**Project Title: Patient Management System**

**Name: Yuvan Nisanth J**

**Date Submitted: 17.09.2024**

**1.Aim of the Project:**

**Primary Objective:**

**1) Develop a robust patient management system that efficiently handles patient information and medical records.**

**2) Achieve seamless integration of patient data management with functionality to generate CSV reports for patient details.**

**Aims:**

**1) Implement a system for storing and retrieving patient details.**

**2) Provide functionalities for printing patient details and saving them in a CSV format.**

**3) Ensure error handling for file operations and user inputs.**

**2. Business Problem or Problem Statement**

**Problem Statement:**

**1) Many healthcare facilities struggle with managing patient information due to fragmented data storage and inefficient record-keeping processes.**

**2) There is a need for a centralized system to manage patient details, track appointments, and maintain medical history in an accessible format.**

**3) This project addresses the challenge of integrating patient data into a cohesive system that supports easy retrieval and CSV report generation, thus enhancing operational efficiency and data accuracy.**

**3. Project Description**

**1) Overview:**

**\* The Patient Management System is designed to manage patient information and generate detailed reports.**

**2) Scope:**

**\* Store patient details including medical history and appointments.**

**\* Generate and save patient information in CSV format for easy sharing and record-keeping.**

**3) Objectives:**

**\* Provide a user-friendly interface for entering and viewing patient data.**

**\* Ensure data integrity and accurate reporting.**

**4) Technologies Used:**

**\* Python programming language for backend logic.**

**\* CSV for data export and storage.**

**\* Basic error handling to manage file operations and user inputs.**

**5) Key Features:**

**\* Store comprehensive patient data.**

**\* Print patient details to the console.**

**\* Export patient details to a CSV file with proper formatting.**

**4.Functionalities**

**Main Functionalities:**

**1) Patient Data Storage:**

**\* Store patient ID, name, age, gender, address, phone, email, medical history, and appointments.**

**2)Details Printing:**

**\* Print patient details including medical history and upcoming appointments to the console.**

**3) CSV File Export:**

**\* Save patient information to a CSV file, ensuring proper formatting and error handling.**

**4) Directory Management:**

**\* Automatically create necessary directories for file storage if they do not exist.**

**5) Error Handling:**

**\* Handle common file-related errors such as permission issues and file not found errors.**

**5.Input Versatility with Error Handling and Exception Handling**

**1) Input Versatility:**

**\* Handles patient ID input from the user and verifies if it exists in the database.**

**\* Manages different types of data including text and dates for patient details and appointments.**

**2) Error Handling:**

**File Operations:**

**\* Checks for permissions and existence of directories before attempting file write operations.**

**3) User Input:**

**\* Provides feedback if the patient ID is invalid or not found in the database.**

**4) Exception Handling:**

**\* Catches and reports errors during file writing, including permission and file not found errors.**

**6. Code Implementation**

**Implementation Insights:**

**Key Algorithms:**

**\* Data retrieval from a dictionary structure.**

**\* CSV file writing using Python’s csv module.**

**Data Structures:**

**\* Use of Python dictionaries and lists to manage patient data and appointments.**

**Code Organization:**

**\* Class-based design for encapsulating patient details and methods.**

**\* Main function to drive the user interaction and handle input/output operations.**

**7.Results and Outcomes**

**1) Results Achieved:**

**\* Successfully implemented a system to manage patient data and generate CSV reports.**

**2) Outcomes:**

**\* Effective data management with error-free CSV file creation.**

**\* Improved user interaction with clear error messages and feedback.**

**3) Visual Representations:**

**\* Sample CSV file output.**

**\* Example screenshots of the console output showing patient details and appointments.**

**8.Conclusion**

**1) Summary:**

**\* The Patient Management System provides a streamlined approach to managing and reporting patient information.**

**\* The project has met its objectives of data storage, retrieval, and export, with a focus on error handling and user feedback.**

**2) Future Developments:**

**\* Potential for integrating with more advanced systems for enhanced data management.**

**\* Possibility to expand functionalities to include patient data updates and more comprehensive reporting.**

**CODING:**

import csv  
  
  
*# Patient class definition*class Patient:  
 def \_\_init\_\_(self, patient\_id, name, age, gender, address, phone, email, medical\_history, appointments, file\_path):  
 self.patient\_id = patient\_id  
  
 self.name = name  
 self.age = age  
 self.gender = gender  
 self.address = address  
 self.phone = phone  
 self.email = email  
 self.medical\_history = medical\_history  
 self.appointments = appointments  
 self.file\_path = file\_path  
  
 def print\_details(self):  
 *# Print patient details to console* print(f"Name: {self.name}")  
 print(f"Age: {self.age}")  
 print(f"Gender: {self.gender}")  
 print(f"Address: {self.address}")  
 print(f"Phone: {self.phone}")  
 print(f"Email: {self.email}")  
 print("Medical History:")  
 print(f" Conditions: {', '.join(self.medical\_history['conditions'])}")  
 print(f" Allergies: {', '.join(self.medical\_history['allergies'])}")  
 print(f" Medications: {', '.join(self.medical\_history['medications'])}")  
 print("Appointments:")  
 for appt in self.appointments:  
 print(f" Date: {appt['date']}, Time: {appt['time']}, Doctor: {appt['doctor']}")  
  
 *# Save patient details to a CSV file* try:  
 *# Ensure the directory exists* with open(self.file\_path, 'w', newline='', encoding='utf-8') as file:  
 writer = csv.writer(file)  
 writer.writerow(['Field', 'Value'])  
 writer.writerow(['Name', self.name])  
 writer.writerow(['Age', self.age])  
 writer.writerow(['Gender', self.gender])  
 writer.writerow(['Address', self.address])  
 writer.writerow(['Phone', self.phone])  
 writer.writerow(['Email', self.email])  
 writer.writerow(['Conditions', ', '.join(self.medical\_history['conditions'])])  
 writer.writerow(['Allergies', ', '.join(self.medical\_history['allergies'])])  
 writer.writerow(['Medications', ', '.join(self.medical\_history['medications'])])  
 for appt in self.appointments:  
 writer.writerow(  
 [f"Appointment on {appt['date']}", f"Time: {appt['time']}, Doctor: {appt['doctor']}"])  
 print(f"CSV file created at {self.file\_path}")  
 except PermissionError:  
 print(f"Permission denied: Unable to write to {self.file\_path}.")  
 except FileNotFoundError:  
 print(f"File not found: {self.file\_path}.")  
 except Exception as e:  
 print(f"An error occurred while writing to the file: {e}")  
  
  
*# Patient database*patient\_database = {  
 'patient\_001': {  
 'name': 'John Doe',  
 'age': 45,  
 'gender': 'Male',  
 'address': '123 Elm Street, Springfield',  
 'phone': '555-1234',  
 'email': 'john.doe@example.com',  
 'medical\_history': {  
 'conditions': ['Hypertension', 'Diabetes'],  
 'allergies': ['Penicillin'],  
 'medications': ['Lisinopril', 'Metformin']  
 },  
 'appointments': [  
 {'date': '2024-09-15', 'time': '10:00', 'doctor': 'Dr. Smith'},  
 {'date': '2024-10-01', 'time': '11:00', 'doctor': 'Dr. Adams'}  
 ]  
 },  
 'patient\_002': {  
 'name': 'Jane Smith',  
 'age': 34,  
 'gender': 'Female',  
 'address': '456 Oak Avenue, Springfield',  
 'phone': '555-5678',  
 'email': 'jane.smith@example.com',  
 'medical\_history': {  
 'conditions': ['Asthma'],  
 'allergies': ['None'],  
 'medications': ['Albuterol']  
 },  
 'appointments': [  
 {'date': '2024-09-20', 'time': '14:00', 'doctor': 'Dr. Lee'}  
 ]  
 }  
}  
  
  
def main():  
 user\_detail = input("ENTER THE PATIENT ID: ").strip()  
  
 *# Check if patient\_id exists in the database* if user\_detail in patient\_database:  
 patient\_data = patient\_database[user\_detail]  
 file\_path = r'E:\csvfile.txt' *# Update the path as needed* try:  
 patient = Patient(  
 patient\_id=user\_detail,  
 name=patient\_data['name'],  
 age=patient\_data['age'],  
 gender=patient\_data['gender'],  
 address=patient\_data['address'],  
 phone=patient\_data['phone'],  
 email=patient\_data['email'],  
 medical\_history=patient\_data['medical\_history'],  
 appointments=patient\_data['appointments'],  
 file\_path=file\_path  
 )  
 patient.print\_details()  
 except TypeError as e:  
 print(f"Missing data in the patient record: {e}")  
 except Exception as e:  
 print(f"An unexpected error occurred: {e}")  
 else:  
 print("Patient ID not found in the database.")  
main()