Burger QuizBot

Projet d'Informatique

M. LENORAIS | J. GOMIS | N. BEN OTHMAN
J. DEWAREZ
1 January 2026

1. Document décrivant le thème et l'approche

QuizBot - Plateforme de Culture Générale Interactive

Thème : Le projet consiste en un Chat-bot de Jeu de Quiz. Il s'agit d'une application console interactive permettant aux utilisateurs de tester leurs connaissances sur divers thèmes (Histoire, Sport, Sciences, etc.). Le système gère la progression des joueurs via un système de score et d'historique.

Approche Technique : Pour répondre aux exigences de robustesse et d'extensibilité, nous avons adopté une architecture modulaire :

- Séparation Données/Code : Le code ne contient pas les questions en dur. Elles sont importées depuis un fichier texte (*questions.txt*) facile à éditer, puis converties en binaire (*.dat*) pour une performance optimale à l'exécution.
- Expérience Utilisateur : Pour éviter la frustration liée aux entrées textuelles strictes, nous avons implémenté un algorithme de Logique Floue basé sur la distance de Levenshtein. Cela permet au bot de comprendre les réponses même avec des fautes de frappe ou des accents manquants.
- Robustesse des Fichiers : Un système de sauvegarde atomique (écriture dans *.tmp* puis renommage) a été mis en place pour éviter toute corruption de données en cas d'arrêt brutal du programme.

2. Nos notes personnelles

Mathis LENORAIS-CLERE:

Rôle : Concepteur & Communication

Pour ce projet de semestre, je me suis occupé de l'identité globale de notre chatbot et de l'aspect utilisateur. Mon travail a d'abord consisté à imaginer le thème « Burger Quiz », qui nous semblait idéal pour créer un jeu interactif.

Concrètement, j'ai défini toute la mécanique de jeu : j'ai choisi les différents thèmes des questions (culture, sport, science...), mis en place le système de points et créé les trois niveaux de difficulté pour que le quiz soit accessible à tous. Pour que l'équipe technique puisse avancer sur une base solide, j'ai aussi rédigé le cahier des charges fonctionnel qui détaillait toutes les fonctionnalités attendues.

Bien que mon rôle principal soit axé sur la conception, j'ai voulu m'impliquer dans la production du code aux côtés de Neil et Julien. J'ai donc aidé à l'implémentation de certaines fonctions du bot et j'ai participé activement aux phases de débugage. Cette polyvalence a été très enrichissante, car elle m'a permis de voir que certaines de mes idées initiales représentaient de vrais défis techniques une fois passées en langage C.

Julien GOMIS:

Rôle : Architecte technique & Qualité

Pour ce projet, ma mission principale était de m'assurer que notre chatbot soit solide, bien organisé et qu'il ne plante pas au moindre bug.

Côté architecture, j'ai travaillé sur la structure modulaire du programme. Avant même de lancer le gros du développement, j'ai défini les fichiers d'en-tête (.h) et les prototypes des fonctions. L'objectif était d'avoir une séparation nette entre la gestion des données (fichiers), la logique du quiz et l'interface utilisateur. Cela nous a permis de travailler en équipe sans que le code ne devienne un fouillis illisible.

Au-delà de l'organisation, j'ai pris en charge le développement des briques les plus complexes du programme. C'est moi qui ai codé les fonctions de bas niveau pour la gestion des fichiers binaires et l'importation du texte, ainsi que l'algorithme de logique floue (distance de Levenshtein). C'était un vrai défi technique d'arriver à ce que le bot comprenne une réponse même avec des fautes de frappe ou des accents manquants.

Enfin, j'ai supervisé l'assemblage final des différents modules pour sortir l'exécutable opérationnel pour la soutenance. Ce rôle m'a vraiment fait prendre conscience qu'une bonne conception en amont est indispensable pour réussir l'intégration d'un projet.

Neil BEN OTHMAN:

Rôle : Développeur & Suivi du projet

En tant que développeur principal, mon but était de traduire nos idées et les spécifications de l'architecte en un programme C qui fonctionne vraiment. Je me suis donc concentré sur le codage du « cœur » du Burger QuizBot.

Côté programmation, je me suis concentré sur la mise en place de la structure de jeu. J'ai codé la boucle principale du quiz pour que les questions s'enchaînent bien et que les scores soient calculés correctement. Je me suis

aussi occupé des interactions de base avec l'utilisateur et de l'affichage des menus, pour que l'interface soit simple et claire. Mon but était de rendre le programme fluide et facile à utiliser pour le joueur.

En plus du code, j'étais responsable du suivi du projet pour qu'on respecte bien la date limite du 8 janvier. C'est également moi qui me suis chargé de créer le diaporama pour notre présentation finale, afin de bien mettre en valeur tout ce qu'on a réalisé. Ce projet m'a vraiment permis de devenir plus à l'aise avec le C, plus particulièrement des notions complexes qu'on a pu mettre en pratique concrètement.

3. Documentation d'Installation et d'Utilisation

Installation

1. Pré-requis : Un compilateur C standard (testé sur Linux et MacOS).
2. Fichiers nécessaires :
   - quizbot.c
   - quizbot.h
   - questions.txt
3. Compilation : Ouvrir un terminal dans le dossier du projet et exécuter la commande :

   ```
   gcc -std=c11 -Wall -Wextra -o quizbot quizbot.c
   ```
4. Initialisation : Au premier lancement, le programme lira automatiquement *questions.txt* pour créer la base de données binaire *questions.dat*.

Guide Utilisateur

1. Lancement : Exécuter **`./quizbot`** (Linux/Mac)
2. Connexion : Entrez votre pseudo.
   - *Note : Le premier utilisateur créé devient automatiquement Administrateur.*
3. Menu Principal :
   - *Jouer :* Lance une série de 20 questions.
   - *Historique :* Affiche vos dernières parties.
   - *Classement :* Affiche le Top 10 des meilleurs scores.
4. En Jeu : Répondez aux questions. Vous pouvez taper "Vrai", "V", "Faux", "F", ou la réponse en toutes lettres. Le système tolère les petites fautes.

5. Administration (Si Admin) : Accessible via le choix 4 du menu. Permet de supprimer des utilisateurs ou d'ajouter de nouvelles questions dynamiquement.

4. Architecture Logicielle

L'application est structurée en deux fichiers principaux respectant le standard C :

1. Interface (quizbot.h) : Définit les constantes, les énumérations et les structures de données.

2. Implémentation (quizbot.c) :

- Module Fichiers (File System) :
    - *load_all_records* / *safe_rewrite_file* : Fonctions génériques utilisant void* pour gérer n'importe quel type de structure.
- Module Logique Floue (Core Logic) :
    - *calculate_levenshtein* : Algorithme mathématique de distance de mots.
    - *check_answer_fuzzy* : Compare l'entrée utilisateur avec les réponses attendues en appliquant une tolérance dynamique.
- Module Interface (UI) :
    - *show_player_menu* / *show_admin_menu* : Boucles d'interaction principales.
    - *start_game* : Moteur du quiz (Sélection -> Mélange -> Interrogation -> Sauvegarde).

5 & 6. Cahier de Recette (Exigences & Résultats)

ID	Exigence du Projet	Statut	Commentaire technique
1	Langage C	OK	Code en C pur.
2	Sauvegarde Conversations	OK	Sauvegarde des résultats et timestamps dans *historique.dat*.
3	BDD Utilisateurs	OK	Fichier *utilisateurs.dat* (Pseudo + Rôle Admin + HighScore).
4	BDD Historique	OK	Fichier *historique.dat* (Liaison User <-> Score/ Date).
5	BDD Réponses (Questions)	OK	Fichier *questions.dat* généré depuis *questions.txt*.
6	2 Modes (Admin/ User)	OK	Menu conditionnel basé sur le flag *is_admin*.
7	Admin : Gestion Users	OK	Fonction *delete_user_account* implémentée.
8	Admin : Gestion Réponses	OK	Fonction *add_new_question* permet l'ajout dynamique sans recompilation.
9	Interaction Utilisateur	OK	Saisie sécurisée (fgets) et interprétation intelligente.
10	Accès Historique	OK	Fonction *show_user_history* disponible pour chaque joueur.