Cahier des Charges : Chatbot Intelligent pour le Site Web de l'ISET

1. Introduction

1.1 Contexte du projet

L'Institut Supérieur des Études Technologiques (ISET) souhaite améliorer l'expérience utilisateur de son site web en intégrant un chatbot intelligent. Ce système interactif vise à assister les étudiants, le personnel et les visiteurs du site en leur fournissant des réponses rapides et pertinentes à leurs questions, tout en les guidant efficacement vers les pages et ressources désirées.

Le développement de ce chatbot s'inscrit dans une démarche d'amélioration continue des services numériques de l'institution, avec pour objectif de réduire le temps de recherche d'informations et d'optimiser l'accessibilité des ressources disponibles sur le site web.

1.2 Objectifs généraux

1. **Répondre aux questions des utilisateurs** sur divers sujets liés à l'ISET :
    - Informations sur les cours et programmes universitaires
    - Procédures administratives
    - Événements et actualités
    - Ressources disponibles
2. **Optimiser le temps de recherche** en guidant l'utilisateur vers les pages spécifiques du site correspondant à sa demande.
3. **Améliorer l'expérience utilisateur** en proposant une interaction naturelle et fluide via une interface conversationnelle.
4. **Collecter des données** sur les questions fréquemment posées pour enrichir progressivement la base de connaissances.

1.3 Portée du projet

Ce projet couvre le développement complet d'une solution de chatbot basée sur des techniques de NLP et Machine Learning (sans deep learning), son intégration au site web de l'ISET, ainsi que les mécanismes d'évolution et d'amélioration continue du système.

2. Exigences

2.1 Exigences fonctionnelles

2.1.1 Interactions avec l'utilisateur

- **F1**: Le système doit fournir une interface conversationnelle permettant aux utilisateurs de poser des questions en langage naturel.
- **F2**: Le système doit afficher des réponses précises et contextuellement pertinentes aux questions posées.

- **F3**: Le système doit pouvoir rediriger l'utilisateur vers des pages spécifiques du site web de l'ISET via des liens cliquables.
- **F4**: Le système doit gérer les salutations et formules de politesse courantes pour maintenir une conversation naturelle.
- **F5**: Le système doit demander des précisions lorsque la question est ambiguë ou manque de contexte.
- **F6**: Le système doit proposer un système de notation des réponses (/) pour recueillir les feedbacks utilisateurs.
- **F7**: Le système doit suggérer proactivement des informations ou liens complémentaires en fonction du contexte de la conversation.

2.1.2 Traitement des questions

- **F8**: Le système doit identifier l'intention principale de la question de l'utilisateur.
- **F9**: Le système doit extraire les entités clés mentionnées dans les questions (ex: nom du cours, département, procédure).
- **F10**: Le système doit détecter les questions similaires à celles déjà présentes dans sa base de connaissances.
- **F11**: Le système doit être capable de traiter des formulations variées d'une même question.
- **F12**: Le système doit reconnaître les questions hors de son domaine de compétence et indiquer ses limites.

2.1.3 Gestion des données

- **F13**: Le système doit charger sa base de connaissances initiale à partir du fichier `data.csv`.
- **F14**: Le système doit enregistrer les feedbacks des utilisateurs dans un fichier `feedback.csv`.
- **F15**: Le système doit stocker les questions non reconnues dans un fichier `unanswered.csv`.
- **F16**: Le système doit être capable de consulter et exploiter les données de ces fichiers CSV en temps réel.

2.2 Exigences non fonctionnelles

2.2.1 Performance

- **NF1**: Le système doit répondre aux questions en moins de 2 secondes dans 95% des cas.
- **NF2**: Le système doit pouvoir traiter au moins 100 requêtes simultanées.
- **NF3**: Le système doit maintenir un taux de précision global d'au moins 85% dans l'identification des intentions.

2.2.2 Fiabilité

- **NF4**: Le système doit être disponible 99,9% du temps (équivalent à moins de 9 heures d'indisponibilité par an).
- **NF5**: Le système doit être capable de se rétablir automatiquement après une défaillance.
- **NF6**: Le système doit sauvegarder régulièrement les fichiers CSV pour éviter la perte de données.

2.2.3 Utilisabilité

- **NF7**: L'interface utilisateur doit être intuitive et accessible sur tous les appareils (responsive design).

- **NF8**: Le système doit supporter au minimum les navigateurs Chrome, Firefox, Safari et Edge dans leurs versions les plus récentes.
- **NF9**: Le temps d'apprentissage pour un nouvel utilisateur ne doit pas dépasser 2 minutes.

2.2.4 Maintenabilité

- **NF10**: Le code doit être modulaire et bien documenté pour faciliter les évolutions futures.
- **NF11**: Le système doit permettre l'ajout facile de nouvelles questions-réponses sans nécessiter de modification du code.
- **NF12**: Les fichiers CSV doivent suivre une structure standard documentée pour faciliter leur manipulation.

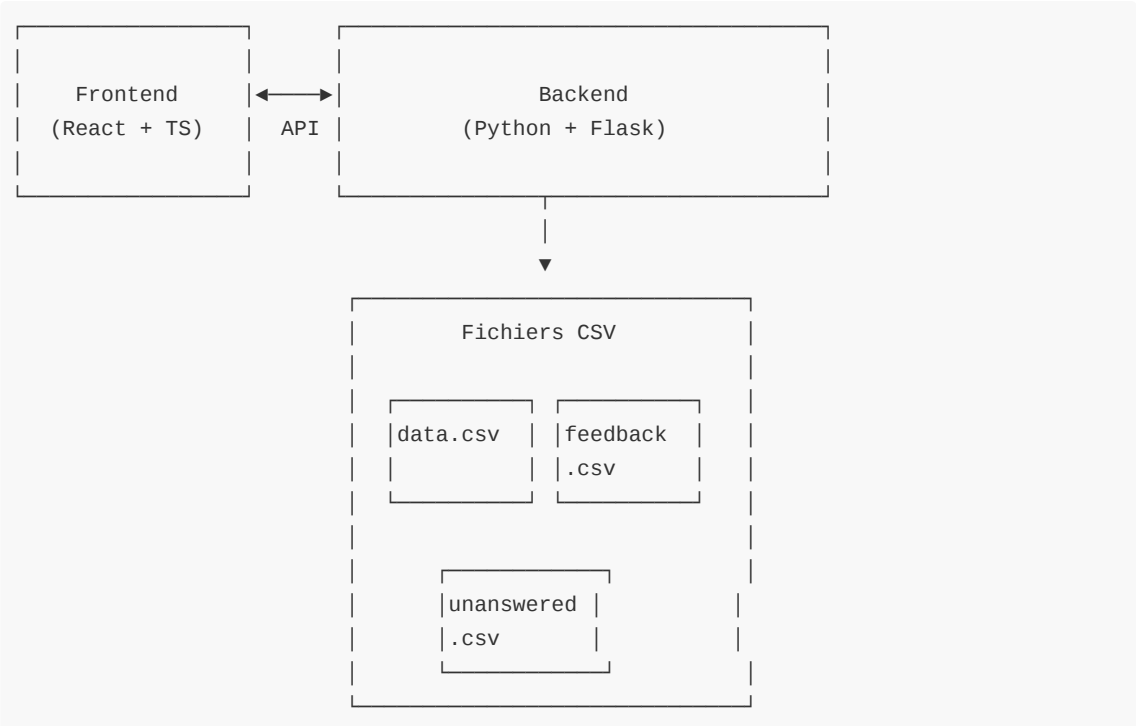
2.2.5 Sécurité

- **NF13**: Le système doit valider et assainir toutes les entrées utilisateur pour prévenir les injections.
- **NF14**: Le système ne doit pas stocker d'informations personnelles identifiables dans les fichiers de logs ou de feedback.
- **NF15**: Les communications entre le frontend et le backend doivent être sécurisées (HTTPS).

3. Architecture technique

3.1 Vue d'ensemble

L'architecture du chatbot intelligent de l'ISET repose sur une séparation claire entre le frontend et le backend, avec une communication via API REST. Cette architecture garantit une évolutivité et une maintenabilité optimales du système.



3.2 Frontend (React + TypeScript)

Le frontend sera développé en React avec TypeScript pour garantir la robustesse du code et une expérience utilisateur optimale.

3.2.1 Composants principaux

- **ChatContainer**: Composant principal encapsulant toute l'interface du chatbot
- **MessageList**: Affichage des messages échangés entre l'utilisateur et le chatbot
- **MessageItem**: Représentation individuelle d'un message (utilisateur ou chatbot)
- **InputBox**: Zone de saisie pour les questions de l'utilisateur
- **FeedbackButtons**: Boutons de feedback (👍/👎) pour évaluer les réponses
- **SuggestionChips**: Suggestions proactives cliquables sur des sujets connexes
- **LinkPreview**: Aperçu des liens vers les pages du site ISET

3.2.2 Gestion d'état

- Utilisation de React Context API ou Redux pour la gestion de l'état global
- Stockage de l'historique des conversations dans le state local
- Tracking des messages en attente de réponse

3.2.3 Communication avec le backend

- Utilisation d'Axios pour les appels API
- Mise en place d'un système de retry en cas d'échec
- Gestion des timeouts et des erreurs

3.3 Backend (Python + Flask)

Le backend sera développé en Python avec le framework Flask, offrant à la fois légèreté et puissance pour le traitement des requêtes et l'implémentation des algorithmes de ML et NLP.

3.3.1 Structure du backend

```
backend/
├── app.py                # Point d'entrée de l'application Flask
├── config.py             # Configuration de l'application
├── requirements.txt      # Dépendances Python
├── data/
│   ├── data.csv          # Base de connaissances principale
│   ├── feedback.csv      # Feedbacks des utilisateurs
│   └── unanswered.csv    # Questions sans réponse
├── models/
│   ├── preprocessor.py   # Prétraitement des textes
│   ├── vectorizer.py     # Vectorisation des questions
│   ├── classifier.py     # Classification des intentions
│   └── searcher.py       # Recherche de similarité
├── utils/
│   ├── csv_handler.py    # Gestion des fichiers CSV
│   ├── text_utils.py     # Utilitaires de traitement de texte
│   └── response_builder.py  # Construction des réponses
└── api/
    ├── routes.py         # Définition des endpoints
    └── middleware.py     # Middleware pour sécurité et logging
```

3.3.2 API Endpoints

- **POST /api/chat**

    - Reçoit les questions des utilisateurs
    - Retourne les réponses appropriées
- **POST /api/feedback**
    - Reçoit les évaluations des réponses (👍/👎)
    - Stocke les feedbacks dans `feedback.csv`
- **GET /api/suggestions**
    - Retourne des suggestions contextuelles basées sur l'historique de conversation
- **GET /api/health**
    - Vérifie l'état du service et ses composants

3.4 Structure des fichiers CSV

3.4.1 data.csv

```
id,question,variations,answer,category,link,keywords
1,"Quelles sont les modalités d'inscription?","Comment s'inscrire?|Procédure
d'inscription","Pour vous inscrire, vous devez soumettre votre dossier au bureau des
inscriptions avant le 15 septembre. Les documents requis
sont...",Administratif,/inscription,inscription;modalités;dossier
2,"Quel est le programme du cours de Machine Learning?","Contenu du cours ML|Syllabus
Machine Learning","Le cours de Machine Learning couvre les concepts suivants:
prétraitement des données, algorithmes supervisés et non-
supervisés...",Académique,/cours/machine-learning,ML;programme;syllabus
```

3.4.2 feedback.csv

```
id,question_id,response_id,rating,comment,timestamp,user_session_id
1,15,42,1,"Réponse très utile",2025-05-01T14:30:45,sess_123456
2,23,51,-1,"Information incorrecte",2025-05-02T09:12:22,sess_234567
```

3.4.3 unanswered.csv

```
id,question,timestamp,user_session_id,suggested_category,confidence_score
1,"Quand aura lieu la journée portes ouvertes?",2025-05-
01T10:15:30,sess_345678,Événements,0.3
2,"Comment contacter le professeur de cybersécurité?",2025-05-
02T16:42:18,sess_456789,Contacts,0.5
```

4. Choix technologiques

4.1 Frontend

Technologie	Justification
React 18	Framework UI moderne, performant et largement adopté permettant la création de composants réutilisables et une gestion efficace du DOM
TypeScript	Apporte un typage statique à JavaScript, réduisant les erreurs de

5	développement et améliorant la maintenabilité du code
Axios	Bibliothèque pour les requêtes HTTP offrant une API simple et des fonctionnalités avancées (intercepteurs, annulation de requêtes)
Tailwind CSS	Framework CSS utilitaire permettant un développement rapide et cohérent de l'interface utilisateur
React Testing Library	Solution pour les tests unitaires et d'intégration orientée sur le comportement plutôt que sur l'implémentation
ESLint & Prettier	Outils de linting et formatage pour maintenir une qualité et cohérence du code

4.2 Backend

Technologie	Justification
Python 3.10+	Langage de référence pour le ML et le NLP avec un écosystème riche de bibliothèques
Flask 2.3+	Framework web léger, flexible et facile à étendre pour créer des API REST
NLTK	Bibliothèque complète pour le traitement du langage naturel (tokenisation, stemming, etc.)
scikit-learn	Bibliothèque ML robuste incluant des algorithmes de classification et de vectorisation
pandas	Outil puissant pour la manipulation des données structurées comme les CSV
Whoosh	Moteur de recherche pure Python pour l'indexation et la recherche de texte
FastText (modèles pré-entraînés)	Solution légère pour les word embeddings sans deep learning
pytest	Framework de test pour garantir la qualité du code backend
Flask-Cors	Extension Flask pour gérer les requêtes cross-origin
gunicorn	Serveur WSGI Python pour le déploiement en production

4.3 Gestion de données

Technologie	Justification
pandas	Manipulation efficace des fichiers CSV avec support des opérations complexes
CSV Dictreader/Dictwriter	API Python standard pour les opérations basiques sur CSV
watchdog	Surveillance des modifications de fichiers CSV pour actualisation dynamique

4.4 DevOps

Technologie	Justification
Docker	Conteneurisation pour garantir la portabilité et la cohérence des environnements
GitHub Actions	CI/CD pour automatiser les tests et le déploiement
Sentry	Surveillance des erreurs en production

5. Étapes de développement

5.1 Préparation des données

5.1.1 Structure et création de `data.csv`

1. **Définition du schéma** du fichier `data.csv` :
    - `id` : Identifiant unique de la paire question-réponse
    - `question` : Formulation principale de la question
    - `variations` : Formulations alternatives séparées par des délimiteurs (|)
    - `answer` : Réponse textuelle complète
    - `category` : Catégorie thématique (Administratif, Académique, etc.)
    - `link` : URL de redirection vers une page spécifique du site
    - `keywords` : Mots-clés associés séparés par des délimiteurs (;)
2. **Collecte initiale des données** :
    - Extraction des FAQ existantes sur le site de l'ISET
    - Interviews avec le personnel administratif pour identifier les questions fréquentes
    - Analyse des logs de recherche du site web pour identifier les besoins d'information
3. **Enrichissement des données** :
    - Génération de variations linguistiques pour chaque question principale
    - Association systématique de mots-clés pertinents
    - Vérification de la cohérence des catégories

5.1.2 Prétraitement des textes

1. **Nettoyage textuel** :
    - Suppression des caractères spéciaux et de la ponctuation non significative
    - Normalisation des caractères (accents, casse)
    - Correction orthographique basique
2. **Tokenisation** :
    - Segmentation des questions en tokens (mots, sous-mots)
    - Filtrage des stopwords (mots vides comme "le", "la", "et", etc.)
3. **Lemmatisation/Stemming** :

- Réduction des mots à leur forme canonique (lemmatisation)
- Alternative: réduction à leur racine (stemming)

4. **Extraction de caractéristiques** :
   - N-grammes (unigrammes, bigrammes)
   - Part-of-speech tagging pour identifier les rôles grammaticaux

5.1.3 Vectorisation des textes

1. **Implémentation du TF-IDF** :
   - Calcul des fréquences des termes (TF)
   - Calcul de l'inverse de la fréquence documentaire (IDF)
   - Génération de la matrice TF-IDF pour l'ensemble des questions
2. **Alternative: Bag-of-Words** :
   - Création du vocabulaire à partir du corpus de questions
   - Vectorisation des questions basée sur les occurrences
3. **Word Embeddings** :
   - Utilisation de modèles FastText pré-entraînés
   - Calcul des vecteurs moyens par question
4. **Réduction de dimensionnalité** (optionnel) :
   - Application de PCA ou t-SNE pour visualiser les clusters de questions
   - Identification des thématiques émergentes

5.2 Développement du backend

5.2.1 Mise en place de l'environnement Flask

1. **Configuration de base** :
   - Initialisation de l'application Flask
   - Configuration des routes API
   - Mise en place des middlewares (CORS, logging, sécurité)
2. **Gestion des fichiers CSV** :
   - Développement des fonctions de lecture/écriture des fichiers CSV
   - Implémentation d'un cache pour optimiser les performances
   - Création de mécanismes de verrouillage pour éviter les conflits d'accès

5.2.2 Implémentation des algorithmes de ML

1. **Classification des intentions** :
   - Entraînement d'un modèle SVM avec les vecteurs TF-IDF
   - Alternative: implémentation de Naive Bayes pour la classification
   - Validation croisée pour l'optimisation des hyperparamètres
2. **Mesure de similarité** :
   - Implémentation de la similarité cosinus entre vecteurs de questions
   - Définition d'un seuil minimal de similarité pour considérer une correspondance
   - Stratégie de fallback quand aucune question similaire n'est trouvée

3. **Extraction d'entités** :
    - Développement de règles pour l'identification des entités nommées
    - Création de patterns de reconnaissance pour les dates, cours, départements, etc.

5.2.3 Moteur de recherche interne

1. **Indexation avec Whoosh** :
    - Création d'un schéma d'indexation pour les questions et réponses
    - Indexation initiale du contenu de `data.csv`
    - Configuration de l'analyseur pour le français (et autres langues si nécessaire)
2. **Optimisation de la recherche** :
    - Configuration des boosts sur les champs prioritaires
    - Implémentation de la correction orthographique
    - Gestion des synonymes via un thésaurus

5.2.4 Gestion des cas inconnus

1. **Détection des questions hors périmètre** :
    - Calcul du score de confiance pour chaque prédiction
    - Identification des questions sous le seuil de confiance
    - Enregistrement dans `unanswered.csv` avec métadonnées
2. **Réponses par défaut** :
    - Configuration de réponses génériques par catégorie
    - Mécanisme d'escalade pour proposer un contact humain
    - Suggestions de reformulation

5.2.5 API REST

1. **Endpoint de chat** :
    - Réception de la question utilisateur
    - Traitement et génération de la réponse
    - Retour structuré avec réponse, liens et suggestions
2. **Endpoint de feedback** :
    - Enregistrement des évaluations utilisateur
    - Association avec la paire question-réponse correspondante
    - Stockage dans `feedback.csv`
3. **Sécurisation de l'API** :
    - Validation des entrées
    - Rate limiting
    - Prévention des attaques par injection

5.3 Développement du frontend

5.3.1 Structure de l'application React

1. **Architecture des composants** :

- Création des composants principaux (ChatContainer, MessageList, etc.)
- Implémentation des interfaces TypeScript
- Organisation des composants selon le pattern Atomic Design

2. **Gestion de l'état** :
   - Configuration du store global
   - Définition des reducers et actions
   - Implémentation des effets secondaires

5.3.2 Interface utilisateur

1. **Design de l'interface de chat** :
   - Implémentation de la bulle de chat (widget)
   - Conception de l'interface conversationnelle
   - Adaptation responsive pour tous les appareils
2. **Interaction utilisateur** :
   - Gestion de la saisie utilisateur
   - Affichage des indicateurs de chargement
   - Animation des transitions entre messages
3. **Composants spéciaux** :
   - Boutons de feedback (/)
   - Affichage de liens enrichis vers les pages du site
   - Chips de suggestions cliquables

5.3.3 Intégration avec le backend

1. **Service d'API** :
   - Implémentation des appels aux endpoints du backend
   - Gestion du cycle de vie des requêtes
   - Traitement des erreurs et retries
2. **Persistance locale** :
   - Stockage de l'historique de conversation
   - Sauvegarde des préférences utilisateur
   - Gestion du cache pour les réponses fréquentes

5.4 Intégration au site web de l'ISET

5.4.1 Modalités d'intégration

1. **Widget flottant** :
   - Développement d'un composant iframe ou script d'intégration
   - Positionnement adaptatif selon le layout du site
   - Ouverture/fermeture contextuelle
2. **Communication cross-domain** :
   - Configuration des entêtes CORS
   - Implémentation de postMessage pour les communications iframe
   - Gestion des tokens d'authentification si nécessaire

5.4.2 Contextualisation

1. **Détection de page** :
    - Récupération de l'URL courante et des métadonnées de la page
    - Ajustement des suggestions selon le contexte
    - Priorisation des réponses pertinentes au contexte actuel
2. **Personnalisation** :
    - Adaptation des couleurs et styles au thème du site
    - Configuration des paramètres d'affichage
    - Options d'intégration par section du site

5.5 Évaluation et optimisation

5.5.1 Métriques d'évaluation

1. **Métriques de performance ML** :
    - Précision (Accuracy): pourcentage de questions correctement traitées
    - Rappel (Recall): capacité à identifier toutes les questions pertinentes
    - F1-Score: moyenne harmonique entre précision et rappel
    - Matrice de confusion pour analyser les erreurs par catégorie
2. **Métriques d'expérience utilisateur** :
    - Taux de satisfaction (pourcentage de)
    - Taux d'abandon (conversations sans conclusion)
    - Temps moyen pour obtenir une réponse satisfaisante
    - Nombre moyen d'échanges par conversation

5.5.2 Procédure de test

1. **Tests unitaires** :
    - Vérification des fonctions individuelles
    - Tests des composants isolés
2. **Tests d'intégration** :
    - Validation du flux complet de traitement
    - Simulation de conversations
3. **Tests utilisateurs** :
    - Sessions avec utilisateurs réels
    - Collecte de feedback qualitatif
4. **Tests de charge** :
    - Simulation de multiples utilisateurs simultanés
    - Mesure des temps de réponse sous charge

5.5.3 Optimisation itérative

1. **Analyse des erreurs** :
    - Identification des questions mal classifiées
    - Examen des cas de faible confiance

2. **Enrichissement du dataset** :
    - Intégration des questions non répondues
    - Ajout de variations pour améliorer la couverture
3. **Ajustement des algorithmes** :
    - Réglage des hyperparamètres
    - Test de méthodes alternatives

6. Plan d'évolution et améliorations

6.1 Support multilingue

1. **Approche pour le support multilingue** :
    - Création de fichiers CSV distincts par langue (data_fr.csv, data_en.csv)
    - Détection automatique de la langue d'entrée
    - Vectorisation et modèles spécifiques par langue
2. **Implémentation technique** :
    - Utilisation de langdetect pour identifier la langue de la question
    - Configuration de tokenizers et stopwords spécifiques à chaque langue
    - Interface permettant à l'utilisateur de changer manuellement la langue

6.2 Système de feedback intelligent

1. **Analyse avancée des feedbacks** :
    - Identification des patterns dans les évaluations négatives
    - Regroupement des questions problématiques
    - Détection des biais dans les réponses
2. **Apprentissage actif** :
    - Priorisation des questions à améliorer en fonction des feedbacks
    - Alertes pour les réponses fréquemment notées négativement
    - Tableau de bord pour les administrateurs

6.3 Graphe de connaissances local

1. **Construction du graphe** :
    - Définition des entités (cours, départements, procédures, etc.)
    - Établissement des relations entre entités
    - Extraction des propriétés pour chaque entité
2. **Exploitation du graphe** :
    - Navigation contextuelle entre questions liées
    - Inférence de réponses pour des questions complexes
    - Enrichissement des réponses avec des informations connexes

6.4 Apprentissage semi-automatique

1. **Traitement de unanswered.csv** :
    - Interface administrateur pour examiner les questions sans réponse

- Suggestions automatiques de réponses basées sur le contenu existant
- Workflow d'approbation pour l'intégration dans data.csv

2. **Exploitation de feedback.csv** :
   - Ajustement des scores de confiance en fonction des évaluations
   - Identification des réponses à reformuler
   - Analyse des tendances temporelles dans les feedbacks

6.5 Fonctionnalités avancées

1. **Mode proactif** :
   - Suggestions contextuelles basées sur le comportement de navigation
   - Notifications sur les mises à jour pertinentes du site
   - Rappels personnalisés sur les procédures à dates limites
2. **Intégration de sources dynamiques** :
   - Connexion aux actualités du site pour les informations récentes
   - Synchronisation avec le calendrier des événements
   - Accès aux informations de disponibilité des ressources

7. Calendrier et livrables

7.1 Phases du projet

Phase	Description
Phase 1: Préparation	Analyse des besoins, constitution du dataset initial, mise en place des environnements
Phase 2: Développement backend	Implémentation des algorithmes ML/NLP, développement de l'API Flask
Phase 3: Développement frontend	Création de l'interface React, intégration avec le backend
Phase 4: Intégration et tests	Intégration au site ISET, tests utilisateurs, optimisations
Phase 5: Déploiement et documentation	Mise en production, documentation technique et utilisateur

7.2 Livrables

1. **Code source** :
   - Repositories GitHub pour le frontend et le backend
   - Documentation technique
   - Tests automatisés
2. **Dataset initial** :
   - Fichier data.csv avec au moins 100 paires question-réponse
   - Structure vide pour feedback.csv et unanswered.csv
3. **Documentation** :

- Manuel utilisateur
- Guide d'administration
- Documentation API
- Rapport de performances

4. **Rapport de projet** :
    - Détails méthodologiques
    - Analyse des résultats
    - Recommandations pour évolutions futures

8. Conclusion

Ce cahier des charges présente une approche complète et structurée pour le développement d'un chatbot intelligent destiné au site web de l'ISET. En s'appuyant sur des techniques de ML et NLP sans deep learning, et en utilisant exclusivement des fichiers CSV comme source de données, le système proposé permettra d'améliorer significativement l'expérience utilisateur du site.

L'architecture modulaire, la méthodologie de développement itérative et les mécanismes d'évolution continue garantiront l'adaptabilité et la pérennité de la solution. Les spécifications techniques détaillées fournissent une base solide pour l'implémentation, tandis que les stratégies de gestion des risques et d'amélioration continue assurent la viabilité à long terme du projet.

La combinaison d'un frontend React/TypeScript ergonomique et d'un backend Python/Flask robuste, s'appuyant sur des algorithmes éprouvés de ML et NLP, permettra de créer une solution à la fois performante et évolutive, capable de s'adapter aux besoins changeants des utilisateurs du site de l'ISET.