Project: WMS Photo & Scan Internal Tool

Client: maciejjurkie856 (Fiverr) **Platform:** Android Tablet (Internal Warehouse Use) **Timeline:** 14 Days **Version:** 1.0

---

1. Project Overview

We are building an internal Flutter application for Android tablets (Android 8.0+). The app's purpose is to document incoming devices in a warehouse facility. It will scan a barcode (BCN), capture multiple photos of the device, upload them to Google Drive, and redirect the user to a Web-based WMS system with specific data parameters.

Key Constraint: The application acts as a bridge between the physical device and the client's web WMS.

---

2. Technical Stack & Environment

- **Framework:** Flutter (Dart).
- **Target OS:** Android 8.0 (Oreo) and higher.
- **Device Type:** Tablets (Orientation: Likely Landscape/Portrait adaptable).
- **Source Code: REQUIRED.** Clean code is essential as the client will perform a security audit on the source code.

Required Libraries/Integrations:

1. **Barcode Scanning:**
   - **Client Request:** "Library with camera barcode functionality like *ZXing Android Embedded*."
   - **Implementation:** Use `flutter_zxing` or `mobile_scanner`. Must use the device **Camera**, not a hardware laser scanner.
2. **Cloud Storage:** Google Drive API (REST API or Google Client SDK).
3. **Authentication:** Google Sign-In (using the Google account already configured on the Android device).

---

3. UI/UX Requirements (Single Screen Flow)

The app consists of a **Main Screen** containing the following elements:

1. **BCN Input Section:**
    - **Textbox:** For manual entry of the BCN (Barcode Control Number).
    - **Scan Button:** Launches the camera view to scan the barcode. Populates the textbox upon success.
2. **Description Section (New Requirement):**
    - **Text Area:** A multi-line input field for the user to add notes about the device condition.
3. **Media Section:**
    - **Camera View/Button:** To take photos of the device.
    - **Internal Gallery (Grid View):**
        - Displays thumbnails of photos taken in the current session.
        - **Action:** Tapping a photo allows the user to view it.
        - **Delete Action:** Small "X" icon on thumbnails to remove "bad" photos before uploading.
4. **Action Section:**
    - **"Upload & Finish" Button:** Triggers the upload process and WMS redirection.

---

4. Functional Logic & Data Flow

Step 1: Authentication

- On app launch, check if the user is signed in via Google.
- If not, trigger the standard `google_sign_in` flow using the primary account on the Android tablet.

Step 2: Data Capture

- User scans BCN (e.g., "12345-DEVICE").
- User types description (e.g., "Screen scratched").
- User captures 1–5 photos. Photos are stored locally in the app cache first.

Step 3: Google Drive Upload (Background Process)

- **Action:** When user clicks "Upload".
- **Logic:**
    1. Create a specific folder on Google Drive (Name convention: Recommend using the BCN Number or Date).
    2. Upload all images from the local gallery to this new folder.
    3. **Crucial:** Retrieve the **Shareable Web Link** (URL) of that specific Google Drive folder.
        - *Example URL format:* `https://drive.google.com/drive/u/0/folders/28CrKaB2ATKHdoXmF-uqYAftioUfrA`

Step 4: WMS Integration (The Hand-off)

- Once the upload is successful, the app must launch the device's default web browser.
- **Target URL:** The client's WMS system (Client will provide base URL, for now, use a placeholder).
- **Query Parameters:** The app must append the following data to the URL:
    1. `bcn` = The scanned barcode value.
    2. `drive_url` = The Google Drive folder link generated in Step 3.
    3. `desc` = The text from the Description field.

Constructed URL Logic: `[WMS_BASE_URL]?bcn=[VALUE]&drive_url=[LINK]&desc=[TEXT]`

---

5. Security & Permissions (Critical)

- **Permissions:** App will need `CAMERA`, `INTERNET`, and `MANAGE_EXTERNAL_STORAGE` (or specific scoped storage access depending on Android version).
- **Data Privacy:** Ensure temporary images are cleared from the device cache after a successful upload to save space on the tablets.

---

6. Out of Scope (Phase 2)

- **Google Chat Integration:** The client requested sending notifications to a Google Chat group. This is **NOT** included in the current build. Do not implement this yet.

---

7. Delivery Checklist

1. [ ] Fully functional APK.
2. [ ] Source Code (Zipped or Git Repo access).
3. [ ] Instructions on how to set up the Google Cloud Console (Client needs to know how to enable Drive API for their own account if they are compiling it themselves).