MITE
Invent Solutions

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

Accredited by NAAC with A+ Grade, An ISO 9001: 2015 CertifiedInstitution(A Unit of Rajalaxmi Education Trust®, Mangalore - 575001) Affiliated to V.T.U., Belagavi, Approved by AICTE, New Delhi

TECHNICAL TRAINING PROJECT

"VACCINATION REGISTRATION SYSTEM"

TEAM MEMBERS: BATCH 036

4MT21CS131: Samruddhi.H.R
4MT21CS140:Sharadhi Shedigume
4MT21CS145:Sanyuktha Shetty
4MT21CS157:Sinchana Shetty
4MT21CS169:Swati Metri
4MT20CS098:Miliyana Reena Dsouza

Table of Contents

1. Abstract

This project presents a comprehensive “Vaccine Registration System” implemented in C. The primary aim of this system is to facilitate the efficient registration, management, and scheduling of vaccine-related activities. It caters to both customers and administrators, offering distinct functionalities for each.
The project is organized into various modules, including registration forms, appointment scheduling, dashboard monitoring, vaccine information management, and appointment administration. Customers can register their personal and medical details, schedule appointments at their preferred vaccination centers, and view relevant information on the dashboard. Administrators, on the other hand, can manage vaccine-related data, oversee appointments, and maintain vaccine stock levels.
The C code project utilizes data structures to store critical information about customers, appointments, and vaccine-related data. It also incorporates file handling to persistently store data in text files for future reference.
Throughout the development process, challenges were encountered and overcome, leading to a robust system capable of serving its intended purpose effectively. The project lays the foundation for further enhancements, such as the integration of additional features and improved user interfaces.
In summary, this C code project contributes to the organization and management of vaccination processes, streamlining operations for both customers and administrators. Future iterations are expected to refine and expand the system's capabilities, making it an invaluable tool in the fight against preventable diseases.
Feel free to adjust the abstract as needed to better fit the specifics of your project or to add any additional details you find relevant.

2. Introduction

2.1 Background

In the face of global health challenges, the rapid development and deployment of vaccines have proven to be one of the most effective strategies for combating infectious diseases. The ongoing COVID-19 pandemic, among other health crises, has underscored the critical importance of efficient vaccination systems. Vaccination not only protects individuals but also contributes to achieving herd immunity, ultimately curbing the spread of diseases.
However, the successful administration of vaccines on a large scale necessitates a robust and organized system for registration, appointment scheduling, and vaccine management. Traditional paper-based methods often fall short in accommodating the demands of mass vaccination efforts. Recognizing this need, our project, the "Vaccine Registration System," was conceived.

2.2 Objectives

The primary objective of this project is to create a streamlined and user-friendly system that addresses the following key objectives:
1. Efficient Registration: Develop a system that allows individuals to register their personal information, medical history, and preferences seamlessly.
2. Appointment Scheduling: Facilitate the scheduling of vaccination appointments, taking into account preferred vaccination centres and available time slots.
3.Dashboard Monitoring: Provide a dashboard for both customers and administrators to monitor vaccine registration, appointments, and stock levels.
4.Vaccine Information Management: Enable administrators to manage vaccine-related data, including vaccine names, manufacturers, types, and doses required.

5.Appointment Administration: Empower administrators to oversee and manage appointments, including rescheduling, cancellations, and sending reminders.
6. Stock Management: Maintain accurate records of vaccine stock levels and allow for the addition of new stock as needed.

3. Technologies Used

1. C Programming Language: The entire codebase is written in the C programming language.
2. Standard C Libraries:
   - `<stdio.h>`: Standard I/O library for input and output operations.
   - `<stdlib.h>`: Standard library for general-purpose functions like memory allocation and program termination.
   - `<string.h>`: Standard library for string manipulation functions.
3. File Handling: The code uses file operations to read from and write to files such as "registration.txt," "appointments.txt," "centers.txt," "vaccine_info.txt," and "stock.txt." These files likely store user and appointment data.
4. Menu-driven Interface: The program relies on a menu-driven user interface that allows users (both customers and administrators) to interact with the system.
5. Structures: The code defines several structures to organize data, such as `struct PersonalInfo`, `struct ContactDetails`, `struct MedicalHistory`, `struct PreferredCenter`, `struct Appointment`, `struct Vaccine`, and `struct VaccineStock`. These structures hold different types of information, such as personal details, medical history, and vaccine information.
6. User Input: The program takes user input for various purposes, including registration, appointment scheduling, and information updates.
7. Control Structures: The code uses control structures like `if-else` and `switch` statements to make decisions based on user input.
8. Functions: The code is modularized into functions, making it easier to manage and understand. Some of the key functions include `customerMenu()`, `adminMenu()`, `registrationForm()`, `appointmentScheduling()`, `dashboard()`, and more.
9.Data Storage: The program appears to store user registration and appointment data in text files, which are read and written using file handling functions.
10. Error Handling: Basic error handling is implemented in cases where files can't be opened or memory allocation fails.

4. System Architecture

4.1 Front-End:
The front-end of this application is entirely text-based, designed for the command-line interface (CLI). Users interact with the application by typing commands and receiving text-based responses. There is no graphical user interface (GUI) involved in this system.

4.2 Back-End:
The back-end of the application includes the core logic and data processing. It consists of several functions and components:

- Main Function (`main()`):The entry point of the application, responsible for presenting the role selection menu (customer or admin) and redirecting the user to the appropriate menu.
- Customer Menu (`customerMenu()`):Handles the customer-related functionalities, including registration, appointment scheduling, dashboard, profile management, and program exit.
- Admin Menu (`adminMenu()`): Manages admin-specific tasks, such as vaccine information management, appointment management, and adding vaccine stock.

- Data Structures: Various data structures are defined using C structures to represent entities such as personal information, contact details, medical history, preferred centers, appointments, vaccines, and vaccine stock.
- File Handling: The application uses file I/O to read from and write to text files to store information about registrations, appointments, vaccine information, centers, and vaccine stock. These files serve as the database.
- Functions: The application is divided into several functions, each responsible for specific tasks, such as handling registration forms, scheduling appointments, managing centers, viewing appointments, etc.
- Error Handling: There is some basic error handling for file operations, but it could be further improved for robustness.

4.3 Database:

The application uses text files as a simple form of a database to store and retrieve information. Each type of data is stored in a separate text file:

*Registration Data (`registration.txt`): Stores information about registered customers, including personal details, contact information, medical history, and preferred centres.

*Appointment Data (`appointments.txt`): Contains details about scheduled appointments, including Aadhaar numbers, dates, times, and center names.

*Vaccine Information Data (`vaccine_info.txt`): Stores information about available vaccines, including their names, manufacturers, types, and doses required.

*Center Data (`centers.txt`): Stores details about preferred centres, such as centre names, addresses, cities, states, and zip codes.

*Vaccine Stock Data (`stock.txt`): Contains information about vaccine stock quantities for different vaccines.

5. Project Modules

1. Main Menu (main):
   - This module is the entry point of the program.
   - It presents the user with two options: Customer and Admin.
   - Based on the user's choice, it directs them to the respective menu.

2. Customer Menu (customerMenu):
   - Provides options for customers to perform various actions related to their registration, appointments, profile, and more.
   - It includes functions such as Registration Form, Appointment Scheduling, Dashboard, Profile Management, and Exit.
   - The user can choose one of these options, and the respective function is called to handle their request.

3. Admin Menu (adminMenu):
   - Provides options for administrators to manage vaccine information, appointments, add vaccine stock, and exit.
   - It includes functions such as Manage Vaccine Information, Manage Appointments, Add Vaccine Stock, and Exit.
   - Similar to the customer menu, the user's choice directs them to the appropriate function.

4. Registration Form (registrationForm):
   - Collects personal information, contact details, medical history, and preferred center information from customers.

- Validates and stores this information in a file (registration.txt) for future reference.

5. Appointment Scheduling (appointmentScheduling):
- Allows customers to schedule appointments by providing their Aadhaar number, date, time, and preferred center name.
- Records the appointment details in a file (appointments.txt) for tracking.

6. Dashboard (dashboard):
- Provides a dashboard for customers with options to monitor registrations, view vaccine stock, manage centers, and generate a general report.
- Users can choose any of these options, and the corresponding function is executed.

7. Manage Centers (manageCenters):
- Allows administrators to add, edit, delete, and view vaccination centers.
- Information about centers is stored in a file (centers.txt).

8. Manage Appointments (manageAppointments):
- Provides functionality for administrators to view appointments, reschedule, cancel, and send reminders to customers.

9. Vaccine Information (vaccineInformation):
- Allows administrators to manage vaccine-related information.
- Options include adding, editing, deleting, and viewing vaccine details.
- Vaccine information is stored in a file (vaccine_info.txt).

10. Profile Management (profileManagement):
- Allows customers to update their personal information, change appointments, and update their medical reports.

11. General Report (generalReport):
- Generates a general report by reading customer registration data from the registration.txt file.
- This report includes personal information, contact details, medical history, and preferred center information.

12. Update Information (updateInformation):
- Part of the profile management module, it lets customers update their personal information.

13. Change Appointment (changeAppointment):
- Part of the profile management module, allows customers to change their appointment details.

14. Update Medical Report (updateMedicalReport):
- Part of the profile management module, lets customers update their medical history, prescription, allergies, and current medications.

15. Add Stock (addStock):
- Part of the admin menu, allows administrators to add vaccine stock information, including the vaccine name and quantity.

These modules collectively make up the Vaccine Registration System and provide functionalities for both customers and administrators to manage their vaccination-related tasks efficiently.

6. Design and Implementation

6.1 Front-End Design

The front-end of this application is entirely text-based, designed for the command-line interface (CLI). Users interact with the application by typing commands and receiving text-based responses. There is no graphical user interface (GUI) involved in this system.

6.2 Back-End Design

The back-end design of the system involves the core logic and functionality. Here are some key aspects:
Modularization: The code is structured into functions for different tasks, making it modular and easier to maintain. Each major feature has its own function, such as `registrationForm()`, `appointmentScheduling()`, and so on.
File Handling: Data is stored and retrieved from text files, which can work for small-scale systems. However, for larger systems, you might consider using a relational database management system (RDBMS) for better data organization, retrieval, and scalability.
User Roles: The system supports two user roles: customers and administrators, with separate menus and functionalities for each role.
Input Validation: Basic input validation is performed, but additional validation and error handling could be added to ensure data integrity and prevent issues with incorrect inputs.
Appointment Management: The system allows customers to schedule appointments and administrators to manage appointments, including rescheduling and cancellation.
Profile Management: Customers can update their information, change appointments, and update their medical reports.
Vaccine Information Management: Administrators can add, edit, and delete vaccine information.
Center Management: Administrators can add, edit, and delete vaccination centers.
Reporting: The system provides a general report feature, allowing administrators to view the information of registered users.

6.3 Database Design

Registration Data: User registration information, including personal details, contact details, medical history, and preferred vaccination center, is stored in a text file named "registration.txt." Each user's data is separated by a delimiter.

Appointment Data: Appointment information, including user Aadhaar number, date, time, and center name, is stored in a text file named "appointments.txt." Each appointment entry is separated by a delimiter.

Vaccine Information: Vaccine details, such as name, manufacturer, type, and doses required, are stored in a text file named "vaccine_info.txt." Each vaccine entry is separated by a delimiter.

Center Information: Information about vaccination centers, including name, address, city, state, and zipcode, is stored in a text file named "centers.txt." Each center entry is separated by a delimiter.

7. Features and Functionality

1.User Role Selection (Customer/Admin):
- During startup, users are prompted to select their role as either a customer or an admin.
- Importance: This feature ensures that the system caters to both end-users (customers) and administrators with different sets of functionalities.

2. Customer Menu:
- Customers have access to various options, including registration, appointment scheduling, dashboard, profile management, and exit.
- Importance: Customers can easily navigate and utilize the system's functionalities through this menu.

3. Admin Menu:
- Administrators have access to options such as managing vaccine information, appointments, adding vaccine stock, and exiting.
- Importance: Admins can control vaccine-related data and appointments to ensure smooth vaccination processes.

4.Registration Form:
- Customers can provide their personal information, contact details, medical history, and preferred vaccination center.
- Importance: This feature collects necessary data for registration and appointment scheduling.

5. Appointment Scheduling:
- Customers can schedule vaccination appointments by providing their Aadhaar number, date, time, and preferred vaccination center.
- Importance: Helps customers secure vaccination slots and manage appointments efficiently.

6.Dashboard:
- Customers can access various functionalities, including monitoring registration, viewing vaccine stock, managing vaccination centers, and generating a general report.
- Importance: Provides customers with an overview and control of their vaccination-related activities.

7.Monitoring Registration:
- Allows customers to view registered individuals' details, such as name, Aadhaar number, contact information, and medical history.
- Importance: Provides transparency and access to personal data for verification.

8. Viewing Vaccine Stock:
- Admins can check the available vaccine stock, including vaccine name and quantity.
- Importance: Ensures admins are aware of the vaccine supply status, aiding in stock management.

9. Manage Centers:
- Admins can add, edit, or delete vaccination centers and view all registered centers.
- Importance: Facilitates the management of vaccination center data and locations.

10. Vaccine Information Management:
- Admins can add, edit, delete, or view vaccine information, including name, manufacturer, type, and required doses.
- Importance: Ensures accurate and up-to-date vaccine data for scheduling appointments.

11. Manage Appointments:
- Admins can view appointments, reschedule them, cancel appointments, and send reminders.
- Importance: Admins can maintain and organize the vaccination schedule effectively.

12. Profile Management:
- Customers can update their personal information, change appointments, and update their medical reports.
- Importance: Empowers customers to maintain their records and manage their vaccination-related details.

13. Updating Information:
- Customers can update their personal information, such as name, date of birth, gender, contact details, and address.
- Importance: Keeps customer records accurate and up-to-date.

14. Changing Appointment:
- Customers can reschedule their vaccination appointments by selecting a new date, time, and center.
- Importance: Offers flexibility to accommodate changing schedules or preferences.

15. Updating Medical Report:

- Customers can update their medical history, prescription, allergies, and current medications.
- Importance: Ensures healthcare providers have access to the most recent medical information during vaccination.

16. Adding Vaccine Stock:
- Admins can add new vaccine stock to the system, including the vaccine name and quantity.
- Importance: Helps maintain an adequate supply of vaccines for vaccination centers.

17. Exit Option:
- Provides an option to exit the application gracefully.
- Importance: Allows users to close the application when they are done.

8. Testing

1.Unit Testing:
- Function-Level Testing:Each function in the code can be individually tested to ensure that it performs the intended operations correctly. For example, functions like `registrationForm()`, `addCenter()`, `addVaccineInformation()`, etc., can be tested to validate that they collect and store data accurately.

2. Integration Testing:
- Testing Function Interactions: Integration testing ensures that different functions work together as expected. For example, you can test the flow from registration (`registrationForm()`) to appointment scheduling (`appointmentScheduling()`) to verify that data is passed correctly between these functions.

3. User Acceptance Testing (UAT):
- Customer User Acceptance Testing:This involves having actual customers (users) interact with the system to ensure it meets their requirements and expectations. Customers should test functionalities such as registration, appointment scheduling, and profile management. They can provide feedback on usability, correctness, and any issues they encounter.
- Admin User Acceptance Testing: Administrators or staff members can perform UAT for admin-related functionalities, such as managing vaccine information, appointments, and centers. They can validate that these functionalities align with their operational needs.

4. End-to-End Testing:
- This type of testing verifies the entire flow of the system, from a user's perspective. You can simulate user journeys, starting with registration, appointment scheduling, and viewing data. End-to-end testing ensures that the entire system works cohesively.

5. Boundary Testing:
- Test the system's behavior at its limits. For example, test with invalid input data, maximum and minimum values, and boundary conditions to ensure the system handles them gracefully.

6. Load Testing:
- Assess how the system performs under heavy loads. This can help identify bottlenecks or performance issues when many users are accessing the system simultaneously, especially for functionalities like appointment scheduling.

7. Security Testing:
- Verify that the system is secure against common security threats, such as SQL injection, cross-site scripting (XSS), and unauthorized access. Ensure that user data is handled securely and that sensitive information is protected.

9. Challenges Faced

1. Error Handling: One common challenge is handling errors effectively. Ensure that your code has proper error-checking mechanisms in place, such as checking if file operations (e.g., fopen) were successful and handling these errors gracefully (e.g., using perror or error messages).

2. Data Validation: It's crucial to validate user input to prevent data corruption or unexpected behavior. You may need to implement input validation routines to ensure that the data entered is of the correct format and within reasonable bounds.

3. File Operations: Dealing with file operations can be tricky. Ensure that you have adequate error handling for file opening, writing, and reading. Additionally, consider implementing backup mechanisms or version control for data files to prevent data loss.

4. Data Consistency: Maintaining data consistency across multiple files can be challenging. Ensure that you design your data storage and retrieval functions to keep data synchronized.

5. Data Privacy and Security: If this system deals with personal or sensitive information, security becomes a paramount concern. Implement proper data encryption and access control mechanisms to protect user data.
6. Testing: Comprehensive testing is essential to ensure your system works as expected. This includes unit testing for individual functions, integration testing for components, and end-to-end testing for the entire system.
7. Data Backup and Recovery: Implement a robust backup and recovery system to prevent data loss in case of system failures or accidental data deletion.
8. Appointment Scheduling Logic: Ensure that your appointment scheduling logic is robust and can handle various scenarios, such as conflicts or rescheduling.

10. Future Enhancements

1. User Authentication: Implement a secure user authentication system to ensure that only authorized users can access and modify their profiles or schedule appointments. This would enhance the security of the system.

2. Appointment Reminders: Implement automated appointment reminders via email or SMS to help reduce missed appointments and improve overall efficiency.

3. Integration with Health Records: Integrate the system with electronic health records (EHRs) to provide healthcare providers with access to patients' medical histories and vaccination records, ensuring comprehensive care.

4. Vaccine Availability Checker: Add a feature that allows users to check the availability of vaccines at different vaccination centers before scheduling an appointment.

5. Vaccine Eligibility Checker: Implement a feature that checks a user's eligibility for a particular vaccine based on their medical history, age, and other relevant factors.

6. Reporting and Analytics: Create detailed reports and analytics to monitor vaccination progress, track vaccine distribution, and identify trends or areas that need attention.

7. Multi-language Support: Offer support for multiple languages to make the system more accessible to a diverse user base.

8. Mobile Application: Develop a mobile app version of the system for users to register, schedule appointments, and receive notifications conveniently on their smartphones.

9. QR Code Integration: Generate and utilize QR codes for appointment verification, making the check-in process faster and more efficient at vaccination centers.

10. Waitlist Management: Implement a waitlist feature that allows users to join a waitlist for appointments when none are available, and they can be notified when slots become free.

11. Feedback and Surveys: Collect feedback from users after their vaccination appointments to gather insights and continuously improve the system.

12. Emergency Notifications: Enable emergency notifications and alerts to inform users about critical updates or changes related to vaccination, such as vaccine recalls or changes in eligibility criteria.

13. Geolocation Integration: Utilize geolocation services to help users find the nearest vaccination centers and display real-time traffic information for their convenience.

14. Data Security Enhancements: Strengthen data security measures to protect users' personal and medical information, considering compliance with data protection regulations.

15. Customizable Templates: Allow healthcare providers to create customizable templates for medical records and reports, ensuring that the system caters to various healthcare practices.

16. Telehealth Integration: Enable telehealth features, allowing users to consult with healthcare professionals virtually, especially for post-vaccination follow-ups and consultations.

17. Vaccine Expiration Alerts: Implement alerts and notifications to healthcare providers regarding vaccine expiration dates to minimize wastage.

18. AI-Based Predictive Modeling: Use artificial intelligence and predictive modeling to forecast vaccine demand, allocate resources efficiently, and anticipate potential bottlenecks.

19. Integration with Government Systems: Collaborate with government health departments to integrate the system with their vaccination tracking systems for more accurate and up-to-date data.

20. Accessibility Features: Ensure that the system complies with accessibility standards (e.g., WCAG) to make it accessible to users with disabilities.

11. Conclusion

In conclusion, the development of a vaccination registration system in C programming offers a practical and efficient solution for managing the critical task of vaccine distribution and administration. This system leverages the power of C programming to create a reliable and user-friendly platform that serves both customers and administrators.

With features such as role-based access, customer registration, appointment scheduling, and data management, the system provides a streamlined and organized approach to vaccination registration. It ensures that individuals can easily access vaccination services, schedule appointments, and update their information while administrators can efficiently oversee vaccine stocks, appointments, and vital information.

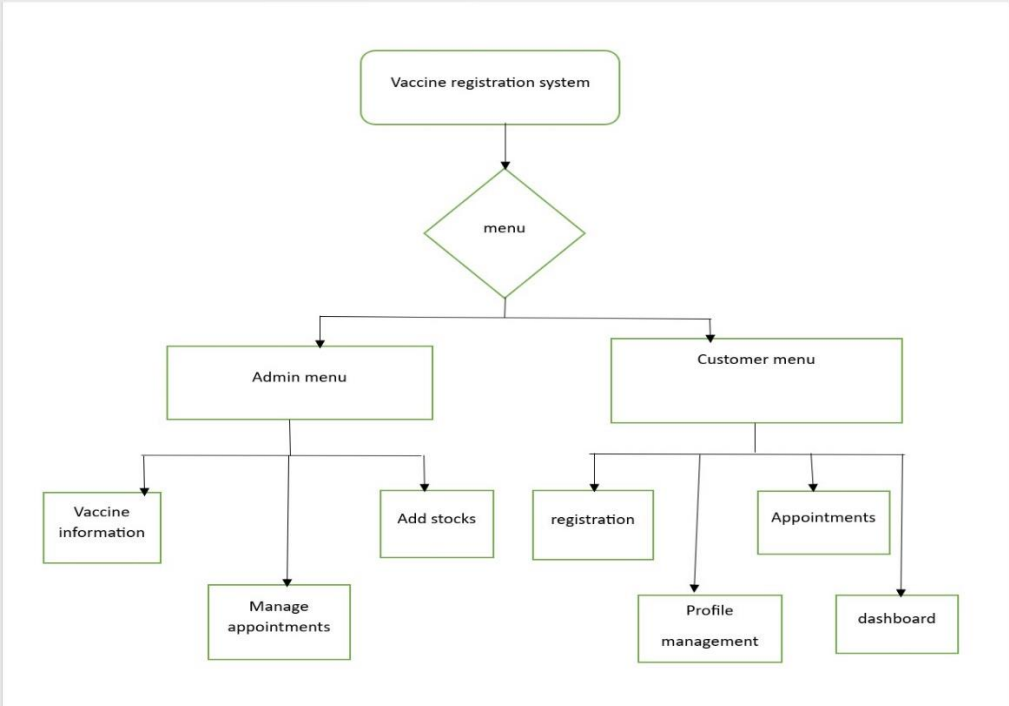
By choosing C programming for this project, we benefit from its performance and flexibility, allowing for a robust and responsive system that can adapt to changing requirements and scale as needed.
In an era where efficient vaccination management is of utmost importance, this C programming-based vaccination registration system plays a crucial role in ensuring that vaccines are distributed effectively, helping to safeguard public health and well-being..

12.References:

1. C Programming Documentation - https://www.cprogramming.com/
2. Stack Overflow - https://stackoverflow.com/
3. C Standard Library Documentation - https://en.cppreference.com/w/c/header

13. Appendices

Appendix A: Flowcharts



Appendix B: Data Structures

```c
struct PersonalInfo {
    char name[50];
    char aadhaar[15];
    char dateOfBirth[15];
    char gender[10];
};
struct ContactDetails {
    char email[50];
    char phone[15];
    char emergencyContact[50];
    char emergencyPhone[15];
    char address[100];
};
struct MedicalHistory {
    char medicalHistory[200];
    char prescription[200];
    char allergies[100];
```

```
    char currentMedications[200];
};
struct PreferredCenter {
    char centerName[50];
    char address[100];
    char city[50];
    char state[50];
    char zipcode[10];
};
struct Appointment {
    char aadhaar[15];
    char date[15];
    char time[10];
    char centerName[50]; // Adding the center name field
};
struct Vaccine {
    char name[50];
    char manufacturer[50];
    char type[20];
    int dosesRequired;
};
struct VaccineStock {
    char vaccineName[50];
    int quantity;
};//components
```

Appendix C: Sample Input/Output

1. User Registration:

Input:

```
Enter your name: John Doe
Enter your Aadhaar Number: 123456789012
Enter your Date of Birth (YYYY-MM-DD): 1990-01-15
Enter your Gender: Male
Enter your Email: johndoe@email.com
Enter your Phone Number: 123-456-7890
Enter your Emergency Contact: Jane Doe
Enter your Emergency Contact Phone: 987-654-3210
Enter your Address: 123 Main St, Cityville, State
Enter your Medical History (if any): None
Enter your Allergies (if any): None
Output:
Registration successful. Your unique ID is: 1001
```

2. Appointment Scheduling:

Input:

```
Enter your Aadhaar Number: 123456789012
Enter the Date for the Appointment (YYYY-MM-DD): 2023-09-15
Enter the Preferred Time (HH:MM AM/PM): 10:00 AM
Enter the Preferred Vaccination Center: City Medical Center
```

Output:

Appointment scheduled successfully for 2023-09-15 at 10:00 AM in City Medical Center.

3.Viewing Available Vaccination Centers:
Input:
Show available vaccination centers
Output:
Available Vaccination Centers:
1. City Medical Center
2. County Health Clinic
3. Suburban Hospital

4.Checking Vaccine Information:
Input:
Tell me about the available vaccines.
Output:
Available Vaccines:
1. COVID-19 Vaccine
- Manufacturer: Vaccine Co.
- Type: mRNA
- Doses Required: 2
2. Flu Vaccine
- Manufacturer: Pharma Corp.
- Type: Inactivated
- Doses Required: 1

5. Checking Vaccine Stock:
Input:
Check vaccine stock.
Output:
Vaccine Stock:
- COVID-19 Vaccine: 150 doses
- Flu Vaccine: 50 doses
6. Canceling an Appointment:
Input:
Enter your unique ID: 1001
Please confirm your appointment cancellation (Y/N): Y
Output:
Appointment for ID 1001 on 2023-09-15 at 10:00 AM in City Medical Center has been canceled.
7.Viewing Scheduled Appointments:
Input:
View my scheduled appointments for Aadhaar Number 123456789012.
Output:
Scheduled Appointments for Aadhaar Number 123456789012:
1. Date: 2023-09-15, Time: 10:00 AM, Location: City Medical Center

8.Checking User Medical History:
Input:
Check medical history for ID 1001.
Output
Medical History for ID 1001:

- None

9.Checking Allergies:

Input:

Check allergies for ID 1001.

Output:

Allergies for ID 1001:

- None

10. Updating User Information:

Input:

Update phone number for ID 1001.

New Phone Number: 555-123-4567

Output:

Phone number for ID 1001 updated successfully.

11. Viewing Vaccination Records:

Input:

View vaccination records for ID 1001.

Output:

Vaccination Records for ID 1001:

1. COVID-19 Vaccine

- Date: 2023-09-15, Time: 10:00 AM, Location: City Medical Center

Appendix E: Error Handling

Error Handling in the Program:

1. User Input Validation:

- The program validates user input to ensure it meets the required format and constraints. For example, when users enter their Aadhaar number, date of birth, or phone numbers, the program checks for valid formats and lengths.
- If user input is invalid or incomplete, the program displays a user-friendly error message explaining the issue and prompts the user to correct it.
- Example Error Messages:
  - "Invalid Aadhaar number. Please enter a 12-digit number."
  - "Date of birth must be in the format YYYY-MM-DD."

2. Data Integrity Checks:

- Before processing appointments or updating vaccine stock, the program checks the integrity of the data in various files.
- If inconsistencies or unexpected data are detected (e.g., scheduling an appointment for a vaccine that doesn't exist or for a center that isn't in the system), the program logs the error and prevents the operation from proceeding.
- Example Error Message:
  - "Error: Appointment cannot be scheduled. The selected vaccine or vaccination center is not available."

3. File Handling Errors:

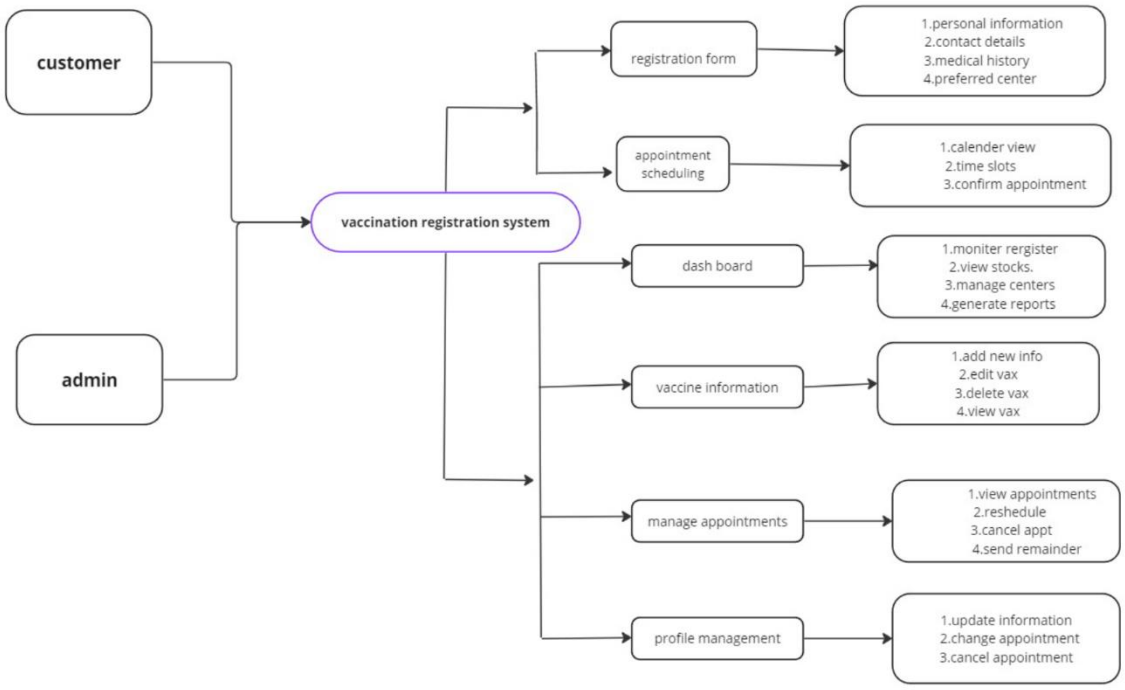
- The program handles errors related to file operations, such as reading from or writing to data files.
- If the program encounters issues like missing data files or permissions problems, it notifies the user and provides guidance on how to resolve the issue.
- Example Error Message:
  - "Error: Unable to access the 'registration.txt' file. Please check file permissions and file existence."

4. Database Connection Errors:

- If the program uses a database to store data, it handles database connection errors gracefully.

- It provides informative error messages in case of database connection failures and advises the user to check database settings or connectivity.
- Example Error Message:
  - "Error: Unable to connect to the database. Please ensure your database server is running and the connection settings are correct."

Appendix E: Mind Map



13.1 Screenshots

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct PersonalInfo {
    char name[50];
    char aadhaar[15];
    char dateOfBirth[15];
    char gender[10];
};
struct ContactDetails {
    char email[50];
    char phone[15];
    char emergencyContact[50];
    char emergencyPhone[15];
```

```
    char address[100];
};
struct MedicalHistory {
    char medicalHistory[200];
    char prescription[200];
    char allergies[100];
    char currentMedications[200];
};
struct PreferredCenter {
    char centerName[50];
    char address[100];
    char city[50];
    char state[50];
    char zipcode[10];
};
struct Appointment {
    char aadhaar[15];
    char date[15];
    char time[10];
    char centerName[50]; // Adding the center name field
};
struct Vaccine {
    char name[50];
    char manufacturer[50];
    char type[20];
    int dosesRequired;
};
struct VaccineStock {
    char vaccineName[50];
    int quantity;
};//components
void customerMenu();
void adminMenu();
int main() { //mainmenue
    int role;
    printf("Welcome to the Vaccine Registration System!\n");
    printf("Please select your role:\n");
    printf("1. Customer\n");
    printf("2. Admin\n");
    printf("Enter your choice: ");
    scanf("%d", &role);
    if (role == 1) {
        customerMenu();
    } else if (role == 2) {
        adminMenu();
    } else {
        printf("Invalid role selection.\n");
    }
    return 0;
}
void customerMenu() {
    int choice;
```

```
    do {
        printf("\nCustomer Menu\n");
        printf("1. Registration Form\n");
        printf("2. Appointment Scheduling\n");
        printf("3. Dashboard\n");
        printf("4. Profile Management\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1: {
                    struct PersonalInfo Person;
                    struct ContactDetails Contact;
                    struct MedicalHistory History;
                    struct PreferredCenter Center;
                    printf("\nRegistration Form\n");
                    printf("Enter Name: ");
                    scanf("%s", Person.name);
                    printf("Enter Aadhaar Number: ");
                    scanf("%s", Person.aadhaar);
                    printf("Enter Date of Birth (YYYY-MM-DD): ");
                    scanf("%s", Person.dateOfBirth);
                    printf("Enter Gender: ");
                    scanf("%s", Person.gender);
                    printf("Enter Email: ");
                    scanf("%s", Contact.email);
                    printf("Enter Phone Number: ");
                    scanf("%s", Contact.phone);
                    printf("Enter Emergency Contact: ");
                    scanf("%s", Contact.emergencyContact);
                    printf("Enter Emergency Phone: ");
                    scanf("%s", Contact.emergencyPhone);
                    printf("Enter Address: ");
                    scanf("%s", Contact.address);
                    printf("Enter Medical History: ");
                    scanf("%s", History.medicalHistory);
                    printf("Enter Prescription: ");
                    scanf("%s", History.prescription);
                    printf("Enter Allergies: ");
                    scanf("%s", History.allergies);
                    printf("Enter Current Medications: ");
                    scanf("%s", History.currentMedications);
                    printf("Enter Preferred Center Name: ");
                    scanf("%s", Center.centerName);
                    printf("Enter Address of Preferred Center: ");
                    scanf("%s", Center.address);
                    printf("Enter City of Preferred Center: ");
                    scanf("%s", Center.city);
                    printf("Enter State of Preferred Center: ");
                    scanf("%s", Center.state);
                    printf("Enter Zipcode of Preferred Center: ");
                    scanf("%s", Center.zipcode);
```

```
                    FILE *file = fopen("registration.txt", "a");
                    if (file == NULL) {
                        printf("Error opening file for writing.\n");
                        return;
                    }
                    fprintf(file, "\n\n\nName: %s\n", Person.name);
                    fprintf(file, "Aadhaar Number: %s\n", Person.aadhaar);
                    fprintf(file, "Date of Birth: %s\n", Person.dateOfBirth);
                    fprintf(file, "Gender: %s\n", Person.gender);
                    fprintf(file, "Email: %s\n", Contact.email);
                    fprintf(file, "Phone Number: %s\n", Contact.phone);
                    fprintf(file, "Emergency Contact: %s\n", Contact.emergencyContact);
                    fprintf(file, "Emergency Phone: %s\n", Contact.emergencyPhone);
                    fprintf(file, "Address: %s\n", Contact.address);
                    fprintf(file, "Medical History: %s\n", History.medicalHistory);
                    fprintf(file, "Prescription: %s\n", History.prescription);
                    fprintf(file, "Allergies: %s\n", History.allergies);
                    fprintf(file, "Current Medications: %s\n",
History.currentMedications);
                    fprintf(file, "Preferred Center Name: %s\n", Center.centerName);
                    fprintf(file, "Address of Preferred Center: %s\n", Center.address);
                    fprintf(file, "City of Preferred Center: %s\n", Center.city);
                    fprintf(file, "State of Preferred Center: %s\n", Center.state);
                    fprintf(file, "Zipcode of Preferred Center: %s\n", Center.zipcode);
                    printf("----------------------------\n");
                    fclose(file);
                    printf("Registration completed successfully.\n");

                    }

                    break;
            case 2:
                  {
                    printf("\nAppointment Scheduling\n");
                    struct Appointment newAppointment;
                    printf("Enter Aadhaar Number: ");
                    scanf("%s", newAppointment.aadhaar);
                    printf("Enter Date (YYYY-MM-DD): ");
                    scanf("%s", newAppointment.date);
                    printf("Enter Time (HH:MM AM/PM): ");
                    scanf("%s", newAppointment.time);
                    printf("Enter Preferred Center Name: ");
                    scanf("%s", newAppointment.centerName);
                    FILE *file = fopen("appointments.txt", "a");
                    if (file == NULL) {
                        printf("Error opening appointments file for writing.\n");
                        return;
                    }
                    fprintf(file, "Aadhaar Number: %s\n", newAppointment.aadhaar);
                    fprintf(file, "Date: %s\n", newAppointment.date);
                    fprintf(file, "Time: %s\n", newAppointment.time);
                    fprintf(file, "Center Name: %s\n", newAppointment.centerName);
```

```
                    fprintf(file, "--------------------\n");
                    fclose(file);
                    printf("Appointment scheduled successfully.\n");
                   }
                   break;
            case 3:{
                    int choice;
                    do {
                        printf("\nDashboard\n");
                        printf("1. Monitor Register\n");
                        printf("2. View Stock\n");
                        printf("3. Manage Centers\n");
                        printf("4. General Report\n");
                        printf("5. Back to Main Menu\n");
                        printf("Enter your choice: ");
                        scanf("%d", &choice);
                        switch (choice) {
                            case 1:
                                 {
                                    FILE *file = fopen("registration.txt", "r");
                                    if (file == NULL) {
                                        printf("Error opening file.\n");
                                        return;
                                    }
                                    char line[200];
                                    printf("\nRegistered People:\n");
                                    printf("------------------\n");
                                    while (fgets(line, sizeof(line), file)) {
                                        printf("%s", line);
                                    }
                                    fclose(file);
                                }
                                break;
                            case 2:
                                   {
                                        printf("Viewing stock...\n");

                                        FILE *file = fopen("stock.txt", "r");
                                        if (file == NULL) {
                                            printf("Error opening stock file.\n");
                                            return;
                                        }
                                        struct VaccineStock stock;
                                        printf("Vaccine Name\tQuantity\n");
                                        printf("-------------------------\n");
                                        while (fread(&stock, sizeof(struct VaccineStock),
1, file) == 1) {

                                            printf("%-15s\t%d\n", stock.vaccineName,
stock.quantity);

                                        }
                                        fclose(file);
                                    }
```

```
                                    break;
                            case 3:
                                   {
                                    int choice;
                                    do {
                                        printf("\nManage Centers\n");
                                        printf("1. Add Center\n");
                                        printf("2. Edit Center\n");
                                        printf("3. Delete Center\n");
                                        printf("4. View Centers\n");
                                        printf("5. Back to Dashboard\n");
                                        printf("Enter your choice: ");
                                        scanf("%d", &choice);
                                        switch (choice) {
                                            case 1:
                                                {
                                                    struct PreferredCenter newCenter;
                                                    printf("\nAdding a new center\n");
                                                    printf("Enter Center Name: ");
                                                    scanf("%s", newCenter.centerName);
                                                    printf("Enter Address: ");
                                                    scanf("%s", newCenter.address);
                                                    printf("Enter City: ");
                                                    scanf("%s", newCenter.city);
                                                    printf("Enter State: ");
                                                    scanf("%s", newCenter.state);
                                                    printf("Enter Zipcode: ");
                                                    scanf("%s", newCenter.zipcode);
                                                    FILE *file = fopen("centers.txt",
"a");

                                                    if (file == NULL) {
                                                        printf("Error opening file for

writing.\n");

                                                        return;
                                                    }
                                                    fprintf(file, "Center Name: %s\n",

newCenter.centerName);

                                                    fprintf(file, "Address: %s\n",

newCenter.address);

                                                    fprintf(file, "City: %s\n",

newCenter.city);

                                                    fprintf(file, "State: %s\n",

newCenter.state);

                                                    fprintf(file, "Zipcode: %s\n",

newCenter.zipcode);

                                                    fprintf(file, "--------------------

\n");

                                                    fclose(file);
                                                    printf("New center added

successfully.\n");

                                                }
                                                break;
```

```
                                        case 2:
                                            {
                                                printf("\nEditing a center\n");
                                                int selectedCenterIndex;
                                                FILE *file = fopen("centers.txt",
"r+");

                                                if (file == NULL) {
                                                    printf("Error opening file for
reading and writing.\n");

                                                    return;
                                                }
                                                fseek(file, selectedCenterIndex *
sizeof(struct PreferredCenter), SEEK_SET);

                                                struct PreferredCenter editedCenter;
                                                fread(&editedCenter, sizeof(struct
PreferredCenter), 1, file);

                                                printf("Enter new Center Name: ");
                                                scanf("%s", editedCenter.centerName);
                                                printf("Enter new Address: ");
                                                scanf("%s", editedCenter.address);
                                                printf("Enter new City: ");
                                                scanf("%s", editedCenter.city);
                                                printf("Enter new State: ");
                                                scanf("%s", editedCenter.state);
                                                printf("Enter new Zipcode: ");
                                                scanf("%s", editedCenter.zipcode);
                                                fseek(file, selectedCenterIndex *
sizeof(struct PreferredCenter), SEEK_SET);

                                                fwrite(&editedCenter, sizeof(struct
PreferredCenter), 1, file);

                                                fclose(file);
                                                printf("Center information edited
successfully.\n");

                                            }
                                            break;
                                        case 3:
                                            {
                                                printf("\nDeleting a center\n");
                                                int selectedCenterIndex;
                                                FILE *file = fopen("centers.txt",
"r+");

                                                if (file == NULL) {
                                                    printf("Error opening centers
file.\n");

                                                    return;
                                                }
                                                long int offset = selectedCenterIndex
* sizeof(struct PreferredCenter);

                                                fseek(file, 0, SEEK_END);
                                                long int fileSize = ftell(file);
                                                for (long int i = offset +
sizeof(struct PreferredCenter); i < fileSize; i++) {
```

```
                fseek(file, i, SEEK_SET);
                char c = fgetc(file);
                fseek(file, i - sizeof(struct
PreferredCenter), SEEK_SET);
                fputc(c, file);
            }
            ftruncate(fileno(file), fileSize -
sizeof(struct PreferredCenter));
            fclose(file);
            printf("Center information deleted
successfully.\n");
        }
        break;
    case 4:
        {
            printf("\nViewing all centers\n");
            FILE *file = fopen("centers.txt",
"r");
            if (file == NULL) {
                printf("Error opening centers
file.\n");
                return;
            }
            struct PreferredCenter center;
            char line[200];
            printf("List of registered
centers:\n");
            printf("============================\n
");
             while (fgets(line, sizeof(line),
file)) {
                printf("%s", line);
            }
            while (fread(&center, sizeof(struct
PreferredCenter), 1, file) == 1) {
                printf("Center Name: %s\n",
center.centerName);
                printf("Address: %s\n",
center.address);
                printf("City: %s\n", center.city);
                printf("State: %s\n",
center.state);
                printf("Zipcode: %s\n",
center.zipcode);
                printf("-------------------------
-\n");
            }
            fclose(file);
        }
        break;
    case 5:
        return;
```

```
                                        default:
                                            printf("Invalid choice. Please try
again.\n");
                                    }
                                } while (1);
                            }
                            break;
                        case 4:
                            {
                            FILE *file = fopen("registration.txt", "r");
                            if (file == NULL) {
                                printf("Error opening registration file.\n");
                                return;
                            }
                            printf("\nGeneral Report\n");
                            printf("================================\n");
                            struct PersonalInfo person;
                            struct ContactDetails contact;
                            struct MedicalHistory history;
                            struct PreferredCenter center;
                            while (fread(&person, sizeof(struct PersonalInfo), 1,
file) == 1) {
                                fread(&contact, sizeof(struct ContactDetails), 1,
file);
                                fread(&history, sizeof(struct MedicalHistory), 1,
file);
                                fread(&center, sizeof(struct PreferredCenter), 1,
file);
                                printf("\nName: %s\n", person.name);
                                printf("Aadhaar: %s\n", person.aadhaar);
                                printf("Date of Birth: %s\n", person.dateOfBirth);
                                printf("Gender: %s\n", person.gender);
                                printf("Email: %s\n", contact.email);
                                printf("Phone: %s\n", contact.phone);
                                printf("Emergency Contact: %s\n",
contact.emergencyContact);
                                printf("Emergency Phone: %s\n",
contact.emergencyPhone);
                                printf("Address: %s\n", contact.address);
                                printf("Medical History: %s\n",
history.medicalHistory);
                                printf("Prescription: %s\n",
history.prescription);
                                printf("Allergies: %s\n", history.allergies);
                                printf("Current Medications: %s\n",
history.currentMedications);
                                printf("Preferred Center: %s\n",
center.centerName);
                                printf("Address: %s\n", center.address);
                                printf("City of Preferred Center: %s\n",
center.city);
```

```
                                    printf("State of Preferred Center: %s\n",
center.state);
                                    printf("Zipcode of Preferred Center: %s\n",
center.zipcode);
                                    printf("==================================\n");
                                }
                                fclose(file);
                            }

                               break;
                        case 5:
                            return;
                        default:
                            printf("Invalid choice. Please try again.\n");
                    }
                } while (1);
            }

            break;
        case 4:
              {
                int choice;
                do {
                    printf("\nProfile Management\n");
                    printf("1. Update Information\n");
                    printf("2. Change Appointment\n");
                    printf("3. Update Medical Report\n");
                    printf("4. Back to Main Menu\n");
                    printf("Enter your choice: ");
                    scanf("%d", &choice);
                    switch (choice) {
                        case 1:
                               {
                                printf("\nUpdating information\n");
                                char aadhaarToUpdate[15];
                                printf("Enter Aadhaar number of the profile to update:
");
                                scanf("%s", aadhaarToUpdate);
                                FILE *file = fopen("registration.txt", "r+");
                                if (file == NULL) {
                                    printf("Error opening registration file.\n");
                                    return;
                                }
                                struct PersonalInfo person;
                                struct ContactDetails contact;
                                while (fread(&person, sizeof(struct PersonalInfo), 1,
file) == 1) {
                                    if (strcmp(person.aadhaar, aadhaarToUpdate) == 0)
{
                                        printf("Enter new Name: ");
                                        scanf("%s", person.name);
                                        printf("Enter new Date of Birth: ");
```

```
                                        scanf("%s", person.dateOfBirth);
                                        printf("Enter new Gender: ");
                                        scanf("%s", person.gender);
                                        printf("Enter new Email: ");
                                        scanf("%s", contact.email);
                                        printf("Enter new Phone: ");
                                        scanf("%s", contact.phone);
                                        printf("Enter new Emergency Contact: ");
                                        scanf("%s", contact.emergencyContact);
                                        printf("Enter new Emergency Phone: ");
                                        scanf("%s", contact.emergencyPhone);
                                        printf("Enter new Address: ");
                                        scanf("%s", contact.address);
                                        fseek(file, -sizeof(struct PersonalInfo),
SEEK_CUR);
                                        fwrite(&person, sizeof(struct PersonalInfo),
1, file);
                                        printf("Information updated successfully.\n");
                                        break;
                                    }
                                }
                                fclose(file);
                            }
                            break;
                        case 2:
                            {
                                printf("\nChanging appointment\n");
                                printf("List of available appointments:\n");
                                int totalAppointments = 0;
                                printf("Enter the index of the appointment you want to
change (0-%d): ", totalAppointments - 1);
                                int selectedAppointmentIndex;
                                scanf("%d", &selectedAppointmentIndex);
                                FILE *file = fopen("appointments.txt", "r+");
                                if (file == NULL) {
                                    printf("Error opening appointments file.\n");
                                    return;
                                }
                                struct Appointment appointment;
                                while (fread(&appointment, sizeof(struct Appointment),
1, file) == 1) {
                                    if (selectedAppointmentIndex == 0) {
                                        printf("Enter new Date: ");
                                        scanf("%s", appointment.date);
                                        printf("Enter new Time: ");
                                        scanf("%s", appointment.time);
                                        printf("Enter new Center Name: ");
                                        scanf("%s", appointment.centerName);
                                        fseek(file, -sizeof(struct Appointment),
SEEK_CUR);
                                        fwrite(&appointment, sizeof(struct
Appointment), 1, file);
```

```
                                        printf("Appointment changed successfully.\n");
                                        break;
                                    }
                                    selectedAppointmentIndex--;
                                }
                                fclose(file);
                            }
                            break;
                        case 3:{
                                printf("\nUpdating medical report\n");
                                char aadhaarToUpdate[15];
                                printf("Enter Aadhaar number of the profile to update:
");
                                scanf("%s", aadhaarToUpdate);
                                FILE *file = fopen("registration.txt", "r+");
                                if (file == NULL) {
                                    printf("Error opening registration file.\n");
                                    return;
                                }
                                struct PersonalInfo person;
                                struct MedicalHistory history;
                                while (fread(&person, sizeof(struct PersonalInfo), 1,
file) == 1) {
                                    if (strcmp(person.aadhaar, aadhaarToUpdate) == 0)
{
                                        printf("Enter updated Medical History: ");
                                        scanf(" %[^\n]", history.medicalHistory);
                                        printf("Enter updated Prescription: ");
                                        scanf(" %[^\n]", history.prescription);
                                        printf("Enter updated Allergies: ");
                                        scanf(" %[^\n]", history.allergies);
                                        printf("Enter updated Current Medications: ");
                                        scanf(" %[^\n]", history.currentMedications);
                                        fseek(file, -sizeof(struct PersonalInfo),
SEEK_CUR);
                                        fwrite(&person, sizeof(struct PersonalInfo),
1, file);
                                        printf("Medical report updated
successfully.\n");
                                        break;
                                    }
                                }
                                fclose(file);
                            }
                            break;
                        case 4:
                            return;
                        default:
                            printf("Invalid choice. Please try again.\n");
                    }
                } while (1);
            }
```

```
                break;
            case 5:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (1);
}
void adminMenu() {
    int choice;
    do {
        printf("\nAdmin Menu\n");
        printf("1. Manage Vaccine Information\n");
        printf("2. Manage Appointments\n");
        printf("3. Add Vaccine Stock\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                   {
                    int choice;
                    do {
                        printf("\nVaccine Information Management\n");
                        printf("1. Add Vaccine Information\n");
                        printf("2. Edit Vaccine Information\n");
                        printf("3. Delete Vaccine Information\n");
                        printf("4. View Vaccine Information\n");
                        printf("5. Back to Main Menu\n");
                        printf("Enter your choice: ");
                        scanf("%d", &choice);
                        switch (choice) {
                            case 1:
                                   {
                                    struct Vaccine newVaccine;
                                    printf("\nAdding new vaccine information\n");
                                    printf("Enter Vaccine Name: ");
                                    scanf("%s", newVaccine.name);
                                    printf("Enter Manufacturer: ");
                                    scanf("%s", newVaccine.manufacturer);
                                    printf("Enter Type: ");
                                    scanf("%s", newVaccine.type);
                                    printf("Enter Number of Doses Required: ");
                                    scanf("%d", &newVaccine.dosesRequired);
                                    FILE *file = fopen("vaccine_info.txt", "a");
                                    if (file == NULL) {
                                        printf("Error opening file for writing.\n");
                                        return;
                                    }
                                    fprintf(file, "Vaccine Name: %s\n", newVaccine.name);
```

```
                                    fprintf(file, "Manufacturer: %s\n",
newVaccine.manufacturer);
                                    fprintf(file, "Type: %s\n", newVaccine.type);
                                    fprintf(file, "Doses Required: %d\n",
newVaccine.dosesRequired);
                                    fprintf(file, "--------------------\n");
                                    fclose(file);
                                    printf("New vaccine information added
successfully.\n");
                                }
                                break;
                            case 2:

                                {
                                    printf("\nEditing vaccine information\n");
                                    int selectedVaccineIndex;
                                    FILE *file = fopen("vaccine_info.txt", "r+");
                                    if (file == NULL) {
                                        printf("Error opening file for reading and
writing.\n");
                                        return;
                                    }
                                    long int offset = selectedVaccineIndex * sizeof(struct
Vaccine);
                                    fseek(file, offset, SEEK_SET);
                                    struct Vaccine editedVaccine;
                                    fread(&editedVaccine, sizeof(struct Vaccine), 1,
file);
                                    printf("Enter new Vaccine Name: ");
                                    scanf("%s", editedVaccine.name);
                                    printf("Enter new Manufacturer: ");
                                    scanf("%s", editedVaccine.manufacturer);
                                    printf("Enter new Type: ");
                                    scanf("%s", editedVaccine.type);
                                    printf("Enter new Number of Doses Required: ");
                                    scanf("%d", &editedVaccine.dosesRequired);
                                    fseek(file, offset, SEEK_SET);
                                    fwrite(&editedVaccine, sizeof(struct Vaccine), 1,
file);
                                    fclose(file);
                                    printf("Vaccine information edited successfully.\n");
                                }
                                break;
                            case 3:
                                {
                                    printf("\nDeleting vaccine information\n");
                                    int selectedVaccineIndex;
                                    FILE *originalFile = fopen("vaccine_info.txt", "r");
                                    if (originalFile == NULL) {
                                        printf("Error opening file for reading.\n");
                                        return;
                                    }
```

```
                                    FILE *tempFile = fopen("temp_vaccine_info.txt", "w");
                                    if (tempFile == NULL) {
                                        printf("Error opening temporary file for
writing.\n");
                                        fclose(originalFile);
                                        return;
                                    }
                                    struct Vaccine vaccine;
                                    int currentIndex = 0;
                                while (fread(&vaccine, sizeof(struct Vaccine), 1,
originalFile) == 1) {
                                    if (currentIndex != selectedVaccineIndex) {
                                        fwrite(&vaccine, sizeof(struct Vaccine), 1,
tempFile);
                                    } currentIndex++;
                                }
                                    fclose(originalFile);
                                    fclose(tempFile);
                                    remove("vaccine_info.txt");
                                    rename("temp_vaccine_info.txt", "vaccine_info.txt");
                                    printf("Vaccine information deleted successfully.\n");
                                }
                                break;
                            case 4:
                                {
                                    printf("\nViewing all vaccine information\n");
                                    FILE *file = fopen("vaccine_info.txt", "r");
                                    if (file == NULL) {
                                        printf("Error opening vaccine information
file.\n");
                                        return;
                                    }
                                    struct Vaccine vaccine;
                                    char line[200];
                                    printf("List of registered vaccines:\n");
                                    printf("============================\n");
                                     while (fgets(line, sizeof(line), file)) {
                                        printf("%s", line);
                                    }
                                    while (fread(&vaccine, sizeof(struct Vaccine), 1,
file) == 1) {
                                        printf("Vaccine Name: %s\n", vaccine.name);
                                        printf("Manufacturer: %s\n",
vaccine.manufacturer);
                                        printf("Type: %s\n", vaccine.type);
                                        printf("Doses Required: %d\n",
vaccine.dosesRequired);
                                        printf("----------------------------\n");
                                    }
                                    fclose(file);
                                }
                                break;
```

```
                        case 5:
                            return;
                        default:
                            printf("Invalid choice. Please try again.\n");
                    }
                } while (1);
            }
            break;
        case 2:{
                int choice;
                do {
                    printf("\nAppointment Management\n");
                    printf("1. View Appointments\n");
                    printf("2. Reschedule Appointment\n");
                    printf("3. Cancel Appointment\n");
                    printf("4. Send Reminder\n");
                    printf("5. Back to Main Menu\n");
                    printf("Enter your choice: ");
                    scanf("%d", &choice);
                    switch (choice) {
                        case 1:
                            {
                                printf("\nViewing appointments\n");
                                FILE *file = fopen("appointments.txt", "r");
                                if (file == NULL) {
                                    printf("Error opening appointments file.\n");
                                    return;
                                }
                                struct Appointment appointment;
                                printf("List of scheduled appointments:\n");
                                printf("==============================\n");
                                while (fread(&appointment, sizeof(struct
Appointment), 1, file) == 1) {
                                    printf("Aadhaar Number: %s\n",
appointment.aadhaar);
                                    printf("Date: %s\n", appointment.date);
                                    printf("Time: %s\n", appointment.time);
                                    printf("Center Name: %s\n",
appointment.centerName);
                                    printf("--------------------------\n");
                                }
                                fclose(file);
                            }
                        break;
                        case 2:
                            {
                               printf("\nRescheduling an appointment\n");
                               int selectedAppointmentIndex;
                               FILE *file = fopen("appointments.txt", "r+");
                               if (file == NULL) {
                                   printf("Error opening appointments file.\n");
                                   return;
```

```
                                    }
                                    long int offset = selectedAppointmentIndex *
sizeof(struct Appointment);
                                    fseek(file, offset, SEEK_SET);
                                    struct Appointment rescheduledAppointment;
                                    fread(&rescheduledAppointment, sizeof(struct
Appointment), 1, file);
                                    printf("Enter new Date: ");
                                    scanf("%s", rescheduledAppointment.date);
                                    printf("Enter new Time: ");
                                    scanf("%s", rescheduledAppointment.time);
                                    fseek(file, offset, SEEK_SET);
                                    fwrite(&rescheduledAppointment, sizeof(struct
Appointment), 1, file);
                                    fclose(file);
                                    printf("Appointment rescheduled successfully.\n");
                                }
                                break;
                            case 3:
                                {
                                    printf("\nCanceling an appointment\n");
                                    int selectedAppointmentIndex;
                                    FILE *originalFile = fopen("appointments.txt", "r");
                                    if (originalFile == NULL) {
                                        printf("Error opening appointments file.\n");
                                        return;
                                    }
                                    FILE *tempFile = fopen("temp_appointments.txt", "w");
                                    if (tempFile == NULL) {
                                        printf("Error opening temporary file for
writing.\n");
                                        fclose(originalFile);
                                        return;
                                    }
                                    struct Appointment appointment;
                                    int currentIndex = 0;
                                    while (fread(&appointment, sizeof(struct Appointment),
1, originalFile) == 1) {
                                        if (currentIndex != selectedAppointmentIndex) {
                                            fwrite(&appointment, sizeof(struct
Appointment), 1, tempFile);
                                        }
                                        currentIndex++;
                                    }
                                    fclose(originalFile);
                                    fclose(tempFile);
                                    remove("appointments.txt");
                                    rename("temp_appointments.txt", "appointments.txt");
                                    printf("Appointment canceled successfully.\n");
                                }
                                break;
                            case 4:
```

```
                                {
                                 printf("\nSending a reminder\n");
                                 int selectedAppointmentIndex;
                                 FILE *file = fopen("appointments.txt", "r");
                                 if (file == NULL) {
                                     printf("Error opening appointments file.\n");
                                     return;
                                 }
                                 struct Appointment appointment;
                                 int currentIndex = 0;
                                 while (fread(&appointment, sizeof(struct Appointment),
1, file) == 1) {

                                     if (currentIndex == selectedAppointmentIndex) {
                                         printf("Sending a reminder for the following

appointment:\n");

                                         printf("Aadhaar Number: %s\n",

appointment.aadhaar);

                                         printf("Date: %s\n", appointment.date);
                                         printf("Time: %s\n", appointment.time);
                                         printf("Center Name: %s\n",

appointment.centerName);

                                         printf("---------------------------\n");
                                         printf("Reminder sent.\n");
                                         break;
                                     }
                                     currentIndex++;
                                 }
                                 fclose(file);
                             }
                             break;
                         case 5:
                             return;
                         default:
                             printf("Invalid choice. Please try again.\n");
                     }
                 } while (1);
             }

              break;
         case 3: {
                 printf("Adding new stock...\n");

                 FILE *file = fopen("stock.txt", "a");
                 if (file == NULL) {
                     printf("Error opening stock file.\n");
                     return;
                 }
                 struct VaccineStock stock;
                 printf("Enter vaccine name: ");
                 scanf("%s", stock.vaccineName);
                 printf("Enter quantity: ");
                 scanf("%d", &stock.quantity);
```

```
                    fwrite(&stock, sizeof(struct VaccineStock), 1, file);
                    fclose(file);
                    printf("Stock added successfully.\n");
                }

                break;
            case 4:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (1);
}
```

13.2 Code Snippets

1.Main Menu and Role Selection:

```
int main() {
   int role;
   printf("Welcome to the Vaccine Registration System!\n");
   printf("Please select your role:\n");
   printf("1. Customer\n");
   printf("2. Admin\n");
   printf("Enter your choice: ");
   scanf("%d", &role);
   if (role == 1) {
      customerMenu();
   } else if (role == 2) {
      adminMenu();
   } else {
      printf("Invalid role selection.\n");
   }
   return 0;
}
```

2. Customer Menu:

```
void customerMenu() {
   int choice;
   do {
      printf("\nCustomer Menu\n");
      // Display options for the customer
      printf("1. Registration Form\n");
      printf("2. Appointment Scheduling\n");
      printf("3. Dashboard\n");
      printf("4. Profile Management\n");
      printf("5. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);
      switch (choice) {
         // Implement functions for each menu option
```

```
        case 1:
          registrationForm();
          break;
        case 2:
          appointmentScheduling();
          break;
        case 3:
          dashboard();
          break;
        case 4:
          profileManagement();
          break;
        case 5:
          printf("Exiting...\n");
          exit(0);
        default:
          printf("Invalid choice. Please try again.\n");
      }
   } while (1);
}
```

3.Admin Menu:

```
void adminMenu()
{
          int choice;
          do
          {
                    printf("\nAdmin Menu\n");
                    printf("1. Manage Vaccine Information\n");
                    printf("2. Manage Appointments\n");
                    printf("3. Add Vaccine Stock\n");
                    printf("4. Exit\n");
                    printf("Enter your choice: ");
                    scanf("%d", &choice);
                     switch (choice)
                     {
                              case 1: vaccineInformation();
                              break;
                              case 2: manageAppointments();
                               break;
                              case 3: addStock();
                              break;
                              case 4: printf("Exiting...\n");
                               exit(0);
                              default: printf("Invalid choice. Please try again.\n");
                    }
          } while (1);
}
```