Hospital Management System

Project documentation

OOP project

Group Members :

Maya Maged 120230241

Karen Labib 120230259

Mairoun 120230136

Kirolos Khairy 120230237

Overview:

The Hospital Management System is a Python project that helps organize basic hospital operations using OOP, a database, and a simple GUI and SQlite. The system allows patients to view doctors, book appointments, and see their prescriptions and bills. Doctors can log in, view their patients, create appointments, and write prescriptions. Staff can make offline booking and administrative tasksAll information is saved in an SQLite database, and the interface is built using Tkinter. The project aims to make hospital tasks easier, more organized, and accessible through a clean and interactive application.

OOP Design :

1 - Classes

The system uses the following main classes:

- **Person** (Base class)
- **Patient** (inherits Person)
- **Doctor** (inherits Person)
- **Staff** (inherits Person)
- **Appointment**
- **Prescription**
- **Bill**

2- Attributes

Person

- id
- name
- age
- gender

Patient

- problem
- assigned_doctor
- appointments_list
- prescriptions_list
- payment_status

Doctor

- specialty
- patient_list
- schedule

Staff

- role
- access_level

Appointment

- appointment_id
- patient_id
- doctor_id
- date
- notes

Prescription

- prescription_id
- patient_id
- doctor_id
- medicine
- dosage

- price

Bill

- patient_id
- items
- total_cost
- payment_status

3- Methods

Patient

- view_doctors()
- book_appointment()
- view_prescriptions()
- view_bill()
- pay_bill()

Doctor

- view_patients()
- write_prescription()
- create_appointment()
- view_appointments()

Staff

- register_patient_offline()
- create_appointment_for_patient()
- view_all_bills()
- search_records()
- delete_patient()
- update_patient_data()

Appointment

- save()
- delete()
- update()

Prescription

- save()

Bill

- calculate_total()
- update_payment_status()

4-OOP concepts Applied:

1- Inheritance

A clear inheritance hierarchy is used:

```
        Person
   /     |      \
 Patient   Doctor    Staff
```

This structure allows shared attributes (name, age, gender, ID) to be reused.

2-Polymorphism

Polymorphism is achieved by:

- Overriding show_details() in Patient, Doctor, and Staff
- Staff and Doctor both have appointment functions but behave differently
- Search methods vary depending on whether the user searches patients, doctors, or bills

3- Encapsulation

- Sensitive attributes (billing, medical problem, payment status) are hidden using private or protected variables
- Getter/Setter methods used to access and modify attributes safely

4- Abstraction

Abstraction is used through:

- Database layer in database.py to hide SQL queries from GUI
- Bill calculation hidden inside the Bill class

5- System Modules

```
project/
  main.py
  gui_main.py
  database.py
  hospital.db
  patient.py
  doctor.py
  appointment.py
  bill.py
  prescription.py
  staff.py
```

6- Application Features

- Patient Features

- Login
- View doctors
- Book appointments
- View prescriptions
- View bill
- Check payment status

- Doctor Features

- Login
- View assigned patients
- Create appointments
- Write prescriptions
- View appointment schedule
- View patient history

- Staff Features

- Register patients offline
- Book appointments for walk-in patients
- Edit patient information
- Delete incorrect records
- View all bills
- Update payment status
- Search all hospital records (patients/doctors/bills/appointments)
- Print or prepare bills

7- System Design (Simple)

Class diagram simplified

```
           Person
     /      |       \
 Patient     Doctor     Staff
     |        |       |
Appointment — Prescription — Bill
```

8- Database Structure

Patients

(id, name, age, gender, problem, assigned_doctor)

Doctors

(id, name, specialty)

Appointments

(id, patient_id, doctor_id, date, notes)

Prescriptions

(id, patient_id, doctor_id, medicine, dosage, price)

Bills

(patient_id, total_cost, payment_status)

9- How to Run the Project:

10- Screeshots / images :

11- Team Members & Contributions

Kirolos Tasks

- GUI design

Maya Tasks

- Database design
- Appointment module
- Search system **Patients + Appointments**
- Documentation
- Patient Dashboard

Karen Tasks

- Doctor dashboard
- Prescription system
- Bill calculation
- Search system **Doctors + Prescriptions + Bills**

Mairoun Tasks

- Staff dashboard
- CRUD (add/edit/delete)
- Payment status system
- Testing

12- Conclusion

The Hospital Management System demonstrates how OOP, GUI development, and database integration can work together to form a complete, functional application.
The project follows a modular structure, supports multiple roles (Patient, Doctor, Staff), and includes advanced features such as billing, search, offline registration, and CRUD operations.
The system is extendable and serves as a strong real-world simulation of hospital workflows.

---

8. Additional Notes

Requirements

- Python
- Tkinter
- SQLite3
- **Dependencies**

Future Enhancements

- Pharmacy module with the medicines needed for each specialty
- Export bills to PDF
- Medical Records File with X-ray files, Blood test results to be sent for the lab when the doctor requires, and the test results are sent to the doctor and viewed in the patient's history