Doc and export as PDF. It contains:

- OOA model (objects, attributes, methods, inheritance)
- Class design + rationale
- Code walkthrough (key functions & how they satisfy the rubric)
- Test results (sample console output you already ran)
- LLM usage (exact prompts and how you used ChatGPT)
- Self-check against rubric and short compilation/run instructions

Small Clinic Management System — Project Report

1. Project Overview

This project implements a **Small Clinic Management System** in C++ to demonstrate Object-Oriented Programming (OOP) concepts: classes, objects, encapsulation, inheritance, polymorphism, and basic resource management. The system supports patients (regular + chronic), doctors, appointments, and medical history. It includes a ClinicManager that holds all created objects and demonstrates schedule/complete/cancel workflows and basic tests.

2. Object-Oriented Analysis (OOA)

2.1 Objects (nouns)

1. **Patient** — represents a patient (regular).
2. **ChronicPatient** — specialized patient (inherits from Patient).
3. **Doctor** — medical staff who handles appointments.
4. **Appointment** — a scheduled visit linking a patient and a doctor.
5. **MedicalRecord** (implemented as medicalHistory strings in code) — past visits/treatments.
6. **ClinicManager** — top-level manager that stores lists of patients, doctors, and appointments.

2.2 Attributes (per object)

1. **Patient**: name, id, age, medicalHistory (vector<string>).
2. **ChronicPatient**: inherits Patient + conditionType, lastCheckupDate.
3. **Doctor**: name, id, specialty, assignedAppointments (vector<string>).
4. **Appointment**: date, time, reason, status (scheduled/completed/canceled), pointers to Patient and Doctor.
5. **ClinicManager**: vectors of Patient*, Doctor*, Appointment*.

2.3 Methods (per object)

1. **Patient**: constructor/destructor, scheduleAppointment(date, reason), updateHistory(), displayInfo().
2. **ChronicPatient**: override scheduleAppointment() and displayInfo() to show chronic-specific details.
3. **Doctor**: constructor/destructor, assignAppointment(), updateAppointmentStatus(), displayInfo().
4. **Appointment**: constructor/destructor, cancel(), complete(), display().

5. **ClinicManager**: addPatient(), addDoctor(), addAppointment(), show functions, destructor to free allocated memory.

2.4 Inheritance relationships

1. ChronicPatient : public Patient (single inheritance).
2. Patient is the base for regular and chronic patients.

3. Class Design & Rationale

1. **Encapsulation**: Data members are private/protected. Public methods (displayInfo, getters, setters) control access.
2. **Constructors / Destructors**: Each class has a constructor for initialization and a destructor (prints a message) to demonstrate resource cleanup. ClinicManager destructor deletes all heap objects.
3. **Polymorphism/Override**: Patient::scheduleAppointment is virtual and overridden in ChronicPatient to illustrate behavior differences (chronic patients require frequent checks).
4. **Association**: Appointment holds pointers to Patient & Doctor (shows relationships).
5. **Manager class**: ClinicManager centralizes object lifecycle and display operations; this addresses the instructor's feedback asking for a manager / system-level class.

4. Code Walkthrough (key points)

Patient

- scheduleAppointment() — base behavior prints scheduling; updateHistory() appends strings to medicalHistory.
- displayInfo() prints patient id, name, age, and medical history.

ChronicPatient

- Overrides scheduleAppointment() to print a message indicating the patient requires regular checkups.
- displayInfo() prints chronic condition info as well.

Doctor

- assignAppointment() adds a simple string summary to assignedAppointments.
- updateAppointmentStatus() demonstrates how doctor-side status updates could be recorded.

Appointment

- On construction, prints a creation message and registers a summary with the Doctor.
- cancel() and complete() set status and print confirmation; display() shows date/time/reason/status and linked patient/doctor.

ClinicManager

- Holds vectors of pointers (heap allocated in main()), and its destructor cleans up (prevents memory leaks).

Testing Hooks

- main() creates 3 doctors, 3 patients (2 regular + 1 chronic), adds history, creates appointments, schedules them (calls scheduleAppointment()), then complete() and cancel() for two appointments. Finally prints all patients, doctors, and appointments. An “invalid test” demonstrates the chronic scheduling message (can be extended with real date validation).

5. Testing & Sample Output

How to compile & run

Sample console output (captured)

```
Patient Alice created
Patient Bob created
ChronicPatient Charlie with condition Diabetes created
Doctor Dr. Smith created
Doctor Dr. John created
Doctor Dr. Alice created
Appointment created for Alice with Dr. Dr. Smith
Appointment created for Bob with Dr. Dr. John
Appointment created for Charlie with Dr. Dr. Alice
Patient Alice scheduled appointment on 2025-09-10 for General Checkup
Patient Bob scheduled appointment on 2025-09-12 for Dermatology Consult
ChronicPatient Charlie requires frequent check-ups. Appointment on 2025-09-15 for Diabetes
Follow-up scheduled.
Appointment completed on 2025-09-10 10:00
Appointment canceled on 2025-09-12 14:00

--- All Patients ---
Patient(ID:1, Name:Alice, Age:30)
  Medical History:
   - 2025-07-01: Flu - Paracetamol
Patient(ID:2, Name:Bob, Age:40)
  Medical History:
   - 2025-08-05: Skin allergy - Antihistamine
ChronicPatient(ID:3, Name:Charlie, Condition:Diabetes, LastCheckup:2025-06-01)
  Medical History:
   - 2025-06-15: Diabetes check - Insulin

--- All Doctors ---
Doctor(ID:101, Name:Dr. Smith, Specialty:Cardiology)
  Assigned Appointments:
   - 2025-09-10 10:00 - General Checkup
Doctor(ID:102, Name:Dr. John, Specialty:Dermatology)
```

```
  Assigned Appointments:
   - 2025-09-12 14:00 - Dermatology Consult
Doctor(ID:103, Name:Dr. Alice, Specialty:Endocrinology)
  Assigned Appointments:
   - 2025-09-15 09:00 - Diabetes Follow-up

--- All Appointments ---
Appointment(2025-09-10 10:00, Reason:General Checkup, Status:completed, Patient:Alice,
Doctor:Dr. Smith)
Appointment(2025-09-12 14:00, Reason:Dermatology Consult, Status:canceled, Patient:Bob,
Doctor:Dr. John)
Appointment(2025-09-15 09:00, Reason:Diabetes Follow-up, Status:scheduled,
Patient:Charlie, Doctor:Dr. Alice)

[Invalid Test] ChronicPatient attempting to schedule too-soon appointment:
ChronicPatient Charlie requires frequent check-ups. Appointment on 2025-09-11 for Too soon
follow-up scheduled.
...
ClinicManager destroyed and all objects freed.
```

Notes on tests

The output demonstrates object creation, scheduling, status updates, printed lists, and cleanup (destructors).

The "invalid test" currently shows a message from ChronicPatient::scheduleAppointment — you can enhance this to perform real date comparisons and reject scheduling inside a forbidden window.

6. LLM (ChatGPT) Usage (ethical disclosure)

I used ChatGPT in the following ways (assistance only — final code written and tested by me):

Brainstorming object list and method responsibilities (prompt):

"Suggest objects, attributes and methods for a small clinic management system (patients, doctors, appointments)."

Design review for class relationships and suggestions for ClinicManager.

"How to manage memory and objects in C++ for a small system — examples of a manager class."

Formatting & comments: asked for conciseness in comments and a single-file version for easy submission.

"Give me a single-file C++ project structure for a clinic system with classes Patient, Doctor, Appointment, ChronicPatient, ClinicManager."