AMRITA
VISHWA VIDYAPEETHAM

DATABASE MANAGEMENT SYSTEMS LAB

22CSC381

PATIENT MANAGEMENT SYSTEM

REVIEW - 1

INSTRUCTOR: Dr. JEYAKUMAR G

SRIBALAKRISHNAN P – CB.PS.I5DAS22148

Table of Contents

CHAPTER 1

ABSTRACT

OVERVIEW:

The Patient Management System(PMS) is designed to simplify and automate various hospital related activities, reducing the need for manual process and paperwork. This system optimizes patient data management, enhancing workflow efficiency and streamlining resource utilization.

AIM:

This project aims to implement a reliable and user-friendly system that reduces errors, improves access to data, and ensures timely dissemination of important medical information to patients and healthcare professionals.

CHAPTER 2

KEY OPERATIONS:

1. Patient Registration: Patients can easily register by providing essential details such as name, age, medical history, and insurance information. This data is stored in the Patient Information table, ensuring a comprehensive record for future reference.
2. Administrative Management: Administrators utilize an intuitive interface to manage patient records, including generating and issuing prescriptions. These records, along with administrative credentials and activities, are stored in the Administrator Details table. Prescriptions are automatically generated and emailed to patients in PDF format for easy access and download.
3. Patient Updates: Patients have the capability to update or modify their personal and medical information as needed. This functionality is supported by the Patient Information table, which maintains up-to-date records of all changes.

4. Appointment Scheduling: The system allows for efficient scheduling of appointments between patients and healthcare professionals. Appointment details, including date, time, and consultation purpose, are recorded in the Appointment Details table.
5. Treatment Tracking: The system tracks ongoing treatments and procedures for each patient, recording relevant details in the Treatment Records table. This helps in monitoring patient progress and ensuring continuity of care.
6. Prescription Management: Prescriptions are automatically created based on patient needs and are logged in the Prescription Records table. Patients receive their prescriptions via email with a downloadable PDF link.
7. Billing and Payments: The system manages billing for medicines and services, recording transactions and payment details in the Bill Details table. This ensures accurate financial tracking and easy access to billing history.
8. Lab Test Management: Lab tests and their results are recorded in the Lab Test Results table. This allows for efficient management of test orders, results, and follow-up actions.
9. Room Allocation: Information about room availability, status, and allocation is managed in the Room Accommodation table. This helps in efficiently utilizing hospital resources and managing patient room assignments.
10. Insurance Processing: The system maintains records of patients' insurance providers and coverage details in the Insurance Records table. This facilitates seamless processing of insurance claims and verification.
11. Medical Records Management: Comprehensive medical records for each patient are stored in the Medical Records table, ensuring that all relevant medical information is readily available for healthcare providers.
12. Data Security and Access Control: The system implements robust security measures to safeguard patient data and control access based on user roles. Separate logins for administrators and patients ensure secure and appropriate access to information.

13. Reporting and Analytics: The system generates reports on various aspects such as patient demographics, treatment outcomes, and billing summaries. These reports aid in decision-making and strategic planning.
14. Communication Facilitation: The system provides tools for effective communication between medical staff and patients, including notification of appointments, reminders for upcoming treatments, and alerts for critical medical information.
15. System Maintenance and Support: The system includes features for routine maintenance and support, ensuring smooth operation and addressing any technical issues promptly.

TABLES:

1. Patient Information: Stores personal and medical history details.
2. Doctor Details: Stores information about assigned doctors.
3. Prescription Records: Logs all prescriptions issued to patients.
4. Administrator Details: Records admin credentials and their activities.
5. Appointment Details: Stores information of all the appointments fixed.
6. Bill Details: Records for all medicines sold and billing their billing details.
7. Lab test: Stores information all the tests and taken in the lab with their results.
8. Room Accomodation: Stores information about the rooms and their status.

OUTPUT:

1. Detailed prescription information, including a downloadable PDF link sent to patients via email.
2. Comprehensive medical history and treatment reports for patient reference.
3. Information on assigned doctors and consultation details.
4. Accessible prescription download links for patient convenience.
5. Billing summaries and payment records for accurate financial tracking.
6. Lab test results and related information for follow-up actions.
7. Appointment schedules and reminders for patients and healthcare providers.
8. Room allocation details, including availability and status updates.
9. Insurance coverage summaries and claims processing information.

10. Patient progress reports and treatment updates for healthcare professionals.
11. Analytics reports on patient demographics, treatment outcomes, and system usage.
12. Communication logs, including notifications, alerts, and reminders.
13. System activity reports for administrative oversight and performance monitoring.
14. Security audit logs to track access and ensure data integrity.

CHAPTER 3

PROJECT PREVIEW:

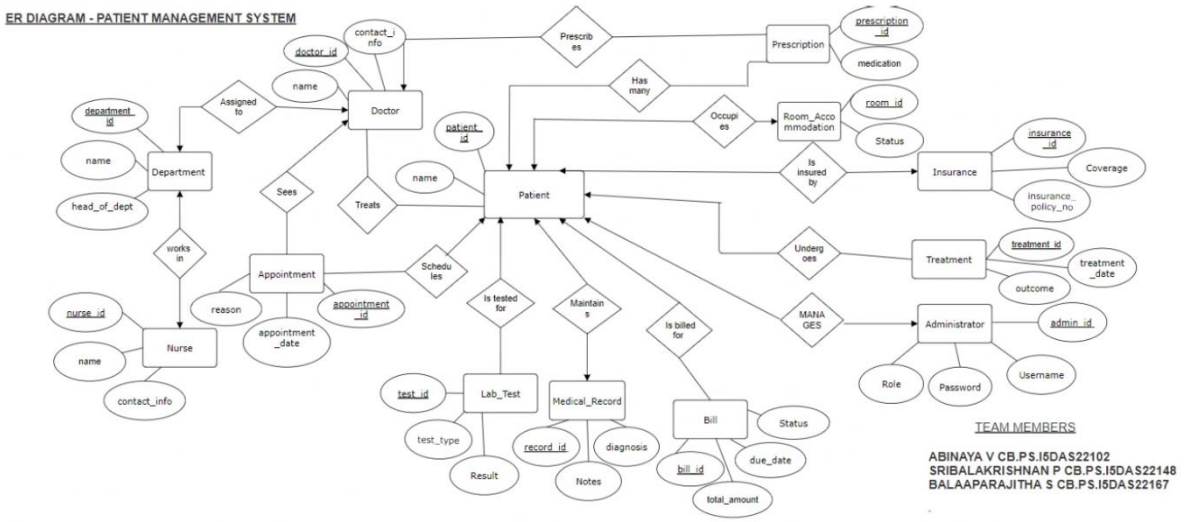
a. Integration of telemedicine features for remote consultations.
b. Mobile application for patient access and notifications.
c. Advanced analytics using machine learning for predictive healthcare insights.

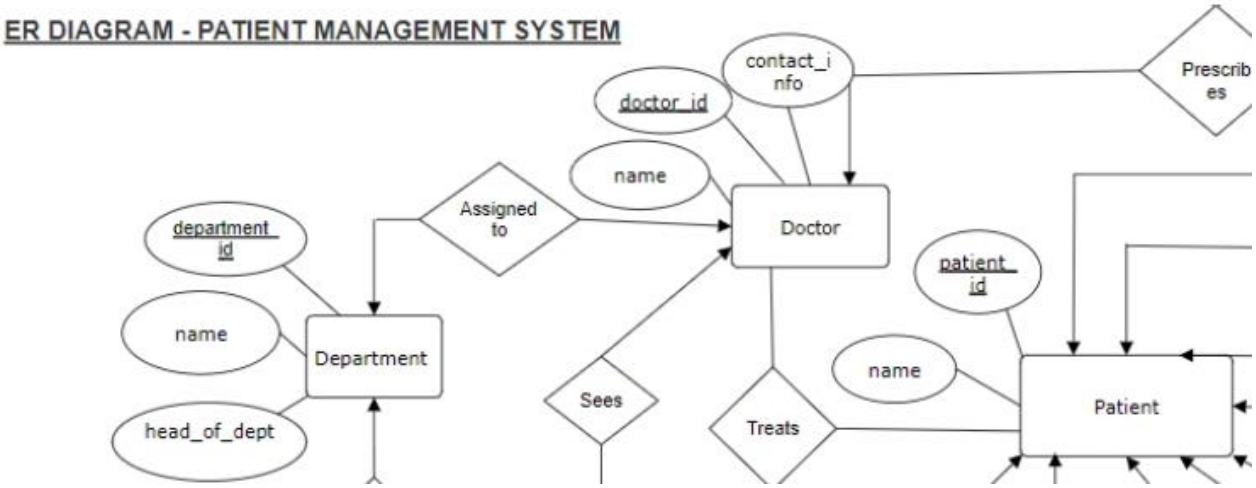
Tools Used: Streamlit, SQlite, pandas, numpy, hashlib, plotly

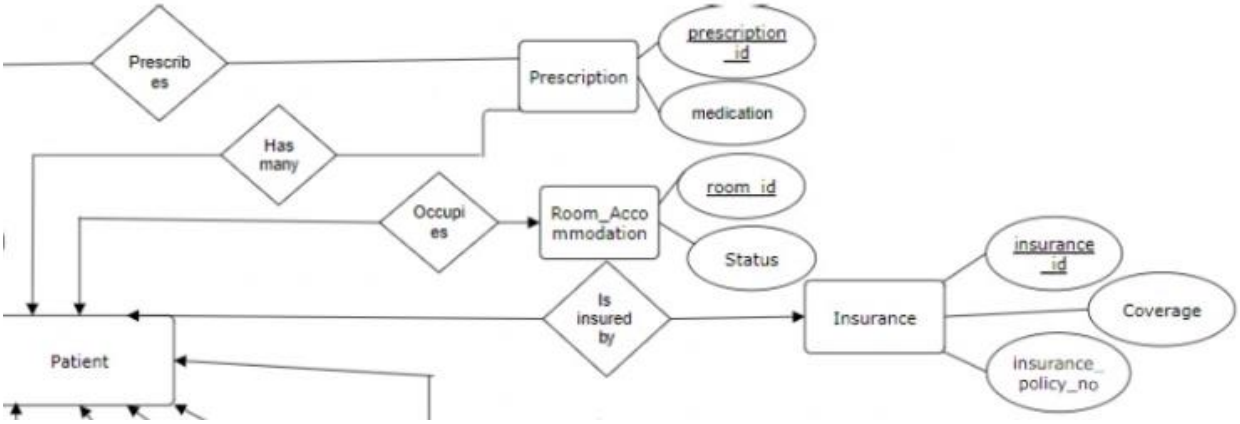
---

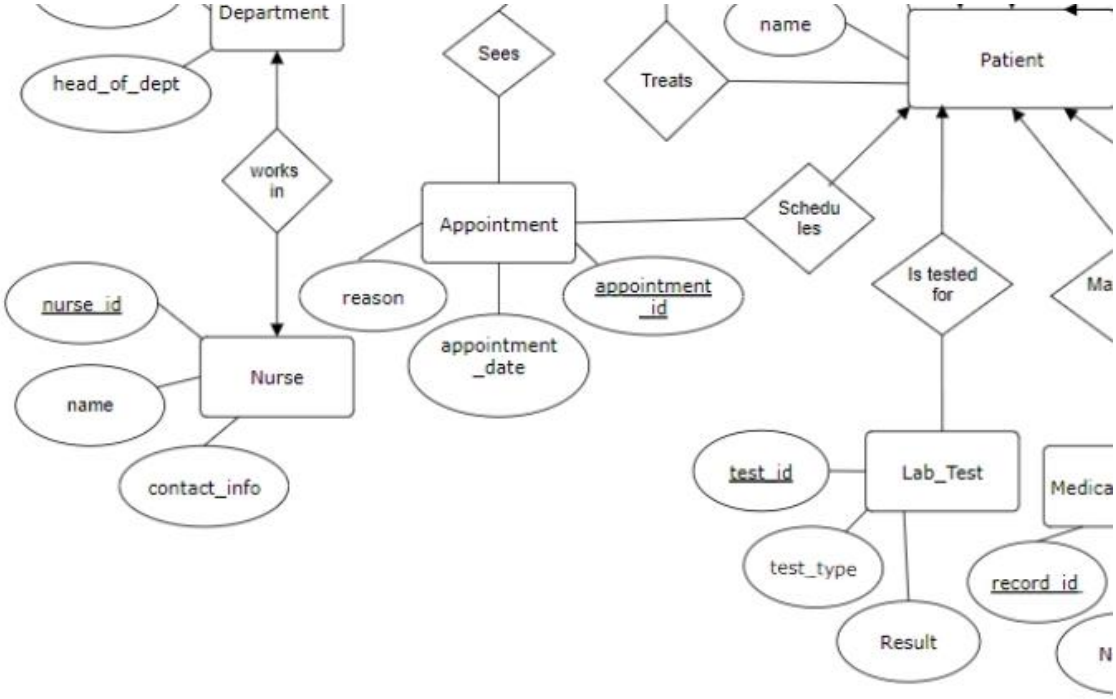
CHAPTER 4

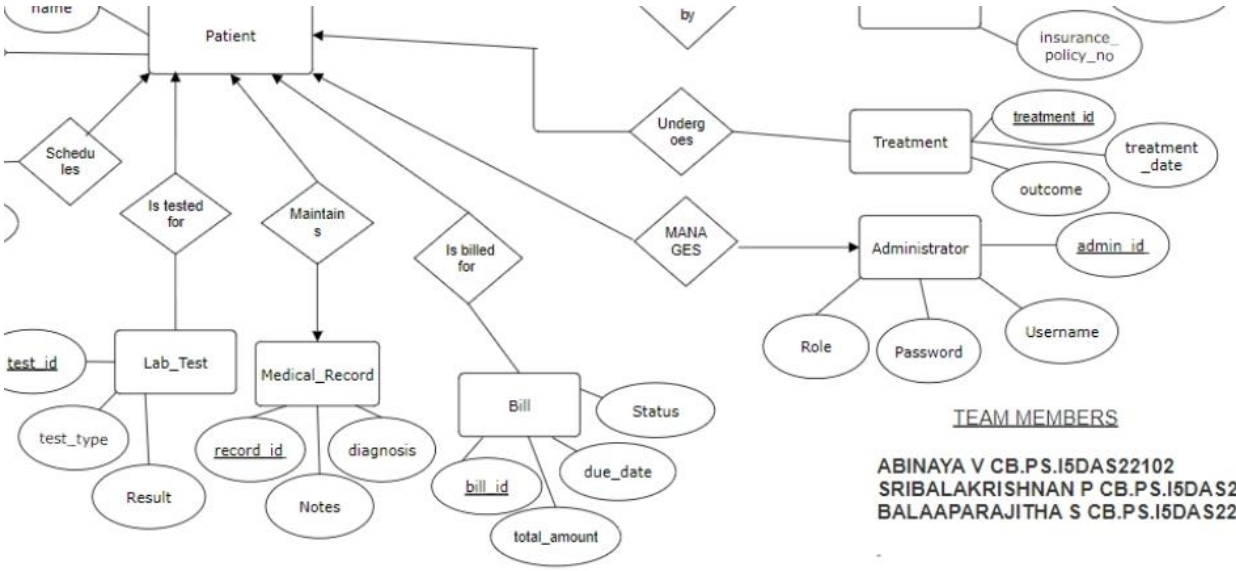
PROJECT DESIGN:







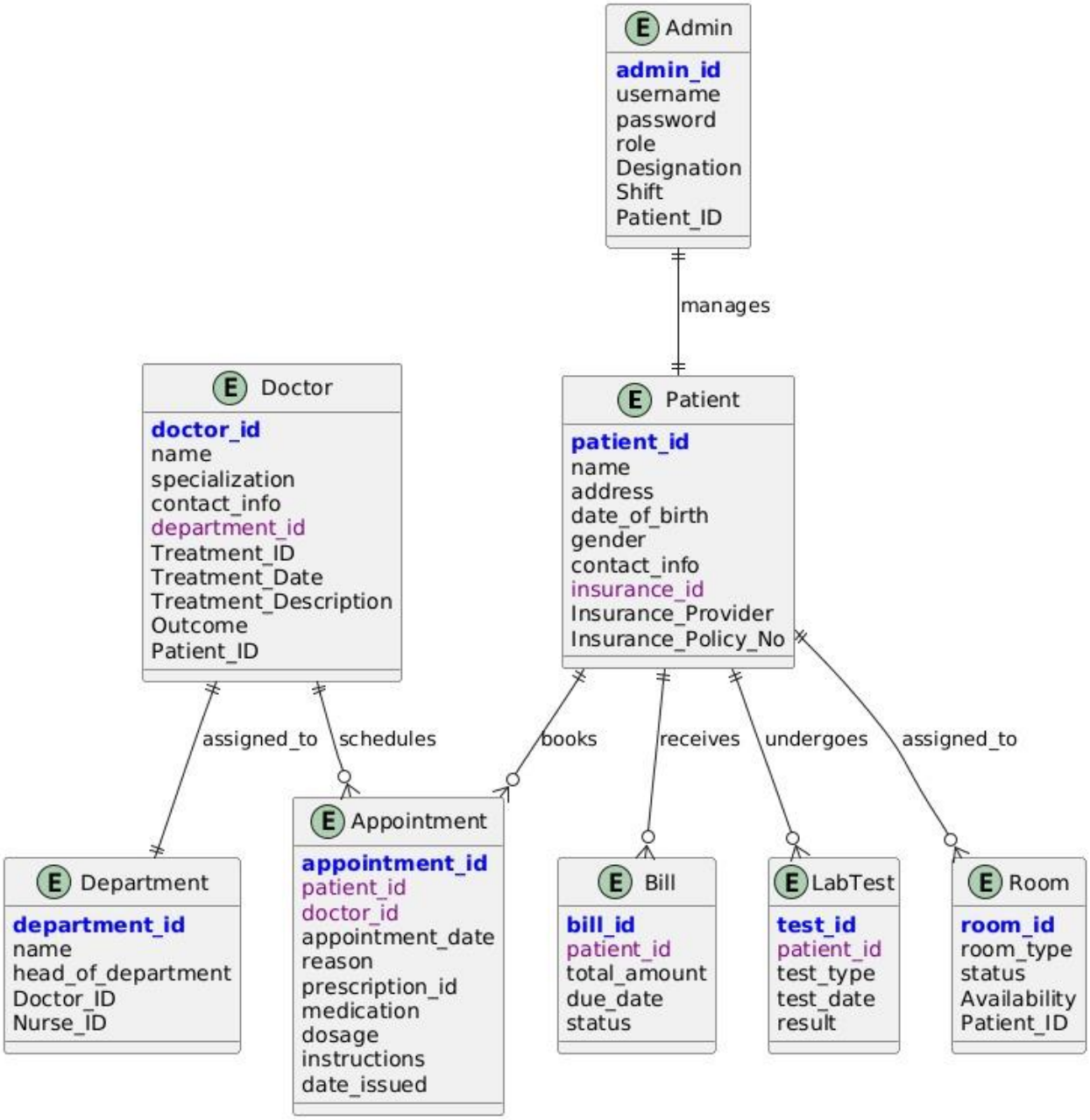




TEAM MEMBERS

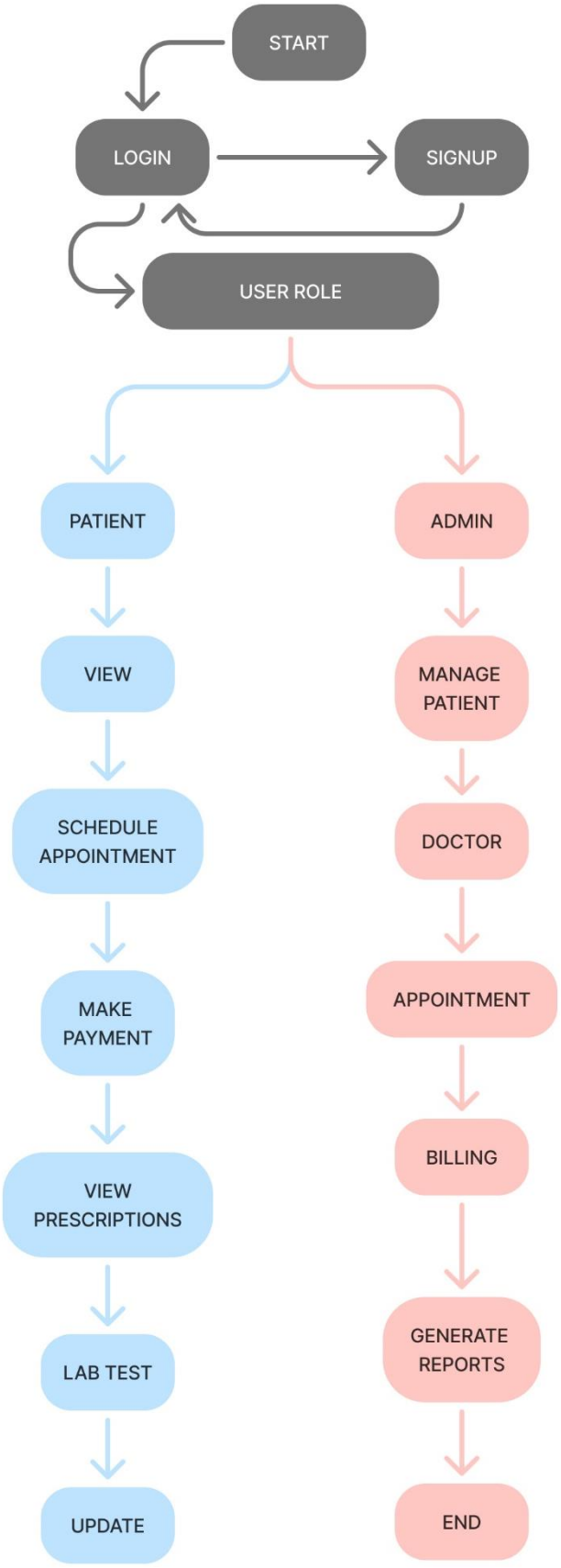
ABINAYA V CB.PS.I5DAS22102
SRIBALAKRISHNAN P CB.PS.I5DAS22148
BALAAPARAJITHA S CB.PS.I5DAS22167

SCHEMA:


Admin
admin_id
username
password
role
Designation
Shift
Patient_ID
manages
Doctor
doctor_id
name
specialization
contact_info
department_id
Treatment_ID
Treatment_Date
Treatment_Description
Outcome
Patient_ID
Patient
patient_id
name
address
date_of_birth
gender
contact_info
insurance_id
Insurance_Provider
Insurance_Policy_No
assigned_to
schedules
books
receives
undergoes
assigned_to
Department
department_id
name
head_of_department
Doctor_ID
Nurse_ID
Appointment
appointment_id
patient_id
doctor_id
appointment_date
reason
prescription_id
medication
dosage
instructions
date_issued
Bill
bill_id
patient_id
total_amount
due_date
status
LabTest
test_id
patient_id
test_type
test_date
result
Room
room_id
room_type
status
Availability
Patient_ID

CHAPTER 5

PROJECT ANALYSIS:



CHAPTER 6

NORMALIZATION:

1. Patient Table:

First Normal Form (1NF):

- Ensure atomic values: All columns have atomic values.
- The table is in 1NF.

Second Normal Form (2NF):

- Partial Dependency: Patient_ID is the primary key for patient information, while Insurance_ID is the primary key for insurance details.
- Split into two tables:
    - Patient Table:
        - Patient_ID, Patient_Name, DOB, Gender, Address, Contact_Info
    - Insurance Table:
        - Insurance_ID, Insurance_Provider, Insurance_Policy_No, Patient_ID (Foreign Key)

Third Normal Form (3NF):

- No transitive dependencies. Both tables are in 3NF.

Boyce-Codd Normal Form (BCNF):

- Both tables satisfy BCNF. The candidate keys function properly as superkeys.

2. Doctor Table:

First Normal Form (1NF):

- Ensure atomic values: All columns have atomic values.
- The table is in 1NF.

Second Normal Form (2NF):

- Partial Dependency: Doctor_ID is the primary key for doctor information, while Treatment_ID is the key for treatment details.
- Split into two tables:
    - Doctor Table:
        - Doctor_ID, Doctor_Name, Specialization, Contact_Info
    - Treatment Table:
        - Treatment_ID, Treatment_Date, Treatment_Description, Outcome, Patient_ID, Doctor_ID (Foreign Key)

Third Normal Form (3NF):

- No transitive dependencies. Both tables are in 3NF.

Boyce-Codd Normal Form (BCNF):

- Both tables satisfy BCNF.

3. Appointment Table

First Normal Form (1NF):

- Ensure atomic values: All columns have atomic values.
- The table is in 1NF.

Second Normal Form (2NF):

- Partial Dependency: Prescription_ID is the primary key for prescription information, while Appointment_ID is the key for appointment details.
- Split into two tables:
    - Prescription Table:
        - Prescription_ID, Medication, Dosage, Instructions, Date_Issued, Patient_ID, Doctor_ID

- Appointment Table:
  - Appointment_ID, Appointment_Date, Reason, Patient_ID, Doctor_ID

Third Normal Form (3NF):

- No transitive dependencies. Both tables are in 3NF.

Boyce-Codd Normal Form (BCNF):

- Both tables satisfy BCNF.

4. LabTest Table

First Normal Form (1NF):

- Ensure atomic values: All columns have atomic values.
- The table is in 1NF.

Second Normal Form (2NF):

- Partial Dependency: The primary key is Test_ID, and all other attributes depend on it.
- The table is already in 2NF.

Third Normal Form (3NF):

- No transitive dependencies. The table is in 3NF.

Boyce-Codd Normal Form (BCNF):

- The table satisfies BCNF.

5. Bill Table

First Normal Form (1NF):

- Ensure atomic values: All columns have atomic values.

- The table is in 1NF.

Second Normal Form (2NF):

- Partial Dependency: The primary key is Bill_ID, and all other attributes depend on it.
- The table is already in 2NF.

Third Normal Form (3NF):

- No transitive dependencies. The table is in 3NF.

Boyce-Codd Normal Form (BCNF):

- The table satisfies BCNF.

6. Admin Table

First Normal Form (1NF):

- Remove repeating groups for Admin_Credentials and Activities.
- Split into two tables:
    - Admin Table:
        - Admin_ID, Admin_Name, Designation, Shift, Emp_ID
    - Admin credentials Table (Relationship):
        - Admin_ID, Username, Password

Second Normal Form (2NF):

- No partial dependencies. Admin_ID is the primary key.
- No changes.

Third Normal Form (3NF):

- No transitive dependencies exist.
- No changes.

Boyce-Codd Normal Form (BCNF):

No changes.

Third Normal Form (3NF):

- No transitive dependencies. Both tables are in 3NF.

Boyce-Codd Normal Form (BCNF):

- Both tables satisfy BCNF.

7. Room Table

First Normal Form (1NF):

- Ensure atomic values: All columns have atomic values.
- The table is in 1NF.

Second Normal Form (2NF):

- Partial Dependency: The primary key is Room_ID, and all other attributes depend on it.
- The table is already in 2NF.

Third Normal Form (3NF):

- No transitive dependencies. The table is in 3NF.

Boyce-Codd Normal Form (BCNF):

- The table satisfies BCNF.

8. Department Table

First Normal Form (1NF):

- Ensure atomic values: All columns have atomic values.
- The table is in 1NF.

Second Normal Form (2NF):

- Partial Dependency: Dept_ID is the primary key for department information, but there are dependencies between Doctor_ID and Nurse_ID that must be separated.
- Split into two tables:
    - Department Table:
        - Dept_ID, Department_Name, Head_of_Department
    - Nurse Table (Relationship):
        - Nurse_ID, name, contact_info, department_id

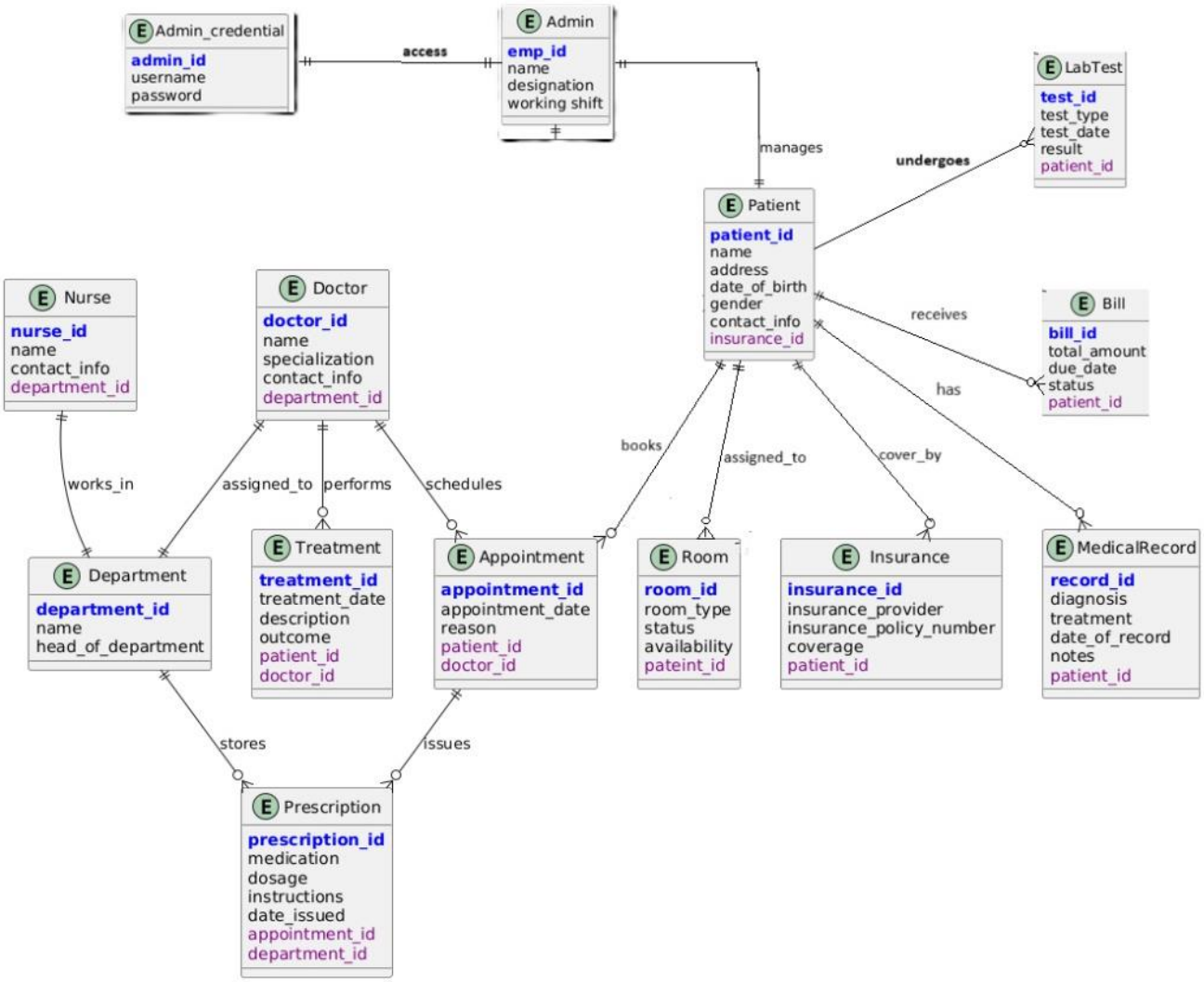
Third Normal Form (3NF):

- No transitive dependencies. Both tables are in 3NF.

Boyce-Codd Normal Form (BCNF):

- Both tables satisfy BCNF.

Normalized schema:



CHAPTER 7

BACKEND DESIGN:

CREATING TABLES

1. Patient Information

```
CREATE TABLE Patient (
    patient_id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    age INTEGER,
    gender TEXT,
    address TEXT,
    contact_number TEXT,
    medical_history TEXT
);
```

2. Doctor Details

```
CREATE TABLE Doctor (
    doctor_id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    specialization TEXT,
    contact_number TEXT,
   FOREIGN KEY (department_id) REFERENCES Department(department_id)
);
```

3. Prescription Records

```
CREATE TABLE Prescription (
    prescription_id INTEGER PRIMARY KEY AUTOINCREMENT,
    patient_id INTEGER,
    doctor_id INTEGER,
    prescription_date DATE,
    medication TEXT,
    dosage TEXT,
    FOREIGN KEY (patient_id) REFERENCES Patient(patient_id),
    FOREIGN KEY (doctor_id) REFERENCES Doctor(doctor_id)
);
```

4. Treatment Records

```
CREATE TABLE Treatment (
    treatment_id INTEGER PRIMARY KEY AUTOINCREMENT,
    patient_id INTEGER,
    doctor_id INTEGER,
    treatment_description TEXT,
    treatment_start_date DATE,
    treatment_end_date DATE,
    FOREIGN KEY (patient_id) REFERENCES Patient(patient_id),
    FOREIGN KEY (doctor_id) REFERENCES Doctor(doctor_id)
);
```

5. Admin Credentials Details

```
CREATE TABLE Administrator (
    admin_id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT NOT NULL UNIQUE,
    password TEXT NOT NULL,
);
```

6. Appointment Details

```
CREATE TABLE Appointment (
    appointment_id INTEGER PRIMARY KEY AUTOINCREMENT,
    patient_id INTEGER,
    doctor_id INTEGER,
    appointment_date DATE,
    appointment_time TEXT,
    FOREIGN KEY (patient_id) REFERENCES Patient(patient_id),
    FOREIGN KEY (doctor_id) REFERENCES Doctor(doctor_id)
);
```

7. Bill Details

```
CREATE TABLE Bill (
    bill_id INTEGER PRIMARY KEY AUTOINCREMENT,
    room_type INTEGER,
    patient_id INTEGER,
    total_amount REAL,
```

```
    payment_date DATE,
    FOREIGN KEY (patient_id) REFERENCES Patient(patient_id)
);
```

8. Lab Test

```
CREATE TABLE LabTest (
    test_id INTEGER PRIMARY KEY AUTOINCREMENT,
    patient_id INTEGER,
    test_name TEXT,
    test_date DATE,
    test_results TEXT,
    FOREIGN KEY (patient_id) REFERENCES Patient(patient_id)
);
```

9.

```
CREATE TABLE Room (
    room_id INTEGER PRIMARY KEY AUTOINCREMENT,
    patient_id INTEGER,
    room_status TEXT,
    room_availability TEXT,
    FOREIGN KEY (patient_id) REFERENCES Patient(patient_id),
);
```

10. Insurance

```
CREATE TABLE Insurance (
```

```
    insurance_id INTEGER PRIMARY KEY AUTOINCREMENT,
    patient_id INTEGER,
    insurance_provider TEXT,
    coverage_details TEXT,
    FOREIGN KEY (patient_id) REFERENCES Patient(patient_id)
);
```

11. Medical Record

```
CREATE TABLE MedicalRecord (
    record_id INTEGER PRIMARY KEY AUTOINCREMENT,
    patient_id INTEGER,
    doctor_id INTEGER,
    visit_date DATE,
    diagnosis TEXT,
    treatment TEXT,
    FOREIGN KEY (patient_id) REFERENCES Patient(patient_id),
    FOREIGN KEY (doctor_id) REFERENCES Doctor(doctor_id)
);
```

12. Admin:

```
CREATE TABLE  Admin (
        emp_id INTEGER PRIMARY KEY,
        name TEXT NOT NULL,
       designation TEXT NOT NULL,
```

```
    working_shift TEXT NOT NULL
);
```

13.Department:

```
CREATE TABLE Department (
    department_id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    HOD TEXT NOT NULL
);
```

INSERTION OF RECORDS:

```
import sqlite3
# Connect to SQLite database (or create it if it doesn't exist)
conn = sqlite3.connect('hospital_management.db')
cursor = conn.cursor()

# Insert records into Patient table
cursor.executemany[
    ('John', 45, 'Male', '123 Street, City', '555-1234', 'Diabetes'),
    ('Jane', 32, 'Female', '456 Avenue, City', '555-5678', 'Hypertension'),
    ('Emily', 28, 'Female', '789 Boulevard, City', '555-9876', 'Asthma'),
    ('Michael', 60, 'Male', '1010 Lane, City', '555-1122', 'Heart Disease'),
    ('Olivia', 22, 'Female', '1212 Road, City', '555-3344', 'Allergies')
])
```

```
# Insert records into Doctor table
cursor.executemany( [
    ('Dr. Smith', 'Cardiology', '555-1111', 1),
    ('Dr. Adams', 'Neurology', '555-2222', 2),
    ('Dr. Patel', 'Orthopedics', '555-3333', 3),
    ('Dr. Wilson', 'Pediatrics', '555-4444', 4),
    ('Dr. Lee', 'Dermatology', '555-5555', 5)
])
# Insert records into Prescription table
cursor.executemany( [
    (1, 1, '2024-09-01', 'Metformin', '500mg'),
    (2, 2, '2024-09-02', 'Lisinopril', '10mg'),
    (3, 3, '2024-09-03', 'Albuterol', '90mcg'),
    (4, 4, '2024-09-04', 'Aspirin', '75mg'),
    (5, 5, '2024-09-05', 'Cetirizine', '10mg')
])
# Insert records into Treatment table
cursor.executemany( [
    (1, 1, 'Cardiac rehabilitation', '2024-08-01', '2024-08-10'),
    (2, 2, 'Neurological assessment', '2024-08-02', '2024-08-05'),
    (3, 3, 'Fracture healing', '2024-08-03', '2024-08-15'),
    (4, 4, 'Pediatric checkup', '2024-08-04', '2024-08-04'),
    (5, 5, 'Skin rash treatment', '2024-08-05', '2024-08-12')
])
```

```
# Insert records into Administrator table
cursor.executemany( [
    ('admin1', 'pass123'),
    ('admin2', 'pass456'),
    ('admin3', 'pass789'),
    ('admin4', 'pass1011'),
    ('admin5', 'pass1213')
])

# Insert records into Appointment table
cursor.executemany( [
    (1, 1, '2024-09-10', '10:00 AM'),
    (2, 2, '2024-09-11', '11:00 AM'),
    (3, 3, '2024-09-12', '12:00 PM'),
    (4, 4, '2024-09-13', '01:00 PM'),
    (5, 5, '2024-09-14', '02:00 PM')
])

# Insert records into Bill table
cursor.executemany([
    (1, 1, 1000.0, '2024-09-15'),
    (2, 2, 1500.0, '2024-09-16'),
    (3, 3, 2000.0, '2024-09-17'),
    (4, 4, 2500.0, '2024-09-18'),
```

```
    (5, 5, 3000.0, '2024-09-19')
])
# Insert records into LabTest table
cursor.executemany( [
    (1, 'Blood Test', '2024-09-01', 'Normal'),
    (2, 'MRI', '2024-09-02', 'Normal'),
    (3, 'X-Ray', '2024-09-03', 'Fracture Detected'),
    (4, 'Urine Test', '2024-09-04', 'Normal'),
    (5, 'Skin Biopsy', '2024-09-05', 'Rash Detected')
])

# Insert records into Room table
cursor.executemany( [
    (1, 'Occupied', 'No'),
    (2, 'Occupied', 'No'),
    (3, 'Vacant', 'Yes'),
    (4, 'Occupied', 'No'),
    (5, 'Vacant', 'Yes')
])

# Insert records into Insurance table
cursor.executemany( [
    (1, 'HealthCare Inc.', 'Full Coverage'),
    (2, 'MediSure', 'Partial Coverage'),
```

```
    (3, 'HealthFirst', 'Full Coverage'),
    (4, 'SafeLife', 'Partial Coverage'),
    (5, 'GlobalHealth', 'Full Coverage')
])

# Insert records into MedicalRecord table
cursor.executemany([
    (1, 1, '2024-09-01', 'Diabetes', 'Metformin'),
    (2, 2, '2024-09-02', 'Hypertension', 'Lisinopril'),
    (3, 3, '2024-09-03', 'Fracture', 'Cast and Rest'),
    (4, 4, '2024-09-04', 'Pediatric Checkup', 'Vitamins'),
    (5, 5, '2024-09-05', 'Skin Rash', 'Cream and Antihistamines')
])

# Insert records into AdminCredentials table
cursor.executemany( [
    (1, 'Alice', 'Manager', 'Morning'),
    (2, 'Bob', 'Assistant Manager', 'Evening'),
    (3, 'Charlie', 'Clerk', 'Morning'),
    (4, 'Diana', 'Supervisor', 'Night'),
    (5, 'Edward', 'Coordinator', 'Evening')
])
```

```
# Insert records into Department table
cursor.executemany([
    (1, 'Cardiology', 'Dr. Smith'),
    (2, 'Neurology', 'Dr. Adams'),
    (3, 'Orthopedics', 'Dr. Patel'),
    (4, 'Pediatrics', 'Dr. Wilson'),
    (5, 'Dermatology', 'Dr. Lee')
])
# Commit the changes and close the connection
conn.commit()
conn.close()
print("Data inserted into all tables successfully.")
```
