**PROJECT**

**ON**

**HOSPITAL MANAGEMENT SYSTEM**

**Project Overview:**

This document provides a comprehensive overview of the Hospital Management System (HMS) project. The system aims to efficiently manage patient records, doctor appointments, and medical inventory within a hospital setting.

**Project Goals:**

* Efficiently store and retrieve patient records.
* Optimize doctor appointment scheduling.
* Manage medical inventory and track stock levels.
* Provide a foundation for future enhancements (e.g., billing, patient history, security).

**Code:**

import random

class Patient:

def \_\_init\_\_(self, id, name, age, ailment):

self.id = id

self.name = name

self.age = age

self.ailment = ailment

class TreeNode:

def \_\_init\_\_(self, patient):

self.patient = patient

self.left = None

self.right = None

class PatientRecords:

def \_\_init\_\_(self):

self.root = None

def insert(self, patient):

if self.root is None:

self.root = TreeNode(patient)

else:

self.\_insert(self.root, patient)

def \_insert(self, node, patient):

if patient.id < node.patient.id:

if node.left is None:

node.left = TreeNode(patient)

else:

self.\_insert(node.left, patient)

else:

if node.right is None:

node.right = TreeNode(patient)

else:

self.\_insert(node.right, patient)

def search(self, patient\_id):

if self.root is None:

return None

else:

return self.\_search(self.root, patient\_id)

def \_search(self, node, patient\_id):

if node is None or node.patient.id == patient\_id:

return node

elif patient\_id < node.patient.id:

return self.\_search(node.left, patient\_id)

else:

return self.\_search(node.right, patient\_id)

def print\_patients(self):

if self.root:

self.\_print\_patients(self.root)

def \_print\_patients(self, node):

if node:

self.\_print\_patients(node.left)

print(f"Patient ID: {node.patient.id}, Name: {node.patient.name}, Age: {node.patient.age}, Ailment: {node.patient.ailment}")

self.\_print\_patients(node.right)

class Doctor:

def \_\_init\_\_(self, id, name, specialization):

self.id = id

self.name = name

self.specialization = specialization

class DoctorAppointment:

def \_\_init\_\_(self, doctor, patient, time\_slot):

self.doctor = doctor

self.patient = patient

self.time\_slot = time\_slot

class DoctorSchedule:

def \_\_init\_\_(self):

self.schedule = {} # {doctor\_id: [Appointment, ...]}

def schedule\_appointment(self, doctor, patient, time\_slot):

if doctor.id not in self.schedule:

self.schedule[doctor.id] = []

self.schedule[doctor.id].append(DoctorAppointment(doctor, patient, time\_slot))

def print\_schedule(self):

for doctor\_id, appointments in self.schedule.items():

print(f"Doctor ID: {doctor\_id}")

for appointment in appointments:

print(f" Patient: {appointment.patient.name}, Time: {appointment.time\_slot}")

class InventoryItem:

def \_\_init\_\_(self, name, quantity):

self.name = name

self.quantity = quantity

class MedicalInventory:

def \_\_init\_\_(self):

self.inventory = {}

def add\_item(self, item):

self.inventory[item.name] = item

def update\_item(self, item\_name, quantity):

if item\_name in self.inventory:

self.inventory[item\_name].quantity = quantity

def check\_stock(self, item\_name):

if item\_name in self.inventory:

return self.inventory[item\_name].quantity

return None

def low\_stock\_alert(self, threshold):

low\_stock\_items = []

for item in self.inventory.values():

if item.quantity < threshold:

low\_stock\_items.append(item.name)

return low\_stock\_items

def print\_inventory(self):

for item\_name, item in self.inventory.items():

print(f"Item: {item\_name}, Quantity: {item.quantity}")

# ... (Security class, if needed)

def main():

# Create instances

patient\_records = PatientRecords()

doctor\_schedule = DoctorSchedule()

medical\_inventory = MedicalInventory()

# Sample data

patient1 = Patient(1, "vicky", 26, "Flu")

patient2 = Patient(2, "Afroz", 25, "Cold")

patient\_records.insert(patient1)

patient\_records.insert(patient2)

doctor1 = Doctor(101, "Dr. jalandhar", "General Medicine")

doctor2 = Doctor(102, "Dr. charan", "Pediatrics")

doctor\_schedule.schedule\_appointment(doctor1, patient1, "10:00 AM")

doctor\_schedule.schedule\_appointment(doctor2, patient2, "11:00 AM")

item1 = InventoryItem("Aspirin", 100)

item2 = InventoryItem("Bandage", 50)

medical\_inventory.add\_item(item1)

medical\_inventory.add\_item(item2)

# Output

print("Patient Records:")

patient\_records.print\_patients()

print("\nDoctor Schedule:")

doctor\_schedule.print\_schedule()

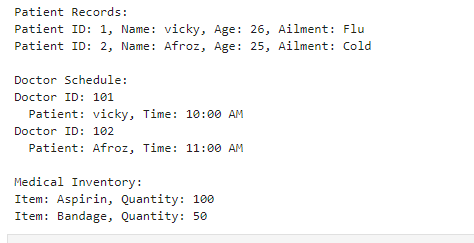
print("\nMedical Inventory:")

medical\_inventory.print\_inventory()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:**



**Explanation:**

**This project is like a simple computer system for a hospital.**

**Imagine a hospital.** It needs to keep track of patients, doctor appointments, and medicines, right?

**This computer program helps with that:**

* **Keeps track of patients:** It remembers who the patients are, how old they are, what's wrong with them, and gives them a number (ID).
* **Manages doctor appointments:** It schedules when patients can meet with doctors.
* **Controls the medicine cabinet:** It knows what medicines are there and how many pills or bottles are left.

**But it's a basic system.** It doesn't look pretty, and it can't do everything a real hospital needs. To make it better, we could add things like:

* A nice screen to look at
* A way to save information when the computer turns off
* Making sure only doctors and nurses can see patient information

**So, it's a starting point for a bigger hospital computer system.**