MediCore

HOSPITAL MANAGEMENT SYSTEM

Tanisha Sriram – 3122235001144
V.Yadushree - 3122235001144

Problem Statement

Managing patient, doctor, treatment, and billing information in hospitals is often complex and prone to errors when done manually. To improve efficiency and accuracy, a centralized digital Hospital Management System is needed.

This project aims to design and implement a relational database that:

- Stores and manages patient, doctor, intern, treatment, and billing details.
- Maintains accurate medical histories and medication records.
- Handles room and department allocations efficiently.
- Uses proper normalization to avoid redundancy and ensure data consistency.

The system will streamline hospital operations and provide reliable, organized data management.

Relations with their attributes

Patient

- PatientID (Primary Key)
- First_Name
- Last_Name
- MedicalHistory
- Contact
- DOB
- RoomID (Foreign Key)

Room

- RoomID (Primary Key)
- Type
- Capacity

Bill

- PatientID (Primary Key, Foreign Key)
- BillNumber
- Amount
- DateIssued

Medication

- MedID (Primary Key)
- Name
- Dosage
- SideEffects

Patient_Medication

- PatientID (Primary Key, Foreign Key)

- MedID (Primary Key, Foreign Key)

Doctor

- DoctorID (Primary Key)
- Name
- Specialty
- InternID (Foreign Key)

Patient_Doctor

- PatientID (Primary Key, Foreign Key)
- DoctorID (Primary Key, Foreign Key)

Treatment

- TreatmentID (Primary Key)
- TreatmentType
- Cost

Patient_Treatment

- PatientID (Primary Key, Foreign Key)
- TreatmentID (Primary Key, Foreign Key)

Doctor_Treatment

- DoctorID (Primary Key, Foreign Key)
- TreatmentID (Primary Key, Foreign Key)

Intern

- InternID (Primary Key)
- Name
- College
- Year

Treatment_Intern

- TreatmentID (Primary Key, Foreign Key)

- InternID (Primary Key, Foreign Key)

Doctor_Medication

- DoctorID (Primary Key, Foreign Key)
- MedID (Primary Key, Foreign Key)

Department

- DeptID (Primary Key)
- DeptName
- ManagedBy (Foreign Key referencing DoctorID)

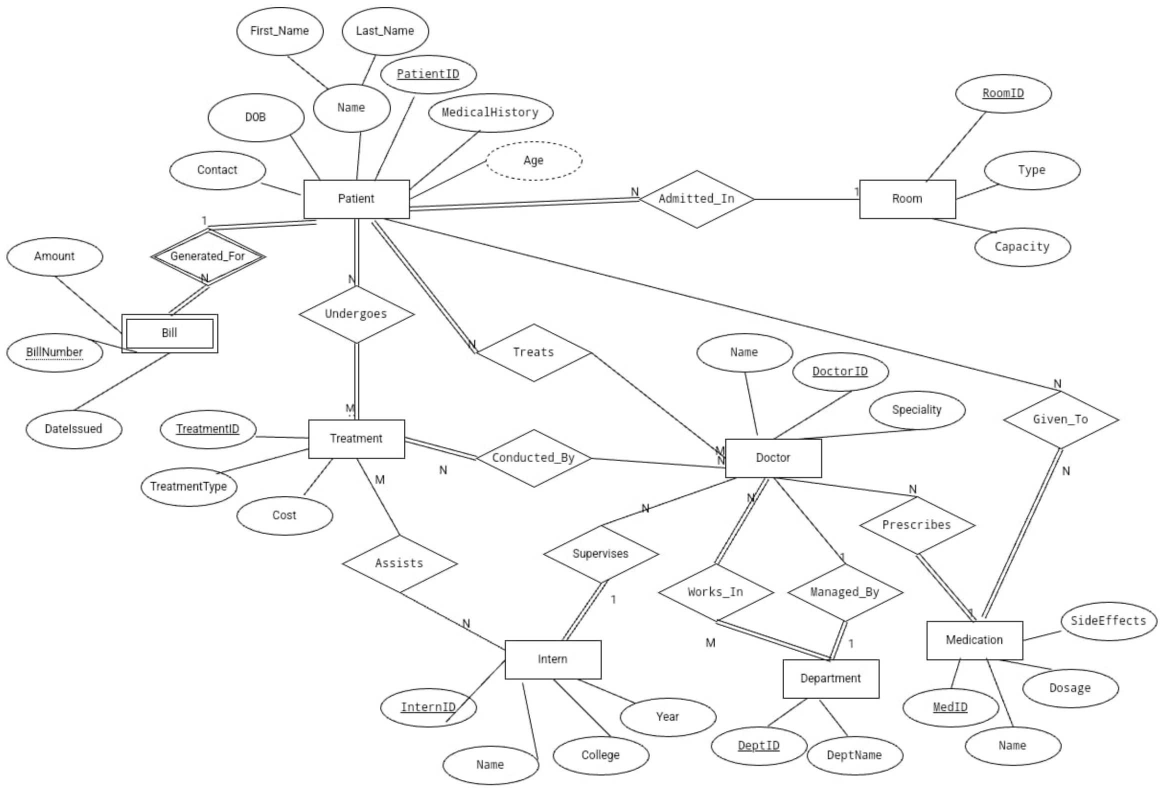
Works_In

- DoctorID (Primary Key, Foreign Key)
- DeptID (Primary Key, Foreign Key)

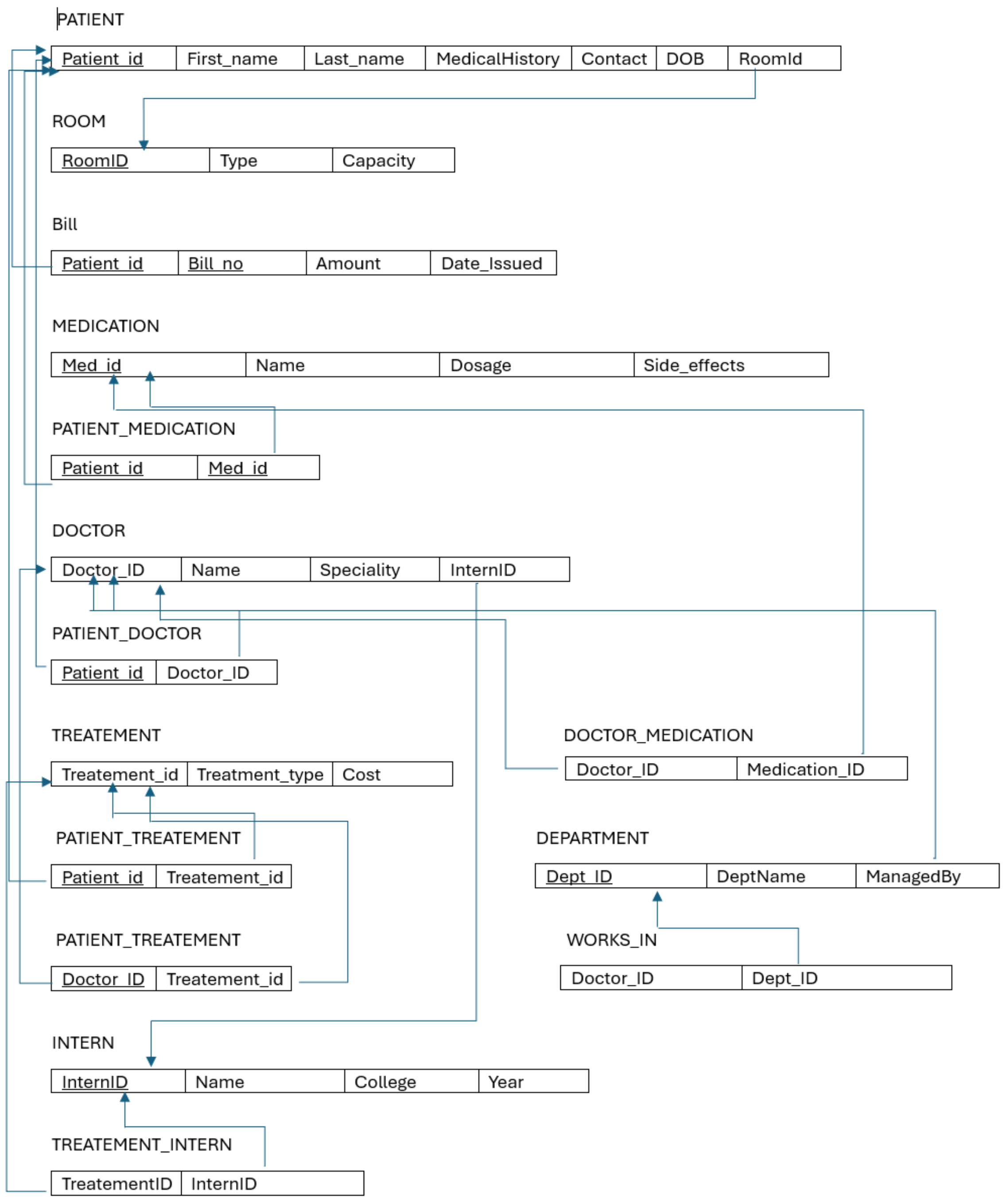
Relationships

- **Patient** admitted in **Room** (Many-to-One)
- **Patient** undergoes **Treatment** (Many-to-Many)
- **Treatment** conducted by **Doctor** (Many-to-Many)
- **Intern** assists **Treatment** (Many-to-Many)
- **Doctor** supervises **Intern** (One-to-Many)
- **Doctor** works in **Department** (Many-to-One)
- **Doctor** prescribes **Medication** (Many-to-Many)
- **Medication** given to **Patient** (Many-to-Many)
- **Bill** generated for **Patient** (One-to-One)

ER Diagram


First_Name
Last_Name
PatientID
Name
DOB
MedicalHistory
Contact
Age
Patient
Admitted_In
Room
RoomID
Type
Capacity
Amount
Generated_For
Bill
BillNumber
DateIssued
Undergoes
Treats
TreatmentID
Treatment
TreatmentType
Cost
Conducted_By
Name
DoctorID
Speciality
Doctor
Given_To
Assists
Supervises
Works_In
Managed_By
Prescribes
Medication
SideEffects
Dosage
MedID
Name
Intern
InternID
Name
College
Year
Department
DeptID
DeptName

ER to Relational Mapping


PATIENT
Patient_id | First_name | Last_name | MedicalHistory | Contact | DOB | RoomId
ROOM
RoomID | Type | Capacity
Bill
Patient_id | Bill_no | Amount | Date_Issued
MEDICATION
Med_id | Name | Dosage | Side_effects
PATIENT_MEDICATION
Patient_id | Med_id
DOCTOR
Doctor_ID | Name | Speciality | InternID
PATIENT_DOCTOR
Patient_id | Doctor_ID
TREATEMENT
Treatement_id | Treatment_type | Cost
PATIENT_TREATEMENT
Patient_id | Treatement_id
PATIENT_TREATEMENT
Doctor_ID | Treatement_id
INTERN
InternID | Name | College | Year
TREATEMENT_INTERN
TreatementID | InternID
DOCTOR_MEDICATION
Doctor_ID | Medication_ID
DEPARTMENT
Dept_ID | DeptName | ManagedBy
WORKS_IN
Doctor_ID | Dept_ID

Functional Dependencies

PATIENT

- Patient_id → First_name, Last_name, MedicalHistory, Contact, DOB, RoomId

ROOM

- RoomID → Type, Capacity

BILL

- Bill_no → Patient_id, Amount, Date_Issued (assuming Bill_no is unique)

MEDICATION

- Med_id → Name, Dosage, Side_effects

PATIENT_MEDICATION

- (Patient_id, Med_id) → (no extra attributes; composite PK)

DOCTOR

- Doctor_ID → Name, Speciality
- InternID → Doctor_ID (possible issue — if an Intern works under one doctor only)

PATIENT_DOCTOR

- (Patient_id, Doctor_ID) → (composite relationship; no extra attributes)

TREATEMENT

- Treatement_id → Treatment_type, Cost

PATIENT_TREATEMENT

- (Patient_id, Treatement_id) → (composite relationship; no extra attributes)

DOCTOR_TREATEMENT

- (Doctor_ID, Treatement_id) → (composite relationship; no extra attributes)

INTERN

- InternID → Name, College, Year

TREATEMENT_INTERN

- (TreatementID, InternID) → (composite relationship; no extra attributes)

DOCTOR_MEDICATION

- (Doctor_ID, Medication_ID) → (composite relationship; no extra attributes)

DEPARTMENT

- Dept_ID → DeptName, ManagedBy

WORKS_IN

- (Doctor_ID, Dept_ID) → (composite relationship; no extra attributes)

Normalization

1NF (First Normal Form) - YES

Atomic values: All fields contain atomic values (like First_name, Last_name, Contact, RoomId, etc.) — no repeating groups or arrays.

Unique rows: Each table has a primary key that ensures uniqueness (like Patient_id, RoomID, Doctor_ID, etc.).

2NF (Second Normal Form) - YES

- **It's in 1NF**.
- **No partial dependencies:**
  - Composite tables like PATIENT_TREATEMENT use a combination of Patient_id and Treatement_id as foreign keys, and no non-prime attribute depends on part of a composite key.
  - All non-key attributes in every table fully depend on the entire primary key.

3NF (Third Normal Form) - YES

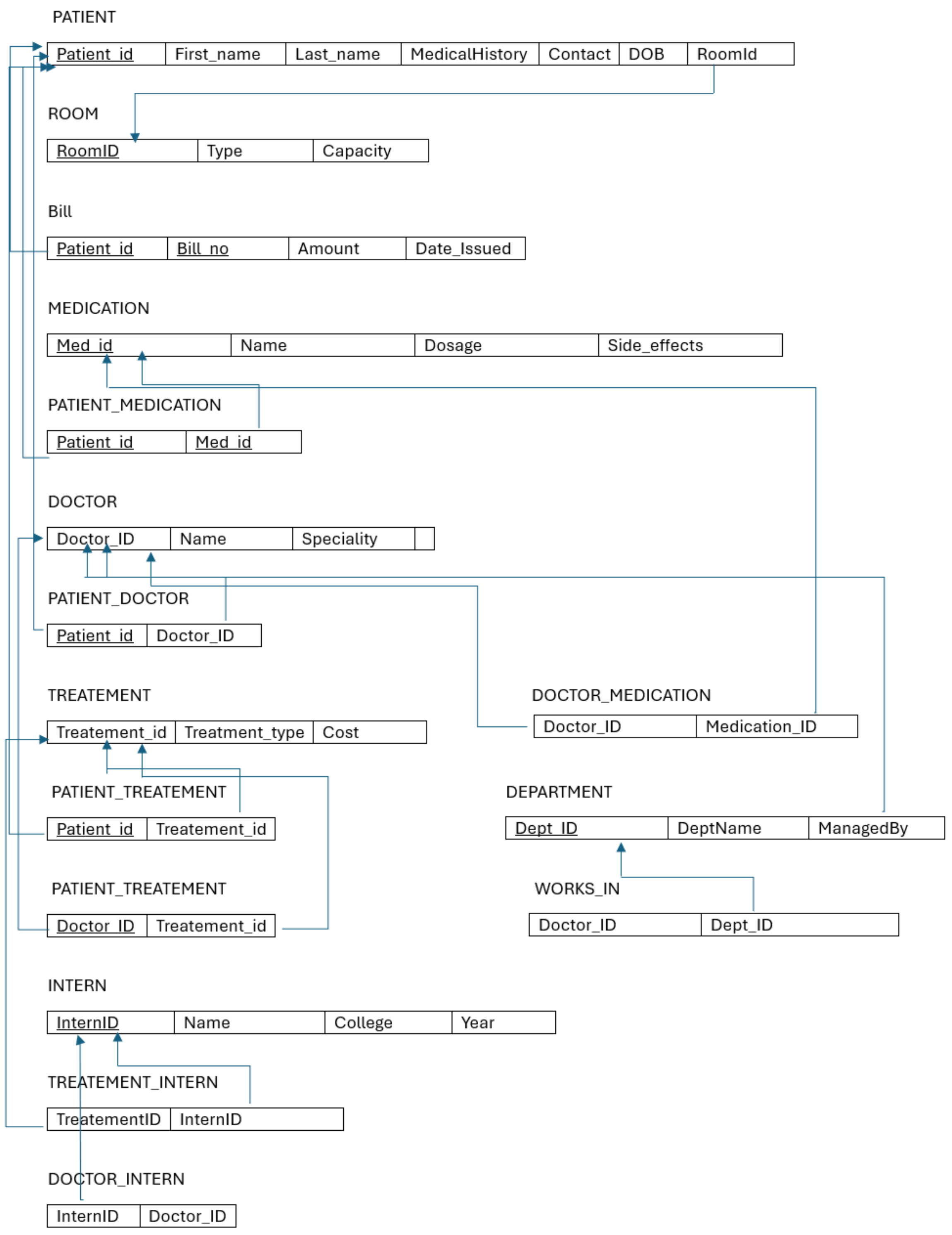
- **It's in 2NF**.
- **No transitive dependencies:**
  - Non-key attributes (like Speciality, DeptName, Amount, Date_Issued) depend only on the primary key.
  - For example:
    - Doctor table: Speciality depends only on Doctor_ID.
    - Bill table: Amount and Date_Issued depend only on the composite key of (Patient_id, Bill_no).

BCNF – NO

If InternID → Doctor_ID
then decompose:

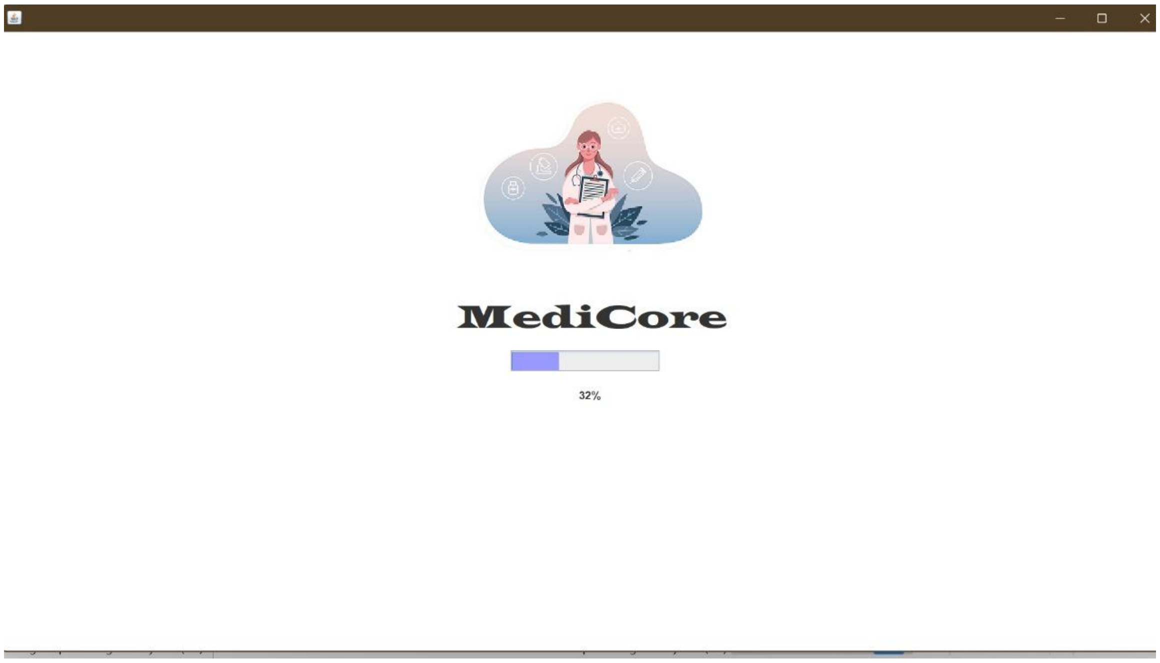
- DOCTOR(Doctor_ID, Name, Speciality)
- INTERN_DOCTOR(InternID, Doctor_ID)

Final Schema

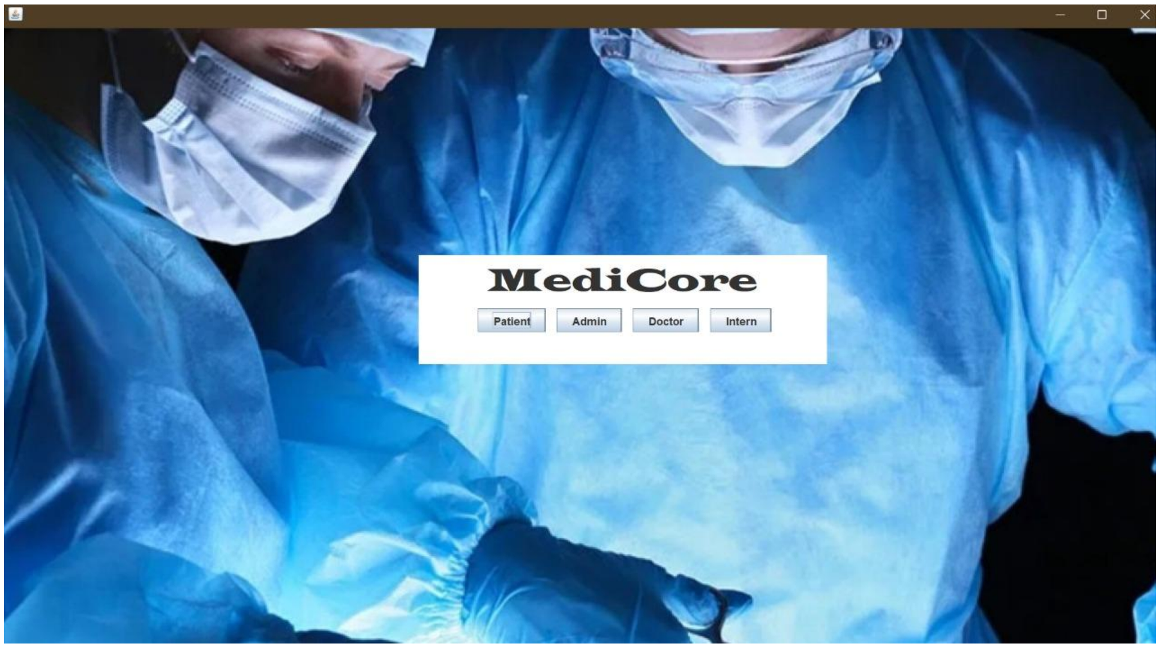

PATIENT
Patient_id | First_name | Last_name | MedicalHistory | Contact | DOB | RoomId
ROOM
RoomID | Type | Capacity
Bill
Patient_id | Bill_no | Amount | Date_Issued
MEDICATION
Med_id | Name | Dosage | Side_effects
PATIENT_MEDICATION
Patient_id | Med_id
DOCTOR
Doctor_ID | Name | Speciality
PATIENT_DOCTOR
Patient_id | Doctor_ID
TREATEMENT
Treatement_id | Treatment_type | Cost
DOCTOR_MEDICATION
Doctor_ID | Medication_ID
PATIENT_TREATEMENT
Patient_id | Treatement_id
DEPARTMENT
Dept_ID | DeptName | ManagedBy
PATIENT_TREATEMENT
Doctor_ID | Treatement_id
WORKS_IN
Doctor_ID | Dept_ID
INTERN
InternID | Name | College | Year
TREATEMENT_INTERN
TreatementID | InternID
DOCTOR_INTERN
InternID | Doctor_ID

Sample Screenshots

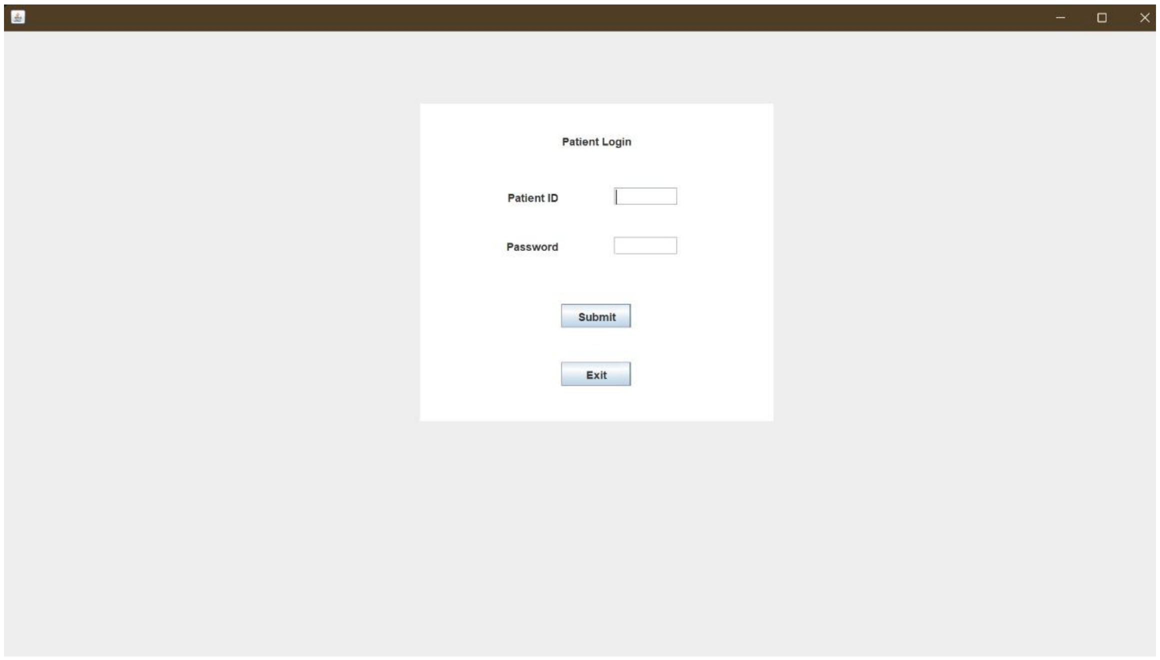
Loading Screen



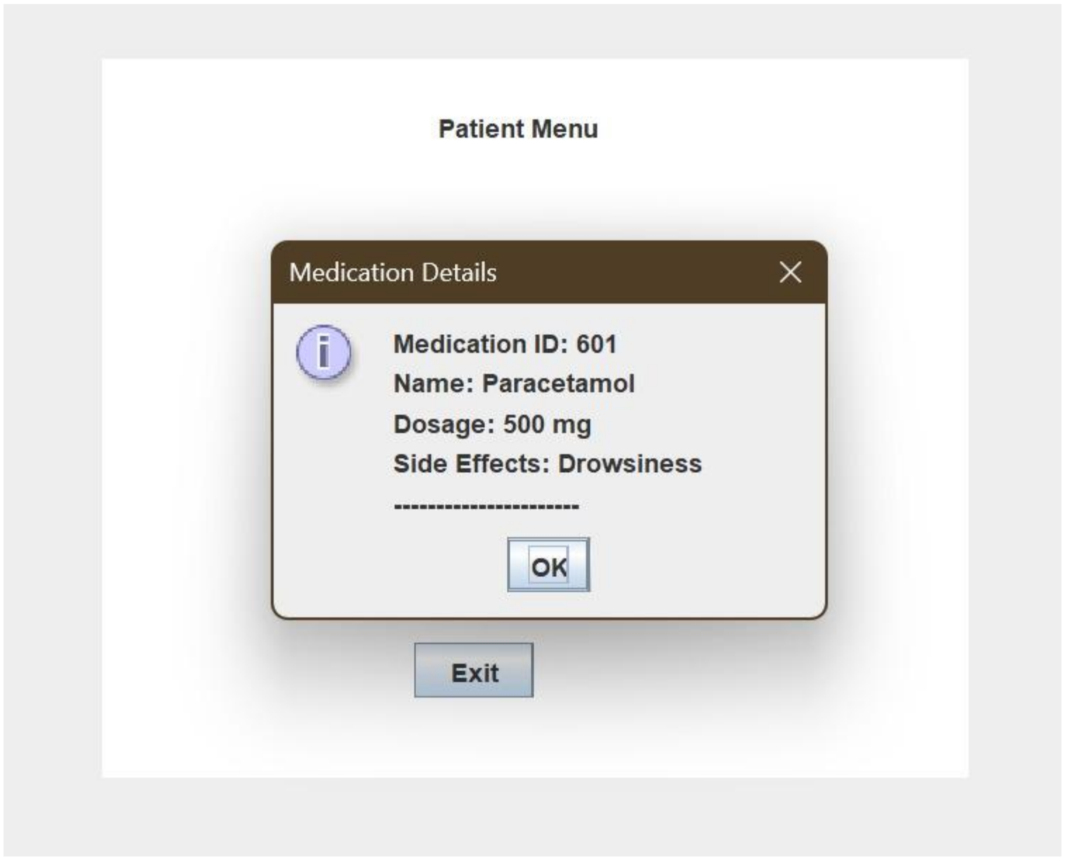
Menu



Login Menus

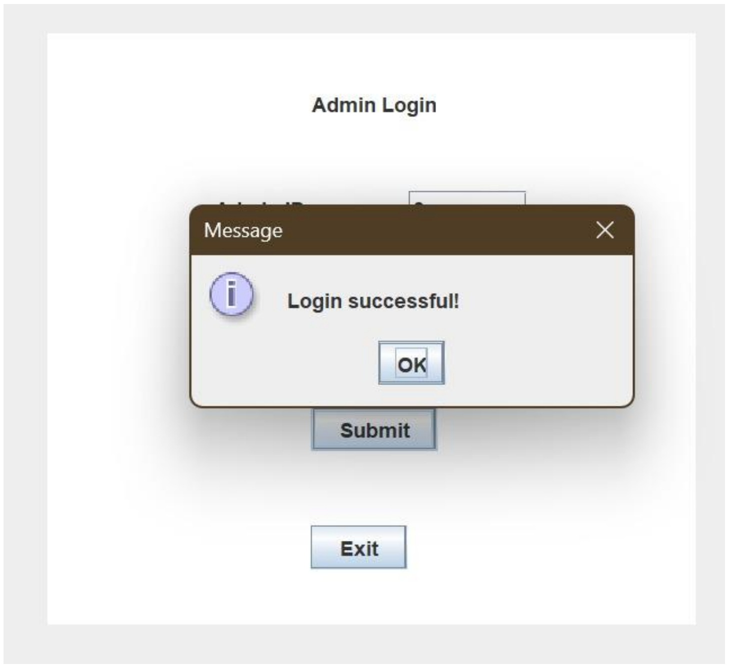


Pop-Up Boxes for details



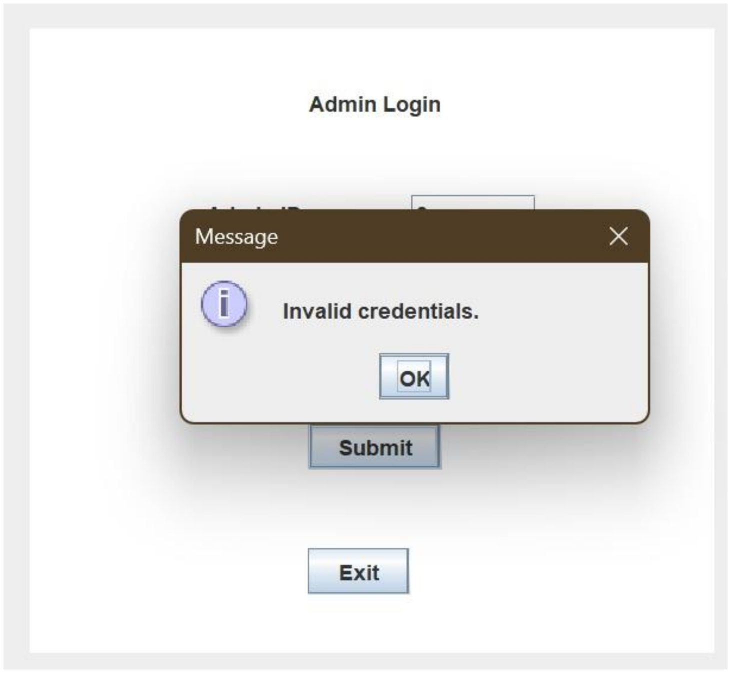
Medication ID: 601
Name: Paracetamol
Dosage: 500 mg
Side Effects: Drowsiness

Login Successful



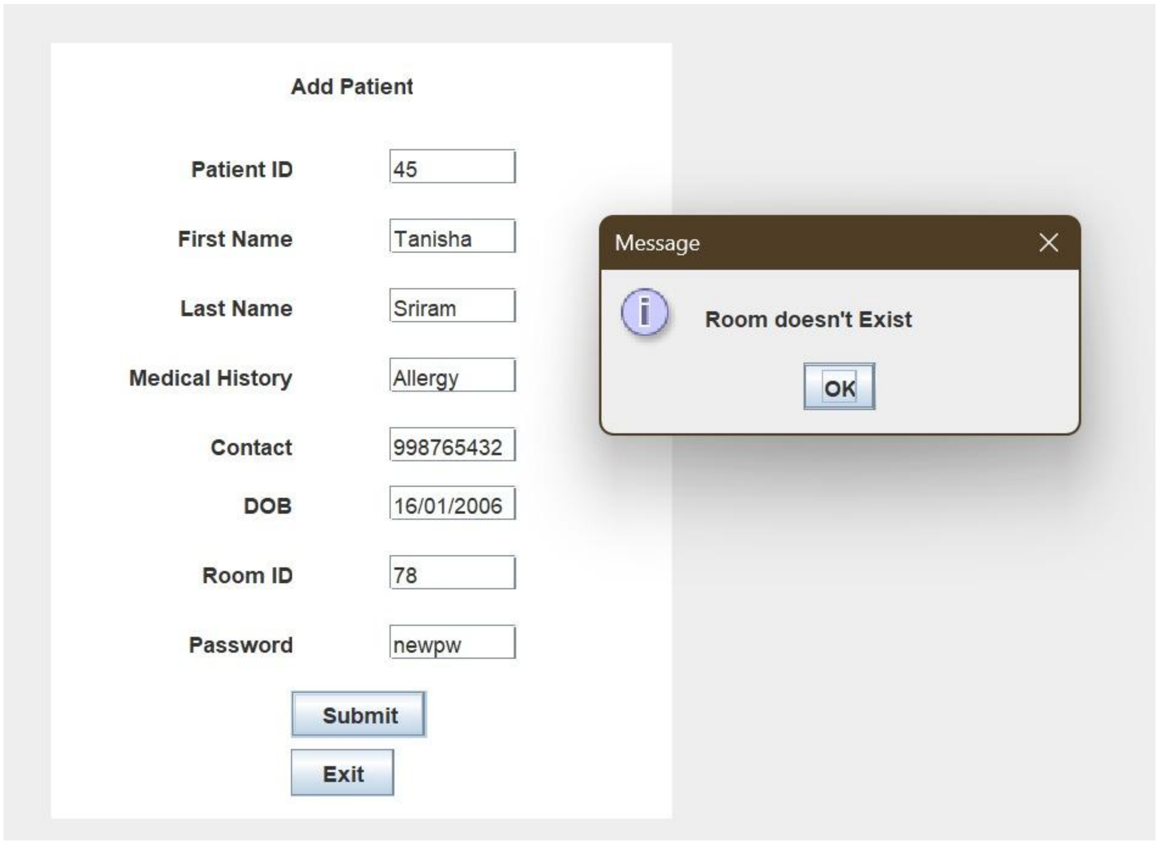
Login successful!

Invalid Credentials



Invalid credentials.

Foreign key violations



Add Patient

Patient ID	45
First Name	Tanisha
Last Name	Sriram
Medical History	Allergy
Contact	998765432
DOB	16/01/2006
Room ID	78
Password	newpw

Room doesn't Exist

Learning Outcomes

- Understood the concept of functional dependencies and how they define relationships between attributes in a relational database schema.
- Practiced identifying the normal form of a relational schema using functional dependencies and learned to verify compliance with 1NF, 2NF, 3NF, and BCNF.
- Gained practical experience in applying normalization techniques step-by-step to eliminate data redundancy and anomalies.
- Learned to decompose relations while ensuring lossless join and dependency preservation in the database schema.
- Understood the importance of atomicity, data integrity, and clean design principles in real-world database applications.
- Strengthened skills in ER modeling, converting ER diagrams into structured relational schemas, and mapping entity and relationship sets appropriately.
- Worked on many-to-many and one-to-many relationship handling using composite relations and normalization.
- Developed a fully functional Hospital Management System database using Oracle Database for relational data storage and management.
- Gained hands-on experience with NetBeans IDE for developing and integrating database operations using SQL queries within a Java application environment.
- Learned to connect Java applications to Oracle databases through JDBC, ensuring seamless interaction between the front-end interface and the back-end database.