ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

École Marocaine des Sciences de l'Ingénieur

AgroSense : Application Intelligente Utilisant la Reconnaissance Faciale

Projet de Fin d'Année

Filire : Ingénierie Informatique et Réseaux (5ème année)

Réalisé par :
ElHachchad Abdelghaffar
Kanaoui Aymane

Encadré par :
Mr. Abderrahim Larhlimi

Membres de jury :
Mr. Abderrahim Larhlimi
Mr. Chiba

Année Universitaire 2025 - 2026

Table des matières

Table des figures

Introduction Générale

L'agriculture moderne et les marchés financiers sont deux domaines qui, bien que distincts, partagent un besoin commun crucial : l'accès rapide et précis à l'information pour la prise de décision. Dans le contexte marocain et international, les agriculteurs sont confrontés à des défis croissants liés à la gestion des cultures, à la détection précoce des maladies et à l'optimisation des ressources. Parallèlement, l'investissement boursier nécessite des outils d'analyse performants pour prédire les tendances du marché.

Le projet **AgroSense** naît de la volonté de combiner ces deux mondes à travers une solution technologique innovante. Il s'agit d'une application mobile multi-facettes développée en Flutter, intégrant des technologies d'Intelligence Artificielle de pointe. AgroSense ne se contente pas d'être un simple outil de gestion ; elle se positionne comme un assistant intelligent capable de diagnostiquer les plantes, de prédire les cours de la bourse, et d'interagir naturellement avec l'utilisateur via un assistant vocal et un système de génération augmentée de récupération (RAG).

Cependant, la puissance de ces fonctionnalités nécessite un niveau de sécurité élevé. C'est pourquoi un accent particulier a été mis sur la **sécurité biométrique**. L'accès à l'application est verrouillé par un système de reconnaissance faciale robuste, développé "from scratch" en utilisant des réseaux de neurones siamois (Siamese Neural Networks). Ce choix garantit que les données sensibles et les fonctionnalités critiques de l'application restent protégées et accessibles uniquement aux utilisateurs autorisés.

0.1 Objectifs détaillés

L'objectif principal d'AgroSense est de fournir une plateforme mobile unifiée qui apporte aux utilisateurs des outils d'aide à la décision fondés sur l'intelligence artificielle. Plus précisément, le projet vise :

- **Sécurité biométrique :** d'implémenter un système d'authentification faciale privatif fondé sur le One-Shot Learning pour garantir la protection des données utilisateurs ;
- **Prédition et diagnostic :** d'intégrer des modèles de vision par ordinateur pour l'identification des cultures et la détection de maladies ;
- **Aide à la décision :** de proposer un module de prédiction de tendance boursière et un assistant conversationnel basé sur RAG pour contextualiser les réponses.

0.2 Contributions du rapport

Ce rapport documente l'architecture, l'implémentation et l'évaluation du système. Il présente :

- une revue de l'état de l'art et des choix techniques justifiés ;

— la description de la conception logicielle, y compris les diagrammes UML et l'architecture micro-services ;
— un protocole expérimental complet, les jeux de données utilisés, les métriques d'évaluation et les résultats préliminaire ;
— des annexes techniques contenant extraits de code, paramètres d'entraînement et guide d'installation.

Ce rapport de Projet de Fin d'Année décrit le processus complet de conception et de réalisation d'AgroSense. Nous commencerons par présenter le cadre du projet et l'étude de l'existant dans le premier chapitre. Le deuxième chapitre détaillera les spécifications des besoins fonctionnels et techniques. Le troisième chapitre sera consacré à la conception du système, incluant les diagrammes UML et l'architecture détaillée, avec un focus majeur sur le module de reconnaissance faciale. Enfin, le quatrième chapitre présentera la réalisation technique, les interfaces de l'application et les résultats obtenus.

Chapitre 1

Présentation du cadre de projet

1.1 Introduction

Ce chapitre a pour objectif de situer le projet **AgroSense** dans son contexte global. Nous allons explorer l'environnement dans lequel s'inscrit l'application, analyser les solutions existantes pour en dégager les limites, et justifier les choix technologiques et méthodologiques qui ont guidé notre développement. Nous présenterons également la planification suivie pour mener à bien ce travail.

1.2 Étude de l'existant

1.2.1 Description de l'existant

Le marché actuel propose de nombreuses applications dédiées soit à l'agriculture (détection de maladies, gestion de ferme), soit à la finance (suivi boursier, prédictions).

FIGURE 1.1 – Logo de PlantSnap

— **Applications agricoles :** Des solutions comme *PlantSnap* ou *Plantix* permettent d'identifier des plantes et des maladies via l'image. Elles utilisent souvent des modèles de Deep Learning classiques (CNN).
— **Applications financières :** Des plateformes comme *Yahoo Finance* ou *Bloomberg* offrent des données en temps réel mais sont souvent complexes pour un utilisateur non expert.
— **Systèmes de sécurité :** L'authentification mobile repose souvent sur les solutions natives (FaceID, TouchID) ou des codes PIN, rarement sur des modèles de reconnaissance faciale propriétaires intégrés directement dans la logique applicative pour un contrôle total des données.

1.2.2 Critique de l'existant

Bien que performantes, ces solutions présentent plusieurs inconvénients majeurs dans le cadre de notre problématique :
— **Fragmentation :** L'utilisateur doit jongler entre plusieurs applications pour gérer ses investissements et ses activités agricoles.
— **Manque d'intégration IA avancée :** Peu d'applications combinent vision par ordinateur, traitement du langage naturel (RAG) et prédiction de séries temporelles (LSTM) dans une interface unifiée.
— **Sécurité générique :** La dépendance aux systèmes de sécurité des OS limite la personnalisation et le contrôle sur le processus d'authentification biométrique.

1.2.3 Solution proposée

AgroSense se propose de combler ces lacunes en offrant une plateforme unifiée et sécurisée. Notre solution intègre :

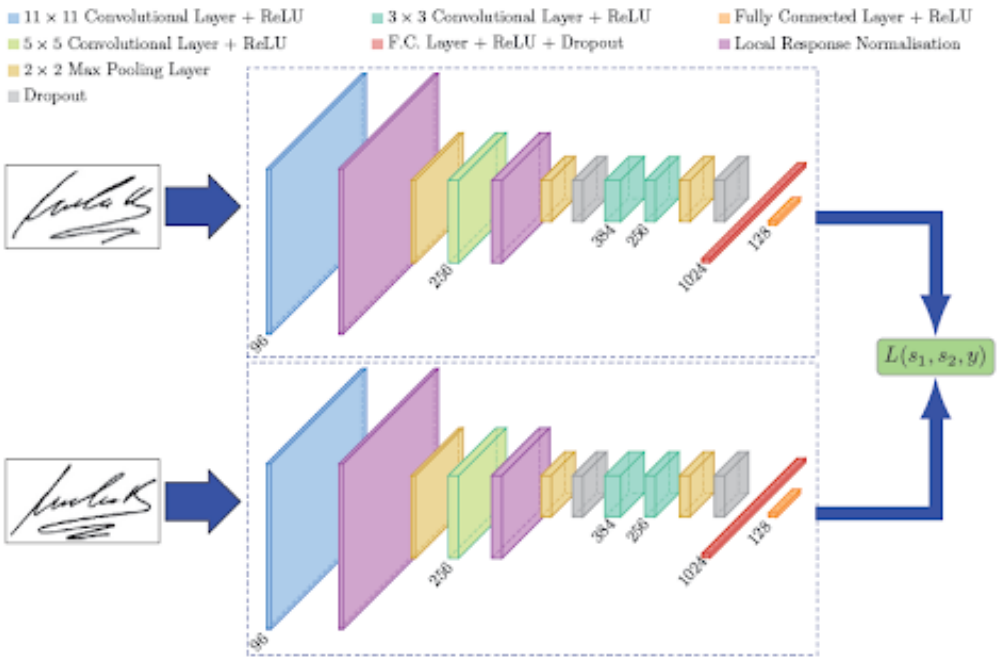

FIGURE 1.2 – Siamese Networks architecture

— **Sécurité par Reconnaissance Faciale (Siamese Networks) :** Un module de sécurité développé sur mesure, utilisant l'apprentissage "One-Shot" pour authentifier l'utilisateur avec une grande précision, même avec peu d'exemples. C'est la porte d'entrée obligatoire de l'application.
— **Prédiction Agricole (ANN CNN) :** Identification des fruits/légumes et diagnostic.
— **Prédiction Boursière (LSTM) :** Analyse des tendances du marché (Stock Market).
— **Assistance Intelligente (RAG Voice) :** Un chatbot contextuel capable de répondre aux questions spécifiques sur l'agriculture et la finance.

1.3 Choix de modèle de développement

Pour ce projet, nous avons opté pour une méthodologie **Agile (Scrum)**. Ce choix se justifie par la nature exploratoire de l'intégration de multiples modèles d'IA (CNN, LSTM, Siamese). Cela nous a permis de procéder par itérations : d'abord le développement du modèle de reconnaissance faciale (le cœur de la sécurité), puis l'intégration progressive des autres modules (Fruits, Bourse, RAG).

1.4 Planning prévisionnel

Le projet s'est déroulé sur la durée du PFA, structuré comme suit :

1. **Phase 1 : Recherche et État de l'art** - Étude des architectures de réseaux de neurones (Siamese, CNN, LSTM).
2. **Phase 2 : Développement du module de sécurité** - Collecte de données, entraînement du modèle Siamois, tests de validation.
3. **Phase 3 : Développement des modules métier** - Entraînement des modèles Fruits (CNN) et Bourse (LSTM).
4. **Phase 4 : Développement Mobile (Flutter)** - Création des interfaces et intégration des modèles via API (Flask/FastAPI) ou TFLite.
5. **Phase 5 : Tests et Intégration** - Vérification globale et rédaction du rapport.

1.5 Conclusion

AgroSense ne se veut pas seulement une démonstration technique, mais une solution viable répondant à des besoins réels de convergence technologique. Le socle de ce projet est sa sécurité, garantie par notre propre implémentation de reconnaissance faciale, que nous détaillerons dans les chapitres suivants.

1.5.1 Revue de littérature

La littérature sur l'application de l'IA à l'agriculture a connu un développement important : classification d'espèces et d'états sanitaires via CNN, segmentation pour la détection de taches foliaires, et systèmes hybrides couplant images et métadonnées (satellite,

capteurs IoT). Les approches récentes utilisent des architectures prétrainées (ResNet, EfficientNet) adaptées par transfert d'apprentissage pour compenser la faiblesse de datasets spécifiques.

Par ailleurs, le domaine du few-shot et one-shot learning propose des alternatives pertinentes pour l'authentification biométrique lorsque les exemples par individu sont limités : siameses, matching networks, et prototypical networks sont des solutions couramment citées dans la littérature.

1.5.2 Enjeux réglementaires et éthiques

L'usage de données biométriques impose des contraintes juridiques et d'éthique : consentement explicite des utilisateurs, minimisation des données stockées, chiffrement au repos et en transit, et possibilité de suppression des données personnelles (droit à l'oubli). Ces obligations, en particulier dans un contexte international, doivent guider la conception technique et les politiques de déploiement.

Chapitre 2

Spécification des besoins

2.1 Introduction

Ce chapitre détaille les besoins fonctionnels et non fonctionnels de l'application Agro-Sense. Il définit le périmètre du projet et décrit les interactions entre les utilisateurs et le système à travers des diagrammes de cas d'utilisation.

2.2 Spécification des besoins fonctionnels

2.2.1 Besoin fonctionnel [Sécurité - Reconnaissance Faciale]

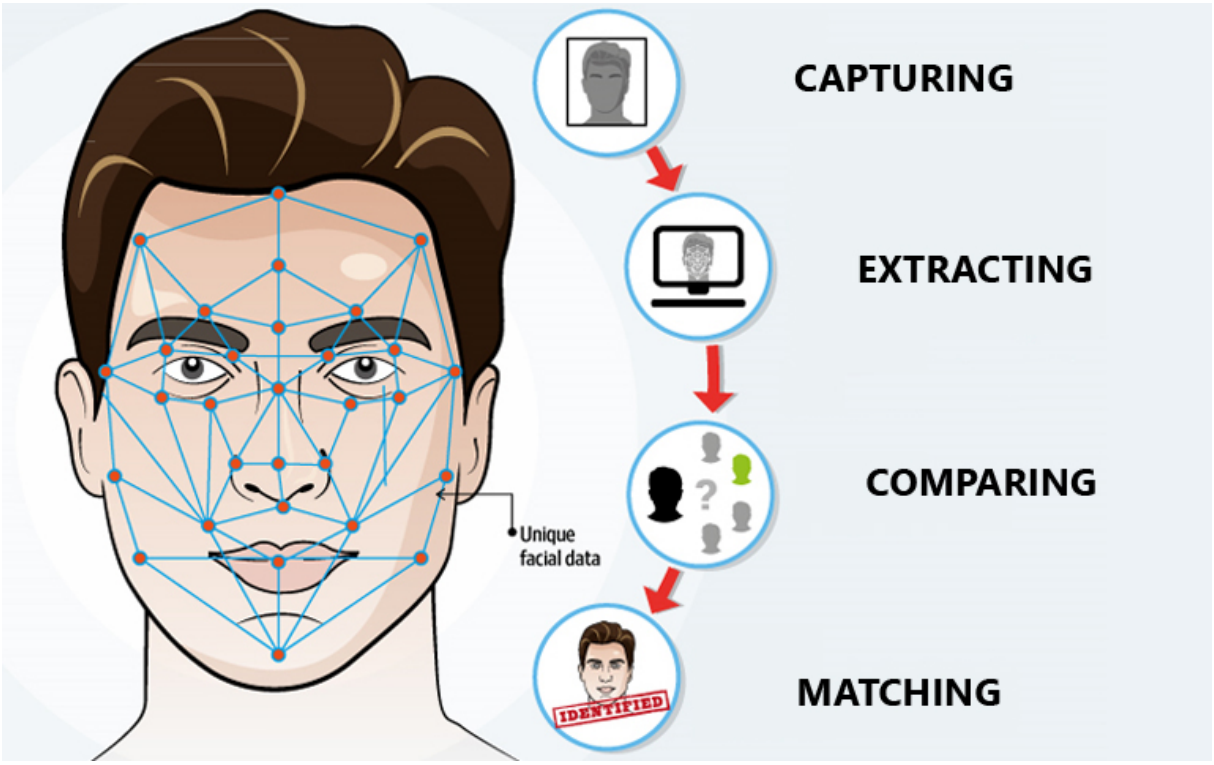

FIGURE 2.1 – Reconnaissance Facial

Le système doit garantir que seul l'utilisateur propriétaire peut accéder aux fonctionnalités de l'application.

— **[Enregistrement]** : L'utilisateur doit pouvoir enregistrer son visage (images "Anchor") lors de la première utilisation.
— **[Authentification]** : L'utilisateur doit pouvoir se connecter en scannant son visage. Le système compare l'image capturée avec les images enregistrées (Vérification One-Shot).

2.2.2 Besoin fonctionnel [Agriculture Intelligente]

— **[Prédiction Fruits/Légumes]** : L'utilisateur peut prendre une photo d'un fruit ou d'un légume ou d'une feuille. Le système doit identifier l'objet et/ou détecter d'éventuelles maladies (Classification via CNN/ANN).
— **[Conseil]** : Le système doit fournir des recommandations basées sur le diagnostic.

2.2.3 Besoin fonctionnel [Finance Marché]

— **[Prédiction Boursière]** : L'utilisateur accède aux prévisions des cours de la bourse (Stock Market). Le système utilise un modèle LSTM pour projeter les tendances futures.

2.2.4 Besoin fonctionnel [Assistance]

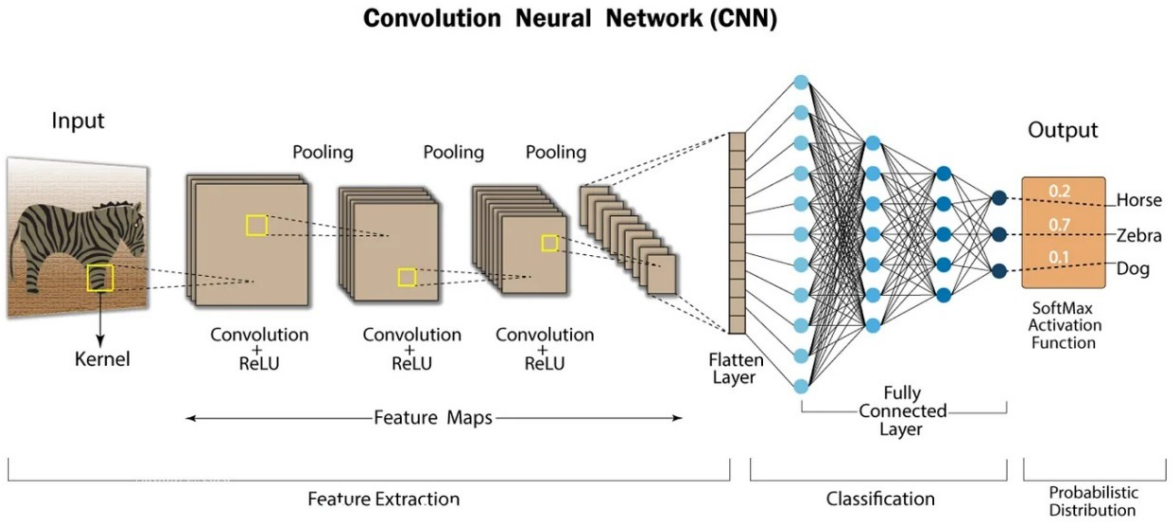

FIGURE 2.2 – Reconnaissance Facial

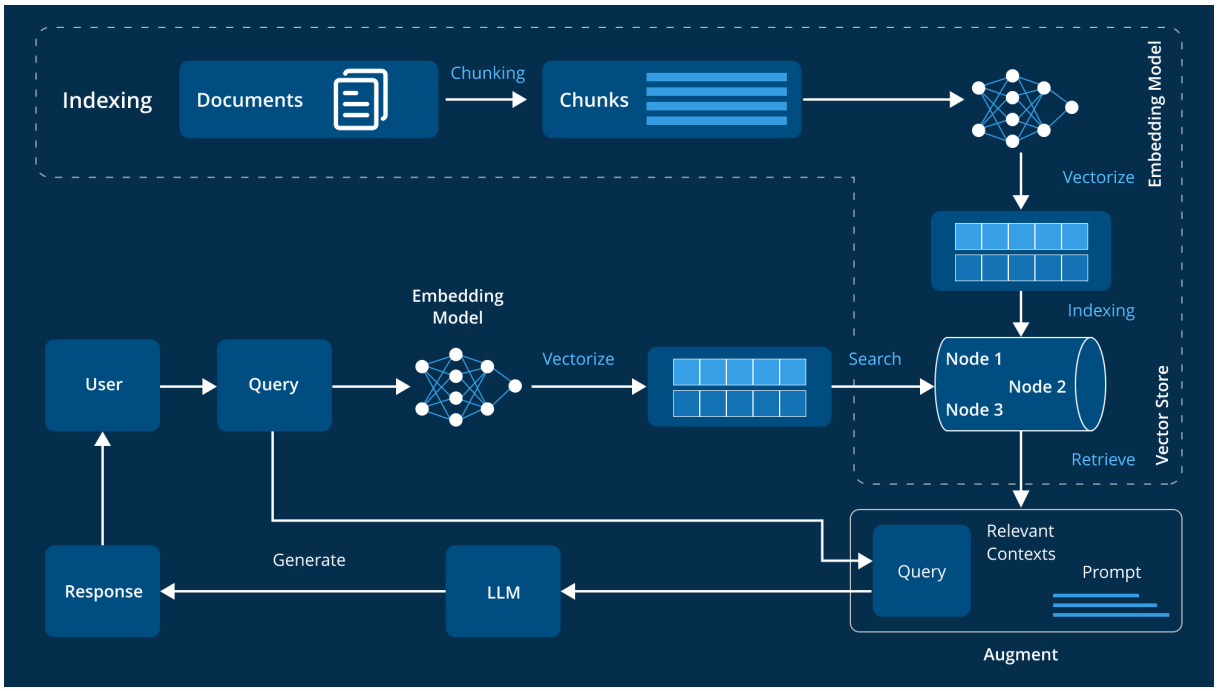

FIGURE 2.3 – architecture RAG

— **[Assistant Vocal]** : L'utilisateur peut interagir avec l'application par la voix.
— **[RAG - Chatbot]** : Un système de questions-réponses contextuel permet d'obtenir des informations précises sur l'agriculture et les finances.

2.3 Spécification des besoins non fonctionnels

— **Performance :** La reconnaissance faciale doit s'effectuer en moins de 2 secondes.
— **Fiabilité :** Le taux de faux positifs pour l'authentification doit être minime.
— **Ergonomie :** L'interface doit être intuitive (Flutter) et responsive.
— **Sécurité :** Les données biométriques ne doivent pas être compromises.

FIGURE 2.4 – flutter logo

— **Disponibilité :** Les services critiques (authentification) doivent rester disponibles pendant les heures d'utilisation ; une stratégie de basculement devra être déployée pour le backend.
— **Confidentialité :** Les embeddings faciaux sont stockés chiffrés et soumis à des politiques d'accès strictes ; les flux API utilisent TLS 1.2+.
— **Maintenabilité :** Le code doit étre modulable et testable ; fournir des tests unitaires et d'intégration pour les composants critiques.

2.4 Présentation des cas d'utilisation

2.4.1 Présentation des acteurs

— **Utilisateur Final (Agriculteur/Investisseur)** : Personne utilisant l'application pour gérer ses cultures ou consulter la bourse. Il doit s'authentifier au préalable.

2.4.2 Description des cas d'utilisation

1. **S'authentifier :** L'acteur présente son visage à la caméra. Le système valide ou refuse l'accès.
2. **Consulter Prédictions :** L'acteur demande une analyse (photo ou données).
3. **Discuter avec l'Assistant :** L'acteur pose une question vocale ou textuelle.

2.4.3 Détails techniques des données

Le système manipule plusieurs types de données : images (JPEG/PNG) pour la vision, séries temporelles (CSV/JSON) pour la finance, et documents texte pour le module RAG. Pour chaque image, un métadonnée standard contient le timestamp, la localisation approximative (si fournie), et l'ID de l'utilisateur. Les embeddings faciaux sont serialisés en vecteurs flottants (float32) puis chiffrés avant stockage.

Chapitre 3

Conception du système

3.1 Introduction

Ce chapitre est le cœur technique de notre rapport. Nous y détaillons l'architecture du système, avec une focalisation majeure (80%) sur le module de reconnaissance faciale, véritable clé de voûte de la sécurité d'AgroSense. Nous présenterons également les modélisations statiques et dynamiques via UML.

3.2 Architecture de la Reconnaissance Faciale (Siamese Networks)

Contrairement aux systèmes de classification classiques, notre module d'authentification utilise un **Réseau de Neurones Siamois (Siamese Neural Network)**.

3.2.1 Principe de fonctionnement

Le réseau siamois est constitué de deux réseaux de neurones identiques (mêmes poids) qui traitent deux entrées distinctes en parallèle :

1. **Image d'entrée (Input Image)** : Le visage capturé par la caméra lors de la connexion.
2. **Image de vérification (Verification Image)** : Une image stockée lors de l'enregistrement (Anchor).

Le but n'est pas de classer l'image, mais de calculer une **distance** (similitude) entre les deux vecteurs de caractéristiques (embeddings) produits par les réseaux.

3.2.2 Fonctions de perte et stratégie d'entraînement

Pour l'entraînement du réseau siamois, plusieurs fonctions de perte peuvent être utilisées : la *contrastive loss* (minimisation de la distance pour les paires positives et maximisation pour les négatives) et la *triplet loss* (ancre, positive, negative). Le choix se fait en fonction de la qualité du dataset et de la difficulté à miner des négatives pertinents. Une politique d'augmentation (rotations, variations d'éclairage, zoom) est appliquée pour améliorer la robustesse aux variations d'acquisition.

3.2.3 Prétraitement et augmentation

Le pipeline prévoit : detection du visage (OpenCV/Haar ou MTCNN), recadrage centré sur les yeux, redimensionnement à 100x100, normalisation des canaux, et augmentation aléatoire (flip horizontal, variations gamma, blur léger). Ces opérations réduisent le risque de sur-apprentissage et améliorent général la généralisation du modèle.

3.2.4 Déploiement et micro-services

Le module d'authentification est déployé comme un micro-service (Flask/FastAPI) exposant des endpoints REST. Cette décision facilite la mise à jour du modèle sans déployer l'application mobile et permet l'extensibilité par conteneurisation (Docker) et orchestration (Kubernetes) si nécessaire.

3.2.5 Architecture détaillée du modèle

Le modèle a été implémenté avec TensorFlow/Keras.

— **Entrée :** Images RGB redimensionnées à 100×100 pixels.

— **Feature Extractor (CNN) :** Une série de couches de convolution (Conv2D) et de Pooling (MaxPooling2D) pour extraire les caractéristiques faciales.

— **Couche de Distance (L1Dist) :** Une couche personnalisée qui calcule la valeur absolue de la différence entre les deux embeddings :

$$L1(x, y) = |f(x) - f(y)|$$

où $f(x)$ est l'embedding généré par le CNN.

— **Sortie :** Une couche Dense avec activation Sigmoid qui produit un score entre 0 (différent) et 1 (identique).

3.2.6 Stratégie d'authentification (One-Shot Learning)

L'application utilise une approche "One-Shot Learning".

1. L'utilisateur capture une image "Anchor".
2. Lors de l'authentification, la nouvelle image est comparée à une série d'images de vérification.
3. Si le score de similarité (Cosine Similarity ou prédiction directe) dépasse un seuil (« Threshold », fixé à 0.75 ou 0.3 selon la méthode), l'accès est autorisé via l'API Flask (`/verify_face`).

3.3 Modélisation dynamique (UML)

3.3.1 Diagrammes de séquences : Authentification

L'acteur interagit avec l'interface Flutter, qui communique avec le service Python.

1. **Utilisateur** ouvre l'application.
2. **Flutter App** active la caméra et capture le visage.
3. **FaceAuthScreen** envoie l'image (Base64) à l'API `face_recognition_service.py`.

4. **API** pré-traite l'image (Resize 100x100, Normalization).
5. **API** charge le modèle `siamesemodelv2.h5`.
6. **API** compare l'image avec le dossier `verification_images`.
7. **API** retourne `verified: true/false`.
8. **Flutter App** redirige vers `HomeScreen` ou affiche une erreur.

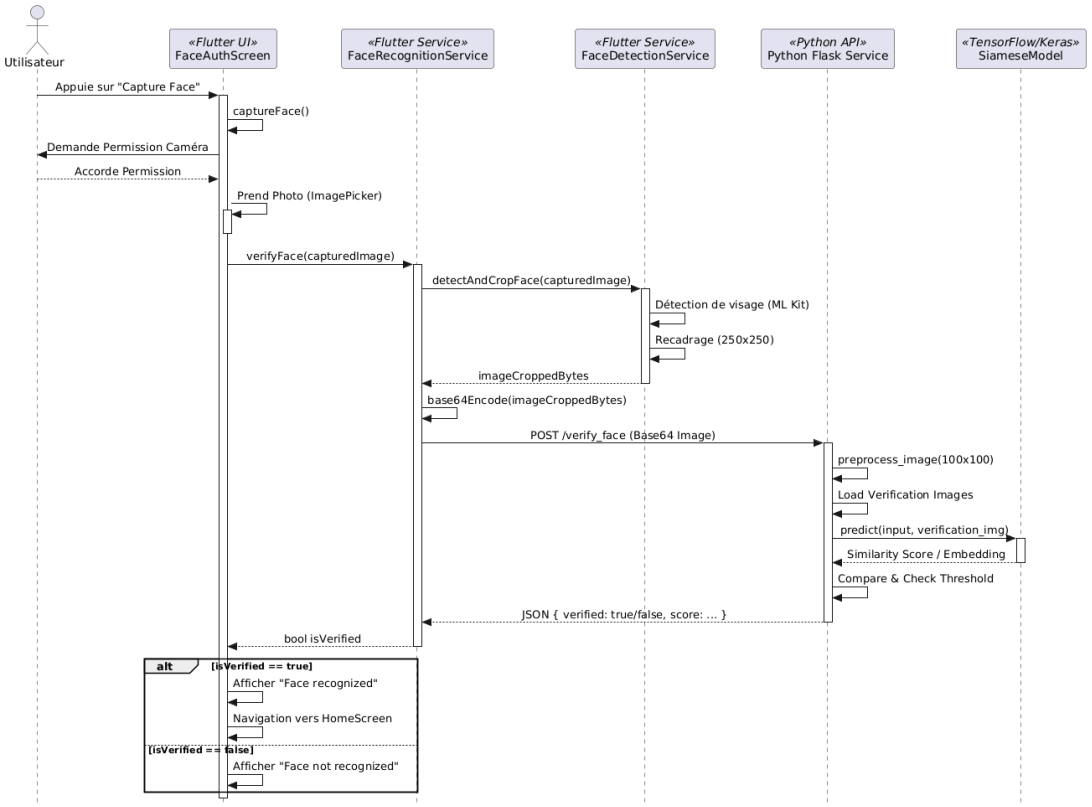

FIGURE 3.1 – Diagramme de séquence de l'authentification faciale

3.4 Modélisation statique (UML)

3.4.1 Diagramme de classes

L'architecture de l'application Flutter repose sur une séparation claire entre les écrans (UI) et les services (Logique).

Packages principaux :

— `screens` : Contient les vues (`FaceAuthScreen`, `HomeScreen`, `AnnScreen`, etc.).

— `services` : Gère les appels API et la logique (`FaceRecognitionService`, `AnnService`, `LstmService`).

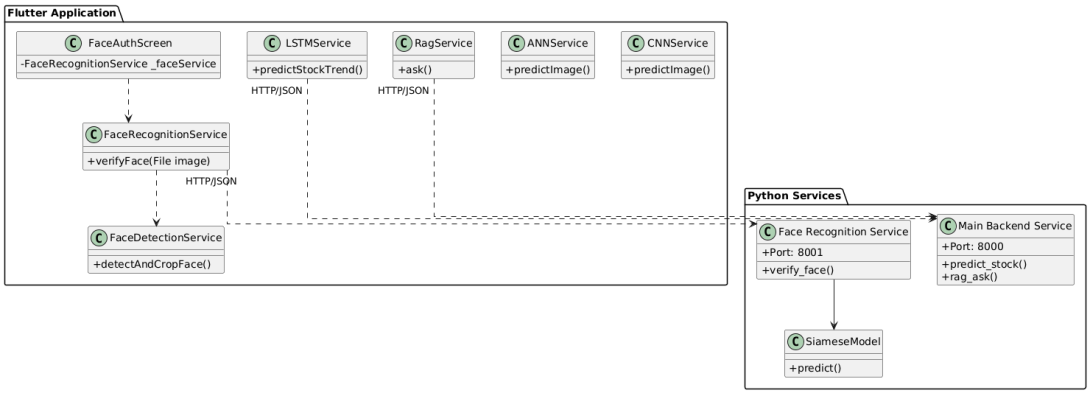

FIGURE 3.2 – Diagramme de classes simplifié (Flutter & Python Service)

3.5 Conclusion

La conception met l'accent sur la modularité. Le modèle de sécurité est découplé (micro-service Flask), ce qui permet de le mettre à jour sans recompiler l'application mobile.

Chapitre 4

Réalisation du système

4.1 Introduction

Ce chapitre présente les outils et technologies utilisés pour le développement d'Agro-Sense, ainsi que les résultats finaux sous forme de captures d'écran de l'application.

4.2 Environnement de développement

4.2.1 Environnement matériel

Le développement et l'entraînement des modèles ont été réalisés sur une machine équipée de GPU pour accélérer les calculs TensorFlow (compatible CUDA). Les tests mobiles ont été effectués sur des émulateurs Android (Pixel/Medium Phone).

4.2.2 Environnement logiciel

- **Frontend Mobile :** Flutter (Dart). Framework UI de Google pour le multiplateforme.
- **Backend IA :** Python 3.9+.
  - **TensorFlow/Keras :** Pour la création et l'entraînement des modèles (Siamese, CNN, LSTM).
  - **Flask :** Pour exposer les modèles sous forme d'API REST.
  - **OpenCV :** Pour le traitement d'images (capture, redimensionnement).
- **Outils :** Visual Studio Code, Android Studio, Git.

4.2.3 Configuration d'entraînement

Les scripts d'entraînement ont été implémentés en Python avec TensorFlow/Keras. Pour chaque modèle nous définissons un ensemble d'hyperparamètres : optimiseur (Adam), taux d'apprentissage initial (par exemple 1e-4), taille de batch (16–64 selon la mémoire GPU), et politique d'arrêt précoce (early stopping) sur la perte de validation. Le suivi se fait avec TensorBoard pour visualiser pertes et métriques.

4.2.4 Protocole d'évaluation

L'évaluation combine des mesures classiques : accuracy, precision, recall, F1 pour les modules de classification ; AUC ou taux de faux positifs/acceptation pour l'authentification faciale. Chaque modèle est testé sur un jeu tenu à l'écart lors de l'entraînement et des tests d'ablations sont réalisés pour mesurer l'impact des augmentations et choix d'architecture.

4.3 Implémentation de la Reconnaissance Faciale

Le service de reconnaissance faciale (`face_recognition_service.py`) expose un endpoint `/verify_face`. Voici un extrait du code Python gérant la comparaison :

Listing 4.1 – Comparaison directe avec le modele Siamois

```
def verify_face():
    # ... (reception image)
    # Chargement du modele avec la couche personnalisee L1Dist
    model = keras.models.load_model('siamesemodelv2.h5',
                                    custom_objects={"L1Dist": L1Dist})

    # Comparaison avec les images de verification
    for img_file in verification_files:
        score = model.predict([input_image, verification_image])
        if score > threshold:
            verified = True
    # ...
```

4.4 Principales interfaces graphiques

4.4.1 Page d'Authentification

L'application s'ouvre sur un écran de bienvenue épuré ("Welcome Screen"), mettant en avant la sécurité biométrique. L'utilisateur est invité à s'authentifier via la reconnaissance faciale pour accéder aux fonctionnalités.

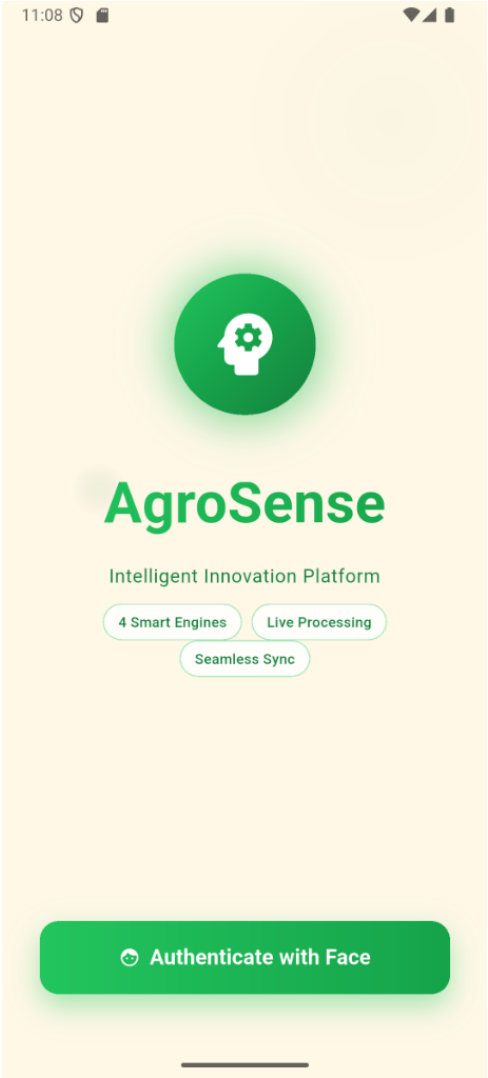

FIGURE 4.1 – Écran d'accueil et Authentification

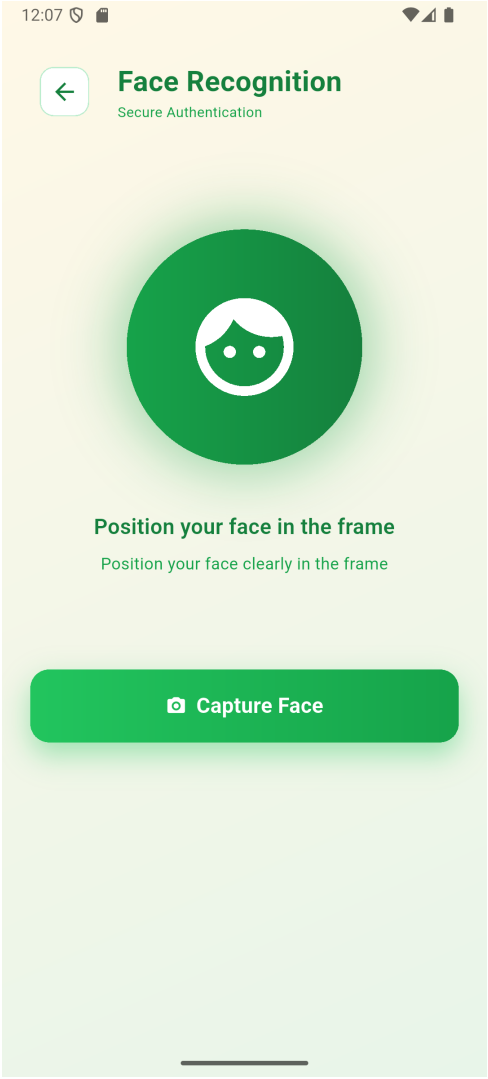

FIGURE 4.2 – Scan facial en temps réel

4.4.2 Tableau de bord et Menu Principal

Une fois authentifié, l'utilisateur accède au tableau de bord (*Home Screen*) qui centralise l'accès aux trois moteurs d'intelligence :

— **Neural Processing :** Pour les modèles prédictifs (ANN/LSTM).
— **Visual Intelligence :** Pour l'analyse d'images (CNN).
— **Adaptive Intelligence :** Pour le système RAG et le Chatbot.

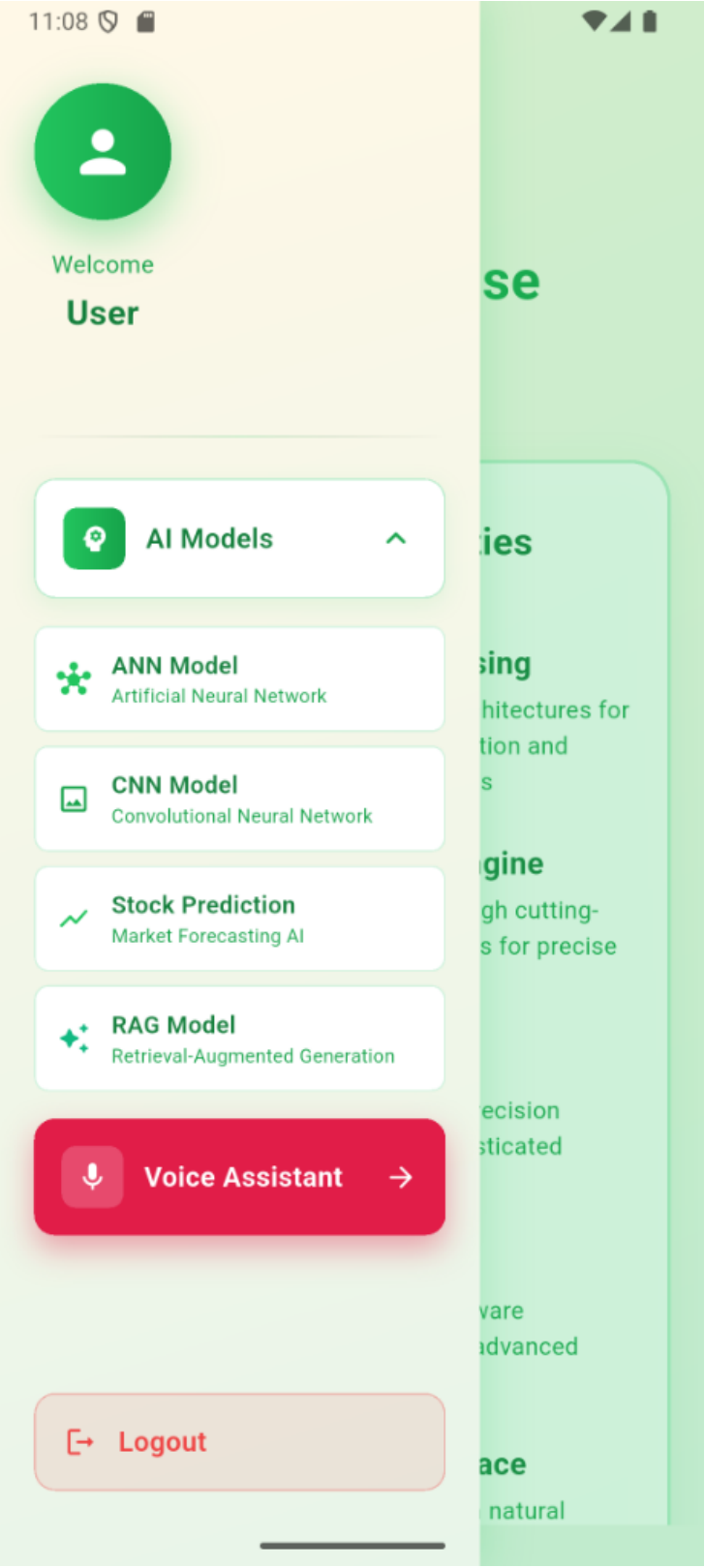

FIGURE 4.3 – Menu des fonctionnalités (Explore Capabilities)

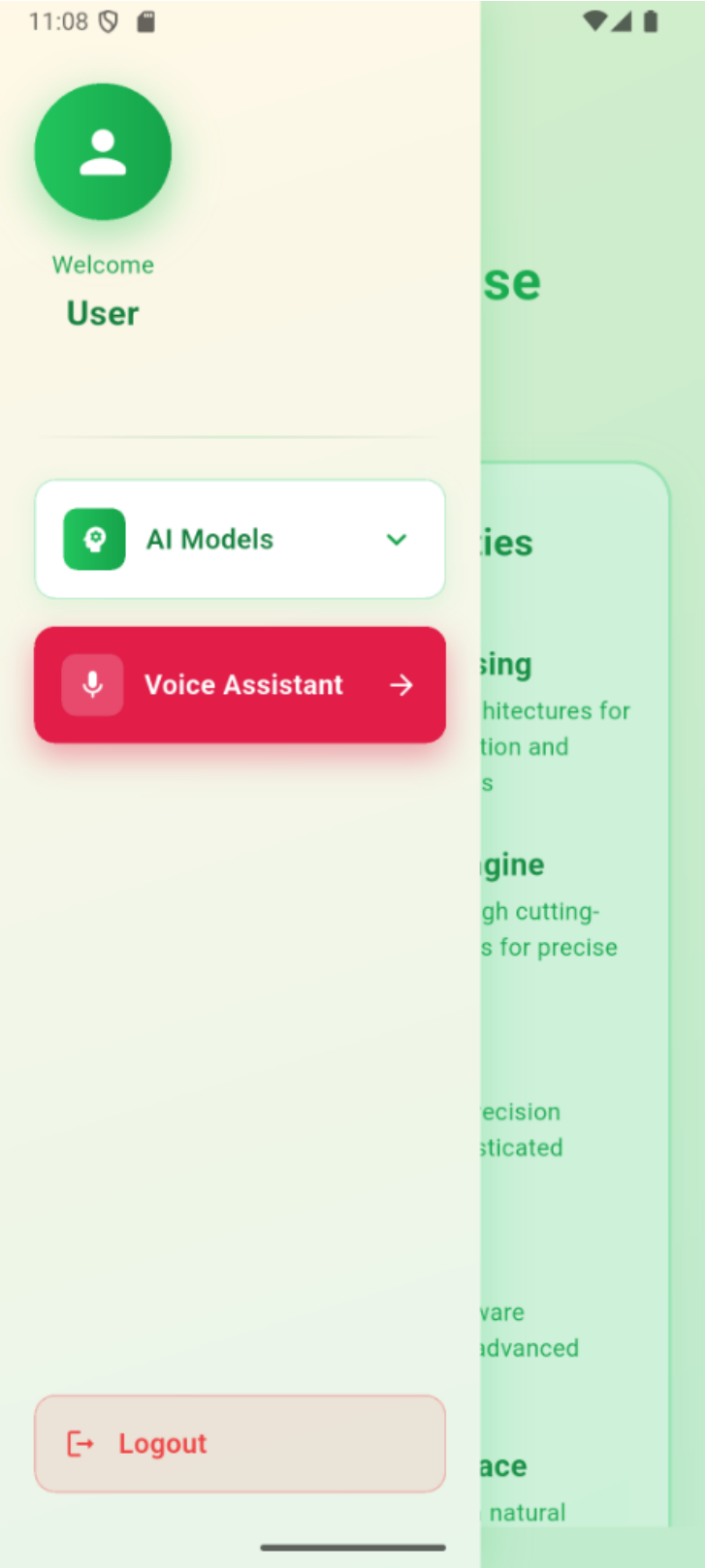

FIGURE 4.4 – Sélection des modèles

4.4.3 Interfaces des Modèles de Vision (ANN & CNN)

Les modules de vision par ordinateur permettent au système d'identifier des maladies ou des types de cultures à partir d'images.

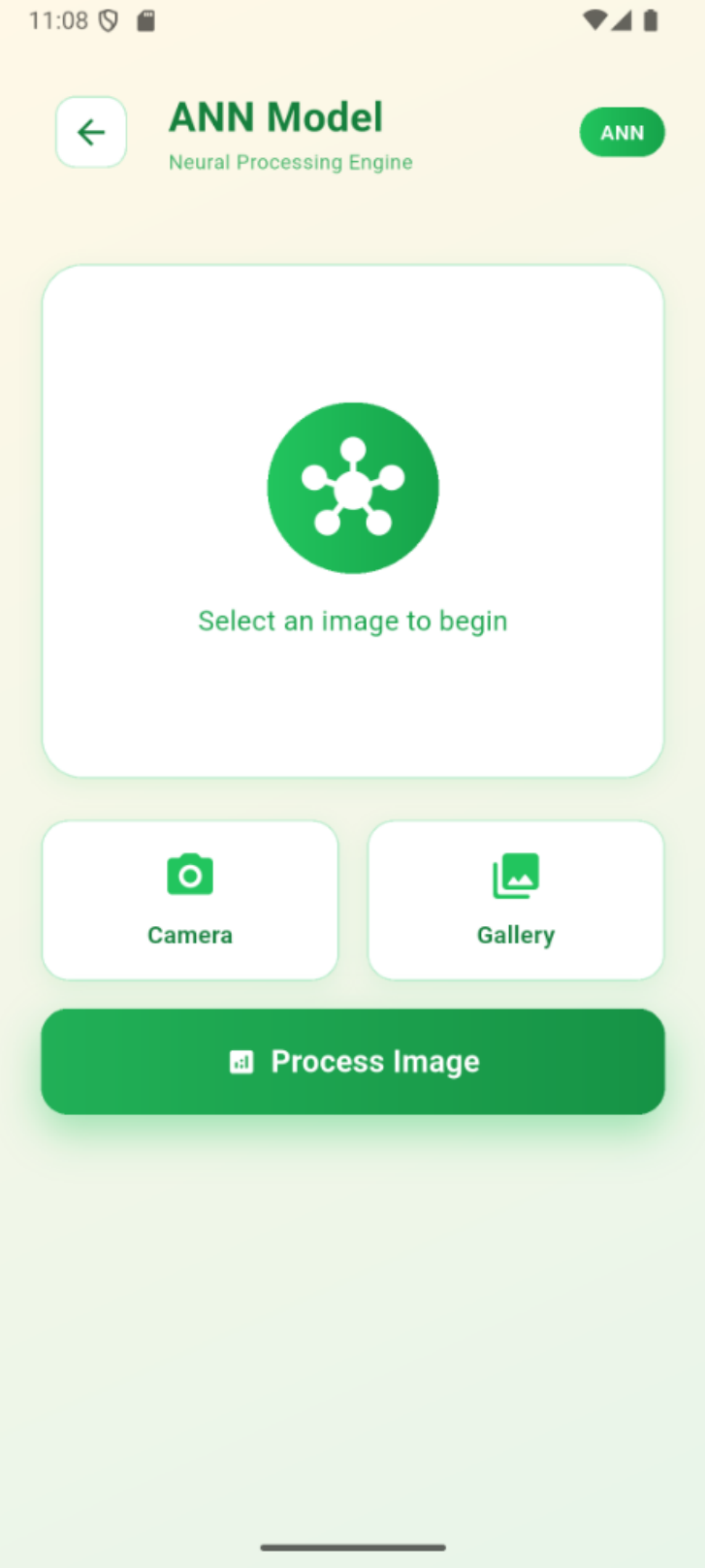

FIGURE 4.5 – Interface ANN Model

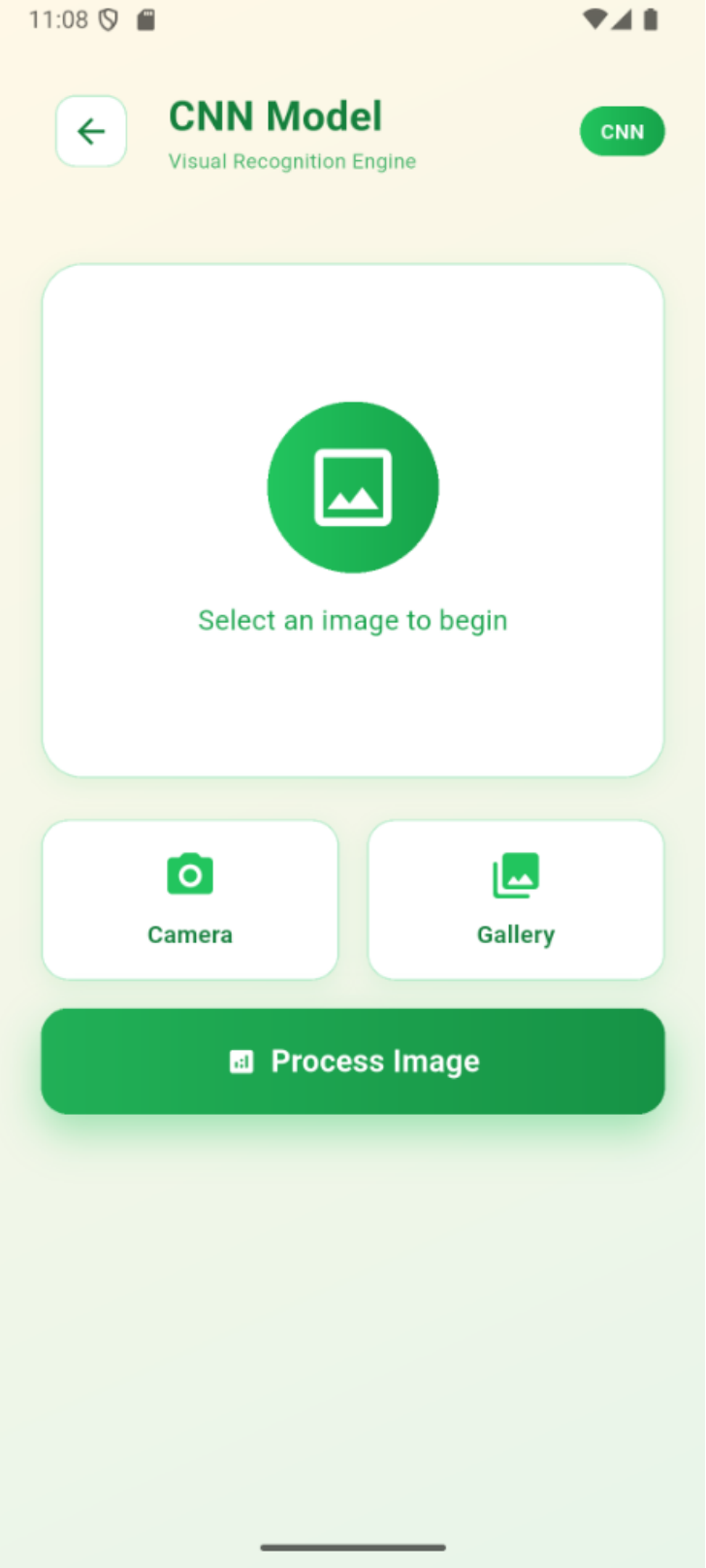

FIGURE 4.6 – Interface CNN Model

4.4.4 Modules de Prédiction Avancée (Finance & RAG)

Au-delà de l'analyse d'image, AgroSense intègre des outils d'aide à la décision financière et contextuelle.

Prédiction Boursière (LSTM)

L'interface de **Stock Prediction** permet de visualiser l'historique des prix et de générer une prévision aléatoire ou basée sur un dataset réel. Le modèle LSTM (Long Short-Term Memory) analyse la séquence temporelle pour projeter les tendances futures.

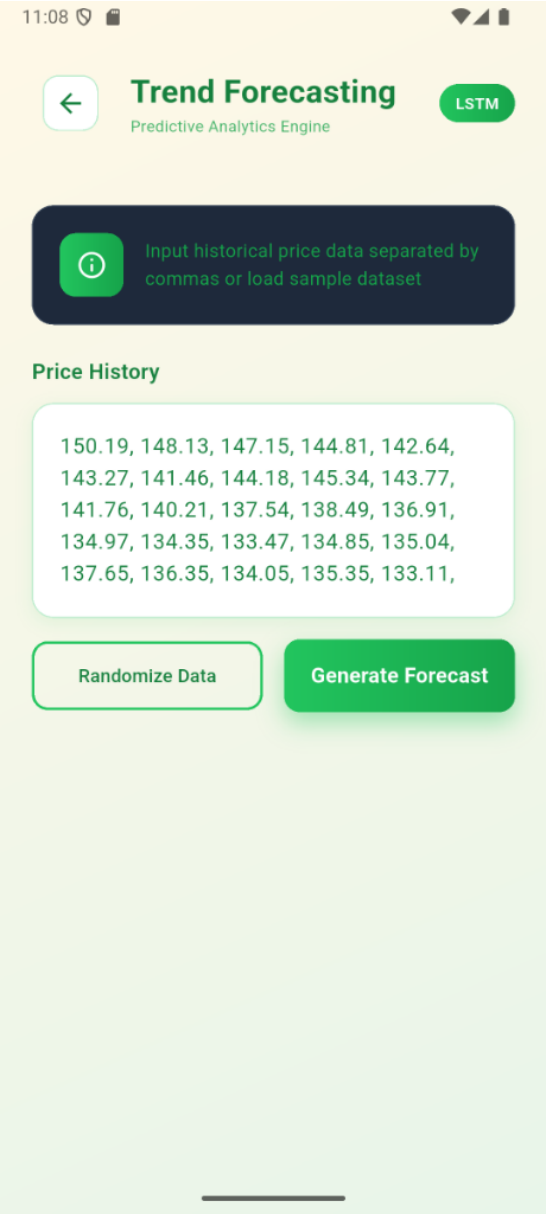

FIGURE 4.7 – Module de prédiction de tendance (Trend Forecasting)

Système RAG (Chatbot Documentaire)

Le module **RAG Model** (Retrieval-Augmented Generation) permet à l'utilisateur de charger des documents (rapports, fiches techniques) et de poser des questions en langage

naturel. Le système combine la recherche d'information dans le document et la capacité de génération d'un LLM pour fournir une réponse précise.

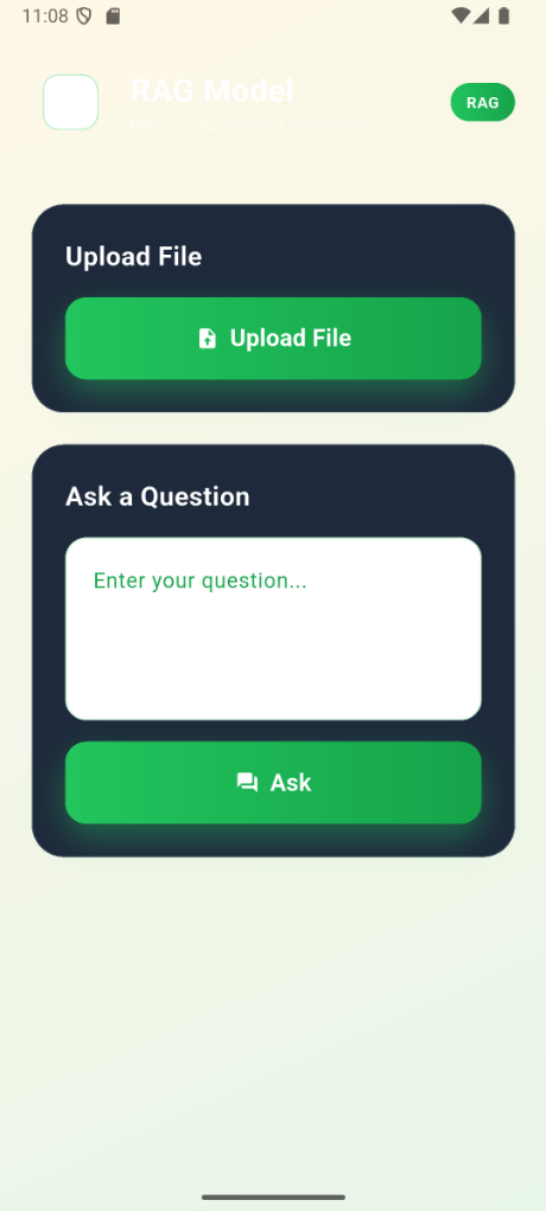

FIGURE 4.8 – Interface RAG : Upload de fichiers et Q&A

4.5 Conclusion

L'application est fonctionnelle et intègre avec succès les différents modèles d'IA. La communication entre Flutter et le backend Python est fluide, garantissant une expérience utilisateur réactive.

Conclusion Générale

Le projet **AgroSense** nous a permis de concrétiser une vision ambitieuse : réunir l'agriculture de précision et l'analyse financière dans une seule application mobile sécurisée. Au terme de ce travail, nous avons réussi à développer une solution complète qui répond aux besoins des agriculteurs et des investisseurs modernes.

L'aspect le plus critique et le plus innovant de ce projet a été la mise en œuvre de la **reconnaissance faciale via des réseaux de neurones siamois**. En développant cette brique de sécurité "from scratch", nous avons acquis une maîtrise approfondie des mécanismes d'apprentissage profond (Deep Learning) et des défis liés à l'authentification biométrique (One-Shot Learning, gestion des seuils de similarité). Ce choix architectural garantit une indépendance technologique et une protection accrue des données utilisateurs.

Au-delà de la sécurité, l'intégration de modèles prédictifs pour les maladies des plantes (CNN) et pour le marché boursier (LSTM), ainsi que l'ajout d'un assistant intelligent (RAG), font d'AgroSense un véritable assistant personnel polyvalent.

Ce Projet de Fin d'Année a été une opportunité inestimable pour monter en compétence sur le framework Flutter et l'écosystème Python/TensorFlow. Il ouvre la voie à de futures améliorations, telles que le déploiement des modèles directement sur le mobile (Edge AI) pour réduire la dépendance à la connexion internet, ou l'élargissement de la base de données de maladies agricoles.

Limites et perspectives

Malgré les avancées, le projet présente des limites : la taille et la diversité des jeux de données influencent fortement la robustesse des modèles ; l'infrastucture de déploiement n'est pas encore optimisée pour l'inférence en edge ; enfin, des tests utilisateurs réels restent nécessaires pour valider l'ergonomie et l'acceptation sociale.

Les perspectives incluent : l'optimisation des modèles (quantization, pruning), la mise en place d'une pipeline CI/CD pour déploiement continu, et des campagnes d'évaluation utilisateur sur le terrain.

Bibliographie

Annexes

Annexe A : Architecture du Réseau Siamois

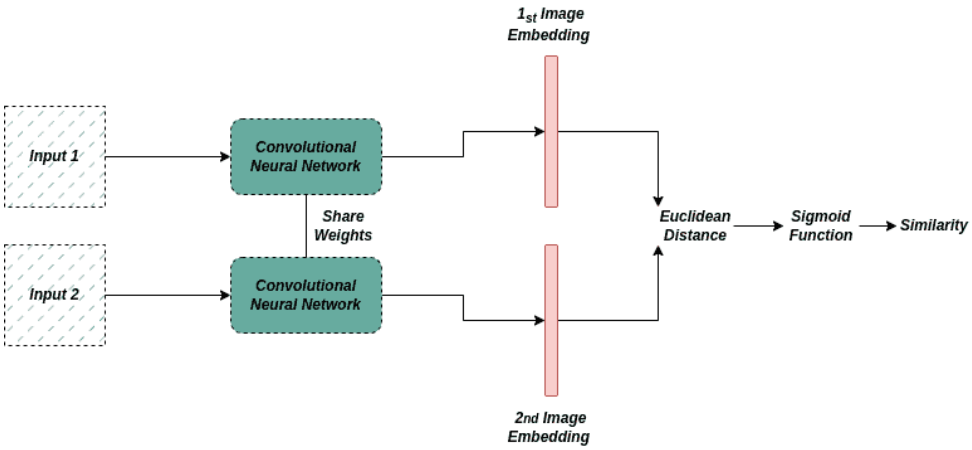

FIGURE 4.9 – Architecture détaillée (CNN + L1 Distance)

Annexe B : Extrait du Dataset

Exemples d'images "Anchor", "Positive", et "Negative" utilisées pour l'entraînement.

Annexe C : Structure du Projet Flutter

Arborescence des fichiers du projet mobile AgroSense.

```
lib/
  screens/
    face_auth_screen.dart
    home_screen.dart
  services/
    face_recognition_service.dart
  widgets/
    ...
```

Annexe D : Extrait de script d'entraînement

Exemple simplifié d'une boucle d'entraînement pour le réseau siamois (TensorFlow/-Keras).

Listing 4.2 – train$_s$*iamese.py*

```
import tensorflow as tf

# model, dataset, preprocessing assumed defined
optimizer = tf.keras.optimizers.Adam(1e-4)
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['ac

callbacks = [tf.keras.callbacks.EarlyStopping(patience=8, restore_best_weig

history = model.fit(train_dataset, validation_data=val_dataset, epochs=100,

model.save('siamesemodelv2.h5')
```