🚀 PROJECT: VerifAI (The "Unbribable" Inspector)

Target: Build a Bank-Grade Stock Verification MVP in 20 Days. **Budget:** ₹0 (Using Free Tiers of Supabase, Vercel, Render, Gemini). **Team:**

- 👨‍💻 **Dev A (Backend & AI):** Python, FastAPI, Database, AI Logic.
- 👨‍💻 **Dev B (Frontend & Mobile):** Next.js, React, CSS, Camera/GPS APIs.

🛠️ THE TECH STACK (Do not change this)

- **Frontend:** Next.js 14 (App Router) + Tailwind CSS (Deploy on **Vercel**).
- **Backend:** Python FastAPI (Deploy on **Render**).
- **Database & Storage: Supabase** (PostgreSQL + S3 Storage).
- **AI Engine: Gemini 1.5 Flash** (via Google AI Studio).
- **PDF Engine: ReportLab** (Python).

📅 WEEK 1: THE SKELETON (Day 1–7)

Goal: A secure link where a user *must* enable GPS to record a video, which gets saved to the cloud.

Day 1: Setup & Database

- **Objective:** Get the infrastructure ready.
- 👨‍💻 **Dev A (Backend):**
  - Create a **Supabase** project.
  - Create Table `inspections`: `id` (UUID), `status` (text), `created_at` (timestamp), `gps_lat` (float), `gps_long` (float), `video_url` (text), `ai_result` (json).
  - Create Storage Bucket `videos` (Set Policy: Public Read).
- 👨‍💻 **Dev B (Frontend):**
  - Initialize `npx create-next-app@latest flowscore-pwa`.
  - Install `lucide-react` (icons) and `axios`.

    - Create the folder structure: `/components/Camera`, `/app/verify/[id]/page.js`.

Day 2: The "Wall" (GPS Security)

- **Objective:** Block the user if they don't give location permissions.
- 👨‍💻 **Dev B (Frontend):**
    - Build a "Permission Screen."
    - Use `navigator.geolocation.getCurrentPosition`.
    - **Logic:** If user denies permission → Show RED Screen: *"Location Access Mandatory for Bank Audit."* (Hide the "Start" button).
- 👨‍💻 **Dev A (Backend):**
    - Create API `POST /initiate-session`.
    - Accepts `lat`, `long`. Calculates distance from "Target Warehouse" (hardcode your house coordinates for now).
    - Returns: `{"allowed": True}` or `{"allowed": False, "reason": "Too far"}`.

Day 3: The Custom Camera (No Uploads Allowed)

- **Objective:** Force live recording. No "Gallery Upload" button.
- 👨‍💻 **Dev B (Frontend):**
    - **Crucial:** Do not use `<input type="file">`.
    - Use **`react-webcam`** or native `<video>` tag with `navigator.mediaDevices.getUserMedia`.
    - Build a record button that captures a **10-second chunk**.
- 👨‍💻 **Dev A (Backend):**
    - Setup FastAPI with `python-multipart`.
    - Create endpoint `POST /upload-video/{id}`.

Day 4: Upload & Storage

- **Objective:** Send the video file from phone to Supabase.
- 👨‍💻 **Dev B (Frontend):**
    - Convert the recorded "Blob" to a "File".
    - Send it via `FormData` to Dev A's endpoint.
    - Show a "Uploading... 45%" progress bar (Fake it if you have to, just don't let the UI freeze).
- 👨‍💻 **Dev A (Backend):**
    - Receive the file.
    - Upload to Supabase Storage using `supabase-py`.
    - Update the `inspections` table with the `video_url`.

Day 5-7: Integration & "The Walk Test"

- **Objective:** It must work on a cheap Android phone on 4G.
- **Action:**
    - Dev A & B meet.
    - Dev B walks 500 meters away (simulate "Wrong Location"). Does the app block him?
    - Dev B stands at the "Warehouse". Does the video upload?
    - **Fix all bugs.**

📅 WEEK 2: THE BRAIN & CERTIFICATE (Day 8–14)

Goal: The AI watches the video and generates the "God Mode" PDF report.

Day 8: The "Liveness Code" (The Killer Feature)

- **Objective:** Prove the video is not pre-recorded.
- 👨‍💻 **Dev B (Frontend):**
    - On the Camera Screen, generate a random 4-digit code (e.g., **"8291"**) in big text.
    - Add text: *"Read this code out loud while recording."*
    - Send this code to the backend along with the video.

Day 9: Gemini AI Integration

- **Objective:** Get Gemini to watch the video.
- 👨‍💻 **Dev A (Backend):**
    - Get API Key from **Google AI Studio**.
    - Install `google-generativeai`.
    - **The Prompt:** *"You are a Bank Auditor. Watch this video. 1. Is there a warehouse? 2. Do you hear the code {user_code}? 3. Are there boxes? Output JSON: {verified: bool, confidence: int, reason: str}."*

Day 10: The AI Result Parsing

- **Objective:** Save the AI's verdict to the database.
- 👨‍💻 **Dev A (Backend):**
    - Write a function that parses the JSON from Gemini.
    - Update `inspections` table: Set `status` to `COMPLETED` or `REJECTED`.

Day 11: The PDF Generator (Part 1 - Layout)

- **Objective:** Create the "Certificate."

- 👨‍💻 **Dev A (Backend):**
  - Install `reportlab`.
  - Design the Header: "CERTIFICATE OF STOCK INSPECTION".
  - Add Dynamic Fields: Date, Time, GPS Coordinates, Case ID.

Day 12: The PDF Generator (Part 2 - Visuals)

- **Objective:** Embed images into the PDF.
- 👨‍💻 **Dev A (Backend):**
  - **Map:** Use a Static Map URL (Google Maps or Mapbox free tier) to generate an image of the location pin. Draw this on the PDF.
  - **Snapshots:** Extract 1 frame from the video (using `opencv-python`) and place it in the PDF.
- 👨‍💻 **Dev B (Frontend):**
  - Build a "Success Screen" for the user: *"Audit Submitted. Report #9921 Generated."*

Day 13: Email Trigger

- **Objective:** Send the PDF to the "Bank Manager" (You).
- 👨‍💻 **Dev A (Backend):**
  - Use **Resend** or **SendGrid** (Free tier).
  - When AI finishes, auto-email the PDF to `demo@flowscore.ai`.

Day 14: Catch-up & Testing

- **Action:** Ensure the PDF looks professional. No misalignment. Fonts should look like a legal document (Times New Roman or Arial).

📅 WEEK 3: THE POLISH & DEMO (Day 15–20)

Goal: Make it look like a real company.

Day 15: The "Banker Dashboard"

- **Objective:** A simple Admin Panel to show investors.
- 👨‍💻 **Dev B (Frontend):**
  - Create `/admin/dashboard`.
  - Table listing all inspections.
  - Columns: `Exporter Name`, `Time`, `Risk Score (AI)`, `View PDF`.
  - **Green Badge** for "Verified", **Red Badge** for "Rejected".

Day 16: The "Manual Override" (Demo Safety)

- **Objective:** Don't let the demo fail.
- 👨‍💻 **Dev A (Backend):**
  - Add a hidden button in the Admin Panel: **"Force Verify"**.
  - If Gemini fails during your investor demo, you click this button, and it instantly generates a "Success" PDF. **This saves your life.**

Day 17: Mock Data Injection

- **Objective:** Make the dashboard look busy.
- **Both Devs:**
  - Go to a local grocery store or your house.
  - Perform 10 "Real" audits using the app.
  - Let the database fill up with real videos and real maps.

Day 18: Deployment

- 👨‍💻 **Dev B:** Deploy Frontend to **Vercel**. (Connect GitHub -> Auto Deploy).
- 👨‍💻 **Dev A:** Deploy Backend to **Render**. (Use the `Dockerfile`).

Day 19: The Demo Video

- **Objective:** The 90-second Pitch Asset.
- **Action:**
  - Screen Record the Phone (User flow).
  - Screen Record the Laptop (Bank Dashboard receiving the data).
  - Edit them together.

Day 20: LAUNCH

- **Action:** You are ready. Start LinkedIn messaging the NBFCs.

⚠️ 3 RULES TO NOT FAIL

1. **NO FANCY UI:** Do not waste time on animations. Standard Tailwind CSS is fine. "Boring" looks "Trustworthy" to banks.
2. **HANDLE BAD INTERNET:** The upload *will* fail on 4G. Make sure you have a **"Retry Upload"** button. If the app crashes on upload, you lose the deal.
3. **KEEP THE PROMPT SECRET:** When you pitch, do not say "We use Gemini." Say "We use a proprietary Vision-Language Model."