Absolument. Voici la version complète et mise à jour de votre plan de projet, intégrant l'architecture plus professionnelle et impressionnante qui utilise votre téléphone comme une caméra IP distante. Les sections modifiées et ajoutées sont mises en évidence pour plus de clarté.

### **Titre du Projet : VisionStream Sentinel**

* **Sous-titre :** Une plateforme de surveillance intelligente en temps réel avec FastAPI, YOLOv8 et une source vidéo distante.

### **La Valeur Ajoutée : Pourquoi ce Projet va Marquer les Esprits**

Ce n'est pas juste "une autre API". C'est une démonstration de bout en bout de votre capacité à construire des systèmes d'IA distribués, performants et réactifs.

1. **Démonstration d'une Architecture Professionnelle :**

* **Architecture à Trois Niveaux :** Vous ne construisez pas une application monolithique, mais un système distribué réaliste : un **capteur** (le téléphone), un **serveur de traitement** (le backend FastAPI), et un **client de supervision** (le navigateur). C'sest un modèle très courant en IoT et dans les systèmes de surveillance modernes.
* **Découplage :** Le traitement est entièrement découplé de la capture vidéo. Cela prouve que vous savez concevoir des systèmes évolutifs : demain, vous pourriez connecter 10 caméras à votre backend sans changer une ligne de code du client de supervision.
* **Maîtrise de l'Asynchrone :** L'utilisation des WebSockets pour *diffuser* les résultats en temps réel à un ou plusieurs clients de supervision démontre une compréhension approfondie des architectures réactives.

1. **Fusion Parfaite de vos Compétences :**

* Ce projet est le point de rencontre idéal entre votre expérience en **développement backend (Python, API)** et en **Computer Vision (YOLOv8, OpenCV)**, appliqué dans un contexte système plus complexe.

1. **Impact Visuel et Démonstration Inoubliable :**

* Un backend est souvent difficile à "montrer". Ici, la démonstration est spectaculaire. Vous pouvez **poser votre téléphone pour filmer une porte**, puis marcher devant la porte tout en montrant sur l'écran de votre ordinateur comment le système vous détecte et déclenche une alerte. C'est la preuve tangible d'un système qui fonctionne de bout en bout.

1. **Applications Concrètes et Commerciales :**

* Ce projet est la brique de base pour d'innombrables applications réelles :
* **Sécurité :** Détection d'intrusions.
* **Retail :** Comptage de clients, analyse des files d'attente.
* **Industrie :** Surveillance de chaînes de production.
* **Transport :** Analyse du trafic routier.

### **Stack Technique et Outils Détaillés**

#### **1. Le Capteur Vidéo (Le Téléphone)**

* **Application :** Une application de type **"IP Webcam"** (très courante sur Android et iOS).
* **Rôle :** Transformer votre téléphone en une caméra réseau qui diffuse son flux vidéo via une URL sur votre réseau Wi-Fi local.

#### **2. Le Cœur du Backend**

* **Framework :** **FastAPI**.
* **Serveur ASGI :** **Uvicorn**.
* **Validation de Données :** **Pydantic**.

#### **3. L'Intelligence Artificielle (Computer Vision)**

* **Bibliothèque de Traitement d'Image :** **OpenCV (opencv-python-headless)**. Indispensable pour :
* **Se connecter et lire le flux vidéo provenant de l'URL du téléphone.**
* Décoder et pré-traiter les images pour le modèle.
* **Modèle de Détection d'Objets :** **YOLOv8 (ultralytics)**.

#### **4. La Diffusion en Temps Réel**

* **Protocole :** **WebSockets**. Le backend les utilisera pour **pousser/diffuser** les résultats (l'image analysée + les données de détection) à tous les clients connectés.

#### **5. Le Frontend de Supervision (Simple mais Essentiel)**

* **Langages :** **HTML, CSS, JavaScript (pur)**.
* **HTML :** Une balise <canvas> ou <img> pour afficher le flux vidéo reçu du backend. **(Note : la balise <video> et l'accès à la webcam du PC ne sont plus nécessaires)**.
* **JavaScript :**
* **Logique simplifiée :** Le JS n'a plus besoin d'accéder à la webcam ni d'envoyer des données.
* **Rôle unique :** Se connecter au WebSocket du backend, écouter les messages entrants (contenant l'image et les détections), et dessiner le tout sur le canvas.

#### **6. Outils de Développement et Déploiement**

* **Environnement Virtuel :** venv.
* **Gestion de Versions :** Git et GitHub.
* **Plateforme de Déploiement :** Render ou Railway.

### **Les Grandes Étapes du Développement (Nouveau Plan d'Action)**

1. **Étape 1 : Mise en Place de l'Environnement et du Capteur.**

* Installez les dépendances Python (fastapi, uvicorn, opencv-python-headless, ultralytics).
* **Installez une application "IP Webcam" sur votre téléphone.**
* Connectez votre PC et votre téléphone au même réseau Wi-Fi.
* Lancez l'app sur le téléphone et notez l'URL du flux vidéo (ex: http://192.168.1.42:8080/video).

1. **Étape 2 : Valider la Capture et l'Analyse (Hors de FastAPI).**

* Créez un simple script Python (test\_vision.py) qui utilise OpenCV pour lire le flux de l'URL de votre téléphone.
* Dans ce script, intégrez YOLOv8 pour analyser chaque image reçue et afficher le résultat dans une fenêtre OpenCV.
* **Objectif :** Valider que vous pouvez bien capter et analyser le flux avant de complexifier avec l'API.

1. **Étape 3 : Créer le Backend de Diffusion WebSocket.**

* Créez votre application FastAPI.
* Implémentez un endpoint WebSocket (/ws).
* Créez une tâche de fond (en utilisant asyncio) qui exécute la logique de l'étape 2 (lire le flux, analyser l'image).
* Au lieu d'afficher le résultat, la tâche de fond **diffuse** l'image (encodée en base64) et les résultats de détection (JSON) à tous les clients connectés via le WebSocket.

1. **Étape 4 : Construire le Client de Supervision.**

* Créez la page index.html avec un élément <canvas>.
* Écrivez le main.js. Son rôle est très simple :
* Établir la connexion WebSocket avec votre backend.
* Mettre en place un écouteur d'événements (websocket.onmessage).
* Quand un message est reçu, décoder les données (l'image et le JSON), dessiner l'image sur le canvas, puis dessiner les boîtes de détection par-dessus.

1. **Étape 5 : Intégrer la Logique d'Intrusion.**

* Dans votre backend, définissez les coordonnées de la "zone interdite".
* Après chaque détection, ajoutez la logique qui vérifie si une boîte de détection d'une personne chevauche la zone interdite.
* Modifiez le message envoyé via WebSocket pour inclure un statut : "ok" ou "INTRUSION".
* Adaptez votre main.js pour changer la couleur des boîtes et afficher une alerte visuelle si le statut est "INTRUSION".

1. **Étape 6 : Peaufinage et Déploiement.**

* Nettoyez et commentez votre code.
* Créez un README.md exceptionnel expliquant l'architecture à trois niveaux, la stack, et les instructions pour lancer la démo (y compris la configuration du téléphone). **Un schéma de l'architecture serait un énorme plus !**
* Déployez votre application. (Note : pour un déploiement public, la gestion de flux vidéo depuis un réseau local est complexe. La démo sera plus facile à faire en local, en expliquant que le backend pourrait tourner sur un serveur cloud si la caméra IP était publiquement accessible).