CSCI3310 Project Proposal: AllergyGuard

Group Information

- **ID** : 18
- **Members:**
    i. **SO Chun Ning** (AIST) - 1155192846 (solo)

1. Languages

- **Kotlin**: The primary programming language for the Android application, chosen for its modern syntax, null safety, and official support from Google.
- **XML**: For designing UI layouts and resources.

2. Focus

Problem Solving: This project focuses on a specific, high-stakes problem—food safety for travelers with allergies. While it uses existing technologies (OCR), the application applies them creatively to filter and highlight dangerous information rather than just providing a raw translation.

3. Main Task & Features

Main Task: To prevent allergic reactions by scanning food labels and menus in real-time, identifying simplified keywords, and alerting the user if a selected allergen is present.

Target Features:

1. **Profile Management**: Users can select from a list of common allergens (Peanuts, Gluten, Shellfish, etc.) or add custom keywords.
2. **Smart Scan (OCR)**: Use the camera to scan text on packaging or menus.
3. **Allergen Detection Logic**: Analyze scanned text and compare it against the user's active profile using fuzzy string matching to account for OCR imperfections.
4. **Visual Alerts**: Overlay simple indicators (Green/Red) on the screen to provide immediate feedback.
5. **History Log**: Save scanned results and locations for future reference.

4. Technologies, Libraries & APIs

- **Google ML Kit (Text Recognition API)**: For on-device Optical Character Recognition (OCR) to convert images to text.
- **CameraX**: For easy-to-use, lifecycle-aware camera controlling.
- **Room Database**: For persisting user profiles, custom allergen words, and scan history locally on the device.
- **Coroutines**: For handling background tasks (OCR processing and Database operations) without freezing the UI.
- **ViewBinding**: For safer interaction with UI elements.

5. Difficulty Ranking (1 = Easiest, 5 = Hardest)

1. **Profile Management (Difficulty: 2)**: Standard CRUD operations using Room Database and RecyclerView.
2. **Camera Integration (Difficulty: 3)**: Implementing CameraX to get a preview stream and capturing images is straightforward but requires careful lifecycle management.
3. **Visual Overlay UI (Difficulty: 4)**: Drawing a dynamic overlay over the camera preview that matches the position of the text requires coordinate mapping and custom views.
4. **ML Kit Integration & Data Pipeline (Difficulty: 4)**: Configuring the text recognizer, handling the asynchronous data stream, and ensuring the app doesn't crash due to memory issues on the emulator.
5. **Fuzzy Matching Logic (Difficulty: 5)**: Implementing a robust algorithm to detect "Peanut" even if OCR sees "Peanu t" or "PeAnut" (case/whitespace/typo handling) is the core logic challenge.