Cahier des charges

Groupe :

- Rayane Tabouri
- Ariles Tahraoui
- Aziz Djouder
- Akram Bouhraoua

1. Introduction

○ Contexte du projet :

Face à un quotidien souvent surchargé, il est devenu indispensable d'optimiser la gestion des repas et des stocks alimentaires afin de réduire le gaspillage et de faciliter la planification. Le projet consiste à développer une application mobile de gestion des repas et des stocks d'ingrédients.

○ Objectifs du projet :

- Permettre aux utilisateurs d'enregistrer, suivre et mettre à jour les ingrédients disponibles.
- Recommander des recettes adaptées aux ingrédients en stock, en fonction des avis des autres utilisateurs.
- Générer automatiquement une liste de courses optimisée pour compléter le stock si besoin.
- Offrir une gestion dédiée de l'alimentation animale, incluant :
  - La saisie du programme nutritionnel.
  - Des alertes pour les horaires de repas.
  - Le suivi du stock de nourriture pour animaux.
- Fournir une interface mobile intuitive, sécurisée et compatible avec Android.

○ Périmètre du projet :

Fonctionnalités incluses :

- Planification des repas.
- Mise à jour des stocks.

- Mise à disposition de recettes qui prennent en compte le stock.
- Système de recommandation pour les recettes en fonction des avis des utilisateurs.
- Rappel des courses avec la liste des ingrédients nécessaires aux recettes planifiées.
- Abonnement pour avoir accès aux avantages.
- Recommandations de magasins partenaires.
- Pub contextuelle.

Limites :

- Pas d'intégration avec des services de commande en ligne ou de livraison dans la première version.
- Conçu pour un usage individuel, avec possibilité d'extension ultérieure à une gestion multi-utilisateur.
- Pas de gestion de contenu média (images, audio…).

2. Description des besoins

- **Besoins fonctionnels :**

Gestion des repas :

- Suivi des stocks.
- Suggestion de recettes.
- Génération de listes de courses.
- Alertes de courses et de repas.

Expérience utilisateur :

- Interface intuitive.
- Compatible Android.
- Historique des repas et alertes.
- Mode hors ligne.
- Abonnement.
- Recommandations de magasins.

- **Besoins non fonctionnels :**

Performance : Réactivité et fluidité.

Sécurité : Protection des données et authentification sécurisée.

Disponibilité : Compatible Android, sauvegarde des données.

Ergonomie : Interface simple et accessible.

Scalabilité : Facilité d'ajout de nouvelles fonctionnalités.

3. Spécifications techniques

- **Architecture technique :**

  Frontend : Application mobile native Android en Java.

  Backend : Serveur d'API développé en Python (FastAPI ou Django REST Framework) assurant la gestion des données.

  Base de données : Base relationnelle ou Firebase Firestore hébergée sur le cloud.

  Communication :

  - Échanges via des appels API REST en JSON.
  - Notifications locales sur l'appareil de l'utilisateur.
  - Authentification (OAuth 2.0 ou Firebase Authentication).

- **Technologies utilisées :**

  Langages : Java (Android studio), Python (Kivy, FastAPI ou Django REST Framework)

  Base de données : Base relationnelle (PostgreSQL) ou NoSQL (Firebase Firestore).

  Notifications et authentification :

  - Firebase Cloud Messaging pour les notifications push.
  - Firebase Authentication ou OAuth 2.0 pour la gestion des comptes utilisateurs.

- **Environnement de développement et déploiement :**

  IDE :Android studio, VS Code.

  Gestion de versions : GitHub.

Déploiement :

- *Application* : Google Play Store (Android).
- *Base de données* (PostgreSQL) : AWS RDS, Google Cloud SQL ou Heroku Postgres.
- *Backend* :
    - Hébergement sur une plateforme cloud (Heroku, AWS, Google Cloud Platform ou Azure).
    - Mise en place d'un pipeline CI/CD (GitHub Actions, Jenkins) pour automatiser les tests et les déploiements.

4. Livrables

- **Liste des livrables :**

Phase d'analyse et de conception :

- Cahier des charges finalisé.
- Documents de conception (diagrammes UML, maquettes, wireframes)
- Rapport d'analyse des besoins.

Phase de développement :

- Code source du frontend et du backend.
- Documentation technique (architecture, installation, configuration).
- Version alpha de l'application.

Phase de tests :

- Jeu de tests unitaires et d'intégration.
- Rapports de tests (comportement, performance, sécurité).
- Retour d'expérience des tests utilisateurs (version bêta).

Phase de déploiement :

- Version finale de l'application (disponible sur les stores ou en démonstration).
- Manuel utilisateur et guide de maintenance.
- Rapport de déploiement et de suivi post-lancement.

- **Critères de validation**

Documents et conception **:**

- Conformité aux besoins et spécifications du cahier des charges.
- Validation par le commanditaire et les parties prenantes.

Code source et développement :

- Passage réussi des tests unitaires et d'intégration.
- Respect des normes de sécurité et des performances attendues.

Tests et validation utilisateur :

- Absence de bugs critiques identifiés lors des tests.
- Retours positifs des utilisateurs lors des phases bêta et d'essais.

Déploiement final :

- Conformité aux spécifications fonctionnelles et non fonctionnelles.
- Validation par le déploiement sur les plateformes cibles (Google Play, ou environnement de démonstration).
- Documentation complète facilitant la prise en main et la maintenance.

5. Planification et gestion des risques

- **Phases du projet :**

Phase 1 : Analyse et conception

- Rédaction du cahier des charges
- Élaboration des maquettes, schémas UML et spécifications techniques
- Réunion de validation avec le commanditaire

Phase 2 : Développement

- Mise en place de l'architecture technique (frontend, backend, base de données).
- Développement des modules de gestion des repas, des ingrédients et de l'alimentation animale.
- Intégration des fonctionnalités de notifications et d'authentification.

Phase 3 : *Tests*

- Réalisation des tests unitaires et d'intégration.
- Tests de performance, de sécurité et d'ergonomie.
- Recueil des retours utilisateurs sur la version bêta.

Phase 4 : Déploiement et maintenance

- Déploiement de l'application sur les stores ou en environnement de démonstration.
- Formation des utilisateurs et mise à disposition de la documentation.
- Suivi post-déploiement et correction d' éventuels bugs.

- **Gestion des risques**

Identification des risques :

- Retards dans le développement ou la livraison des fonctionnalités.

- Problèmes techniques liés à l'intégration entre le frontend et le backend.
- Non-conformité aux exigences de sécurité et de performance.
- Retours négatifs des tests utilisateurs et manque d'ergonomie.
- Difficultés d'hébergement et de déploiement sur les plateformes cibles.

6. Budget et ressources

- **Estimation des coûts**

Développement logiciel :

- Licences logicielles et abonnements (IDE, outils de gestion de projet, services cloud) : 0€
- Hébergement backend et base de données : 0€

Tests et déploiement :

- Outils de tests automatisés et CI/CD : 0€
- Frais de publication sur les stores (Google Play) : 25€

Maintenance et support :

- Coût de maintenance corrective et évolutive post-déploiement : 0€

- **Ressources humaines et matérielles**

Ressources humaines :

- Développeurs : 1 à 2 développeurs mobiles (JavaScript ou Java pour Android).
- 1 développeur backend (Python).
- Designer UI/UX : Pour la conception de l'interface utilisateur et l'expérience globale.
- Testeurs : 1 ou 2 personnes pour réaliser les tests fonctionnels et d'intégration.
- 1 chef de projet : Pour coordonner les activités, gérer le planning et communiquer avec le commanditaire.

Ressources matérielles :

- Postes de travail : Ordinateurs performants adaptés au développement mobile et backend.
- Appareils de test : Smartphones et tablettes (Android) pour les tests réels.
- Environnement cloud : Accès aux services de cloud pour l'hébergement de l'API, de la base de données et la gestion des notifications.

7. Contraintes

- **Techniques :**

  Compatibilité multiplateforme : L'application doit fonctionner efficacement sur Android (et potentiellement iOS à l'avenir).

  Performance : Garantir des temps de réponse rapides et une navigation fluide, même en cas d'utilisation intensive, en mode en ligne et hors ligne.

  Sécurité : Protéger les données personnelles avec des mécanismes d'authentification sécurisés, de chiffrement et de sauvegarde.

  Interfaçage : Assurer une communication fiable entre le frontend et le backend, ainsi qu'une intégration aisée avec des services tiers (notifications, authentification).

- **Légales et réglementaires**

  Protection des données : Conformité au RGPD (ou autres réglementations locales) pour la collecte, le traitement et la conservation des données utilisateur.

  Propriété intellectuelle : Respect des droits d'auteur pour les contenus utilisés (recettes, images, etc.) et vérification des licences logicielles.
  gestion des notifications.
  Conformité aux stores : Respect des conditions d'acceptation et des normes imposées par les plateformes de publication (Google Play Store).

- **Budgétaires**

  Limitation des coûts : Respecter le budget alloué pour le développement, les outils, l'hébergement et la maintenance.

  Optimisation des ressources : Utiliser des solutions open source ou des abonnements économiques lorsque cela est possible, tout en garantissant la qualité et la sécurité du projet.

  Gestion des imprévus : Prévoir une marge budgétaire pour couvrir d'éventuelles extensions ou ajustements nécessaires en cours de projet.

8. Suivi et maintenance

- **Plan de suivi**

Suivi de projet :

- Utilisation d'outils de gestion de projet (Jira ou Notion) pour suivre l'avancement des tâches et le respect du planning.
- Réunions régulières (hebdomadaires ou bi-hebdomadaires) pour faire le point sur l'état d'avancement, les problèmes rencontrés et ajuster le planning si nécessaire.
- Mise en place d'un tableau de bord pour suivre les indicateurs clés (avancement, bugs identifiés, taux de résolution).

Suivi technique :

- Intégration d'un système de CI/CD pour automatiser les tests et déploiements.
- Surveillance continue de l'application via des outils d'analytics et de monitoring (Firebase Analytics, Google Analytics).
- Mise en place de rapports réguliers sur la performance et la sécurité de l'application.

- **Maintenance post-lancement**

Maintenance corrective :

- Correction rapide des bugs et incidents signalés par les utilisateurs via un système de ticketing.
- Mises à jour régulières pour assurer la compatibilité avec les nouvelles versions des systèmes d'exploitation (Android).

Maintenance évolutive :

- Intégration de nouvelles fonctionnalités en fonction des retours utilisateurs et des évolutions du marché.
- Optimisation continue des performances et de la sécurité de l'application.

Support utilisateur :

- Mise en place d'un canal de support (email, forum, chat) pour répondre aux questions et recueillir les feedbacks.
- Documentation et FAQ à jour pour faciliter l'utilisation et la résolution des problèmes courants.