Birzeit University
Faculty of Engineering & Technology
Department of Electrical & Computer Engineering

AI-Powered Meal Recognition & Recipe Assistant

ENCS5300 Section 2

Prepared By:
Rana Musa (1210007)
Layan Burait (1211439)
Haneen Odeh (1210716)

Supervised By:
Dr. Yazan Abu Farha

Graduation Project submitted to the Department of Electrical and Computer Engineering in partial fulfillment of the requirements for the degree of B.Sc. in Computer Engineering

Birzeit
February, 2026

Abstract

This project presents Meal Lens, an AI-powered meal recognition and recipe assistant application that bridges the gap between food discovery and meal preparation. With the use of state-of-the-art computer vision techniques and deep learning models, the system enables users to identify meals through image recognition. The application integrates datasets like Arabic Food 101 and others to support different cuisines. This project brings meal systems to the next level with its innovative incorporation of machine learning into practical culinary applications, offering users a convenient meal recognition solution while allowing extensibility for development down the line. The open and modular nature of the project makes it a foundation for future development in food image recognition technologies.

المستخلص

يُقدم هذا المشروع تطبيقاً ذكياً متكاملاً للتعرف على الأطعمة وتحليلها غذائياً باستخدام أحدث تقنيات الذكاء الاصطناعي والرؤية الحاسوبية. يتميز التطبيق بقدرته على تحليل مكونات الوجبات من خلال الصور الملتقطة بالهاتف الذكي أو الصور المرفوعة من مواقع الكترونية، وتقديم وصفات دقيقة للتحضير مع معلومات غذائية شاملة تشمل القيم الغذائية وتحذيرات الحساسية والسعرات الحرارية وغيرها. يعتمد النظام على قواعد بيانات واسعة تغطي المطابخ العربية والعالمية. يتميز التطبيق بواجهة سهلة الاستخدام وسرعة في الأداء، مع الحفاظ على دقة النتائج وأمان البيانات. يمثل هذا المشروع نقلة نوعية في مجال التطبيقات الذكية للطهي والتغذية، حيث يجمع بين الدقة العلمية والسهولة في الاستخدام، مع إمكانية التطوير المستمر ليشمل ميزات إضافية تخدم مختلف الفئات المستهدفة.

Table of Contents

English Abstract **I**

Arabic Abstract **II**

Table of Contents **III**

List of Tables **VII**

List of Figures **IX**

1 Introduction & Motivation **1**

1.1 Introduction 1

1.2 Motivation 1

1.3 Problem Statement 2

1.4 Methodology 2

1.5 Objectives 3

1.6 Report Structure 3

2 Related Work **5**

2.1 Introduction 5

2.2 AI Photo Recipe Identifier 5

2.3 ChefMate AI 7

2.4 ingreGenius 7

2.5 SideChef 8

2.6 FOODAI 9

2.7 Summary and Comparative Analysis 11

3 Background **12**

3.1 Dataset 13

3.1.1 Arabic Food 101 13

3.1.2 Multi-Class Food Image Dataset . . . 13
3.1.3 UEC FOOD 256 Dataset . . . 14
3.1.4 Cakes Dataset . . . 14
3.1.5 Food-101 Dataset . . . 14
3.1.6 Sri Lankan Food Image Dataset . . . 14
3.1.7 Food Image Classification Dataset . . . 15
3.1.8 Fast food Dataset . . . 15
3.1.9 Custom Collected Dataset from Shutterstock and Google images . . . 15
3.2 Firebase: Overview and Integration Strategy . . . 15
3.2.1 Introduction to Firebase . . . 15
3.2.2 Rationale for Using Firebase in MealLens . . . 15
3.2.3 Key Firebase Services and Benefits . . . 16
3.2.4 Cost Structure: The Free Tier . . . 16
3.2.5 Using Firebase for Authentication and User Management . . . 16
3.2.6 Firebase Authentication Dashboard . . . 17
3.3 APIs . . . 18
3.3.1 Spoonacular API . . . 18
3.3.2 Edemam . . . 19
3.3.3 mealDB . . . 19
3.4 App design Tools . . . 19
3.4.1 Figma . . . 19
3.4.2 Flutter . . . 19
3.5 Image Classification Model . . . 20
3.5.1 Azure Custom Vision . . . 20
3.5.2 Recognition Process . . . 22

4 System Structure **23**
4.1 Application Features . . . 24
4.2 Mobile Features . . . 26
4.3 Service Features . . . 27
4.4 Use-case Diagram . . . 27
4.5 Image Recognition Process Using Custom Vision . . . 28
4.5.1 Custom Vision Dataset . . . 28
4.5.2 Sending Images to the Model . . . 29
4.6 System Architecture Overview . . . 29

4.6.1 Application architectural Layers . . . 30
4.6.2 Data Flow . . . 30
4.6.3 System Sequence Diagram . . . 31
4.6.4 Cost Management . . . 32
4.7 System Database Management . . . 33
4.7.1 Database Structure Overview . . . 33
4.7.2 Collections Description . . . 33
4.7.3 Relationships and Data Flow . . . 35
4.7.4 Database Schema Diagram . . . 36
4.8 UI/UX Design . . . 36

5 Experimental Results and Analysis 47
5.1 Introduction . . . 47
5.2 Evaluation Metrics . . . 47
5.2.1 Accuracy . . . 47
5.2.2 Precision . . . 47
5.2.3 Recall . . . 48
5.2.4 Average Precision (AP) . . . 48
5.2.5 Mean Average Precision (mAP) . . . 48
5.3 Experimental Setup . . . 48
5.3.1 Experimental Dataset . . . 48
5.4 Full System Evaluation . . . 50
5.4.1 Experimental Setup . . . 50
5.4.2 Results and Discussion . . . 50
5.5 User Testing . . . 50
5.5.1 Test User Credentials and Access Validation . . . 51
5.5.2 Image Cropping and Editing . . . 53
5.5.3 Continue as Guest Mode . . . 54
5.5.4 History Management and Deletion . . . 56
5.5.5 Complete Application Demo . . . 57

6 Web Platform Development for MealLense 58
6.1 Introduction and Overview . . . 58
6.2 Platform Architecture . . . 58
6.2.1 Firebase Database Structure . . . 58

6.3 Website Functionality . . . 58
6.3.1 Homepage with Contact Form . . . 58
6.3.2 FAQ System with Search . . . 59
6.3.3 Admin Dashboard . . . 59
6.4 Mobile App Integration . . . 59
6.4.1 Cross-Platform Experience . . . 59
6.4.2 Support Workflow . . . 60
6.5 Deployment . . . 60
6.5.1 Render Hosting . . . 60
6.5.2 Security Rules . . . 60
6.6 Workflows . . . 60
6.6.1 Message Processing . . . 60
6.6.2 FAQ Maintenance . . . 60

7 Conclusion and Future Work **61**
7.1 Conclusion . . . 61
7.2 Future Work . . . 61

8 Appendix **64**
.1 Complete List of Meals . . . 69
.1 Database Schemas . . . 75

List of Tables

2.1 Comparison of key features across selected meal recognition and nutrition applications. . . . 11

4.1 API Free Tier Usage and Limits . . . 33

5.1 Model Accuracy vs. Number of Training Images per Dish . . . 49

1 Selected categories from Arabic food 101 dataset . . . 64

2 Selected categories from the Multi-Class Food Image Dataset with Image Counts . . . 64

3 Selected categories from UEC FOOD 256 Dataset . . . 65

4 Selected categories from the Cakes Dataset . . . 66

5 Selected categories from the Food-101 Dataset . . . 67

6 Selected Meal Categories Collected from Shutterstock and Google Images . 68

7 Selected categories from the Sri Lankan Food Image Dataset [1] . . . 69

8 Selected categories from the Food Image Classification dataset [2] . . . 69

9 Complete list of meal categories used in the dataset . . . 69

10 Fast Food Dataset Classes and Image Distribution . . . 74

11 Users Collection Schema . . . 75

12 Recipes Collection Schema . . . 75

13 Feedback Collection Schema . . . 76

14 Application Statistics Schema . . . 76

15 favorites Collection Field Specifications . . . 77

16 history Collection Field Specifications . . . 78

17 Deleted Recipes Collection Schema . . . 79

18 Access Control Matrix by User Role . . . 79

19 Firestore Collections for Web Platform Operations . . . 79

20 admin_users Collection Field Specifications . . . 80

21 user_questions Collection Field Specifications . . . 80

22 website_messages Collection Field Specifications 81
23 website_faq Collection Field Specifications 82

List of Figures

2.1 An example of AI Photo Recipe Identifier meal recognition. 6
2.2 Pizza meal recognition by IngreGenius 8

3.1 The Authentication dashboard for the MealLens application on Firebase . 18

4.1 use case diagram 28
4.2 Complete System Architecture of MealLens Application 32
4.3 Entity-Relationship Diagram of MealLens Database Architecture 36
4.4 Home screen 37
4.5 Login screen 37
4.6 Sign-up screen 37
4.7 App main dashboard screen. 38
4.8 Choose photo or analyze screen. 38
4.9 Image crop interface. 38
4.10 MealLens app classification result screens. 39
4.11 Detailed ingredients and preparation steps. 40
4.12 Profile screen 42
4.13 Second screen 42
4.14 AI assistant 42
4.15 User history screen. 43
4.16 Setting screen 43
4.17 Admin screen. 44
4.18 Feedback screen 44
4.19 Manage data 45
4.20 Search 45
4.21 Statistics 45
4.22 Add recipe 46
4.23 Edit recipe 46

4.24 Delete recipe 46

5.1 A plot representing the Model Accuracy vs. Number of Training Images per Dish 49
5.2 Image representing the evaluation metrics of the model 50
5.3 Sign up screen 52
5.4 Home screen 52
5.5 enter wrong password 53
5.6 reset password 53
5.7 password reset email sent 53
5.8 uploade Photo from gallery 54
5.9 crope image 54
5.10 rotation image by 90 degree 54
5.11 conform edit image 54
5.12 Take Photo Screen in Guest Mode 55
5.13 Image Preview Before Analysis 55
5.14 Initial Recognition Result 55
5.15 Recipe Overview Display 55
5.16 Ingredients and Quantities Section 56
5.17 Step-by-Step Cooking Instructions 56
5.18 Recipe Source and Metadata 56
5.19 User's history with meals 57
5.20 Allowed individual selection of meal items 57
5.21 Successfully deleted the selected "Croque Madame" item 57

6.1 Homepage Interface 59
6.2 FAQ Search System 59
6.3 Administrative Dashboard 59

Chapter 1

Introduction & Motivation

Contents

1.1 Introduction **1**
1.2 Motivation **1**
1.3 Problem Statement **2**
1.4 Methodology **2**
1.5 Objectives **3**
1.6 Report Structure **3**

1.1 Introduction

People still talk about how smart tech changes things in our busy digital lives. It even affects simple stuff like food. Meal Lens stands out as this cool project using AI for spotting meals and helping with recipes. It pulls in computer vision and deep learning to figure out dishes from pictures. Then it offers custom recipe ideas.

The app called Meal Lens let folks snap a photo of what they ate. They can also pull in images from online sources. Right away the system picks up on the dish. It spots the main ingredients too. Say someone wants to copy a meal from a restaurant they liked. Or maybe tweak their go-to dishes a bit. Even if they just hunt for fresh cooking concepts. Meal Lens delivers smart tips that match their needs.

This setup mixes AI image spotting with a big collection of recipes. Users can dive into food options without much hassle. It targets home cooks mostly. Food fans come in too. Anyone needing ideas in the kitchen will find it useful. Meal Lens connects finding new foods to actually making them. It turns cooking into something more inventive and fun.

The easy layout and smart picks in Meal Lens shift how we deal with food. Regular meals become these enjoyable food adventures.

1.2 Motivation

The prime impetus to this system comes from the incorporation of three factors: **technological development**, **changing user needs**, and the **vision for the democratization of decision-making**.

- **Technological Foundation**: Recent breakthroughs in computer vision and natural language processing finally allow for an advanced food image analysis that accurately captures diverse dishes, adapting to different food preferences. These capabilities form the bedrock of an intelligent system that seamlessly links visual input with practical personalized recommendations.
- **User-oriented demand**: People's lifestyles today demand solutions that make life easier. In an era of packed schedules, when food trends continue to abound, the need is evidently there for personalized tools that save time and help the user make culinary choices effortlessly and health-consciously.
- **Empowerment Vision**: Core to this project is its enhancement of the quality of life through intuitive technology. By giving instant meal insights and tailored recipe suggestions, the system fosters an informed, creative, and enjoyable cooking experience and demonstrates a commitment to practical innovation that will serve real-world needs.

The point at which this advanced technology meets user-centered design and improvement of lifestyle defines the basis for re-imagining human-food interaction in daily life. This system is not only a technical achievement but also a meaningful step toward more accessible and enjoyable culinary experiences.

1.3 Problem Statement

In today's fast-paced lifestyles, many individuals struggle to identify meals, understand their components, and access reliable cooking information efficiently. Searching for recipes, ingredients, and preparation details manually requires time and effort, especially when users encounter unfamiliar dishes. Existing food applications often depend on text-based searches or manual input, which can be inconvenient and inaccessible for some users. Additionally, many systems lack integrated features such as meal history tracking, saving recognized meals, and accessibility support, reducing their overall usability and long-term engagement.

While recent advances in computer vision enable food recognition from images, many available solutions provide limited information beyond basic food identification. There is a need for a unified system that not only recognizes a meal from a photo but also delivers complete and practical cooking-related information in a user-friendly manner. This project addresses this gap by proposing an intelligent meal recognition application that identifies the food name from an image and provides the corresponding recipe, ingredients list, and cooking time. The system also allows users to save meals, view recognition history, and access text-to-speech functionality to enhance accessibility. By combining automated food recognition with informative and assistive features, the proposed solution aims to simplify the cooking experience and improve user convenience.

1.4 Methodology

The system is developed using a modular methodology that integrates computer vision techniques, cloud-based services, and external application programming interfaces (APIs). The process begins with capturing a food image from the user, followed by preprocessing steps such as resizing and normalization to prepare the image for recognition. Food

identification is performed using a deep learning–based image recognition model deployed through Microsoft Azure services, which classifies the input image and returns the recognized food name.

Once the meal is identified, cooking-related information—including recipes, ingredients, and estimated cooking time—is retrieved using independent third-party recipe APIs that operate separately from Azure services. Application-level functionalities such as saving recognized meals and maintaining recognition history are implemented using Firebase for backend processing and cloud data storage. The system is evaluated using real-world user images to ensure accurate recognition, reliable cloud integration, and stable system performance.

1.5 Objectives

This project aims to develop an intelligent food recognition and recipe assistance system with the following primary objectives:

1. To design and implement an accurate computer vision system capable of:
    - Identifying food items from user-submitted images
    - Estimating portion sizes and meal components
    - Recognizing diverse cuisines, including local and regional dishes
2. To create a personalized recommendation engine that:
    - Adapts to individual dietary preferences and restrictions
    - Incorporates cultural and regional food variations
    - Provides suitable alternatives and recipe variations
3. To develop an extensible and scalable platform that:
    - Supports continuous learning from user interactions
    - Facilitates easy integration of new datasets and APIs
    - Ensures a seamless, cross-platform user experience
4. To achieve robust performance in terms of:
    - Recognition accuracy and response time
    - User interface responsiveness and accessibility
    - Support for a wide variety of cuisines and dietary needs

1.6 Report Structure

This report is organized into five main chapters, each addressing a critical aspect of the project:

- **Chapter 1: Introduction & Motivation**
  Introduces the Meal Lense application, its objectives, motivation, and problem statement, and presents the overall methodology and goals of the project.

- **Chapter 2: Related Work**
  - Reviews existing applications and technologies in the domain of AI-powered meal recognition and recipe assistance, highlighting their features, strengths, and limitations.
  - Includes a comparative analysis to contextualize the innovations introduced by Meal Lense.
- **Chapter 3: Background**

  Provides background information on the datasets, external APIs, Azure services, Firebase, and development tools used in building the application.
- **Chapter 4: System Structure**

  Describes the overall system architecture, application features, backend services, system integration, and UI/UX design of the Meal Lense application.
- **Chapter 5: Conclusion and Future Work**
  - Summarizes the project's achievements and its potential impact.
  - Proposes future enhancements, such as multi-language support and integration with wearables.

Chapter 2

Related Work

Contents

2.1 Introduction **5**
2.2 AI Photo Recipe Identifier **5**
2.3 ChefMate AI **7**
2.4 ingreGenius **7**
2.5 SideChef **8**
2.6 FOODAI **9**
2.7 Summary and Comparative Analysis **11**

2.1 Introduction

This section highlights the most similar applications to our meal recognition application, as many existing systems utilize machine learning and computer vision techniques to identify food items from images, offering functionalities such as meal and ingredients recognition, meal instructions, calorie estimation, and other features. These capabilities provide valuable insights for users aiming to maintain a healthy lifestyle. Reviewing these applications helps us identify effective strategies and limitations, guiding the development of our own application to ensure it is both practical and user-friendly.

2.2 AI Photo Recipe Identifier

The *AI Photo Recipe Identifier* application is designed to identify recipes from photos using advanced AI technology. Images of dishes or ingredients are analyzed, and corresponding recipes are provided. Custom recipes can also be generated based on detected ingredients, making it a valuable tool for culinary exploration.

The first step in using the application involves prompting the user to select an image, either by capturing a new photo or uploading an existing one. Accordingly, the initial features offered are "Take Photo" and "Upload Photo." If the user selects the "Take Photo" option, the application provides an additional "Retake" feature, allowing the user to recapture the image before proceeding. This ensures that a clear and suitable photo is

obtained for the analysis process. Once satisfied with the image, the user can proceed by selecting "Use Photo" to continue with recipe identification.

Regardless of the method used to provide the image, the application then analyzes it and presents relevant information, including:

- The name of the meal or a list of identified ingredients, along with a brief description.
- A detailed list of ingredients associated with the meal.
- Step-by-step cooking instructions. If the image contains only ingredients rather than a full dish, the application recommends custom recipes based on the detected ingredients.

Figure 2.1 shows an example of AI Photo Recipe meal recognition, as the meal here is "Molokhia" and it gives its name, ingredients, and its recipe

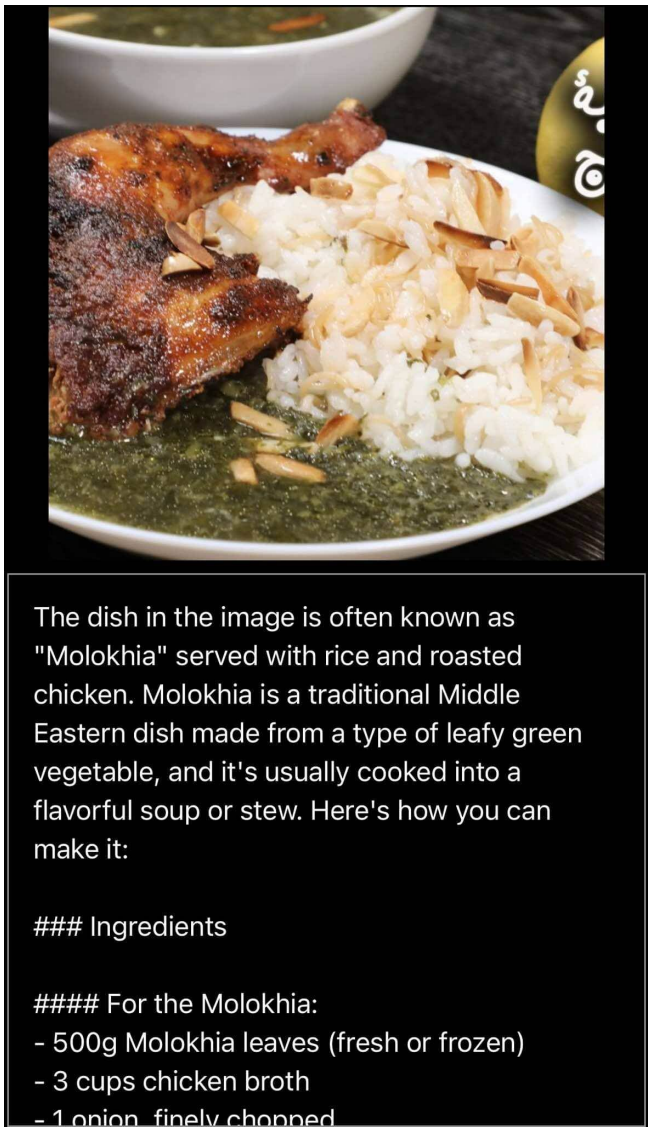

Figure 2.1: An example of AI Photo Recipe Identifier meal recognition.

2.3 ChefMate AI

The *ChefMate AI* [3] application is designed to recognize meal components by scanning an image of the dish. Once the ingredients are identified, the app processes them to suggest a variety of meals that can be prepared using those ingredients. Users are then able to select the most accurate dish from the presented options, after which the app generates a detailed recipe for that specific selection.

ChefMate AI offers several key features that enhance user interaction and functionality:

- **Camera Scan** – captures real-time photos of meals for analysis.
- **Retake Picture** – allows users to retake a photo before processing.
- **Picture Check** – verifies the clarity and usability of the selected image.
- **Generated Recipes** – provides a list of meal suggestions based on detected ingredients.
- **Recipe Description** – displays detailed cooking instructions and ingredient information.
- **Add to Favorites** – enables users to save preferred recipes for quick access.

2.4 ingreGenius

The *ingreGenius* [4] application begins by prompting users to either sign in or create a new account to access its features. Once authenticated, users can scan a meal using their device's camera to identify its ingredients. By selecting the "Identify Meal" option, the app processes the image and generates a comprehensive report containing key nutritional and culinary details. The generated report typically includes the name of the dish, a brief description, estimated calorie count, cooking duration, nutritional information, a complete list of ingredients, and step-by-step recipe instructions. Additionally, users can save their favorite meals within the app for future reference.

The application's key features include:

- **Camera Scan** – captures meal images for ingredient analysis.
- **Generated Recipes** – provides meal suggestions based on identified ingredients.
- **Cooking Time** – estimates the time required to prepare the meal.
- **Calories** – calculates the calorie content of the identified meal.
- **Detailed Ingredients** – lists all components used in the dish.
- **Recipe Description** – includes detailed cooking instructions.
- **Save Recipes** – allows users to store recipes for later use.
- **Add to Favorites** – offers quick access to preferred meals.

Figure 2.2 shows an example of pizza meal recognition by the application ingreGenius, it gives the nutritional information in that meal, as its ingredients, reciepe and other information as cooking time

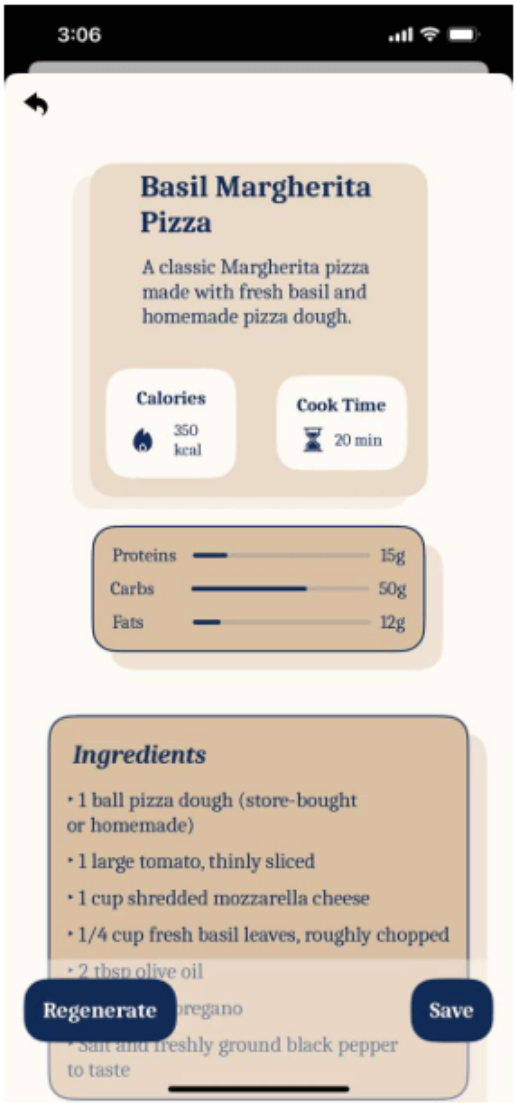

Figure 2.2: Pizza meal recognition by IngreGenius

2.5 SideChef

SideChef is an AI-powered cooking assistant application aimed at making meal preparation easier and more accessible for users of all skill levels. It offers step-by-step recipe instructions, interactive cooking support, and personalized meal planning. By combining user-friendly features with artificial intelligence, SideChef empowers individuals to cook confidently and efficiently, even if they are beginners in the kitchen.

The application's core features include:

- **Step-by-Step Recipes** – clear instructions to guide users through cooking.
- **Ingredient-Based Recipe Search** – allows users to search recipes using available ingredients.

- **Meal Planning** – enables users to organize weekly or daily meals.
- **Voice Command Support** – facilitates hands-free interaction during cooking.
- **Dietary Preferences** – offers filters for specific diets such as vegetarian, keto, or gluten-free.
- **Edit Recipes** – lets users customize recipe steps and ingredients.
- **Cooking Timers** – built-in timers to help manage cooking time effectively.
- **Save Favorite Recipes** – allows bookmarking of preferred meals.
- **Community Features** – lets users share and explore recipes from other home cooks.

User Capabilities:

- Discover new and diverse recipes.
- Cook confidently with guided support.
- Customize and edit recipes based on personal needs.
- Plan meals in advance to stay organized.
- Save time in meal preparation and shopping.
- Share recipes with the cooking community.

Benefits to Users:

- Simplifies the overall cooking process.
- Helps users manage their time more efficiently.
- Contributes to reducing food waste through smart planning.
- Promotes healthy eating habits with personalized options.
- Builds culinary confidence, especially for beginners.

2.6 FOODAI

FOODAI is an AI-powered calorie tracker and nutrition assistant designed to analyze food and deliver comprehensive nutritional insights. By leveraging artificial intelligence, the application assists users in monitoring their calorie intake, understanding the nutritional value of their meals, and making informed dietary decisions. It is particularly useful for individuals who aim to maintain a healthy lifestyle or achieve specific dietary goals.

Key Features:

- **Food Scanning** – detects food items from images for analysis.

- **Calorie Tracking** – monitors daily calorie intake.
- **Nutritional Information** – provides detailed nutritional data per meal.
- **Meal History** – records and stores previously scanned meals.
- **AI Nutritionist** – offers personalized nutritional advice.
- **Dietary Preferences** – allows customization based on user diet types.
- **Recipe Suggestions** – recommends meals based on scanned food items and dietary needs.

User Capabilities:

- Track and manage daily calorie consumption.
- Analyze nutritional content of meals.
- Receive personalized health and dietary advice.
- Explore new and suitable recipe suggestions.
- Maintain a history of meals for progress tracking.

App Functionalities:

- Performs AI-driven food analysis from images.
- Calculates and tracks caloric content.
- Provides real-time nutritional insights.
- Delivers customized dietary recommendations.
- Supports users with intelligent meal planning and goals.

User Benefits:

- Simplifies the process of calorie tracking.
- Encourages healthier eating habits.
- Offers personalized nutrition planning.
- Promotes dietary variety.
- Helps users meet and sustain dietary objectives.

2.7 Summary and Comparative Analysis

To facilitate a clearer comparison and enhance the analysis of the previously discussed applications, Table 2.1 summarizes and contrasts the key features offered by each application. By organizing this information in a structured format, it becomes easier to identify shared functionalities, unique offerings, and potential gaps, ultimately aiding in the evaluation of how these applications relate to and inform the development of our own meal recognition system.

Table 2.1: Comparison of key features across selected meal recognition and nutrition applications.

Feature	**AI Photo**	**ChefMate AI**	**ingreGenius**	**SideChef**	**FOODAI**	**Our Application**
Take Photo	✓	✓	✓	✓	✓	✓
Upload Photo	✓	✓	✓	✓	✓	✓
Image Recognition	✓	✓	✓	✓	✓	✓
Ingredient Identification	✓	✓	✓	✓	✓	✓
Recipe Suggestions	✓	✓	✓	✓	✓	✓
Nutrition Information	✗	✗	✓	✓	✓	✓
Step-by-Step Instructions	✗	✗	✓	✓	✓	✓
User Preferences	✗	✓	✓	✓	✓	✓
Shopping List	✗	✗	✗	✓	✗	✗
Voice Assistance	✗	✗	✗	✓	✗	✓
Social Sharing	✗	✗	✗	✓	✗	✓
Photo Cropping/Editing	✓	✗	✗	✓	✗	✓
Photo History	✓	✗	✓	✓	✓	✓
Cooking Time	✗	✗	✓	✓	✓	✓
Edit Recipe	✗	✗	✗	✓	✗	✗
Save Recipes	✓	✓	✓	✓	✓	✓
Allergen Detection	✗	✗	✗	✗	✗	✓
AI assistant	✗	✗	✗	✗	✗	✓

Features Descriptions:

- **Allergen Detection:** Identifies potential allergens in the ingredients found in the meals.
- **AI Assistant:** An integrated AI assistant that answers user questions about meals, recipes, ingredients, and cooking tips.

Chapter 3

Background

Contents

3.1 Dataset 13
3.1.1 Arabic Food 101 13
3.1.2 Multi-Class Food Image Dataset 13
3.1.3 UEC FOOD 256 Dataset 14
3.1.4 Cakes Dataset 14
3.1.5 Food-101 Dataset 14
3.1.6 Sri Lankan Food Image Dataset 14
3.1.7 Food Image Classification Dataset 15
3.1.8 Fast food Dataset 15
3.1.9 Custom Collected Dataset from Shutterstock and Google images 15
3.2 Firebase: Overview and Integration Strategy 15
3.2.1 Introduction to Firebase 15
3.2.2 Rationale for Using Firebase in MealLens 15
3.2.3 Key Firebase Services and Benefits 16
3.2.4 Cost Structure: The Free Tier 16
3.2.5 Using Firebase for Authentication and User Management . . . 16
3.2.6 Firebase Authentication Dashboard 17
3.3 APIs 18
3.3.1 Spoonacular API 18
3.3.2 Edemam 19
3.3.3 mealDB 19
3.4 App design Tools 19
3.4.1 Figma 19
3.4.2 Flutter 19
3.5 Image Classification Model 20
3.5.1 Azure Custom Vision 20
3.5.2 Recognition Process 22

3.1 Dataset

For this project, a comprehensive multi-class food image dataset was utilized, formed as a combination of the datasets specified in this chapter along with a set of images collected manually from online sources such as Google. The combined dataset contains 61,906 training images organized into 223 distinct food categories, covering a wide variety of dishes from different cuisines, including traditional Arabic meals, popular desserts, and international foods ensuring that the model is exposed to various cuisines, presentation styles, and cooking methods.

To maintain consistency and improve model generalization, each meal category was standardized to contain approximately 300 images whenever possible. This helped reduce class imbalance, ensuring that no single class dominates the training process, and provided sufficient samples for effective model learning. It is worth noting that, due to the nature of food photography and the use of multiple sources, some images appeared in more than one meal category. These overlapping or visually similar images were carefully curated to prevent confusion during model training and to maintain the integrity of each class.

Additionally, the dataset incorporates images with a variety of backgrounds, lighting conditions, and perspectives, simulating real-world scenarios where meals are photographed in different environments. This diversity enhances the robustness of the model, allowing it to recognize meals accurately even when presented under challenging or unfamiliar conditions. Overall, the final dataset represents a comprehensive and balanced collection of meal categories, providing a strong foundation for training a reliable and high-performing food recognition system. as listed in Appendix 9. Below are the datasets that we used to construct our final dataset.

3.1.1 Arabic Food 101

The "Arabic Food 101" [5] dataset features comprehensive information on 19 renowned and traditional Jordanian dishes, showcasing the extensive richness of Arabic cuisine. It portrays local ingredients, cooking styles, and cultural factors, rendering it a valuable database for food analysis. In our project, this data set plays a fundamental role in training machine learning models to effectively identify Jordanian meals and suggest original recipes. With the use of structured ingredient and preparation method data from "Arabic Food 101," [5] the AI system can further improve its recognition of dishes based on images and offer context-specific cooking instructions. The public domain license (CC0) of the dataset also allows for easy integration into the project, promoting the objective of combining AI technology with conventional cooking knowledge.

In this work, we incorporated all unique recipes from the Arabic Food 101 dataset into our training images. Appendix 1. lists all dishes included.

3.1.2 Multi-Class Food Image Dataset

The Multi-Class Food Image Dataset [6] contains 41,900 images across 20 distinct food categories, including: Bacon, Banana, Bread, Broccoli, Butter, Carrots, Cheese, Chicken, Cucumber, Eggs, Fish, Lettuce, Milk, Onions, Peppers, Potatoes, Sausages, Spinach, Tomatoes, and Yogurt. The dataset was created using manual web-scraping from Google

Images, utilizing diverse search terms such as "item on white background", "item isolated", and "item in hand" to ensure variety and capture different real-world imaging scenarios.

This dataset has been widely applied in mobile-based food recognition systems, where user-taken food photos are classified into one of the available categories, and the recognized items are subsequently transformed into recipe suggestions.

For our project, we selected only three categories from this dataset to include in our training images.The selection was limited because the other categories primarily consist of individual ingredients or items that do not represent complete meals, whereas our project focuses on recognizing full dishes suitable for recipe generation and meal suggestions. Appendix 2 lists the chosen food items.

3.1.3 UEC FOOD 256 Dataset

The UEC FOOD 256 dataset [7] is a large-scale collection of food images containing 256 food categories, each accompanied by bounding box annotations. It represents a wide range of Japanese and international dishes and is widely used for food recognition research. For this project, we selected a subset of representative meals from the dataset listed in Appendix 3. The other categories were excluded because they consist mainly of highly regional or less internationally recognized dishes, many of which are very specific to Japanese cuisine and may not be familiar to the target users of our application

3.1.4 Cakes Dataset

The Cakes Dataset [8], available on Kaggle, is a curated collection of cake images designed for computer vision tasks, particularly image classification and potentially object detection. This dataset represents an excellent resource for exploring automated machine learning capabilities in food recognition, retail applications, and culinary AI systems. We selected a subset of representative recipes from the dataset, listed in Appendix 4.

3.1.5 Food-101 Dataset

The Food-101 [9] dataset is a widely recognized benchmark in computer vision for food recognition tasks, containing 101,000 images across 101 food categories with 750 training and 250 test images per category. This dataset provides a comprehensive resource for evaluating automated machine learning systems in culinary image classification, offering real-world variations in lighting, presentation, and background. For our study, we selected a diverse subset of 20 representative food categories from the dataset, as detailed in Appendix 5.

3.1.6 Sri Lankan Food Image Dataset

The **Sri Lankan Food Image Dataset** [1] is a valuable resource of manually collected and labeled food images. This dataset was integrated to diversify our model's training data with traditional Sri Lankan cuisine. Specifically, we incorporated four distinct meal categories from this dataset to enhance our model's recognition capabilities. The selected categories and their corresponding image counts are detailed in Appendix 7.

3.1.7 Food Image Classification Dataset

The **Food Image Classification using ResNet50** dataset [2] is a specialized collection developed for training deep learning models on Indian cuisine recognition. Created by Gaurav Duttakiit, this dataset focuses on high-quality images of traditional Indian dishes. We integrated three key categories from this dataset to enrich our model's understanding of popular Indian foods, particularly North Indian cuisine. These additions help the model distinguish between similar-looking dishes and improve its cultural food recognition accuracy. The selected categories and their image counts are presented in Appendix 8.

3.1.8 Fast food Dataset

The Fast Food Classification Dataset [10] is a labeled image dataset created for training machine learning or deep learning models to recognize fast food items from photos. It contains images of 10 different fast-food categories, suitable for use in image classification tasks. The selected categories and their image counts are presented in Appendix 10.

3.1.9 Custom Collected Dataset from Shutterstock and Google images

Due to the limited representation of Arabic dishes in existing food datasets, we collected additional images from Shutterstock and Google Images to enhance our training dataset. These sources provide professionally captured photographs covering a wide range of cuisines and presentation styles.

A substantial number of images were gathered for each meal category, significantly increasing the size and diversity of the training corpus. This approach ensures that Arabic dishes, which were underrepresented in other datasets, are adequately included. The supplementary images help the model better recognize food items under various visual conditions and improve its performance on culturally specific dishes, as detailed in Appendix 6.

3.2 Firebase: Overview and Integration Strategy

3.2.1 Introduction to Firebase

Firebase is a comprehensive Backend-as-a-Service (BaaS) platform developed by Google, designed to accelerate the development of mobile and web applications. It provides a suite of integrated tools and services that handle common backend tasks such as user authentication, real-time databases, cloud storage, hosting, and serverless functions. By leveraging Firebase, developers can focus on building user-centric features without managing complex server infrastructure, significantly reducing development time and operational overhead.

3.2.2 Rationale for Using Firebase in MealLens

The primary objective of integrating Firebase into the **MealLens** project was to establish a robust, scalable, and secure backend foundation while accelerating the development lifecycle. Specifically, Firebase was chosen to:

- **Expedite Development:** Rapidly implement essential features like user authen-

tication and data persistence using pre-built, well-documented SDKs.

- **Ensure Cross-Platform Consistency:** Utilize unified Firebase SDKs for Flutter, Android, iOS, and Web to maintain identical behavior across all target platforms.
- **Leverage Scalable Infrastructure:** Build on Google's cloud infrastructure to ensure the application remains performant and available as user adoption grows.
- **Minimize Operational Complexity:** Eliminate the need for manual server management, database administration, and security implementation for core backend services.

3.2.3 Key Firebase Services and Benefits

- **Firebase Authentication:** Provided a secure, out-of-the-box solution for managing user identities, supporting multiple sign-in methods (Email/Password, Google Sign-In, Anonymous). This saved considerable development effort and enhanced application security.
- **Cloud Firestore:** Served as a flexible, scalable NoSQL database for storing structured user data (profiles, saved recipes, history). Its real-time synchronization capability ensured instant data updates across user devices.
- **Cloud Storage:** Offered secure and scalable object storage for user-uploaded meal images, simplifying file management and delivery.
- **Firebase Hosting:** Enabled fast, secure deployment of the web application with global CDN and automatic SSL certification.

3.2.4 Cost Structure: The Free Tier

Firebase operates on a freemium model, with a **Generous Free Tier (Spark Plan)** that is typically sufficient for development, testing, and initial launch phases. This tier includes quotas such as 10,000 monthly active users for Authentication, 1 GiB of stored data for Firestore, and 10 GB of hosting storage. For the MealLens project, this allowed full access to enterprise-grade backend services at **zero cost**. Costs are only incurred if usage exceeds these limits, transitioning to a pay-as-you-go model (Blaze Plan), making it an economically viable choice for student projects and startups.

3.2.5 Using Firebase for Authentication and User Management

Building upon the Firebase infrastructure, the application implements a dedicated authentication layer. This component is crucial for delivering a secure and seamless user experience, enabling sign-up, sign-in, guest access, and social login.

The authentication architecture was designed using a layered *Service Model* in Flutter, promoting code reusability and separation of concerns. The core logic is encapsulated in the `lib/services/auth_service.dart` file.

Core Implemented Functions:

1. **Sign-Up with Email and Password:** Allows new users to create an account. Credentials are securely stored in **Firebase Authentication**, and the user profile is immediately updatable (e.g., with a display name).

2. **Sign-In:** Validates user credentials against Firebase's secure backend. Upon successful verification, it establishes an authenticated session and returns a `User` object for client-side personalization.

3. **Guest Sign-In:** Facilitates immediate access to core app functionalities without registration. Firebase generates a unique, temporary anonymous identifier, allowing for a personalized session without compromising user privacy.

4. **Google Sign-In:** Simplifies authentication by leveraging users' existing Google accounts. The process integrates the `google_sign_in` package with Firebase Auth for a secure, streamlined flow.

5. **Authentication State Management:** Utilizes Firebase's `authStateChanges` stream to monitor the user's login status dynamically. This enables the UI to react instantly—for example, by redirecting to the home screen upon successful login or showing the login screen after sign-out.

Integration and Benefits: This modular approach, built on Firebase, streamlined the development of a secure authentication system. It ensured best practices in security (like credential hashing) were followed by default, while providing the flexibility needed for features like guest mode. The integration of Firebase Authentication formed the cornerstone of the MealLens user management system, effectively balancing security, user privacy, and a frictionless onboarding experience.

3.2.6 Firebase Authentication Dashboard

Figure 3.1 shows the Firebase Authentication dashboard for the MealLens application, displaying registered users with their email identifiers, authentication providers, account creation dates, last sign-in timestamps, and unique user UIDs. This dashboard demonstrates the successful integration of Firebase Authentication, enabling secure user management, session tracking, and seamless sign-in experiences across multiple providers.

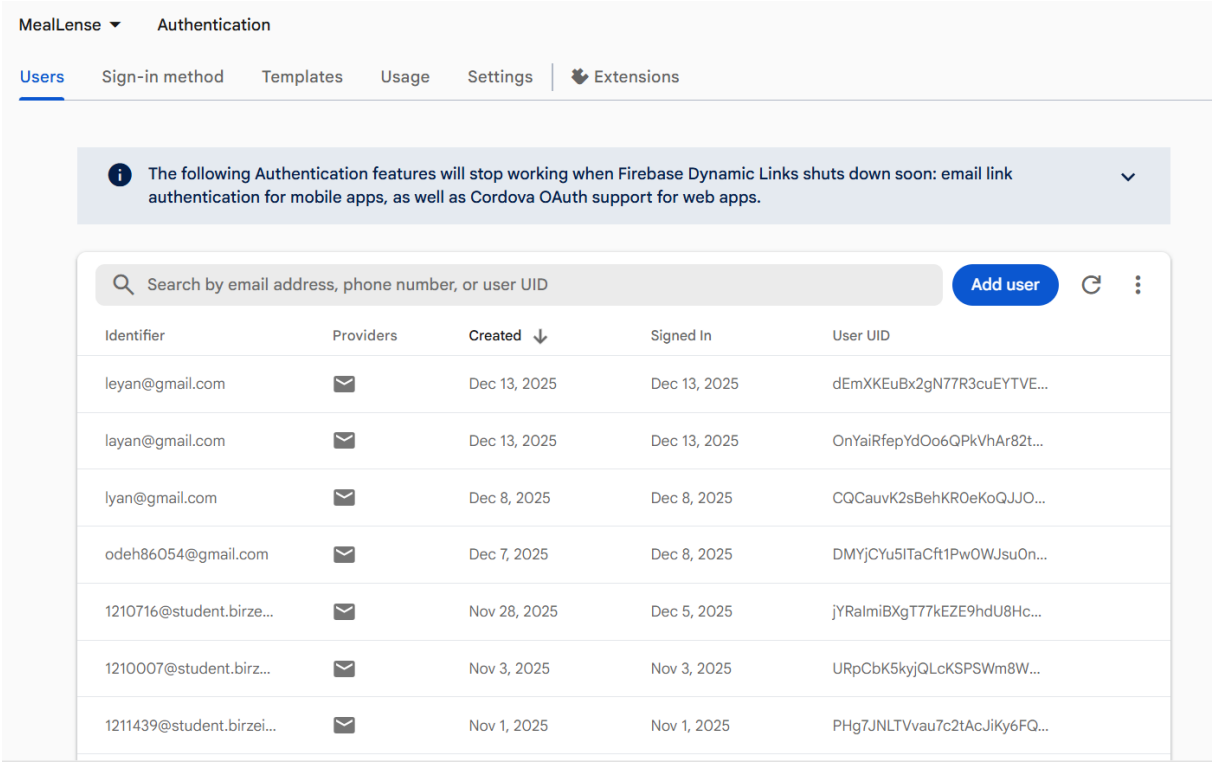

Figure 3.1: The Authentication dashboard for the MealLens application on Firebase

3.3 APIs

In software development, APIs—short for Application Programming Interfaces—play a big role in connecting different systems. They allow one program to ask another for data or services, like when a mobile app gets recipe information from an online source. Instead of building everything from scratch, developers use APIs to save time and keep their code organized. A good example would be using a food recipe API to display meal details without storing the entire database yourself. APIs simplify communication between tools and services, making it easier to build smart, connected applications. Overall, they're a practical way to make modern programs work smoothly together. There are many APIs that are related to food recipes and instructions in the internet such as Spoonacular API [11], Edamam Food and Recipe API [12], Yummly API [13], TheMealDB [14] and many others. In the following sections, two of these APIs will be discussed in detail.

3.3.1 Spoonacular API

The spoonacular [11] API is a universal food and recipe that allows developers to build applications capable of searching for recipes, retrieving recipe detailed ingredients and cooking instructions, as well as analyzing nutritional information. The spoonacular API has advanced features such as meal planning, diet filtering (e.g., vegan, gluten-free), and food-related data such as wine pairings.

Spoonacular [11] uses a custom-built food database that combines information from:

- Public recipe websites (like *Foodista*, *Taste of Home*, etc.)

- USDA FoodData Central for nutrition info
- Their own curation and user-submitted content

3.3.2 Edemam

The Edamam API [15] is an online service that provides developers with access to large databases of food and nutrition information. It allows applications to analyze recipes, calculate calories and nutrients, and search for detailed data about ingredients and meals. The API is commonly used in health, diet, and fitness apps to help users track what they eat and make better food choices. By sending simple requests to the Edamam API, developers can quickly integrate reliable nutrition analysis features into their own mobile or web applications.

3.3.3 mealDB

TheMealDB [16] is a free and open online recipe database that provides access to a lot of meals from around the world through a simple and developer-friendly API. It offers detailed information for each recipe, including the meal name, ingredients with measurements, preparation instructions, cuisine type, category, images, and even YouTube video links for visual.

3.4 App design Tools

3.4.1 Figma

For our food recognition app project, we selected Figma as our primary design tool due to its cost-effectiveness, user-friendly interface, and robust collaboration features. As a cloud-based platform, Figma enabled seamless real-time collaboration, allowing our team to simultaneously edit designs, provide immediate feedback, and streamline decision-making processes without the need for file exchanges.

We utilized Figma to create comprehensive designs for all app screens, including the camera interface and food details page. A key advantage was Figma's interactive prototyping capabilities, which allowed us to link screens and simulate user navigation flows, enabling us to validate the user experience prior to development. Additionally, Figma's component and style management features ensured design consistency across all interfaces, maintaining a cohesive visual identity. By leveraging Figma, we efficiently transformed our concepts into professional, developer-ready designs, providing clear visual references to guide the development process [17].

3.4.2 Flutter

To ensure our AI-Powered Meal Recognition & Recipe Assistant works seamlessly across both Android and iOS devices, we've chosen Flutter, Google's open-source UI framework, for its ability to deliver high-performing, native-feeling apps from a single codebase. Flutter's cross-platform compatibility allows us to develop one unified version of the app that runs smoothly on both operating systems, eliminating the need for separate codebases while maintaining consistent performance—especially crucial for features like real-time food recognition. The framework's customizable widget system enables us to design an

efficient and visually appealing interface, complete with real-time camera functions (using plugins like camera and tflite), smooth animations for recipe browsing, and adaptive themes for light or dark mode. Flutter also simplifies integrating AI/ML and backend services, such as TensorFlow Lite for on-device food recognition (reducing latency and enhancing privacy), Firebase ML Kit or Spoonacular APIs for recipe details, and local databases like Hive or SQLite for offline access. Development is further streamlined by features like Hot Reload for instant updates, a robust plugin ecosystem for tasks like authentication and state management, and scalability for future additions like voice-guided cooking. While challenges exist—such as accessing native hardware features via platform channels or optimizing AI models for lower-end devices—Flutter's blend of cross-platform support, UI flexibility, and backend integration makes it ideal for creating a fast, scalable, and user-friendly meal planning app that prioritizes accessibility, performance, and design [18].

3.5 Image Classification Model

3.5.1 Azure Custom Vision

Introduction to Azure Custom Vision

Azure Custom Vision [19], developed by Microsoft, is a cloud-based service designed for building custom image classification and object detection models. The service leverages **Automated Machine Learning (AutoML)** to automatically handle model training, optimization, and selection of suitable algorithms and hyperparameters, allowing developers to create accurate models without deep expertise in machine learning. Azure Custom Vision supports both image classification (single-label and multi-label) and object detection, enabling precise recognition of objects or regions within images. Users can upload their own custom datasets and iteratively improve model performance by adding new images and retraining. The service provides a web-based interface and SDKs, making it easy to train, test, and evaluate models. Once trained, models can be deployed as cloud APIs for real-time predictions or exported for edge deployment to run on devices with limited resources.

Dataset Management in Custom Vision

When images are uploaded and tagged, Custom Vision organizes them into distinct datasets: a **training set**, an automatically generated **validation set** (used internally for model evaluation), and an optional **testing set** if provided by the user. The **training set** is used to teach the model to recognize patterns and features in the images, while the **validation set** helps the system evaluate model performance during training and guides the AutoML process in selecting optimal algorithms and hyperparameters. If a **testing set** is provided, it allows users to independently assess model accuracy on unseen data, giving a realistic measure of performance.

AutoML Architecture Selection

Depending on the project type, AutoML explores multiple state-of-the-art model architectures. For classification tasks, it evaluates variants of **ResNet**, **MobileNet**, and **EfficientNet**, selecting the most suitable architecture based on factors such as dataset

size, image resolution, and domain type (e.g., Food, Retail, General).

The AutoML process automates model selection, hyperparameter tuning, and optimization, so users do not need to specify algorithms manually. It systematically tests different architectures through iterative training, evaluating each model's performance using the validation set. During this process, AutoML considers metrics such as accuracy, precision, recall, and inference speed to balance model performance with computational efficiency. This enables the system to recommend or deploy a model that is both accurate and efficient, suitable for real-time or edge applications.

Classification Algorithms

Azure AutoML employs a diverse range of classification algorithms:

Linear Models

- **Logistic Regression**: Baseline linear classifier with probabilistic outputs
- **Linear SVC**: Support Vector Classifier with linear kernel for high-dimensional data
- **Stochastic Gradient Descent (SGD)**: Efficient linear classifier with online learning capability

Tree-Based Methods

- **Decision Tree**: Interpretable tree-based classifier with explicit decision rules
- **Random Forest**: Ensemble of decision trees with bagging for improved stability
- **Extremely Randomized Trees:** Highly randomized trees that reduce variance further
- **Light GBM**: Gradient boosting framework optimized for speed and efficiency
- **Gradient Boosting**: Sequential ensemble method that minimizes prediction errors
- **XGBoost**: Optimized gradient boosting with regularization and parallel processing

Instance-Based Methods

- **K Nearest Neighbors**: Non-parametric method based on similarity measures

Probabilistic Models

- **Naive Bayes**: Enhanced Bayesian classifier assuming feature independence

Support Vector Machines

- **Support Vector Classification (SVC)**: Kernel-based method for non-linear boundaries

Automated Hyperparameter Optimization

During each training iteration, Custom Vision automatically adjusts key hyperparameters including learning rate, batch size, number of epochs, image augmentation options, and network depth or width for supported models. Each iteration represents a distinct AutoML trial with a unique parameter configuration. A dedicated validation set—automatically generated and held out from training—is used in every trial to evaluate model performance consistently and efficiently.

Model Evaluation and Selection

After completing all AutoML trials, Custom Vision evaluates each model using metrics such as precision, recall, and average precision (AP). All trials are compared, and the best-performing model is selected according to project requirements: precision-focused for balanced datasets, recall-focused for critical applications (e.g., medical or safety), or overall performance for general use cases.

Model Optimization for Deployment

Once the optimal model is selected, Custom Vision performs an additional optimization step when exporting models for mobile or edge deployment. Platform-specific optimizations are applied to make models smaller, faster, and more energy-efficient:

- **TensorFlow Lite**: Automatically applies pruning to remove unnecessary neurons and lightweight architectural modifications.
- **ONNX**: Applies quantization (converting 32-bit to 8-bit numbers) and graph optimization to remove redundant operations.
- **iOS CoreML**: Automatically fuses layers (e.g., convolution and activation) into single optimized operations.

Training Modes

Custom Vision offers two training modes:

- **Quick Training**: Executes fewer AutoML trials with minimal tuning, resulting in faster training times.
- **Advanced Training**: Conducts an extended AutoML search across more architectures and deeper hyperparameter tuning.

3.5.2 Recognition Process

- **Feature Extraction:** After receiving the image, the Custom Vision model analyzes it using a deep learning network to extract meaningful visual features such as color patterns, textures, shapes, and distinctive components that characterize different meals.
- **Classification:** The extracted features are compared against the learned patterns of all 223 trained meal tags. Based on this comparison, the model generates a ranked list of possible meal tags and a confidence score for each tag (ranging from 0 to 1).

Chapter 4

System Structure

Contents

4.1 Application Features 24
4.2 Mobile Features 26
4.3 Service Features 27
4.4 Use-case Diagram 27
4.5 Image Recognition Process Using Custom Vision 28
4.5.1 Custom Vision Dataset 28
4.5.2 Sending Images to the Model 29
4.6 System Architecture Overview 29
4.6.1 Application architectural Layers 30
4.6.2 Data Flow 30
4.6.3 System Sequence Diagram 31
4.6.4 Cost Management 32
4.7 System Database Management 33
4.7.1 Database Structure Overview 33
4.7.2 Collections Description 33
4.7.3 Relationships and Data Flow 35
4.7.4 Database Schema Diagram 36
4.8 UI/UX Design 36

This chapter presents the internal structure and main elements of the created application in detail. It describes the main features implemented on the application level, highlights the mobile-specific features leveraged to enhance usability, and shows the functionality handled by the server side. The graphical user interface (GUI) is also presented, demonstrating how the system offers an intuitive and clear-cut user experience.

4.1 Application Features

Application features are employed to explain the basic functionalities built within the app itself for fulfilling user requirements. They include tools and behavior developed by the developers. Below are the features used in our recipe application:

- **Sign Up**
  Users can create a new account using email or social login to save their preferences, recipes, and history securely.
- **Sign In**
  Existing users can log in to their account to access saved recipes, previous uploads, and personalized settings.
- **Continue as Guest**
  Users can use the app without creating an account, allowing quick access to basic features like photo recognition and recipe suggestions.
- **Help link**
  Users can access our website without creating an account, users can search for an FAQ question.the website gives information about the project, team members, and the supervisor.
- **Profile Page**
  Users that have an account can view and edit their personal information, preferences, and saved recipes.
- **Take Photo**
  The users capture food pictures directly from the device camera.
- **Upload Photo**
  The users can upload previously clicked food photos from the phone gallery.
- **Photo Cropping/Rotating**
  Users can crop and edit an image before sending it for recognition. The rotation feature allows clockwise rotation of the image.
- **Image Recognition**
  The app identifies the food from the uploaded or clicked image.
- **Ingredient Identification**
  The system provides the possible ingredients and their quantities according to the identified food image.
- **Recipe Suggestions**
  The app provides suitable recipe ideas according to the identified dish.
- **Nutrition Information**
  The app estimates and displays nutritional values related to the identified meal.
- **Allergen Detection**
  The app detects possible allergens present in the ingredients or recipes identified.

- **Step-by-Step Instructions**
  Users are given step-by-step, simple-to-follow cooking instructions for the suggested recipe.

- **Feedback**
  Users can submit their comments, suggestions, and ratings about the app or recipes.

- **Social Sharing**
  Users can share their recipe results to social applications.

- **User Preferences**
  Users can personalize their experience, for example, switching to dark mode and can disable History results.

- **Text-To-Speech**
  The app includes a simple voice assistant that can read the meal result aloud to the user.

- **Photo History**
  History of previously scanned or uploaded food pictures is stored within the app.

- **Save Recipes**
  Users can save favorite recipes within the app for future use.

- **AI Assistant Page**
  The app provides an AI-powered assistant that can answer questions, suggest recipes, and give cooking guidance interactively.

- **Admin Management Features**
  Based on the AdminSettingsScreen, the following administrative features are available:

  - **Data Management**
    Local Database Management Admins can add, edit, or delete recipes in the local database through the Manage Local Database interface.
  - **Recipe Statistics**
    Admins can view total recipes count, total users count, and deleted recipes statistics.
  - **Enhanced Recipe Management**
    The admin dashboard provides comprehensive recipe management capabilities:
    * **Enhanced Search:** Search functionality now includes English titles.
    * **Full Recipe Editing:** A dedicated editing window allows modification of all recipe fields (title, description, ingredients, instructions, nutrition, etc.).The system saves only the modified data, with immediate updates to the Firestore database.
    * **Recipe Creation:** A complete form for adding new recipes with support for all data fields including nutritional information, ingredients list, and cooking instructions.

    - **Updated Categories:** The main dish category has been removed. Recipe categories are now: Breakfast, Lunch, Dinner, Snack, Dessert , soup ,Salad ,Bread , Beverage and All.

- **User Interface Improvements**
  The admin interface and general application UI have been enhanced for better usability:
    - Improved recipe details display with better formatting.
    - Ingredients list is now clearly displayed in the detailed recipe view.
    - Enhanced form layouts and input validation across all administrative forms.
- **Usage Analytics**
  Admins can view comprehensive app usage statistics including user counts and activity metrics.
- **Web Integration**
  Website Access Direct link to open the MealLense website in an external browser.
- **Admin Authentication**
  Secure login and logout with Firebase authentication and audit logging.

4.2 Mobile Features

Mobile features play a crucial role when it comes to enhancing the performance and user experience of modern applications. They enable seamless interaction between the user and the app by leveraging the native capabilities of the mobile device. These features assist in improving accessibility, responsiveness, and personalization, thus making the application more user-friendly and efficient. By integrating with the native mobile infrastructure, the app can accomplish things more effectively and provide users with a continuous, context-aware experience that ties into their everyday habits.

Our application has several mobile features that are explained below:

- **Camera Access**
  Allows the app to use the camera on the phone to take a photo from within the app.
- **Gallery Access**
  Allows users to upload images saved on their phone's gallery.
- **Internet Connectivity**
  Enables the app to connect to online services, such as sending images to a server for processing.
- **Touchscreen Interface**
  Allows users to interact with the app through tap, swipe, and other touch events.
- **AI Integration**
  Connects the app to artificial intelligence models to ask and get information about food or recipes.

4.3 Service Features

The service features outline the primary abilities of your AI meal app, which is designed to simplify food tracking and cooking. These include AI meal recognition, personalized recipe generation, and nutrition analysis for facilitating healthy eating. Additional features like allergen alerts, logging of meals, and user feedback mechanisms enhance safety, convenience, and continuous improvement of the app.

- **AI Meal Recognition** — Advanced image recognition analyzes food images to identify dishes and meals names.
- **APIs Service**- Return step-by-step recipes with weighed ingredients and cooking instructions for recognized meals, provide nutritional analysis and possible allergies in the meal.
- **Profile Management** — Allows users to change account details, or delete data or account with one-click controls and admins to enhance privileges for application management .

Data Management Services

- **Firebase Backend Services** — Cloud-based storage for recipes, user information, feedback, and admin logs.
- **Cloud Firestore Integration** — Cloud-based database for storing and retrieving application data, recipes, user information, and admin logs.
- **Search Functionality** — Provides comprehensive search capabilities across the recipe database.

Administrative Services

- **Admin Dashboard Services** — Admin dashboard for application management, accessible via the web or mobile.
- **Analytics Services** — Statistics for application usage, user activities, etc.
- **Audit Services** — Security audit of all activities for application management.

External Integration Services

- **Website Integration** — Integration of mobile application with the website for better brand management.
- **URL Launch Services** — Integration of mobile application with the browser for opening URLs.

4.4 Use-case Diagram

Use case diagram provides a high-level description of the system's functional operation by presenting the interactions among the application and the users (actors). It visually presents the core functionality offered by the system. The diagram helps identify system

boundaries and describe the expected actions for each user role and thus it is easy to understand the overall workflow of the application. Figure 4.1 shows the use case diagram of our application, the diagram shows the user features as well as system features

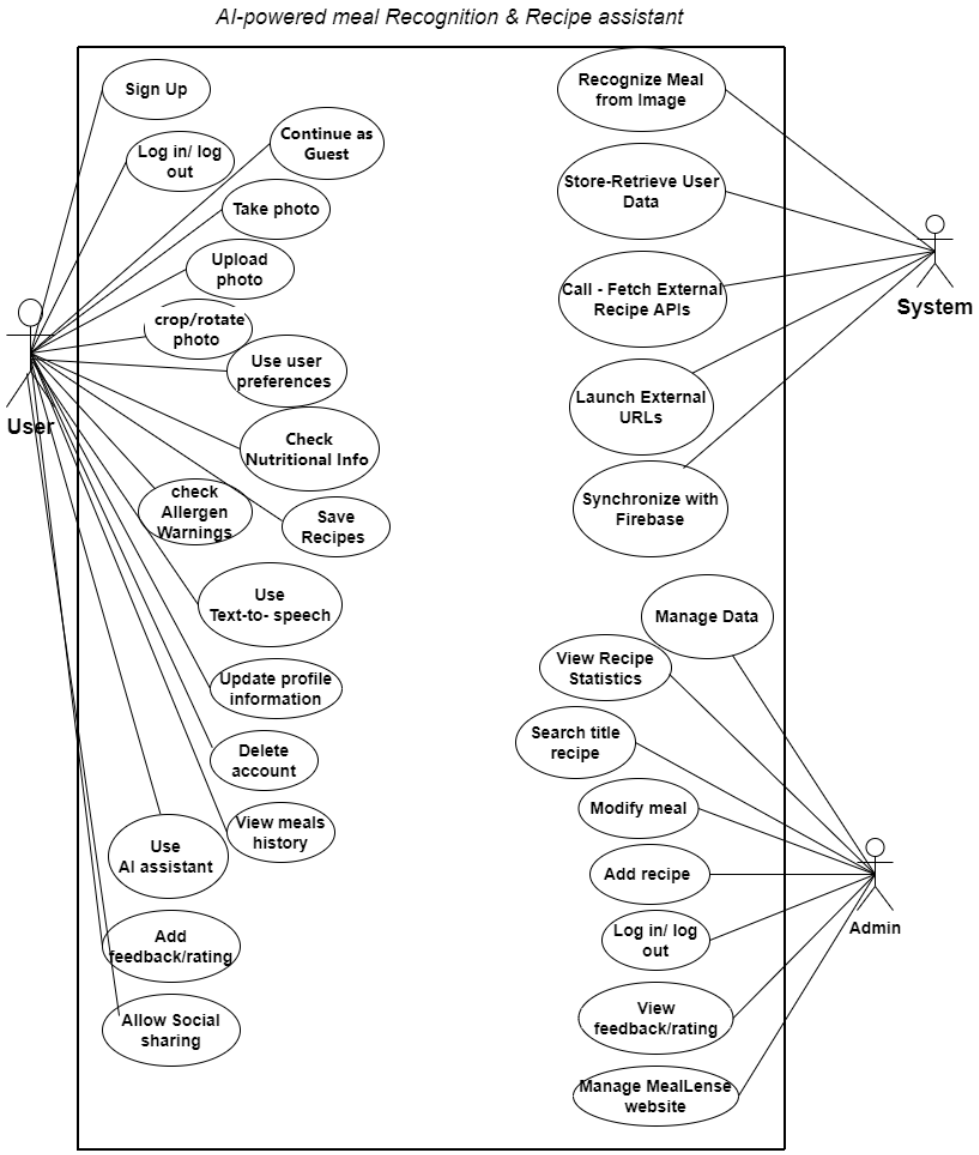

Figure 4.1: use case diagram

4.5 Image Recognition Process Using Custom Vision

In our application, image recognition is performed using the Microsoft Azure Custom Vision service. This service allows us to train a custom deep-learning model using images collected for each meal category.

4.5.1 Custom Vision Dataset

Our dataset comprises approximately 223 meal tags, each representing a distinct dish such as Kebab, Mansaf, Pasta, and others. For each tag, we collected more than 250 training images to capture diverse variations including different shapes, lighting conditions, plate

presentations, and camera angles.

4.5.2 Sending Images to the Model

User Uploads an Image
The user either takes a photo using the device camera or selects an existing image from the gallery.

Image Preprocessing
Before sending the image to Custom Vision:

- The image is converted to a binary format.
- The app ensures the image meets Custom Vision's size and format requirements (JPEG/ JPG/ PNG).

Sending the Image to the Custom Vision Prediction API
The app sends an HTTPS POST request to the Custom Vision Prediction Endpoint, which includes:

- The image file (as binary data)
- The Prediction Key
- The Project ID and Iteration Name of the trained model

Custom Vision Receives the Image
The Custom Vision backend processes the image using the trained deep-learning model associated with our project.

Returning the Prediction to the Application
Custom Vision sends the prediction results back to the application in the form of a JSON response. This response includes the top predicted meal, additional candidate meals, and their corresponding confidence scores. The application then displays all these results to the user.

The **confidence score** represents the model's level of certainty for each predicted meal. It is a numerical value between 0 and 1, where a higher value indicates greater confidence that the predicted meal matches the input image. For example, a confidence score of 0.95 means that the model is highly confident in its prediction, while lower scores indicate less certainty.

4.6 System Architecture Overview

The MealLens application implements a comprehensive, multi-layered architecture designed to provide robust food recognition and recipe retrieval services. This architecture harmoniously integrates mobile device capabilities with cloud-based services, creating a seamless user experience while ensuring scalability, reliability, and maintainability. The system's design follows modern software engineering principles, emphasizing modularity, separation of concerns, and efficient resource utilization.

4.6.1 Application architectural Layers

Frontend Layer

The presentation layer constitutes the user-facing interface, implemented using Flutter framework to ensure cross-platform compatibility across iOS and Android devices. This layer manages all user interactions, visual rendering, and immediate feedback mechanisms.

The presentation layer communicates exclusively with the business logic layer through well-defined interfaces, ensuring that UI components remain decoupled from business rules and data management concerns.

Service Layer

This layer manages all external service integrations, providing abstraction and fault tolerance for third-party dependencies. Each external service is encapsulated behind a dedicated adapter that handles communication protocols, error handling, and response parsing:

- **AI Recognition Service (Azure Custom Vision)**: Provides sophisticated image analysis capabilities for dish identification with confidence scoring
- **Authentication Service (Firebase Auth)**: Offers secure user authentication, account management, and social login integration.

Data Access Layer

The data access layer abstracts all data storage and retrieval operations, providing a unified interface for accessing various data sources while hiding implementation details. This layer supports multiple data source types:

- **External Recipe APIs**: Integrates with multiple recipe providers including Spoonacular, Edamam, and TheMealDB to ensure comprehensive recipe coverage.
- **Firebase Firestore**: Serves as the main repository for user profiles, favorites, history, and locally created recipes with real-time synchronization capabilities.
- **Local Caching System**: Implements in-memory caching with intelligent eviction policies to optimize performance for frequently accessed data.
- **Data Transformation**: Converts data between different formats and structures to ensure compatibility across various sources.

4.6.2 Data Flow

The system implements a unidirectional data flow architecture that ensures predictable state management and facilitates debugging and testing. The typical data flow for a recipe recognition request proceeds through the following stages:

1. User login The user has the choice to use the application by two ways, either sign in with his account or continue as a guest which limits the access of some features such as send feedback.

2. Image Acquisition and Preprocessing The user initiates the process by capturing a food image using the device camera or selecting an existing image from the gallery. The presentation layer handles image acquisition, performs basic preprocessing (rotation, cropping, format conversion), and validates image quality before transmission.

3. Dish Recognition Request The processed image is forwarded to the Azure Custom Vision service.

4. Multi-Source Recipe Retrieval Upon receiving dish identification results, the system processes recipe requests through a multi-stage retrieval mechanism that begins with a cache lookup, performing an immediate check against the in-memory cache for recently accessed recipes to ensure fast responses. If no match is found, the process moves to an external APIs fallback, at first a request is sent to Spoonacular API, if there is no match with the meal name or the number of requests exceeds the limit,then the request is sent to Edamam API, same as previous if issue occured it is sent to TheMealDB API using intelligent fallback logic to maximize the chances of retrieving relevant results. Finally, if external sources do not provide sufficient data, a local database search is performed through Firebase Firestore, applying multiple matching strategies to locate the most appropriate recipes.

5. APIs Results Extraction the APIs employed typically return structured data in JSON (JavaScript Object Notation) format, the JSON format is extracted and matched to the UI components.

6. User Preferences the user can do multiple actions, such as saving recipe result, send feedback, use text-to-speech feature which will read the meal results and other things.

4.6.3 System Sequence Diagram

Diagram 4.2 represents a sequence diagram of the MealLens application

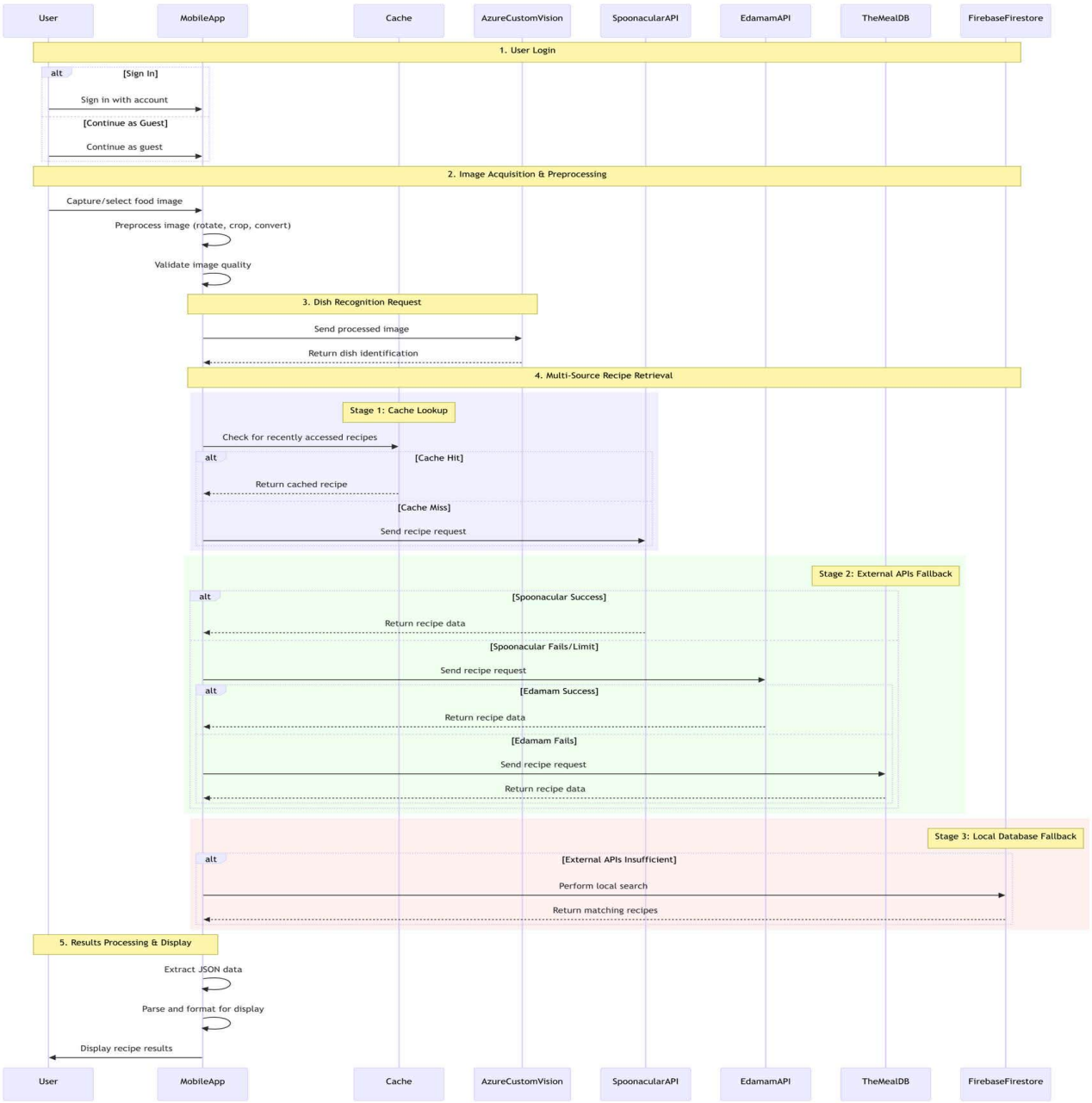

Figure 4.2: Complete System Architecture of MealLens Application

4.6.4 Cost Management

To ensure the system remains affordable and scalable, careful cost management strategies were applied throughout development. The meal recognition module was implemented using Microsoft Custom Vision, with a fixed allocated budget of $300, which covered model training, hosting, and prediction requests within the project scope. For recipe and nutrition data, three external APIs were integrated: Spoonacular, Edamam, and TheMealDB. Each of these services offers a free request tier, which was fully utilized to minimize operational costs. The system is designed with a fallback and load-balancing strategy: when the free request limit of one API is reached, the application automatically switches to another available API. This approach ensures continuous service availability while avoiding additional charges, making the overall solution both cost-efficient and

sustainable.

Table 4.1: API Free Tier Usage and Limits

API	**Trial / Free Tier**	**Rough Max Requests**
Spoonacular	Free tier	∼150 calls/day [20]
Edamam	Short trial / limited free	Varies by plan; no reliable free daily quota [15]
TheMealDB	Free unlimited (no limits published)	Unlimited usage [21]

4.7 System Database Management

4.7.1 Database Structure Overview

The database follows a hierarchical model combining user-specific subcollections with global collections. This design ensures secure data isolation and optimized query performance. The main collections include: **Users** for authentication Table 11, **Recipes** as the central repository Table 12, **Favorites** and **History** as user subcollections, **Ratings** for aggregated scores, **Feedback** for user suggestions Table 13, **Deleted_Recipes** for recovery Table 17, **Admin_Log** for administrative auditing, and additional collections for web platform functionality. 20

The architecture leverages Firebase Firestore's document-oriented model with real-time synchronization and offline persistence capabilities. Key design principles include hierarchical data organization for natural security boundaries, denormalization for read performance, and aggregated models for frequently accessed data.

4.7.2 Collections Description

Users Collection

The Users collection contains user authentication and profile data see Table 11. Each user document acts as the root for user-specific subcollections, establishing clear data ownership and security boundaries through Firebase Security Rules. The collection integrates with Firebase Authentication while extending functionality with application-specific metadata including usage statistics and preferences.

Recipes Collection

This collection stores comprehensive recipe information including ingredients, nutrition, and preparation instructions Table 12.

Feedback Collection

The Feedback collection gathers user suggestions about recipes and features, providing insights for continuous application improvement Table 13.

Favorites Subcollection

Stored under each user document, Favorites contains denormalized copies of saved recipes. The hierarchical design ensures users access only their own favorites while improving

query performance through single-document reads. Critical recipe fields are duplicated to eliminate join operations during frequent access patterns. 15

History Subcollection

This subcollection tracks user interactions with the food recognition system, recording scanned meals and recognized dishes for personalized experiences. 16

Ratings Collection

Ratings use an aggregated model storing sums and counts rather than individual records, enabling instant average calculation with minimal database operations. This design achieves O(1) complexity for rating displays by pre-computing aggregates during write operations.

Deleted Recipes Collection

This collection implements a soft-delete pattern, allowing recovery of removed recipes while maintaining administrative audit trails Table 17. Deleted recipes retain their original identifiers.

Security and Access Control

Role-based access control implements three user roles Table 18. Anonymous users have read-only access to recipes and ratings. Authenticated users can access their own data, read recipes, and write ratings/reviews. Administrative users possess full access to all collections including admin logs, with additional privileges for content moderation and user management.

Admin Log Collection

Admin_Log tracks all administrative actions, providing accountability and security monitoring for privileged operations. Each log entry includes comprehensive context including IP addresses, device information, and action parameters for complete auditability. 20

Web Platform Collections

Additional collections support web platform functionality 19: **admin_users** for website administrator authentication 20, **user_questions** for unanswered inquiries 21, **website_messages** for contact form submissions 22 and **website_faq** for knowledge base management 23.

Storage Strategy

User-uploaded meal images are processed through Azure Custom Vision API for recognition only and are not permanently stored. Recipe images reference external URLs with content delivery network (CDN) delivery, minimizing local storage requirements while focusing database resources on structured data.

4.7.3 Relationships and Data Flow

The database implements three relationship types. Hierarchical relationships organize user-specific data as subcollections under user documents, creating natural security boundaries. Reference relationships connect Favorites and History entries to Recipes using foreign keys while maintaining denormalized copies for performance. Aggregation relationships enable Ratings and Feedback collections to gather data from multiple users toward individual recipes, with pre-computed summaries for efficient access.

Data flow follows consistent patterns: user interactions create entries in History and update Favorites; recipe modifications trigger updates in denormalized copies; administrative actions generate audit trails; and statistical aggregates are periodically updated from source collections.

4.7.4 Database Schema Diagram

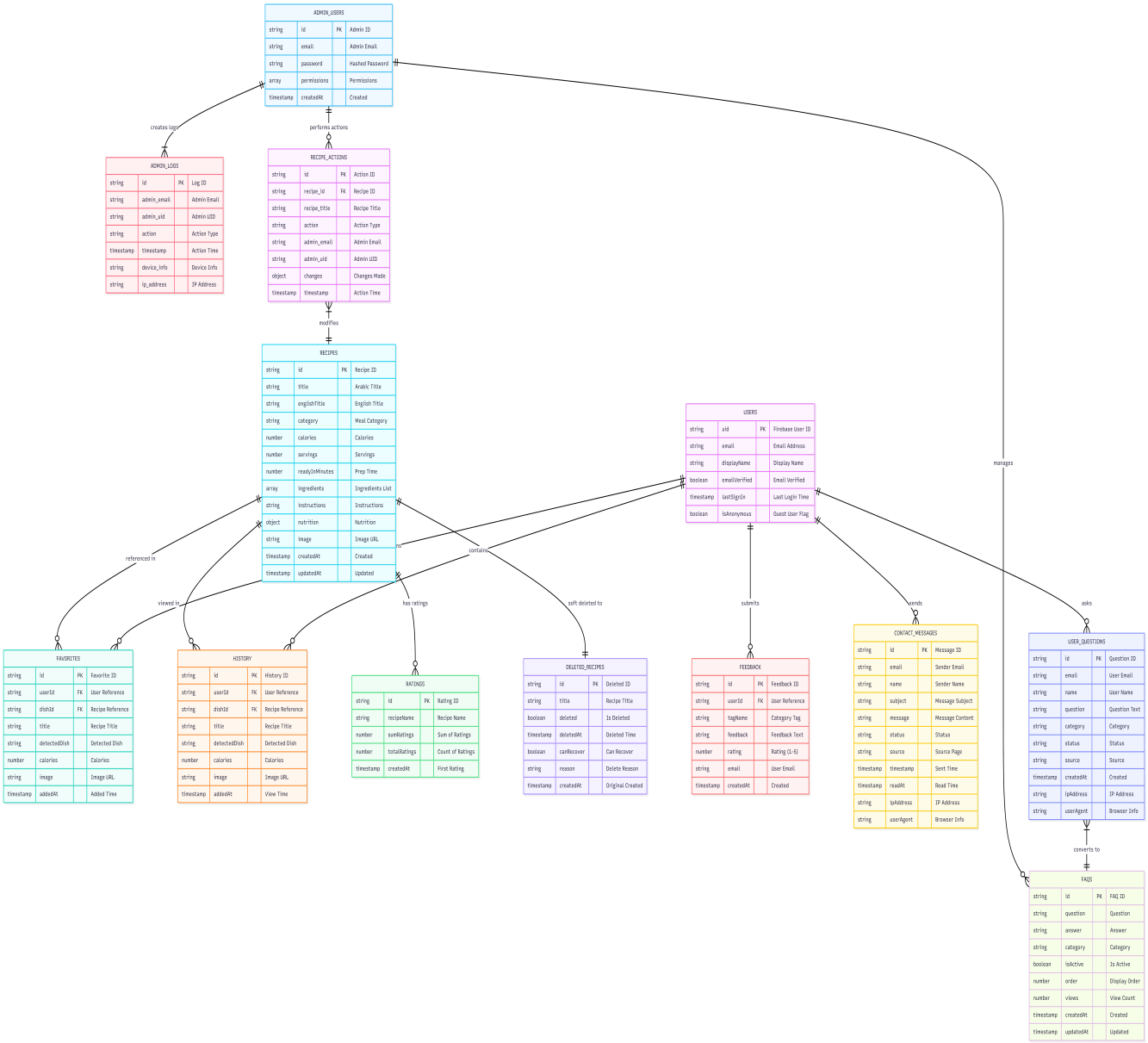

Figure 4.3: Entity-Relationship Diagram of MealLens Database Architecture

The diagram illustrates the complete database architecture with Users as root nodes containing hierarchical subcollections (Favorites and History). Reference connections link user data to Recipes, while aggregation patterns consolidate ratings and feedback. Administrative collections maintain separation for security auditing. Web platform collections extend the architecture for external interactions while maintaining data consistency through shared references. 4.3

4.8 UI/UX Design

As part of our UI/UX design project, we designed several interfaces for the MealLens application to ensure a clean, intuitive, and functional user experience.

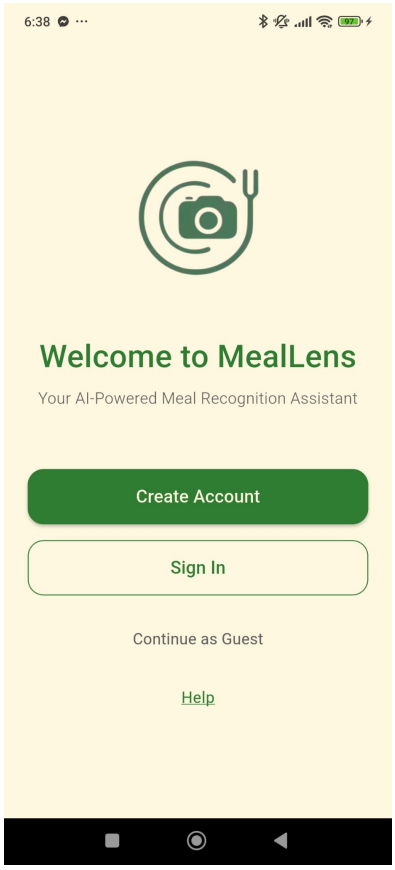

Figure 4.4: Home screen

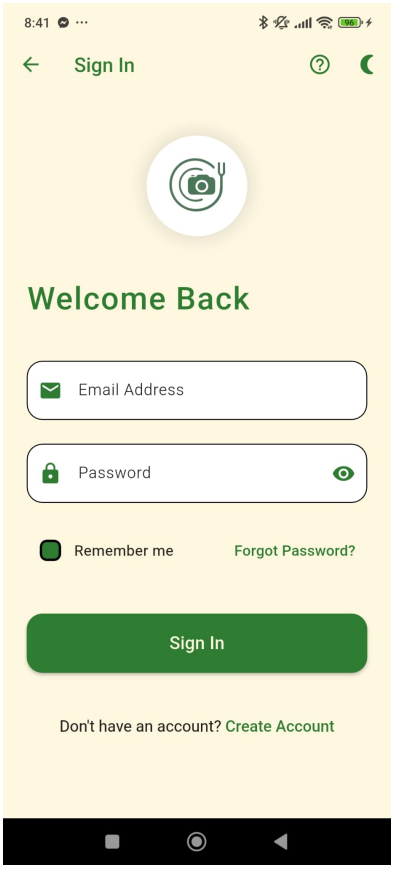

Figure 4.5: Login screen

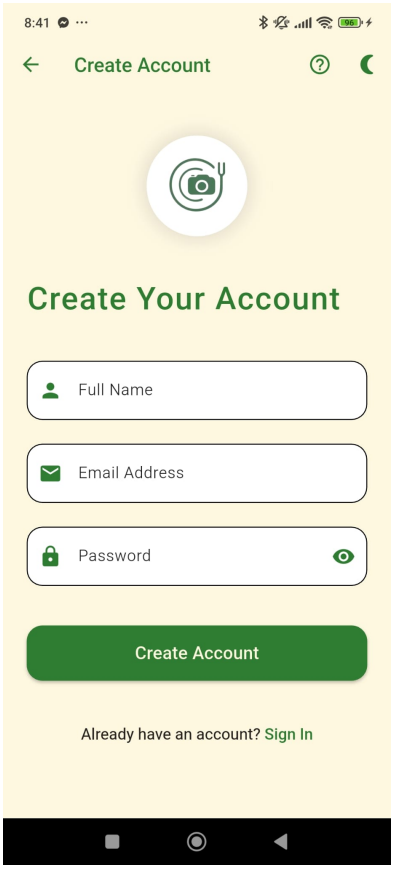

Figure 4.6: Sign-up screen

Figure 4.4 represents the main entry screen for the application. This screen gives the user three main options to choose from. These options are either to sign in using an existing account, to create a new account, or to continue as a guest.

Figure 4.5 shows the Sign In screen, which enables registered users to access their accounts.

Figure 4.6 shows the Sign Up screen, which enables the user to register and create an account within the application.

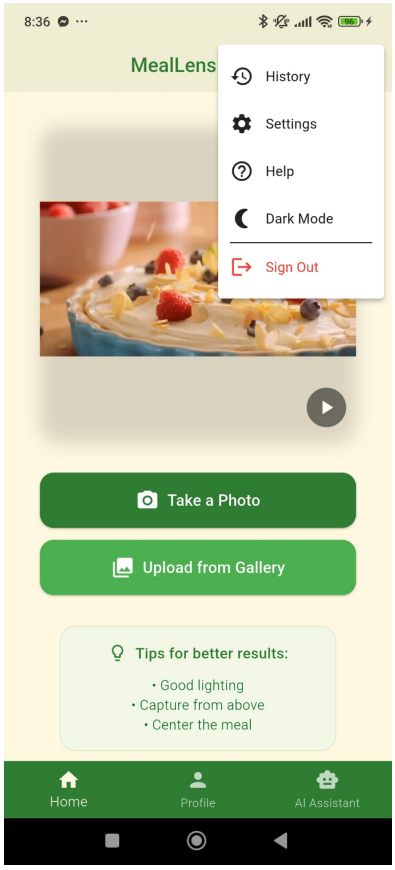

Figure 4.7: App main dashboard screen.

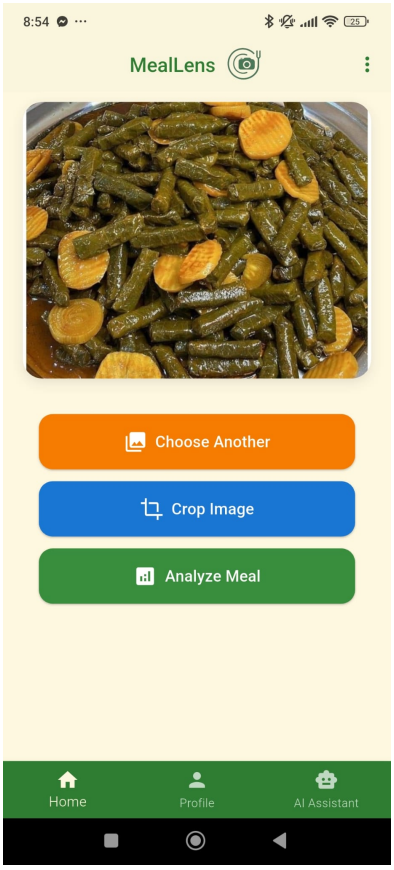

Figure 4.8: Choose photo or analyze screen.

Figure 4.9: Image crop interface.

Figure 4.7 depicts the Main Dashboard Screen, which is the main interface of the application. This interface allows the user to select an image from the gallery or take a new photo using the camera. It also allows the user to access other features such as dark mode, settings, and history.

Figure 4.8 represents the Image Options Screen, which appears after an image is selected or captured. It displays the chosen photo and offers three main actions: changing the image, cropping it, or proceeding with meal analysis

Figure 4.9 illustrates the Crop Screen, where users can adjust the selected image. It enables them to crop the relevant part of the meal and remove unnecessary background elements. This step helps improve the accuracy of the recognition and analysis process.

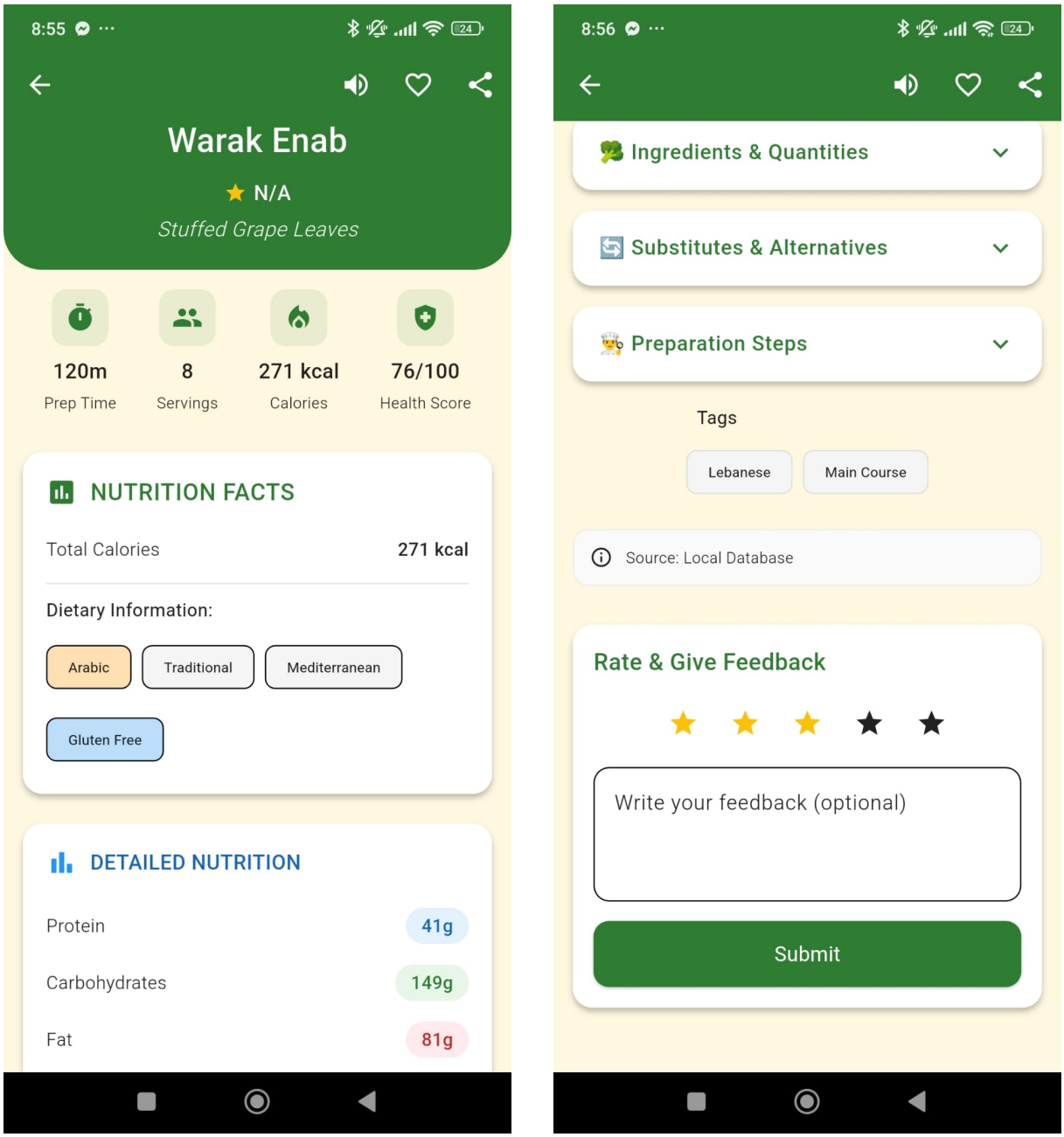

Figure 4.10: MealLens app classification result screens.

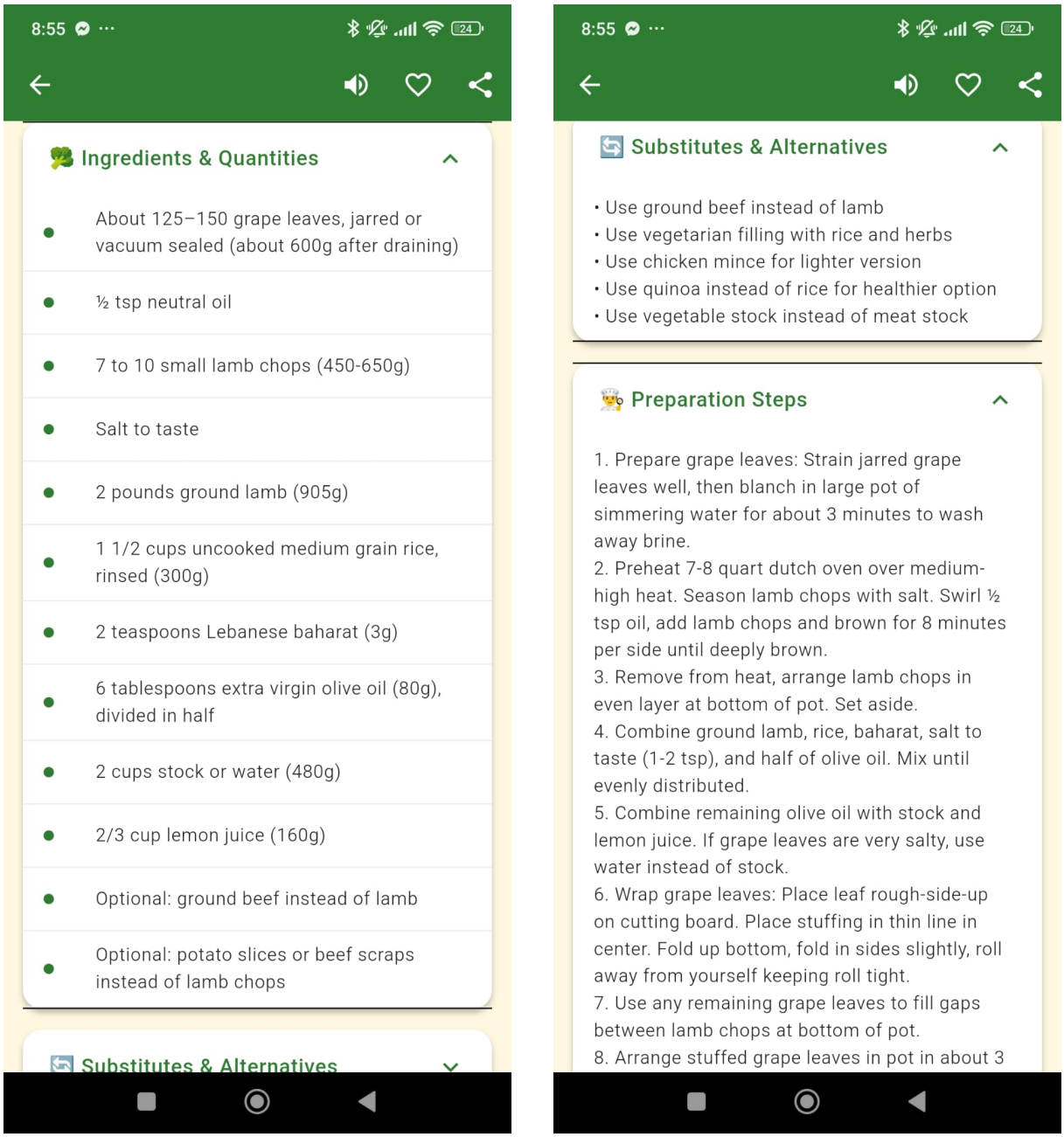

Figure 4.11: Detailed ingredients and preparation steps.

The result screen of the MealLens application displays detailed information about the selected recipe. And it contains the following

- **Recipe Title:** Displays the name of the recipe.
- **Rating:** Shows the recipe rating, e.g., N/A if not yet rated. Star icons provide a visual representation of rating.
- **Prep Time Icon and Text:** Displays the estimated preparation time.
- **Servings Icon and Text:** Indicates the number of servings the recipe yields.
- **Calories Icon and Text:** Shows the total calorie content of the recipe.

- **Share Button:** It enables users to share the recipe information via social media or other apps.
- **Favorite Button:** Users tap this to mark the recipe as favorite so it's saved in the user profile for easy access later.
- **Health Score Icon and Text:** Displays a healthiness score for the recipe, indicating nutritional quality.
- **Nutrition Facts Section:** A card showing the total calories and dietary information tags:
  - **Dietary Tags:** Buttons like Arabic, Traditional, Mediterranean, Gluten Free highlight recipe type and dietary suitability.
  - **Detailed Nutrition Section:** Breaks down macronutrients with values:
    * Protein (e.g., 41g)
    * Carbohydrates (e.g., 149g)
    * Fat (e.g., 81g)
- **Ingredients and Quantities Section:** Expandable/collapsible area showing required ingredients and their amounts.
- **Substitutes and Alternatives Section:** Expandable section providing alternative ingredients for dietary flexibility.
- **Preparation Steps Section:** Expandable section containing step-by-step cooking instructions.
- **Tags:** Shows cuisine type and course category (e.g., Lebanese, Main Course) for filtering or classification purposes.
- **Source Information:** Displays the origin of the recipe, such as "Local Database".
- **Rate and Give Feedback Section:** Allows users to provide feedback and rate the recipe:
  - Star rating (1–5 stars)
  - Optional text feedback input
  - Submit button to send feedback

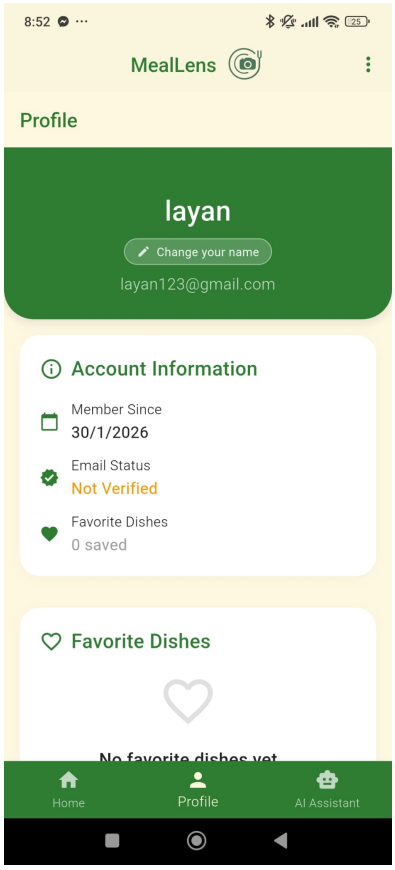

Figure 4.12: Profile screen

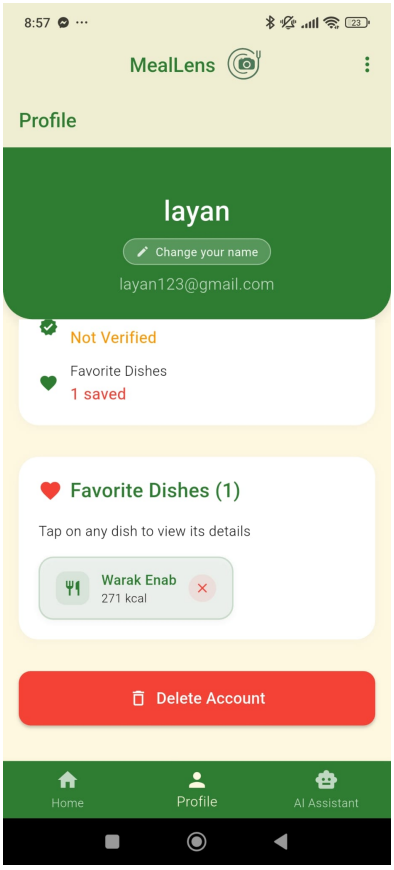

Figure 4.13: Second screen

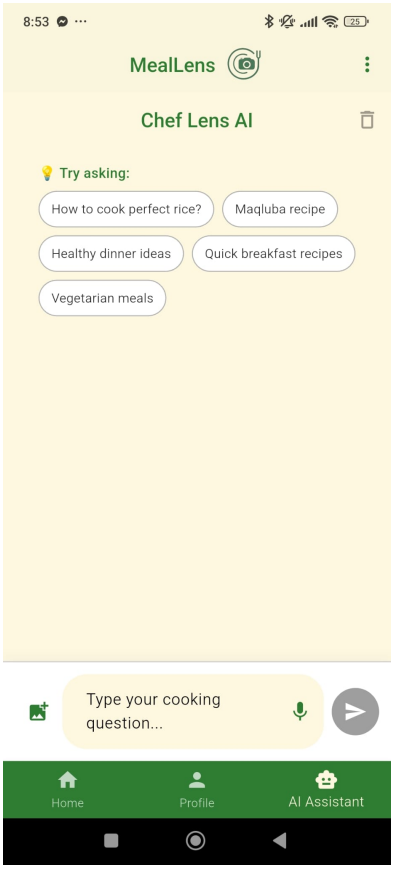

Figure 4.14: AI assistant

Figure 4.12 and figure 4.13 show the profile screen that allows users to manage their personal information and access their saved content. Users can change their display name directly from the profile screen. Additionally, they can view all their saved recipes in a well-organized list.

Figure 4.14 shows The AI assistant screen that allows users to ask questions for more details if they want.

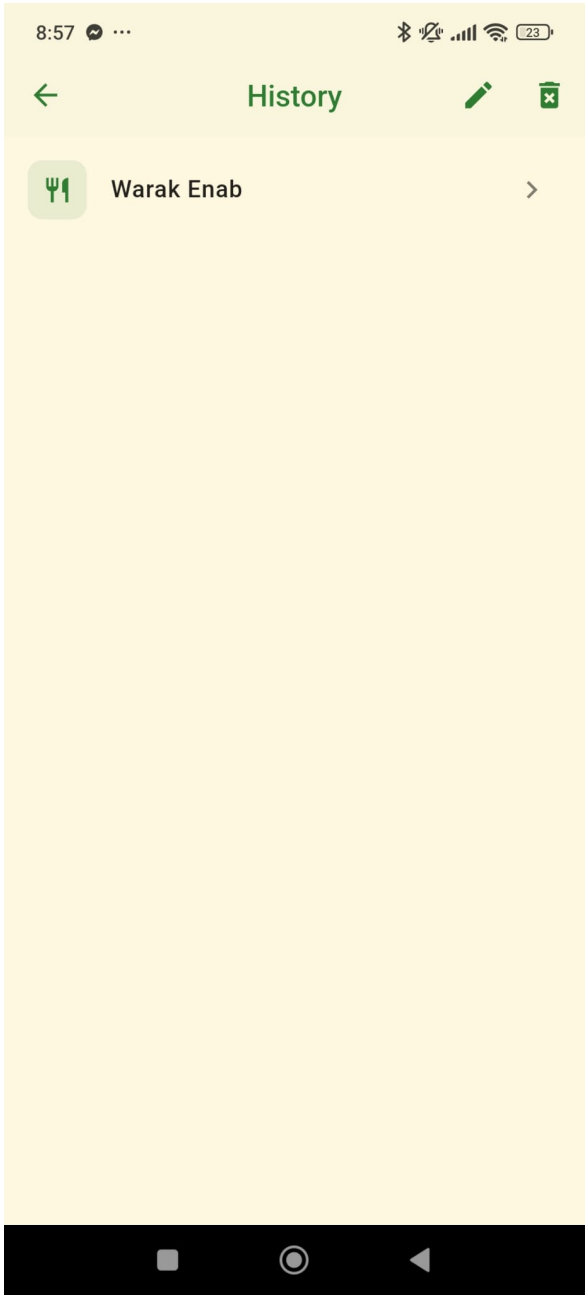

Figure 4.15: User history screen.

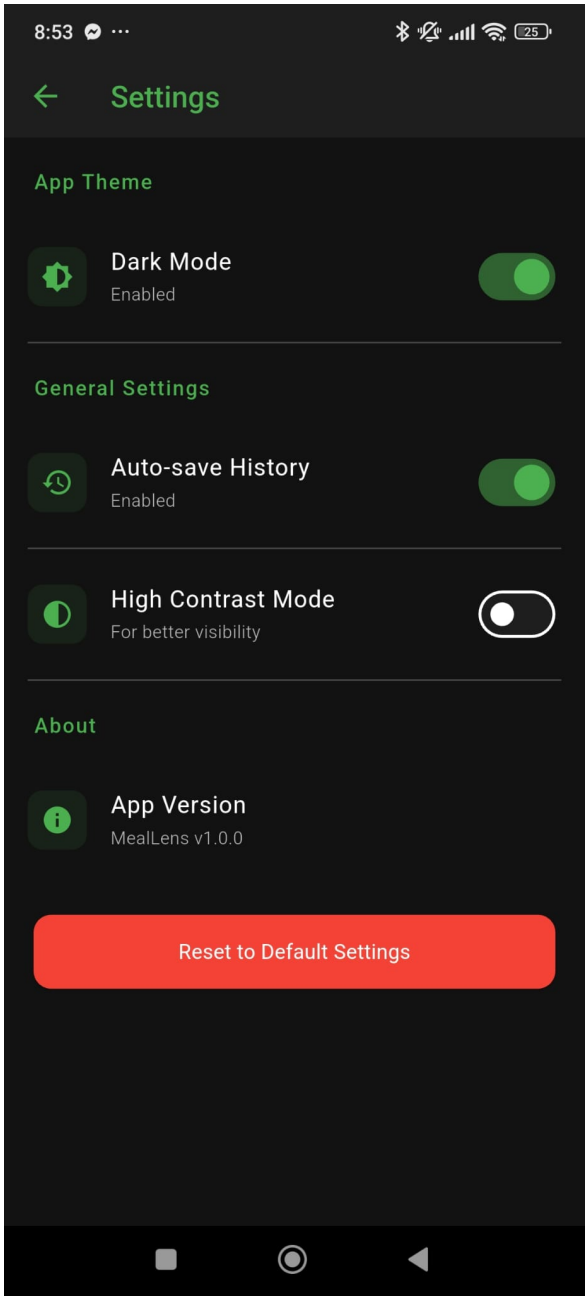

Figure 4.16: Setting screen

Figure 4.15 shows the history screen that shows a list of previously classified recipes, enabling users to quickly revisit past interactions and track their activity within the app.

Figure 4.16 is the settings screen that allows users to customize the app experience. Users can enable dark mode, toggle automatic saving of history, and view the current app version. The settings are designed for easy access and intuitive navigation.

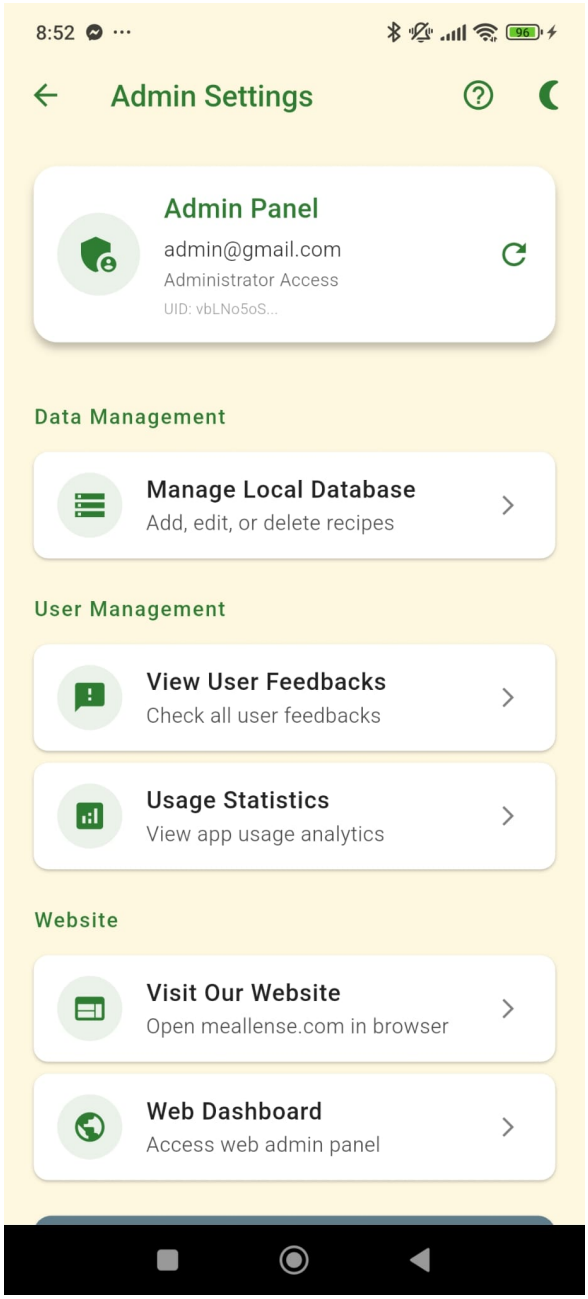

Figure 4.17: Admin screen.

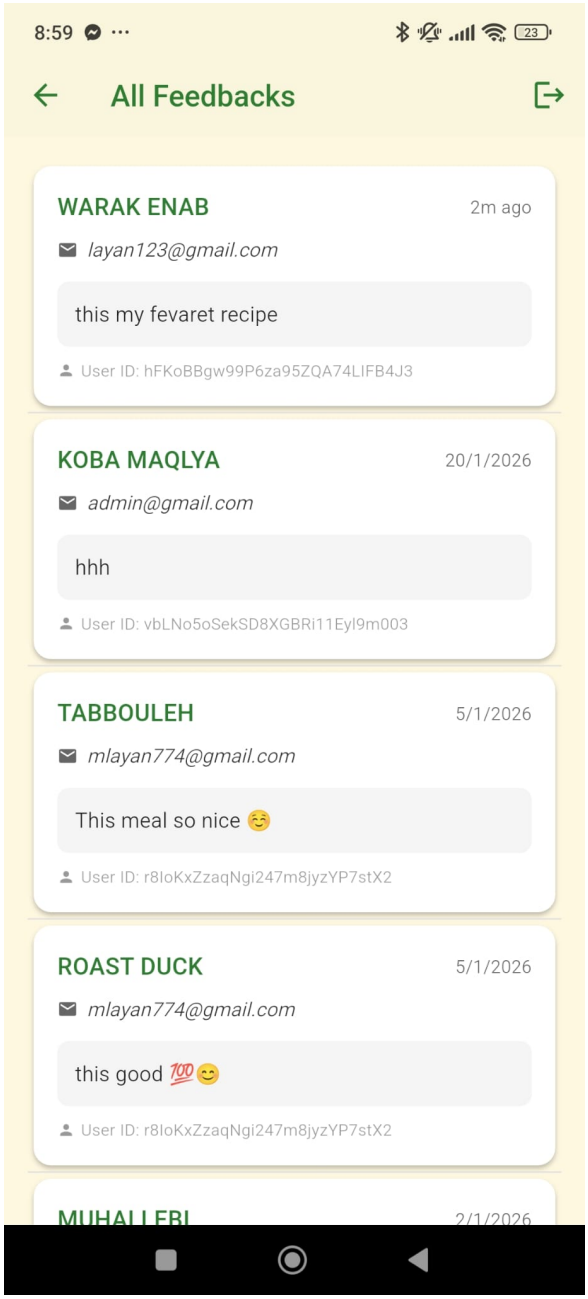

Figure 4.18: Feedback screen

Figure 4.17 is the admin screen that provides administrators with full access to the local database of recipes. Admins can add new recipes, edit existing ones, delete and manage the overall content. The interface also allows viewing user feedback, helping administrators monitor user satisfaction and make improvements.

Figure 4.18 is the feedback interface that displays all feedback submitted by users. Admins can review comments, suggestions, and ratings, providing insights into user preferences and potential areas for app enhancement.

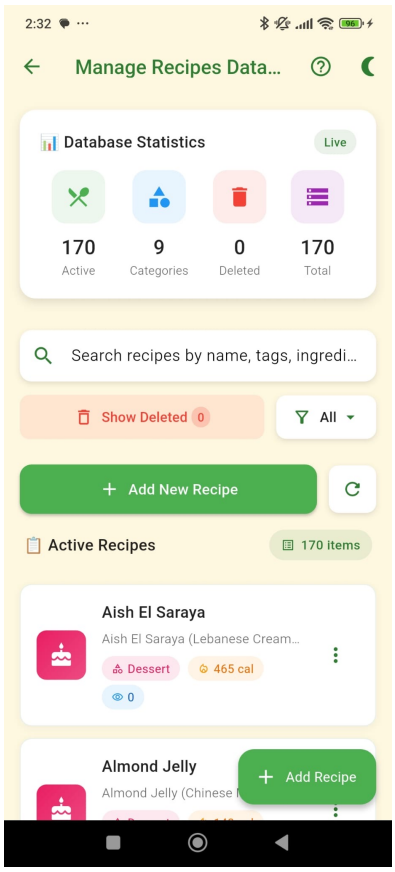

Figure 4.19: Manage data

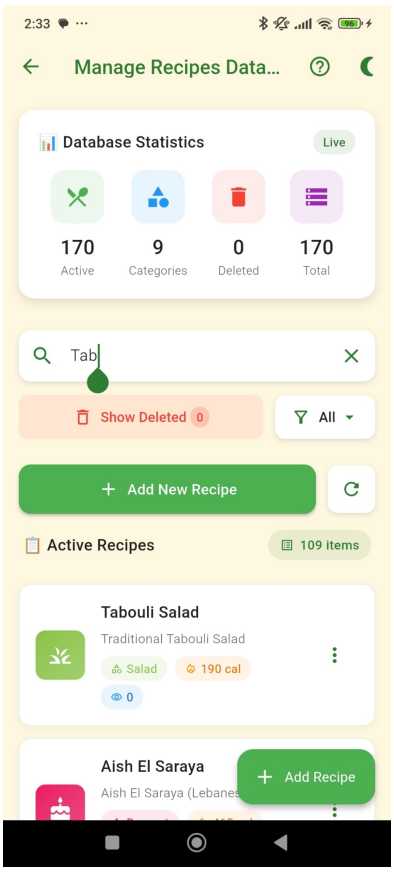

Figure 4.20: Search

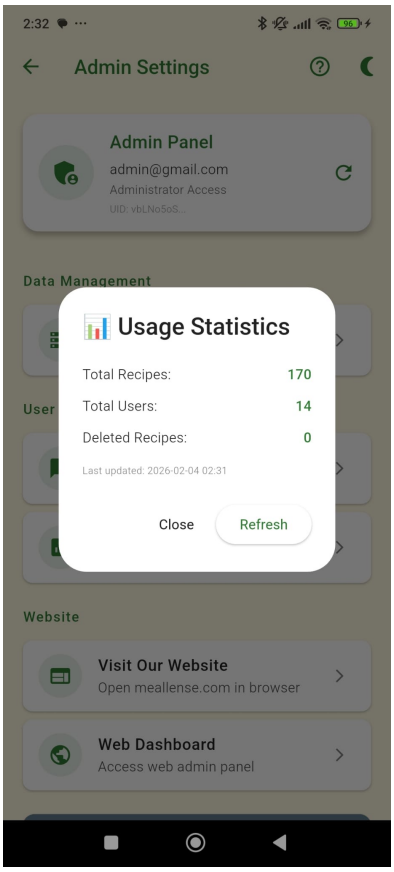

Figure 4.21: Statistics

Figure 4.19 displays the main management screen, where administrators can view database statistics, active recipes, and categories, as well as add new recipes.

Figure 4.20 demonstrates the search functionality, allowing admin to quickly filter recipes by name, making it easier to locate specific items within the database.

Figure 4.21 presents the usage statistics panel, summarizing total recipes, users, and deleted entries, providing administrators with an overview of system activity and data management insights.

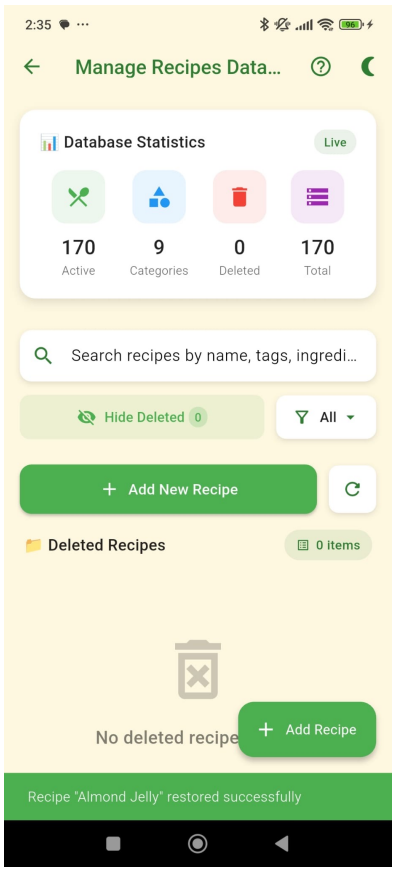

Figure 4.22: Add recipe

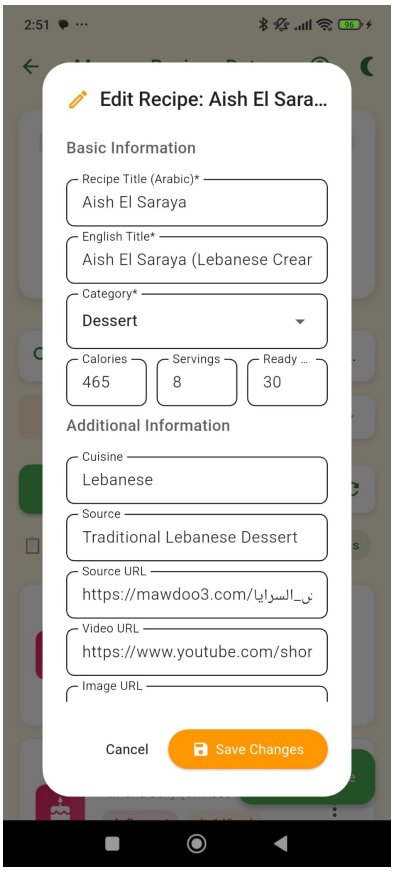

Figure 4.23: Edit recipe

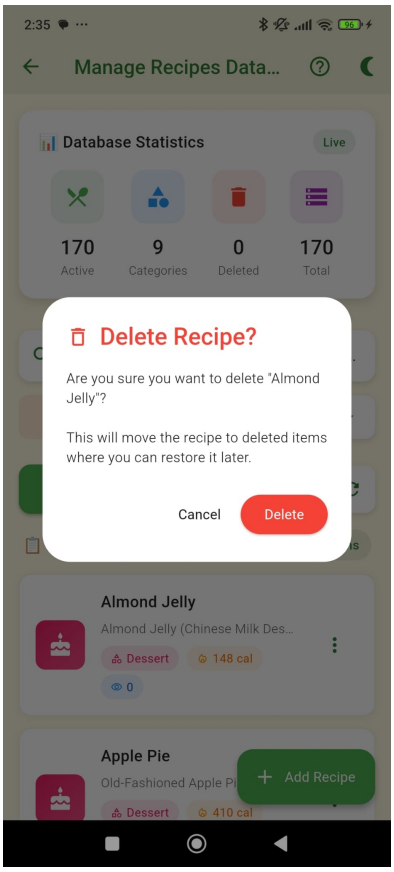

Figure 4.24: Delete recipe

Figure 4.22 presents the interface for adding a new recipe, where administrator can input essential detailsfacilitating structured data entry into the recipe database.

Figure 4.23 illustrates the recipe editing screen, enabling administrator to modify existing recipe information including titles, nutritional values, cuisine type, and source links, ensuring recipe data remains accurate and up-to-date.

Figure 5.21 demonstrates the deletion confirmation dialog, which prompts administrator to confirm removal of a selected recipe while informing them that the item can be restored later from deleted items, providing a safety mechanism against accidental data loss.

Chapter 5

Experimental Results and Analysis

5.1 Introduction

This chapter presents the quantitative evaluation and analysis of the Custom Vision model and the application. It includes the evaluation metrics used, the calculations performed, results in tables and figures, statistical analysis, and a short discussion of the findings and limitations.

5.2 Evaluation Metrics

To evaluate the performance of the proposed meal recognition model, standard classification metrics were adopted. These metrics are widely used in machine learning and computer vision tasks to assess the quality and reliability of prediction models.

5.2.1 Accuracy

Accuracy measures the proportion of correctly classified samples among all predictions made by the model. It is one of the most commonly used evaluation metrics in classification tasks and provides a general indication of model performance.

$$\text{Accuracy} = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. A higher accuracy value indicates a better overall classification performance.

5.2.2 Precision

Precision measures the proportion of correctly predicted positive samples out of all samples predicted as positive by the model.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

A high precision indicates that the model produces few false positive predictions.

5.2.3 Recall

Recall measures the proportion of actual positive samples that were correctly identified by the model.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

A high recall indicates that the model successfully detects most of the relevant samples.

5.2.4 Average Precision (AP)

Average Precision summarizes the trade-off between precision and recall across different classification thresholds. It is defined as the area under the Precision–Recall curve.

$$\text{AP} = \int_0^1 p(r)\, dr$$

AP provides a single-value measure of overall model performance.

5.2.5 Mean Average Precision (mAP)

Mean Average Precision (mAP) is a global evaluation metric used to measure the overall performance of a multi-class classification model. Unlike simple precision, which is calculated at a single confidence threshold, mAP summarizes model performance across all possible thresholds and across all classes. For each class, an Average Precision (AP) value is computed based on the Precision–Recall curve. The mAP value is then obtained by averaging the AP values of all classes.

The mathematical rule for calculating mAP is:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$$

where N is the total number of classes and AP_i is the Average Precision of class i. This metric provides a comprehensive and fair representation of how well the model performs across the entire dataset rather than focusing on a single class or threshold.

5.3 Experimental Setup

5.3.1 Experimental Dataset

To investigate the effect of training dataset size on meal classification performance, we conducted a systematic experiment using fast-food dataset [10],we chose from it five selected meal types: Burger, Pizza, Donut, Fries, and Hot Dog. For each meal, we incrementally increased the number of training images per class, starting with 5 images, then 25 images, then 70 images, and finally 150 images. The model was trained separately on each subset, while evaluation was consistently performed on a fixed test dataset to ensure fair assessment of generalization.

Table 5.1: Model Accuracy vs. Number of Training Images per Dish

Dish Name	5 images	25 images	70 images	150 images
Burger	**37.2%**	**70.45%**	**66.67%**	**76.1%**
Pizza	**37.5%**	**45.24%**	**67.5%**	**84.1%**
Donut	**52.3%**	**78.7%**	**86.5%**	**95.1%**
Fries	**19.5%**	**66.67%**	**80.56%**	**90%**
Hot Dog	**18.6%**	**76.9%**	**80.56%**	**87.5%**

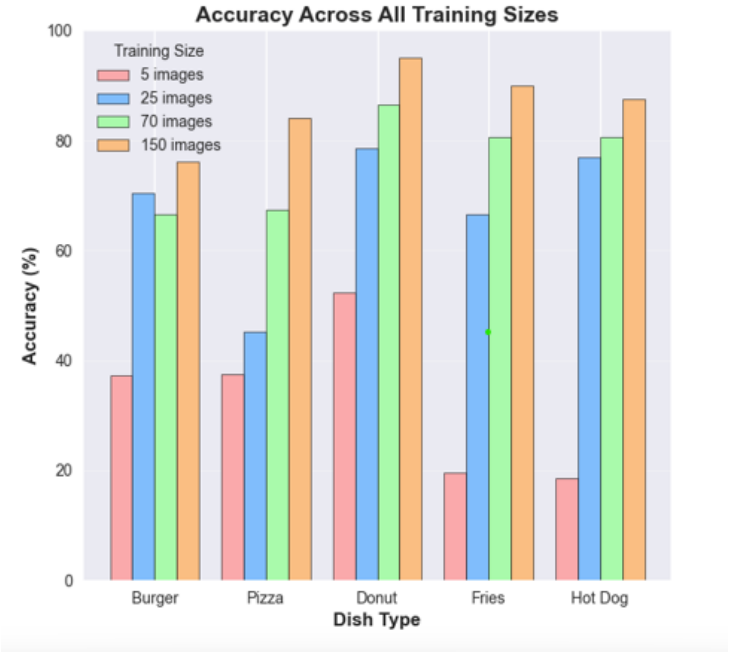

Figure 5.1: A plot representing the Model Accuracy vs. Number of Training Images per Dish

Table 5.1 and Figure 5.1 show that model accuracy consistently improved as the number of training images increased. With only 5 images per dish, the model struggled to generalize, resulting in relatively low accuracy across all meals. Increasing the dataset to 25 images per class significantly improved performance, indicating that the model benefits from additional samples to capture the visual variability of each meal. Optimal performance was generally achieved at 150 images per dish, with accuracies ranging from 76.1% to 95.1%, demonstrating that a sufficiently large training set is crucial for reliable classification.

The results also highlight variability among different meals. For instance, Donut and Fries showed dramatic improvements with increased training images, reflecting the higher visual complexity or intra-class variability of these dishes. Meanwhile, simpler or more distinct meals, such as Burger and Hot Dog, reached high accuracy even with a moderate number of images, suggesting that some dishes are easier for the model to distinguish. Overall, these findings emphasize the importance of collecting at least 150 images per meal type to ensure robust performance in automated meal classification systems.

5.4 Full System Evaluation

This section presents the experimental evaluation of the proposed MealLens system using the metrics described in the previous section. The model was trained using the Microsoft Custom Vision platform on a dataset consisting of 61.906 of meal images distributed across 223 different food categories.

5.4.1 Experimental Setup

The dataset was divided into training and validation sets using Custom Vision's default configuration. Model performance was measured on the validation set using precision, recall, and average precision.

5.4.2 Results and Discussion

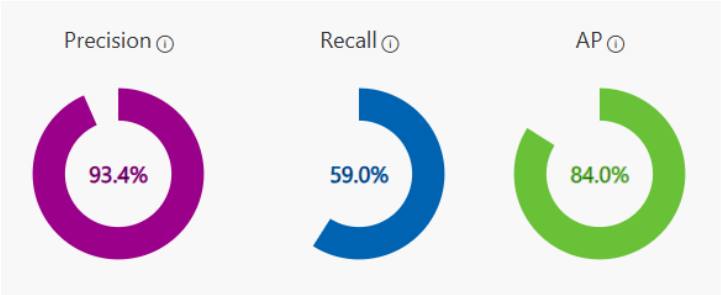

Figure 5.2: Image representing the evaluation metrics of the model

- **Precision: 93.4%** The high precision indicates that when the system predicts a meal, the result is correct in most cases, with very few false positives.
- **Recall: 59.0%** The recall score shows that the system correctly identifies around two-thirds of the actual meal images. Some meals are missed, particularly visually similar dishes.
- **Average Precision (AP): 84.0%** The AP score reflects strong overall performance across all confidence thresholds, demonstrating the robustness of the model despite the large number of classes.
- **Mean Average Precision(mAP): 82.00%** This mAP result indicates that the overall performance of the system classification model is strong. That value confirms that the system is capable of reliable multi-class food recognition despite the large number of distinct classes.

5.5 User Testing

As part of the development cycle, we conducted comprehensive user testing for the MealLens app to evaluate usability, functionality, and error handling across various scenarios. The testing covered authentication flows, meal analysis, and edge cases to ensure a robust user experience.

5.5.1 Test User Credentials and Access Validation

This subsection describes the test credentials and procedures used to verify that users can successfully authenticate and access the system during the usability and functional testing phase. The purpose of this test is to ensure that valid users are able to register, log in, and interact with the application without errors.

Login with Correct Credentials

This test case verifies that a registered user can successfully log into the application using valid credentials and gain access to the system.

Objective: To confirm that the system authenticates users correctly when valid login information is provided.

Test Steps:

1. Log out from the current session.
2. Enter the registered email address.
3. Enter the corresponding password.
4. Press the *Sign In* button

Expected Result: The login is completed successfully, and the user is redirected to the main dashboard where the user profile is loaded and accessible.

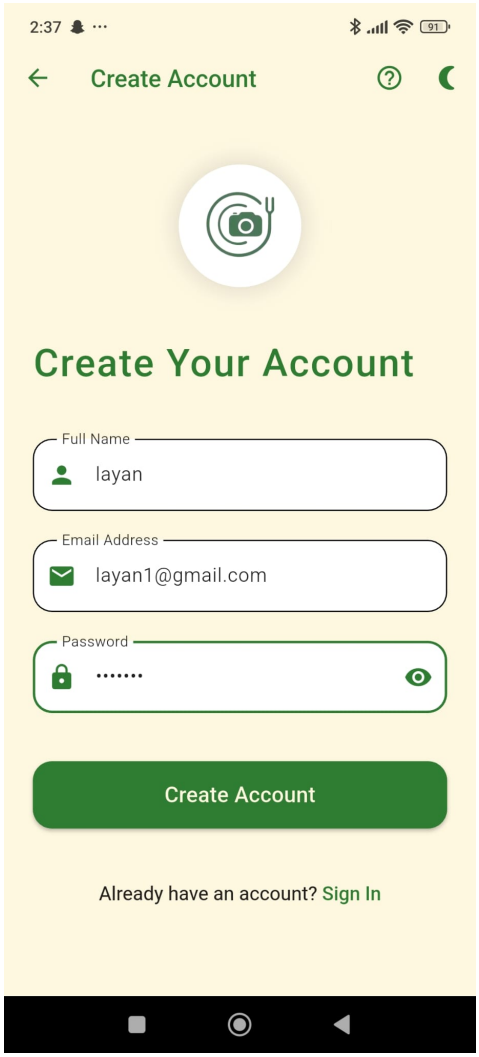

Figure 5.3: Sign up screen

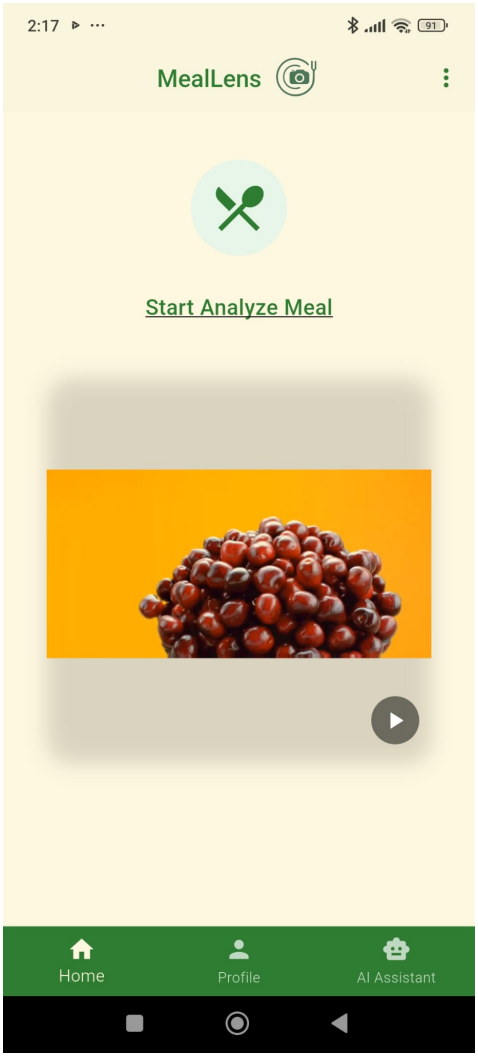

Figure 5.4: Home screen

Login with Wrong Password

This test case evaluates the system's ability to handle authentication errors when an incorrect password is entered.

Objective: To verify that the system displays an appropriate error message when a user attempts to log in using an incorrect password.

Test Steps:

1. Enter a valid registered email address.
2. Enter an incorrect password.
3. Press the *Sign In* button (see Figure 5.5).
4. Select the *Reset Password* option to recover access if needed.

Expected Result: The system denies access and displays the error message: *"Incorrect password. Try again or reset."*

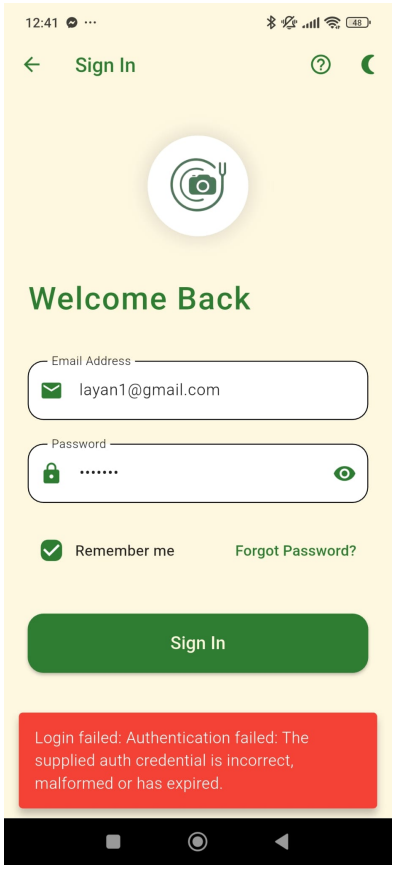

Figure 5.5: enter wrong password

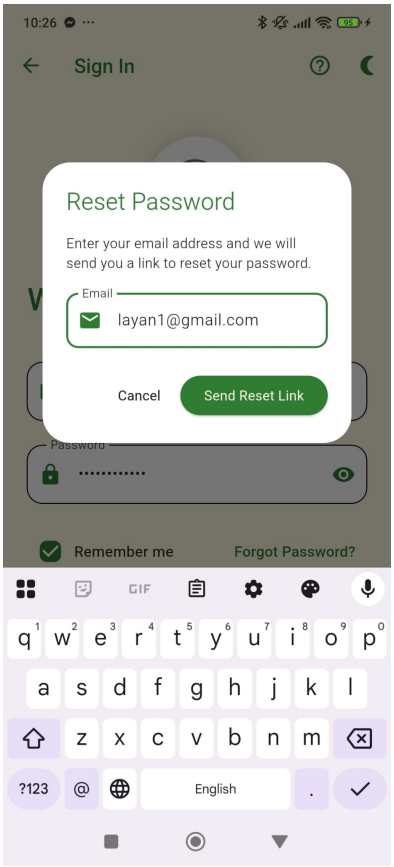

Figure 5.6: reset password

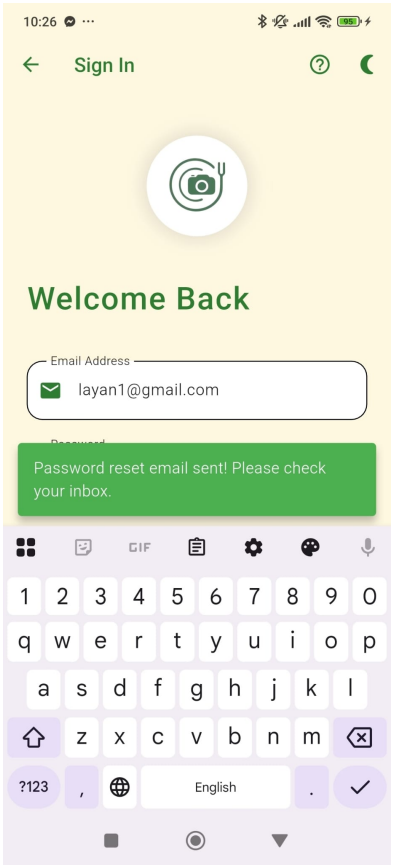

Figure 5.7: password reset email sent

5.5.2 Image Cropping and Editing

This test case evaluates the effectiveness of the image pre-processing tools provided by the application, which are required to enhance image quality prior to meal recognition.

Objective: To verify that the cropping and rotation tools allow users to refine selected food images accurately for better recognition results.

Test Steps:

1. After selecting an image (see Figure 5.8), tap the *Crop* option.
2. Adjust the crop area and apply rotation as needed.
3. Confirm the edited image.

Expected Result: The cropping tool (Figure 5.9) and rotation tool (Figure 5.10) operate correctly, allowing precise refinement of the image. The edited image is successfully processed by the image recognition system (Figure 5.11), confirming the effectiveness of the pre-processing stage.

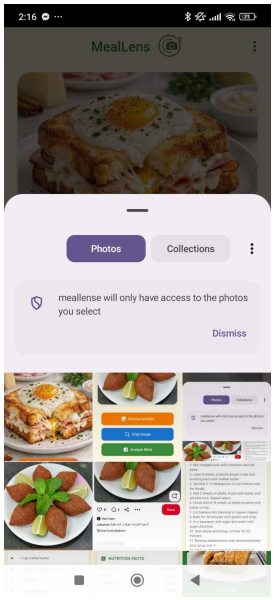

Figure 5.8: uploade Photo from gallery

Figure 5.9: crope image

Figure 5.10: rotation image by 90 degree

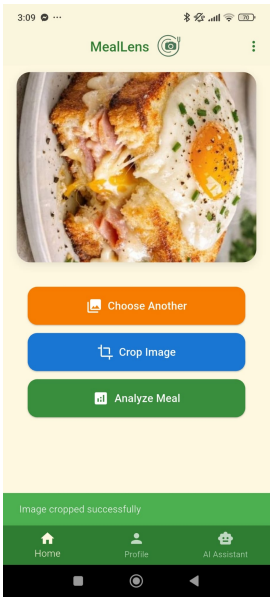

Figure 5.11: conform edit image

5.5.3 Continue as Guest Mode

- **Objective:** To test the Guest Mode functionality and evaluate the complete image analysis workflow for recognizing a traditional Tabbouleh dish.
- **Test Scenario:** A user accesses the application without creating an account, captures a photo of a Tabbouleh dish, and receives a complete meal analysis including recipe, ingredients, and preparation instructions.
- **Steps:**
  1. From the application home screen, select the *Continue as Guest* option.
  2. From the main dashboard in guest mode, select the *Take Photo* option Figure 5.12.
  3. Capture a photo of a prepared Tabbouleh dish using the device's camera Figure 5.13.
  4. Review the captured image in the preview screen and confirm to proceed with analysis Figure 5.14.
  5. The application sends the image to the Azure Custom Vision model for recognition and retrieves recipe data from external APIs.
- **Results:** The test was successful. The application performed as expected in Guest Mode:
  - **Meal Identification:** Correctly identified the dish as **Tabbouleh**.
  - **Core Features Access:** Full access to meal recognition, recipe viewing, ingredient lists, and cooking instructions was granted.
  - **Detailed Information Provided:** The application displayed:

* Complete recipe with title and description
* Detailed ingredients list with quantities Figure 5.16
* Step-by-step preparation instructions Figure 5.17
* Recipe source and additional metadata Figure 5.18

– **Guest Mode Restrictions:** As expected in Guest Mode:

* **Disabled:** Saving recipes to favorites
* **Disabled:** Accessing personal history
* **Disabled:** Profile customization features
* **Disabled:** Feedback
* **Enabled:** All core recognition and recipe viewing features

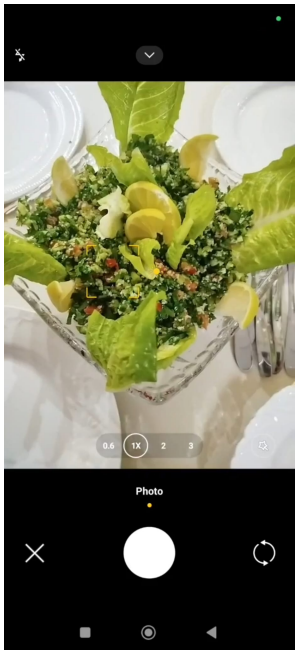

Figure 5.12: Take Photo Screen in Guest Mode

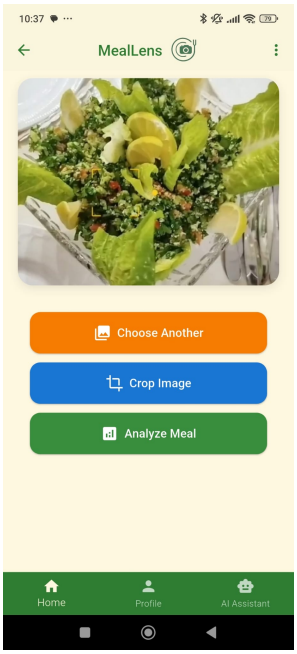

Figure 5.13: Image Preview Before Analysis

Figure 5.14: Initial Recognition Result

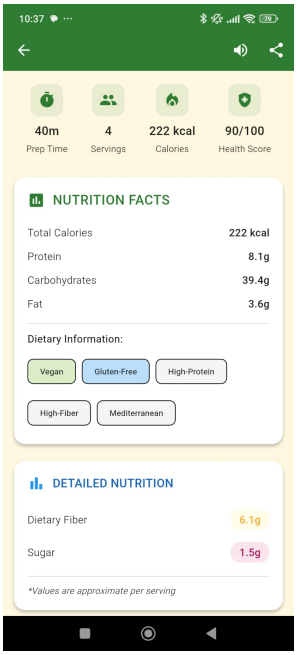

Figure 5.15: Recipe Overview Display

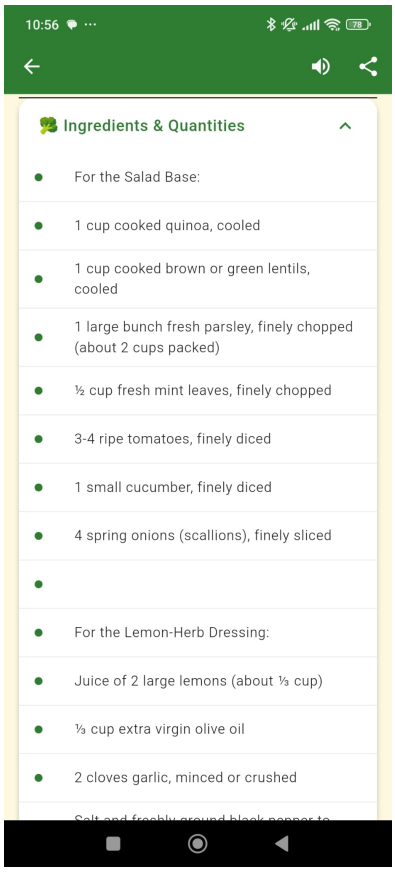

Figure 5.16: Ingredients and Quantities Section

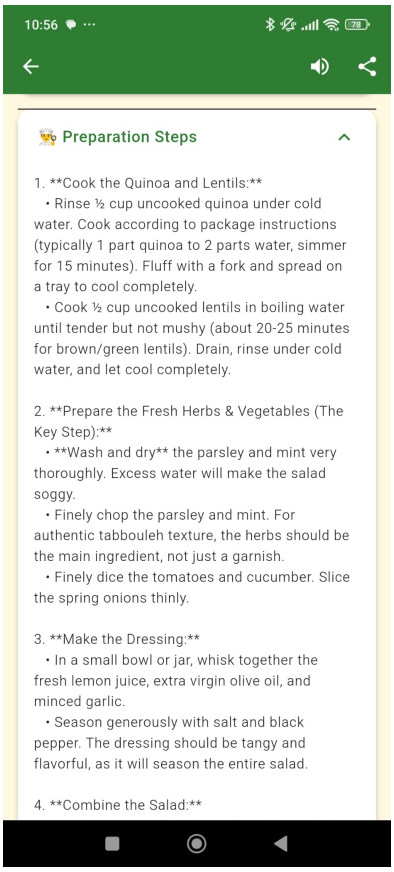

Figure 5.17: Step-by-Step Cooking Instructions

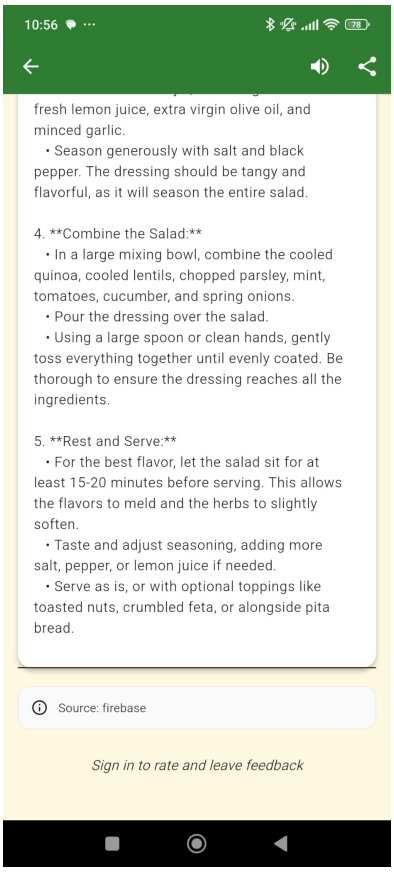

Figure 5.18: Recipe Source and Metadata

The Guest Mode functionality has proving to be effective in providing instant access to the fundamental value proposition set of the MealLens application, namely the AI-based meal detection and recipe assistance. Although the personalization aspects remain restricted, the overall analysis process appears to work smoothly, ensuring the user experiences the power of the application, increasing the incentive to create an account to access the features. The test is effective in validating that the application design correctly manages anonymous user sessions and data privacy/security boundaries.

5.5.4 History Management and Deletion

- **Objective:** To test the history tracking feature and verify that users can successfully view and delete items from their meal recognition history.
- **Steps:**
  1. From the main interface, navigate to the History section.
  2. Browse the list of previously recognized meals.
  3. Enter the selection mode (e.g., by tapping an edit or select icon).
  4. Select one or more items (e.g., "Croque Madame") for deletion.
  5. Tap the delete button and confirm the action when prompted.

- **Result:** The process was successful. The application correctly performed the following actions:
  - Displayed the user's history with meals including Croque Madame and Apple Dumplings In Fanta Sauce 5.19.
  - Allowed individual selection of meal items 5.20.
  - Successfully deleted the selected "Croque Madame" item and displayed a confirmation message: Deleted 1 item.
  - Updated the history list to reflect the deletion 5.21

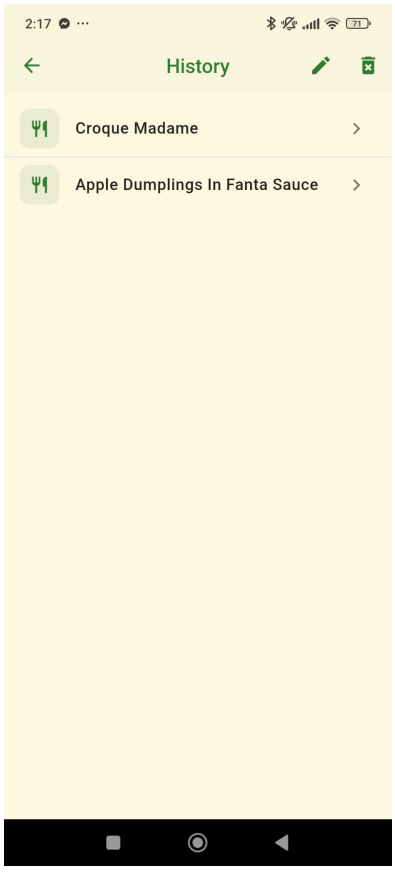

Figure 5.19: User's history with meals

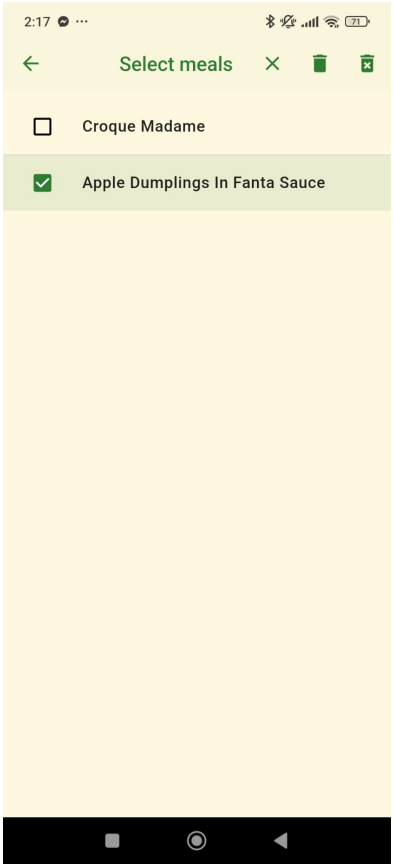

Figure 5.20: Allowed individual selection of meal items

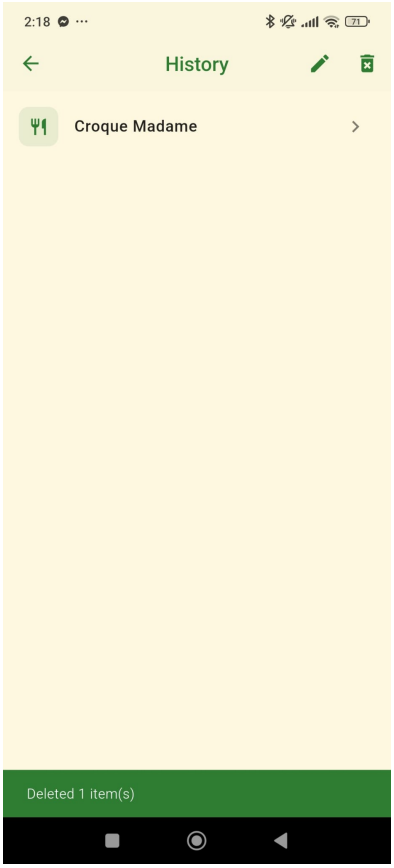

Figure 5.21: Successfully deleted the selected "Croque Madame" item

5.5.5 Complete Application Demo

A full end-to-end demonstration of the *MealLens* application was conducted, showcasing its core workflows and key features.

Complete Demo Video: The full demonstration of the application is available at the following link.[1]

[1]Demo video available at: `https://drive.google.com/file/d/1kSVt2u5JZZMaCr7x-OKB7LxImC74GIKB/view?usp=drivesdk`

Chapter 6

Web Platform Development for MealLense

6.1 Introduction and Overview

A web platform was developed as an extension to the MealLense mobile app, serving as a complementary interface for user support and administrative management. This system operates independently but connects to the main Firebase backend, creating a holistic ecosystem where mobile handles food recognition while the web manages content and user communication. The platform targets two groups: users needing support or information, and administrators requiring content management tools. It is deployed on Render at MealLence Link[1] and integrated into the mobile app's help section.

6.2 Platform Architecture

6.2.1 Firebase Database Structure

The platform uses four Firestore collections optimized for web operations. Complete schema details are in **Appendix 19**.

Collection Overview

The platform employs four specialized collections: `website_admins` for administrator authentication, `website_faq` for bilingual knowledge base management, `website_messages` for user communication storage, and `user_questions` for unanswered inquiry tracking. Each collection is designed with specific field structures documented in the appendix tables.

6.3 Website Functionality

6.3.1 Homepage with Contact Form

The homepage provides the main user entry point with an integrated contact system.Figure 6.1

[1] `https://meallense-website-help.onrender.com/index.html`

6.3.2 FAQ System with Search

The FAQ interface offers comprehensive knowledge discovery through full-text search with relevance ranking and category filtering. It supports both Arabic and English content, tracks view counts for popularity insights, and seamlessly transitions to question submission when answers aren't found 6.2. Figure 6.2

6.3.3 Admin Dashboard

The administrative dashboard provides centralized control through management modules. The message manager handles incoming communications with filtering and status tracking, while the FAQ editor manages add , delete and edit any FAQ in firebase.Figure 6.3

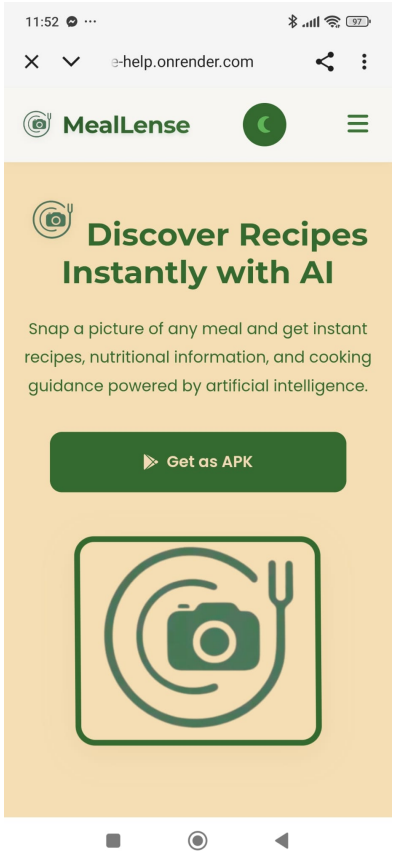

Figure 6.1: Homepage Interface

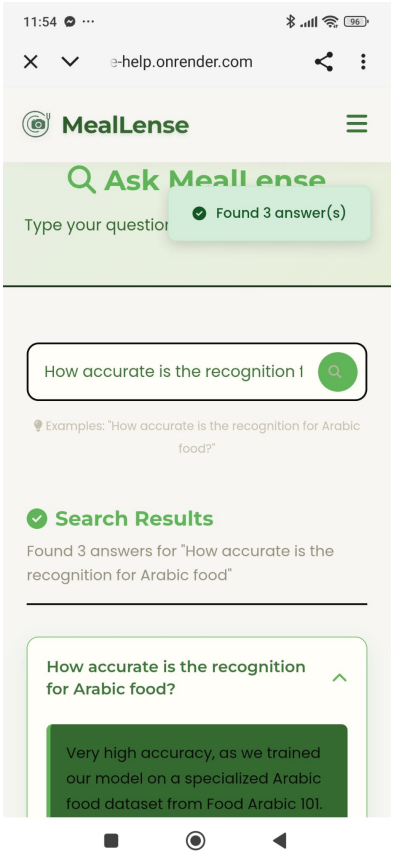

Figure 6.2: FAQ Search System

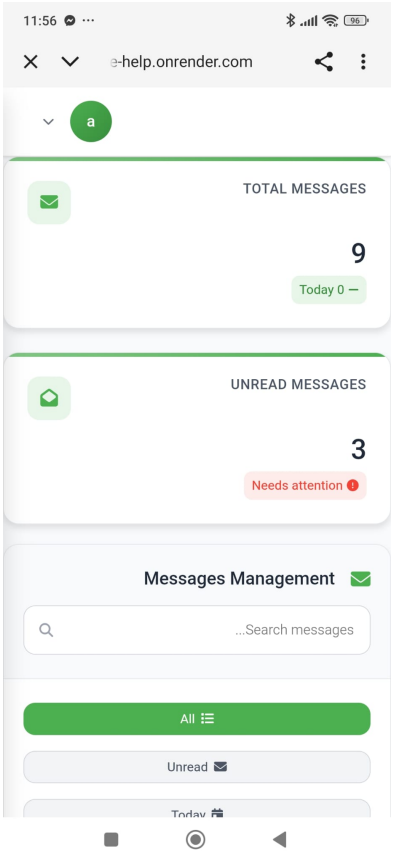

Figure 6.3: Administrative Dashboard

6.4 Mobile App Integration

6.4.1 Cross-Platform Experience

The mobile application integrates the web platform through a direct access link in its help section. Users can transition seamlessly to web-based support without authentication requirements, maintaining consistent information presentation between both platforms while utilizing each interface's strengths.

6.4.2 Support Workflow

The integrated support system follows a streamlined workflow: users initiate from the mobile app help section, access the website via the embedded link, attempt self-service through FAQ search, submit contact forms when needed, receive administrative review and email response, with common issues feeding back into application improvements. This creates a closed-loop system that enhances both platforms.

6.5 Deployment

6.5.1 Render Hosting

The platform is deployed on Render cloud hosting with automatic SSL certification through Let's Encrypt. It utilizes CDN distribution for global performance, automated build processes on repository updates, and secure environment variable management for Firebase credentials configuration.

6.5.2 Security Rules

Firebase security rules implement balanced access control: FAQ content has public read access for user accessibility, message and question submissions require validation without full authentication, administrative collections restrict access to verified credentials, with additional data validation and rate limiting measures to prevent system abuse.

6.6 Workflows

6.6.1 Message Processing

The message handling workflow begins with user form submission and metadata capture, proceeds through admin notification and priority-based triage, continues with direct email response using collected contact information, concludes with status resolution updates, and includes periodic pattern analysis to identify common user challenges for systemic improvements.

6.6.2 FAQ Maintenance

Knowledge base maintenance follows a structured process starting with gap identification through user interactions or administrative observation. It progresses through research and validation of accurate information, bilingual content creation with clear organization, editorial review for quality assurance, publication with proper categorization, ongoing analytics monitoring for effectiveness tracking, and regular content updates to maintain relevance and accuracy 6.2.

Chapter 7

Conclusion and Future Work

Contents

7.1 Conclusion 61

7.2 Future Work 61

7.1 Conclusion

The MealLens project has developed an AI-based meal recognition and recipe assistant that bridges seamlessly between food discovery and practical meal preparation. Computer vision, cloud services, and APIs are integrated to accurately classify 223 different classes of meals through user-uploaded images. Using a well-designed multi-tiered architecture with Flutter, Firebase, and Azure Custom Vision, the application instantly delivers recipes, nutritional information, cooking instructions and others to users through an intuitive mobile interface.

Some key achievements are a high-quality dataset containing 61,906 training images of different cuisines, applying intelligent fallback mechanisms to make API utilization cost-effective, and both mobile and web platforms that have features to meet different user requirements. The system will solve the identified problem of inefficient food identification and recipe searching by offering image-based recognition, which is more natural and accessible than traditional text-based methods.

7.2 Future Work

Future improvments for our MealLens application could be adding more different meals from all world cuisines which will fit more users. In addition, adding the feature of multi language support, incorporating real-time meal recognition for video feeds, adding personalized recommendations based on dietary preferences or allergies. Additionally, integrating advanced nutrition analysis and meal planning could further enrich the user experience.

Bibliography

[1] M. Benjamin, "Sl food image dataset," https://www.kaggle.com/datasets/manojbenjamin/sl-food-image-dataset, 2024, accessed: 2024-12-15. [Online]. Available: https://www.kaggle.com/datasets/manojbenjamin/sl-food-image-dataset

[2] G. Duttakiit, "Food image classification using resnet50," https://www.kaggle.com/code/gauravduttakiit/food-image-classification-resnet50/input, 2024, accessed: 2024-12-15. [Online]. Available: https://www.kaggle.com/code/gauravduttakiit/food-image-classification-resnet50/input

[3] KRTNP, "Chefmate: Smart kitchen assistant with ai," https://github.com/KRTNP/ChefMate-Smart-Kitchen-Assistant-with-AI, 2025, accessed: 2025-06-16. [Online]. Available: https://github.com/KRTNP/ChefMate-Smart-Kitchen-Assistant-with-AI

[4] Google, "Ingregenius | Gemini API Developer Competition | Google AI for Developers," n.d., accessed July 4, 2025. [Online]. Available: https://ai.google.dev/competition/projects/ingregenius

[5] A.-T. Arar, "Arabic food 101," https://www.kaggle.com/datasets/araraltawil/arabic-food-101, 2023, accessed: 2025-06-16.

[6] L. Boyd, "Multi-class food image dataset," https://www.kaggle.com/datasets/liamboyd1/multi-class-food-image-dataset, 2025, accessed: 2025-12-08.

[7] r. Kuo, "Uecfood256," https://www.kaggle.com/datasets/rkuo2000/uecfood256, 2014, accessed: 2025-12-08.

[8] Ishikanaik, "Cakes dataset," https://www.kaggle.com/datasets/ishikanaik/cakes-dataset, 2025, accessed: 2025-12-08.

[9] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – mining discriminative components with random forests," in *European Conference on Computer Vision*, 2014.

[10] U. Saxena, "Fast food classification dataset," https://www.kaggle.com/datasets/utkarshsaxenadn/fast-food-classification-dataset, 2023, accessed: 2026.

[11] Spoonacular Team, "Spoonacular food api," https://spoonacular.com/food-api, 2024, accessed: 2025-06-16.

[12] Edamam, "Edamam food and recipe api," https://www.edamam.com/, 2024, accessed: 2025-06-16. [Online]. Available: https://www.edamam.com/

[13] Yummly, "Yummly api documentation," https://developer.yummly.com/, 2024, accessed: 2025-06-16. [Online]. Available: https://developer.yummly.com/

[14] TheMealDB, "Themealdb: Open recipe database," https://www.themealdb.com/api.php, 2024, accessed: 2025-06-16. [Online]. Available: https://www.themealdb.com/api.php

[15] "Edamam nutrition analysis api," https://developer.edamam.com/edamam-nutrition-api, 2026, accessed: 2026.

[16] TheMealDB Team, "Themealdb," https://www.themealdb.com/, 2024, accessed: 2025-06-16.

[17] Figma Help Center, "What is figma?" 2025, accessed: 2025-06-16. [Online]. Available: https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma

[18] Wikipedia contributors, "Flutter (software)," 2025, accessed: 2025-06-16. [Online]. Available: https://en.wikipedia.org/wiki/Flutter_(software)

[19] Microsoft, "What is custom vision?" Microsoft Learn documentation, 2025, https://learn.microsoft.com/azure/ai-services/custom-vision-service/overview.

[20] "Spoonacular food api pricing," https://spoonacular.com/food-api/pricing, 2026, accessed: 2026.

[21] "Themealdb — free recipe api and database," https://www.themealdb.com/, 2026, accessed: 2026.

Chapter 8

Appendix

Contents

.1 Complete List of Meals 69

.1 Database Schemas 75

Table 1: Selected categories from Arabic food 101 dataset

No.	Selected Meal	Number of Images
1	Apple pie	176
2	Falafel	173
3	Kibbeh	175
4	Kunafa	183
5	Mahashi	157
6	Mandi	172
7	Mansaf	165
8	Qatayef	177
9	Rice with milk	153
10	Shawarma	175
11	Shishabark	151
12	Kebab	184
13	Mojadra	174
14	Pastries	176
15	Samosas	177
16	Fava Beans	111
17	Makmoura	136
18	Koshari	183
19	Musakhan	162

Table 2: Selected categories from the Multi-Class Food Image Dataset with Image Counts

No.	Selected Meal	Number of Images
1	Spinach	170
2	Chicken	147
3	Fish	134

Table 3: Selected categories from UEC FOOD 256 Dataset

No.	Selected Meal	Number of Images
1	Mushroom risotto	114
2	French toast	102
3	Chicken nugget	102
4	French bread	102
5	Bagel	104
6	Tortilla	102
7	Nachos	104
8	Scrambled egg	101
9	Muffin	112
10	Popcorn	116
11	Sushi	151
12	Rice	172
13	Beef curry	155
14	Fried rice	123
15	Toast	165
16	Croissant	118
17	Ramen noodles	278
18	Spaghetti	148
19	Omelet	160
20	Sausage	118
21	Grilled salmon	115
22	Steak	108
23	Boiled chicken and vegetables	105
24	Shrimp	118
25	Steamed meat dumplings	114
26	Fried shrimp	114
27	Potato salad	127
28	Green salad	249
29	Macaroni salad	104
30	Chinese soup	160
31	Beef bowl	162
32	Rice ball	101
33	Mango pudding	103
34	Almond jelly	103
35	Crepe	101
36	Pancake	105
37	Waffle	167
38	Rare cheesecake	112
39	Parfait	100
40	Roast duck	116
41	Hot pot	107
42	Moon cake	104
43	Custard tart	100

Continued on next page

No.	Selected Meal	Number of Images
44	Lemon fig jelly	108
45	Spicy chicken salad	102
46	Fried mussel pancakes	102
47	Deep fried chicken wing	110
48	Coconut milk soup	111
49	Fried spring rolls	104
50	Steamed rice roll	104
51	Brownie	108
52	Eggplant with garlic sauce	113
53	Winter melon soup	118
54	Chinese pumpkin pie	102
55	Crullers	111
56	Churro	168

Table 4: Selected categories from the Cakes Dataset

No.	Selected Meal	Number of Images
1	Red velvet cake	161
2	Chocolate cake	211
3	Cheesecake	225

Table 5: Selected categories from the Food-101 Dataset

No.	Selected Food Category	Number of Images
1	Ice cream	201
2	Hummus	217
3	Hot and sour soup	232
4	Greek salad	211
5	Garlic bread	246
6	Frozen yogurt	206
7	Fried calamari	251
8	French onion soup	223
9	Cup cake	246
10	Chicken quesadilla	269
11	Beet salad	195
12	Caesar salad	235
13	Club sandwich	183
14	Cannoli	260
15	Chicken curry	216
16	Baklava	201
17	Bread pudding	207
18	Carrot cake	280
19	Baby Back Ribs	997
20	Guacamole	491
21	Huevos Rancheros	282
22	Gnocchi	474
23	Foie Gras	429
24	Filet Mignon	467
25	Escargots	410
26	Eggs Benedict	430
27	Deviled Eggs	466
28	Creme Brulee	338
29	Crab Cakes	521
30	Clam Chowder	439
31	Ceviche	444
32	Bruschetta	415

Table 6: Selected Meal Categories Collected from Shutterstock and Google Images

No.	Meal Name	Number of Images
1	Aish El Saraya	111
2	Bamyeh	294
3	Basbousa	207
4	Fasolia	291
5	Fattoush	147
6	Freekeh with chicken	386
7	Halawa	125
8	Halawet el jibn	142
9	Kabsa	131
10	Kebbeh Nayeh	376
11	Kebbeh	175
12	Koba balaban	217
13	Koba Maqlya	384
14	Koba mashoua	323
15	Kofta Tahini	375
16	Lasagne	291
17	Maftool	209
18	Mahshi Kosa	301
19	Mahshi Malfoof	304
20	Makdous	370
21	Manakish Cheese	409
22	Manakish Zaatar	393
23	Maqloba	122
24	Ma'amoul	393
25	Meshabak sweet	118
26	Molokhia	134
27	Muhallebi	150
28	Namoora	116
29	Ouzi	169
30	Qamar Al-Din Dessert	72
31	Sahlab	168
32	Shish Tawook	132
33	Thareed	136
34	Tomato Sause Kofta	385
35	Umm Ali	149
36	Zainab Fingers sweet	162
37	Znoud El Sit sweet	131
38	Banana Bread	363
39	Chocolate Chip Cookies	360
40	Macarons	641
41	Panna Cotta	368
42	Cupcakes	246
43	Cheesecake Brownies	346
44	Truffles	153
45	Fruit Tart	191

Table 7: Selected categories from the Sri Lankan Food Image Dataset [1]

No.	Meal Name	Number of Images
1	Pizza - Cheese Lovers	100
2	Pizza - Hot And Spicy Chicken	100
3	Pizza - Sausage Delight	100
4	Pizza - Spicy Veggie With Paneer	100
5	Pizza - Veggie Supreme	100
6	Red Rice	100
7	Samba Rice	100
8	Strawberry Milkshake	100
9	Vegetable Rolls	100
10	Vegetable Pastry	100
11	Noodles	100
	Total:	**690 Images**

Table 8: Selected categories from the Food Image Classification dataset [2]

No.	Meal Name	Number of Images
1	Paani Puri	115
2	Dal Makhani	205
3	Butter Naan	173
4	Coffee	166
Total Categories:		**3**
Total Images:		**493**

.1 Complete List of Meals

Table 9: Complete list of meal categories used in the dataset

No.	Meal Name
1	Aish El Saraya
2	Almond Jelly
3	Apple Pie
4	Aseeda
5	Avocado Bagel Sandwich
6	Baby Back Ribs
7	Bagel Bread
8	Baked Potato
9	Baklava
10	Bamyeh
11	Banana Cake
12	Basbousa
13	BBQ Chicken Pizza
14	Beef Bean Soup

No.	Meal Name
15	Beef Bowl
16	Beef Curry
17	Beet Salad
18	Biryani
19	Boiled Chicken
20	Bread Pudding
21	Brownies
22	Bruschetta
23	Burger
24	Butter Naan Bread
25	Caesar Salad
26	Cannoli
27	Carrot Cake
28	Ceviche
29	Cheesecake
30	Cheesecake Brownie
31	Chicken Broth Soup
32	Chicken Curry
33	Chicken Fatteh
34	Chicken Liver
35	Chicken Nugget
36	Chicken Quesadilla
37	Chicken Rolls
38	Chinese Pumpkin Pie
39	Chinese Soup
40	Chocolate Cake
41	Chocolate Chip Cake
42	Churro
43	Cinnamon Rolls
44	Clam Chowder
45	Club Bagel Sandwich
46	Club Sandwich
47	Coconut Milk Soup
48	Corn Soup
49	Crab Cakes
50	Cream Puff
51	Creme Brulee
52	Crepe
53	Crispy Chicken
54	Croissant
55	Croque Madame
56	Crullers
57	Cup Cakes
58	Custard Tart
59	Dal Makhani
60	Date Cake

No.	Meal Name
61	Deep Fried Chicken
62	Deviled Eggs
63	Dhokla
64	Donut
65	Dynamite Shrimp
66	Eggplant with Garlic Sauce
67	Eggs Benedict
68	Escargots
69	Falafel
70	Fasolia
71	Fattoush
72	Fava Beans
73	Filet Mignon
74	Fish Fillet
75	Focaccia Breakfast
76	Focaccia with Cheese
77	Freekeh with Chicken
78	French Bread
79	French Onion Soup
80	Fried Calamari
81	Fried Mussel Pancakes
82	Fried Rice
83	Fried Shrimp
84	Fried Spring Rolls
85	Fries
86	Frozen Yogurt
87	Fruit Tart
88	Garlic Bread
89	Georgian Khachapuri
90	Gnocchi
91	Greek Salad
92	Green Salad
93	Grilled Chicken
94	Grilled Salmon
95	Guacamole
96	Halawa
97	Halawet El Jibn
98	Harees
99	Healthy Fruit Bowl
100	Hot and Sour Soup
101	Hot Dog
102	Hot Pot
103	Huevos Rancheros
104	Hummus
105	Ice Cream
106	Jalebi

No.	Meal Name
107	Kabsa
108	Kebab
109	Kebbeh
110	Kebbeh Nayeh
111	Kofta Tahini
112	Koshari
113	Kunafa
114	Lasagne
115	Lava Cake
116	Lemon Fig Jelly
117	Lentils Soup
118	Luqaimat
119	Ma'amoul
120	Macaroni Salad
121	Macarons
122	Maftool
123	Mahshi Kosa
124	Mahshi Malfoof
125	Makdous
126	Mandi
127	Mango Pudding
128	Mansaf
129	Maqloba
130	Markook
131	Masala Dosa
132	Meshabak Sweet
133	Mojadra
134	Molokhia
135	Moon Cake
136	Muffin
137	Muhallebi
138	Musakhan
139	Mushroom Risotto
140	Nachos
141	Namoora
142	Noodles
143	Omelet
144	Ouzi
145	Oysters
146	Paani Puri
147	Pancake
148	Panna Cotta
149	Parfait
150	Pasta Salad
151	Pastries
152	Pea Stew

No.	Meal Name
153	Peas with Meat
154	Pizza Cheese Lovers
155	Pizza Hot and Spicy
156	Pizza Sausage Deluxe
157	Popcorn
158	Popcorn Chicken
159	Potato Salad
160	Potato Soup
161	Pretzel
162	Pumpkin Soup
163	Qamar Al-Din Dessert
164	Qatayef
165	Quinoa Salad
166	Ramen Noodle
167	Rare Cheesecake
168	Red Velvet Cake
169	Rendang
170	Rice Ball
171	Rice Thai
172	Rice with Milk
173	Rice with Vermicelli
174	Roast Duck
175	Safeha
176	Sahlab
177	Samba Rice
178	Sambosa
179	Sandwich with Chicken
180	Santa Fe Salad
181	Sardines Fish
182	Sausage Breakfast
183	Scrambled Egg Breakfast
184	Seafood Pizza
185	Shakshuka
186	Shawarma
187	Shish Barak
188	Shish Tawook
189	Shrimp
190	Spaghetti
191	Spicy Chicken Soup
192	Spinach
193	Steak
194	Steamed Meat Dumplings
195	Steamed Rice Roll
196	Strawberry Jam Dessert
197	Strawberry Milkshake
198	Strawberry Shortcake

No.	Meal Name
199	Stuffed Duck
200	Stuffed Onion
201	Sushi
202	Sweet French Toast
203	Tabouli Salad
204	Taco
205	Taquito
206	Tiramisu
207	Toast
208	Tomato Sauce Kofta
209	Tomato Soup
210	Tortilla
211	Tripe and Intestines
212	Truffles Sweet
213	Tuna Fish
214	Umm Ali
215	Vegetable Noodles
216	Vegetable Pizza
217	Waffles
218	Walnut Shortcake
219	Warak Enab
220	White Beans with Meat
221	Zaatar Manakish
222	Zainab Fingers
223	Znoud El Sit Sweet

Table 10: Fast Food Dataset Classes and Image Distribution

Class ID	Food Category	Number of Images
1	Baked Potato	300
2	Burger	300
3	Crispy Chicken	300
4	Donut	300
5	Fries	300
6	Hot Dog	300
7	Pizza	300
8	Sandwich	300
9	Taco	300
10	Taquito	300

.1 Database Schemas

Table 11: Users Collection Schema

Field	Type	Description
uid	String	Firebase Auth unique identifier
email	String	User's email address
displayName	String	User's display name
emailVerified	Boolean	Email verification status
lastSignIn	Timestamp	Last authentication timestamp
isAnonymous	Boolean	Guest user identification
totalScans	Integer	Total image recognition operations
createdAt	Timestamp	Account creation time
updatedAt	Timestamp	Last profile update

Table 12: Recipes Collection Schema

Field	Type	Description
id	String	Unique recipe identifier
title	String	Recipe title in Arabic
englishTitle	String	Recipe title in English
category	String	Food category classification
calories	Integer	Total calories per serving
servings	Integer	Number of servings
readyInMinutes	Integer	Preparation time in minutes
views	Integer	View counter for popularity tracking
allergies	Array	List of potential allergens
tags	Array	Search and filter tags
ingredients	Array	Complete ingredients list
instructions	String	Step-by-step preparation instructions
nutrition	Object	Nutritional breakdown object
image	String	URL to recipe image (external storage)
createdAt	Timestamp	Recipe creation time
updatedAt	Timestamp	Last modification time

Table 13: Feedback Collection Schema

Field	**Type**	**Description**
id	String	Unique feedback identifier
userId	String	Submitting user reference
tagName	String	Related recipe or feature identifier
feedback	String	User's comments and suggestions
rating	Integer	Numerical rating value
email	String	User contact email
createdAt	Timestamp	Feedback submission time

Table 14: Application Statistics Schema

Field	**Type**	**Description**
total_requests	Integer	Total API requests processed
last_request	Timestamp	Most recent request timestamp
recipe_views	Integer	Recipe page view count
image_uploads	Integer	User image upload count
recipe_scans	Integer	Food recognition scan count
app_opens	Integer	Application launch count
active_sessions	Integer	Concurrent active user sessions
added_recipes	Integer	New recipes added
modified_recipes	Integer	Recipes modified

favorites Collection Schema

Table 15: favorites Collection Field Specifications

Field Name	Data Type	Description
addedAt	Timestamp	The date and time when the dish was added to favorites, in UTC format.
allergies	Array	An array containing allergens related to the dish (e.g., peanuts, nuts, shellfish). May be empty.
calories	Number	The total number of calories in the dish.
detectedDish	String	The name of the dish as detected by the recognition system (e.g., "Ceviche").
diets	Array	An array of dietary tags the dish adheres to (e.g., "Gluten Free", "Dairy Free").
dishId	String	A unique identifier for the dish (e.g., "ceviche").
englishTitle	String	The English title of the dish (e.g., "Shrimp Ceviche").
exercise	Null	A placeholder field for future exercise-related data. Currently null.
healthScore	Number	A numerical score representing the healthiness of the dish (e.g., 61).
image	String	A URL pointing to an image of the dish.
ingredients	Array	An array of strings listing the required ingredients for the dish.
instructions	String	Step-by-step cooking instructions for preparing the dish.
nutrition	Map	A nested map containing detailed nutritional information: calories, carbs, fat, protein.
rating	Null	A placeholder for user ratings. Currently null.
readyInMinutes	Number	The total preparation and cooking time required in minutes.
servings	Number	The number of servings the recipe yields.
source	String	The source of the recipe data (e.g., "Local Database").
substitutes	String	Suggestions for ingredient substitutions or variations.
tags	Array	An array of descriptive tags for the dish (e.g., "Latin American", "Appetizer").
title	String	The primary title/name of the dish (e.g., "Ceviche").
userId	String	The unique Firestore identifier of the user who added the dish to favorites.

history Collection Schema

Table 16: history Collection Field Specifications

Field Name	Data Type	Description
addedAt	Timestamp	The date and time the dish was logged in the history, in UTC format.
allergies	Array	An array containing allergens related to the dish. May be empty.
calories	Number	The total number of calories in the dish.
category	String	The meal category (e.g., "Pasta").
clientAddedAt	Timestamp	The client-side timestamp when the dish was added (can be useful for offline sync).
detectedDish	String	The name of the dish as detected (e.g., "Lasagne").
diets	Array	An array of dietary tags (e.g., "Vegetarian Friendly", "Gluten Free").
englishTitle	String	The English title of the dish (e.g., "Lasagne").
healthScore	Number	A numerical score representing the healthiness of the dish (e.g., 50).
image	String	A URL pointing to an image of the dish.
ingredients	Array	An array of strings listing the required ingredients.
instructions	String	Step-by-step cooking instructions.
mealKey	String	A unique key identifier for the meal from its source (e.g., "lasagne").
nutrition	Map	A nested map containing nutritional breakdown: calories, carbs, fat, protein.
readyInMinutes	Number	The total preparation and cooking time required in minutes.
servings	Number	The number of servings the recipe yields.
source	String	The source of the recipe data (e.g., "TheMealDB").
sourceUrl	String	The original URL where the recipe can be found online.
substitutes	String	Suggestions for ingredient substitutions or variations.
tags	Array	An array of descriptive tags (e.g., "Main Course").
title	String	The primary title/name of the dish (e.g., "Lasagne").
userId	String	The unique Firestore identifier of the user who owns this history record.

Table 17: Deleted Recipes Collection Schema

Field	Type	Description
id	String	Original recipe identifier
title	String	Recipe title
deleted	Boolean	Deletion status indicator
deletedAt	Timestamp	Deletion execution time
canRecover	Boolean	Recovery eligibility status
deletedBy	String	Administrative user identifier
reason	String	Deletion justification
createdAt	Timestamp	Original creation time

Table 18: Access Control Matrix by User Role

User Role	Users	Favorites	History	Recipes	Ratings
Anonymous Users	Restricted	Restricted	Restricted	Read Only	Restricted
Authenticated Users	Own Data Only	Own Data Only	Own Data Only	Read Only	Write only
Administrative Users	Full Access	Full Access	Full Access	Full Access	Full Access

Web-Specific Collections

Table 19: Firestore Collections for Web Platform Operations

Collection Name	Description
`website_messages`	Primary storage for all contact form submissions from website visitors. Contains complete message details, sender information, and metadata for tracking message lifecycle from submission through administrative response.
`website_faq`	Central repository for Frequently Asked Questions displayed to website visitors. Supports bilingual content (Arabic/English), categorization, view tracking, and content lifecycle management through active/inactive status flags.
`user_questions`	Temporary storage for questions submitted by users when existing FAQ content does not address their inquiries. Serves as input queue for expanding the knowledge base and direct user support.
`admin_users`	Authentication and authorization database for website administrators. Stores administrator credentials and access permissions separate from the main Firebase Authentication system used by mobile application users.

admin_users Collection Schema

Table 20: admin_users Collection Field Specifications

Field Name	Data Type	Description
id	String	Unique document identifier automatically generated by Firestore. Used as primary key for document retrieval and reference.
email	String	Administrator's email address serving as primary authentication credential. Must be unique across the collection.
password	String	Hashed password for administrator authentication. In production deployments, this field contains securely hashed values rather than plaintext passwords.

user_questions Collection Schema

Table 21: user_questions Collection Field Specifications

Field Name	Data Type	Description
id	String	Unique document identifier automatically generated by Firestore for each submitted question.
category	String	User-selected categorization of the question type, matching FAQ categories for organizational consistency.
createdAt	Timestamp	Server-side timestamp recording exact submission time for queue management and response time tracking.
email	String	Submitter's email address for response delivery and follow-up communication.
ipAddress	String	Client IP address captured for security monitoring, duplicate submission detection, and geographic analysis.
name	String	Submitter's provided name for personalized response addressing.
question	String	Complete question text as entered by the user, preserved with original formatting and content.
source	String	Submission origin identifier distinguishing between FAQ page submissions and other website contact points.
status	String	Processing workflow state indicator with values including "pending," "answered," or "converted" to FAQ.
userAgent	String	Complete HTTP user agent string capturing browser type, version, operating system, and device information.

website_messages Collection Schema

Table 22: website_messages Collection Field Specifications

Field Name	Data Type	Description
id	String	Unique document identifier automatically generated by Firestore for audit trail and reference purposes.
email	String	Sender's contact email address for response delivery and communication tracking.
ipAddress	String	Client IP address captured for security validation, geographic analysis, and duplicate submission prevention.
message	String	Complete message content as submitted by the user, preserving original formatting and structure.
name	String	Sender's provided name for personalized response addressing and relationship management.
page	String	Source page URL capturing submission context for analytical understanding of user navigation patterns.
readAt	Timestamp	Administrative processing timestamp recording when message was reviewed by support personnel.
source	String	Submission channel identifier distinguishing between contact forms and other website interaction points.
status	String	Workflow state tracking message progression through "unread," "read," "replied," and "archived" stages.
subject	String	Message topic summary for quick administrative review and categorization purposes.
timestamp	Timestamp	Original submission timestamp for queue management and response time calculation metrics.
userAgent	String	Complete browser and device information for technical context and support issue diagnosis.

website_faq Collection Schema

Table 23: website_faq Collection Field Specifications

Field Name	Data Type	Description
`id`	String	Unique document identifier automatically generated by Firestore for each FAQ entry.
`answer`	String	Complete answer content with support for multilingual text, HTML formatting, and embedded media references.
`category`	String	Organizational classification determining display grouping and filtering options for website visitors.
`createdAt`	Timestamp	Initial creation timestamp for content lifecycle tracking and audit purposes.
`isActive`	Boolean	Visibility control flag allowing administrators to temporarily hide content without permanent deletion.
`order`	Number	Display sequence position within category groups, enabling manual ordering of FAQ presentation.
`question`	String	Question text as displayed to website visitors, often in multiple languages for bilingual support.
`updatedAt`	Timestamp	Last modification timestamp for content freshness tracking and version management.
`views`	Number	Cumulative view counter tracking FAQ popularity and user engagement metrics.