**SYSTÈME DE RECOMMANDATION DES FICHIERS MULTIMÉDIA**

# **EXPLICATION DU PROJET**

**1. INTRODUCTION**

Ce projet vise à développer un **système de recommandation de fichiers multimédias** permettant aux utilisateurs d’accéder facilement à des films, séries, musiques, livres et podcasts en fonction de leurs préférences et de leurs interactions passées. Développé en **langage C**, Ce programme permettra aux utilisateurs de créer un compte, de se connecter, de consulter des fichiers multimédia. Il permettra également de mettra en œuvre des **algorithmes de recommandation avancés** en utilisant des **structures de données optimisées** (listes chaînées, arbres binaires) pour assurer une recherche rapide et une recommandation pertinente basées sur leurs préférences et interactions.

**2. FONCTIONNALITES PRINCIPALES**

**2.1. Gestion des utilisateurs**

* **Création de compte** :
  + L'utilisateur pourra s'inscrire en fournissant des informations personnelles (nom, email, mot de passe, préférences de contenu).
  + Les informations seront stockées dans une base de données.
* **Connexion** :
  + Un utilisateur pourra se connecter à l'aide de son email et mot de passe.
  + Authentification et gestion de session.

**Outils recommandés :**

* Langage C pour le développement.
* SQLite ou MySQL pour la gestion de base de données.
* Bibliothèques C pour le chiffrement des mots de passe (OpenSSL, bcrypt).

**2.2. Accès aux fichiers multimédias**

* Une fois connecté, l'utilisateur pourra accéder à une sélection de fichiers multimédias.
* Chaque fichier multimédia sera accompagné d'une description et d'un lien permettant d'y accéder sur une plateforme externe (Netflix, Spotify, YouTube, etc.).

**Outils recommandés :**

* API pour récupérer des informations sur les fichiers multimédias (OMDb API pour les films, Spotify API pour la musique, Google Books API pour les livres).
* CURL en C pour interagir avec les API.

**2.3. Stockage et gestion des fichiers multimédias**

* Base de données structurée pour stocker les fichiers multimédias avec leurs attributs (titre, genre, lien, description, note moyenne).
* Possibilité de mettre à jour la base de données de manière centralisée, permettant aux machines connectées à Internet d'accéder aux mises à jour en temps réel.

**Outils recommandés :**

* Base de données relationnelle (SQLite, MySQL).
* Structuration des données avec des tables normalisées.
* Serveur distant pour la synchronisation des données (AWS, Firebase, Serveur Web personnel).

**2.4. Système de notation et d'avis**

* Les utilisateurs pourront donner leur avis et noter les fichiers multimédias qu'ils ont consultés.
* Une moyenne des avis sera affichée sur chaque fichier multimédia.

**Outils recommandés :**

* Stockage des avis dans la base de données.
* Interface utilisateur permettant de laisser un commentaire et une note.

**2.5. Algorithme de recommandation**

* **Recommandation basée sur les recherches** : Si un utilisateur recherche souvent un genre particulier, des fichiers similaires lui seront recommandés.
* **Recommandation basée sur les avis** :
  + Avis positifs = recommandation de contenus similaires.
  + Avis négatifs = suppression de ce type de contenu des recommandations.
* **Recommandation collaborative** : Les utilisateurs ayant des profils similaires recevront des recommandations communes.
* **Génération automatique de playlists** : Pour les contenus audio (musiques, podcasts), des playlists seront créées en fonction des préférences.
* **Mise à jour automatique des recommandations** : Les informations et préférences des utilisateurs seront stockées sur un serveur centralisé, permettant d'ajuster les recommandations des autres utilisateurs connectés en temps réel.

**Outils recommandés :**

* Algorithmes de filtrage collaboratif et de filtrage basé sur le contenu.
* Analyse de données en C avec des structures de données (listes chaînées, arbres binaires).
* Serveur centralisé pour la gestion et la mise à jour des préférences utilisateurs.

**3. INTERFACE GRAPHIQUE**

**3.1. Page d'inscription**

* Formulaire de création de compte pour enregistrer les utilisateurs.
* Champs : Nom, Email, Mot de passe, Préférences.

**3.2. Page de connexion**

* Formulaire permettant de se connecter avec Email/Mot de passe.
* Gestion des erreurs (mot de passe incorrect, utilisateur non inscrit).

**3.3. Page d'accueil**

* Liste de fichiers multimédias classés par catégorie.
* Recommandations personnalisées mises à jour dynamiquement en fonction des données stockées sur le serveur.

**3.4. Barre de recherche**

* Permet de rechercher un fichier multimédia par titre, auteur ou genre.

**3.5. Page de détails d'un fichier multimédia**

* Description, moyenne des avis, espace pour donner un avis et lien d'accès.

**Outils recommandés :**

* GTK ou Qt pour développer une interface graphique en C.

**4. CONCLUSION**

Ce projet a pour but de créer une plateforme intuitive permettant aux utilisateurs d'accéder à des fichiers multimédias et d'obtenir des recommandations adaptées à leurs goûts. En combinant le langage C avec une base de données et une interface graphique, il offrira une expérience fluide et personnalisée.

**Technologies clés :**

|  |  |
| --- | --- |
| **Fonctionnalité** | **Outils recommandés** |
| Langage de programmation | C |
| Base de données | SQLite (local) ou MySQL (serveur) |
| API pour les fichiers multimédias | OMDb , Spotify, Google Books |
| Bibliothèques pour API | libcurl, cJSON |
| Sécurisation des mots de passe | OpenSSL, bcrypt |
| Interface utilisateur | GTK ou Qt |
| Algorithme de recommandation | Listes chaînées, arbres binaires, filtrage collaboratif |
| Hébergement des données | MySQL sur AWS ou Firebase |
| Débogage | GDB, Valgrind |

Ce projet offre une solution complète et modulaire pour la recommandation de fichiers multimédias. Son développement suivra une approche progressive en intégrant chaque fonctionnalité de manière structurée.

# **. DETAIL DU PROJET**

## FICHIER DU PROJET

## L'organisation du projet en plusieurs fichiers

**Voici la structure des fichiers : Description détaillée des fichiers et leur rôle**

**📂 src/ *(Code source du projet)***

* **main.c** : Point d’entrée du programme, initialise l’interface et les modules.
* **user.c** : Gère l'inscription, connexion et stockage des préférences des utilisateurs.
* **media.c** : Permet d’accéder aux fichiers multimédias et de les stocker en base de données.
* **database.c** : Interagit avec SQLite/MySQL pour stocker et récupérer des données.
* **recommendation.c** : Implémente les algorithmes de recommandation (basé sur le contenu, collaboratif).
* **review.c** : Gère les avis et notes des utilisateurs.
* **api.c** : Requêtes vers des API externes pour récupérer des fichiers multimédias.
* **encryption.c** : Sécurise les mots de passe avec OpenSSL/bcrypt.
* **utils.c** : Fonctions générales de manipulation de chaînes, affichage, etc.

**📂 include/ *(Fichiers d’en-tête, prototypes de fonctions)***

Chaque fichier .h contient les déclarations des fonctions et structures de données correspondantes aux fichiers .c.

**📂 gui/ *(Interface graphique en GTK ou Qt)***

* **gui.c** : Initialise l'interface et gère la navigation entre fenêtres.
* **login\_window.c** : Fenêtre de connexion.
* **register\_window.c** : Fenêtre d’inscription.
* **home\_window.c** : Page principale affichant les recommandations.
* **media\_window.c** : Page permettant de consulter un fichier multimédia.
* **review\_window.c** : Page permettant aux utilisateurs de laisser un avis et noter un média.

**📂 data/ *(Stockage des bases de données et configurations)***

* **database.sqlite** : Base de données SQLite locale (si utilisée).
* **database.sql** : Script SQL pour initialiser la base de données MySQL.
* **config.json** : Contient les informations de connexion pour MySQL (si utilisé).

**📂 logs/ *(Logs pour le debugging)-(facultatif)***

* **error.log** : Stocke les erreurs détectées dans le programme.
* **access.log** : Journal des connexions des utilisateurs.

**📂 assets/ *(Ressources graphiques et médias)***

* **icons/** : Icônes pour l'interface utilisateur.
* **fonts/** : Polices personnalisées pour l’interface graphique.

**📂 tests/ *(Fichiers de tests unitaires)-(facultatif)***

* **test\_user.c** : Vérifie le bon fonctionnement de la gestion des utilisateurs.
* **test\_media.c** : Teste l’ajout et la récupération des fichiers multimédias.
* **test\_db.c** : Teste la connexion et les requêtes SQL.
* **test\_recommend.c** : Vérifie les algorithmes de recommandation.

**📂 docs/ *(Documentation du projet)-(facultatif)***

* **README.md** : Explication générale du projet.
* **INSTALL.md** : Instructions d’installation et de compilation.
* **USAGE.md** : Manuel d’utilisation.
* **API\_REFERENCE.md** : Documentation des API utilisées.

## ****3. BIBLIOTHEQUES A UTILISER****

Pour optimiser le projet et simplifier son développement, nous utiliserons plusieurs bibliothèques :

* **Gestion des bases de données** : sqlite3.h ou mysql.h
* **Gestion des API** : curl/curl.h, cjson/cJSON.h
* **Sécurité des mots de passe** : openssl/sha.h ou bcrypt.h
* **Interface graphique** : gtk/gtk.h (GTK) ou Qt
* **Gestion mémoire et debug** : valgrind, gdb
* **Manipulation des chaînes de caractères** : string.h
* **Gestion des fichiers** : stdio.h
* **Gestion des sockets et réseau** (si serveur distant) : sys/socket.h, arpa/inet.h

## ****4. FONCTIONS ET PROCEDURES****

|  |  |  |
| --- | --- | --- |
| **Fichier** | **Fonction / Procédure** | **Rôle** |
| user.c | register\_user(name, email, password, preferences) | Inscrit un nouvel utilisateur dans la base de données |
|  | login\_user(email, password) | Vérifie les identifiants et retourne l'ID utilisateur si succès |
|  | update\_preferences(user\_id, new\_preferences) | Met à jour les préférences de l'utilisateur |
| media.c | add\_media(title, genre, description, link) | Ajoute un fichier multimédia à la base |
|  | get\_media\_list() | Retourne la liste de tous les médias disponibles |
|  | search\_media(query) | Recherche un fichier multimédia par titre ou genre |
| database.c | init\_database() | Initialise la base de données SQLite/MySQL et crée les tables |
|  | execute\_query(query) | Exécute une requête SQL générique |
|  | insert\_user\_to\_db(name, email, password, preferences) | Insère un utilisateur dans la base |
|  | get\_all\_media() | Récupère tous les fichiers multimédias stockés |
| recommendation.c | recommend\_by\_preferences(user\_id) | Recommande des médias en fonction des préférences utilisateur |
|  | recommend\_by\_reviews(user\_id) | Recommande des médias basés sur les avis et notes des utilisateurs |
|  | recommend\_by\_collaborative\_filtering(user\_id) | Recommande des médias en fonction des profils similaires |
| review.c | add\_review(user\_id, media\_id, rating, comment) | Ajoute un avis utilisateur sur un média |
|  | get\_media\_rating(media\_id) | Calcule la moyenne des notes d’un média |
| api.c | fetch\_movie\_details(title) | Récupère des informations sur un film depuis OMDb API |
|  | fetch\_music\_details(artist) | Récupère des informations sur un artiste musical depuis Spotify API |
|  | fetch\_book\_details(title) | Récupère des informations sur un livre depuis Google Books API |
| encryption.c | hash\_password(password, hashed) | Génère un hash sécurisé du mot de passe |
|  | verify\_password(password, hashed) | Vérifie si un mot de passe correspond à son hash |
| utils.c | clear\_console() | Efface l’écran de la console |
|  | trim\_whitespace(str) | Supprime les espaces inutiles dans une chaîne |
|  | is\_valid\_email(email) | Vérifie si une adresse e-mail est valide |
| gui/ | **Interface Graphique** | **Navigation entre les Pages** |
|  | login\_window.c | Demande d’identifiants, redirige vers l’accueil après succès |
|  | register\_window.c | Inscription, puis redirection vers connexion |
|  | home\_window.c | Affiche les recommandations et la liste des médias |
|  | media\_window.c | Détails d’un fichier multimédia et ajout d’un avis |
|  | review\_window.c | Affichage et soumission des avis des utilisateurs |
| logs/ | log\_error(message) | Écrit un message d'erreur dans error.log |
|  | log\_access(user\_action) | Enregistre une action utilisateur dans access.log |
| sessions.c | create\_session(user\_id) | Crée une session utilisateur et retourne un token |
|  | verify\_session(session\_id) | Vérifie si une session est active |
|  | destroy\_session(session\_id) | Déconnecte un utilisateur et supprime sa session |