Patient Health Management System

Milestone: EER And UML Diagram

Group: - Data Dynamos
Muskan Sharma
Hemalikaa Thirumavalavan
Mohit Deshpande

(778)869-1739 (Muskan)
(437)217-5392 (Hemalikaa)
(437)662-4519 (Mohit)

sharma.muskan@northeastern.edu
thirumavalavan.h@northeastern.edu
deshpande.mo@northeastern.edu

Percentage of Effort Contributed by Student1: 33.3%

Percentage of Effort Contributed by Student2: 33.3%

Percentage of Effort Contributed by Student3: 33.3%

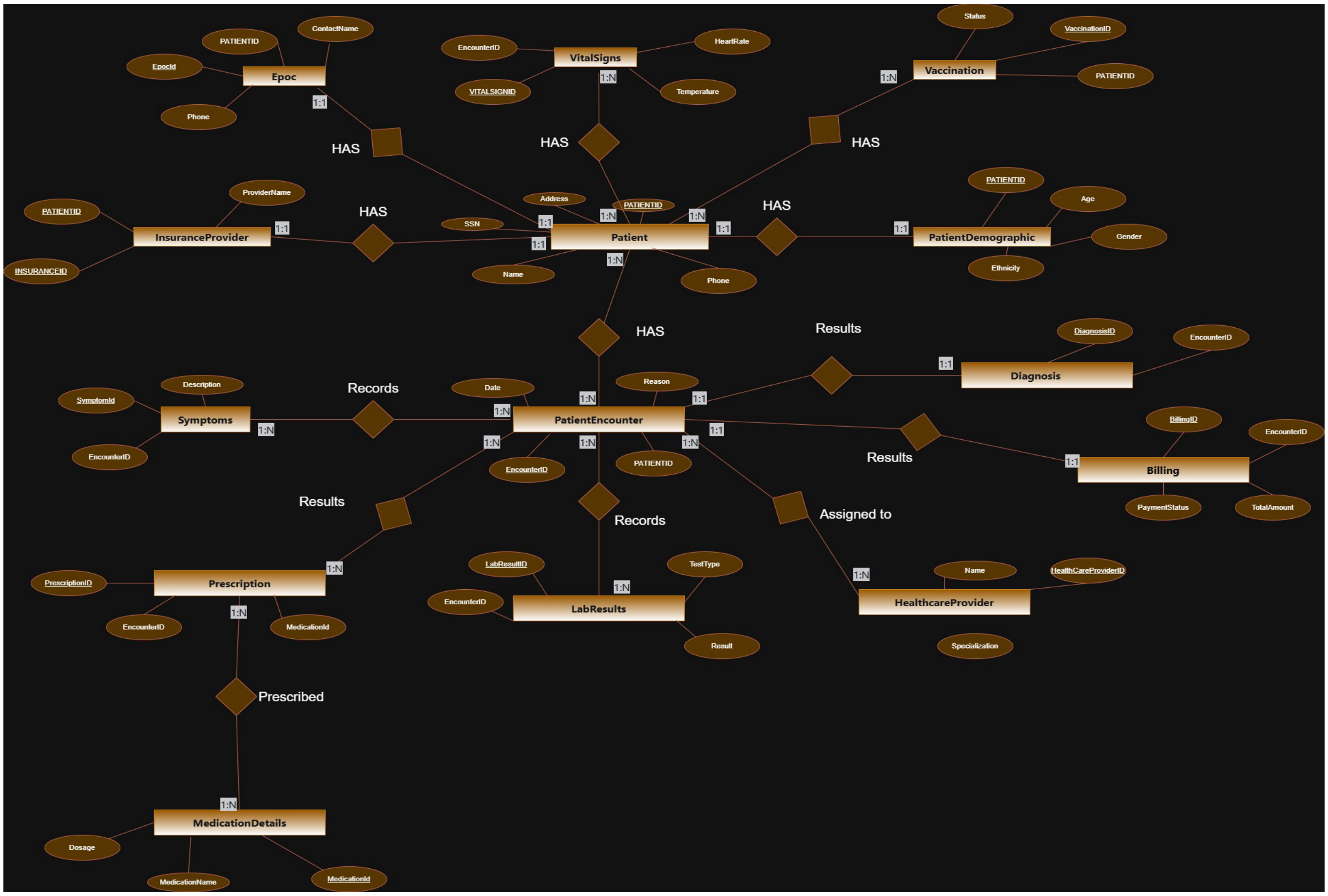
Signature of Student 1: Muskan Sharma

Signature of Student 2: Hemalikaa Thirumavalavan

Signature of Student 3: Mohit Deshpande

Submission Date: 13/2/2025

EER Diagram

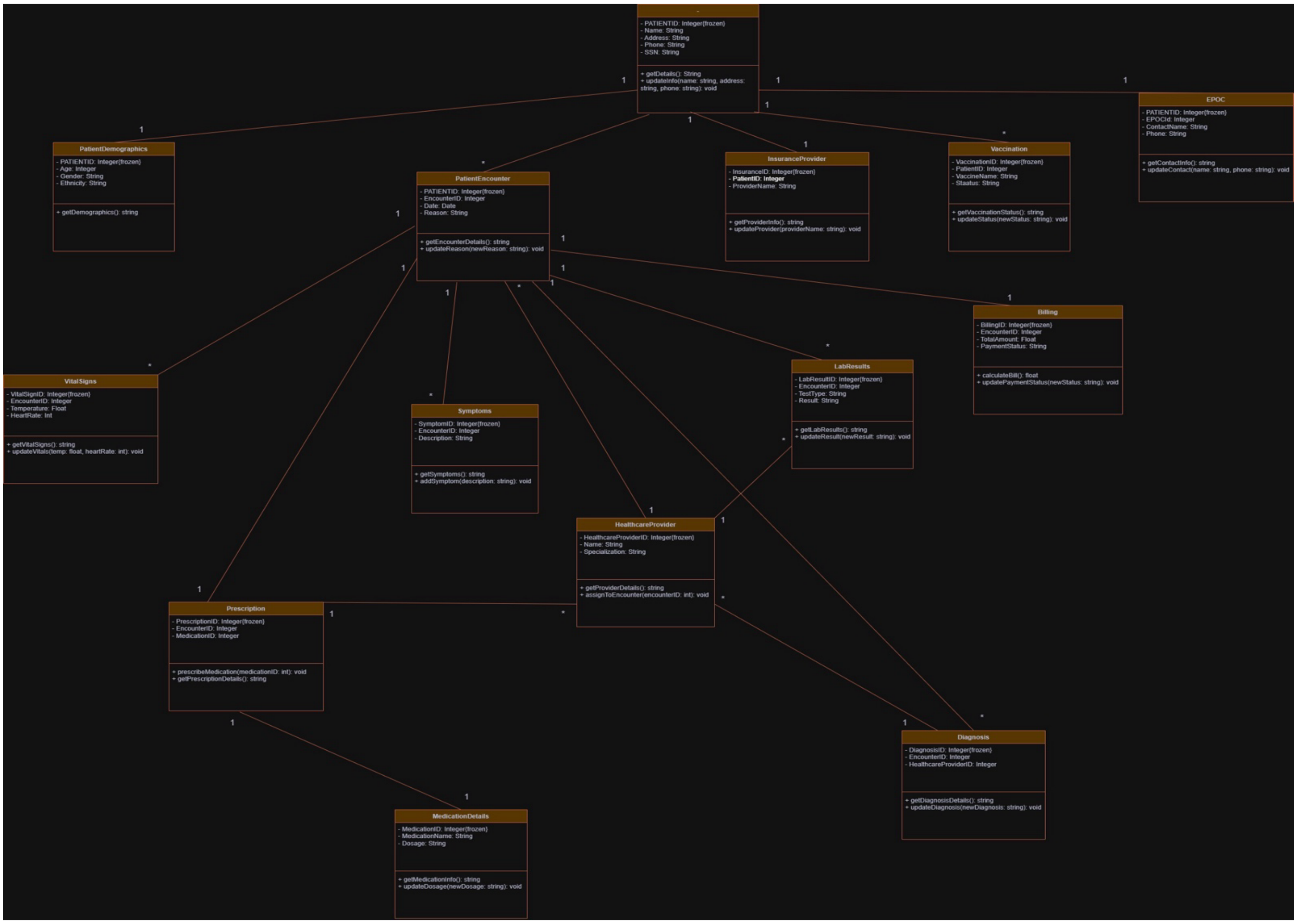


An Enhanced Entity-Relationship (EER) diagram presents a patient health management system to handle patient records together with medical encounter history and diagnosis along with prescriptions and healthcare billing operations. The system relies on the Patient entity which contains essential attributes including PATIENTID and Name alongside Phone and SSN and Address. The database maintains demographic information and medical records and health interaction data through several linked entities which contain the relevant patient data. A one-to-one (1:1) relationship connects the Patient entity to Patient Demographic that holds Age and Gender and Ethnicity information. The Insurance Provider entity links patients to both Provider Name and INSURANCER for recording their insurance details.

A Patient Encounter stands as a fundamental database entity that records both patient sessions along with their healthcare contacts. An EncounterID uniquely identifies each visit that contains both its visit date and its purpose. Symptoms experienced by patients find their place in the Symptoms entity through SymptomID followed by Description attributes. During an encounter medical decisions such as Diagnosis along with Lab Results emerge (1:1 relationship) whereas both produce LabResultID along with Test Type and Result data. The system generates prescriptions through diagnosis records and stores them in the Prescription entity while associated Medication Details contains information about Medication Name alongside defined Dosage.

The system maintains patient vital health indicator records through the VitalSigns entity that tracks HeartRate and Temperature measurements. The Vaccination entity manages vaccination records by keeping track of VaccinationID along with Status. The Billing entity ties to patient encounters for recording financial information through BillingID as well as TotalAmount and PaymentStatus. The HealthcareProvider entity handles medical staff involved in patient visits by storing Name and Specialization attributes as part of its data structure. The controlled schema facilitates proper tracking of patient documents together with health services and historical health data.

UML Diagram


-
- PATIENTID: Integer{frozen}
- Name: String
- Address: String
- Phone: String
- SSN: String
+ getDetails(): String
+ updateInfo(name: string, address: string, phone: string): void
PatientDemographics
- PATIENTID: Integer{frozen}
- Age: Integer
- Gender: String
- Ethnicity: String
+ getDemographics(): string
PatientEncounter
- PATIENTID: Integer{frozen}
- EncounterID: Integer
- Date: Date
- Reason: String
+ getEncounterDetails(): string
+ updateReason(newReason: string): void
InsuranceProvider
- InsuranceID: Integer{frozen}
- PatientID: Integer
- ProviderName: String
+ getProviderInfo(): string
+ updateProvider(providerName: string): void
Vaccination
- VaccinationID: Integer{frozen}
- PatientID: Integer
- VaccineName: String
- Staatus: String
+ getVaccinationStatus(): string
+ updateStatus(newStatus: string): void
EPOC
- PATIENTID: Integer{frozen}
- EPOCId: Integer
- ContactName: String
- Phone: String
+ getContactInfo(): string
+ updateContact(name: string, phone: string): void
Billing
- BillingID: Integer{frozen}
- EncounterID: Integer
- TotalAmount: Float
- PaymentStatus: String
+ calculateBill(): float
+ updatePaymentStatus(newStatus: string): void
VitalSigns
- VitalSignID: Integer{frozen}
- EncounterID: Integer
- Temperature: Float
- HeartRate: Int
+ getVitalSigns(): string
+ updateVitals(temp: float, heartRate: int): void
Symptoms
- SymptomID: Integer{frozen}
- EncounterID: Integer
- Description: String
+ getSymptoms(): string
+ addSymptom(description: string): void
LabResults
- LabResultID: Integer{frozen}
- EncounterID: Integer
- TestType: String
- Result: String
+ getLabResults(): string
+ updateResult(newResult: string): void
HealthcareProvider
- HealthcareProviderID: Integer{frozen}
- Name: String
- Specialization: String
+ getProviderDetails(): string
+ assignToEncounter(encounterID: int): void
Prescription
- PrescriptionID: Integer{frozen}
- EncounterID: Integer
- MedicationID: Integer
+ prescribeMedication(medicationID: int): void
+ getPrescriptionDetails(): string
Diagnosis
- DiagnosisID: Integer{frozen}
- EncounterID: Integer
- HealthcareProviderID: Integer
+ getDiagnosisDetails(): string
+ updateDiagnosis(newDiagnosis: string): void
MedicationDetails
- MedicationID: Integer{frozen}
- MedicationName: String
- Dosage: String
+ getMedicationInfo(): string
+ updateDosage(newDosage: string): void

Through UML (Unified Modeling Language) the class diagram depicts a patient health management system by showing how system entities connect across the platform. The Patient class stands as the central point of the diagram where it maintains attributes including PATIENTID as well as Name, Address, Phone and SSN. The system contains two operations one for detail access (getDetails()) and another for contact information modification. The Patient class links to several other entities through relationships to store critical healthcare documentation information.

The PatientEncounter class exists in tight connection with Patient data to document all medical visits. The EncounterID along with Date and Reason attributes form the essence of this class which provides the getEncounterDetails() and visit information update capabilities. The Symptoms class maintains a relationship with PatientEncounter for tracking symptom details through retrieval and update capabilities. The Patient class contains VitalSigns as its sub-class to store measurements including HeartRate and Temperature which utilizes methods for read and write operations.

The system has Diagnosis functions that connect to PatientEncounter. The DiagnosisID along with EncounterID let the system retrieve and modify diagnosis data within this class. PatientEncounter maintains an association with the Prescription class through which PrescriptionID pairs with MedicationID for storing prescribed medications while providing prescription methods. Using attributes from MedicationDetails together with those of the extended class enables comprehensive record-keeping of prescribed treatments.

PatientEncounter utilizes the LabResults function as an integrated component. LabResultID and TestType and Result are its attributes along with methods that enable both data retrieval and outcome updates. The system links PatientEncounter to Billing through which financial transactions can be tracked by registering BillingID, TotalAmount, PaymentStatus and other attributes. Methods allow billing status updates.

A HealthcareProvider class provides complete patient care features by storing information about HealthProviderID, Name, and Specialization alongside methods to receive provider details and link them to encounters. InsuranceProvider stands as a different class which connects to Patient through InsuranceID along with ProviderName to enable insurance data management.

The system includes various other entities such as Vaccination for storing immunization histories and EPOC which maintains patient emergency contacts. The system operates as a structured and functional unit because each class contains methods that specifically manage data retrieval and update operations. A UML class diagram effectively structures a healthcare system through its capability to unite medical documents with patient relations and prescription information and billing processes.