



ELMAGHOSSI ABDELMOUAIZ
OUSSAMA ACHIBAN

2024
SMI - S4

STUDENTHUB
STUDENTHUB

Faculté des sciences Semlalia

Département informatique

SMI-S4



كلية العلوم
السملاية - مراكش
FACULTÉ DES SCIENCES
SEMLALIA - MARRAKECH

RAPPORT PROJET

Application console de gestion des étudiants STUDENTHUB

PAR :

ABDELMOUAIZ ELMAGHOUSSE
OUSSAM ACHIBAN

DEPOSE LE 21 AVRIL 2024

ENCADRE PAR : Prof. RACHIDA HANNANE (responsable du module)

2023-2024

Introduction

Le projet StudentHub est une initiative développée par deux étudiants de la filière Sciences Mathématiques et Informatique (SMI) de la Faculté des Sciences Semlalia, sous la supervision de la responsable du module, Mme Rachida Hannane. Ce projet vise à fournir une plateforme de gestion des étudiants pour les établissements scolaires, offrant une gamme de fonctionnalités pour simplifier et rationaliser les processus administratifs liés à la gestion des étudiants.

Le développement de StudentHub est né de la nécessité de moderniser et d'optimiser la gestion des données des étudiants au sein des établissements scolaires. En tant que solution informatique, il offre des outils permettant de centraliser les informations des étudiants, de suivre leur parcours académique et d'automatiser certaines tâches administratives, contribuant ainsi à améliorer l'efficacité et la productivité des institutions éducatives.

Ce rapport présentera en détail le projet StudentHub, en mettant en lumière ses objectifs, sa conception, ses fonctionnalités principales ainsi que son implémentation. Nous examinerons également les défis rencontrés lors du processus de développement, les solutions apportées et les perspectives d'amélioration pour l'avenir.

Le projet StudentHub est structuré autour d'un ensemble de fonctionnalités clés, telles que la gestion des étudiants, la consultation des données, le tri, la recherche et la suppression d'étudiants, ainsi que l'authentification sécurisée des utilisateurs. Chacune de ces fonctionnalités contribue à offrir une expérience utilisateur fluide et efficace, tout en garantissant la confidentialité et l'intégrité des données.

Dans les sections suivantes, nous explorerons en détail chaque aspect du projet StudentHub, en mettant en évidence ses spécifications techniques, son architecture logicielle, ses interfaces utilisateur et son processus de développement. Nous examinerons également les résultats obtenus, les retours d'expérience des utilisateurs et les recommandations pour les futures itérations du projet.

Sommaire

Introduction.....	3
Sommaire.....	4
Analyse des besoins.....	5
Conception.....	7
Implémentation du Programme de Gestion des Étudiants en Langage C.....	8
Méthodologie de Traitement du StudentHub.....	10
Restructuration du Programme avec Séparation des Fichiers .h et .c.....	12
Utilisation de l'application STUDENTHUB.....	14
Défis Rencontrés lors de la Restructuration du Programme.....	16
Partitionnement et Phases de Développement du Projet.....	18
Problématique MAKEFILE	25
Conclusion.....	27
Remerciements.....	29

Fin

Analyse des besoins

Dans cette section, nous examinerons en détail les besoins du projet StudentHub, en mettant l'accent sur les besoins fonctionnels et non fonctionnels, ainsi que sur l'analyse des acteurs et des cas d'utilisation.

2.1 Besoins fonctionnels

Les besoins fonctionnels du projet StudentHub sont les fonctionnalités spécifiques que le système doit offrir pour répondre aux exigences des utilisateurs.

Ces besoins sont directement liés aux actions et aux opérations que les utilisateurs pourront effectuer avec l'application. Voici une liste des principaux besoins fonctionnels identifiés :

Gestion des étudiants : Permettre l'ajout, la modification et la suppression des informations des étudiants, y compris les données personnelles, les résultats académiques, etc.

Génération de rapports : Permettre la génération de rapports sur les performances des étudiants, les statistiques de fréquentation, etc.

Authentification et autorisation : Assurer un système d'authentification sécurisé pour les utilisateurs, avec des niveaux d'autorisation appropriés en fonction des rôles (administrateur, enseignant, étudiant, etc.).

2.2 Besoins non fonctionnels

Les besoins non fonctionnels décrivent les exigences du système qui ne sont pas liées à des fonctionnalités spécifiques, mais qui sont essentielles pour garantir la performance, la sécurité et l'utilisabilité de l'application. Voici quelques-uns des besoins non fonctionnels identifiés pour le projet StudentHub :

Sécurité : Assurer la confidentialité des données des étudiants et la sécurité du système contre les accès non autorisés.

Performances : Garantir des temps de réponse rapides, même lors de l'accès simultané de plusieurs utilisateurs au système.

Extensibilité : Concevoir le système de manière à ce qu'il puisse être facilement étendu et amélioré à l'avenir pour prendre en charge de nouvelles fonctionnalités.

Convivialité : Offrir une interface utilisateur intuitive et conviviale pour faciliter l'utilisation du système par les utilisateurs finaux.

2.3 Analyse des acteurs et des cas d'utilisation

L'analyse des acteurs identifie les différentes entités qui interagiront avec le système, tandis que l'analyse des cas d'utilisation décrit les différentes actions que ces acteurs pourront entreprendre. Voici les principaux acteurs identifiés pour le projet StudentHub :

Administrateur : Responsable de la gestion globale du système, y compris la création et la gestion des comptes utilisateurs, la configuration des paramètres du système, etc.

Enseignant : Responsable de l'enseignement des cours, de la saisie des notes, de la consultation des données des étudiants, etc.

Les cas d'utilisation incluront des actions telles que la connexion au système, la gestion des informations des étudiants, la consultation des notes, etc.

Une analyse détaillée de ces cas d'utilisation sera effectuée dans les sections suivantes.

Cette analyse des besoins fournira une base solide pour la conception et le développement ultérieurs du projet StudentHub, en assurant que toutes les exigences des utilisateurs sont prises en compte.

Conception

Dans cette phase, nous allons aborder la conception du projet StudentHub en nous concentrant sur l'architecture système, l'interface utilisateur et d'autres aspects pertinents. Il est à noter que le système est conçu pour s'exécuter sur la plateforme Windows, offrant ainsi une expérience utilisateur familière et compatible avec la majorité des utilisateurs cibles.

3.1 Architecture système

La conception de l'architecture système de StudentHub sera basée sur une approche client-serveur, où le client sera une application de bureau exécutée sur des machines windows.

Client (Application de bureau Windows) : L'interface utilisateur sera développée en utilisant des technologies conviviales pour Windows, offrant ainsi une expérience utilisateur intuitive et familière. Les fonctionnalités principales, telles que la gestion des étudiants, et la génération de rapports en fichier.txt seront accessibles depuis l'interface de l'application de bureau.

3.2 Interface Utilisateur

L'interface utilisateur de StudentHub sera conçue pour être conviviale et intuitive, en mettant l'accent sur la facilité d'utilisation pour les utilisateurs finaux. Voici quelques principes de conception qui guideront le développement de l'interface utilisateur :

Disposition claire et intuitive : Les fonctionnalités principales seront facilement accessibles depuis une disposition claire et intuitive de l'interface utilisateur, permettant aux utilisateurs de trouver rapidement ce dont ils ont besoin.

Navigation fluide : La navigation à travers les différentes sections de l'application sera fluide et cohérente, facilitant ainsi l'accès aux différentes fonctionnalités sans confusion.

Assistance visuelle : Des éléments visuels tels que des icônes intuitives, des couleurs distinctives et des indications visuelles seront utilisés pour guider les utilisateurs tout au long de leur expérience avec l'application.

En adoptant une approche centrée sur l'utilisateur dans la conception de l'interface utilisateur, StudentHub offrira une expérience utilisateur optimale pour tous les utilisateurs, quel que soit leur niveau de compétence technologique.

Implémentation du Programme de Gestion des Étudiants en Langage C

Dans cette section, nous détaillerons l'implémentation du programme de gestion des étudiants en langage C. Le programme est divisé en deux parties distinctes :

Première partie qui affiche directement les résultats sans l'utilisation de fichiers, et une seconde partie qui utilise la manipulation de fichiers pour la gestion des données des étudiants

A. Affichage Direct des Résultats

Voici comment cette partie est mise en œuvre :

Utilisation de Structures de Données :

Les informations des étudiants sont stockées dans des tableaux ou des structures de données en mémoire, au lieu d'être lues à partir de fichiers.

Fonctions de Manipulation de Données :

Des fonctions sont développées pour permettre à l'utilisateur d'effectuer les mêmes opérations que dans la première partie, mais cette fois-ci en interagissant directement avec les structures de données en mémoire.

Affichage des Résultats :

Les résultats sont affichés directement à l'écran à l'aide de fonctions d'affichage, offrant à l'utilisateur une vue instantanée des informations sur les étudiants.

B. Utilisation de Fichiers pour la Gestion des Données

Voici un aperçu de l'implémentation :

Structure de Données :

Une structure Etudiant est définie pour représenter les informations d'un étudiant, comprenant des champs tels que le nom, le prénom, l'identifiant, la filière, etc.

Fonctions pour la Gestion des Fichiers :

Des fonctions sont implémentées pour la lecture et l'écriture des données des étudiants dans des fichiers. Cela inclut la création de nouveaux fichiers, l'ajout d'informations sur les étudiants, la recherche et la modification de données existantes, ainsi que la suppression d'étudiants.

Interaction Utilisateur-Fichier :

Le programme permet à l'utilisateur d'effectuer diverses opérations sur les données des étudiants, telles que l'ajout, la modification et la suppression, en interagissant avec les fichiers.

C. Comparaison des Approches

Les deux approches offrent des avantages et des inconvénients distincts en termes de facilité d'utilisation, de performance et de gestion des données. L'utilisation de fichiers permet une persistance des données entre les exécutions du programme, mais peut-être moins efficace en termes de temps d'accès aux données. En revanche, l'affichage direct des résultats est plus rapide mais ne conserve pas les données entre les sessions.

D. Aperçu général des STRUCTURE, ENUMS, FONCTIONS

```
//// *****-ENUMS
typedef enum { HOMME, FEMME } Sexe;
typedef enum { NON, OUI } Bourse;
typedef enum { CNOPS, CNSS, AUCUNE } Assurance;
typedef enum { NON_RESILIE, RESILIE } Resiliation;
// *****-STRUCTS
typedef struct {
    char nom[50];
    char prenom[50];
    Sexe sexe;
    char date_naissance[20];
    char filiere[50];
    char apogee[20];
    char code_massar[20];
    char date_inscription[20];
    char date_resiliation[20];
    Bourse bourse;
    Assurance assurance;
    Resiliation resiliation;
    int modules;
    float *notes;
    float moyenne;
    char semestre[5];
    int semestres_valides;
    int semestres_non_valides;
    int semestres_bloques;
} Etudiant;
// *****-DECLARATION
void afficherEtudiant(Etudiant listeEtudiants[], int numEtudiants);
void afficher_etudiants_admis(Etudiant listeEtudiants[], int numEtudiants);
void afficher_etudiants_filiere(Etudiant listeEtudiants[], int numEtudiants);
void trier_par_apogee(Etudiant listeEtudiants[], int numEtudiants);
void trier_par_moyenne(Etudiant listeEtudiants[], int numEtudiants);
void trier_par_date_inscription(Etudiant listeEtudiants[], int numEtudiants);
void rechercher_par_apogee(Etudiant listeEtudiants[], int numEtudiants);
void rechercher_par_nom(Etudiant *listeEtudiants, int numEtudiants);
void rechercher_par_prenom(Etudiant *listeEtudiants, int numEtudiants);
void rechercher_etudiants_par_date_inscription(Etudiant *listeEtudiants, int numEtudiants);
```

Méthodologie de Traitement du StudentHub

1 Fonction d'Ajout d'Étudiant

Description : Cette fonction permet à l'utilisateur d'ajouter un nouvel étudiant au StudentHub.

Fonctionnalités :

- Demande à l'utilisateur de saisir les informations de l'étudiant (nom, prénom, ID, filière, etc.).
- Vérifie si l'étudiant existe déjà dans la base de données.
- Ajoute les informations de l'étudiant à la structure de données en mémoire.
- Met à jour le fichier de données des étudiants avec les nouvelles informations.

2. Fonction de Lecture des Données des Étudiants

Description : Cette fonction permet de lire les données des étudiants à partir d'un fichier et de les charger dans la mémoire du programme.

Fonctionnalités :

- Ouvre le fichier de données des étudiants.
- Lit les informations de chaque étudiant à partir du fichier.
- Stocke les données des étudiants dans une structure de données en mémoire.

3. Fonction de Modification des Données d'Étudiant

Description : Cette fonction permet à l'utilisateur de modifier les informations d'un étudiant existant dans le StudentHub.

Fonctionnalités :

- Demande à l'utilisateur de saisir l'ID de l'étudiant à modifier.
- Recherche l'étudiant dans la base de données.
- Permet à l'utilisateur de modifier les informations de l'étudiant.
- Met à jour la structure de données en mémoire et le fichier de données des étudiants avec les nouvelles informations.

4. Fonction de Recherche d'Étudiant

Description : Cette fonction permet à l'utilisateur de rechercher un étudiant dans la base de données du StudentHub.

Fonctionnalités :

- Demande à l'utilisateur de saisir un critère de recherche (nom, ID, filière, etc.).
- Parcourt la structure de données en mémoire pour trouver les étudiants correspondant au critère de recherche.
- Affiche les informations des étudiants trouvés à l'utilisateur.

5. Fonction de Suppression d'Étudiant

Description : Cette fonction permet à l'utilisateur de supprimer un étudiant de la base de données du StudentHub.

Fonctionnalités :

- Demande à l'utilisateur de saisir l'ID de l'étudiant à supprimer.
- Recherche l'étudiant dans la base de données.
- Supprime les informations de l'étudiant de la structure de données en mémoire.
- Met à jour le fichier de données des étudiants en supprimant les informations de l'étudiant.

6. Fonction de Verrouillage de l'Accès

Description : Cette fonction simule une page de connexion en demandant à l'utilisateur de saisir un nom d'utilisateur et un mot de passe pour accéder au programme.

Fonctionnalités :

- Affiche un message de bienvenue et les instructions pour la connexion.
- Demande à l'utilisateur de saisir son nom d'utilisateur et son mot de passe.
- Masque le mot de passe saisi à l'écran pour des raisons de sécurité.
- Vérifie si les informations de connexion sont correctes en les comparant aux identifiants prédéfinis (USERNAME et PASSWORD).
- Affiche un message de connexion réussie si les identifiants sont corrects, sinon affiche un message d'erreur avec le nombre de tentatives restantes.
- Limite le nombre de tentatives de connexion à un maximum défini (MAX_ATTEMPTS).
- Si la connexion est réussie, permet à l'utilisateur d'accéder au menu principal du StudentHub en appelant la fonction menuPrincipal().

Restructuration du Programme avec Séparation des Fichiers .h et .c

Introduction

Dans le cadre de l'amélioration de notre application "STUDENT HUB", nous avons entrepris une restructuration du code afin de le rendre plus modulaire et facile à maintenir. Cette restructuration a impliqué la séparation des différentes fonctionnalités en fichiers .h et .c distincts, suivant les bonnes pratiques de développement logiciel, combinée à l'utilisation de Makefile pour la gestion de la compilation et de l'édition des liens.

Objectif :

L'objectif principal de cette restructuration était de rendre le programme plus organisé et extensible, tout en simplifiant le processus de compilation et de construction du projet. En divisant le code en modules distincts et en utilisant un Makefile, nous visons à faciliter la gestion des fonctionnalités, à améliorer la lisibilité du code et à permettre une évolution plus aisée du logiciel, tout en automatisant les tâches de compilation.

Méthodologie :

La restructuration du programme a été réalisée en plusieurs étapes, combinant la séparation des fichiers .h et .c avec l'utilisation de Makefile :

Analyse de l'existant : Nous avons examiné en détail le code existant pour identifier les différentes fonctionnalités et déterminer les points où une modularisation serait bénéfique.

Définition des modules : Sur la base de notre analyse, nous avons identifié les différentes fonctionnalités du programme et défini les modules correspondants.

Création des fichiers .h et .c : Pour chaque module identifié, nous avons créé un fichier d'en-tête (.h) contenant les déclarations des fonctions associées, ainsi qu'un fichier source (.c) contenant les définitions de ces fonctions.

Développement du Makefile : Nous avons créé un Makefile pour automatiser le processus de compilation et d'édition des liens. Le Makefile définit les règles de compilation pour chaque fichier source (.c) et génère l'exécutable final en liant les différents modules.

Réorganisation de la fonction main : Nous avons réduit la complexité de la fonction main en déplaçant la logique métier dans les différents modules créés. La fonction main se charge désormais principalement de l'initialisation du programme et de l'appel des fonctions appropriées en fonction des choix de l'utilisateur.

Simplification de la fonction main : La fonction main a été condensée en une seule ligne pour libérer autant de mémoire que possible. Cette simplification a été réalisée en appelant une fonction de haut niveau qui orchestre le démarrage du programme.

Tests et validation : Nous avons effectué des tests approfondis pour nous assurer que la restructuration n'avait pas altéré le comportement fonctionnel du programme. Nous avons également vérifié que le programme restait aussi convivial et facile à utiliser qu'auparavant.

Résultats :

La restructuration du programme combinée à l'utilisation de Makefile a produit les résultats suivants :

- **Meilleure organisation du code :** Les fonctionnalités sont désormais regroupées de manière logique dans des modules distincts, ce qui rend le code plus facile à comprendre et à maintenir.
- **Modularité accrue :** Chaque module peut être développé, testé et déployé indépendamment, ce qui facilite l'ajout de nouvelles fonctionnalités ou la correction de bugs.
- **Automatisation du processus de compilation :** L'utilisation de Makefile permet de compiler automatiquement le projet en un seul clic, simplifiant ainsi le processus de développement.
- **Fonction main simplifiée :** La fonction main est maintenant plus concise et se concentre principalement sur la gestion du flux de contrôle et l'interaction avec l'utilisateur.
- **Libération maximale de mémoire :** La simplification de la fonction main en une seule ligne permet de libérer autant de mémoire que possible, ce qui est essentiel pour optimiser les performances, en particulier sur des systèmes embarqués ou à ressources limitées.

Utilisation de l'application STUDENTHUB

Pour faciliter une adoption fluide de l'application StudentHub, il est essentiel de comprendre les différentes fonctionnalités offertes par l'application ainsi que les étapes nécessaires à leur utilisation. Cette section fournit un guide étape par étape sur la façon d'utiliser efficacement l'application pour gérer les données des étudiants.

1. Connexion à l'Application

Avant de pouvoir accéder aux fonctionnalités de l'application, l'utilisateur doit se connecter en fournissant un nom d'utilisateur et un mot de passe valides. La fonction de connexion assure la sécurité et permet de garantir que seules les personnes autorisées peuvent accéder aux données des étudiants.

2. Menu Principal

Une fois connecté avec succès, l'utilisateur est redirigé vers le menu principal de l'application. Ce menu affiche toutes les options disponibles, telles que l'ajout d'un nouvel étudiant, la consultation des étudiants existants, la modification des informations des étudiants, etc.

3. Ajout d'un Nouvel Étudiant

Pour ajouter un nouvel étudiant, l'utilisateur sélectionne l'option correspondante dans le menu principal. Ensuite, l'application guide l'utilisateur à travers le processus de saisie des informations de l'étudiant, telles que le nom, le prénom, l'ID, la filière, etc. Une fois toutes les informations saisies, l'étudiant est ajouté à la base de données de l'application.

4. Consultation des Étudiants

L'option de consultation permet à l'utilisateur d'afficher différentes listes d'étudiants, notamment la liste de tous les étudiants, la liste des étudiants admis seulement et la liste des étudiants d'une filière spécifiée. L'utilisateur peut sélectionner l'option appropriée dans le menu et visualiser les informations des étudiants selon ses besoins.

5. Modification des Informations des Étudiants

Si des modifications doivent être apportées aux informations d'un étudiant existant, l'utilisateur peut sélectionner l'option de modification dans le menu principal. Ensuite, l'application guide l'utilisateur à travers le processus de recherche de l'étudiant par son numéro d'apogée et de mise à jour des informations nécessaires.

6. Suppression d'un Étudiant

Pour supprimer un étudiant de la base de données, l'utilisateur sélectionne l'option de suppression dans le menu principal. Ensuite, l'application guide l'utilisateur à travers le processus de recherche de l'étudiant par son numéro d'apogée et de confirmation de la suppression.

7. Déconnexion

Une fois que l'utilisateur a terminé d'utiliser l'application, il peut se déconnecter en sélectionnant l'option de déconnexion dans le menu principal. Cela ferme la session en cours et ramène l'utilisateur à l'écran de connexion.

8. Aperçu général de l'App :

```
Cette application permet d'effectuer les taches suivantes :
1. Ajouter un ou plusieurs etudiants
2. Modifier les informations sur l'etudiant
3. Consulter :
  a. La liste de tous les etudiants
  b. La liste des etudiants admis seulement
  c. La liste des etudiants d'une filiere specifiee
4. Trier les etudiants par :
  a. Numero Apogee
  b. Moyenne
  c. Date d'inscription
5. Rechercher :
  a. Un etudiant par le Numero apogee
  b. Tous les etudiants ayant meme :
    - Nom
    - Prenom
    - Date d'inscription
6. Supprimer un etudiant
0. Quitter

Choisissez une option (0-6) : |
```

Défis Rencontrés lors de la Restructuration du Programme

Lors de la restructuration du programme "STUDENT HUB", plusieurs défis ont été rencontrés, notamment dans le choix des noms, la gestion du code et la gestion des erreurs liées à l'allocation mémoire. Dans cette partie, nous allons explorer en détail ces défis et décrire comment notre binôme a traité chaque étape du processus de restructuration.

Choix des Noms

L'un des premiers défis auxquels nous avons été confrontés était le choix des noms pour les différents modules, fonctions et variables. Un bon choix de noms est crucial pour la lisibilité et la compréhension du code, mais cela peut parfois être subjectif et complexe. Nous avons dû prendre en compte plusieurs facteurs, tels que la clarté, la concision et la cohérence avec les conventions de nommage existantes dans le projet. Par exemple, nous avons passé un certain temps à discuter et à débattre sur le nom le plus approprié pour un module de gestion des étudiants, en tenant compte à la fois de sa fonctionnalité et de sa relation avec les autres parties du programme.

Gestion du Code

Un autre défi majeur était de gérer efficacement le code existant tout en introduisant de nouvelles fonctionnalités et en modifiant la structure du programme. Nous devons nous assurer de ne pas introduire de régressions ou de nouveaux bugs tout en effectuant des modifications importantes. Cela nécessitait une planification minutieuse, une communication constante entre les membres de l'équipe et l'utilisation d'outils de contrôle de version pour suivre les changements et les révisions du code. Par exemple, nous avons rencontré des conflits lors de la fusion de branches contenant des modifications concurrentes, ce qui a nécessité une collaboration étroite pour résoudre ces conflits de manière appropriée.

Erreurs d'Allocation Mémoire

Un défi particulièrement délicat était de gérer correctement l'allocation et la libération de mémoire pour éviter les fuites de mémoire, les accès invalides et les plantages du programme. Les erreurs liées à l'allocation mémoire peuvent être difficiles à diagnostiquer et à résoudre, surtout dans un programme de grande taille avec de multiples points d'allocation. Nous avons dû examiner attentivement chaque partie du code pour identifier les endroits où la mémoire était allouée et s'assurer qu'elle était correctement libérée une fois qu'elle n'était plus nécessaire. De plus, nous avons mis en place des stratégies de gestion des erreurs pour détecter et signaler les échecs d'allocation mémoire afin de les traiter de manière appropriée, par exemple en affichant un message d'erreur à l'utilisateur et en quittant proprement le programme.

Intégration du Makefile

L'intégration du Makefile constituait également un défi important dans le processus de restructuration du programme. Le Makefile est un outil essentiel pour automatiser le processus de compilation et de construction du programme, ce qui facilite le développement, la gestion et le déploiement du logiciel. Cependant, la configuration d'un Makefile peut être complexe et nécessiter une compréhension approfondie des dépendances du projet et des règles de compilation. Nous avons dû consacrer du temps à la mise en place d'un Makefile robuste et flexible, capable de gérer efficacement les dépendances entre les différents fichiers source, ainsi que les options de compilation et de liaison nécessaires pour produire un exécutable fonctionnel.

De plus, un autre défi est survenu lors de l'utilisation de la fonction gotoxy, qui permet de déplacer le curseur de texte à une position spécifique sur l'écran. Cette fonction est couramment utilisée dans les programmes basés sur l'interface en ligne de commande sous Windows. Cependant, elle n'est pas prise en charge de manière native sur Linux, ce qui a posé un problème lors du portage du programme vers cette plateforme. Nous avons dû trouver des alternatives ou des solutions de contournement pour remplacer l'utilisation de gotoxy afin de garantir le bon fonctionnement du programme sur Linux sans compromettre ses fonctionnalités.

Approche Adoptée par Notre Binôme

Face à ces défis, notre binôme a adopté une approche collaborative et méthodique pour traiter chaque étape de la restructuration du programme. Nous avons commencé par une analyse approfondie des besoins du projet et des contraintes techniques, ce qui nous a permis d'élaborer un plan détaillé pour la mise en œuvre des changements. Ensuite, nous avons réparti les tâches de manière équilibrée entre les membres de l'équipe en fonction de leurs compétences et de leur expérience, tout en restant flexibles et prêts à ajuster notre approche en fonction des défis rencontrés en cours de route.

Nous avons également accordé une attention particulière à la communication et à la coordination entre les membres de l'équipe, en organisant régulièrement des réunions de suivi pour discuter des progrès réalisés, des difficultés rencontrées et des décisions à prendre. Cette communication ouverte et transparente nous a permis de résoudre rapidement les problèmes et de prendre des décisions éclairées tout au long du processus de restructuration.

En outre, nous avons adopté une approche itérative et incrémentale, en effectuant des tests réguliers à chaque étape du processus pour valider les changements et identifier les problèmes potentiels le plus tôt possible. Cela nous a permis de corriger rapidement les erreurs et de minimiser les risques de régression.

Partitionnement et Phases de Développement du Projet

Partitionnement en Deux Étapes

Le projet a été scindé en deux étapes distinctes pour assurer une progression méthodique et un développement efficace. La première étape se concentre sur la mise en place de l'architecture de base, sans manipulation de fichiers. Cette phase est cruciale pour établir les fondations solides du projet et garantir une compréhension claire des exigences et des spécifications. Elle comprend notamment la conception de l'architecture logicielle, la définition des interfaces et la mise en place des structures de données nécessaires. Cette approche permet de réduire les risques liés à la complexité et de se concentrer sur la conception avant d'entrer dans la phase de mise en œuvre.

La deuxième étape du projet consiste à implémenter les fonctionnalités principales en tenant compte des fichiers et des entrées/sorties. Cette phase intervient une fois que l'architecture de base a été établie et que les concepts fondamentaux ont été testés avec succès. Elle implique le développement de modules spécifiques, la gestion des données externes et la création d'interfaces utilisateur le cas échéant. Cette approche itérative permet d'ajouter progressivement des fonctionnalités tout en maintenant la stabilité du système.

Phases de Développement du Code

Analyse des Exigences : Cette phase initiale consiste à comprendre les besoins du projet, à recueillir les spécifications fonctionnelles et à identifier les contraintes techniques.

Conception de l'Architecture : Une fois les exigences clairement définies, l'architecture logicielle est conçue en tenant compte des principes de modularité, de réutilisabilité et de maintenabilité.

Implémentation de la Base : Cette étape se concentre sur la mise en place des structures de base du projet, y compris la création de classes, de fonctions et de bibliothèques nécessaires.

Développement Itératif : Les fonctionnalités sont implémentées de manière itérative, en commençant par les fonctionnalités les plus essentielles et en progressant vers les fonctionnalités plus avancées.

Tests et Validation : À chaque étape du développement, des tests unitaires et des tests d'intégration sont effectués pour vérifier le bon fonctionnement du système et garantir sa conformité aux spécifications.

Optimisation et Maintenance : Une fois que le projet est fonctionnel, des efforts sont déployés pour optimiser les performances, améliorer l'efficacité du code et assurer sa maintenance à long terme.

En suivant ces phases de développement, le projet est réalisé de manière structurée et méthodique, ce qui garantit la qualité du logiciel final et facilite sa gestion tout au long de son cycle de vie.

PART1 :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#ifdef _WIN32
#define CLEAR_SCREEN "cls"
#else
#define CLEAR_SCREEN "clear"
#endif
#define MAX_ATTEMPTS 5
#define USERNAME "admin"
#define PASSWORD "smis4"
#include <windows.h>
//*****-ENUMS
typedef enum { HOMME, FEMME } Sexe;
typedef enum { NON, OUI } Bourse;
typedef enum { CNOPS, CNSS, AUCUNE } Assurance;
typedef enum { NON_RESILIE, RESILIE } Resiliation;
// *****-STRUCTS
typedef struct {
    char nom[50];
    char prenom[50];
    Sexe sexe;
    char date_naissance[20];
    char filiere[50];
    char apogee[20];
    char code_massar[20];
    char date_inscription[20];
    char date_resiliation[20];
    Bourse bourse;
    Assurance assurance;
    Resiliation resiliation;
    int modules;
    float *notes;
    float moyenne;
    char semestre[5];
    int semestres_valides;
    int semestres_non_valides;
    int semestres_bloques;
} Etudiant;
// *****-DECLARATION
void afficherEtudiant(Etudiant listeEtudiants[], int numEtudiants);
void afficher_etudiants_admis(Etudiant listeEtudiants[], int numEtudiants);
void afficher_etudiants_filiere(Etudiant listeEtudiants[], int numEtudiants);
void trier_par_apogee(Etudiant listeEtudiants[], int numEtudiants);
void trier_par_moyenne(Etudiant listeEtudiants[], int numEtudiants);
void rechercher_par_apogee(Etudiant listeEtudiants[], int numEtudiants);
void rechercher_par_nom(Etudiant *listeEtudiants, int numEtudiants);
void rechercher_par_prenom(Etudiant *listeEtudiants, int numEtudiants);
void rechercher_etudiants_par_date_inscription(Etudiant *listeEtudiants, int numEtudiants);
void clearScreen();
void gotoxy(int,int);
void lock();
```

```

}
// *****-MODIFIER
void modifierEtudiant(Etudiant *listeEtudiants, int numEtudiants) {
    char apogee[20];
    printf("Entrez le code d'apogee de l'etudiant a modifier : ");
    scanf("%s", apogee);
    int i;
    for (i = 0; i < numEtudiants; i++) {
        if (strcmp(listeEtudiants[i].apogee, apogee) == 0) {
            printf("Veuillez saisir les nouvelles informations de l'etudiant :\n");
            saisirInfosEtudiant(&listeEtudiants[i]);
            printf("Les informations de l'etudiant ont ete mises a jour avec succes.\n");
            return;
        }
    }
    printf("Aucun etudiant avec le code d'apogee %s n'a ete trouve.\n", apogee);
}
// *****-CONSULTER
void consulter(Etudiant *listeEtudiants, int numEtudiants) {
    printf("\nVeuillez choisir une option de consultation :\n");
    printf("a. La liste de tous les etudiants\n");
    printf("b. La liste des etudiants admis seulement\n");
    printf("c. La liste des etudiants d'une filiere specifiee\n");
    printf("Votre choix : ");
    char optionConsultation;
    scanf(" %c", &optionConsultation);

    switch (optionConsultation) {
        case 'a':
            afficherEtudiant(listeEtudiants, numEtudiants);
            break;
        case 'b':
            afficher_etudiants_admis(listeEtudiants, numEtudiants);
            break;
        case 'c':
            afficher_etudiants_filiere(listeEtudiants, numEtudiants);
            break;
        default:
            printf("\nOption de consultation invalide.\n");
    }
}

```

```

// *****-AFFICHE INFOS-*****
void afficherEtudiant(Etudiant listeEtudiants[], int numEtudiants) {
    printf("\n");
    printf("| Apogee|      Nom      |      Prenom      | Sexe |Date de Naissance |      Filiere      |
Moyenne | Decision |\n");
    int i,j;
    for (i = 0; i < numEtudiants; i++) {
        char decision[10];
        if (listeEtudiants[i].moyenne >= 10.0) {
            int toutesSuperieuresA5 = 1;
            for (j = 0; j < listeEtudiants[i].modules; j++) {
                if (listeEtudiants[i].notes[j] < 5.0) {
                    toutesSuperieuresA5 = 0;
                    break;
                }
            }
            if (toutesSuperieuresA5) {
                strcpy(decision, "\033[1;32m admis  \033[0m ");
            } else {
                strcpy(decision, "\033[1;31m Non Admis \033[0m");
            }
        } else {
            strcpy(decision, "\033[1;31mNon Admis\033[0m");
        }
        printf("|%8s|%15s|%15s|%8s|%18s|%15s|%9.2f|%10s|\n",
            listeEtudiants[i].apogee,
            listeEtudiants[i].nom,
            listeEtudiants[i].prenom,
            (listeEtudiants[i].sexe == HOMME) ? "Homme" : "Femme",
            listeEtudiants[i].date_naissance,
            listeEtudiants[i].filiere,
            listeEtudiants[i].moyenne,
            decision);
    }
}

```

```

// *****-AFFICHE FILIERE-*****
void afficher_etudiants_filiere(Etudiant *listeEtudiants, int numEtudiants) {
    char filiere[50];
    printf("Entrez le nom de la filiere : ");
    scanf("%s", filiere);
    printf("Liste des etudiants de la filiere %s :\n", filiere);
    int i;
    for (i = 0; i < numEtudiants; i++) {
        if (strcmp(listeEtudiants[i].filiere, filiere) == 0) {
            afficherEtudiant(&listeEtudiants[i], 1);
        }
    }
}

// *****-TRIER
void trier(Etudiant *listeEtudiants, int numEtudiants) {
    printf("\nVeuillez choisir une option de tri :\n");
    printf("a. Trier par numero Apogee\n");
    printf("b. Trier par moyenne\n");
    printf("c. Trier par date d inscription\n");
    printf("Votre choix : ");
    char optionTri;
    scanf(" %c", &optionTri);

    switch (optionTri) {
        case 'a':
            trier_par_apogee(listeEtudiants, numEtudiants);
            break;
        case 'b':
            trier_par_moyenne(listeEtudiants, numEtudiants);
            break;
        case 'c':
            trier_par_date_inscription(listeEtudiants, numEtudiants);
            break;
        default:
            printf("\nOption de tri invalide.\n");
    }
}

// *****-SOUS FONCTION TRIER
// *****-TRIE APPOG-*****
void trier_par_apogee(Etudiant *listeEtudiants, int numEtudiants) {
    int i, j;
    for (i = 0; i < numEtudiants - 1; i++) {
        for (j = i + 1; j < numEtudiants; j++) {
            if (strcmp(listeEtudiants[j].apogee, listeEtudiants[i].apogee) < 0) {
                Etudiant temp = listeEtudiants[i];
                listeEtudiants[i] = listeEtudiants[j];
                listeEtudiants[j] = temp;
            }
        }
    }
    printf("Les etudiants ont ete tries par numero Apogee avec succes.\n");
}

```

PART2 :

```
// *****-SOUS FONCTION TRIER
// *****-TRIE APOGEE-*****
void trier_par_apogee(Etudiant *listeEtudiants, int numEtudiants) {
    int i,j;
    for ( i = 0; i < numEtudiants - 1; i++) {
        for ( j = i + 1; j < numEtudiants; j++) {
            if (strcmp(listeEtudiants[j].apogee, listeEtudiants[i].apogee) < 0) {
                Etudiant temp = listeEtudiants[i];
                listeEtudiants[i] = listeEtudiants[j];
                listeEtudiants[j] = temp;
            }
        }
    }
    printf("Les etudiants ont ete tries par numero Apogee avec succes.\n");
    FILE *p=fopen("DATA_TRIE_APOGEE.txt","w");
    if (p == NULL) {
        printf("Erreur lors de l'ouverture du fichier.\n");
        exit(0);
    }
    for(i=0;i<numEtudiants;i++){
        fprintf(p,"%8s|%15s|%15s|%8s|%18s|%15s|%9.2f|\n",
            listeEtudiants[i].apogee,
            listeEtudiants[i].nom,
            listeEtudiants[i].prenom,
            (listeEtudiants[i].sexe == HOMME) ? "Homme" : "Femme",
            listeEtudiants[i].date_naissance,
            listeEtudiants[i].filiere,
            listeEtudiants[i].moyenne);
    }
    fclose(p);
}
```

```
printf("-----|STUDENT HUB|-----\n");
printf("\nCombien d'etudiants voulez-vous ajouter ? ");
int nombreEtudiants;
scanf("%d", &nombreEtudiants);
*listeEtudiants = realloc(*listeEtudiants, (*numEtudiants + nombreEtudiants) * sizeof(Etudiant));
if (*listeEtudiants == NULL) {
    fprintf(stderr, "Erreur d'allocation memoire pour les nouveaux etudiants.\n");
    exit(EXIT_FAILURE);
}
for (i = 0; i < nombreEtudiants; i++) {
    printf("                                \nETUDIANT: %d                                \n", *numEtudiants + i + 1);
    saisirInfosEtudiant(&(*listeEtudiants)[*numEtudiants + i]);
    clearScreen();
}
*numEtudiants += nombreEtudiants;

// Writing added students to file
p = fopen("DATA.txt", "w");
if (p == NULL) {
    printf("Erreur lors de l'ouverture du fichier.\n");
    exit(EXIT_FAILURE);
}
// Write header line
fprintf(p, "| Appogee |      Nom      |      Prenom      | Sexe | Date de Naissance |      Filiere\n");
printf("Moyenne | Decision |\n");

// Write student information
for (i = *numEtudiants - nombreEtudiants; i < *numEtudiants; i++) {
    fprintf(p, "|%20s|%20s|%20s|%8s|%20s|%20s|%9.2f|\n",
        (*listeEtudiants)[i].apogee,
        (*listeEtudiants)[i].nom,
        (*listeEtudiants)[i].prenom,
        ((*listeEtudiants)[i].sexe == HOMME) ? "Homme" : "Femme",
        (*listeEtudiants)[i].date_naissance,
        (*listeEtudiants)[i].filiere,
        (*listeEtudiants)[i].moyenne);
}
fclose(p);
break;
```

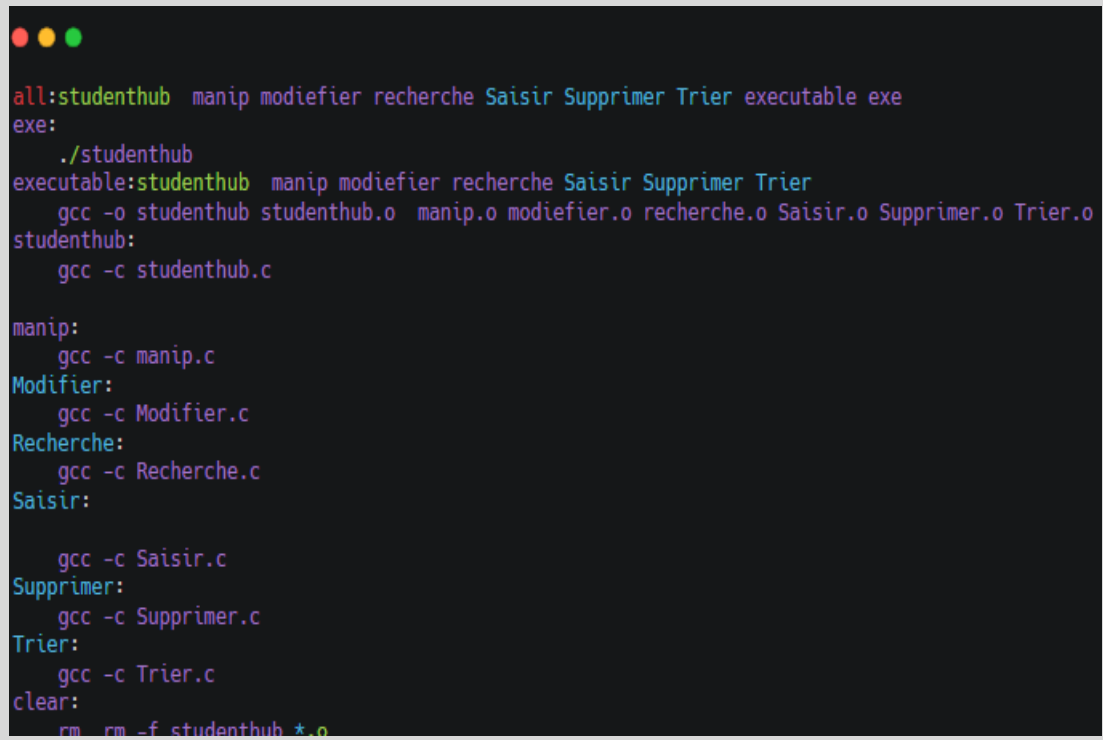
```

// *****-SOUS FONCTION CONSULTER
// *****-AFFICHE INFOS-*****
void afficherEtudiant(Etudiant listeEtudiants[], int numEtudiants) {
    printf("\n");
    printf("| Appogee |      Nom      |      Prenom      | Sexe | Date de Naissance |      Filiere      |
Moyenne | Decision |\n");
    int i,j;
    for ( i = 0; i < numEtudiants; i++) {
        char decision[10];
        if (listeEtudiants[i].moyenne >= 10.0) {
            int toutesSuperieuresA5 = 1;
            for ( j = 0; j < listeEtudiants[i].modules; j++) {
                if (listeEtudiants[i].notes[j] < 5.0) {
                    toutesSuperieuresA5 = 0;
                    break;
                }
            }
            if (toutesSuperieuresA5) {
                strcpy(decision, "Admis");
            } else {
                strcpy(decision, "Non admis");
            }
        } else {
            strcpy(decision, "Non admis");
        }
        printf("|%8s|%15s|%15s|%8s|%18s|%15s|%9.2f|%9s|\n",
            listeEtudiants[i].apogee,
            listeEtudiants[i].nom,
            listeEtudiants[i].prenom,
            (listeEtudiants[i].sexe == HOMME) ? "Homme" : "Femme",
            listeEtudiants[i].date_naissance,
            listeEtudiants[i].filiere,
            listeEtudiants[i].moyenne, decision);
    }
}

// *****-AFFICHE ADMIS-*****
void afficher_etudiants_admis(Etudiant *listeEtudiants, int numEtudiants) {
    printf("Liste des etudiants admis :\n");
    printf("| Appogee |      Nom      |      Prenom      | Sexe | Date de Naissance |      Filiere      |
Moyenne | Decision |\n");
    int i,j;
    for ( i = 0; i < numEtudiants; i++) {
        char decision[10];
        if (listeEtudiants[i].moyenne >= 10.0) {
            int toutesSuperieuresA5 = 1;
            for ( j = 0; j < listeEtudiants[i].modules; j++) {
                if (listeEtudiants[i].notes[j] < 5.0) {
                    toutesSuperieuresA5 = 0;
                    break;
                }
            }
        }
    }
}

```

Fichier makefile :



```
all:studenthub manip modifier recherche Saisir Supprimer Trier executable exe
exe:
    ./studenthub
executable:studenthub manip modifier recherche Saisir Supprimer Trier
    gcc -o studenthub studenthub.o manip.o modifier.o recherche.o Saisir.o Supprimer.o Trier.o
studenthub:
    gcc -c studenthub.c

manip:
    gcc -c manip.c
Modifier:
    gcc -c Modifier.c
Recherche:
    gcc -c Recherche.c
Saisir:
    gcc -c Saisir.c
Supprimer:
    gcc -c Supprimer.c
Trier:
    gcc -c Trier.c
clear:
    rm -f studenthub *.o
```


Problématique MAKEFILE

1. Problème du Makefile:

Le premier problème auquel nous avons été confrontés était lié au Makefile. Malgré la vérification de tous les arguments et la mise en place correcte des règles de compilation et d'exécution, l'application ne fonctionnait pas comme prévu. Cette situation a nécessité une analyse approfondie du Makefile pour identifier et résoudre les problèmes.

Détails du Problème:

- Tous les arguments du Makefile étaient corrects, y compris les règles de compilation et d'exécution.
- Cependant, lors de l'exécution de la commande "make", l'application ne se lançait pas ou présentait des erreurs lors de l'exécution.
- Les directives de compilation étaient correctes, mais il y avait des problèmes avec l'organisation des fichiers sources et des dépendances.

Actions :

1. Analyse du Makefile: Nous avons examiné attentivement le Makefile pour identifier toute erreur de syntaxe ou de logique. Toutes les règles de compilation, les chemins des fichiers sources et des en-têtes, ainsi que les options de l'éditeur de liens ont été vérifiés.
2. Vérification des Dépendances: Nous avons revu les dépendances entre les fichiers sources et les en-têtes pour nous assurer qu'ils étaient correctement spécifiés. Des erreurs dans les dépendances pourraient conduire à une recompilation incorrecte des fichiers.
3. Test avec des Options de Débogage: Nous avons activé les options de débogage pour obtenir des informations détaillées sur les erreurs à l'exécution. Cela nous a permis de localiser plus précisément les problèmes.
4. Comparaison avec des Makefiles Fonctionnels: Nous avons comparé notre Makefile avec des exemples de Makefiles fonctionnels pour détecter toute différence significative dans la structure ou la syntaxe.

Résolution du Problème:

Malheureusement, malgré nos efforts pour résoudre le problème avec le Makefile, nous n'avons pas encore réussi à le résoudre. Les ajustements apportés aux chemins des fichiers et aux règles de compilation n'ont pas abouti à une compilation réussie de l'application.

Prochaines Étapes:

- Nous continuerons à examiner de près le Makefile pour identifier toute autre source potentielle d'erreur.
- Des discussions supplémentaires avec l'équipe de développement et des consultations de ressources externes pourraient être nécessaires pour trouver une solution.
- Si nécessaire, nous pourrions envisager de reconstruire le Makefile à partir de zéro pour garantir sa fonctionnalité.

Aperçue du problème :

```
PS C:\Users\pc\Desktop\projet 2.0\projet 2.0\part1\code separe\functions> make all
gcc -c studenthub.c
gcc -c manip.c
manip.c: In function 'menuPrincipal':
manip.c:81:15: warning: implicit declaration of function 'trier' [-Wimplicit-function-declaration]
      trier(*listeEtudiants, *numEtudiants);
      ^~~~~
make: *** No rule to make target 'modiefier', needed by 'all'. Stop.
PS C:\Users\pc\Desktop\projet 2.0\projet 2.0\part1\code separe\functions> |
```

```
PS C:\Users\pc\Desktop\projet 2.0\projet 2.0\part1\code separe\functions> make
gcc -c studenthub.c
gcc -c manip.c
manip.c: In function 'menuPrincipal':
manip.c:77:17: warning: implicit declaration of function 'consulter' [-Wimplicit-function-declaration]
      consulter(*listeEtudiants, *numEtudiants);
      ^~~~~~
manip.c:80:15: warning: implicit declaration of function 'trier' [-Wimplicit-function-declaration]
      trier(*listeEtudiants, *numEtudiants);
      ^~~~~
manip.c: In function 'lock':
manip.c:142:33: warning: implicit declaration of function 'getch' [-Wimplicit-function-declaration]
      while ((password_char = getch()) != '\r') { // '\r' correspond 'à la touche "Entrée" sur W
indows
make: *** No rule to make target 'modiefier', needed by 'all'. Stop.
```

Conclusion

Le développement de l'application Student Hub a été une expérience enrichissante et instructive, nous permettant de mettre en pratique divers concepts de programmation en langage C. Ce projet visait à créer un système de gestion des étudiants pour une institution éducative, offrant des fonctionnalités telles que l'ajout, la modification, la consultation, le tri, la recherche et la suppression des étudiants.

Réalisation des objectifs

Nous sommes parvenus à atteindre les objectifs principaux du projet, à savoir :

- Conception d'une structure de données adaptée : Nous avons utilisé des structures de données telles que les énumérations et les structures pour organiser les informations des étudiants de manière efficace et logique.
- Implémentation des fonctionnalités de base : L'application permet d'ajouter, de modifier, de consulter, de trier, de rechercher et de supprimer des étudiants, offrant ainsi un ensemble complet de fonctionnalités pour la gestion des données.
- Interface utilisateur conviviale : Nous avons développé une interface utilisateur simple mais fonctionnelle, permettant à l'utilisateur d'interagir facilement avec l'application via le terminal.

Défis rencontrés

Au cours du développement de l'application, nous avons rencontré plusieurs défis :

- Gestion de la mémoire : La gestion dynamique de la mémoire pour stocker les notes des étudiants et la réallocation des tableaux ont nécessité une attention particulière pour éviter les fuites de mémoire et les erreurs de segmentation.
- Gestion des erreurs utilisateur : Nous avons dû mettre en place des mécanismes de gestion des erreurs pour garantir une expérience utilisateur fluide, notamment en cas de saisie incorrecte ou d'opérations non valides.
- Optimisation de l'affichage : L'affichage des données des étudiants de manière claire et lisible tout en tenant compte de la taille de la console a été un défi, nécessitant une mise en forme soignée et efficace.

Perspectives d'amélioration

Malgré la réalisation des objectifs principaux, il reste des pistes d'amélioration pour l'application :

- **Interface utilisateur améliorée :** Une interface utilisateur plus conviviale et interactive, éventuellement avec une interface graphique, pourrait améliorer l'expérience utilisateur.
- **Gestion avancée des données :** Intégrer des fonctionnalités avancées telles que l'exportation/importation de données, la sauvegarde automatique et la gestion des fichiers pourrait rendre l'application plus robuste et polyvalente.
- **Optimisation des performances :** Des optimisations supplémentaires pourraient être apportées pour améliorer les performances de l'application, notamment en termes de temps d'exécution et d'utilisation de la mémoire.

Conclusion générale

En conclusion, le projet Student Hub a été une occasion précieuse de mettre en pratique nos connaissances en programmation en C et de développer une application fonctionnelle répondant aux besoins de gestion des étudiants. Bien que des défis aient été rencontrés en cours de route, le processus de développement a été enrichissant et nous a permis d'acquérir de nouvelles compétences. Avec des améliorations futures, l'application pourrait devenir un outil puissant pour les institutions éducatives dans la gestion efficace de leurs données étudiantes.

Remerciements

Je tiens à exprimer ma sincère gratitude à la Faculté des Sciences Semlalia pour son soutien continu tout au long de ce projet. Les ressources et l'environnement d'apprentissage fournis par la faculté ont joué un rôle essentiel dans la réussite de ce travail.

Je tiens également à remercier chaleureusement Mme Rachida Hannane pour son encadrement attentif et ses conseils avisés tout au long du développement de ce projet. Son expertise et son dévouement ont été d'une valeur inestimable, et je suis reconnaissant(e) pour le temps et les efforts qu'elle a consacrés à ce projet.

Je voudrais également présenter mes excuses pour les problèmes rencontrés avec l'outil Make lors du développement de l'application. Bien que nous ayons été confrontés à des difficultés techniques, nous avons pu surmonter ces obstacles grâce à la collaboration et à la persévérance de toute l'équipe.

Enfin, je souhaite exprimer ma reconnaissance à tous ceux qui ont contribué de près ou de loin à la réalisation de ce projet. Leur soutien et leurs encouragements ont été précieux et ont grandement contribué à notre succès.

Cette expérience a été tout simplement incroyable ! Elle m'a permis de développer non seulement mes compétences techniques, mais aussi mes compétences en travail d'équipe, en gestion de projet et en résolution de problèmes. C'est une aventure que je n'oublierai jamais, et je suis reconnaissant(e) d'avoir eu la chance de la vivre. Merci encore pour cette opportunité inestimable de mettre en pratique nos connaissances académiques dans un projet concret.



FIN



STUDENTHUB

MERCI POUR VOTRE ATTENTION