Submission 2: Detailed Requirements Document for "Munchly"

---

1. Functional Requirements

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

1. **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.
2. **Image Scanning and Recognition**
    - Users can capture images of refrigerator contents using their phone's camera.
    - The app identifies objects in the image using **YOLO**, integrated via a backend service with TensorFlow bindings or external APIs.
3. **Inventory Management**
    - Identified items are automatically added to the user's inventory.
    - Users can manually add, update, or remove items in the inventory.
    - Inventory data is stored in **Firebase Firestore**.
4. **Recipe Generation**
    - Recipes are dynamically suggested based on the items available in the fridge.
    - Recipes can be generated using either:
        - **ChatGPT API** for personalized and creative suggestions.
        - **OLAMA (or equivalent alternative)** for enhanced recipe logic and structure.
    - Recipes are stored in Firebase Firestore for quick retrieval and display.
5. **Notifications and Alerts**
    - Users receive alerts for:
        - Food nearing expiration dates.
        - New recipe suggestions dynamically generated using AI (either ChatGPT or OLAMA).
    - Notifications are powered by **Firebase Cloud Messaging (FCM)**.
6. **Search and Filters**
    - Users can search recipes by keyword and filter them by dietary preferences (e.g., vegan, low-carb).
7. **User Dashboard**
    - The dashboard displays:
        - Inventory status.
        - Upcoming expiration alerts.
        - Suggested recipes.

---

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. YOLO 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 **either ChatGPT API or OLAMA**.
    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.
    - **Firebase Cloud Messaging**: For push notifications.
- **AI Components**:
    - **YOLO**: For object detection and image recognition.
    - **ChatGPT API or OLAMA**: Depending on implementation, one AI model will generate personalized recipes based on fridge contents.
- **Testing Frameworks**: Jest for frontend and backend testing, Supertest for API testing.

Clarification on AI Terminology

The term "AI-driven image recognition" is used because YOLO (You Only Look Once) is a deep learning model trained on vast image datasets. While machine learning techniques, such as convolutional neural networks (CNNs), are used to train YOLO, the final product operates as an AI-powered model capable of intelligent object detection in real-time. For recipe generation, **either ChatGPT or OLAMA** will be used, depending on the implementation choice.