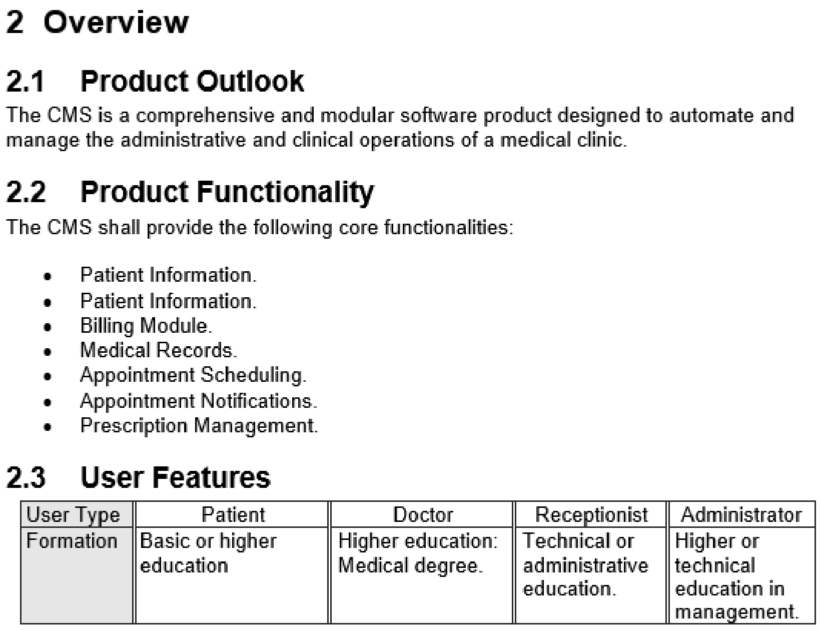
Project Inspection
T4

EVIDENCE:

Ord. Artifact, evaluation/10, observations:

1. Overview



2 Overview

2.1 Product Outlook

The CMS is a comprehensive and modular software product designed to automate and manage the administrative and clinical operations of a medical clinic.

2.2 Product Functionality

The CMS shall provide the following core functionalities:

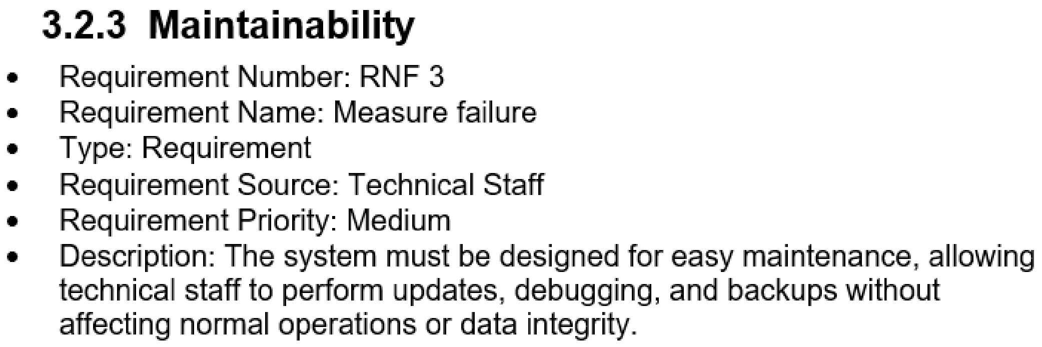
- Patient Information.
- Patient Information.
- Billing Module.
- Medical Records.
- Appointment Scheduling.
- Appointment Notifications.
- Prescription Management.

2.3 User Features

User Type	Patient	Doctor	Receptionist	Administrator
Formation	Basic or higher education	Higher education: Medical degree.	Technical or administrative education.	Higher or technical education in management.

No specific information is available about the client the team will be working with; only general system details are provided. It is essential to clearly identify the client or company for whom the system is being developed.

2. SRS



3.2.3 Maintainability

- Requirement Number: RNF 3
- Requirement Name: Measure failure
- Type: Requirement
- Requirement Source: Technical Staff
- Requirement Priority: Medium
- Description: The system must be designed for easy maintenance, allowing technical staff to perform updates, debugging, and backups without affecting normal operations or data integrity.

The requirement name ("Measure failure") does not match the content of its description, which refers to ease of maintenance, updates, and backups. The name seems to refer to failure measurement, which is more closely related to Reliability (RNF 4).

Features

3. Prioritized Features

Requirement Number	RF 1
Requirement Name	Patient Information Management
Guy	[✓ Requirement] Restriction
Source of Requirement	Administrative staff
Priority of the requirement	[✓ High/Essential] Medium/Desired Low/Optional

Requirement Number	RF 2
Requirement Name	Doctor Information Management
Guy	[✓ Requirement] Restriction
Source of Requirement	Administrative staff
Priority of the requirement	[✓ High/Essential] Medium/Desired Low/Optional

Requirement Number	RF 3
Requirement Name	Billing Module
Guy	[✓ Requirement] Restriction
Source of Requirement	Administrative staff
Priority of the requirement	[✓ High/Essential] Medium/Desired Low/Optional

Requirement Number	RF 4
Requirement Name	Medical Records
Guy	[✓ Requirement] Restriction
Source of Requirement	Administrative staff
Priority of the requirement	[✓ High/Essential] Medium/Desired Low/Optional

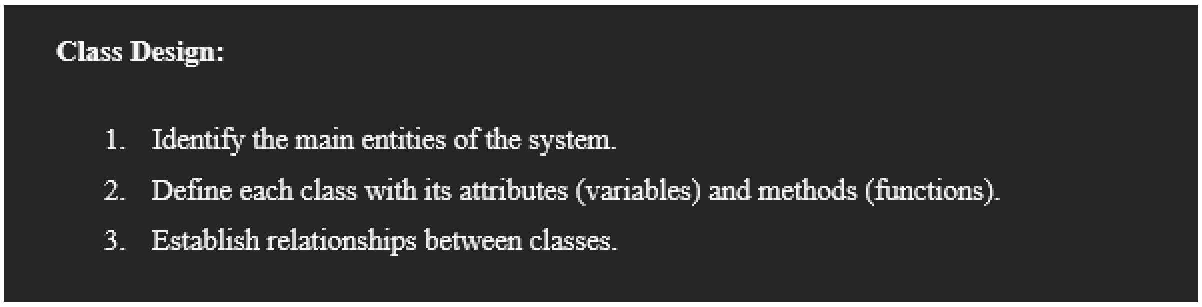
Everything is okey in this part

4. Tasks

While the general description is excellent, the tasks seem more like a development guide than specific tasks.

Example: “Identify the main entities of the system.”

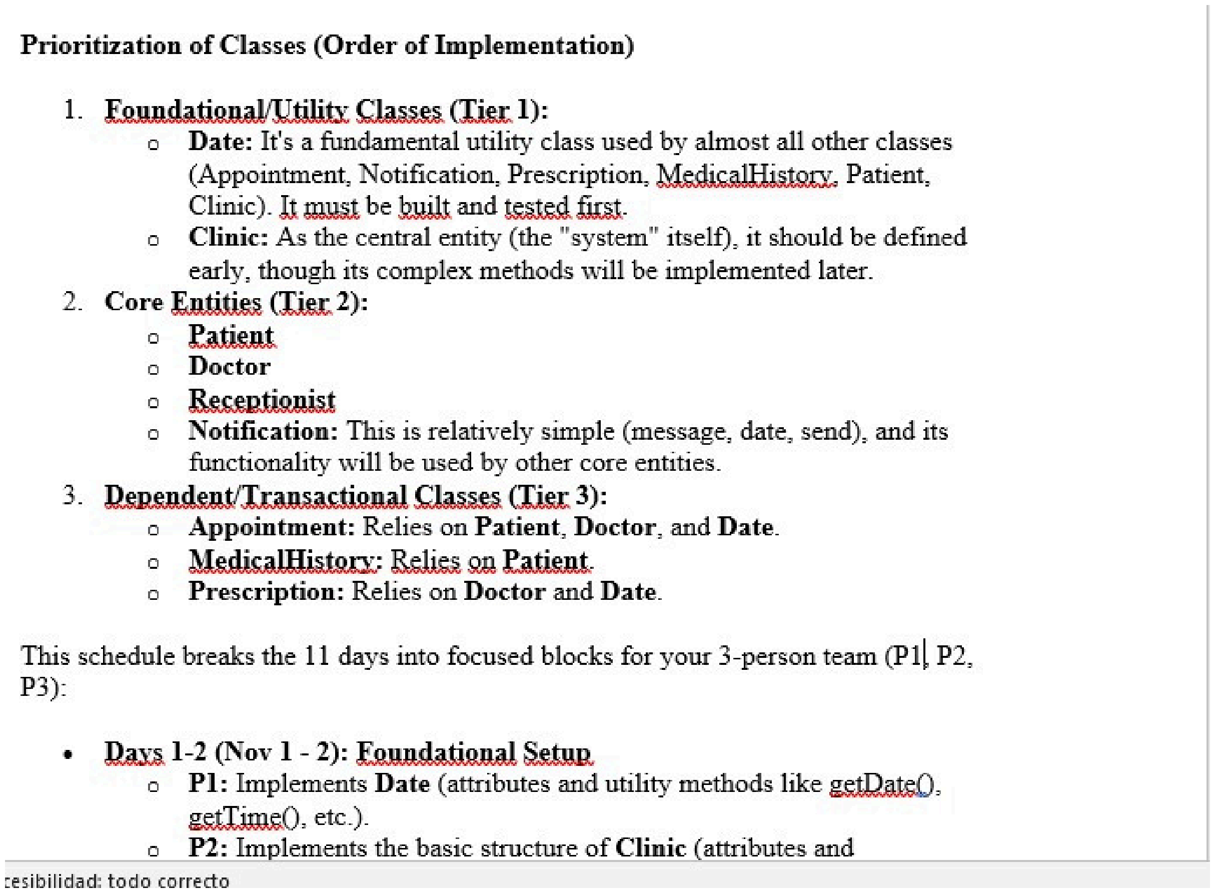
That's fine as a conceptual step, but not as a measurable or assignable task.



Class Design:

1. Identify the main entities of the system.
2. Define each class with its attributes (variables) and methods (functions).
3. Establish relationships between classes.

5. Plan



Prioritization of Classes (Order of Implementation)

1. **Foundational/Utility Classes (Tier 1):**
   - **Date:** It's a fundamental utility class used by almost all other classes (Appointment, Notification, Prescription, MedicalHistory, Patient, Clinic). It must be built and tested first.
   - **Clinic:** As the central entity (the "system" itself), it should be defined early, though its complex methods will be implemented later.
2. **Core Entities (Tier 2):**
   - **Patient**
   - **Doctor**
   - **Receptionist**
   - **Notification:** This is relatively simple (message, date, send), and its functionality will be used by other core entities.
3. **Dependent/Transactional Classes (Tier 3):**
   - **Appointment:** Relies on **Patient**, **Doctor**, and **Date**.
   - **MedicalHistory:** Relies on **Patient**.
   - **Prescription:** Relies on **Doctor** and **Date**.

This schedule breaks the 11 days into focused blocks for your 3-person team (P1, P2, P3):

- **Days 1-2 (Nov 1 - 2): Foundational Setup**
  - **P1:** Implements **Date** (attributes and utility methods like getDate(), getTime(), etc.).
  - **P2:** Implements the basic structure of **Clinic** (attributes and

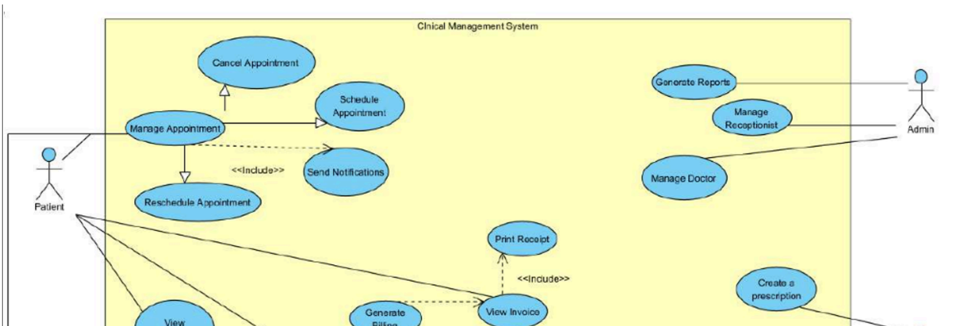
cesibilidad: todo correcto

The plan shows a well-organized and coherent implementation sequence based on class dependencies and includes specific working dates, which demonstrates good planning and coordination. However, it could be improved by briefly describing the expected deliverables or testing criteria for each stage to make the development process more measurable and goal-oriented.

Design

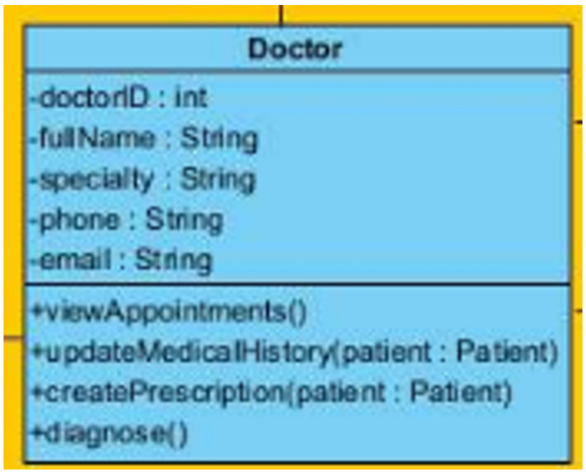
6. Architecture

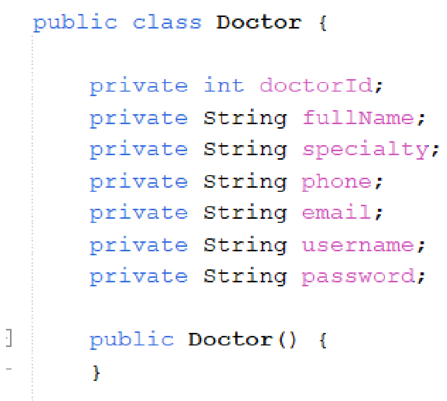
7. Use Case



Everything is okey in this part

8. Class Diagram





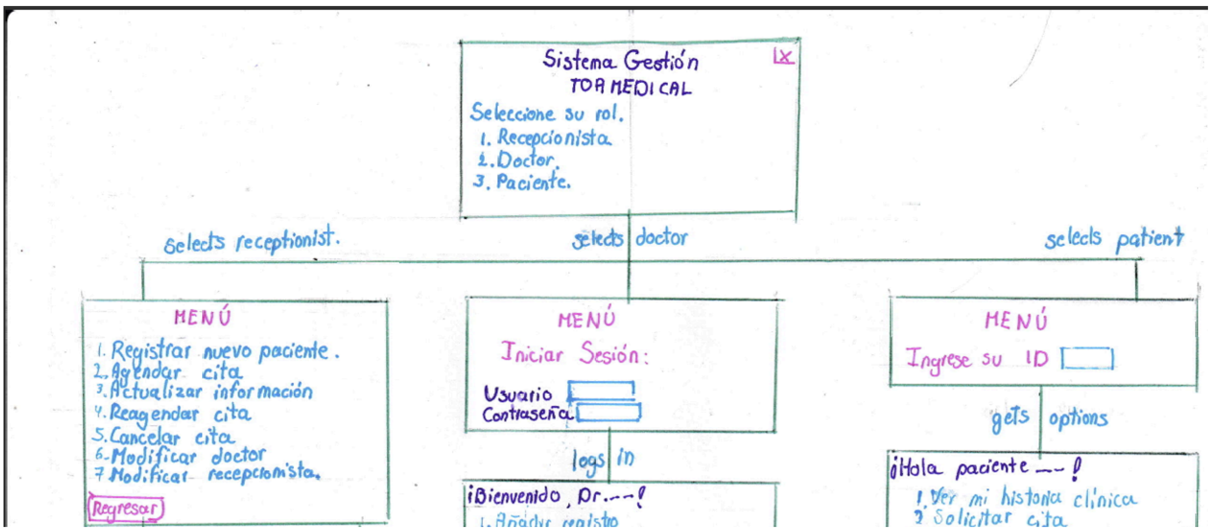
```
public class Doctor {

    private int doctorId;
    private String fullName;
    private String specialty;
    private String phone;
    private String email;
    private String username;
    private String password;

    public Doctor() {
    }
```

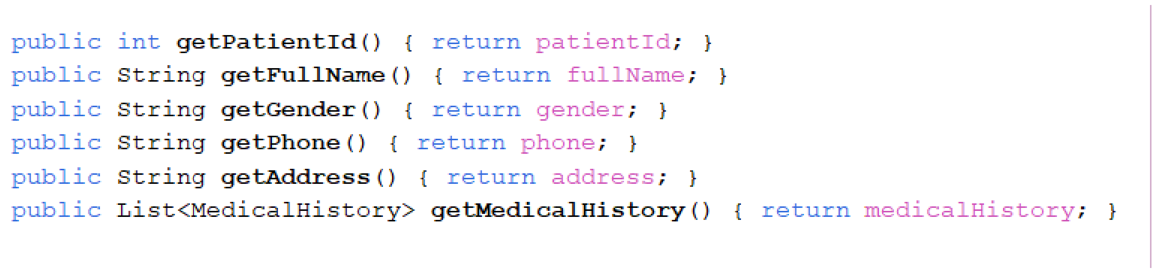
Lack of attributes in the class diagram compared to the code

9. Mockups

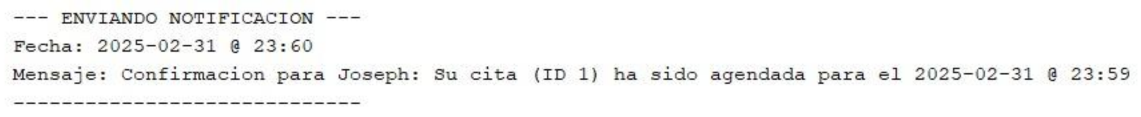


Everything is okey in this part

Prototype:
CMD,
10. functionality validation



```
public int getPatientId() { return patientId; }
public String getFullName() { return fullName; }
public String getGender() { return gender; }
public String getPhone() { return phone; }
public String getAddress() { return address; }
public List<MedicalHistory> getMedicalHistory() { return medicalHistory; }
```



```
--- ENVIANDO NOTIFICACION ---
Fecha: 2025-02-31 @ 23:60
Mensaje: Confirmacion para Joseph: Su cita (ID 1) ha sido agendada para el 2025-02-31 @ 23:59
-----------------------------
```

A lack of good coding practices and validation to prevent the use of past dates have been identified. Controls must be implemented to ensure that dates are selected on or after the current date.

GRADES
EVIDENCE:
Ord. Artifact, evaluation/10, observations:
1. Overview 9.5/10
2. SRS 9/10

Features

3. Prioritized Features 10/10
4. Tasks 9.5/10
5. Plan 9/10

Design

6. Architecture
7. Use Case 10/10

8. Class Diagram 9/10
9. Mockups 10/10

Prototype:
CMD,

10. functionality validation 8.75/10

AVERAGE: 9.41