A Project Report

on

AUTOMATIC PATIENT MANAGEMENT SYSTEMS

Submitted in partial fulfillment of requirements for the award of the course

of

**EGB1202 Python Programming**

Under the guidance of

### Mr. C.NANDHAKUMAR M.E.,

### Assistant Professor / CSE

Submitted By



**ELAMURUGAN M (927623BEE028)**

**DEPARTMENT OF ELECTRONICS AND ELECTRONICS ENGINEERING**

**M.KUMARASAMY COLLEGE OF ENGINEERING**

(Autonomous)

**KARUR – 639 113**

DECEMBER 2024

# M. KUMARASAMY COLLEGE OF ENGINEERING

**(Autonomous Institution affiliated to Anna University, Chennai)**

# KARUR – 639 113

**BONAFIDE CERTIFICATE**

Certified that this project report on **“**AUTOMATIC PATIENT MANAGEMENT SYSTEMS**”** is the bonafide work of ELAMURUGAN **M (927623BEE028)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

|  |  |
| --- | --- |
| Signature | Signature |
| **Mr. C.NANDHAKUMAR M.E.,** | **Dr. J.UMA M.E.,Ph.D.,** |
| **SUPERVISOR,** | **HEAD OF THE DEPARTMENT,** |
| Department of Computer Science and Engineering, | Department of Electronics and Electronics Engineering, |
| M.Kumarasamy College of Engineering, | M.Kumarasamy College of Engineering, |
| Thalavapalayam, Karur -639 113. | Thalavapalayam, Karur -639 113. |



ii

**DEPARTMENT OF ELECTRONICS AND ELECTRONICS ENGINEERING**

**VISION OF THE INSTITUTION**

To emerge as a leader among the top institutions in the field of technical education

**MISSION OF THE INSTITUTION**

* Produce smart technocrats with empirical knowledge who can surmount the global challenges
* Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
* Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

**VISION OF THE DEPARTMENT**

To produce smart and dynamic professionals with profound theoretical and practical knowledge comparable with the best in the field.

**MISSION OF THE DEPARTMENT**

* Produce hi-tech professionals in the field of Electrical and Electronics Engineering by inculcating core knowledge.
* Produce highly competent professionals with thrust on research.
* Provide personalized training to the students for enriching their skills.

**PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

**PEO1:** Graduates will have flourishing career in the core areas of Electrical Engineering and allied disciplines.  
**PEO2:** Graduates will pursue higher studies and succeed in academic/research careers.  
**PEO3:** Graduates will be a successful entrepreneur in creating jobs related to Electrical and Electronics Engineering /allied disciplines.  
**PEO4:** Graduates will practice ethics and have habit of continuous learning for their success in the chosen career.

iii

**PROGRAM OUTCOMES (POs)**

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:**Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

**PSO1:** Apply the basic concepts of mathematics and science to analyse and design circuits, controls, Electrical machines and drives to solve complex problems.  
**PSO2:** Apply relevant models, resources and emerging tools and techniques to provide solutions to power and energy related issues & challenges.  
**PSO3:** Design, Develop and implement methods and concepts to facilitate solutions for electrical and electronics engineering related real world problems.

# 

# V

# ABSTRACT

The implementation of a comprehensive Patient Management System (PMS) is vital for enhancing healthcare delivery in today’s fast-paced medical environment. This system centralizes patient interactions, beginning with streamlined registration and data management, which allows healthcare providers to securely capture and access essential patient information.

The system also gathers and analyzes patient improvement data, allowing for tailored care interventions. Finally, automated discharge summaries ensure that patients receive clear follow-up instructions, promoting continuity of care. Overall, a Patient Management System not only streamlines operations but also elevates the quality of patient care, making it an essential asset for modern healthcare facilities.

Vi

# ABSTRACT WITH POs AND PSOs MAPPING

|  |  |  |
| --- | --- | --- |
| **ABSTRACT** | **POs MAPPED** | **PSOs MAPPED** |
| The implementation of automatic Patient Management System (PMS) is vital for enhancing healthcare delivery in today’s fast-paced medical environment. This system centralizes patient interactions, beginning with streamlined registration and data management, which allows healthcare providers to securely capture and access essential patient information. | **Po1**  **Po2**  **Po3**  **Po4**  **po5**  **Po6**  **Po8** |  |

Note: 1- Low, 2-Medium, 3- High

**SUPERVISOR HEAD OF THE DEPARTMENT**

vii

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER** No. | TITLE | **PAGE** No. |
|  | ABSTRACT |  |
| 1 | INTRODUCTION |  |
|  | 1.1 Objective | 01 |
|  | 1.2 Overview | 01 |
|  | 1.3 Python Programming concepts | 02 |
| 2 | PROJECT METHODOLOGY |  |
|  | 2.1Proposed Work | 03 |
|  | 2.2 Block Diagram | 04 |
| 3 | MODULE DESCRIPTION |  |
|  | 3.1 Patient Registration Module | 05 |
|  | 3.2 Appointment Scheduling Module | 05 |
|  | 3.3 Prescription and Medication Management Module | 05 |
|  | 3.4 Discharge and Transfer Module | 05 |
|  | 3.5 Patient Feedback and Satisfaction Module | 06 |
| 4 | RESULTS AND DISCUSSION | 07 |
| 5 | CONCLUSION | 08 |
|  | REFERENCES | 09 |
|  | APPENDIX | 10 |

# viii

# CHAPTER 1

# INTRODUCTION

# Objective

# The objective of this presentation is to showcase the design and implementation of a Python-based **Automatic Patient Management System**. This system aims to improve healthcare delivery by automating essential hospital functions such as patient registration, room allocation, prescription management, discharge processes, and patient feedback collection. The focus is on creating a user-friendly, modular system that enhances operational efficiency while laying the groundwork for future scalability and improvements.

# Overview

This project, designed for a Python programming course, presents a comprehensive **Patient Management System (PMS)** with functionalities including:

1. **Patient Registration and Appointment Scheduling**: Automates data collection and appointment scheduling for new and existing patients.
2. **Room and Prescription Management**: Handles room allocations, tracks medications, and manages patient prescriptions.
3. **Discharge and Billing**: Simplifies discharge procedures with automated billing based on hospital stay and treatments.
4. **Patient Feedback and Analytics**: Gathers feedback to evaluate and improve service quality.
5. **Proposed System Architecture**: Implements features using Python programming concepts like classes, objects, list comprehensions, and exception handling for efficient system operations.

The discussion also highlights the limitations, such as lack of data persistence and scalability, and suggests future enhancements like integration with databases, realistic billing, and a graphical interface.

**1.3 Python Programming Concepts**

* Classes and Objects
* Constructor Method
* Random Module
* List Comprehensions
* Control Flow
* String Representation
* User Input
* Matrix Representation
* Basic Exception Handling

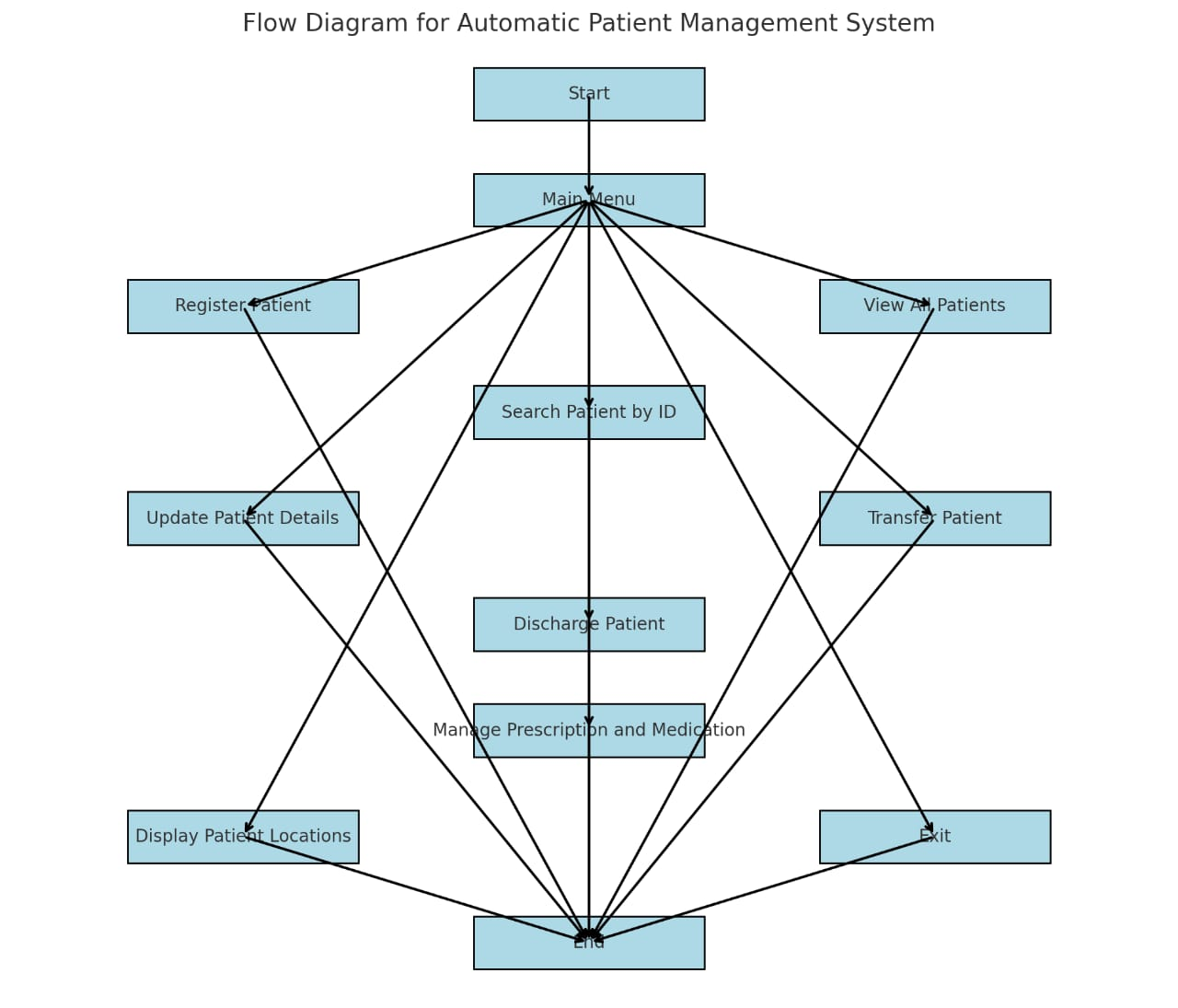
**CHAPTER 2**

**PROJECT METHODOLOGY**

**2.1Proposed Work**

1. Registering Patients: When a new patient arrives, the system collects their personal details like name, age, contact info, and assigned doctor. It then checks for an available room and assigns the patient a space. An appointment is scheduled for the patient, and their details are stored in the system.
2. Viewing and Searching Patients: Staff can view all registered patients or search for a specific patient by their ID. This allows easy access to a patient's details.
3. Updating Patient Details: If any patient information needs to be updated (such as a change in contact details or doctor), the system allows staff to make those changes.
4. Transferring Patients: If a patient needs to be moved to a different room, the staff can select a new room, and the system checks if it's available. If it is, the patient’s room assignment is updated.
5. Managing Prescriptions: Doctors or staff can add medications and injections to a patient's prescription. The system tracks the medications, dosages, and timings for each patient.
6. Discharging Patients: When a patient is ready to leave, the system generates a bill based on their stay duration, room charges, medications, and any other treatments. The patient is then discharged, and their room is marked as available again.
7. Room and Patient Location: The system also shows which rooms are occupied or empty, helping staff manage room assignments more efficiently.
8. Main Menu: All of these functions are available through a simple menu that guides hospital staff to perform their tasks in a straightforward way.

**2.2 Block Diagram**



**Figure 2.2.1**

**CHAPTER 3**

**MODULE DESCRIPTION**

#### ****3.1 Patient Registration Module****

This module collects comprehensive patient details, including name, age, contact information, and medical history. It also assigns a unique patient ID and a room in the hospital, ensuring that all records are systematically maintained for easy access and efficient tracking.

#### ****3.2 Appointment Scheduling Module****

Handles the scheduling of patient appointments with doctors based on availability. It sends automated reminders, ensuring that patients and medical personnel are well-informed of their schedules, minimizing delays and overlaps.

#### ****3.3 Prescription and Medication Management Module****

Automates the prescription issuance process by allowing doctors to add and manage patient medications. The system monitors dosage schedules, medication types, and refill requirements, ensuring accurate treatment.

#### ****3.4 Discharge and Transfer Module****

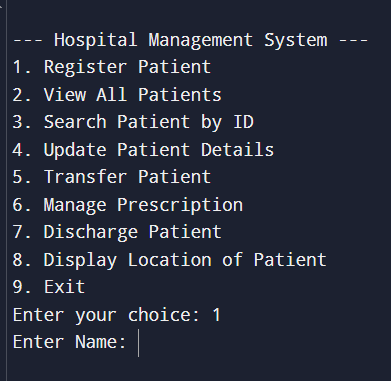
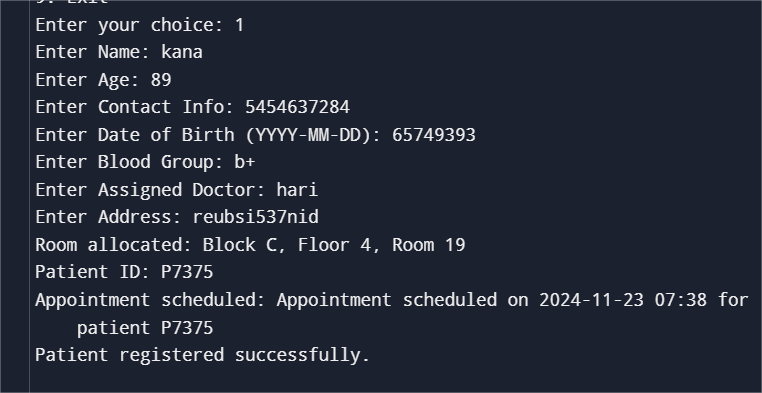
Manages the discharge process by calculating bills based on room charges, treatments, and medication costs. It also facilitates patient transfers between departments, ensuring continuity of care and optimizing resource utilization.

#### ****3.5 Patient Feedback and Satisfaction Module****

This module gathers valuable feedback from patients about their experience, analyzing responses to assess satisfaction levels. This information aids in identifying service improvement areas and maintaining high-quality care standards.

**CHAPTER 4**

**RESULTS AND DISCUSSION**

**Discussion :**

The system demonstrates a modular and efficient approach for managing essential hospital operations, including patient registration, room allocation, prescription management, and discharge processes. It is effective for small-scale use, offering a clear and interactive CLI interface. However, the system has some limitations. The lack of data persistence means that all records are lost upon exiting the program, making it unsuitable for long-term use without integration with a database.

Randomized billing, while functional for demonstration purposes, does not reflect actual patient expenses and would need to be replaced with accurate, service-based calculations for real-world applications. Limited input validation increases the risk of errors or invalid data being entered. Additionally, the fixed hospital layout limits scalability, making it challenging to adapt to larger facilities.

Future enhancements such as persistent storage, a realistic billing module, advanced validation, and a graphical user interface (GUI) would significantly improve usability, reliability, and scalability, allowing the system to serve medium- to large-sized hospitals effectively.

**CHAPTER 5**

**CONCLUSION**

The **Automatic Patient Management System** represents a significant step forward in modernizing hospital management practices. By integrating patient registration, appointment scheduling, prescription management, and discharge processes into a cohesive, Python-based system, it delivers enhanced operational efficiency and improved patient care.

While the current implementation is optimized for small-scale use with a command-line interface (CLI), its limitations, such as the lack of data persistence and randomized billing, highlight opportunities for future enhancement. By adopting persistent storage, advanced validation mechanisms, and a graphical user interface (GUI), the system can scale to meet the needs of larger healthcare facilities.

In conclusion, the project lays a solid foundation for an adaptable and comprehensive hospital management system. Its modular approach and potential for scalability make it a valuable asset for advancing healthcare management in increasingly demanding environments.

**REFERENCES:**

1. Elmasri, R., & Navathe, S. B. (2016). Fundamentals of Database Systems (7th Edition). Pearson.

This book provided foundational knowledge for structuring and managing data systematically, which influenced the design of the patients dictionary and room allocation logic in the code.

1. Kanetkar, Y. (2019). Let Us Python: Python Is Future, Embrace It Fast. BPB Publications.

Concepts on Python programming basics, including CLI design and modular programming, were referenced to build the interactive functionalities of the system.

1. Goyal, M. (2016). Hospital Management and Administration: Principles and Practice. CBS Publishers & Distributors.

Helped understand real-world hospital workflows like patient registration, room allocation, and billing, which were modeled into the system.

1. Tiwari, A., & Goyal, A. (2020). Practical Python Programming. McGraw Hill.

Provided practical examples and guidance on implementing Python programs with user input, error handling, and modular design.

**APPENDIX**

import random

from datetime import datetime

HOSPITAL = {"Block A": [[None for \_ in range(50)] for \_ in range(4)],

"Block B": [[None for \_ in range(50)] for \_ in range(4)],

"Block C": [[None for \_ in range(50)] for \_ in range(4)]}

patients = {}

def get\_random\_room\_locations():

locations = []

for block, floors in HOSPITAL.items():

for floor\_num, floor in enumerate(floors):

for room\_num in range(len(floor)):

locations.append((block, floor\_num, room\_num))

random.shuffle(locations)

return locations

def register\_patient():

patient\_id = f"P{random.randint(1000, 9999)}"

name = input("Enter Name: ")

age = int(input("Enter Age: "))

contact = input("Enter Contact Info: ")

dob = input("Enter Date of Birth (YYYY-MM-DD): ")

blood\_group = input("Enter Blood Group: ")

doctor\_assigned = input("Enter Assigned Doctor: ")

address = input("Enter Address: ")

for block, floor\_num, room\_num in get\_random\_room\_locations():

if HOSPITAL[block][floor\_num][room\_num] is None:

HOSPITAL[block][floor\_num][room\_num] = patient\_id

room\_location = f"{block}, Floor {floor\_num + 1}, Room {room\_num + 1}"

print(f"Room allocated: {room\_location}")

appointment = generate\_appointment(patient\_id)

patients[patient\_id] = {

"Patient ID": patient\_id,

"Name": name,

"Age": age,

"Contact": contact,

"DOB": dob,

"Blood Group": blood\_group,

"Doctor": doctor\_assigned,

"Address": address,

"Room": room\_location,

"Appointment": appointment,

"Tablets": [],

"Injection": [],

"Surgery": None

}

print(f"Patient ID: {patient\_id}")

print(f"Appointment scheduled: {appointment}")

print("Patient registered successfully.")

return

print("No available rooms.")

def generate\_appointment(patient\_id):

appointment\_date = datetime.now().strftime("%Y-%m-%d %H:%M")

return f"Appointment scheduled on {appointment\_date} for patient {patient\_id}"

def view\_all\_patients():

if not patients:

print("No patients registered.")

return

for patient in patients.values():

print(patient)

def search\_patient\_by\_id():

patient\_id = input("Enter Patient ID: ")

patient = patients.get(patient\_id)

if patient:

print(patient)

else:

print("Patient not found.")

def update\_patient\_details():

patient\_id = input("Enter Patient ID: ")

patient = patients.get(patient\_id)

if patient:

print("Updating details for:", patient["Name"])

patient["Name"] = input("Enter new name: ") or patient["Name"]

patient["Age"] = input("Enter new age: ") or patient["Age"]

patient["Contact"] = input("Enter new contact info: ") or patient["Contact"]

patient["DOB"] = input("Enter new date of birth (YYYY-MM-DD): ") or patient["DOB"]

patient["Blood Group"] = input("Enter new blood group: ") or patient["Blood Group"]

patient["Doctor"] = input("Enter new assigned doctor: ") or patient["Doctor"]

patient["Address"] = input("Enter new address: ") or patient["Address"]

print("Details updated successfully.")

else:

print("Patient not found.")

def transfer\_patient():

patient\_id = input("Enter Patient ID: ")

patient = patients.get(patient\_id)

if patient:

current\_location = patient["Room"]

print(f"Current location: {current\_location}")

block, floor, room = parse\_location(current\_location)

HOSPITAL[block][floor][room] = None

print("Select new location for transfer:")

block = input("Enter Block (A/B/C): ").strip().upper()

floor = int(input("Enter Floor (1-4): ")) - 1

room = int(input("Enter Room (1-50): ")) - 1

new\_location = f"Block {block}, Floor {floor + 1}, Room {room + 1}"

if HOSPITAL[f"Block {block}"][floor][room] is None:

HOSPITAL[f"Block {block}"][floor][room] = patient\_id

patient["Room"] = new\_location

print(f"Patient transferred to: {new\_location}")

else:

print("Selected room is occupied. Transfer failed.")

else:

print("Patient not found.")

def manage\_prescription\_and\_medication():

patient\_id = input("Enter Patient ID: ")

patient = patients.get(patient\_id)

if patient:

print(f"Managing prescription for {patient\_id}")

problem = input("Enter the patient's problem: ")

while True:

injection\_needed = input("Add injection (Yes-1, No-2)? ")

if injection\_needed == "1":

name = input("Enter injection name: ")

dosage = input("Enter dosage: ")

ml = input("Enter dosage in ML: ")

patient["Injection"].append({"Name": name, "Dosage": dosage, "ML": ml})

elif injection\_needed == "2":

break

while True:

medication\_needed = input("Add medication (Yes-1, No-2)? ")

if medication\_needed == "1":

name = input("Enter medication name: ")

timings = {

"Morning": input("Morning (1/0): ") == "1",

"Afternoon": input("Afternoon (1/0): ") == "1",

"Evening": input("Evening (1/0): ") == "1",

"Night": input("Night (1/0): ") == "1",

}

days = int(input("Enter number of days: "))

patient["Tablets"].append({"Name": name, "Timings": timings, "Days": days})

elif medication\_needed == "2":

break

print(f"Updated prescription for {patient\_id}:")

print("Injections:", patient["Injection"])

print("Medications:", patient["Tablets"])

else:

print("Patient not found.")

def discharge\_patient():

patient\_id = input("Enter Patient ID: ")

patient = patients.pop(patient\_id, None)

if patient:

block, floor, room = parse\_location(patient["Room"])

HOSPITAL[block][floor][room] = None

stay\_duration = random.randint(1, 10)

room\_charge = stay\_duration \* random.randint(1000, 3000)

medicine\_charge = sum(random.randint(500, 2000) for \_ in patient["Tablets"])

injection\_charge = sum(random.randint(100, 500) for \_ in patient["Injection"])

surgery\_charge = random.randint(50000, 100000) if patient["Surgery"] else 0

total\_bill = room\_charge + medicine\_charge + injection\_charge + surgery\_charge

print("\n--- Bill Details ---")

print(f"Stay Duration: {stay\_duration} days")

print(f"Room Charge: {room\_charge} INR")

print(f"Medicine Charge: {medicine\_charge} INR")

print(f"Injection Charge: {injection\_charge} INR")

print(f"Surgery Charge: {surgery\_charge} INR")

print(f"Total Bill: {total\_bill} INR")

feedback\_messages = [

"Excellent service! Very satisfied.",

"Good facilities, but could improve in room cleanliness.",

"Doctors and nurses were very caring and helpful.",

"Overall experience was pleasant and satisfactory.",

"Impressed by the speed and efficiency of the services.",

"Needs improvement in patient communication, but good overall."

]

feedback = random.choice(feedback\_messages)

print("\n--- Feedback ---")

print(f"Patient Feedback: {feedback}")

print("Patient discharged successfully.")

else:

print("Patient not found.")

def display\_location\_matrix():

for block, floors in HOSPITAL.items():

print(f"\n--- {block} ---")

for floor\_num, floor in enumerate(floors):

print(f"Floor {floor\_num + 1}:")

for room\_num in range(0, 50, 10):

row = [

patients[HOSPITAL[block][floor\_num][i]]["Patient ID"] if HOSPITAL[block][floor\_num][i] else "Empty"

for i in range(room\_num, room\_num + 10)

]

print(" | ".join(row))

print()

def parse\_location(location):

parts = location.split(", ")

block = parts[0]

floor = int(parts[1].split(" ")[1]) - 1

room = int(parts[2].split(" ")[1]) - 1

return block, floor, room

def main():

while True:

print("\n--- Hospital Management System ---")

print("1. Register Patient")

print("2. View All Patients")

print("3. Search Patient by ID")

print("4. Update Patient Details")

print("5. Transfer Patient")

print("6. Manage Prescription")

print("7. Discharge Patient")

print("8. Display Location of Patient")

print("9. Exit")

choice = input("Enter your choice: ")

if choice == "1":

register\_patient()

elif choice == "2":

view\_all\_patients()

elif choice == "3":

search\_patient\_by\_id()

elif choice == "4":

update\_patient\_details()

elif choice == "5":

transfer\_patient()

elif choice == "6":

manage\_prescription\_and\_medication()

elif choice == "7":

discharge\_patient()

elif choice == "8":

display\_location\_matrix()

elif choice == "9":

print("Exiting the system.")

break

else:

print("Invalid choice. Please try again.")

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

main()