**Project Report: Phonebook Management System**

**Abstract:**

The Phonebook Management System project aims to develop a comprehensive solution for organizing and managing contacts efficiently. This system allows users to perform tasks such as adding new contacts, viewing contact details, searching for specific contacts, and deleting entries. An administrative mode adds additional functionality and access control.

**1. Introduction**

**1.1 Background**

The need for an organized and accessible repository of contact information is essential in today's fast-paced world. The Phonebook Management System addresses this need by providing a user-friendly platform for managing contacts effectively.

**1.2 Objectives**

The primary objectives of this project are as follows:

- Create an intuitive and easy-to-use interface for contact management.

- Implement robust contact data storage and retrieval mechanisms.

- Enable users to perform common phonebook tasks efficiently.

- Implement an administrator mode for enhanced control over the phonebook.

**2. Technologies Used**

The project utilizes the following technologies and tools:

- C programming language

- Standard Input/Output (stdio) library

- `<stdio.h>`, `<string.h>`, and `<stdbool.h>` standard headers

**3. System Architecture**

**3.1 Front-End**

The front-end of the system focuses on user interactions, presenting a command-line interface for managing contacts.

**3.2 Back-End**

The back-end handles core logic, including contact manipulation, password validation, and file I/O operations.

**3.3 Database**

Contact data is stored persistently in a text file ("phonebook.txt") using a comma-separated value (CSV) format.

**4. Project Modules**

**4.1 Module 1: Contact Management**

This module encompasses core contact management features, such as adding, viewing, searching, and deleting contacts.

**4.2 Module 2: Admin Functionality**

Module 2 provides an administrator mode with additional functionalities and access privileges.

**5. Design and Implementation**

**5.1 Front-End Design**

The front-end design emphasizes user-friendliness, with a command-line interface optimized for efficient contact management.

**5.2 Back-End Design**

The back-end design modularizes functions for contact manipulation, ensures secure password validation, and manages file I/O operations.

**5.3 Database Design**

Contact data is stored in "phonebook.txt" with each line representing a contact's name, phone number, and address in a structured format.

**6. Features and Functionality**

**6.1 Feature 1**: Add Contacts

Users can input contact information, including name, phone number, and address, which is then stored in the phonebook.

**6.2 Feature 2**: View Contacts

The system allows users to view the complete list of contacts in the phonebook.

**6.3 Feature 3**: Search Contacts

Users can search for specific contacts by providing a phone number.

**6.4 Feature 4**: Delete Contacts

Contacts can be deleted, and the phonebook is updated accordingly.

**6.5 Feature 5**: Admin Mode

An administrator mode provides additional functionalities for authorized users, enhancing control over the phonebook.

**7. Testing**

Testing is essential to ensure system reliability and accuracy.

**7.1 Unit Testing**

Individual functions and modules underwent testing to verify correctness.

**7.2 Integration Testing**

Integration testing was conducted to ensure seamless interaction between system components.

**7.3 User Acceptance Testing**

User testing ensured the system's usability and gathered valuable user feedback.

**8. Challenges Faced**

The development process presented several challenges, including UI design, file I/O, and input validation.

**9. Future Enhancements**

Future enhancements may include user authentication, data encryption, and support for a larger number of contacts.

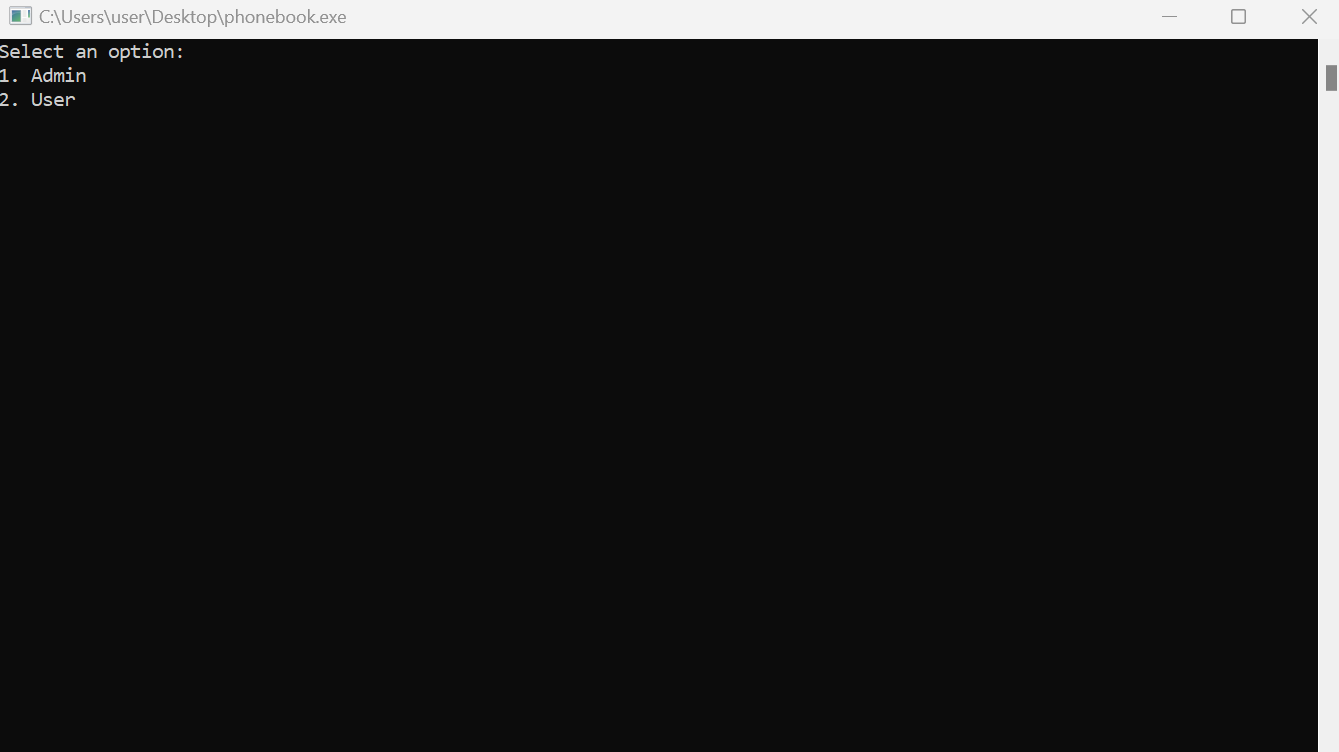
**10. Conclusion**

The Phonebook Management System offers an efficient solution for contact organization. It provides a user-friendly interface, essential contact management features, and an admin mode for enhanced control.

11. **References**

Stackoverflow,chatgpt,gfg

**12.1 Screenshots**

****

**12.2 Code Snippets:**

**#include <stdio.h>**

**#include <string.h>**

**#include <stdbool.h>**

**struct Contact {**

**char name[50];**

**char phone[15];**

**char address[100];**

**};**

**bool validateAdminPassword() {**

**char password[20];**

**printf("Enter admin password: ");**

**scanf("%s", password);**

**if (strcmp(password, "1234") == 0) {**

**return true;**

**} else {**

**printf("Incorrect admin password. Please try again.\n");**

**return false;**

**}**

**}**

**void addContactToFile(FILE \*file, const struct Contact \*contact) {**

**fprintf(file, "%s,%s,%s\n", contact->name, contact->phone, contact->address);**

**}**

**void loadContactsFromFile(FILE \*file, struct Contact phonebook[], int \*numContacts) {**

**struct Contact newContact;**

**while (fscanf(file, "%[^,],%[^,],%[^\n]\n", newContact.name, newContact.phone, newContact.address) != EOF) {**

**phonebook[\*numContacts] = newContact;**

**(\*numContacts)++;**

**}**

**}**

**void saveContactsToFile(FILE \*file, const struct Contact phonebook[], int numContacts) {**

**int i;**

**for (i = 0; i < numContacts; i++) {**

**fprintf(file, "%s,%s,%s\n", phonebook[i].name, phonebook[i].phone, phonebook[i].address);**

**}**

**}**

**void displayContacts(struct Contact phonebook[], int numContacts) {**

**if (numContacts == 0) {**

**printf("Phonebook is empty.\n");**

**return;**

**}**

**printf("Phonebook Contacts:\n");**

**int i;**

**for (i = 0; i < numContacts; i++) {**

**printf("Contact %d:\n", i + 1);**

**printf("Name: %s\nPhone: %s\nAddress: %s\n",**

**phonebook[i].name, phonebook[i].phone, phonebook[i].address);**

**printf("--------------------------------\n");**

**}**

**}**

**int searchContactByPhone(struct Contact phonebook[], int numContacts, const char \*phoneNumber) {**

**int i;**

**for (i = 0; i < numContacts; i++) {**

**if (strcmp(phonebook[i].phone, phoneNumber) == 0) {**

**printf("Contact Found:\n");**

**printf("Name: %s\nPhone: %s\nAddress: %s\n",**

**phonebook[i].name, phonebook[i].phone, phonebook[i].address);**

**return i; // Return index of the found contact**

**}**

**}**

**return -1; // Contact not found**

**}**

**void deleteContact(struct Contact phonebook[], int \*numContacts, int index) {**

**int i;**

**if (index >= 0 && index < \*numContacts) {**

**for (i = index; i < (\*numContacts) - 1; i++) {**

**phonebook[i] = phonebook[i + 1];**

**}**

**(\*numContacts)--;**

**printf("Contact deleted.\n");**

**} else {**

**printf("Invalid index.\n");**

**}**

**}**

**void addContact(struct Contact phonebook[], int \*numContacts) {**

**if (\*numContacts >= 100) {**

**printf("Phonebook is full.\n");**

**return;**

**}**

**struct Contact newContact;**

**printf("Enter name: ");**

**scanf("%s", newContact.name);**

**printf("Enter phone number: ");**

**scanf("%s", newContact.phone);**

**printf("Enter address: ");**

**scanf(" %[^\n]", newContact.address);**

**phonebook[\*numContacts] = newContact;**

**(\*numContacts)++;**

**printf("Contact added successfully.\n");**

**}**

**int main() {**

**int choice;**

**bool isAdmin = false;**

**printf("Select an option:\n");**

**printf("1. Admin\n");**

**printf("2. User\n");**

**scanf("%d", &choice);**

**if (choice == 1) {**

**isAdmin = validateAdminPassword();**

**if (!isAdmin) {**

**return 1; // Exit if admin password is incorrect**

**}**

**}**

**FILE \*phonebookFile = fopen("phonebook.txt", "a+"); // Open for appending and reading**

**if (phonebookFile == NULL) {**

**printf("Error opening file.\n");**

**return 1;**

**}**

**struct Contact phonebook[100];**

**int numContacts = 0;**

**loadContactsFromFile(phonebookFile, phonebook, &numContacts);**

**do {**

**printf("\nPhonebook Menu:\n");**

**printf("1. Add Contact\n");**

**printf("2. Display Contacts\n");**

**if (isAdmin) {**

**printf("3. Search Contact by Phone\n");**

**printf("4. Delete Contact\n");**

**}**

**printf("5. Exit\n");**

**printf("Enter your choice: ");**

**scanf("%d", &choice);**

**switch (choice) {**

**case 1:**

**addContact(phonebook, &numContacts);**

**if (isAdmin) {**

**addContactToFile(phonebookFile, &phonebook[numContacts - 1]); // Add to file**

**}**

**break;**

**case 2:**

**displayContacts(phonebook, numContacts);**

**break;**

**case 3:**

**if (isAdmin) {**

**char searchPhone[15];**

**printf("Enter the phone number to search: ");**

**scanf("%s", searchPhone);**

**int index = searchContactByPhone(phonebook, numContacts, searchPhone);**

**if (index == -1) {**

**printf("Contact not found.\n");**

**}**

**} else {**

**printf("Unauthorized operation.\n");**

**}**

**break;**

**case 4:**

**if (isAdmin) {**

**int deleteIndex;**

**printf("Enter the index of the contact to delete: ");**

**scanf("%d", &deleteIndex);**

**deleteContact(phonebook, &numContacts, deleteIndex);**

**saveContactsToFile(phonebookFile, phonebook, numContacts); // Update file after deletion**

**} else {**

**printf("Unauthorized operation.\n");**

**}**

**break;**

**case 5:**

**printf("Exiting...\n");**

**break;**

**default:**

**printf("Invalid choice. Please select a valid option.\n");**

**}**

**} while (choice != 5);**

**fclose(phonebookFile);**

**return 0;**

**}**