# Submission 2: "Munchly" App Requirements

## 1. Functional Requirements

The functional requirements define the main features and interactions of the "Munchly" application.

### User Authentication and Profile Management

- Users can securely register, log in, and manage their profiles using Firebase Authentication.

- Users can configure preferences, such as dietary restrictions and favorite ingredients.

### Image Scanning and Recognition

- Users can capture images of refrigerator contents using their phone's camera.

- The app identifies objects in the image using Google Vision API. The principle of object recognition draws inspiration from models like YOLO, but the implementation relies on Google Vision API for reliable and scalable image recognition.

### Inventory Management

- Identified items are automatically added to the user's inventory.

- Users can manually add, update, or remove items in the inventory.

- Users can search and filter recipes within the recipe database.

- Inventory data is stored in Firebase Firestore.

### Recipe Generation

- Recipes are dynamically suggested based on the items available in the fridge.  
- Recipes are generated using Gemini, an AI-powered component integrated directly with Firebase, ensuring efficient and personalized recipe generation.  
- Recipes are stored in Firebase Firestore for quick retrieval and display.

### User Dashboard

- The dashboard displays:  
 - Inventory status.  
 - Suggested recipes.  
 - Options to create a recipe, add items for recipe creation, and search/filter recipes within the database.

## 2. Use Case Analysis

### Use Case: Scan Refrigerator and Suggest Recipes

- Pre-condition: User is logged in.

- Basic Flow:

1. User captures an image of the refrigerator.  
 2. The Google Vision API processes the image and identifies items.  
 3. Identified items are added to the inventory.  
 4. The app queries Firebase for matching recipes.  
 5. If no matching recipes are found, the app generates personalized recipes using Gemini AI.  
 6. Recipes are displayed to the user.

- Alternate Flow:

- User manually corrects or adds missing items.

## 3. Technological Requirements

Programming Language: JavaScript

Development Platform: VSCode

Frontend Framework: React (with React Native for mobile compatibility, if required)

Backend Framework: Express.js

### Backend Services:

- Firebase Firestore: Cloud database for inventory and recipes.

- Firebase Storage: For storing fridge images.

- Firebase Authentication: For secure user login and profiles.

### AI Components:

- Google Vision API: For object detection and image recognition, ensuring accurate identification of fridge contents.  
- Gemini AI: Integrated for recipe generation, leveraging Firebase for seamless operation and personalized outputs.

### Testing Frameworks:

- Jest for frontend and backend testing.

- Supertest for API testing.

## 4. Clarification on AI Terminology

The term "AI-driven image recognition" refers to the integration of the Google Vision API, which utilizes advanced machine learning algorithms to identify objects in images. While models like YOLO are widely recognized for their real-time object detection capabilities, Munchly leverages the Google Vision API for its robust and accessible features.

For recipe generation, the term "AI-powered recipe generation" highlights the use of Gemini, an AI solution connected with Firebase, which dynamically creates recipes based on user preferences and inventory data.