AARYA SANGEETH

Object Oriented Programming

(ENGR-UH 2510) Spring 2025 : FINAL PROJECT

VACCINE INOCULATION SYSTEM

1.1 Introduction

The vaccine inoculation system is designed and tailored in order to streamline the process of managing patient records, scheduling appointments, and tracking vaccine stocks in a clinic. The system is built using Object-Oriented Programming (OOP) principles so as to ensure modularity, scalability, and maintainability. It primarily addresses real-world challenges such as maintaining patient registration, tracking vaccine administration, and vaccine inventory management, making it a practical tool that can be further scaled and developed for healthcare providers.
The system is divided into four main components: **Patient Management, Admin Module, Staff Module, and Vaccine Management**, each implemented using OOP concepts like inheritance, polymorphism, encapsulation, and class friendship.

1.2 Approach

The system is designed to handle the complexities and tasks of a real-world vaccination clinic, focusing on efficiency, scalability, and user-friendliness. The approach is divided into the following key areas:

1. Patient Management:
- The system allows patients to register themselves into the system via the Admin Module, replicating how patients need to register and file in healthcare centres through the center's administration.
- Once registered, patients can opt to schedule appointments, view their vaccination status, and also check their appointment details, if any. The system uses a static map (**`patient_records`**) to store patient objects during program execution, enabling quick lookups and updates.

2. Admin Module:
- The admin module provides functionalities for clinic administrators, such as viewing all patient records, scheduling appointments, and managing vaccine stocks.
- Admins can also place orders for additional vaccine supplies if the stock falls below a predefined (currently set to 1000 doses) threshold.

3. Staff Module:
- The staff module enables clinic staff to update patient vaccination records and view current vaccine stocks.
- Staff members can administer vaccine doses to patients, updating their vaccination status and reducing the vaccine stock accordingly with each administered dose.

4.Vaccine Management:
- The system tracks vaccine types, stocks, and required doses using a hierarchical class structure.
- The base Vaccine class is extended by specific vaccine types (e.g., Pfizer, Moderna) via inheritance, which allows the system to display vaccine specific information.

5. File Handling:
- The system uses CSV files to store patient records, appointments, and vaccine stocks. This ensures data persistence and allows for easy integration with external systems.

1.3 Solution

This system has been thoughtfully designed to simulate a real-world vaccine inoculation clinic. Below is a detailed explanation of my solution, including how the different classes interact and an overview of the overall program flow.

Patient Class:

- It stores attributes such as **`name, id, vacc_type, doses_done`**, and **`status`** for clinic patients.
- The Admin class can access private members of the Patient class (e.g., **`app_date, timing`**) through class friendship. This allows admins to schedule appointments for patients.
- The Staff class can update a patient's vaccination status and dose count using the **`Patient::UpdateDoses()`**method.
- The Patient class stores the type of vaccine a patient has chosen (**`vacc_type`**), which is used by the Staff class to update vaccine stocks after registering a vaccination.

Admin Class:

- The Admin class handles administrative tasks such as viewing patient records, scheduling appointments, and managing vaccine stocks.
- The **`Admin::appointments()`** function schedules appointments for patients by calling the **`Patient::setAppointment()`** method, which is made accessible via class friendship.
- The Admin class can check vaccine stocks and place orders for additional supplies using the **`Vaccine::getCurrentStock()`** method.

Staff Class:

- The Staff class is responsible for updating patient vaccination records and viewing vaccine stocks.
- he **`Staff::update_dose()`** function increments a patient's dose count and updates their vaccination status using the **`Patient::UpdateDoses()`** method.
- The Staff class reduces the vaccine stock when a dose is administered, using the **`Vaccine::stock`** map.

Vaccine Class:

- The Vaccine class serves as the base class for specific vaccine types (e.g., Pfizer, Moderna). It tracks vaccine stocks and required doses.
- The **`Vaccine::createVaccine()`** function dynamically creates vaccine objects based on the patient's chosen vaccine type.
- The Staff class, via class friendship updates the vaccine stock when a dose is administered, using the **`Vaccine::stoc`**k map.

1.4 Program Flow

Patient Registration:

1. When a patient registers, they provide their name, Emirates ID, and preferred vaccine type.
2. A new Patient object is created and added to the patient_records map.

Appointment Scheduling:

1. An admin schedules an appointment for a patient by entering their Emirates ID, vaccine type, appointment date, and time.
2. The system checks if the patient already exists in the **`patient_records`** map. If not, a new Patient object is created.

3. The appointment details are written to the **`Appointments.csv`** file using the **`Patient::record_appointment()`** function.

Vaccine Administration:

1. A staff member administers a vaccine dose to a patient by entering their Emirates ID.
2. The system retrieves the patient's record from the **`patient_records`** map and increments their dose count using the **`Patient::UpdateDoses()`** method.
3. If the patient does not have a record, the system prompts them to go to admin to register themselves because vaccinations are only registered if the patient is on file.
4. The vaccine stock is updated by decrementing the corresponding entry in the **`Vaccine::stock`** map.

Vaccine Stock Management:

1. Admins and staff can view the current vaccine stock using the **`Vaccine::getCurrentStock()`** method.
2. If the stock of a particular vaccine falls below the predefined threshold, the admin can place an order for additional supplies.

1.5 Program Structure :

1.5.1 Program Flow Diagram

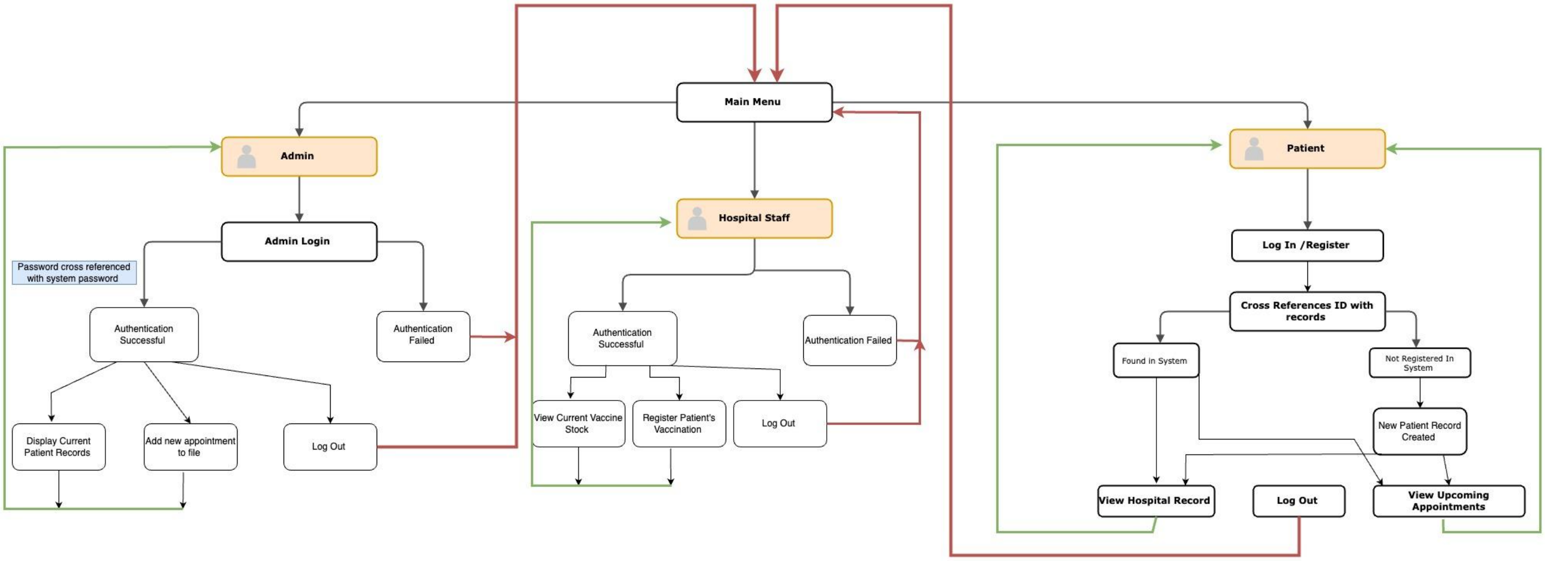

Main Menu
Admin
Admin Login
Password cross referenced with system password
Authentication Successful
Authentication Failed
Display Current Patient Records
Add new appointment to file
Log Out
Hospital Staff
Authentication Successful
Authentication Failed
View Current Vaccine Stock
Register Patient's Vaccination
Log Out
Patient
Log In /Register
Cross References ID with records
Found in System
Not Registered In System
New Patient Record Created
View Hospital Record
Log Out
View Upcoming Appointments

Fig 1.5 (a) Program Flow

1.5.2 UML Diagram

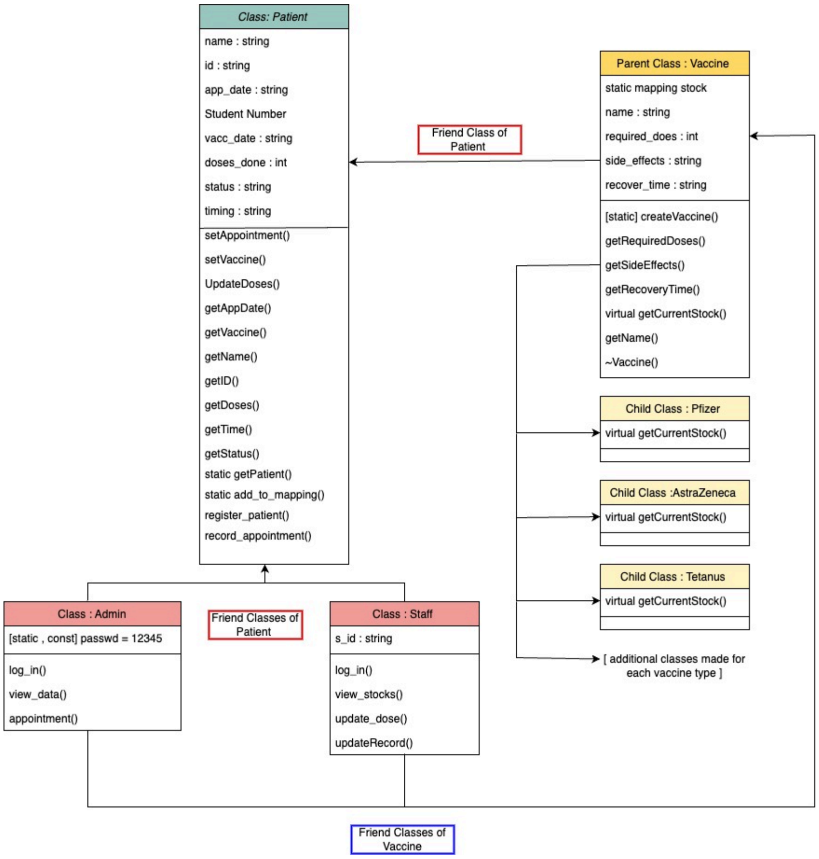
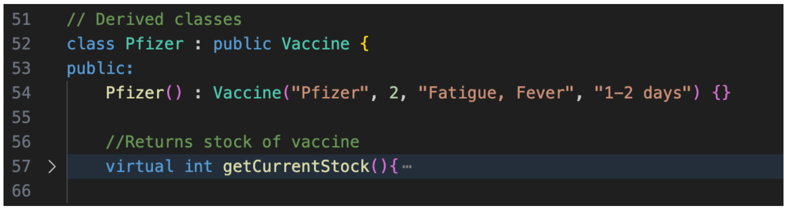


Fig. 1.5 (b) UML Diagram for the Program

1.6 OOP Paradigm Implementation

1. Inheritance :

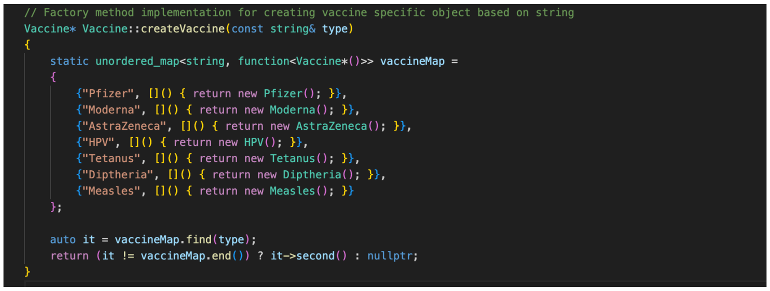


```
// Derived classes
class Pfizer : public Vaccine {
public:
    Pfizer() : Vaccine("Pfizer", 2, "Fatigue, Fever", "1–2 days") {}

    //Returns stock of vaccine
    virtual int getCurrentStock(){⋯
```

Fig 1.6 (1) Parent and Child Classes

- The Pfizer class inherits from the base Vaccine class. This demonstrates inheritance, as Pfizer reuses the functionality of the Vaccine class, like name, required_doses, side_effects, and recovery_time while adding its own specific details.

2. Runtime Polymorphism :

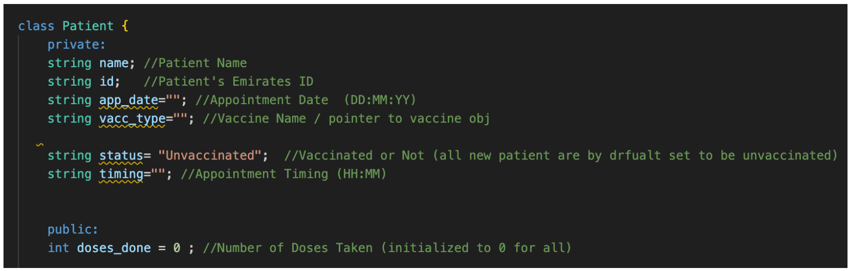


```
// Factory method implementation for creating vaccine specific object based on string
Vaccine* Vaccine::createVaccine(const string& type)
{
    static unordered_map<string, function<Vaccine*()>> vaccineMap =
    {
        {"Pfizer", []() { return new Pfizer(); }},
        {"Moderna", []() { return new Moderna(); }},
        {"AstraZeneca", []() { return new AstraZeneca(); }},
        {"HPV", []() { return new HPV(); }},
        {"Tetanus", []() { return new Tetanus(); }},
        {"Diptheria", []() { return new Diptheria(); }},
        {"Measles", []() { return new Measles(); }}
    };

    auto it = vaccineMap.find(type);
    return (it != vaccineMap.end()) ? it->second() : nullptr;
}
```

Fig 1.6 (2) Vaccine Factory Function

- The createVaccine function uses polymorphism to dynamically create objects of specific vaccine types (e.g., Pfizer, Moderna) based on the input string. This allows the system to handle different vaccine types flexibly without hardcoding object creation.

3. Encapsulation :

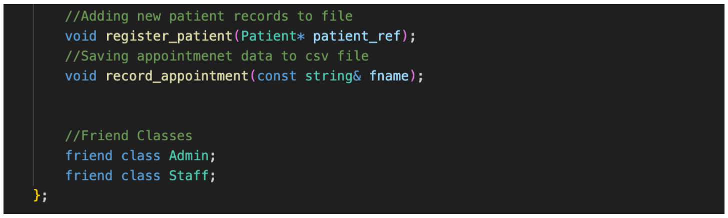


```
class Patient {
    private:
    string name; //Patient Name
    string id;   //Patient's Emirates ID
    string app_date=""; //Appointment Date  (DD:MM:YY)
    string vacc_type=""; //Vaccine Name / pointer to vaccine obj

    string status= "Unvaccinated";  //Vaccinated or Not (all new patient are by drfualt set to be unvaccinated)
    string timing=""; //Appointment Timing (HH:MM)


    public:
    int doses_done = 0 ; //Number of Doses Taken (initialized to 0 for all)
```

Fig 1.6 (3) Attribute Encapsulation in Class Patient

- The Patient class encapsulates private data members (e.g., name, id, app_date) and provides public methods (e.g., getName, setAppointment) to access and modify them.This demonstrates encapsulation, ensuring data integrity and controlled access to sensitive information.

4. Class Friendship :

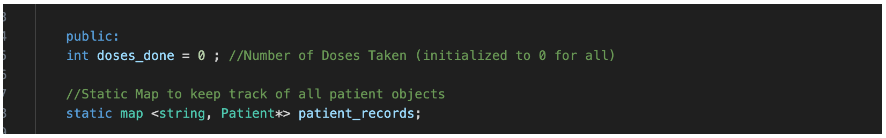


```
    //Adding new patient records to file
    void register_patient(Patient* patient_ref);
    //Saving appointmenet data to csv file
    void record_appointment(const string& fname);


    //Friend Classes
    friend class Admin;
    friend class Staff;
};
```

Fig 1.6 (4) Friend Classes of Class Patient

- The Admin and Staff classes are declared as friends of the Patient class, allowing them to access private members of Patient. This demonstrates class friendship, enabling secure and controlled access to sensitive patient data.

5. Static Class Members :



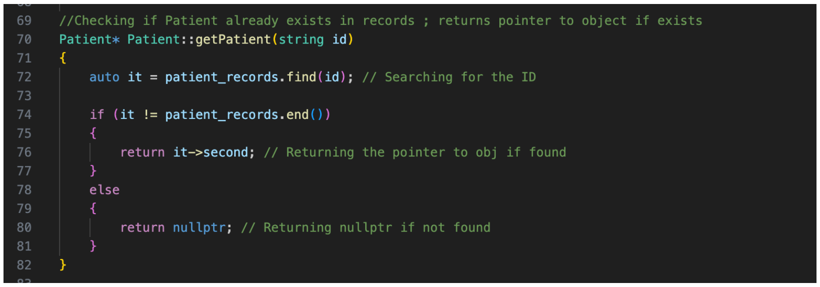
```
    public:
    int doses_done = 0 ; //Number of Doses Taken (initialized to 0 for all)

    //Static Map to keep track of all patient objects
    static map <string, Patient*> patient_records;
```

Fig 1.6 (5) Static Class Members of Class Patient

- The patient_records map is declared as a static member of the Patient class, allowing it to be shared across all instances of the class. This ensures that patient records are globally accessible and consistent.

1.7 Optimization Implementation

1. Static Mapping for Patient Information Lookups :

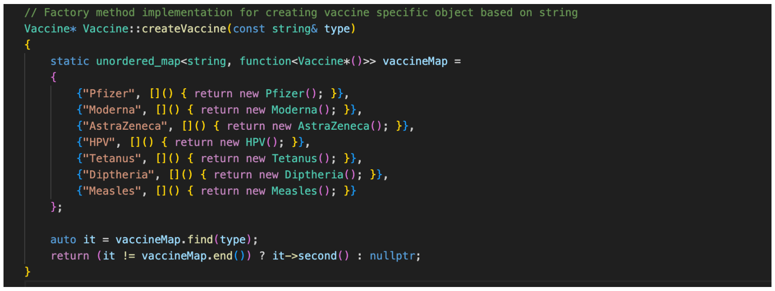


```cpp
//Checking if Patient already exists in records ; returns pointer to object if exists
Patient* Patient::getPatient(string id)
{
    auto it = patient_records.find(id); // Searching for the ID

    if (it != patient_records.end())
    {
        return it->second; // Returning the pointer to obj if found
    }
    else
    {
        return nullptr; // Returning nullptr if not found
    }
}
```

Fig 1.7 (1) Static Mapping

- The patient_records mapping enables O(1) lookups, insertions, and deletions using Emirates ID as the key, avoiding slow iterations and file parisng. It stores pointers to patient objects, reducing memory overhead and ensuring efficiency even with large datasets.

2. Polymorphism for Vaccine Management :

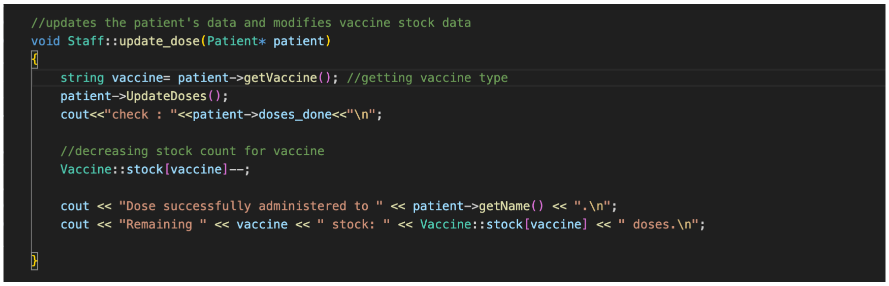


```cpp
// Factory method implementation for creating vaccine specific object based on string
Vaccine* Vaccine::createVaccine(const string& type)
{
    static unordered_map<string, function<Vaccine*()>> vaccineMap =
    {
        {"Pfizer", []() { return new Pfizer(); }},
        {"Moderna", []() { return new Moderna(); }},
        {"AstraZeneca", []() { return new AstraZeneca(); }},
        {"HPV", []() { return new HPV(); }},
        {"Tetanus", []() { return new Tetanus(); }},
        {"Diptheria", []() { return new Diptheria(); }},
        {"Measles", []() { return new Measles(); }}
    };

    auto it = vaccineMap.find(type);
    return (it != vaccineMap.end()) ? it->second() : nullptr;
}
```

Fig 1.7 (3) Function Implementing Polymorphism in Class Vaccine

- The createVaccine function uses a factory map to create vaccine objects dynamically, allowing easy extension without modifying existing code. It creates only the needed vaccine, optimizing memory usage.

3. Avoiding Redundant Computations :

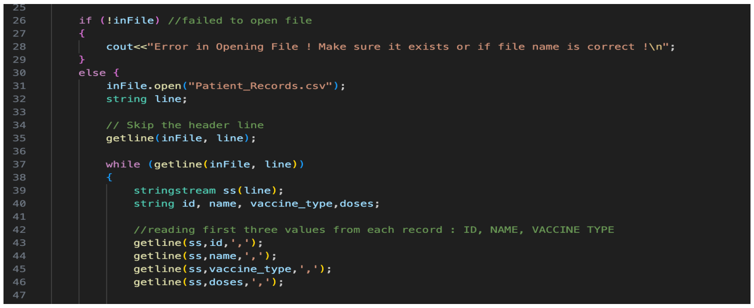


```cpp
//updates the patient's data and modifies vaccine stock data
void Staff::update_dose(Patient* patient)
{
    string vaccine= patient->getVaccine(); //getting vaccine type
    patient->UpdateDoses();
    cout<<"check : "<<patient->doses_done<<"\n";

    //decreasing stock count for vaccine
    Vaccine::stock[vaccine]--;

    cout << "Dose successfully administered to " << patient->getName() << ".\n";
    cout << "Remaining " << vaccine << " stock: " << Vaccine::stock[vaccine] << " doses.\n";

}
```

Fig 1.7 (4) Multiple Updates in Single Function

- The update_dose function updates the patient's dose count and vaccine stock in one step, reducing function calls and computations. Using pointers minimizes memory usage by avoiding temporary variables.

4. Efficient File Handling :



```
    if (!inFile) //failed to open file
    {
        cout<<"Error in Opening File ! Make sure it exists or if file name is correct !\n";
    }
    else {
        inFile.open("Patient_Records.csv");
        string line;

        // Skip the header line
        getline(inFile, line);

        while (getline(inFile, line))
        {
            stringstream ss(line);
            string id, name, vaccine_type,doses;

            //reading first three values from each record : ID, NAME, VACCINE TYPE
            getline(ss,id,',');
            getline(ss,name,',');
            getline(ss,vaccine_type,',');
            getline(ss,doses,',');

```

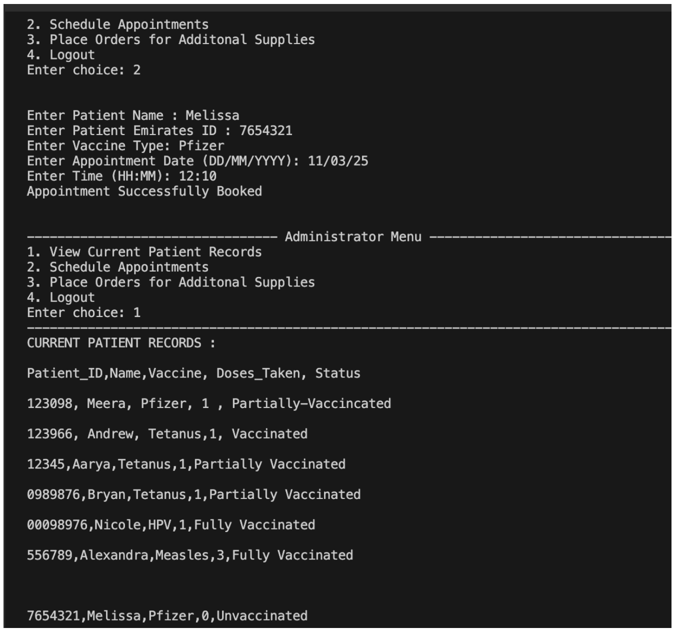
Fig 1.7 (5) File Handling Function in Class Patient

- The function reads the file line by line with \`getline\`, reducing memory usage for large files. It uses stringstream to parse fields efficiently and processes data in a single pass, minimizing I/O operations for better performance.

1.8 Output to Console :

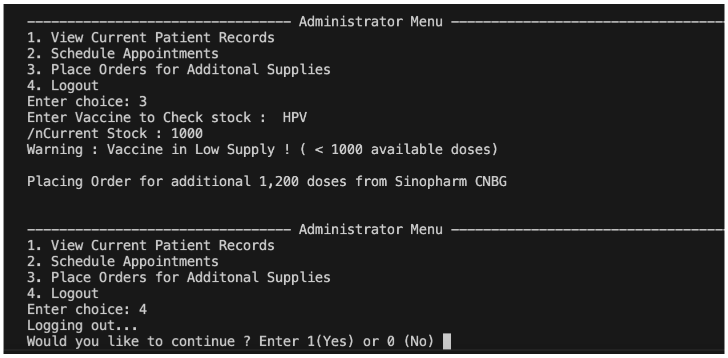
1. **User Role : Clinic Admin**

a. Registering new patient and displaying Current Patient Records



```
2. Schedule Appointments
3. Place Orders for Additonal Supplies
4. Logout
Enter choice: 2


Enter Patient Name : Melissa
Enter Patient Emirates ID : 7654321
Enter Vaccine Type: Pfizer
Enter Appointment Date (DD/MM/YYYY): 11/03/25
Enter Time (HH:MM): 12:10
Appointment Successfully Booked


---------------------------------- Administrator Menu ----------------------------------
1. View Current Patient Records
2. Schedule Appointments
3. Place Orders for Additonal Supplies
4. Logout
Enter choice: 1
----------------------------------------------------------------------------------------
CURRENT PATIENT RECORDS :

Patient_ID,Name,Vaccine, Doses_Taken, Status

123098, Meera, Pfizer, 1 , Partially-Vaccincated

123966, Andrew, Tetanus,1, Vaccinated

12345,Aarya,Tetanus,1,Partially Vaccinated

0989876,Bryan,Tetanus,1,Partially Vaccinated

00098976,Nicole,HPV,1,Fully Vaccinated

556789,Alexandra,Measles,3,Fully Vaccinated



7654321,Melissa,Pfizer,0,Unvaccinated
```

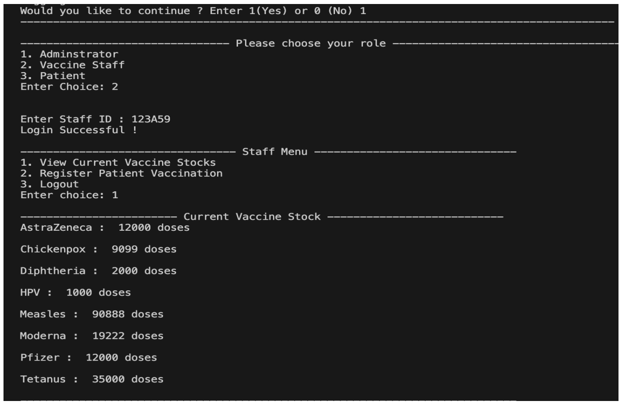
b. Checking vaccine stock



```
----------------------------------- Administrator Menu ----------------------------------
1. View Current Patient Records
2. Schedule Appointments
3. Place Orders for Additonal Supplies
4. Logout
Enter choice: 3
Enter Vaccine to Check stock :  HPV
/nCurrent Stock : 1000
Warning : Vaccine in Low Supply ! ( < 1000 available doses)

Placing Order for additional 1,200 doses from Sinopharm CNBG


----------------------------------- Administrator Menu ----------------------------------
1. View Current Patient Records
2. Schedule Appointments
3. Place Orders for Additonal Supplies
4. Logout
Enter choice: 4
Logging out...
Would you like to continue ? Enter 1(Yes) or 0 (No)
```

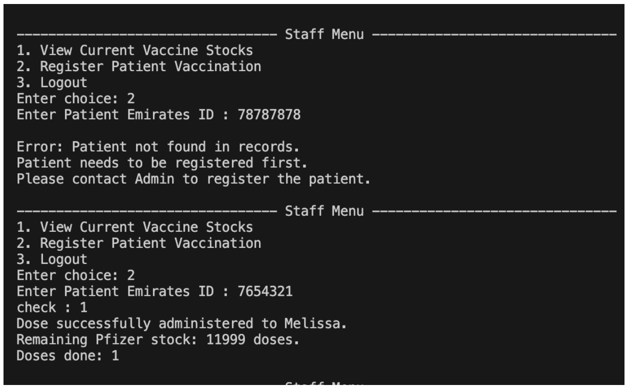
2. **User Role : Clinic Staff**

a. Viewing current Vaccine Stock

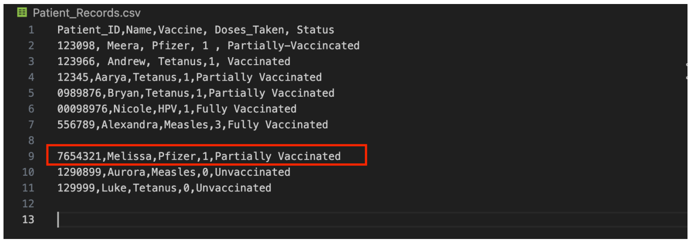


```
Would you like to continue ? Enter 1(Yes) or 0 (No) 1
------------------------------------------------------------------------------------------

--------------------------------- Please choose your role ---------------------------------
1. Adminstrator
2. Vaccine Staff
3. Patient
Enter Choice: 2


Enter Staff ID : 123A59
Login Successful !

---------------------------------- Staff Menu ----------------------------------
1. View Current Vaccine Stocks
2. Register Patient Vaccination
3. Logout
Enter choice: 1

------------------------ Current Vaccine Stock ---------------------------
AstraZeneca :  12000 doses

Chickenpox :  9099 doses

Diphtheria :  2000 doses

HPV :  1000 doses

Measles :  90888 doses

Moderna :  19222 doses

Pfizer :  12000 doses

Tetanus :  35000 doses
```

b. Registering vaccine to patient



```
---------------------------------- Staff Menu ----------------------------------
1. View Current Vaccine Stocks
2. Register Patient Vaccination
3. Logout
Enter choice: 2
Enter Patient Emirates ID : 78787878

Error: Patient not found in records.
Patient needs to be registered first.
Please contact Admin to register the patient.

---------------------------------- Staff Menu ----------------------------------
1. View Current Vaccine Stocks
2. Register Patient Vaccination
3. Logout
Enter choice: 2
Enter Patient Emirates ID : 7654321
check : 1
Dose successfully administered to Melissa.
Remaining Pfizer stock: 11999 doses.
Doses done: 1
```



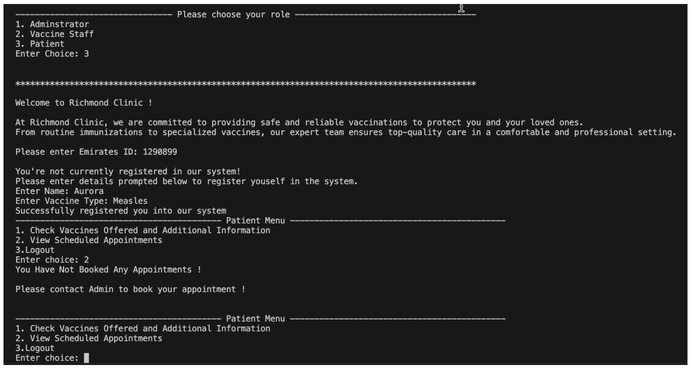
Patient_Records.csv

```
Patient_ID,Name,Vaccine, Doses_Taken, Status
123098, Meera, Pfizer, 1 , Partially-Vaccincated
123966, Andrew, Tetanus,1, Vaccinated
12345,Aarya,Tetanus,1,Partially Vaccinated
0989876,Bryan,Tetanus,1,Partially Vaccinated
00098976,Nicole,HPV,1,Fully Vaccinated
556789,Alexandra,Measles,3,Fully Vaccinated

7654321,Melissa,Pfizer,1,Partially Vaccinated
1290899,Aurora,Measles,0,Unvaccinated
129999,Luke,Tetanus,0,Unvaccinated


```

After registering the vaccine to Patient, her record has been updated to reflect the change.

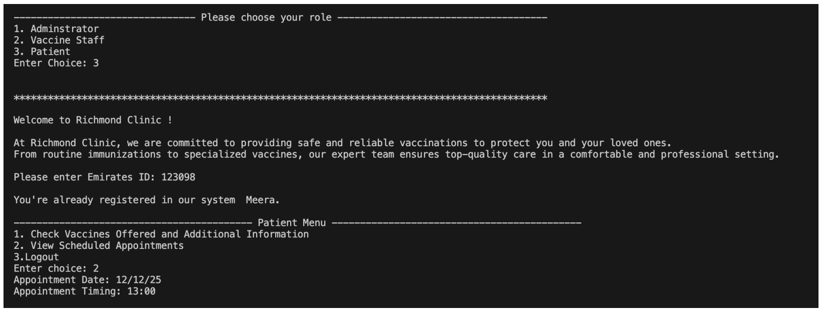
3. User Role : Patient

a. Patient is unregistered



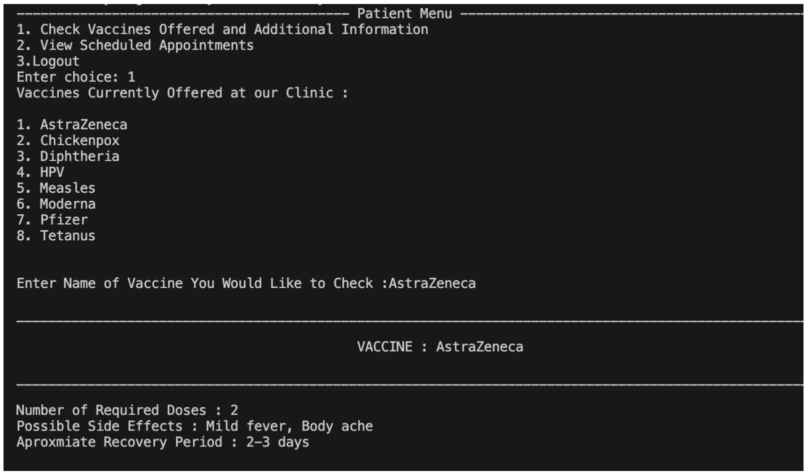
```
-------------------------------- Please choose your role -------------------------------------
1. Adminstrator
2. Vaccine Staff
3. Patient
Enter Choice: 3


**********************************************************************************************

Welcome to Richmond Clinic !

At Richmond Clinic, we are committed to providing safe and reliable vaccinations to protect you and your loved ones.
From routine immunizations to specialized vaccines, our expert team ensures top-quality care in a comfortable and professional setting.

Please enter Emirates ID: 1290899

You're not currently registered in our system!
Please enter details prompted below to register youself in the system.
Enter Name: Aurora
Enter Vaccine Type: Measles
Successfully registered you into our system
------------------------------------------ Patient Menu --------------------------------------------
1. Check Vaccines Offered and Additional Information
2. View Scheduled Appointments
3.Logout
Enter choice: 2
You Have Not Booked Any Appointments !

Please contact Admin to book your appointment !


------------------------------------------ Patient Menu --------------------------------------------
1. Check Vaccines Offered and Additional Information
2. View Scheduled Appointments
3.Logout
Enter choice: 
```

b. Patient is already registered in system and has an appointment booked



```
-------------------------------- Please choose your role -------------------------------------
1. Adminstrator
2. Vaccine Staff
3. Patient
Enter Choice: 3


**********************************************************************************************

Welcome to Richmond Clinic !

At Richmond Clinic, we are committed to providing safe and reliable vaccinations to protect you and your loved ones.
From routine immunizations to specialized vaccines, our expert team ensures top-quality care in a comfortable and professional setting.

Please enter Emirates ID: 123098

You're already registered in our system  Meera.

------------------------------------------ Patient Menu --------------------------------------------
1. Check Vaccines Offered and Additional Information
2. View Scheduled Appointments
3.Logout
Enter choice: 2
Appointment Date: 12/12/25
Appointment Timing: 13:00
```

c. Viewing Vaccine Specific Information



```
------------------------------------------ Patient Menu --------------------------------------------
1. Check Vaccines Offered and Additional Information
2. View Scheduled Appointments
3.Logout
Enter choice: 1
Vaccines Currently Offered at our Clinic :

1. AstraZeneca
2. Chickenpox
3. Diphtheria
4. HPV
5. Measles
6. Moderna
7. Pfizer
8. Tetanus


Enter Name of Vaccine You Would Like to Check :AstraZeneca

----------------------------------------------------------------------------------------------------

                                           VACCINE : AstraZeneca

----------------------------------------------------------------------------------------------------

Number of Required Doses : 2
Possible Side Effects : Mild fever, Body ache
Aproxmiate Recovery Period : 2-3 days
```

References

GeeksforGeeks. (n.d.). C++ File Handling: Reading and Writing Files. Retrieved from https://www.geeksforgeeks.org/file-handling-c-classes/

Stack Overflow. (n.d.). How to Parse a CSV File in C++. Retrieved from https://stackoverflow.com/questions/1120140/how-can-i-read-and-parse-csv-files-in-c

Programiz. (n.d.). Polymorphism in C++. Retrieved from https://www.programiz.com/cpp-programming/polymorphism

World Health Organization (WHO). (2021). Digital Solutions for Vaccine Management: Lessons from COVID-19. Retrieved from https://www.who.int/publications/i/item/9789240032097

Capital Health. (n.d.). Vaccinations. Retrieved from https://capitalhealth.ae/services/vaccinations/

Emirates News Agency (WAM). (2021, March 29). UAE commences COVID-19 vaccine production with 'Hayat-Vax'. Retrieved from https://www.wam.ae/en/article/hszrc1g3-uae-commences-covid-19-vaccine-production-with