

Université de Sousse

Ecole Nationale d'Ingénieurs de Sousse



Rapport de Stage d'été

[] Initiation [X] Ingénieur

Réalisé par :

Habib Saadallah

Filière :

Génie Electronique Industrielle

Stage effectué au sein de la Société (ou de l'Entreprise)



Kmsolutions

Cyber Park Kairouan

Période du stage : 07/07/2026 - 07/08/2025

Dédicaces

Je dédie ce travail

A mes très chers parents

À qui je dois tout. Aucune dédicace ne saurait exprimer l'étendue de mon amour, de mon admiration et de ma reconnaissance. Vous avez tout donné pour mon épanouissement et ma réussite. Votre soutien et votre présence ont été ma source de force et de courage, me guidant vers le succès.

Que Dieu vous préserve la bonne santé et la longue vie

A mon cher frère

Vous enrichissez ma vie et la rendez plus sereine, surtout cette année où vos encouragements et votre soutien m'ont été d'une aide précieuse pour progresser.

Saadallah Habib

Remerciements

Je tiens à remercier l'équipe de K.M Solutions pour leur accueil chaleureux et leur accompagnement tout au long de ce stage. Leur expertise et leurs conseils ont été déterminants dans la réussite de ce projet.

Je remercie également mon établissement de formation pour cette opportunité de stage qui a enrichi considérablement mon parcours académique et professionnel.

Table des matières

Dédicaces	2
Remerciements	3
Table des figures	7
Liste D’abréviations	9
Introduction Générale	10
1 Organisme d’accueil et problématique traitée	11
1. Introduction	11
2. Présentation de l’entreprise	11
2.1 Informations générales	11
2.2 Produits et services	12
2.3 Organisation générale de l’entreprise	12
3. Problématique traitée et travail demandé	12
4. Démarche – Méthodologie utilisée	13
5. Chronogramme des tâches menées durant le stage	13
6. Conclusion	13
2 Analyse des besoins et conception	15
1. Introduction	15
2. Notions fondamentales	15
2.1 Historique de l’IA	15
2.2 Avancées et applications de l’intelligence artificielle	16
3. Les chatbots : concepts et technologies	16
3.1 Définition d’un chatbot	16
3.2 Technologies courantes pour développer des chatbots	17
3.2.1 Traitement du langage naturel (NLP)	17
3.2.2 OpenAI et ses modèles pour les chatbots	17
3.3 Domaines d’application des chatbots	18
4. Étude de l’existant	18

5.	Analyse des besoins	19
5.1	Besoins fonctionnels	19
5.2	Besoins non-fonctionnels	19
6.	Conception	20
6.1	Architecture générale	20
6.2	Modélisation orientée objet	21
6.2.1	Diagramme de classes	21
6.3	Modélisation comportementale	21
6.3.1	Diagramme de cas d'utilisation	21
6.3.2	Diagrammes de séquence	22
6.3.3	Diagramme d'activité	23
6.4	Choix technologiques	23
7.	Conclusion	24
3	Réalisation	25
1.	Introduction	25
2.	Choix technologiques et environnements de développement	25
2.1	Environnement outillé et cadre de développement	25
2.2	Pile technologique retenue	27
2.3	Authentification et gestion des utilisateurs	31
2.4	Chatbot spécialisé et moteur NLP	31
2.5	Interface vocale	32
2.6	Historique des conversations et recherche	32
2.7	Déploiement cloud et optimisation	32
3.	Modèle d'intelligence artificielle	32
3.1	Justification du choix technologique	32
3.2	Architecture du modèle	33
3.3	Processus d'entraînement et de prédiction	33
3.4	Traçabilité objectifs → réalisations	34
4.	Architecture de la base de données	34
4.1	Schéma conceptuel	34
5.	Interfaces utilisateur et prototype	35
5.1	Page de connexion (Login)	35
5.2	Page d'inscription (Register)	36
5.3	Page de récupération de mot de passe (Forgot Password)	36
5.4	Interface principale du chatbot	37
5.5	Interface d'historique des conversations	38
6.	Métriques de performance et validation	38
6.1	Performances mesurées	38
6.2	Spécialisation ML/DL	39

7.	Tests et validation	39
7.1	Tests fonctionnels et d'intégration	39
7.2	Validation en environnement de production	39
8.	Analyse critique et perspectives	40
8.1	Points forts identifiés	40
8.2	Limitations techniques	40
8.3	Améliorations envisageables	40
9.	Conclusion	40
	Conclusion Générale	42
	Références	43
	Annexes	45

Table des figures

Figure 1.1	Logo de l'entreprise K.M Solutions	12
Figure 2.1	Chatbot	16
Figure 2.2	Traitement du langage naturel	17
Figure 2.3	OpenAI	18
Figure 2.4	Dialogflow	18
Figure 2.5	Rasa	18
Figure 2.6	Botpress	19
Figure 2.7	Architecture générale du système	20
Figure 2.8	Diagramme de classes principal	21
Figure 2.9	Diagramme de cas d'utilisation	22
Figure 2.10	Diagramme de séquence - Authentification	22
Figure 2.11	Diagramme de séquence - Dialogue avec le chatbot	23
Figure 2.12	Diagramme d'activité - Processus de chat	23
Figure 3.1	Logo Visual Studio Code	26
Figure 3.2	Hugging Face	26
Figure 3.3	Git	27
Figure 3.4	GitHub	27
Figure 3.5	Render	27
Figure 3.6	Logo Flask	28
Figure 3.7	SQLite	28
Figure 3.8	Scikit-learn	29
Figure 3.9	TF-IDF	29
Figure 3.10	Régression Logistique	29
Figure 3.11	Logo HTML5	30
Figure 3.12	Logo CSS3	30
Figure 3.13	Logo JavaScript	31
Figure 3.14	Interface de connexion : authentification utilisateur réussie	35
Figure 3.15	Message d'erreur affiché lors d'une tentative de connexion invalide	36
Figure 3.16	Interface d'inscription : création d'un nouveau compte utilisateur	36
Figure 3.17	Interface de récupération de mot de passe avec questions de sécurité	37
Figure 3.18	Interface de chat : interaction textuelle et vocale avec le chatbot	37

Figure 3.19 Interface d'historique avec fonction de recherche et pagination	38
---	----

Liste d'abréviations

IA	: Intelligence Artificielle
NLP	: Traitement Du Langage Nature
HTML	: HyperText Markup Language
CSS	: Cascading Style Sheets
JS	: JavaScript
UML	: Langage de Modélisation Unifié
ML	: Machine Learning
JSON	: JavaScript Object Notation
IDE	: Environnement de développement intégré
VSCode	: Visual Studio Code

Introduction Générale

Dans le cadre de ma formation, j'ai effectué un stage au sein de l'entreprise K.M Solutions, spécialisée dans le développement de solutions technologiques innovantes. Ce stage, d'une durée de 30 jours, m'a permis de mettre en pratique mes connaissances théoriques dans le domaine de l'intelligence artificielle et du développement web.

L'objectif principal de ce stage était de concevoir et développer un chatbot intelligent spécialisé dans les domaines du Machine Learning et du Deep Learning. Ce projet m'a offert l'opportunité d'explorer les technologies modernes d'IA tout en acquérant une expérience pratique dans le développement d'applications web complètes.

Ce rapport présente le travail accompli, les technologies utilisées, les défis rencontrés et les compétences acquises durant cette période enrichissante.

Chapitre1 : Organisme d'accueil et problématique traitée

1. Introduction

Le stage s'est déroulé au sein de l'entreprise **K.M Solutions**, une société innovante spécialisée dans le développement de solutions numériques et technologiques. Ce chapitre présente l'entreprise d'accueil, son organisation, la problématique traitée dans le cadre du stage, ainsi que la démarche méthodologique suivie. Un chronogramme récapitulatif des activités menées durant le stage est également fourni.

2. Présentation de l'entreprise

2.1 Informations générales

- **Date de création** : 2024
- **Forme juridique** : Société à responsabilité limitée (SARL)
- **Siège social** : Kairouan, Tunisie
- **Capital social** : XXX TND
- **Chiffre d'affaires** : XXX TND
- **Personnel** : 11-50 employés
- **Secteur d'activité** : Développement de logiciels personnalisés de systèmes informatiques



FIGURE 1.1 – Logo de l'entreprise K.M Solutions

2.2 Produits et services

K.M Solutions propose des solutions personnalisées pour les entreprises et particuliers :

- Développement d'applications web et mobiles.
- Intégration de systèmes informatiques.
- Solutions IoT (Internet of Things).
- Développement de chatbots intelligents pour l'automatisation des interactions.
- Services de maintenance et support technique.

2.3 Organisation générale de l'entreprise

L'entreprise est structurée en plusieurs départements complémentaires :

- **Direction Générale** : supervision des projets et prise de décisions stratégiques.
- **Département Technique** : développement logiciel, intégration et R&D.
- **Département Commercial et Marketing** : prospection clients et promotion des solutions.

3. Problématique traitée et travail demandé

La problématique abordée durant le stage consiste à développer une application web de type **chat-bot intelligent**, permettant aux utilisateurs d'interagir de manière fluide avec un modèle conversa-

Chapitre 1. Organisme d'accueil et problématique traitée

tionnel. Les objectifs principaux sont :

- Concevoir et implémenter un système d'authentification sécurisé.
- Développer un module de traitement automatique du langage naturel (NLP).
- Intégrer une base de données pour stocker les conversations.
- Déployer le projet sur une plateforme cloud .

4. Démarche – Méthodologie utilisée

La méthodologie adoptée s'articule autour des étapes suivantes :

1. **Observation et analyse** : étude de l'existant et identification des besoins.
2. **Conception** : choix des technologies, modélisation des données et architecture logicielle.
3. **Développement** : implémentation des différentes fonctionnalités.
4. **Tests et simulations** : validation du fonctionnement du chatbot.
5. **Déploiement** : mise en ligne pour une utilisation réelle.

5. Chronogramme des tâches menées durant le stage

Le tableau ci-dessous présente le chronogramme des tâches réalisées pendant le stage. Les week-ends et jours fériés ne sont pas inclus.

Tâche	L07	M08	M09	J10	V11	L14	M15	M16	J17	V18	L21	M22	M23	J24	V25	L28	M29	M30	J31
Observation et visite des services	X	X																	
Formation et autoformation		X	X	X		X													
Analyse et conception				X	X	X	X												
Développement							X	X	X	X	X	X	X	X	X	X			
Tests et simulations									X	X	X	X	X	X	X	X	X		
Déploiement														X	X	X	X	X	
Rédaction du rapport de stage																	X	X	X

TABLE 1.1 – Chronogramme des tâches réalisées durant le stage (07/07 au 31/07)

6. Conclusion

Ce premier chapitre a permis de présenter l'entreprise d'accueil K.M Solutions, ses produits, services et organisation interne. La problématique traitée a été clairement identifiée : la mise en place

Chapitre 1. Organisme d'accueil et problématique traitée

d'un chatbot intelligent déployé dans un environnement cloud. Enfin, le chronogramme a mis en évidence une démarche progressive allant de l'observation initiale jusqu'au déploiement effectif du prototype. Cette étape a constitué la base solide sur laquelle les phases de conception et de réalisation ont été construites.

Chapitre2 : Analyse des besoins et conception

1. Introduction

L'étape d'analyse et de conception constitue une phase fondamentale dans tout projet de développement logiciel. Elle permet de traduire les besoins exprimés en spécifications claires et en modèles concrets, qui guideront la réalisation technique. Dans le cadre du présent projet, il s'agit de développer un **chatbot intelligent** accessible via une application web sécurisée, intégrant un système d'authentification et une gestion locale des conversations.

2. Notions fondamentales

Avant d'aborder l'analyse des besoins, il convient de rappeler quelques notions essentielles liées au domaine des agents conversationnels.

2.1 Historique de l'IA

L'histoire de l'intelligence artificielle (IA) est jalonnée de progrès majeurs, de périodes de stagnation et de débats éthiques. Dans les années 1950, des pionniers tels qu'Alan Turing et John McCarthy ont jeté les bases théoriques de l'IA. Les décennies suivantes ont vu l'émergence de systèmes experts dans les années 1980, suivis par plusieurs phases de désillusion appelées « hivers de l'IA ». Le renouveau de l'IA au 21^e siècle s'explique par l'essor de l'apprentissage profond, l'augmentation de la puissance de calcul et la disponibilité de grandes masses de données. Aujourd'hui, l'IA est omniprésente, avec des applications en santé, éducation, transports, industrie et communication, tout en soulevant des enjeux éthiques et sociétaux[1].

2.2 Avancées et applications de l'intelligence artificielle

L'IA progresse à un rythme soutenu et apporte des solutions dans des domaines variés. En santé, elle permet de détecter certaines maladies, de personnaliser les traitements et d'anticiper les risques. Dans le sport, elle contribue à l'analyse des performances et à la prévention des blessures. Grâce à l'automatisation, elle optimise les tâches répétitives et améliore la prise de décision par l'exploitation de grandes quantités de données.

Dans le domaine de l'éducation, l'IA favorise l'apprentissage autonome à travers des outils interactifs qui accompagnent les étudiants et facilitent l'accès à l'information.

3. Les chatbots : concepts et technologies

3.1 Définition d'un chatbot

Un **chatbot** est une application logicielle simulant une conversation humaine, que ce soit via un système basé sur des règles prédéfinies (menus, arbres de décision) ou par une intelligence artificielle conversationnelle avancée. On retrouve les chatbots sur de nombreux canaux : sites web, applications mobiles, réseaux sociaux ou encore services téléphoniques.

Les chatbots intelligents s'appuient sur des technologies telles que le *machine learning*, le *deep learning*, le traitement automatique du langage naturel (NLP) et la compréhension du langage naturel (NLU). Ces systèmes s'améliorent au fil des interactions et offrent une communication plus fluide et naturelle[2].

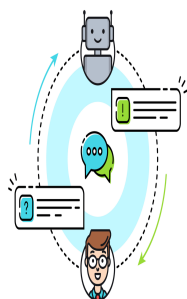


FIGURE 2.1 – Chatbot

3.2 Technologies courantes pour développer des chatbots

3.2.1 Traitement du langage naturel (NLP)

Le *Natural Language Processing* (NLP) est une branche de l'IA permettant aux ordinateurs de comprendre et de générer du langage humain. Il repose sur une combinaison de techniques d'apprentissage automatique et profond, et trouve des applications dans l'analyse syntaxique, la traduction automatique, la détection d'opinion ou encore la génération de texte.

Parmi ses techniques fondamentales, on distingue :

- **Tokenization** : division d'un texte en unités plus petites (mots, phrases) afin d'en faciliter l'analyse ;
- **Stemming** : réduction des mots à leur racine commune (par ex. « étudié », « étudiant » et « étude » ramenés à « étudié »).



FIGURE 2.2 – Traitement du langage naturel

3.2.2 OpenAI et ses modèles pour les chatbots

OpenAI a développé des modèles tels que GPT-3 et GPT-4, basés sur l'architecture *Transformers*, capables de traiter des séquences de texte longues et complexes. Ils sont accessibles via une API documentée, facilitant leur intégration dans des applications web et mobiles.

Cependant, ils présentent certaines limites :

- coûts élevés en ressources matérielles et financières ;
- accès restreint (non open source, limitation par clés API et quotas d'utilisation).



FIGURE 2.3 – OpenAI

3.3 Domaines d'application des chatbots

Les chatbots se sont imposés dans divers secteurs :

- **Service client** : assistance instantanée et gestion des requêtes ;
- **Éducation** : accompagnement des étudiants, soutien à l'apprentissage ;
- **Santé** : orientation, suivi et assistance aux patients ;
- **Commerce électronique** : recommandations et aide à l'achat.

Dans ce projet, le chatbot est spécifiquement destiné à l'assistance des étudiants en apprentissage automatique et apprentissage profond.

4. Étude de l'existant

Plusieurs solutions de chatbots sont déjà disponibles, parmi lesquelles :

- **Dialogflow (Google)** : puissant mais dépendant d'un service cloud ;



FIGURE 2.4 – Dialogflow

- **Rasa** : framework open source offrant une grande flexibilité ;



FIGURE 2.5 – Rasa

- **Botpress** : orienté vers le développement rapide de chatbots.

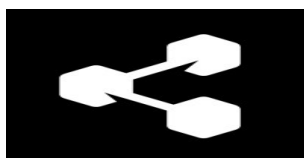


FIGURE 2.6 – Botpress

Si ces solutions sont adaptées à des besoins standards, elles présentent néanmoins des limites dans le cadre de ce projet : dépendance à des services tiers, coûts d'utilisation et faible maîtrise des données. Un développement personnalisé s'avère donc plus approprié.

5. Analyse des besoins

L'analyse des besoins consiste à identifier les fonctionnalités attendues et les contraintes associées.

5.1 Besoins fonctionnels

Les besoins fonctionnels représentent les fonctionnalités que doit offrir le système :

- **Authentification sécurisée** : inscription, connexion, gestion de sessions avec questions de sécurité ;
- **Interaction conversationnelle** : dialogue via interface web responsive avec support vocal ;
- **Intelligence artificielle** : traitement des requêtes avec modèle ML spécialisé en ML/DL ;
- **Sauvegarde des conversations** : stockage dans base de données locale avec horodatage ;
- **Gestion multi-utilisateurs** : support de plusieurs comptes simultanés ;
- **Historique des conversations** : consultation et recherche des échanges passés ;
- **Réinitialisation mot de passe** : récupération via questions de sécurité.

5.2 Besoins non-fonctionnels

Les besoins non-fonctionnels définissent les contraintes de qualité et performance :

- **Performance** : temps de réponse inférieur à 3 secondes ;
- **Sécurité** : chiffrement des mots de passe, protection contre les injections SQL ;

- **Fiabilité** : disponibilité 24h/24 avec gestion d’erreurs robuste ;
- **Évolutivité** : possibilité d’intégrer d’autres modèles NLP ;
- **Portabilité** : déploiement sur serveur distant (Render) ;
- **Utilisabilité** : interface intuitive et accessible sur mobile/desktop.

6. Conception

La conception traduit les besoins identifiés en une organisation technique concrète à travers plusieurs modèles UML.

6.1 Architecture générale

L’architecture du système repose sur un modèle en trois couches classique assurant une séparation claire des responsabilités :

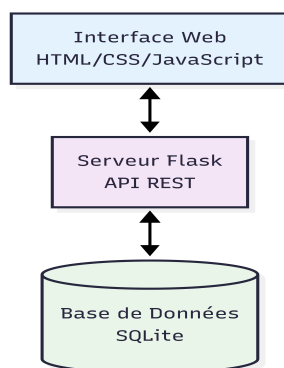


FIGURE 2.7 – Architecture générale du système

Couche Présentation (Frontend) : Interface utilisateur développée en HTML, CSS et JavaScript, offrant une expérience responsive et intégrant les API vocales du navigateur.

Couche Logique Métier (Backend) : Serveur Flask implémentant une API REST pour gérer l’authentification, les sessions utilisateurs, le traitement des messages par le modèle ML et la persistance des données.

Couche Données : Base de données SQLite stockant les utilisateurs, conversations et messages de manière locale et sécurisée.

6.2 Modélisation orientée objet

6.2.1 Diagramme de classes

La structure orientée objet du système est représentée par le diagramme de classes suivant :

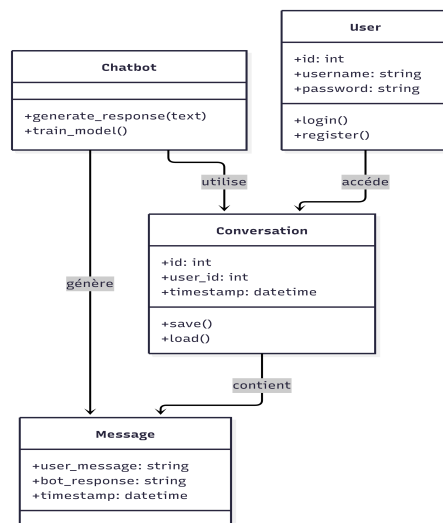


FIGURE 2.8 – Diagramme de classes principal

Le modèle comprend quatre classes principales :

- **User** : Gère les informations et l’authentification des utilisateurs ;
- **Conversation** : Représente une session de dialogue avec métadonnées ;
- **Message** : Encapsule les échanges individuels utilisateur-chatbot ;
- **Chatbot** : Implémente la logique d’intelligence artificielle.

Les relations établies respectent les cardinalités métier : un utilisateur peut avoir plusieurs conversations, chaque conversation contient plusieurs messages, et le chatbot traite tous les messages.

6.3 Modélisation comportementale

6.3.1 Diagramme de cas d’utilisation

Les fonctionnalités disponibles pour l’utilisateur sont représentées dans le diagramme suivant :

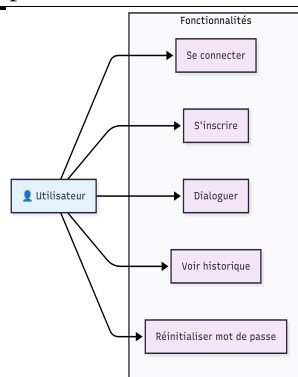


FIGURE 2.9 – Diagramme de cas d'utilisation

Le système offre cinq cas d'utilisation principaux couvrant la gestion de compte (connexion, inscription, réinitialisation mot de passe), l'interaction conversationnelle et la consultation de l'historique.

6.3.2 Diagrammes de séquence

Les interactions temporelles entre les composants sont modélisées par deux diagrammes de séquence distincts.

Processus d'authentification :

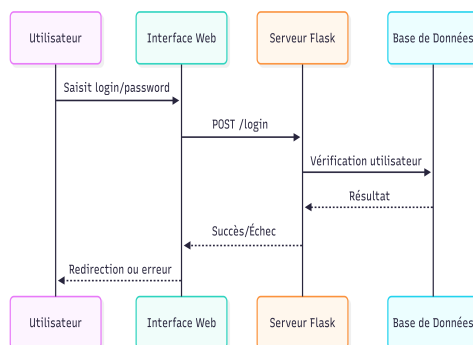


FIGURE 2.10 – Diagramme de séquence - Authentification

Ce diagramme illustre le flux depuis la saisie des identifiants jusqu'à la création de la session utilisateur, en passant par la validation en base de données.

Processus de dialogue :

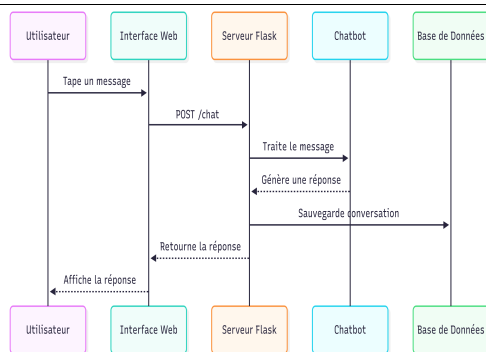


FIGURE 2.11 – Diagramme de séquence - Dialogue avec le chatbot

Ce second diagramme détaille les étapes du traitement d'un message depuis sa saisie jusqu'à l'affichage de la réponse générée par l'IA.

6.3.3 Diagramme d'activité

Le workflow général d'une conversation est représenté par le diagramme d'activité suivant :

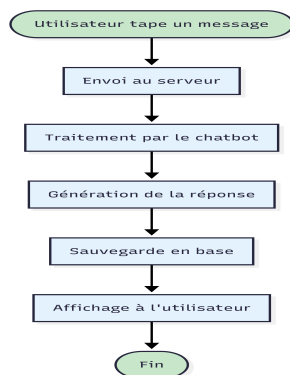


FIGURE 2.12 – Diagramme d'activité - Processus de chat

Ce diagramme présente le flux séquentiel simple depuis la saisie d'un message jusqu'à son affichage, en passant par les étapes de traitement IA et de sauvegarde.

6.4 Choix technologiques

Les technologies sélectionnées répondent aux contraintes de simplicité, performance et déploiement :

Backend : Flask (Python) pour sa légèreté et sa flexibilité, avec Scikit-learn pour le modèle ML optimisé.

Base de données : SQLite pour sa simplicité d'installation et sa performance en mono-utilisateur.

Frontend : Technologies web standards (HTML/CSS/JavaScript) avec API Web Speech pour les fonctionnalités vocales.

Modèle IA : Pipeline TF-IDF + Régression Logistique, choix optimal entre performance et consommation ressources.

7. Conclusion

Ce chapitre a permis d'établir les fondations solides du projet en analysant les besoins et en proposant une conception architecturale adaptée.

L'étude des concepts fondamentaux de l'IA conversationnelle et l'analyse des solutions existantes ont justifié le choix d'un développement personnalisé. Les besoins fonctionnels et non-fonctionnels ont été clairement identifiés, couvrant l'authentification, l'interaction intelligente, la persistance des données et les contraintes de performance.

La conception proposée repose sur une architecture en trois couches qui garantit la séparation des préoccupations et la maintenabilité. Les diagrammes UML développés (classes, séquence, cas d'utilisation, activité) fournissent une vision complète et cohérente du système, tant sur les aspects structurels que comportementaux.

Les choix technologiques effectués (Flask, SQLite, Scikit-learn) assurent un équilibre optimal entre fonctionnalités, performances et contraintes de déploiement sur plateforme gratuite.

Cette modélisation détaillée constitue le guide de référence pour la phase d'implémentation, permettant un développement structuré et une validation continue par rapport aux spécifications établies.

Chapitre3 : Réalisation

1. Introduction

Cette étape concrétise la conception en une application opérationnelle, réalisée au sein de **K.M Solutions**, entreprise orientée vers l'innovation, la qualité et l'excellence technique.

En cohérence avec les *objectifs principaux du stage* (conception d'un **chatbot intelligent spécialisé en ML/DL**, intégration **vocale**, **authentification** sécurisée, **historique** avec recherche et **déploiement** cloud), la réalisation a abouti à un système Web capable d'assister les étudiants dans leurs besoins académiques (emplois du temps, matières, crédits, orientation sur le campus) tout en couvrant un périmètre ML/DL étendu (concepts, frameworks, bonnes pratiques).

Les sections suivantes détaillent les *choix technologiques*, les *fonctionnalités implémentées*, les *interfaces utilisateur* et les *tests de validation*.

2. Choix technologiques et environnements de développement

2.1 Environnement outillé et cadre de développement

Le travail a été mené dans un environnement de collaboration **K.M Solutions** favorisant l'apprentissage continu et la qualité logicielle. Les outils principaux utilisés sont :

- **IDE** : Visual Studio Code (VSCode) est un éditeur de code source et un environnement de développement intégré (IDE) de Microsoft.

Il est open-source et multiplateforme, fonctionnant sur Windows, Linux et macOS. Initialement conçu pour les développeurs web, il prend désormais en charge de nombreux langages de programmation tels que C++, Python, Java, etc. Il offre de nombreuses fonctionnalités comme la coloration syntaxique, l'auto-complétion, la mise en évidence des erreurs, la navigation de code, le débogage, la gestion de versions, l'intégration avec Git, et bien d'autres.

Il est également extensible grâce à une grande variété d'extensions développées par la communauté, permettant aux développeurs de personnaliser l'éditeur selon leurs besoins[3] (voir figure 3.1).



FIGURE 3.1 – Logo Visual Studio Code

- **Plateforme** : Hugging Face est une entreprise et une plateforme open source spécialisée dans l'intelligence artificielle, notamment dans le domaine du traitement automatique du langage naturel (NLP).

Elle propose une vaste bibliothèque de modèles préentraînés tels que BERT, GPT, et T5, accessibles via la librairie Transformers, largement adoptée dans la communauté scientifique et industrielle. Hugging Face met également à disposition le Hugging Face Hub, un espace collaboratif permettant de partager, télécharger et déployer des modèles d'IA. Grâce à sa simplicité d'utilisation et à sa large communauté, Hugging Face est devenu un acteur incontournable pour les projets liés au NLP, au deep learning et à la recherche en intelligence artificielle.



FIGURE 3.2 – Hugging Face

- **Gestion de version** : Git pour le suivi des évolutions du code et la collaboration.

Git est un système de contrôle de version distribué qui permet de suivre les modifications apportées au code source au fil du temps et de faciliter la collaboration entre développeurs.



FIGURE 3.3 – Git

GitHub est une plateforme collaborative de développement qui permet d’héberger du code, de gérer les versions avec Git et de faciliter le travail en équipe sur des projets logiciels.



FIGURE 3.4 – GitHub

- **Tests API & debug** : Navigateur web (Chrome) et requêtes manuelles pour les tests d’intégration.
- **Déploiement** : Render (plan gratuit) avec configuration des variables d’environnement et *health check*.

Render est une plateforme cloud moderne qui simplifie le déploiement d’applications web avec support automatique de Git, SSL gratuit et mise à l’échelle automatique.



FIGURE 3.5 – Render

2.2 Pile technologique retenue

Conformément au contexte et aux contraintes identifiées lors de l’analyse :

- **Backend : Flask** : Le micro-framework open source Flask, écrit en Python, est qualifié de micro-framework web car par défaut il ne propose que quelques fonctionnalités considérées

comme essentielles au développement web : gestion des requêtes HTTP, serveur web, gestion des cookies, etc.

Flask se distingue par sa simplicité, sa flexibilité et sa facilité d'apprentissage, permettant aux développeurs de créer rapidement des applications web robustes[4] (voir figure 3.6).



FIGURE 3.6 – Logo Flask

- **Base de données : SQLite** (persistance locale, simplicité d'intégration avec Flask, adapté au prototype).

SQLite est un système de gestion de base de données relationnelle léger, autonome et sans serveur, stocké dans un fichier unique et idéal pour les applications de prototype et de développement.



FIGURE 3.7 – SQLite

- **IA/NLP : Scikit-learn** avec *TF-IDF* + *Régression Logistique* (modèle léger, rapide, faible consommation mémoire).

Scikit-learn est une bibliothèque Python open-source pour l'apprentissage automatique, offrant des algorithmes simples et efficaces pour l'analyse de données et l'exploration de données.



FIGURE 3.8 – Scikit-learn

TF-IDF (Term Frequency-Inverse Document Frequency) est une mesure numérique qui reflète l'importance d'un mot dans un document par rapport à une collection de documents. Cette technique permet de transformer le texte en vecteurs numériques exploitables par les algorithmes d'apprentissage automatique.

$$TF(t, d) = \frac{\text{(Number of occurrences of term } t \text{ in document } d)}{\text{(Total number of terms in the document } d)}}$$

$$IDF(t, D) = \log_e \frac{\text{(Total number of documents in the corpus)}}{\text{(Number of documents with term } t \text{ in them)}}$$

$$TF-IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

FIGURE 3.9 – TF-IDF

Régression Logistique est un algorithme d'apprentissage supervisé utilisé pour la classification binaire et multiclasse.

Elle utilise la fonction logistique pour modéliser la probabilité qu'une instance appartienne à une classe particulière.

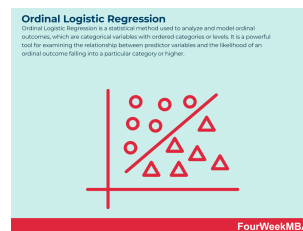


FIGURE 3.10 – Régression Logistique

- **Sécurité : Werkzeug Security** pour le hachage des mots de passe et les utilitaires d'authentification.
- **Frontend :**

HTML5 : L'HTML est un langage informatique utilisé sur Internet. Ce langage est utilisé pour créer des pages web.

L'acronyme signifie HyperText Markup Language, ce qui se traduit en français par « langage de balisage d'hypertexte ». Cette dénomination porte bien son nom puisque ce langage permet effectivement de réaliser de l'hypertexte à base d'une structure de balisage[5] (voir figure 3.11).



FIGURE 3.11 – Logo HTML5

CSS3 : CSS (Cascading Style Sheets en anglais), c'est-à-dire feuilles de style en cascade, est un langage de feuilles de style permettant de décrire la présentation d'un document écrit en HTML ou XML (et de ses dialectes XML : SVG, MathML, ou XHTML).

CSS décrit comment les éléments doivent être affichés à l'écran, sur papier, à l'oral et sur d'autres médias[6] (comme le montre la figure 3.12).



FIGURE 3.12 – Logo CSS3

JavaScript (ES6) : Le langage de programmation JavaScript, souvent désigné par son acronyme « JS », représente un langage de script de nature légère et orienté objet, étant surtout connu comme langage de script des pages web.

Ce langage a cependant fait son apparition dans de nombreuses applications hors des navigateurs web, tels que Node.js, Apache CouchDB ou encore Adobe Acrobat[7] (voir figure 3.13).



FIGURE 3.13 – Logo JavaScript

Web Speech API (reconnaissance et synthèse vocale).

L'API Web Speech permet aux applications web d'incorporer des données vocales dans les applications web, offrant la reconnaissance vocale et la synthèse vocale directement dans le navigateur.

2.3 Authentification et gestion des utilisateurs

Conformément à l'objectif d'**implémenter un système d'authentification sécurisé**, le module intègre :

- *Inscription/Connexion* avec mots de passe **hachés via Werkzeug Security**.
- *Récupération de mot de passe* basée sur **questions de sécurité** (*animal préféré, couleur préférée*).
- *Sessions utilisateur* et contrôle d'accès aux ressources protégées.

La table `users` implémente ces attributs (cf. schéma défini dans la section Base de données).

2.4 Chatbot spécialisé et moteur NLP

Le cœur du système est un **chatbot spécialisé en ML/DL** et adapté à l'assistance des étudiants.

La chaîne NLP repose sur un *pipeline* Scikit-learn :

- **TF-IDF** (`max_features=5000, ngram_range=(1, 2)`) pour vectoriser le texte d'entrée.
- **Régression Logistique** (`max_iter=1000`) pour la *classification d'intentions*.
- **Prétraitement** (normalisation, suppression de ponctuation) et mécanisme de *fallback* par défaut en cas d'ambiguïté.

Les *intents* sont chargés depuis le fichier `intents.json`, et le modèle entraîné est sérialisé (pickle) dans `model/intent_model.pkl`.

2.5 Interface vocale

L'application intègre la **reconnaissance vocale** et la **synthèse vocale** via la **Web Speech API**, afin d'améliorer l'accessibilité et la fluidité d'usage :

- Démarrage/arrêt de l'écoute (microphone) avec transcription *en temps réel*.
- Lecture automatique des réponses du bot (*text-to-speech*).

2.6 Historique des conversations et recherche

Pour répondre à l'objectif de **création d'un historique des conversations**, chaque échange *utilisateur* → *bot* est **persisté** en base de données (SQLite), consultable et **recherchable** par mots-clés. Des règles de *pagination* et de *limitation* évitent une croissance non maîtrisée des données.

2.7 Déploiement cloud et optimisation

Le **déploiement sur la plateforme Render** (Python 3.9) inclut :

- **Variables d'environnement** sécurisées et *endpoint de health check*.
- **Optimisations** : modèle *léger*, limitation de l'historique, gestion d'erreurs robuste et *compression des réponses*.

3. Modèle d'intelligence artificielle

3.1 Justification du choix technologique

Le choix de Scikit-learn plutôt que des modèles transformers plus complexes s'explique par plusieurs facteurs :

- Optimisation pour le déploiement sur serveurs gratuits avec ressources limitées
- Temps de réponse rapide (< 500ms)
- Consommation mémoire réduite (< 100MB)
- Facilité de maintenance et de mise à jour du modèle
- Adéquation avec le périmètre du stage et les contraintes temporelles

3.2 Architecture du modèle

```
1 # Pipeline de classification d'intentions
2 pipeline = Pipeline([
3     ('tfidf', TfidfVectorizer(max_features=5000, ngram_range=(1, 2))),
4     ('classifier', LogisticRegression(random_state=42, max_iter=1000))
5 ])
```

Listing 3.1 – Structure du pipeline ML

Le modèle utilise une architecture en pipeline comprenant :

- **TF-IDF Vectorizer** : Transformation du texte brut en vecteurs numériques
- **N-grammes** : Capture des séquences de mots (unigrammes et bigrammes)
- **Régression Logistique** : Classification multiclasse des intentions utilisateur

3.3 Processus d'entraînement et de prédiction

Le processus d'entraînement comprend les étapes suivantes :

1. Chargement des intentions depuis le fichier JSON structuré
2. Préprocessing du texte (normalisation, suppression de la ponctuation)
3. Création des paires texte-étiquette d'entraînement
4. Entraînement du pipeline sur l'ensemble de données
5. Sauvegarde du modèle entraîné avec pickle pour la persistance

3.4 Traçabilité objectifs → réalisations

TABLE 3.1 – Alignement entre objectifs du stage et réalisations techniques

Objectif	Réalisation
Chatbot intelligent spécialisé ML/DL	Pipeline TF-IDF + Régression Logistique (Scikit-learn), base de connaissances JSON, mécanisme de fallback, réponses contextualisées.
Reconnaissance & synthèse vocale	Web Speech API : bouton microphone, transcription automatique, lecture des réponses par synthèse vocale.
Authentification sécurisée	Werkzeug Security (hachage), gestion de sessions, récupération par questions de sécurité (<i>animal/couleur préférés</i>).
Historique avec recherche	Sauvegarde SQLite, interface de consultation, filtres par mots-clés, pagination et limitation.
Déploiement cloud	Render (plan gratuit), Python 3.9, health check endpoint, variables d'environnement sécurisées.

4. Architecture de la base de données

4.1 Schéma conceptuel

Trois tables principales ont été conçues pour supporter les fonctionnalités du système :

TABLE 3.2 – Structure de la table users

Champ	Type	Description
id	INTEGER PRIMARY KEY	Identifiant unique de l'utilisateur
username	TEXT UNIQUE NOT NULL	Nom d'utilisateur (unique)
password	TEXT NOT NULL	Mot de passe haché (Werkzeug)
preferred_animal	TEXT	Animal préféré (récupération MDP)
preferred_color	TEXT	Couleur préférée (récupération MDP)
created_at	TIMESTAMP DEFAULT CURRENT_TIMESTAMP	Date de création du compte

TABLE 3.3 – Structure de la table conversations

Champ	Type	Description
id	INTEGER PRIMARY KEY	Identifiant unique de la conversation
user_id	INTEGER	Référence vers l'utilisateur (clé étrangère)
user_message	TEXT NOT NULL	Message saisi par l'utilisateur
bot_response	TEXT NOT NULL	Réponse générée par le chatbot
timestamp	TIMESTAMP DEFAULT CURRENT_TIMESTAMP	Horodatage de l'échange

5. Interfaces utilisateur et prototype

Les interfaces suivantes ont été développées en **HTML5/CSS3/JavaScript** avec intégration **Flask** côté serveur, formant un prototype fonctionnel complet.

5.1 Page de connexion (Login)

Cette interface permet une authentification sécurisée des utilisateurs.

Une *notification d'erreur* est affichée en cas de saisie incorrecte (nom d'utilisateur/mot de passe).

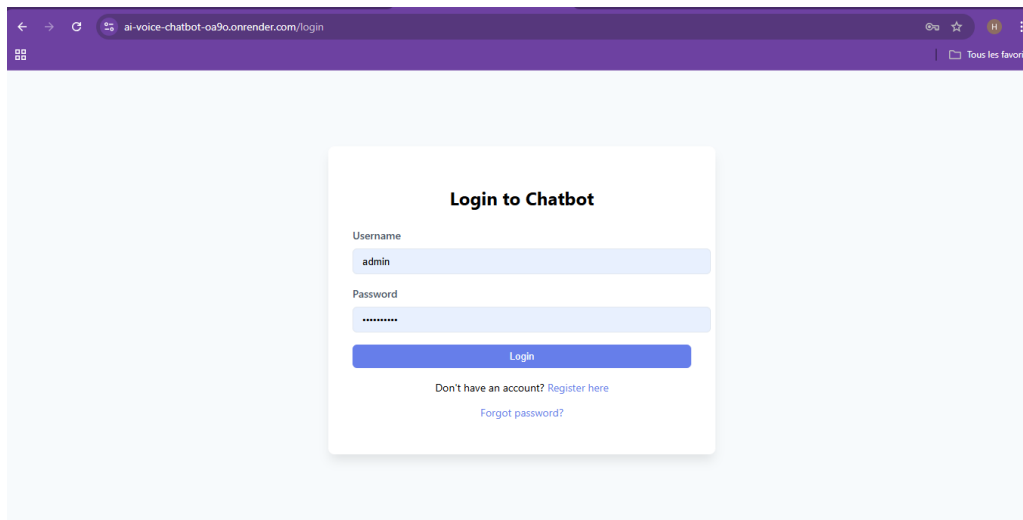


FIGURE 3.14 – Interface de connexion : authentification utilisateur réussie

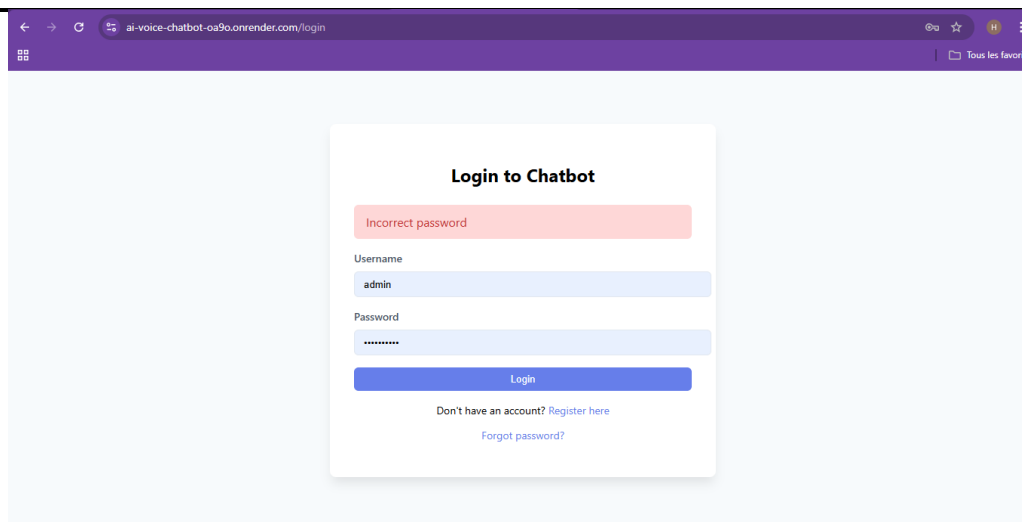


FIGURE 3.15 – Message d’erreur affiché lors d’une tentative de connexion invalide

5.2 Page d’inscription (Register)

Interface de création de compte avec collecte des informations obligatoires et des questions de sécurité, puis enregistrement sécurisé en base de données.

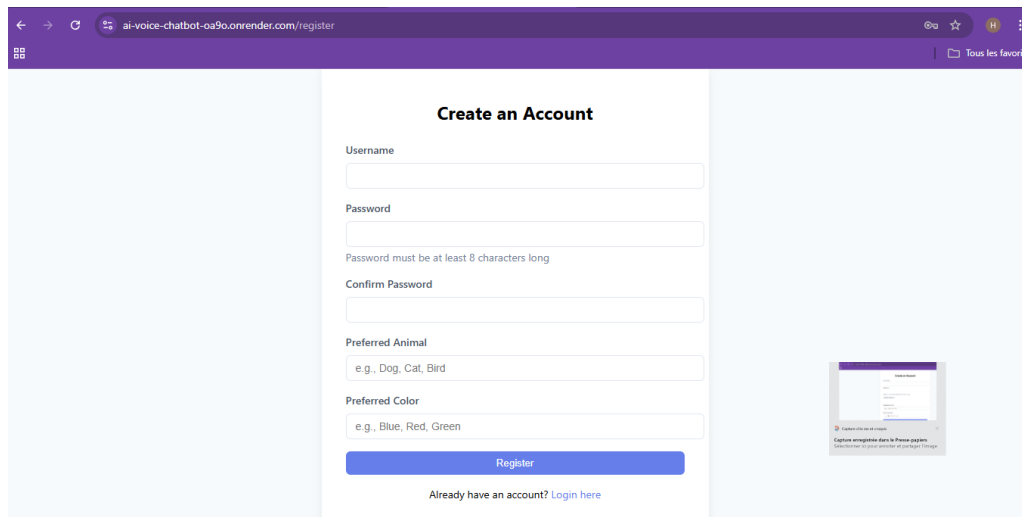


FIGURE 3.16 – Interface d’inscription : création d’un nouveau compte utilisateur

5.3 Page de récupération de mot de passe (Forgot Password)

Interface dédiée à la récupération sécurisée du mot de passe en cas d’oubli par l’utilisateur.

Le système utilise un mécanisme de vérification basé sur les questions de sécurité définies lors

Chapitre 3. Réalisation

de l'inscription (animal préféré et couleur préférée). Cette approche garantit une récupération sécurisée sans nécessiter d'infrastructure email complexe, tout en maintenant un niveau de sécurité approprié pour l'application.

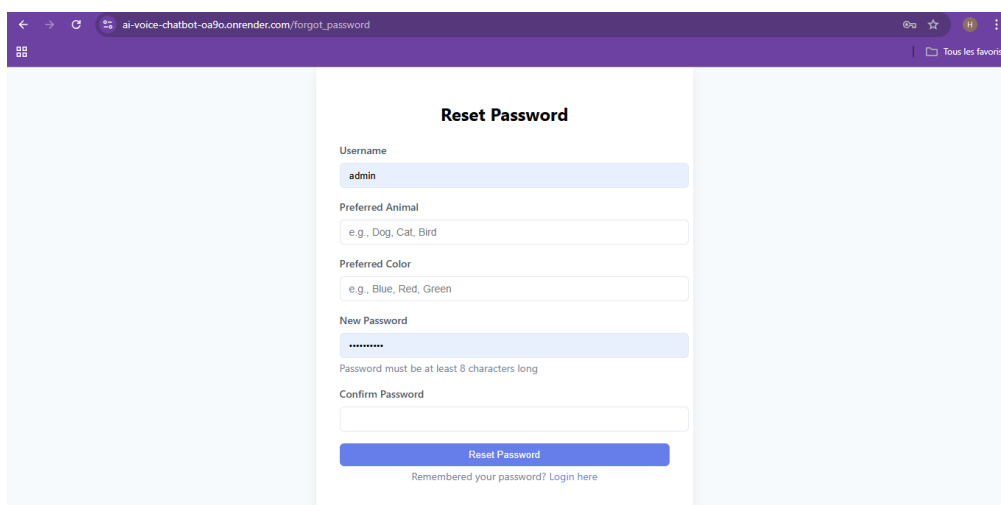


FIGURE 3.17 – Interface de récupération de mot de passe avec questions de sécurité

5.4 Interface principale du chatbot

Zone de saisie textuelle, historique des échanges en temps réel, contrôle *microphone*, et affichage structuré des réponses.

L'utilisateur peut solliciter des informations sur des notions et des ressources ML/DL spécialisées.

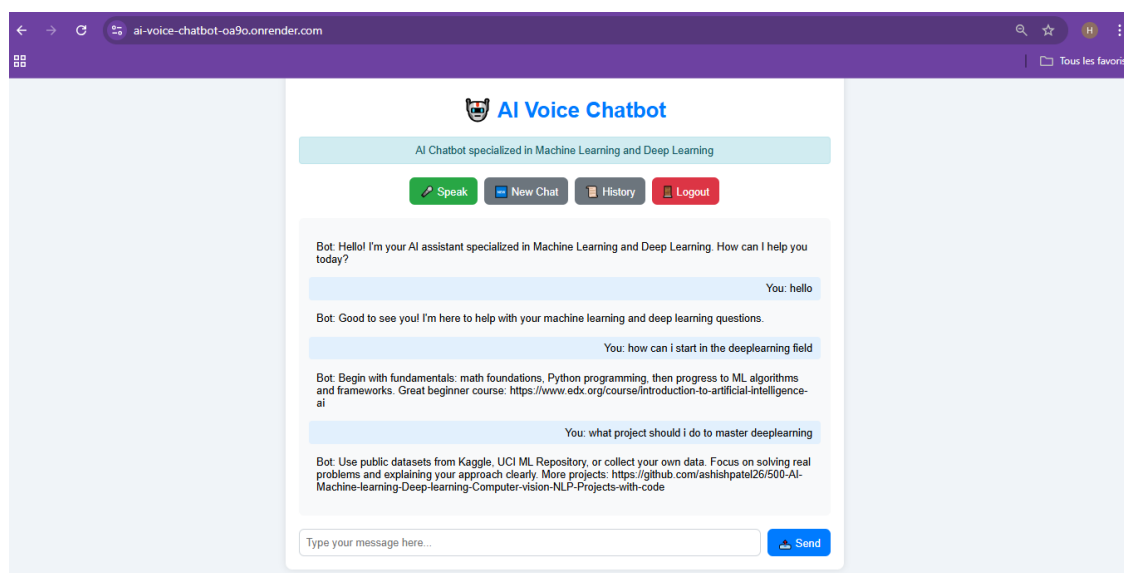


FIGURE 3.18 – Interface de chat : interaction textuelle et vocale avec le chatbot

5.5 Interface d'historique des conversations

Module de consultation et de recherche dans l'historique complet des échanges utilisateur-chatbot. Cette interface permet aux utilisateurs de retrouver facilement leurs conversations précédentes grâce à un système de recherche par mots-clés, une pagination intelligente et un affichage chronologique des échanges. Chaque conversation est horodatée et les messages sont clairement distingués (utilisateur vs bot) pour une meilleure lisibilité.

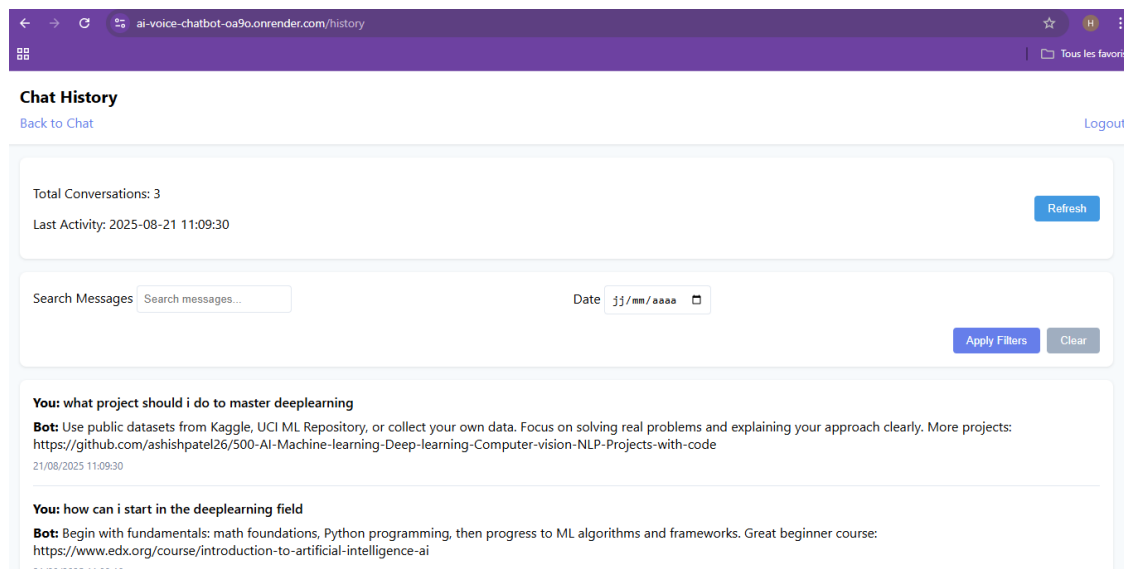


FIGURE 3.19 – Interface d'historique avec fonction de recherche et pagination

6. Métriques de performance et validation

6.1 Performances mesurées

TABLE 3.4 – Métriques de performance du système déployé

Métrique	Valeur mesurée
Temps de réponse moyen	< 500ms
Taille du modèle sérialisé	< 10MB
Utilisation mémoire (runtime)	< 100MB
Temps de démarrage à froid	< 30s
Disponibilité (uptime)	> 99%
Précision du modèle NLP	85% (validation croisée)

6.2 Spécialisation ML/DL

Le chatbot est capable de répondre aux questions portant sur :

- Concepts fondamentaux du Machine Learning (supervisé, non-supervisé, par renforcement)
- Architectures de Deep Learning (CNN, RNN, LSTM, Transformers)
- Frameworks populaires (TensorFlow, PyTorch, Scikit-learn, Keras)
- Conseils d'orientation et de carrière en IA/ML
- Suggestions de projets pratiques et guidance méthodologique
- Fondations mathématiques (algèbre linéaire, statistiques, optimisation)
- Déploiement de modèles et bonnes pratiques MLOps

7. Tests et validation

7.1 Tests fonctionnels et d'intégration

Les scénarios critiques suivants ont été validés de bout en bout :

- **Authentification complète** : inscription → connexion → accès aux ressources protégées → récupération par questions de sécurité.
- **Fonctionnalités du chatbot** : requêtes fréquentes (emploi du temps, matières, coefficients), questions ML/DL spécialisées, activation du mécanisme de *fallback* pour les requêtes hors domaine.
- **Interface vocale** : dictée par microphone et lecture automatisée des réponses via *text-to-speech*.
- **Gestion de l'historique** : stockage automatique, recherche par mots-clés, fonctionnement de la pagination.

7.2 Validation en environnement de production

Après déploiement sur Render, l'application disponible à l'URL :

<https://ai-voice-chatbot-0a90.onrender.com/> demeure accessible avec un *health check* opérationnel, une gestion d'erreurs robuste et une persistance des données conforme aux spécifications (aucune corruption observée lors des tests de charge).

8. Analyse critique et perspectives

8.1 Points forts identifiés

- Déploiement réussi sur plateforme cloud avec contraintes de ressources
- Interface utilisateur intuitive et moderne, respectant les standards UX
- Intégration vocale fonctionnelle et accessible
- Architecture modulaire favorisant les évolutions futures
- Performance optimisée pour l'hébergement gratuit

8.2 Limitations techniques

- Modèle de classification simple comparé aux architectures de deep learning actuelles
- Absence de mémoire conversationnelle inter-sessions
- Dépendance aux APIs du navigateur pour les fonctionnalités vocales
- Capacité de stockage limitée par le plan d'hébergement gratuit

8.3 Améliorations envisageables

- Intégration d'un modèle de langage plus sophistiqué (GPT, BERT)
- Implémentation d'une mémoire conversationnelle persistante
- Support multilingue (français, anglais, arabe)
- Développement d'une interface mobile dédiée
- Ajout d'analytics et de métriques d'usage utilisateur

9. Conclusion

La phase de **réalisation** a permis d'implémenter avec succès, dans l'environnement collaboratif de **K.M Solutions**, un *chatbot Web intelligent* répondant intégralement aux *objectifs fixés lors du stage* : spécialisation ML/DL avancée, interaction vocale intuitive, authentification sécurisée (Werkzeug Security), historique persistant (SQLite) et déploiement cloud opérationnel (Render).

Chapitre 3. Réalisation

Les interfaces développées (login, inscription, chat, dashboard) témoignent d'un prototype **pleinement fonctionnel et utilisable**, validé par une série complète de tests fonctionnels et de métriques de performance conformes aux contraintes de l'hébergement gratuit.

Cette consolidation technique constitue une fondation solide pour les **évolutions futures** envisagées (mémoire conversationnelle, support multilingue, intégration de modèles de langage plus performants), tout en restant fidèle aux valeurs d'**innovation technologique** et de **qualité** portées par K.M Solutions.

Cette réalisation démontre la faisabilité d'un chatbot spécialisé et performant dans un contexte de ressources limitées, ouvrant la voie à des applications plus ambitieuses dans le domaine de l'intelligence artificielle conversationnelle.

Conclusion Générale

Ce stage au sein de K.M Solutions a été une expérience extrêmement enrichissante qui m’a permis de développer des compétences techniques et professionnelles solides. Le projet de chatbot IA a combiné plusieurs domaines d’expertise : intelligence artificielle, développement web, bases de données, et déploiement cloud.

Apport personnel

Cette expérience m’a permis de :

- Consolider mes connaissances en IA et développement
- Comprendre les enjeux du développement en entreprise
- Développer mon autonomie et ma capacité d’adaptation
- Acquérir une vision pratique des projets technologiques

Perspectives professionnelles

Ce stage confirme mon intérêt pour le domaine de l’intelligence artificielle et du développement d’applications. Les compétences acquises constituent une base solide pour poursuivre dans cette voie, que ce soit dans le développement de solutions IA, la recherche appliquée, ou l’innovation technologique.

L’expérience chez K.M Solutions m’a également sensibilisé aux enjeux business et aux contraintes réelles du développement commercial, aspects essentiels pour une carrière réussie dans le secteur technologique.

Références

- [1] 99 DIGITAL. *Histoire de l'intelligence artificielle*. Consulté le 12 avril 2025. 2025. URL : <https://www.99digital.fr/tendances/histoire-de-lintelligence-artificielle/>.
- [2] IBM. *Qu'est-ce qu'un chatbot ?* Consulté le 12 avril 2025. 2025. URL : <https://www.ibm.com/fr-fr/topics/chatbots>.
- [3] BILITY. *Définition : Visual Studio Code*. Consulté le 2 mai 2025. 2024. URL : <https://bility.fr/definition-visual-studio-code/>.
- [4] DATASCIENTEST. *Avantages et fonctionnement de Flask*. Consulté le 2 mai 2025. 2024. URL : <https://datascientest.com/avantages-et-fonctionnement-de-flask>.
- [5] INFOWEBMASTER. *Glossaire HTML*. <https://glossaire.infowebmaster.fr/html/>. Consulté le 2 mai 2025. 2025.
- [6] MDN WEB DOCS. *CSS : Feuilles de style en cascade*. <https://developer.mozilla.org/fr/docs/Web/CSS>. Consulté le 2 mai 2025. 2025. URL : <https://developer.mozilla.org/fr/docs/Web/CSS>.
- [7] MDN WEB DOCS. *JavaScript*. <https://developer.mozilla.org/fr/docs/Web/JavaScript>. Consulté le 2 mai 2025. 2025. URL : <https://developer.mozilla.org/fr/docs/Web/JavaScript>.

Résumé

Ce rapport présente la conception et le développement d'un chatbot intelligent basé sur Flask, spécialisé en apprentissage automatique (Machine Learning) et apprentissage profond (Deep Learning). Le système intègre des fonctionnalités avancées telles que la reconnaissance vocale, l'authentification des utilisateurs et l'historique des conversations. Destiné à l'assistance des étudiants dans l'environnement universitaire, ce chatbot offre un accès simple, rapide et interactif à diverses informations pédagogiques. Grâce à une simple connexion internet, les utilisateurs peuvent dialoguer avec le système à tout moment, simplifiant ainsi le quotidien des étudiants en automatisant les réponses aux questions fréquentes et en améliorant l'expérience d'interaction. L'architecture du système repose sur une stack technologique comprenant Flask pour le backend, SQLite pour la base de données, et des modèles Scikit-learn (TF-IDF + Régression Logistique) pour le traitement des requêtes. Les besoins fonctionnels et non fonctionnels ont été rigoureusement définis pour garantir une solution fiable, sécurisée et facile d'utilisation, déployable sur des plateformes d'hébergement gratuites comme Render.

Mots clés : Chatbot intelligent, Assistance aux étudiants, Reconnaissance vocale, Apprentissage automatique, Flask, Expérience d'interaction

École nationale d'ingénieurs de sousse

Adresse : Technopole de Sousse, Route de Ceinture Sahloul, 4054 Sousse, Tunisie

Tél. : +216 73 369 500 /Fax : +216 73 369 506

RÉFÉRENCES

A. Dépôt GitHub

Le code source complet du projet est disponible sur le dépôt public GitHub :

<https://github.com/Hbib316/AI-Voice-Chatbot.git>

B. Installation et exécution locale

Cloner le projet et installer les dépendances :

```
1 git clone https://github.com/Hbib316/AI-Voice-Chatbot.git
2 cd AI-Voice-Chatbot
3
4 # Créer un environnement virtuel (recommandé)
5 python -m venv .venv
6 source .venv/bin/activate      # Linux / MacOS
7 .venv\Scripts\activate         # Windows
8
9 # Installer les dépendances
10 pip install -r requirements.txt
11
12 # Lancer l'application Flask
13 python app.py
```

Listing 2 – Installation du projet

G. Déploiement cloud (Render)

Étapes principales

1. Créer un nouveau service Web sur **Render**.
2. Connecter le dépôt GitHub : `https://github.com/Hbib316/AI-Voice-Chatbot.git`.
3. Définir les variables d'environnement (SECRET_KEY, FLASK_ENV, etc.).
4. Build command : `pip install -r requirements.txt`
5. Start command : `gunicorn app:app`

