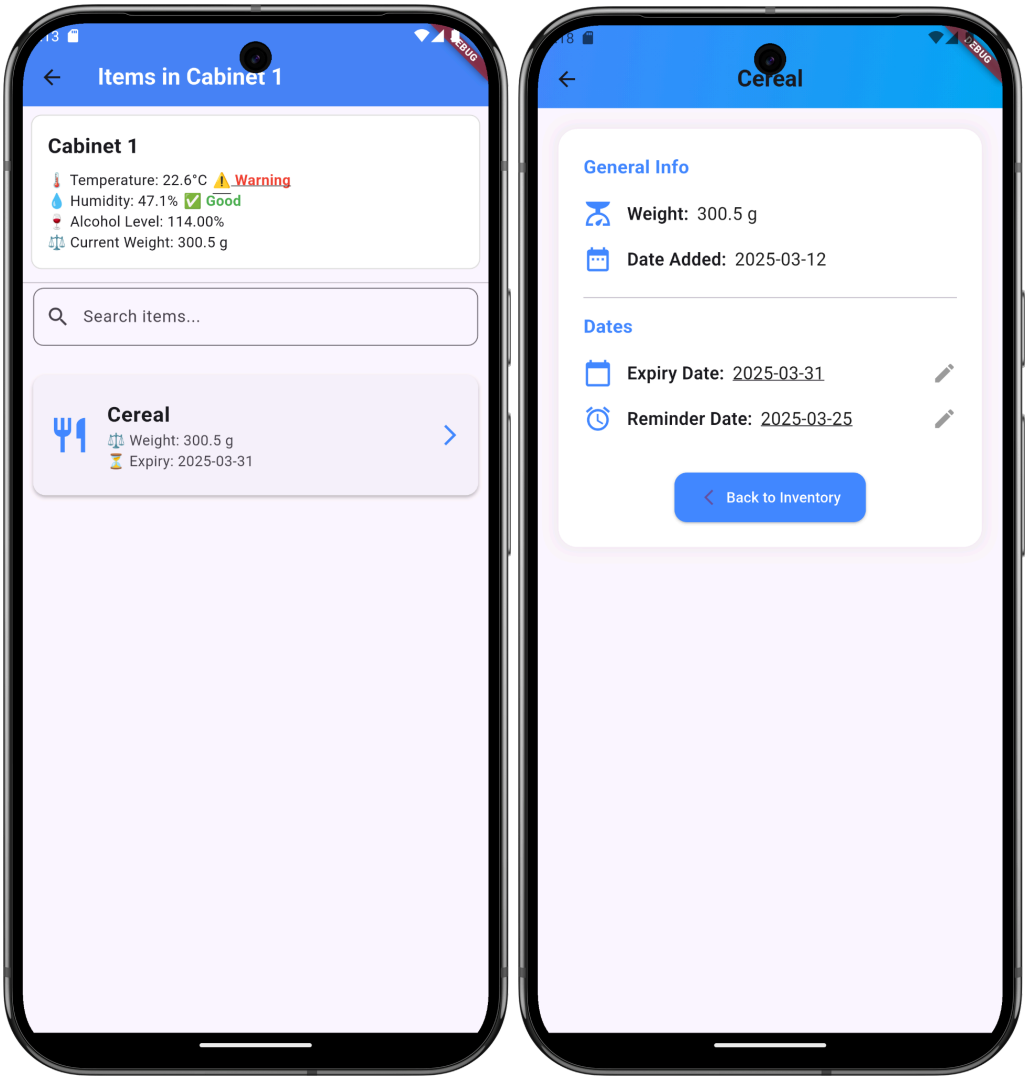


Fig. 47: Location Details Page **Fig.48: Item Details Page**

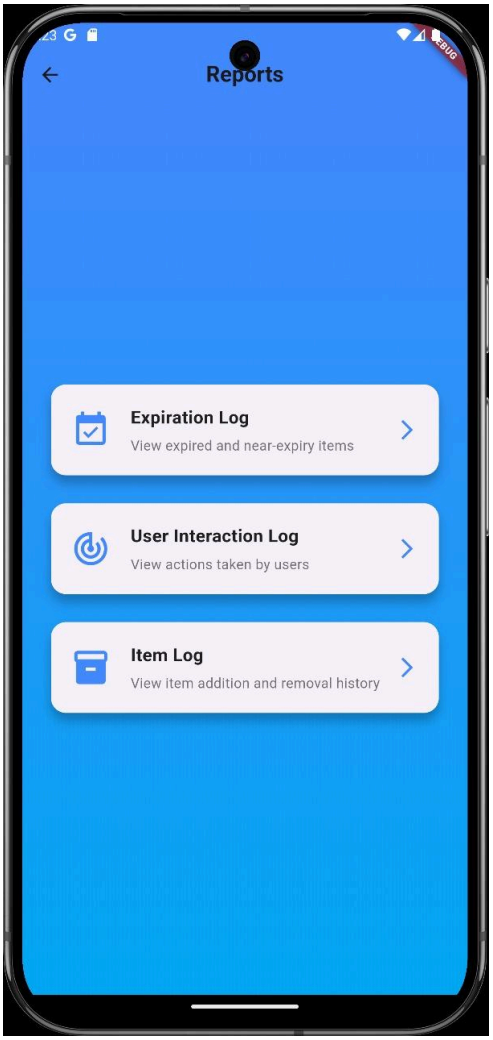


Fig. 49: Location Details Page

- The Shopping List Page allows the user to generate a shopping list automatically based on missing items in the inventory and the ingredients required for recipes. Additionally, users can manually add or remove items from the shopping list as needed. The interface also allows users to browse the generated shopping list to review or edit it before shopping. Items on the list will automatically be removed once they are placed in the kitchen inventory, ensuring real-time synchronization. A visual

representation is shown in Fig. 50 and a detailed view of the implementation can be seen in Fig. 51.

Fig. 50: Shopping List Page Functionality

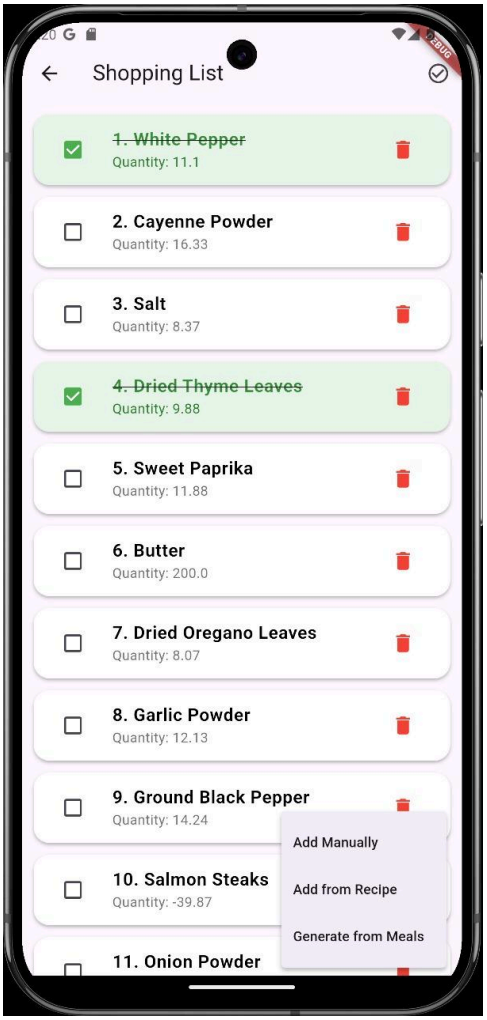


Fig. 51: Shopping List Page Implementation

Fig. 52: Implementation of the System

5.9. Recipe Recommendation System:

The Recipe Recommendation System utilizes user interaction data—specifically recipes that have been viewed, saved, and rated—to generate personalized suggestions. Each recipe is stored with a set of predefined keywords that describe its characteristics (e.g., “high protein,” “low sugar,” “vegetarian”), enabling the system to find and recommend similar recipes through keyword matching and content-based filtering.

The recommendation logic involves several components. First, user behavior is analyzed using the interactions dataset, which allows the system to identify preferences based on the frequency and type of engagement with certain recipes. This data is then cross-referenced with the recipes dataset (sourced from Food.com), filtering for relevant keywords, nutritional properties (e.g., high calorie, low carb), or preparation attributes (e.g., short cook time, simple ingredients).

Additionally, the system incorporates ingredient matching, comparing the list of ingredients in a recipe with the user's current inventory (retrieved from their inventory collection). Recipes are ranked not only by preference similarity, but also by how many required ingredients the user already possesses. In cases where ingredients are missing, the system flags them and can optionally suggest substitutes or add them to the user’s shopping list.

To guide users further, the system also displays aggregated review scores and user-written reviews from the interactions dataset, supporting informed decision-making. A diagram or flowchart outlining the full recommendation pipeline—from user input to final output—can be found in Fig 53.

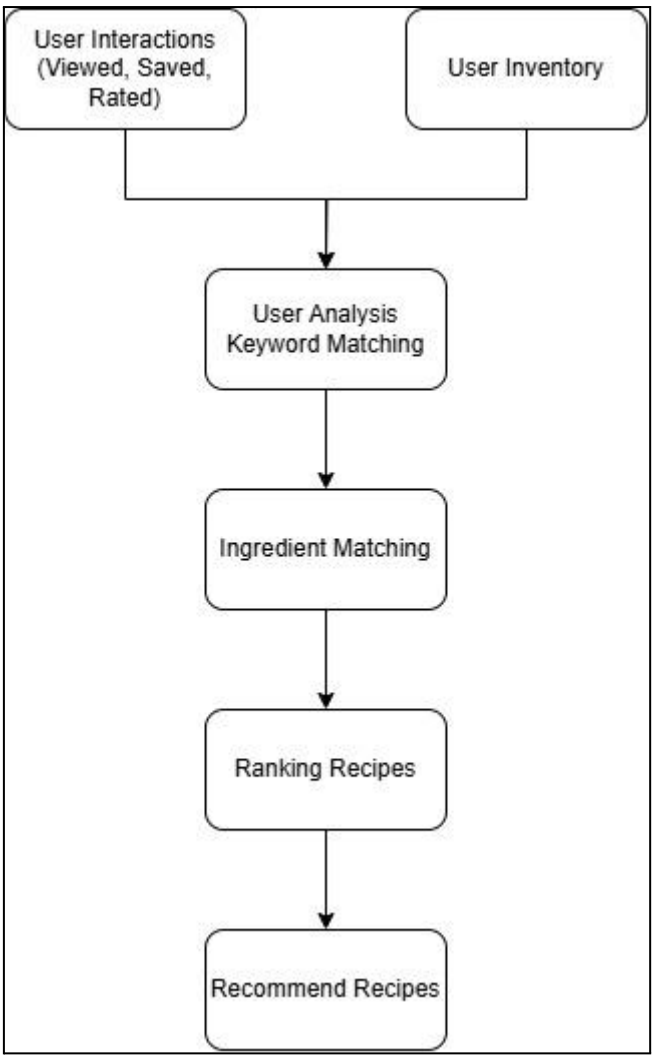


Fig. 53: Recommendation System WorkFlow

5.9.1 Dataset Description:

The Food.com Recipes and Reviews dataset consists of multiple CSV files containing structured data on recipes and user interactions. The key files used in this system are described in the following table:

File Name	Description	Key Attributes
recipes_csv	The recipes.csv dataset contains rich details for each recipe, including titles, authors, preparation and cooking times, ingredients with quantities, nutritional information, images, and step-by-step instructions. It supports diverse recipe types—from low-fat desserts to complex dishes like Biryani—highlighting metadata, dietary attributes, and user ratings for culinary exploration or analysis.	['RecipeId', 'Name', 'AuthorId', 'AuthorName', 'CookTime', 'PrepTime', 'TotalTime', 'DatePublished', 'Description', 'Images', 'RecipeCategory', 'Keywords', 'RecipeIngredientQuantities', 'RecipeIngredientParts', 'AggregatedRating', 'ReviewCount', 'Calories', 'FatContent', 'SaturatedFatContent', 'CholesterolContent', 'SodiumContent', 'CarbohydrateContent', 'FiberContent', 'SugarContent', 'ProteinContent', 'RecipeServings', 'RecipeYield', 'RecipeInstructions']
reviews_csv	The `reviews.csv` dataset contains detailed recipe metadata, including identifiers, author information, cooking and preparation times, publication dates, descriptions, images, categories, keywords, ingredient details, nutritional values, aggregated ratings, and review counts. It serves as a comprehensive source for analyzing recipe content and user engagement across various cuisines and dietary preferences.	['ReviewId', 'RecipeId', 'AuthorId', 'AuthorName', 'Rating', 'Review', 'DateSubmitted', 'DateModified']

Table. 10: Dataset Description Table

When a user requests a meal suggestion, the Flutter app communicates with the Raspberry Pi, where a Python-based script processes the inventory data stored in Firebase and matches it with a recipe data set. The recommendation model considers user preferences and available

ingredients to generate a list of suggested recipes. Additionally, the system continuously adapts to changes in user preferences and eating habits using machine learning algorithms to provide personalized and relevant recommendations. The model leverages the recipes.csv file, which includes detailed attributes such as recipe names, authors, cook/prep times, ingredient lists, instructions, categories, and nutritional information. It also utilizes the reviews.csv data, which captures user feedback through fields like ratings, review counts, comments, and recipe associations—further refining suggestions based on popularity, satisfaction, and relevance, instances of the relevant columns have been described in tables 11,12,13 and 14.

RecipeId	**Name**	**AuthorId**	**Author Name**	**CookTime**	**PrepTime**	**TotalTime**	**DatePublished**
38	Low-Fat Berry Blue Frozen Dessert	1533	Dancer	PT24H	PT45M	PT24H45M	1999-08-09T21:46:00Z
39	Biryani	1567	elly9812	PT25M	PT4H	PT4H25M	1999-08-29T13:12:00Z

Fig. 11: Recipe.csv features (1)

Keywords	RecipeIngredient Quantities	RecipeIngredientParts	AggregatedRating	ReviewCount
"Dessert", "Low Protein", "Low Cholesterol", "Healthy", "Free Of...", "Summer", "Weeknight", "Freezer", "Easy"	"4", "1/4", "1", "1"	"blueberries", "granulated sugar", "vanilla yogurt", "lemon juice"	4.5	4
"Chicken Thigh & Leg", "Chicken", "Poultry", "Meat", "Asian", "Indian", "Weeknight", "Stove Top"	"1", "4", "2", "2", "8", "1/4", "8", "1/2", "1", "1", "1/4", "1/4", "1/2", "1/4", "2", "3", NA, "2", "1", "1", "8", "2", "1/3", "1/3", "1/3", "6"	"saffron", "milk", "hot green chili peppers", "onions", "garlic", "clove", "peppercorns", "cardamom seed", "cumin seed", "poppy seed",	3	1

Table.12: Recipe.csv features (2)

Calories	FatContent	Cholesterol Content	Sodium Content	Carbohydrate Content	Fiber Content	Sugar Content	Protein Content	Recipe Serving
170.9	2.5	8	29.8	37.1	3.6	30.2	3.2	4
1110.7	58.8	372.8	368.4	84.4	9	20.4	63.4	6

Table. 13: Recipe.csv features (3)

5.10 Integration

For the integration between the hardware and software parts of the Smart Kitchen project, we use python scripts to process the sensor inputs given the hardware, and upload the required values to the database to be shown to the user. This is done by a controller script and numerous sub scripts to ensure proper functioning of the system.

5.10.1 Arduino Upload Script

This script is run on the arduino and ensures the sensors are initialized correctly, and connects with the RPI through TCP and sends the collected sensor data for that specific location to the RPI. There is a matching receiving script on the RPI inorder to connect to the arduino and receive the data. Each Arduino also takes the average of the load cells readings to ensure a more accurate reading before sending to the RPI.

5.10.2 RPI Receiving and Control Script

Also called as the main script, it establishes a socket connection with the arduino to receive incoming data. This data is then processed and used to update the database directly via the firebase admin SDK and firestore. Each Arduino will have a dedicated receiving script on the

RPI to ensure modularity, facilitate segmentation, and prevent data processing delays caused by concurrent transmissions from multiple locations. After processing the data and interpreting the nature of the incoming data, It will combine the data with the camera information and call upload scripts. These upload scripts are run on a different thread so that incoming data is still being received and processed while the data is being uploaded (in case it takes time to upload).

5.10.3 Upload Script

This script also runs on the Raspberry Pi and is invoked by the Control Script. Its primary function is to upload relevant data to the dataset such as cabinet conditions and identified objects while ensuring uninterrupted data processing. It receives the necessary values for upload from the Control Script and additionally handles the upload of miscellaneous information such as log data and other auxiliary metrics.

5.10.4 Expiration Date Script

This script is designed to run on the Raspberry Pi and will execute at regular intervals (approximately every 2 hours). It will review all items, compare their expiration dates and reminder thresholds, and generate appropriate alerts when those dates are reached or exceeded. Details can be seen in Figs.43-48

5. Validation, verification and Performance Analysis Plan

6.1 Testing

To ensure high-quality output and reliability for the Smart Kitchen Food Management System, testing will follow industry-standard methodologies, including both black-box and white-box testing. Black-box testing will focus on verifying the functionality and user experience without knowledge of the internal code, while white-box testing will examine the internal logic and flow of the application to identify issues in implementation.

Each module of the system will undergo comprehensive testing based on the use-case diagram, and specific testing methodologies will be applied to ensure robustness. Below are the modules and their respective testing strategies:

A. Inventory Tracking

1. **Functional Testing:**
   - Verify that sensors, cameras, and weighing scales accurately monitor food items in the kitchen (quantity, and location).
   - Ensure the system triggers notifications when items reach a predefined threshold, prompting restock reminders.
2. **Code Testing:**
   - Conduct unit tests on sensor integration and data flow between the sensors and the system to ensure accurate tracking.
   - Test third-party integrations (e.g., API calls to external stock systems).
3. **User Interface Testing:**
   - Validate that users can easily view and update the status of inventory.
   - Test the clarity and accuracy of alerts for restocking and threshold notifications.

B. Expiration Management

1. **Functional Testing:**
    - Ensure that the system tracks and assigns expiration dates accurately along the database and sends timely reminders for items nearing expiration.
2. **Code Testing:**
    - Test algorithms for calculating and/or fetching the expiration dates and ensuring notifications are triggered on time.
3. **User Interface Testing:**
    - Test the accessibility and clarity of expiration-related notifications in the app and the UI.

C. Meal Planning

1. **Functional Testing:**
    - Verify that the system suggests personalized meal plans based on available ingredients.
    - Ensure users can input their recipes, and the system learns their preferences over time to offer better suggestions.
2. **Code Testing:**
    - Conduct unit tests for the recipe database, user preference learning algorithms, and integration with the inventory system.
    - Test that meal plan suggestions update in real-time based on ingredient availability.
3. **User Interface Testing:**
    - Ensure that the meal planning interface is simple, allowing users to input and select recipes easily.
    - Test the accuracy and relevance of recipe suggestions based on the available ingredients.

D. Smart Shopping Assistance

1. **Functional Testing:**
    - Verify that the system creates accurate shopping lists based on meal plan requirements and inventory analysis.
    - Test integration with online shopping platforms, ensuring items can be ordered directly from the app.
2. **Code Testing:**
    - Test shopping list generation logic and integration with inventory data.
    - Ensure smooth interaction with online ordering APIs, if available.
3. **User Interface Testing:**
    - Ensure the shopping list is presented clearly to the user, and that the option to order online is easily accessible and functional.

E. Mobile Application

1. **Functional Testing:**
    - Verify that the mobile app can connect to the system and receive notifications, reminders, and meal-planning information.
    - Test push notifications for inventory updates, meal plan suggestions, and expiration reminders.
2. **Code Testing:**
    - Test the mobile app's connectivity to the central system, ensuring data synchronization between devices.
    - Test that notifications are sent and received correctly on various mobile platforms.
3. **User Interface Testing:**
    - Conduct usability tests to ensure the mobile app is user-friendly and accessible on different devices.

- Test the display and accessibility.

6.2 Validation and Verification (V&V)

Both validation and verification will be conducted throughout the development process.

V&V Activities:

A. Feedback:

- Regular meetings with advisors, to gain feedback on system progress, discuss development challenges, and refine the project's direction.

B. Internal Validation:

- **Code Inspections**: Conduct regular code reviews to catch issues early, confirm adherence to coding standards, and ensure that all functions perform as specified.
- **Design Reviews**: Review design documents and diagrams to ensure the architecture supports all necessary functionalities, such as data tracking and notification systems.
- **Test Planning**: Develop detailed test plans based on functional requirements to systematically validate each feature.

C. User Acceptance Testing (UAT):

- Engage users such as students to interact with the system in real-world scenarios.
- Collect feedback on usability, functionality, and overall user experience, incorporating this feedback into the final design refinements.

6.3 Experimentation

Experimentation plays a critical role in optimizing the performance of the smart kitchen food management system by replicating real-world conditions. These aim to test the interaction between software and hardware, ensuring seamless integration and reliability under varying scenarios. The process is divided into three key areas:

A. Feature Testing with Data

Different scenarios are used to evaluate the system's ability to monitor and respond to temperature changes and other environmental conditions. For instance:

- Sensors are exposed to a range of virtual temperature changes, from freezing to ambient, to assess their accuracy and calibration.
- The system's food tracking capabilities are tested by emulating different storage conditions, including perishable and non-perishable items, ensuring accurate categorization and updates.
- Notifications triggered by the system are verified under diverse conditions to confirm timely alerts for users about spoilage risks or replenishment needs.

B. Notification Refinement and Testing

The alert system is tested to ensure notifications are clear, timely, and reliable. Key steps include:

- Scenarios such as expiring food, temperature changes, and inventory shortages to generate accurate alerts.
- Improving notification wording and timing to make them user-friendly and actionable.
- Reducing false alerts by refining thresholds and detection logic.
- Testing user interactions to ensure notifications are effective and not overwhelming.3

C. Internal Testing

The system undergoes rigorous internal testing using real data to verify its functionality. Key steps include:

- Testing with real-world environmental data, such as actual readings from a kitchen setup, to validate sensor and software accuracy.
- Simulating cases, such as sensor malfunctions or network interruptions, to evaluate system resilience and recovery mechanisms.
- Observe the timeliness and reliability of alerts under all test conditions to confirm the system meets user expectations.
- Collecting performance metrics, such as alert response times and data accuracy rates, to fine-tune system operations.

Through these simulations, the smart kitchen food management system ensures high reliability, accuracy, and a user-friendly experience before deployment.

7. Project Management Plan

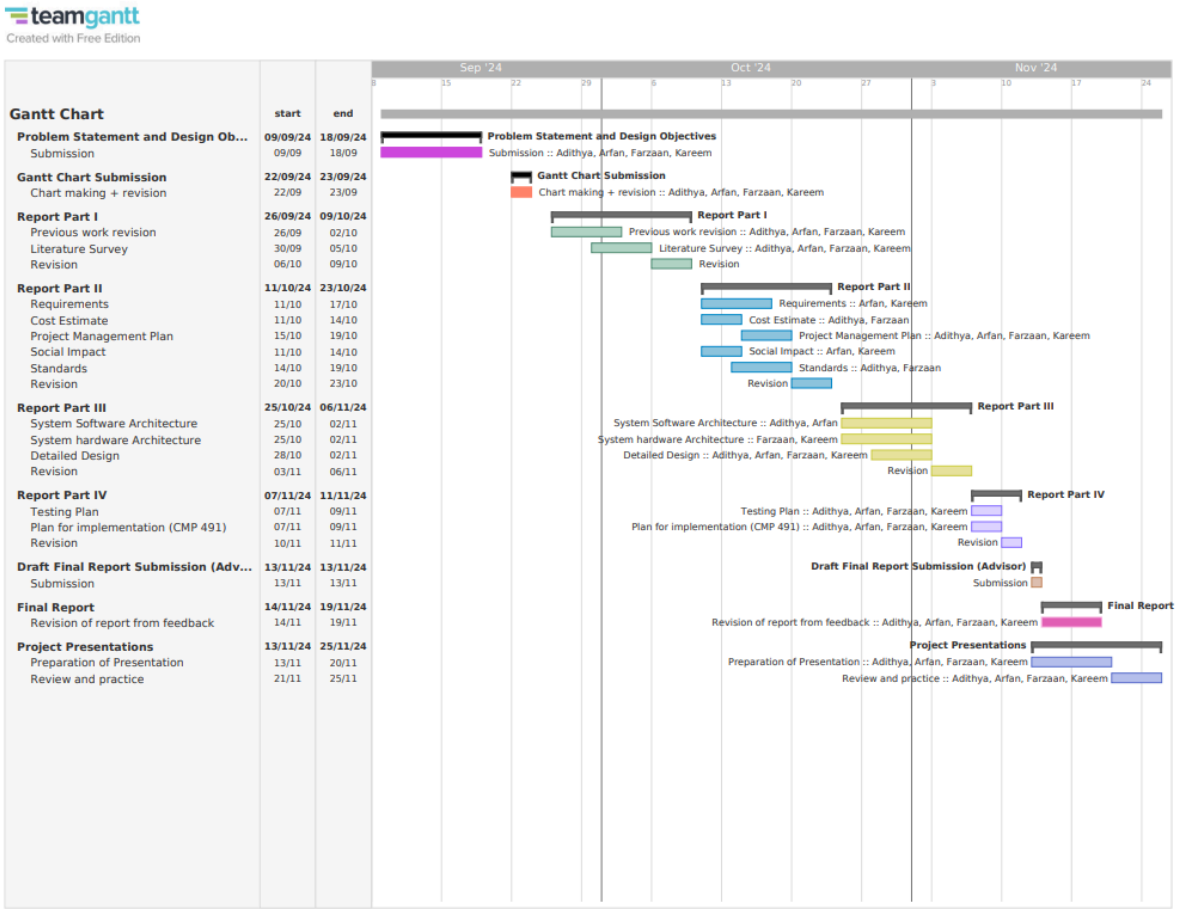


Fig. 31 Project Plan

The Gantt chart outlines the project timeline, starting with defining the objectives and submitting the problem statement. Then follows with the creation and submission of the Gantt chart itself. From September to November, the team works on four major report sections, covering requirements, specifications, system architecture, and design/testing plans. The project wraps up with a presentation. Each phase is assigned specific team members and deadlines to ensure steady progress and timely completion. The figure shows the project timeline, from the problem statement on September 18 to the final presentation on November 25. All tasks were finished on time, with the schedule updated after each task was completed.

8. Project Global, Economic, Societal Impact

The development of our Smart Kitchen Food Management System represents a significant advancement in sustainable food consumption and waste reduction practices.

On a global scale, this project addresses one of the pressing issues of our time- food waste which is a contributor to greenhouse gas emissions and resource depletion. By providing a solution that encourages better food inventory management. Additionally, as awareness of environmental sustainability grows, this project aligns with global movements toward more responsible consumption practices, contributing to efforts such as the United Nations Sustainable Development Goals (SDGs), particularly goal 12: Responsible Consumption and Production.

Economically, the system helps households reduce costs by minimizing food waste, ensuring better purchasing decisions, and optimizing meal planning. This particularly is beneficial for low-income households, where food savings can significantly impact financial well-being.

From a societal standpoint, the suggested Smart Kitchen system makes an important contribution to the development of sustainable food practices and responsible consumption. The system encourages users to make smart food decisions by utilising real-time inventory monitoring and AI-driven recipe recommendations, decreasing waste and increasing the usability of available ingredients. This not only increases individual knowledge of food conservation, but it also coincides with larger global efforts to address food hunger and environmental sustainability. Furthermore, the technology has the potential to dramatically improve meal preparation for people with hectic schedules by making intelligent recommendations based on available items and dietary preferences. In such cases, the Smart Kitchen may help to organise resources, reduce spoilage, and ensure fair meal distribution,

strengthening its social value and revolutionary potential in a variety of culinary and community settings.

9. Standards

Following international standards in a smart kitchen food management system is essential for ensuring that the technology is safe, reliable, and user-friendly. These standards address important aspects such as connectivity, software quality, energy efficiency, user interaction, and the integration of machine learning, making the system effective and dependable.

A. Communications

- **ISO/IEC 14543**: This standard is for home electronic system architecture, which includes smart appliances in home networks. It covers compatibility among smart kitchen devices.
- **ISO/IEC 29180**: Relevant for wireless communication in IoT environments, this standard addresses network connectivity, ensuring the system's communication is efficient and secure.

B. Coding

- **ISO/IEC 25010**: This standard is for software quality requirements, such as functionality, reliability, and maintainability. This ensures the code behind the smart kitchen system is robust and error-free.
- **ISO/IEC 27001**: This standard ensures data security, focusing on protecting user data collected by IoT devices. The system needs to safeguard user privacy and maintain secure food storage and management data.

C. Machine Learning

- **ISO/IEC 22989**: This standard outlines principles for developing and deploying machine learning systems. It ensures the algorithms used for tasks such as inventory tracking, expiration prediction, and shopping lists are accurate, ethical, and efficient.
- **ISO/IEC 23053**: This standard provides a framework for evaluating the quality of AI models. It ensures that machine learning models are reliable, adaptive to user behaviors, and optimized for real-time food management applications.

D. Power

- **ISO 50001**: A standard for energy management systems, ISO 50001 helps optimize the power consumption of IoT appliances in the kitchen. This ensures devices consume power efficiently, lowering operational costs and environmental impact.
- **ISO/IEC 30134**: For energy efficiency, it is particularly relevant if devices are used within a data center environment for cloud-based applications, as it measures and helps manage energy use.

E. Quality

- **ISO 9001**: This quality management system standard ensures consistent product quality and continuous improvement in device performance.

F. Human Involvement

- **ISO 9241-210**: This human-centered design process standard ensures that the smart kitchen system is user-friendly and addresses user needs, enhancing food management efficiency.

- **ISO/IEC 82304-1**: This standard guides the health and safety of health software, which can apply to any IoT system monitoring food safety, such as ensuring freshness or quality of stored food items.

10. Conclusion

Our project aims to present an innovative solution to address food waste and streamline kitchen management through the integration of IoT, Computer Vision, and machine learning technologies.

Our team recognized the increasing need for efficient household systems that promote sustainability and reduction of unnecessary waste and this project reflects our commitment to developing a solution that is accessible, convenient, and impactful. Through our system, users can monitor food inventory in real-time, receive alerts about expiring items, and access personalized meal plans based on available ingredients, helping them make informed decisions and reduce waste.

10.1 Summary

The Smart Kitchen Food Management System consists of several key components: a user-friendly mobile application that connects to a central processing system, a cloud-based database for data storage and advanced analytics, and a range of cameras and sensors to monitor kitchen items and environmental conditions. These components work in tandem to create a seamless experience for users, allowing them to interact with the system.

Throughout the project, we have identified potential challenges including privacy concerns associated with in-kitchen monitoring devices, compatibility with existing smart home devices, and the physical setup requirements of the system. However, we are certain with further refinement and user feedback, these limitations can be addressed to improve user acceptance and make our smart kitchen a feasible addition to modern homes.

10.2 Future Work

While the current system provides a comprehensive foundation for smart inventory management and reduction of food wastes, several areas have been identified for future development to enhance functionality, scalability, and user experience:

10.2.1 Enhanced Object Recognition

Due to the limited time we had, we could only register 12 items to the object detection system. While we had a web scraper and auto tagger system, it was a time consuming process and took time to train the model further.

10.2.2 Hardware Optimization

The current load cell setup, while functionally stable, occupies significant space due to the thickness of the plates used for support due to their material (wood). Future versions could explore thinner, more compact designs without compromising on structural integrity and accuracy. A sleeker form would improve usability in confined spaces and enhance the overall aesthetic appeal of the system.

10.2.3 Expansion and Accuracy of Recipe and Ingredient Data

The current implementation includes approximately 6,000 recipes and their associated ingredients, along with a limited set of additional ingredients. In the future, the system could be expanded to include the full dataset of available recipes. In addition, steps can be taken to improve the completeness and accuracy of the nutritional information stored for each ingredient and recipe, ensuring greater reliability for users tracking dietary intake.

10.2.4 Sensor Accuracy and Efficiency

Sensor performance, particularly with load cells, can be improved in terms of both accuracy and consistency. The current margin of error of approximately ±2.0 grams could be reduced by employing higher-precision load cells or incorporating additional sensors for redundancy and cross-verification. The load cells also have a maximum weight of 50kgs, which can be improved for larger cabinets or locations. Improvements in sensor calibration and error correction techniques could also contribute to more reliable data.

10.2.5 Intelligent Script Enhancements

The backend scripts that handle data interpretation and uploading currently function as intended but have room for improvement. Future development efforts could focus on enhancing these scripts to more accurately interpret incoming data, accurately detect edge cases, and ensure more robust and error-free database updates. Modularizing these scripts further may also improve maintainability and scalability.

References

[1] “The paradox of hunger and food loss and waste,” Knowledge Repository, https://openknowledge.fao.org/server/api/core/bitstreams/d6b025b8-2ab7-47f2-b336-87a1de1e67e8/content/agrifood-solutions-to-climate-change-2023/paradox-of-hunger-and-food-loss-waste.html (accessed Sep. 19, 2024).

[2] "Smart kitchen technology solutions," *Smart Kitchens*. [Online]. Available: https://smartkitchens.ae. [Accessed: Oct. 8, 2024].

[3] "Family Hub™ Refrigerator Overview," *Samsung*. [Online]. Available: https://www.samsung.com/us/explore/family-hub-refrigerator/overview/. [Accessed: Oct. 8, 2024].

[4] Siemens, “Smart Cooling,” Siemens Home, [Online]. Available: https://www.siemens-home.bsh-group.com/ae/appliances/cooling/smart-cooling [Accessed: Oct. 8, 2024].

[5] M. Chaiyaporn, A. Verma, and G. Saxena, "Smart shelf for smart kitchen: An Internet of things initiative," *International Journal of Advanced Engineering Research and Science*, vol. 2, no. 12, pp. 33-36, Dec. 2015.

[6] A.-D. Floarea and V. Sgârciu, "Smart refrigerator: A next generation refrigerator connected to the IoT," in 2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Ploiesti, Romania, 2016, pp. 1-6, doi: 10.1109/ECAI.2016.7861170.

[7] S. Adhikari, “Smart Kitchen Management Using IoT Technology,” *International Journal of Applied Engineering Research*, vol. 14, no. 3, pp. 351-358, 2019. Available: https://www.academia.edu/40443889/Smart_Kitchen_Management_Using_IOT_Technology

[8] D. B, D. A. P, M. Bharathi S, S. R, B. V, and R. M, "IoT Based Smart Refrigerator Using Machine Learning," in *2024 2nd International Conference on Artificial Intelligence and

Machine Learning Applications Theme: Healthcare and Internet of Things (AIMLA)*, Namakkal, India, 2024, pp. 1-6, doi: 10.1109/AIMLA59606.2024.10531466.

[9] R. Luo et al., "IoT-based Smart Kitchen Pantry System," *IEEE Access*, vol. 11, pp. 53065-53075, Jul. 2023, doi: 10.1109/ACCESS.2023.3067050. Available: https://ieeexplore.ieee.org/document/10303307

[10] C. Y. Kim, D. Chou, M. Reininger, and Y. Reiss, "Pocket Pantry: A Smart Kitchen Storage System," in *Proceedings of the ACM Conference on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2020. doi: http://dx.doi.org/10.1145/0000000.0000000

[11] M. A. Khan, M. H. B. Shahid, H. Mansoor, U. Shafique, M. B. Khan, and A. u. R. Khan, "IoT based Grocery Management System: Smart Refrigerator and Smart Cabinet," in *2019 International Conference on Systems of Collaboration Big Data, Internet of Things & Security (SysCoBIoTS)*, Casablanca, Morocco, 2019, pp. 1-5. doi: 10.1109/SysCoBIoTS48768.2019.9028031

[12] T. Ahmad et al., "IoT-Based Food Inventory Tracking System for Domestic and Commercial Kitchens," *International Journal of Smart Home*, vol. 16, no. 2, pp. 55-60, Sep. 2021. Available: https://www.researchgate.net/publication/354820204

[13] N. Umapathi and S. Sabbani, "An Internet of Things (IoT)-based Approach for Real-Time Kitchen Monitoring Using NodeMCU 1.0," in *Futuristic Communication and Network Technologies*, A. Sivasubramanian, P. N. Shastry, and P. C. Hong, Eds. Singapore: Springer, 2022, vol. 792, pp. 1-10. doi: 10.1007/978-981-16-4625-6_

[14] T. Ahmad et al., "Towards Autonomous IoT Analytics: Architectures and Collaborative Intelligence Paradigms," *IEEE Access*, vol. 7, pp. 64029-64048, 2019, doi: 10.1109/ACCESS.2019.2914146. Available: https://ieeexplore.ieee.org/document/8776761

[15] A. Baker, “Smart Kitchen Management Using IoT: A Survey,” *Journal of Food Engineering*, vol. 8, no. 1, pp. 131-150, 2021. Available: https://www.academia.edu/download/64548643/IRJET-V7I31041.pdf

[16] A. Anand, "IoT based smart kitchen using sensors," *International Journal of Research and Analytical Reviews (IJRAR)*, vol. 6, no. 2, pp. 471-476, Jun. 2019. [Online]. Available: https://www.ijrar.com/upload_issue/ijrar_issue_20543710.pdf

[17] F. Barr, "Design and Implementation of a Smart Pantry System," *IEEE Transactions on Consumer Electronics*, vol. 68, no. 1, pp. 125-130, Jan. 2022. Available: https://ieeexplore.ieee.org/document/9028031

[18] A. Balaji, B. Sathyasri, S. Vanaja, M. N. Manasa, M. Malavega, and S. Maheswari, "Smart Kitchen Wardrobe System Based on IoT," in *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, Trichy, India, 2020, pp. 865-871. doi: 10.1109/ICOSEC49089.2020.9215459

[19] S. U. Shariff, M. G. Gurubasavanna, and C. R. Byrareddy, "IoT-Based Smart Food Storage Monitoring and Safety System," in *International Conference on Computer Networks and Communication Technologies*, S. Smys, R. Bestak, J. Z. Chen, and I. Kotuliak, Eds. Singapore: Springer, 2019, vol. 15, pp. 1-10. doi: 10.1007/978-981-10-8681-6_57

[20] A. A. Mohsin, S. U. Shariff, and M. K. H. Noor, "Smart kitchen management system using IoT," *Semantics Scholar*, 2020. [Online]. Available: https://pdfs.semanticscholar.org/6c5d/174185ca71310e248d0451097a6ee01c90e9.pdf

[21] R. Girish, "Towards an IoT-based Smart Kitchen for Domestic Use," *International Journal of Recent Technology and Engineering*, vol. 9, no. 1, pp. 60-70, Jan. 2020. Available: https://www.oaijse.com/VolumeArticles/FullTextPDF/833_31.IOT_BASED_SMART_REFRIGERATOR.pdf.

[22] R. Thomas, "Smart Refrigerator Implementation Using IoT for Commercial Kitchens," *IEEE Transactions on Consumer Electronics*, vol. 12, no. 3, pp. 275-290, Jun. 2022.

[23] A. R. Ashwathan, A. V. P. Y., G. S., and K. K., "Object Detection in IoT-Based Smart Refrigerators Using CNN," *School of Computer Science and Engineering, VIT University*, Chennai, India, 2023. https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119769026.ch11

[24] S. Palanisamy and S. N, "Sensor Based Industrial Kitchen Foodstuffs Monitoring System," *International Journal of Computer Communication and Informatics*, vol. 3, no. 1, pp. 26–43, May 2021. doi:10.34256/ijcci2113

[25] S. Gupta *et al.*, "Smart refrigerator based on 'internet of things,'" *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, Mar. 2021. doi:10.1109/icacite51222.2021.9404612

[26] N. Nadar, Y. Ali, Y. Ali, S. Khade, and T. Goskula, "Intelligent Refrigerator," *International Journal For Advance Research In Science & Technology*, vol. 11, no. 6, Jun. 2021.

[27] D. B *et al.*, "IOT based smart refrigerator using machine learning," *2024 2nd International Conference on Artificial Intelligence and Machine Learning Applications Theme: Healthcare and Internet of Things (AIMLA)*, vol. 4, pp. 1–6, Mar. 2024. doi:10.1109/aimla59606.2024.10531466

[28] D. K. Shah *et al.*, "Smart kitchen: Real time monitoring of kitchen through IOT," *2022 3rd International Conference on Intelligent Engineering and Management (ICIEM)*, pp. 718–722, Apr. 2022. doi:10.1109/iciem54221.2022.9853161

[29] S. Goel, S. Harish, and V. Ramachandran, "IOT based smart refrigerator and Shopping System prototype," *2023 Innovations in Power and Advanced Computing Technologies (i-PACT)*, vol. 6, pp. 1–5, Dec. 2023. doi:10.1109/i-pact58649.2023.10434662

[30] H. Nasir, W. B. Aziz, F. Ali, K. Kadir, and S. Khan, "The implementation of IOT based Smart Refrigerator System," *2018 2nd International Conference on Smart Sensors and Application (ICSSA)*, Jul. 2018. doi:10.1109/icssa.2018.8535867

[31] L. Garg, K. Ramesh, G. Garg, A. Portelli, and A. Jamal, “Kitchen genie: An intelligent internet of things system for household inventory management,” *Lecture Notes in Electrical Engineering*, pp. 3–20, Sep. 2019. doi:10.1007/978-3-030-30577-2_1

[32] M. A. Ahmed and R. Rajesh, "Implementation of Smart Refrigerator based on Internet of Things," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 9, no. 2, pp. 1023-1026, Dec. 2019 https://www.researchgate.net/profile/Abrar-Mohammed-7/publication/349246132_Implementation_of_Smart_Refrigerator_based_on_Internet_of_Things/links/602685424585158939 9b6db9/Implementation-of-Smart-Refrigerator-based-on-Internet-of-Things.pdf

[33] P. C. Sane, H. K. Barapatre, and A. Sanghavi, “SMART REFRIGERATOR AND VEGETABLE IDENTIFICATION SYSTEM USING IMAGE PROCESSING AND IOT,” *Open Access International Journal of Science & Engineering*, vol. 6, no. 4, Apr. 2021.

[34] Devanathan B, Naga Kamalesh Mathala, Suyampulingam A, and Ilango K, “Secure and Energy-Efficient Smart Home Automation: A User-Based Fingerprint Security System,” Mar. 2024, doi: https://doi.org/10.1109/incos59338.2024.10527495.

[35] J. Moy Chatterjee, R. Kumar, M. Khari, D. Thi Hung, and D.-N. Le, “Internet of Things based system for Smart Kitchen,” *International Journal of Engineering and Manufacturing*, vol. 8, no. 4, pp. 29–39, Jul. 2018, doi: https://doi.org/10.5815/ijem.2018.04.04.

[36] S. A. Sivakumar, S Vijay Murugan, V. Nagaraj, Balachandra Pattanaik, Manjula Pattnaik, and Muruganantham Ponnusamy, “Internet of Things based Smart Cooking System,” *2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Nov. 2021, doi: https://doi.org/10.1109/i-smac52330.2021.9640946.

[37] Manasa Sandeep and C. Nandini, “Implementation of Iot Based Smart Cooking Environment,” Mar. 2019, doi: https://doi.org/10.1109/icatiece45860.2019.9063796.

[38] S. Rezwan, W. Ahmed, M. A. Mahia, and M. R. Islam, “IoT Based Smart Inventory Management System for Kitchen Using Weight Sensors, LDR, LED, Arduino Mega and

NodeMCU (ESP8266) Wi-Fi Module with Website and App," *IEEE Xplore*, Oct. 01, 2018. https://ieeexplore.ieee.org/abstract/document/8776761/?casa_token=sA0BFMW3beEAAAAA:WwaFkll8vYmpCYcVJ8WdHljXpYy2FqIaIlCQoIO51lZMmRut7NUeLktExttYLcZ6lWodbO5vYQ (accessed Sep. 20, 2021).

[39] [1] C. Y. Kim, M. Reininger, D. Chou, and Y. Reiss, Pocket pantry: A smart kitchen storage system, http://www.cs.umd.edu/~dchou/papers/838j_paper.pdf (accessed Oct. 13, 2024).

[40] S. Chaudhary *et al.*, "ChefAI.IN: Generating Indian recipes with AI algorithm," *2022 International Conference on Trends in Quantum Computing and Emerging Business Technologies (TQCEBT)*, Oct. 2022. doi:10.1109/tqcebt54229.2022.10041463

[41] H. Jabeen, J. Weinz, and J. Lehmann, "Autochef: Automated generation of cooking recipes," *2020 IEEE Congress on Evolutionary Computation (CEC)*, vol. 11453, pp. 1–7, Jul. 2020. doi:10.1109/cec48606.2020.9185605

[42] "HX711 LOAD CELL AMPLIFIER+50Kg - Besomi Electronics," Besomi Electronics, Nov. 18, 2024. https://besomi.com/product/hx711-load-cell-amplifier50kg/

[43] Besomi, "Arduino Giga R1 Wi-Fi," [Online]. Available: https://besomi.com/product/arduino-giga-r1-wi-fi/.

[44] Besomi, "DFR0066 SHT10 Digital Temp & Humidity Sensor," [Online]. Available: https://besomi.com/product/dfr0066-sht10-digital-temp-humidity-sensor/.

[45] Besomi, "MQ-3 Gas Sensor Module," [Online]. Available: https://besomi.com/product/mq-3-gas-sensor-module-alcohol-ethanol-and-smoke-sensor/.

[46] Besomi, "Gravity: Huskylens AI Vision Sensor," [Online]. Available: https://besomi.com/product/video-ic-development-tools-gravity-huskylens-an-easy-to-use-ai-vision-sensor-pre-order/.

[47] Amazon.ae, "eufy Security 2K Indoor Cam Pan & Tilt, Home Security Indoor Camera," [Online]. Available: https://www.amazon.ae/eufy-Security-Assistants-Tracking-Required/dp/B086KZ8X6S/.

[48] Besomi, "Raspberry Pi 4 8GB 7 Inch Screen Kit," [Online]. Available: https://besomi.com/product/raspberry-pi-4-8gb-7-inch-screen-kit/.

[49] E. Cervera, "GPU-Accelerated Vision for Robots: Improving System Throughput Using OpenCV and CUDA," in IEEE Robotics & Automation Magazine, vol. 27, no. 2, pp. 151-158, June 2020, doi: 10.1109/MRA.2020.2977601.