SECD2523 -16 Database: 20242025 – SEMESTER 1

Proposal

Hospital Management System.

LECTURER: Sharifah

NAME	MATRIC ID
Liu Ruoyang	A23MJ4022
Pranto Anik Islam	A23MJ4024
KAHLAN SULTAN	A23MJ4021
Buguoshun	A23MJ4019
Saumik Hasan	A23MJ3009

Table of Contents

1.1 An overview for current system：

Hospital management systems (HMS) emerged to streamline the management and financial processes of medical institutions. Its goal is to make patient care more efficient and high-quality while also reducing operating costs.

Features of existing system：

1. Patient registration and appointment:

The system enables efficient patient registration and appointment booking, with patients registering online or onsite, reducing wait times and improving patient flow.

2. Electronic Medical Records (EMR):

Electronic medical records store comprehensive patient information, including medical history, treatment options and test results, facilitate communication between medical providers and ensure patient data is easily accessible.

3. Financial management:

These include tools to manage bills, insurance claims and payments. This feature automates financial processes, reduces errors, and ensures timely revenue collection.

4. Reporting and Analysis:

HMS generates a variety of reports on patient demographics, financial performance and operational efficiency to aid decision-making and strategic planning.

5. User Access and Security:

Role-based access control ensures sensitive patient information is protected while allowing medical professionals access to necessary data.

There are still many challenges, such as insufficient user experience and information fragmentation. While existing hospital management systems improve operational efficiency and patient care, continued updates and improvements will help them remain relevant in a rapidly changing healthcare environment.

1.2 Database Planning

Database planning defines the goals and requirements for the system, focusing on data security, ease of use, efficiency, and scalability. It outlines how data will be organized, accessed, and managed, ensuring the database supports hospital operations and improves overall performance.

1.2.1 Mission Statement:

Holistic Improvement: Our mission is to create a comprehensive database management system that addresses various needs within the hospital, ensuring all departments can collaborate effectively.

Patient-Centric Approach: We are dedicated to enhancing patient care by ensuring that healthcare providers have timely access to relevant medical information, which helps in delivering personalized treatment.

Technological Integration: The system will leverage the latest technologies to facilitate smooth data flow between departments, reducing delays and improving overall hospital efficiency.

Empowerment of Staff: By simplifying data management tasks, we aim to empower hospital staff, allowing them to dedicate more time to patient interactions and improving service quality.

1.2.2 Mission Objectives:

Data Security: Ensure the protection of patient data by implementing strong encryption and access controls to maintain privacy and confidentiality.

Ease of Use: Design an intuitive and simple interface to minimize training time and allow hospital staff to use the system efficiently.

Efficient Data Management: Streamline the process of entering, retrieving, and updating patient and hospital records to improve workflow and reduce delays.

Scalability: Build a system capable of growing with the hospital, accommodating more patients, departments, and new technologies as needed.

Regulatory Compliance: Ensure the system complies with all relevant healthcare regulations and standards to avoid legal issues and maintain the integrity of patient care.

1.3 System Definition

Hospital Management System (HMS) is an all-in-one package of handling almost everything for hospitals. It can be considered as the voice of all users — Administrators, Doctors, Nursing Staffs and front desk Receptionists, Patients, Pharmacists, path lab Physicians Management which makes flow of communication between two person so easy because these are built for betterment in patient care. The Hospital Management System provides a platform for appointment scheduling, patient records maintenance, medicine delivery record keeping and also needful features of billing & inventory control to improve work power and deliver breakthrough care.

1.3.1 System Boundaries

In-Scope Functions:

- **User Management:** Manage different parts of the system by various user roles (admin,, doctor,), nurse and more.
- **Patient Records:** Ability to track and manage medical histories, lab results, treatment plans etc.
- **Appointment Management System**: Scheduling this section consists of the scheduling system integrated, users may be able to schedule or check-in for purposes such as appointments with staff.
- **Billing & Insurance Processing:** Billing patients, insurance verification and payment.
- **Medication Management:** Prescription processing, medication administration tracking, and inventory management.
- **Laboratory Management:** Managing test orders, results entry, and tracking lab workflows.
- **Reporting and Analytics:** Generating operational, financial, and performance reports.

Out-of-Scope Functions:

- **External Healthcare Systems Integration:** Integration with third-party healthcare systems or electronic health records (EHR) not within the hospital.
- **Telehealth Services:** Features related to telemedicine or remote consultations not directly part of the HMS.
- **Non-Medical Services:** Management of services unrelated to patient care, such as facilities management or cafeteria services.

User Roles and Interfaces:

- **Admin:** Dashboard, user management, reports.
- **Doctor:** Access to patient records, appointment management, electronic prescriptions.
- **Nurse:** Patient care data, medication administration, task management.
- **Receptionist:** Appointment scheduling, check-in/check-out processes, billing.
- **Patient:** Access to health records, appointment management, billing information.
- **Pharmacy:** Inventory management, prescription processing, dispensing records.
- **Lab Technician:** Test order management and results entry.
- **Management:** Financial reporting, operational metrics, staff performance assessment.

Data Storage and Security:

The HMS will securely store sensitive patient and operational data, implementing robust security measures to ensure compliance with healthcare regulations such as HIPAA (Health Insurance Portability and Accountability Act).This definition and boundary outline provides a clear framework for the functionalities and limitations of the Hospital Management System, facilitating focused development and implementation while ensuring that user needs are adequately addressed.

System Limitations

Here are some potential limitations of our Hospital Management System (HMS):

1. External Healthcare Systems Integration:

The HMS may not integrate with external healthcare systems or electronic health records (EHR) platforms, potentially leading to data silos and hindering comprehensive patient care.

2. Non-Medical Services Management: The system does not handle non-medical services, such as facilities management or cafeteria services, which may lead to inefficiencies in managing the overall hospital operations.

3. Scalability:

Depending on the design, the system may face challenges in scaling to accommodate increased patient loads or additional functionalities as the hospital grows.

4. User Training:

Adequate training may be required for various user roles to effectively navigate and utilize the system, which could lead to initial inefficiencies or errors.

5. Data Migration and Legacy Systems:

Transitioning from existing legacy systems to the HMS may present challenges in data migration, compatibility, and system integration.

6. Dependency on Internet Connectivity:

If the HMS is cloud-based, reliance on internet connectivity may pose issues during outages or slow connections, affecting accessibility and performance.

7. Security Risks:

- Despite implementing robust security measures, there is always a risk of data breaches or cyber-attacks that can compromise sensitive patient information.

8. Limited Customization:

The system may have limited customization options for specific hospital needs, which could impact user satisfaction and system usability.

9. Technical Support and Maintenance:

The efficiency of the HMS may depend on the availability and quality of technical support, which can be a limitation if not adequately addressed.

1.3.2 Hospital Major User View

Here are some users' views in this system. Admin view, doctor view, nurse view, reception view, patient view, pharmacy view, lab technician view, management view.

1. Admin View

- **Dashboard:** Overview of key performance indicators and statistics.
- **User Management:** Oversee access for staff and patients.
- **Reports:** Create operational and financial reports.

2. Doctor View

- **Patient Records:** Access to medical histories and treatment plans.
- **Appointments:** Organize and oversee patient visits.
- **Prescriptions:** Create and manage electronic prescriptions.

3. Nurse View

- **Patient Care:** Access to vital signs and care plans.
- **Medication Administration:** Monitor medication schedules.
- **Task Management:** Review daily responsibilities.

4. Receptionist View

- **Appointment Scheduling:** Handle patient bookings.
- **Check-in/Check-out:** Manage patient arrivals and departures.
- **Billing:** Process payments and verify insurance information.

5. Patient View

- **Health Records:** Access medical history and lab results.
- **Appointments:** Organize and manage visit schedules.
- **Billing Information:** Review and pay bills.

6. Pharmacy View

- **Inventory Management:** Monitor medication supplies.
- **Prescription Processing:** Handle prescriptions from healthcare providers.
- **Dispensing:** Record medications dispensed to patients.

7. Lab Technician View

- **Test Orders:** Manage incoming laboratory requests.
- **Results Entry:** Input and track lab results.

8. Management View

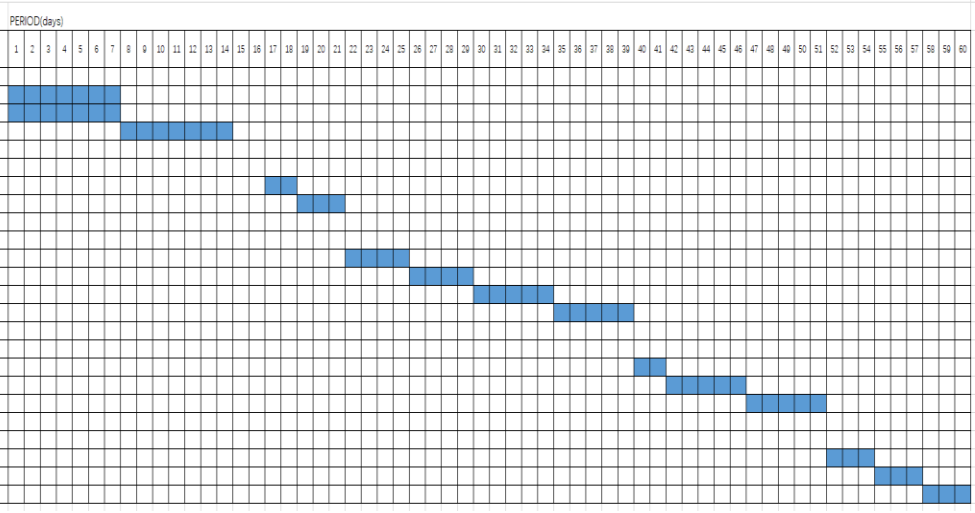
- **Financial Reports:** Summary of income and expenses.
- **Operational Metrics:** Track key performance indicators.
- **Staff Performance:** Assess staff productivity.

These perspectives enhance operational efficiency, improve patient care, and facilitate communication throughout the hospital.

1.4 Gantt Chart

PROJECT PLAN

TASK	PLAN START	PLAN DURATION	START	END
PLANNING				
Prepare Project Proposal	1	7	16/10/2024	22/10/2024
Read Company Report	1	7	16/10/2024	22/10/2024
Proposed an operational feasibility study	8	7	17/10/2024	23/10/2024
Requirement : Gather Information				
Conduct Interview	17	2	01/11/2024	02/11/2024
Administer questionaires	19	3	03/11/2024	05/11/2024
Analysis & Design				
Generate Model (Diagram)	22	4	06/11/2024	09/11/2024
Create Entity Relationship Diagram (ERD)	26	4	10/11/2024	13/11/2024
Generate Logical Diagram for new system(DFD)	30	5	14/11/2024	18/11/2024
Design Interface(GUI)	35	5	19/11/2024	23/11/2024
Implementation(System Prototype)				
Finalize definition of data structures	40	2	24/11/2024	25/11/2024
Create database tables	42	5	26/11/2024	30/11/2024
Implement the program coding	47	5	01/12/2024	05/12/2024
Demo of to-be system prototype				
Present the project	52	3	06/12/2024	08/12/2024
Finalise project report	56	3	10/12/2024	12/12/2024
Release a trial version and collect feedback	58	3	12/12/2024	14/12/2024



SECD2523 -16 Database: 20242025 – SEMESTER 1

LECTURER: SHARIFAH NURIZA BINTI SYED MUHAMMAD NAQUIB AL 'AIDRUS

Phase 2

NAME	MATRIC ID
Liu Ruoyang	A23MJ4022
Pranto Anik Islam	A23MJ4024
KAHLAN SULTAN	A23MJ4021
Buguoshun	A23MJ4019
Saumik Hasan	A23MJ3009

2.0 Database Design Requirements

2.1 Data Requirements

Entity	Data to be stored	Requirements of Data
Patient	1.patient ID 2.Full Name 3.Date of Birth 4.Contact Number 5. Email Adress	- [1] Patient ID is unique. - [4] Contact number must follow the format XXX-XXX-XXXX. - [5] Patients must provide a valid email address during registration.
Doctor	1.Doctor ID 2.Full Name 3.Speciality 4.Contact Number	- [1] Doctor ID is unique. - [3] Specialty must be specified and must match qualifications. - [4] Contact number must follow the format XXX-XXX-XXXX.
Nurse	1.Nurse ID 2.Full Name 3.Departemetn	- [1] Nurse ID is unique. - [3] Contact number must follow the format XXX-XXX-XXXX. - [4] Department must be assigned upon hiring.
Appointment	1.Appointment ID 2.Patient ID 3.Doctor ID 4.Date and Time 5.Status	- [1] Appointment ID is unique. - [2] [3] Patient ID and Doctor ID must be valid references. - [4] Each appointment must have a scheduled date and time. - [5] Status must be updated after each appointment (e.g., Completed, Cancelled).
Department	1.Department ID 2.Departement Name 3.Location	- [1] Department ID is unique. - [2] Department name must be specified and cannot be blank.

		- [3] Location must be designated (e.g., Floor or Building).
Treatment	1.Treatment ID 2.Patient ID 3.Doctor ID 4.Treatment Description 5. Date of Treatment	- [1] Treatment ID is unique. - [3] Treatment description must be detailed and clear. - [4] Date of treatment must be recorded.
Medication	1.Medication ID 2.Name 3. Dosage	- [1] Medication ID is unique. - [2] Medication name must be clearly specified. - [3] Dosage must be defined and follow medical standards.
Bill	1. Bill ID 2. Patient ID 3. Amount 4. Payment Status	- [1] Bill ID is unique. - [2] Patient ID must be valid and linked to an existing patient. - [3] Amount must be a positive decimal value. - [4] Payment status must be updated upon payment processing.
Lab Test	1. Test ID 2. Patient ID 3. Test Name 4. Data Conducted 5.Lab ID	- [1] Test ID is unique. - [2] Patient ID must be valid and linked to an existing patient. - [3] Test name must be clearly specified. - [4] Date conducted must be recorded and valid. -[5]The Lab ID specifies where the test is performed.
Visitor	1. Visitor ID 2. Patient ID 3. Relationship	- [1] Visitor ID is unique. - [2] Patient ID must be valid and linked to an existing patient. - [3] Visitor's relationship to the patient must be specified.

2.2 Transaction Requirements

Entity	Data	Data Entry	Data Update	Data Deletion	Data Queries

Patient	1.patient ID 2.Full Name 3.Date of Birth 4.Contact Number 5. Email Address	Sign up by patient, receptionist	Update information by patient	Delete account by receptionist	Query on patient data by doctor, nurse, patient, appointment, medication, bill, lab test, visitor, emergency
Doctor	1.Doctor ID 2.Full Name 3.Speciality 4.Contact Number	Sign up by doctor	Update information by doctor	Delete account by doctor	Query on doctor data by patient, doctor, treatment, medication
Nurse	1.Nurse ID 2.Full Name 3.Departemetn	Sign up by nurse	Update information by nurse	Delete account by nurse	Query on nurse data by patient, emergency
Appointment	1.Appointment ID 2.Patient ID 3.Doctor ID 4.Date and Time 5.Status	Entered by patient	Update information by patient, doctor	Delete appointment by patient, doctor	Query on appointment data by doctor, patient
Department	1.Department ID 2.Department Name 3.Location	Entered by manager	Update information by manager	Delete department data by manager	Query on department data by everyone
Treatment	1.Treatment ID 2.Patient ID 3.Doctor ID 4.Treatment Description 5. Date of Treatment	Entered by doctor	Update information by doctor	Delete treatment by doctor	Query on treatment data by doctor, nurse, patient, medication, lab test
Medication	1.Medication ID 2.Name 3. Dosage	Add new medication information	Update existing medication	Delete medication records	Query medication data for doctor,nurses,and pharmacists regarding prescriptions and treatments

Bill	1. Bill ID 2. Patient ID 3. Amount 4. Payment Status	Create a new bill for a patient	Update billing details	Delete a bill record	Query bill information for patients, doctors, and billing departments
Lab Test	1. Test ID 2. Patient ID 3. Test Name 4. Data Conducted 5.Lab ID	Record a new lab test for a patient	Update details of an existing lab test	Remove a lab test record	Query lab test information for doctors, nurses, and patients
Visitor	1. Visitor ID 2. Patient ID 3. Relationship	Add a new visitor record for a patient	Update visitor details	Remove a visitor record	Query visitor information for patients, staff, and administration

2.3 Cross-Reference Analysis

The table shows the cross reference between the Patient, Doctor, Nurse and visitors views with the main type of data used by all the entities.

	Doctor	Nurse	Visitor	Patient
Patient	X	X	X	X
Doctor	X	X		X
Nurse	X	X	X	X
Appointment	X	X		X
Department	X	X		X
Treatment	X	X		X
Medication	X	X		X
Lab Test	X	X		X
Bill	X	X		X

Visitor	X	X	X	X
Inventory	X	X		
Emergency	X	X		X

SECD2523-DATABASE

20242025 – SEMESTER 1

PHASE 3

Conceptual Design

FACULTY OF MJIIT

NAME	MATRIC ID
Liu Ruoyang	A23MJ4022
Pranto Anik Islam	A23MJ4024
Kahlan Sultan	A23MJ4021
Bu Guoshun	A23MJ4019
Saumik Hasan	A23MJ3009

1. **ERD Design**

The conceptual Entity-Relationship Diagram is designed to model the hospital management system by capturing entities, attributes, relationships, and their cardinalities. The design adheres to the system requirements outlined in Phase 2.

Explanation of Relationships

a. **Staff - Subclasses (Doctor, Nurse)**:
   Constraint: Optional, Disjoint (OR)
   Explanation: Every Staff member can be a Doctor, a Nurse, etc. (Optional) A Staff member cannot be both Doctor and Nurse(OR).

b. **Patient - Appointment**:
   Relationship Name: Has
   Multiplicity: 0..1 (Patient) to 1..* (Appointment)
   Explanation: A Patient can have multiple Appointments or none, but each Appointment must be linked to one Patient.

c. **Doctor - Appointment**:
   Relationship Name: Schedules
   Multiplicity: 0..1 (Doctor) to 1..* (Appointment)
   Explanation: A Doctor can schedule multiple Appointments or none, but each Appointment is managed by one Doctor.

d. **Patient - Treatment**:
   Relationship Name: Receives
   Multiplicity: 1..1 (Patient) to 1..* (Treatment)
   Explanation: A Patient can undergo one or multiple Treatments, but each Treatment is associated with one Patient.

e. **Doctor - Treatment**:
   Relationship Name: Provides
   Multiplicity: 1..1 (Doctor) to 1..* (Treatment)
   Explanation: A Doctor can administer one or multiple Treatments, but each Treatment is conducted by one Doctor.

f. **Patient - Bill**:
   Relationship Name: Pays
   Multiplicity: 1..1 (Patient) to 1..* (Bill)
   Explanation: Each Patient can generate one or multiple Bills, but each bill belongs to one patient.

g. **Patient - LabTest**:
   Relationship Name: Undergoes
   Multiplicity: 0..1 (Patient) to 1..* (LabTest)

Explanation: A Patient can undergo multiple LabTests or none, but each LabTest must be associated with one Patient.

h. **LabTest - Department**:

Relationship Name: ConductedIn

Multiplicity: 1..* (LabTest) to 0..1 (Department)

Explanation: Each LabTest is performed in one Department, but a Department may conduct many or may not conduct LabTests.

i. **Patient - Visitor**:

Relationship Name: VisitedBy

Multiplicity: 0..1 (Patient) to 1..* (Visitor)

Explanation: A Patient can have multiple Visitors or none, but each Visitor is associated with one Patient.

j. **Treatment - Medication**:

Relationship Name: Prescribes

Multiplicity: 0..* (Treatment) to 1..* (Medication)

Explanation: A Treatment may require multiple Medications or none, and each Medication can be prescribed for multiple Treatments.

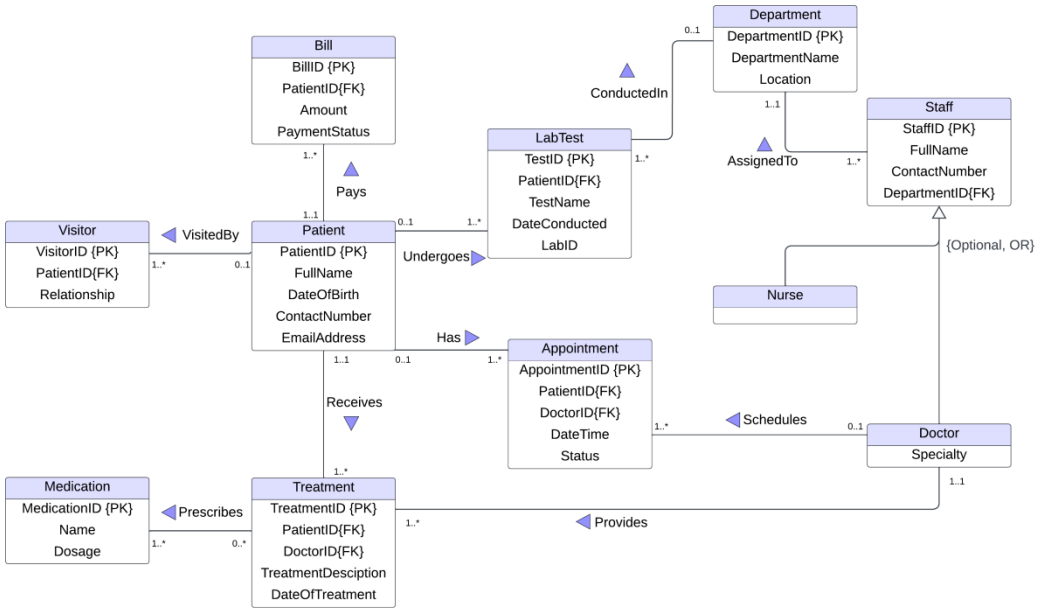
k. **Staff - Department**:

Relationship Name: AssignedTo

Multiplicity: 1..* (Staff) to 1..1 (Department)

Explanation: A Staff must be assigned to one Department, but a department can have one or multiple Staffs.

The ERD is shown below.



Data Dictionary

Overview

The Data Dictionary provides a detailed description of the entities, attributes, primary keys (PK), foreign keys (FK), and their roles in the database schema for the hospital management system.

Entities and Attributes:

Patient:

a. PatientID (PK): Unique identifier for each patient.
b. Name: Full name of the patient.
c. DOB: Date of birth of the patient.
d. ContactNumber:Phone number of patient
e. EmailAddress:Patient's email

Visitor:

a. VisitorID(PK):Unique identifier for each visitor
b. PatientID(FK):Unique identifier for each patient
c. Relationship:Visitor-patient relationship

Appointment:

a. AppointmentID (PK): Unique identifier for each appointment.
b. PatientID (FK): Links to PatientID in the Patient entity.
c. DoctorID (FK): Links to DoctorID in the Doctor entity.
d. Date Time: Date of the appointment.
e. Status:The specific location of the reservation

Doctor:

a. DoctorID (PK): Unique identifier for each doctor.
b. Name: Full name of the doctor.
c. Specialization: Area of expertise of the doctor.

Treatment:

a) TreatmentID (PK): Unique identifier for each treatment.
b) PatientID (FK): Links to PatientID in the Patient entity.
c) DoctorID (FK): Links to DoctorID in the Doctor entity
d) Details: Description of the treatment provided
e) DOB: Date of birth of the patient

Bill:

a. BillID (PK): Unique identifier for each bill.
b. PatientID (FK): Links to PatientID in the Patient entity.
c. Amount: Total amount charged.
d. PaymentStatus:Pay specific location

LabTest:

a. LabTestID (PK): Unique identifier for each lab test.
b. PatientID (FK): Links to PatientID in the Patient entity.
c. TestName:Name of test
d. DateConducted:date for lab test
e. LabID:Unique identifier for each lab

Department:

a. DepartmentID (PK): Unique identifier for each department.
b. Name: Name of the department.
c. Location:Department location

Medication:

a. MedicationID (PK): Unique identifier for each medication.
b. Name: Name of the medication.
c. Details: Description of the medication.

Staff:

a. StaffID (PK): Unique identifier for each staff.
b. DepartmentID (FK): Links to DepartmentID in the Department entity.
c. FullName: Full name of the staff.
d. ContactNumber：Phone number of staff

Entity	Attribute	Key (PK/FK)	Description	Data Type
Patient	PatientID	PK	Unique identifier for each patient	INTEGER
	Name		Full name of the patient	VARCHAR(100)
	DOB		Date of birth of the patient	DATE
	Gender		Gender of the patient	CHAR(1)
	ContactNumber		Phone number of the patient	VARCHAR(15)
	EmailAddress		Patient's email address	VARCHAR(100)
Visitor	VisitorID	PK	Unique identifier for each visitor	INTEGER
	PatientID	FK	Foreign key referencing PatientID	INTEGER
	Relationship		Relationship of the visitor to the patient	VARCHAR(50)
Appointment	AppointmentID	PK	Unique identifier for each appointment	INTEGER
	PatientID	FK	Foreign key referencing PatientID	INTEGER
	DoctorID	FK	Foreign key referencing DoctorID	INTEGER

Entity	Attribute	Key (PK/FK)	Description	Data Type
	DateTime		Date and time of the appointment	TIMESTAMP
	Status		Status of the appointment	VARCHAR(20)
Doctor	DoctorID	PK	Unique identifier for each doctor	INTEGER
	Name		Full name of the doctor	VARCHAR(100)
	Specialization		Area of expertise	VARCHAR(100)
Treatment	TreatmentID	PK	Unique identifier for each treatment	INTEGER
	PatientID	FK	Foreign key referencing PatientID	INTEGER
	DoctorID	FK	Foreign key referencing DoctorID	INTEGER
	Details		Description of the treatment provided	TEXT
Bill	BillID	PK	Unique identifier for each bill	INTEGER
	PatientID	FK	Foreign key referencing PatientID	INTEGER
	Amount		Total amount charged	DECIMAL(10,2)
	PaymentStatus		Payment status (e.g., Paid, Pending)	VARCHAR(20)

Entity	Attribute	Key (PK/FK)	Description	Data Type
LabTest	LabTestID	PK	Unique identifier for each lab test	INTEGER
	PatientID	FK	Foreign key referencing PatientID	INTEGER
	DepartmentID	FK	Foreign key referencing DepartmentID	INTEGER
	TestName		Name of the lab test	VARCHAR(100)
	DateConducted		Date of the lab test	DATE
Department	DepartmentID	PK	Unique identifier for each department	INTEGER
	Name		Name of the department	VARCHAR(100)
	Location		Department location	VARCHAR(100)
Medication	MedicationID	PK	Unique identifier for each medication	INTEGER
	Name		Name of the medication	VARCHAR(100)
	Details		Description of the medication	TEXT
Staff	StaffID	PK	Unique identifier for each staff	INTEGER

Entity	Attribute	Key (PK/FK)	Description	Data Type
	DepartmentID	FK	Foreign key referencing DepartmentID	INTEGER
	FullName		Full name of the staff	VARCHAR(100)
	ContactNumber		Phone number of the staff	VARCHAR(15)
Nurse	NurseID	PK	Unique identifier for each nurse	INTEGER
	DepartmentID	FK	Foreign key referencing DepartmentID	INTEGER
	Name		Full name of the nurse	VARCHAR(100)

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Malaysia-Japan International Institute of Technology (MJIIT)

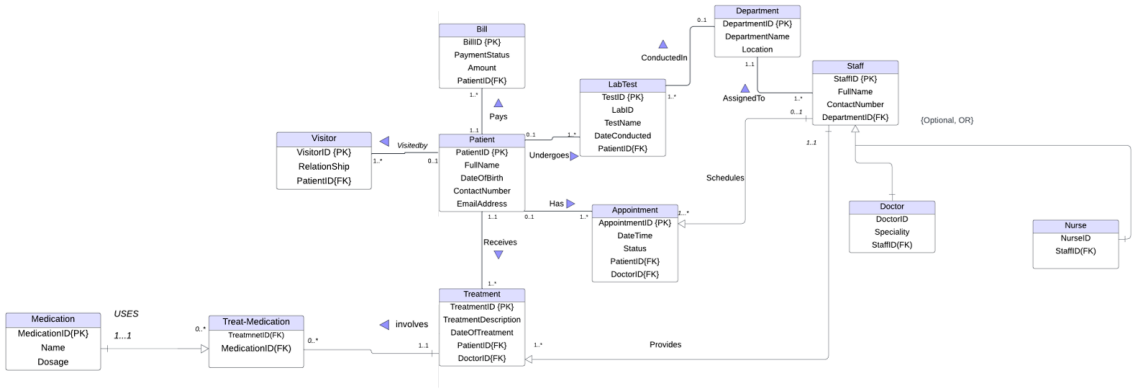
SECD2523-DATABASE
20242025 – SEMESTER 1
PHASE 4

Logical Design

FACULTY OF MJIIT

NAME	MATRIC ID
Liu Ruoyang	A23MJ4022
Pranto Anik Islam	A23MJ4024
Kahlan Sultan	A23MJ4021
Bu Guoshun	A23MJ4019
Saumik Hasan	A23MJ3009

- **Logical ERD :**



- **Relations schema for the above Logical ERD:**

1.Patient(PatientID (PK), FullName, DateOfBirth, ContactNumber, EmailAddress)
2.Visitor(Visitor (PK), Relationship, PatientID (FK))
3.Bill(BillID (PK), Amount, PaymentStatus, PatientID (FK))
4.LabTest(TestID (PK), TestName, DateConducted, LabID, PatientID (FK))
5.Department(DepartmentID (PK), DepartmentName, Location)
6.Staff(StaffID (PK), FullName, ContactNumber, DepartmentID (FK))
7.Doctor(DoctorID (PK), Specialty, StaffID (FK))
8.Nurse(NurseID (PK), StaffID (FK))
9.Appointment(AppointmentID (PK), DateTime, Status, PatientID (FK), DoctorID (FK))

10.Medication(MedicationID (PK), Name, Dosage)
11.Treatment(TreatmentID (PK), TreatmentDescription, DateOfTreatment, PatientID (FK), DoctorID (FK))
12.Treat-Medication (TreatmentID, MedicationID)

- **Normalization up till BCNF :**

1. **Patient**(PatientID (PK), FullName, DateOfBirth, ContactNumber, EmailAddress)

    - **First Normal Form**(removing repeating group)
      The relation is in 1NF
    - **Second Normal Form**(removing partial dependency)

The relation is in 2NF

- **Third Normal Form**(3NF):

- Potential dependency: ContactNumber → FullName, DateOfBirth, EmailAddress
    - A contact number might uniquely identify a patient, indicating a transitive dependency.

Decompose into 3NF:

To remove the transitive dependency, we split the table into:

1. **Patient :**
    - Attributes: PatientID (PK), Fullname, DateOfBirth , EmailAddress .
2. **Patient**:
    - Attributes: PatientID (PK,FK), ContactNumber .

Boyce-Codd Normal Form (BCNF)

Checking BCNF for Each Table:

- Both table satisfies BCNF.

2. Visitor(Visitor (PK), Relationship, PatientID (FK))
Functional Dependencies (FDs):
VisitorID → Relationship, PatientID

- **First Normal Form**(removing repeating group)
  The relation is in 1NF
- **Second Normal Form**(removing partial dependency)
  The relation is in 2NF
- **Third Normal Form**(removing transitive dependency)

Third Normal Form (3NF)

- To remove potential transitive dependencies we can make VisitorID and PatientID as composite Primary Key.

  Now the Schema is Visitor((VisitorID,PatientID)(PK),Relationship)

  - **1NF:** Satisfied because all attributes are atomic.

  - **2NF:** Satisfied because there are no partial dependencies.
  - **BCNF:** Also satisfied because every functional dependency has a superkey as its determinant.

3. Bill(BillID (PK), Amount, PaymentStatus, PatientID (FK))
Functional Dependencies (FDs):

BillID→Amount,PaymentStatus,PatientID
(The BillID determines all other attributes.)

- **First Normal Form**(removing repeating group)
  The relation is in 1NF
- **Second Normal Form**(removing partial dependency)
  The relation is in 2NF
- **Third Normal Form**(removing transitive dependency)
- **BCNF :**
  This table already satisfies BCNF

4.**LabTest**(TestID (PK), TestName, DateConducted, LabID, PatientID (FK))
Functional Dependencies (FDs):

TestID→TestName,DateConducted,LabID,PatientID
(The TestID uniquely determines all other attributes.)

- **First Normal Form**(removing repeating group)

The relation is in 1NF

- **Second Normal Form**(removing partial dependency)

The relation is in 2NF

- **Third Normal Form**(removing transitive dependency)

1. Satisfy 2NF.
2. Remove transitive dependencies (non-prime attributes should not depend on other non-prime attributes).

Functional Dependencies:

1. TestID → TestName, DateConducted, LabID, PatientID (Primary key determines all attributes).
2. Possible dependency: LabID → TestName (If a specific lab always conducts specific tests).

To remove the transitive dependency (LabID → TestName), we decompose the table.

Decomposition into 3NF:

LabTest:

Attributes:

- TestID (PK, shared across subclasses).
- TestName (common to both tables).

LabTestInfo:

- TestID .
- LabID to .

LabTest Instances:

DateConducted, LabID, and PatientID .

Here LabID and PatientID together made composite Primary Key but as LabID or PatientID alone cannot determined DateCounducted so we are free form partial dependencies as well as 2NF.

Here we cannot find any partial dependency or transitive dependencies so we are free from 3NF.

BCNF: All the tables satisfies BCNF.

5. Department(DepartmentID (PK), DepartmentName, Location)
Functional Dependencies (FDs):
DepartmentID→DepartmentName,Location
(The DepartmentID uniquely determines all other attributes.)
First Normal Form(removing repeating group)
The relation is in 1NF

- **Second Normal Form**(removing partial dependency)
  The relation is in 2NF
- **Third Normal Form**(removing transitive dependency)

Criteria for 3NF:

1. Satisfy 2NF.
2. Remove transitive dependencies (non-prime attributes should not depend on other non-prime attributes).

- **BCNF:** This table already satisfies BCNF .

6. **Staff**(StaffID (PK), FullName, ContactNumber, DepartmentID (FK))
Functional Dependencies (FDs):
StaffID→FullName,ContactNumber,DepartmentID

- **First Normal Form**(removing repeating group)
  The relation is in 1NF
  - **Second Normal Form**(removing partial dependency)

  The relation is in 2NF
  - **Third Normal Form**(removing transitive dependency)

1. Satisfy 2NF.
2. Remove transitive dependencies (non-prime attributes should not depend on other non-prime attributes).

BCNF: This table already satisfies BCNF

7.Doctor(DoctorID (PK), Specialty, StaffID (FK))
Functional Dependencies (FDs):
DoctorID→Specialty,StaffID
(The DoctorID uniquely determines both the doctor's Specialty and StaffID.)

- **First Normal Form**(removing repeating group)
  The relation is in 1NF
- **Second Normal Form**(removing partial dependency)
  The relation is in 2NF
- **Third Normal Form**(removing transitive dependency)

1. Satisfy 2NF.

2. Remove transitive dependencies (non-prime attributes should not depend on other non-prime attributes).We removed StaffID from Doctor table to remove potential transitive dependencies and anomaly. We added Name attribute to get doctors name. As there is no transitive dependency the table satisfies 3NF.

Functional Dependencies:

1. DoctorID → Specialty,Name

    - The primary key (DoctorID) determines all other attributes.

There are no transitive dependencies so the table satisfies 3NF.

- **BCNF: This table is already in BCNF .**

8.Nurse(NurseID (PK), StaffID (FK))
Functional Dependencies (FDs):

1. NurseID→StaffID(FK)
   (The NurseID uniquely determines the associated StaffID.)

    - **First Normal Form**(removing repeating group)
      The relation is in 1NF
    - **Second Normal Form**(removing partial dependency)
      The relation is in 2NF
    - **Third Normal Form**(removing transitive dependency)

1. Satisfy 2NF.
2. Remove transitive dependencies (non-prime attributes should not depend on other non-prime attributes).

    - We romevode StaffID form Nurse table to remove potential dependencies and added Name for more details about Nurse .NurseID is the primary key and determines all other attributes.

Since the only determinant is the primary key (NurseID), the table satisfies 3NF.

- **BCNF:** This table satisfies BCNF

9.Appointment(AppointmentID (PK), DateTime, Status, PatientID (FK), DoctorID (FK))
Functional Dependencies (FDs):

AppointmentID→DateTime,Status,PatientID,DoctorID
AppointmentID uniquely determines all other attributes in the relation.

- **First Normal Form**(removing repeating group)
  The relation is in 1NF
- **Second Normal Form**(removing partial dependency)
  The relation is in 2NF
- **Third Normal Form**(removing transitive dependency)

1. Satisfy 2NF.
2. Remove transitive dependencies (non-prime attributes should not depend on other non-prime attributes).

Functional Dependencies:

1. AppointmentID → DateTime, Status, PatientID, DoctorID

    - The primary key (AppointmentID) determines all other attributes.

There are no transitive dependencies because DateTime, Status, PatientID, and DoctorID do not depend on one another.
Thus, the table satisfies 3NF.

- **BCNF:**

1. Satisfy 3NF.
2. For every functional dependency, the determinant must be a superkey.

Functional Dependencies:

- AppointmentID → DateTime, Status, PatientID, DoctorID

- AppointmentID is the primary key and determines all other attributes.

Since the only determinant is the primary key (AppointmentID), the table satisfies BCNF.

10.Medication(MedicationID (PK), Name, Dosage)
Functional Dependencies (FDs):
MedicationID→Name,Dosage
(The MedicationID uniquely determines both Name and Dosage.)

- **First Normal Form**(removing repeating group)

The relation is in 1NF

- **Second Normal Form**(removing partial dependency)
  The relation is in 2NF
- **Third Normal Form**(removing transitive dependency)

1. Satisfy 2NF.
2. Remove transitive dependencies (non-prime attributes should not depend on other non-prime attributes).

Functional Dependencies:

1. MedicationID → Name, Dosage

- The primary key (MedicationID) determines all other attributes.

- **BCNF:** This table satisfies BCNF.

11.Treatment(TreatmentID (PK), TreatmentDescription, DateOfTreatment, PatientID (FK), DoctorID (FK))

Functional Dependencies (FDs):
TreatmentID→TreatmentDescription,DateOfTreatment,PatientID,DoctorID
(The TreatmentID uniquely determines all other attributes.)

- **First Normal Form**(removing repeating group)
  The relation is in 1NF
- **Second Normal Form**(removing partial dependency)
  The relation is in 2NF
- **Third Normal Form**(removing transitive dependency)

Criteria for 3NF:

1. Satisfy 2NF.
2. Remove transitive dependencies (non-prime attributes should not depend on other non-prime attributes).

Functional Dependencies:

1. TreatmentID → TreatmentDescription, DateOfTreatment, PatientID, DoctorID
   - The primary key (TreatmentID) determines all other attributes.

There are no transitive dependencies
because TreatmentDescription, DateOfTreatment, PatientID, and DoctorID do not depend on one another or any other non-prime attribute.
Thus, the table satisfies 3NF.

BCNF:

1. Satisfy 3NF.
2. For every functional dependency, the determinant must be a superkey.

Functional Dependencies:

1. TreatmentID → TreatmentDescription, DateOfTreatment, PatientID, DoctorID
    - TreatmentID is the primary key and determines all other attributes.

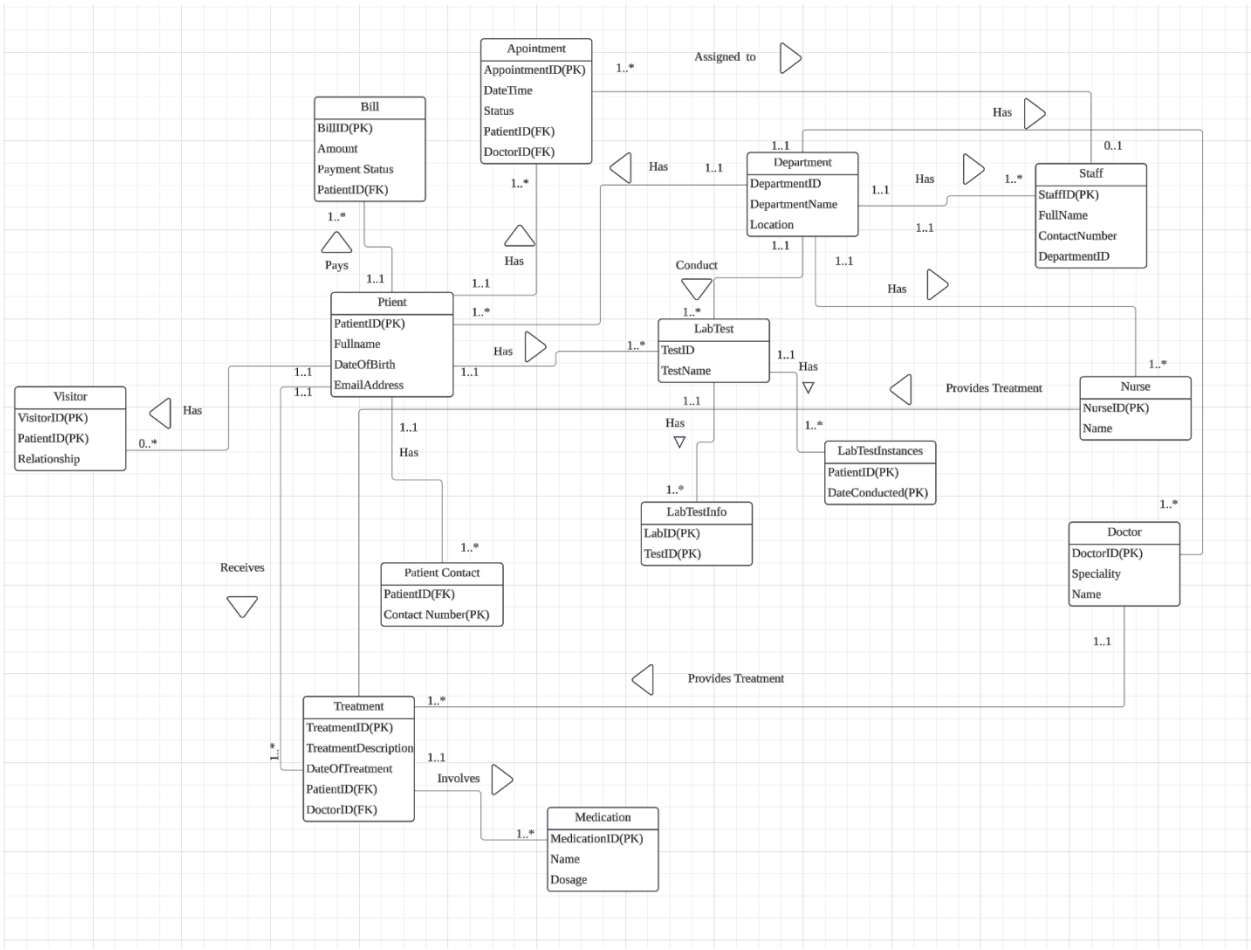
Since the only determinant is the primary key (TreatmentID), the table satisfies BCNF.

12.Treat-Medication (TreatmentID, MedicationID)
Functional Dependencies (FDs):
TreatmentID,MedicationID→(No additional attributes)
(Both attributes together form the composite primary key; no additional attributes are part of this schema.)

- **First Normal Form**(removing repeating group)
  The relation is in 1NF
- **Second Normal Form**(removing partial dependency)
  The relation is in 2NF
- **Third Normal Form**(removing transitive dependency)
  The relation is in 3NF

- **BCNF (** removing non-candidate key determinant)
  The relation is already in BCNF

Final Logical ERD:



Updated Data Dictionary for the Final Logical ERD:

Entity Name	Attribute Name	Data Type	Description	Key Type	Constrain
Patient	PatientID	INT	Unique Identifier for a Patient	Primary Key	Unique and Not null
	FullName	VARCHAR	Full name of the Patient		
	DateOfBirth	DATE	Patient's Date of Birth		
	EmailAddress	VARCHAR	Email Address of the Patient		
Patient Contact	PatientID	INT	Identifier of the Patient	Foreign Key	Not Null
	ContactNumber	VARCHAR	Patient's Contact Number	Primary Key	Unique and Not Null
Visitor	VisitorID	INT	Unique Identifier for the visitor	Primary Key	Unique and Not Null
	PatientID	INT	Identifier for the related Patient	Primary Key, Foreign Key	Not Null
	Relationship	VARCHAR	Relationship to Patient		
Bill	BillID		Unique identifier for Bill	Primary Key	Unique and Not Null

	Amount	DECIMAL	Total Bill Amount		
	PaymentStatus	VARCHAR	Status of Payment(Paid/ Unpaid)		
	PatientID	INT	Identifier for the related Patient	Foreign Key	
LabTest	TestID	INT	Unique Identifier for the Test.	Primary Key	Unique and Not Null
	TestName	VARCHAR	Name of the test		
LabTestInfo	LabID	INT	Unique Identifier for the Lab	Primary Key	Unique and Not Null
	TestID	INT	Identifier for the Related Test	Foreign Key	
LabTestInstances	TestID	INT	Identifier for the related Test	Primary Key	Unique and Not NULL
	PatientID	INT	Identifier for the related Patient	Primary Key	
	DateConducted	DATE	Date of the Lab Test when it was conducted.		
Department	DepartmentID	INT	Unique identifier for the Department.	Primary Key	Unique and Not Null
	DepartmentName	VARCHAR			
	Location	VARCHAR			
Staff	StaffID	INT	Unique Identifier for the staff member	Primary Key	Unique and Not Null
	FullName	VARCHAR	Name of the Staff member		
	ContactNumber	VARCHAR	Contact Number of the Staff member		

	DepartmentID	INT	Identifier for the related Departmen	Foreign key	
Doctor	DoctorID	INT	Unique identifier for a doctor	Primary Key	Unique and Not Null
	Speciality	VARCHAR	Doctor's area of expertise		
	Name	VARCHAR			
Nurse	NurseID	INT	Unique Identifier for a nurse	Primary Key	Unique and Not Null
	Name	VARCHAR			
Appointment	AppoinmentID	INT	Unique identifier for the appointment	Primary Key	Unique and Not Null
	DateTime	DATETIME			
	Status	VARCHAR	Status of the appointment		
	PatientID	INT	Identifier for the related Patient	Foreign Key	
	DoctorID	INT	Identifier for the assigned doctor	Foreign Key	
Treatment	TreatmentID	INT	Unique identifier for a treatment	Primary Key	Unique and Not Null
	TreatmentDescription	VARCHAR			
	DateOfTreatment	DATE	Date of the treatment		
	PatientID	INT	Identifier for the treated patient	Foreign Key	
	DoctorID	INT	Identifier for the doctor providing treatment	Foreign Key	
Medication	MedicationID	INT	Unique Identifier for medication	Primary Key	Unique and Not Null
	Name	VARCHAR	Name of the medication		

	Dosage	VARCHAR	Recommended dosage for medication		

Validation of Logical ERD based on the system's transaction requirements:

Creating Database Hospital and all the Tables:

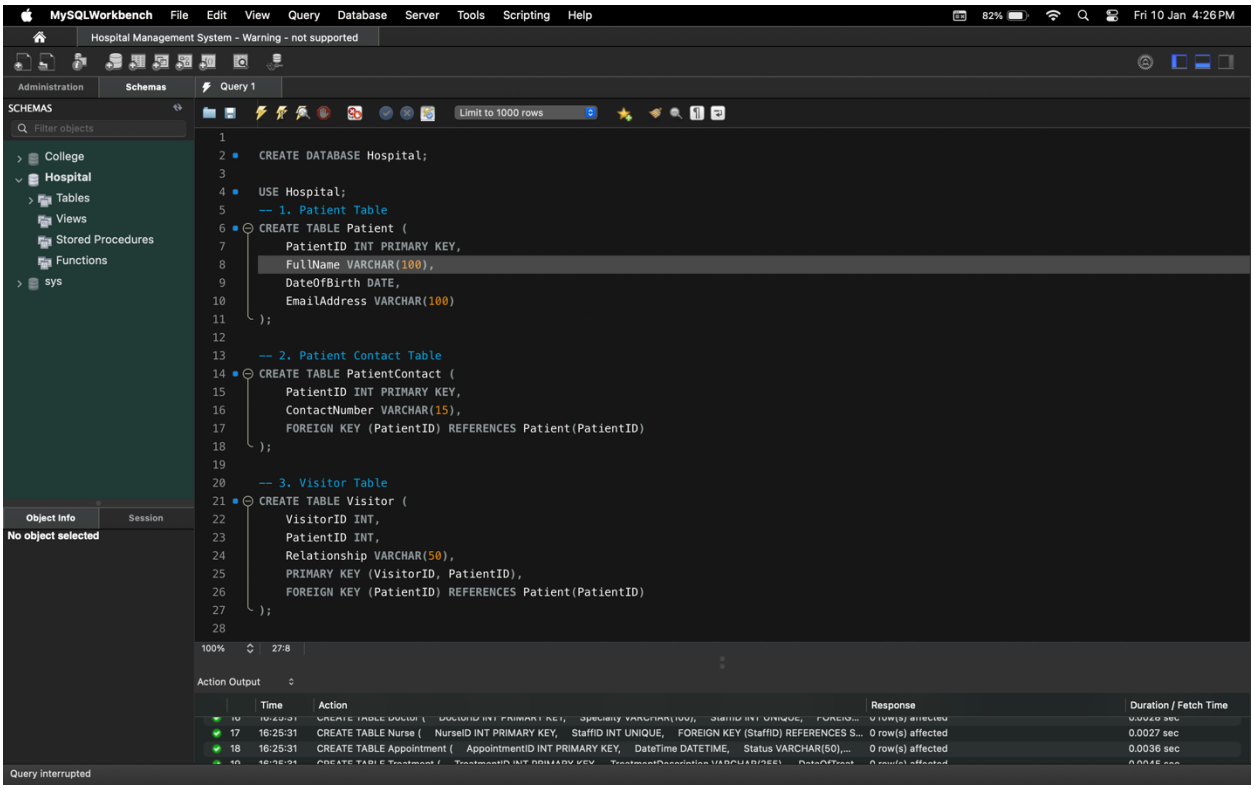
MYSQL Query:

```
CREATE DATABASE Hospital;
USE Hospital;
-- 1. Patient Table
CREATE TABLE Patient (
    PatientID INT PRIMARY KEY,
    FullName VARCHAR(100),
    DateOfBirth DATE,
    EmailAddress VARCHAR(100)
);

-- 2. Patient Contact Table
CREATE TABLE PatientContact (
    PatientID INT PRIMARY KEY,
    ContactNumber VARCHAR(15),
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID)
);

-- 3. Visitor Table
CREATE TABLE Visitor (
    VisitorID INT,
    PatientID INT,
    Relationship VARCHAR(50),
    PRIMARY KEY (VisitorID, PatientID),
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID)
);

-- 4. Bill Table
CREATE TABLE Bill (
    BillID INT PRIMARY KEY,
    Amount DECIMAL(10, 2),
    PaymentStatus VARCHAR(50),
    PatientID INT,
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID)
```

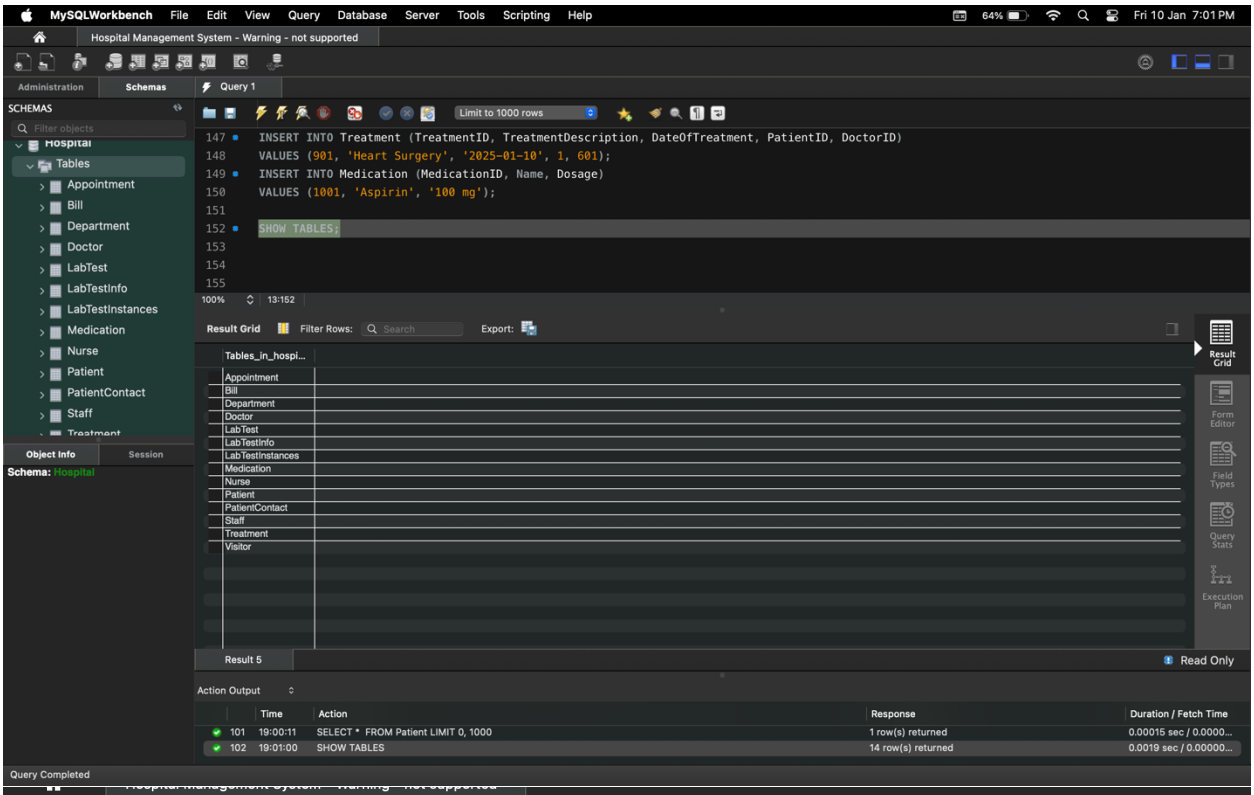
```
);

-- 5. LabTest Table
CREATE TABLE LabTest (
   TestID INT PRIMARY KEY,
   TestName VARCHAR(100)
);

-- 6. LabTestInfo Table
CREATE TABLE LabTestInfo (
   LabID INT PRIMARY KEY,
   TestID INT,
   FOREIGN KEY (TestID) REFERENCES LabTest(TestID)
);

-- 7. LabTestInstances Table
CREATE TABLE LabTestInstances (
   TestID INT,
   PatientID INT,
   DateConducted DATE,
   PRIMARY KEY (TestID, PatientID),
   FOREIGN KEY (TestID) REFERENCES LabTest(TestID),
   FOREIGN KEY (PatientID) REFERENCES Patient(PatientID)
);

-- 8. Department Table
CREATE TABLE Department (
   DepartmentID INT PRIMARY KEY,
   DepartmentName VARCHAR(100),
   Location VARCHAR(100)
);

-- 9. Staff Table
CREATE TABLE Staff (
   StaffID INT PRIMARY KEY,
   FullName VARCHAR(100),
   ContactNumber VARCHAR(15),
   DepartmentID INT,
   FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);

-- 10. Doctor Table
CREATE TABLE Doctor (
   DoctorID INT PRIMARY KEY,
   Specialty VARCHAR(100),
     Name VARCHAR
```

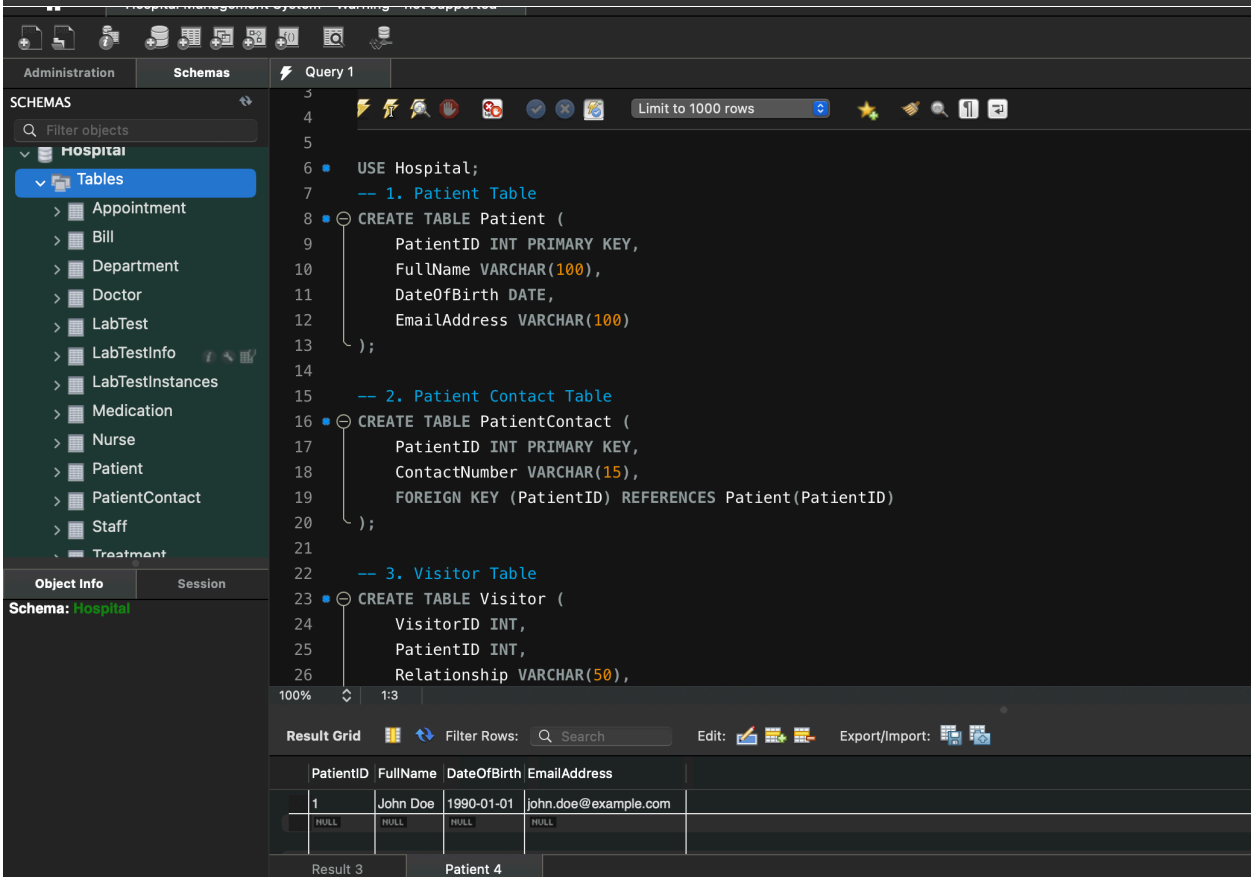
```
);

-- 11. Nurse Table
CREATE TABLE Nurse (
    NurseID INT PRIMARY KEY,
    Name VARCHAR
);

-- 12. Appointment Table
CREATE TABLE Appointment (
    AppointmentID INT PRIMARY KEY,
    DateTime DATETIME,
    Status VARCHAR(50),
    PatientID INT,
    DoctorID INT,
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID)
);

-- 13. Treatment Table
CREATE TABLE Treatment (
    TreatmentID INT PRIMARY KEY,
    TreatmentDescription VARCHAR(255),
    DateOfTreatment DATE NOT NULL,
    PatientID INT,
    DoctorID INT,
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID)
);

-- 14. Medication Table
CREATE TABLE Medication (
    MedicationID INT PRIMARY KEY,
    Name VARCHAR(100),
    Dosage VARCHAR(50)
);
```

Database and all the table created successfully

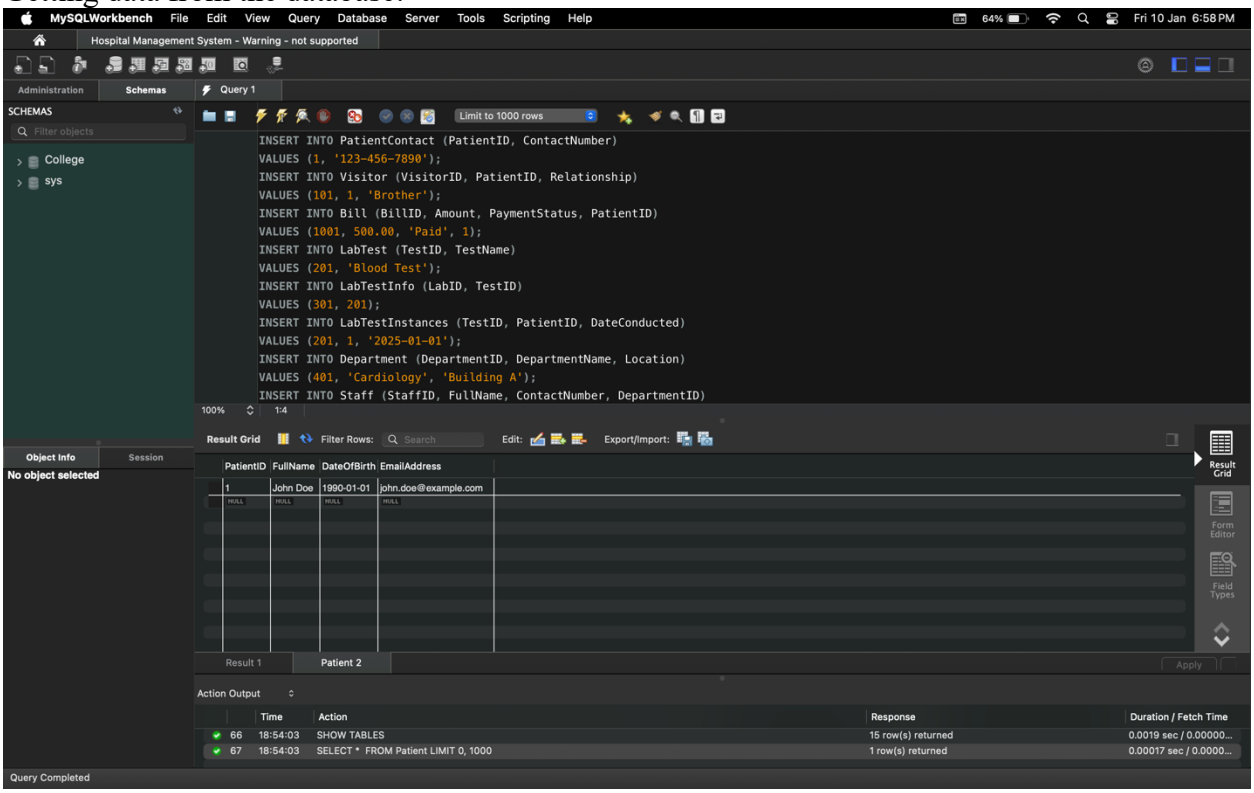

MySQLWorkbench File Edit View Query Database Server Tools Scripting Help
Fri 10 Jan 4:26 PM
Hospital Management System - Warning - not supported
Administration
Schemas
Query 1
SCHEMAS
Filter objects
College
Hospital
Tables
Views
Stored Procedures
Functions
sys
Limit to 1000 rows
1 CREATE DATABASE Hospital;
3 USE Hospital;
4 -- 1. Patient Table
5 CREATE TABLE Patient (
6 PatientID INT PRIMARY KEY,
7 FullName VARCHAR(100),
8 DateOfBirth DATE,
9 EmailAddress VARCHAR(100)
10);
11 -- 2. Patient Contact Table
12 CREATE TABLE PatientContact (
13 PatientID INT PRIMARY KEY,
14 ContactNumber VARCHAR(15),
15 FOREIGN KEY (PatientID) REFERENCES Patient(PatientID)
16);
17 -- 3. Visitor Table
18 CREATE TABLE Visitor (
19 VisitorID INT,
20 PatientID INT,
21 Relationship VARCHAR(50),
22 PRIMARY KEY (VisitorID, PatientID),
23 FOREIGN KEY (PatientID) REFERENCES Patient(PatientID)
24);
Object Info
Session
No object selected
100%
27:8
Action Output
Time
Action
Response
Duration / Fetch Time
17 16:25:31 CREATE TABLE Nurse (NurseID INT PRIMARY KEY, StaffID INT UNIQUE, FOREIGN KEY (StaffID) REFERENCES S... 0 row(s) affected 0.0027 sec
18 16:25:31 CREATE TABLE Appointment (AppointmentID INT PRIMARY KEY, DateTime DATETIME, Status VARCHAR(50),... 0 row(s) affected 0.0036 sec
Query interrupted

Show Tables:


MySQLWorkbench File Edit View Query Database Server Tools Scripting Help
Hospital Management System - Warning - not supported
Administration Schemas Query 1
SCHEMAS
Hospital
Tables
Appointment
Bill
Department
Doctor
LabTest
LabTestInfo
LabTestInstances
Medication
Nurse
Patient
PatientContact
Staff
Object Info Session
Schema: Hospital
147 INSERT INTO Treatment (TreatmentID, TreatmentDescription, DateOfTreatment, PatientID, DoctorID)
148 VALUES (901, 'Heart Surgery', '2025-01-10', 1, 601);
149 INSERT INTO Medication (MedicationID, Name, Dosage)
150 VALUES (1001, 'Aspirin', '100 mg');
151
152 SHOW TABLES;
153
154
155
Limit to 1000 rows
100% 13:152
Result Grid Filter Rows: Search Export:
Tables_in_hospi...
Appointment
Bill
Department
Doctor
LabTest
LabTestInfo
LabTestInstances
Medication
Nurse
Patient
PatientContact
Staff
Treatment
Visitor
Result 5
Read Only
Action Output
Time Action Response Duration / Fetch Time
101 19:00:11 SELECT * FROM Patient LIMIT 0, 1000 1 row(s) returned 0.00015 sec / 0.0000...
102 19:01:00 SHOW TABLES 14 row(s) returned 0.0019 sec / 0.00000...
Query Completed


Administration Schemas Query 1
SCHEMAS
Filter objects
Hospital
Tables
Appointment
Bill
Department
Doctor
LabTest
LabTestInfo
LabTestInstances
Medication
Nurse
Patient
PatientContact
Staff
Object Info Session
Schema: Hospital
Limit to 1000 rows
4
5
6 USE Hospital;
7 -- 1. Patient Table
8 CREATE TABLE Patient (
9 PatientID INT PRIMARY KEY,
10 FullName VARCHAR(100),
11 DateOfBirth DATE,
12 EmailAddress VARCHAR(100)
13);
14
15 -- 2. Patient Contact Table
16 CREATE TABLE PatientContact (
17 PatientID INT PRIMARY KEY,
18 ContactNumber VARCHAR(15),
19 FOREIGN KEY (PatientID) REFERENCES Patient(PatientID)
20);
21
22 -- 3. Visitor Table
23 CREATE TABLE Visitor (
24 VisitorID INT,
25 PatientID INT,
26 Relationship VARCHAR(50),
100% 1:3
Result Grid Filter Rows: Search Edit: Export/Import:
PatientID FullName DateOfBirth EmailAddress
1 John Doe 1990-01-01 john.doe@example.com
NULL NULL NULL NULL
Result 3 Patient 4

Getting data from the database:



As you can see all the table is created and our database connection was successful the validation is completed base on system's transaction requirement.

UTM UNIVERSITI TEKNOLOGI MALAYSIA | Malaysia-Japan International Institute of Technology (MJIIT)

SECD2523-DATABASE

20242025 - SEMESTER 1

PHASE 5

FACULTY OF MJIIT

NAME	MATRIC ID
Liu Ruoyang	A23MJ4022
Pranto Anik Islam	A23MJ4024
Kahlan Sultan	A23MJ4021
Bu Guoshun	A23MJ4019

Relational Database Schemas

1. Patient Table

This table stores the basic information of patients.

- **Attributes**:
  - PatientID (Primary Key): Unique identifier for each patient.
  - FullName: Name of the patient.
  - DateOfBirth: Date of birth of the patient.
  - EmailAddress: Email address of the patient.
- **Relationships**:
  - Referenced by PatientContact, Visitor, Bill, LabTestInstances, Appointment, and Treatment tables.

2. PatientContact Table

This table manages contact information for patients.

- **Attributes**:
  - PatientID (Primary Key, Foreign Key): Links to Patient(PatientID).
  - ContactNumber: Contact number of the patient.
- **Relationships**:
  - One-to-One with the Patient table.

3. Visitor Table

This table records visitor information associated with patients.

- **Attributes**:

    - VisitorID: Unique identifier for the visitor.
    - PatientID (Foreign Key): Links to Patient(PatientID).
    - Relationship: Relationship of the visitor to the patient.
- **Relationships**:
    - Many-to-Many relationship between visitors and patients.

4. Bill Table

This table handles billing information for patients.

- **Attributes**:
    - BillID (Primary Key): Unique identifier for each bill.
    - Amount: The bill amount.
    - PaymentStatus: Status of the payment (e.g., Paid, Pending).
    - PatientID (Foreign Key): Links to Patient(PatientID).
- **Relationships**:
    - One-to-Many with the Patient table.

5. LabTest Table

This table contains information about the types of lab tests available.

- **Attributes**:
    - TestID (Primary Key): Unique identifier for the test.
    - TestName: Name of the test.
- **Relationships**:
    - Referenced by LabTestInfo and LabTestInstances.

6. LabTestInfo Table

This table provides additional information about lab tests.

- **Attributes**:
    - LabID (Primary Key): Unique identifier for the lab test info.
    - TestID (Foreign Key): Links to LabTest(TestID).
- **Relationships**:
    - One-to-Many relationship with the LabTest table.

7. LabTestInstances Table

This table records individual lab tests conducted for patients.

- **Attributes**:
    - TestID (Primary Key, Foreign Key): Links to LabTest(TestID).
    - PatientID (Primary Key, Foreign Key): Links to Patient(PatientID).
    - DateConducted: The date the lab test was conducted.
- **Relationships**:
    - Many-to-Many relationship between LabTest and Patient.

8. Department Table

This table manages information about hospital departments.

- **Attributes**:
    - DepartmentID (Primary Key): Unique identifier for the department.
    - DepartmentName: Name of the department.
    - Location: Location of the department.
- **Relationships**:
    - Referenced by the Staff table.

9. Staff Table

This table stores information about hospital staff.

- **Attributes**:
    - StaffID (Primary Key): Unique identifier for each staff member.
    - FullName: Name of the staff member.
    - ContactNumber: Contact number of the staff member.
    - DepartmentID (Foreign Key): Links to Department(DepartmentID).
- **Relationships**:
    - One-to-Many relationship with the Department table.
    - Referenced by Doctor and Nurse tables.

10. Doctor Table

This table records information about doctors.

- **Attributes**:
    - DoctorID (Primary Key): Unique identifier for each doctor.
    - Specialty: Specialization of the doctor.
    - StaffID (Unique, Foreign Key): Links to Staff(StaffID).
- **Relationships**:
    - One-to-One with the Staff table.
    - Referenced by Appointment and Treatment tables.

11. Nurse Table

This table records information about nurses.

- **Attributes**:
    - NurseID (Primary Key): Unique identifier for each nurse.
    - StaffID (Unique, Foreign Key): Links to Staff(StaffID).
- **Relationships**:
    - One-to-One with the Staff table.

12. Appointment Table

This table handles patient appointments with doctors.

- **Attributes**:
    - AppointmentID (Primary Key): Unique identifier for each appointment.
    - DateTime: Date and time of the appointment.
    - Status: Status of the appointment (e.g., Scheduled, Completed).
    - PatientID (Foreign Key): Links to Patient(PatientID).
    - DoctorID (Foreign Key): Links to Doctor(DoctorID).
- **Relationships**:
    - Many-to-Many relationship between Patient and Doctor.

13. Treatment Table

This table records treatments provided to patients.

- **Attributes**:
    - TreatmentID (Primary Key): Unique identifier for each treatment.
    - TreatmentDescription: Description of the treatment.
    - DateOfTreatment: Date the treatment was provided.
    - PatientID (Foreign Key): Links to Patient(PatientID).
    - DoctorID (Foreign Key): Links to Doctor(DoctorID).
- **Relationships**:
    - One-to-Many with Patient and Doctor tables.
    - Referenced by the TreatMedication table.

14. Medication Table

This table manages information about medications.

- **Attributes**:

    - MedicationID (Primary Key): Unique identifier for each medication.
    - Name: Name of the medication.
    - Dosage: Dosage details for the medication.
- **Relationships**:
    - Referenced by the TreatMedication table.

15. TreatMedication Table

This table represents the relationship between treatments and medications.

- **Attributes**:
    - TreatmentID (Primary Key, Foreign Key): Links to Treatment(TreatmentID).
    - MedicationID (Primary Key, Foreign Key): Links to Medication(MedicationID).
- **Relationships**:
    - Many-to-Many relationship between Treatment and Medication.

2. Queries with Detailed Explanations

Basic Queries

1. **Retrieve all patients' basic information.**

```sql
-- 1. Retrieve all patients' basic information.
SELECT PatientID, FullName, DateOfBirth, EmailAddress
FROM Patient;
```

177 %

Results | Messages

	PatientID	FullName	DateOfBirth	EmailAddress
1	1	John Smith	2001-01-15	john.smith@gmail.com
2	2	Mary Johnson	2003-03-22	mary.johnson@gmail.com
3	3	Robert Brown	2000-07-08	robert.brown@gmail.com
4	4	Jennifer Davis	2005-12-02	jennifer.davis@gmail.com
5	5	Michael Wilson	1995-05-10	michael.wilson@gmail.com
6	6	Emily Moore	2002-11-23	emily.moore@gmail.com
7	7	Daniel Taylor	1975-08-19	daniel.taylor@gmail.com
8	8	Sarah Anderson	2008-02-15	sarah.anderson@gmail.com
9	9	David Thomas	2007-06-11	david.thomas@gmail.com
10	10	Laura Jackson	2010-09-25	laura.jackson@gmail.com
11	11	James White	1977-04-30	james.white@gmail.com
12	12	Linda Harris	2004-10-09	linda.harris@gmail.com
13	13	Andrew Martin	1981-01-20	andrew.martin@gmail.com
14	14	Elizabeth Thompson	2006-07-14	elizabeth.thompson@gmail.com
15	15	Christopher Garcia	2009-03-07	christopher.garcia@gmail.com
16	16	Patricia Martinez	1988-05-18	patricia.martinez@gmail.com
17	17	Matthew Robinson	2003-11-01	matthew.robinson@gmail.com
18	18	Barbara Clark	1991-02-25	barbara.clark@gmail.com
19	19	Joshua Lewis	2000-06-30	joshua.lewis@gmail.com
20	20	Karen Hall	2012-08-08	karen.hall@gmail.com

   - **Function**: Displays basic details for all patients, including ID, name, date of birth, and email address.
   - **Purpose**: Provides a quick overview of patient records.

2. **Retrieve all doctors' names and their specialties.**

```sql
-- 2. Retrieve all doctors' names and their specialties.
SELECT DoctorID, Specialty
FROM Doctor;
```

177 %

Results | Messages

	DoctorID	Specialty
1	1	Cardiology
2	2	Neurology
3	3	Orthopedics
4	4	Radiology
5	5	Pediatrics
6	6	Oncology
7	7	Emergency
8	8	Dermatology
9	9	Gastroenterology
10	10	Endocrinology
11	11	Urology
12	12	Ophthalmology
13	13	Psychiatry
14	14	Surgery
15	15	Anesthesiology
16	16	Nephrology
17	17	Pulmonology
18	18	Hematology
19	19	Rheumatology
20	20	Rehabilitation

- **Function**: Shows the IDs and specialties of all doctors.
- **Purpose**: Understands the distribution of medical specialties in the hospital.

3. **Retrieve all scheduled appointments after a specific date.**

```sql
-- 3. Retrieve all scheduled appointments after a specific date.
SELECT AppointmentID, DateTime, Status, PatientID, DoctorID
FROM Appointment
WHERE DateTime > '2024-01-01' AND Status = 'Scheduled';
```

177 %

Results | Messages

	AppointmentID	DateTime	Status	PatientID	DoctorID
1	13	2024-01-05 12:30:00.000	Scheduled	13	13
2	14	2024-02-10 13:30:00.000	Scheduled	14	14
3	16	2024-04-20 15:30:00.000	Scheduled	16	16
4	17	2024-05-25 16:30:00.000	Scheduled	17	17
5	19	2024-07-15 18:30:00.000	Scheduled	19	19
6	20	2024-08-20 09:30:00.000	Scheduled	20	20

- **Function**: Lists appointments scheduled after a specific date, including appointment ID, time, status, patient, and doctor details.
- **Purpose**: Tracks upcoming appointments.

4. **Retrieve all medications with their dosages.**

```sql
-- 4. Retrieve all medications with their dosages.
SELECT MedicationID, Name, Dosage
FROM Medication;
```

177 %

Results | Messages

	MedicationID	Name	Dosage
1	1	Paracetamol	500mg
2	2	Amoxicillin	250mg
3	3	Ibuprofen	400mg
4	4	Ciprofloxacin	500mg
5	5	Metformin	850mg
6	6	Atorvastatin	10mg
7	7	Omeprazole	20mg
8	8	Losartan	50mg
9	9	Aspirin	81mg
10	10	Clopidogrel	75mg
11	11	Levothyroxine	100mcg
12	12	Cetirizine	10mg
13	13	Loratadine	10mg
14	14	Prednisone	20mg
15	15	Albuterol	90mcg
16	16	Insulin	10 units
17	17	Metoprolol	25mg
18	18	Gabapentin	300mg
19	19	Hydrochlorothiazide	25mg
20	20	Fluoxetine	20mg

- **Function**: Lists all medications and their dosages.
- **Purpose**: Displays a catalog of available medications.

5. **Retrieve all treatments provided by a specific doctor.**

```sql
-- 5. Retrieve all treatments provided by a specific doctor.
SELECT TreatmentID, TreatmentDescription, DateOfTreatment
FROM Treatment
WHERE DoctorID = 1;
```

177 %

Results | Messages

	TreatmentID	TreatmentDescription	DateOfTreatment
1	1	Routine Checkup	2023-01-16

- **Function**: Lists treatments conducted by a specific doctor (DoctorID = 1), including treatment description and date.
- **Purpose**: Tracks the treatment history of an individual doctor.

Aggregate and Statistical Queries

6. **Count the total number of patients in the system.**

```sql
-- 6. Count the total number of patients in the system.
SELECT COUNT(*) AS TotalPatients
FROM Patient;
```

177 %

Results | Messages

	TotalPatients
1	20

- **Function**: Counts the total number of patients in the hospital system.
- **Purpose**: Provides an overall patient count.

7. **Count the number of appointments completed by each doctor.**

```sql
-- 7. Count the number of appointments completed by each doctor.
SELECT DoctorID, COUNT(AppointmentID) AS CompletedAppointments
FROM Appointment
WHERE Status = 'Completed'
GROUP BY DoctorID;
```

177 %

Results | Messages

	DoctorID	CompletedAppointments
1	2	1
2	5	1
3	7	1
4	10	1
5	12	1
6	15	1
7	18	1

- **Function**: Counts the number of appointments completed by each doctor.
- **Purpose**: Analyzes doctor workload and completion rates.

8. **Retrieve the total amount of bills paid by all patients.**

```sql
-- 8. Retrieve the total amount of bills paid by all patients.
SELECT SUM(Amount) AS TotalPaidBills
FROM Bill
WHERE PaymentStatus = 'Paid';
```

177 %

Results | Messages

	TotalPaidBills
1	3500.00

- **Function**: Calculates the total amount of all paid bills.
- **Purpose**: Tracks total hospital revenue from paid bills.

9. **Count the number of treatments provided for each specialty.**

```sql
-- 9. Count the number of treatments provided for each specialty.
SELECT Doctor.Specialty, COUNT(Treatment.TreatmentID) AS TreatmentCount
FROM Treatment
JOIN Doctor ON Treatment.DoctorID = Doctor.DoctorID
GROUP BY Doctor.Specialty;
```

177 %

Results | Messages

	Specialty	TreatmentCount
1	Anesthesiology	1
2	Cardiology	1
3	Dermatology	1
4	Emergency	1
5	Endocrinology	1
6	Gastroenterology	1
7	Hematology	1
8	Nephrology	1
9	Neurology	1
10	Oncology	1
11	Ophthalmology	1
12	Orthopedics	1
13	Pediatrics	1
14	Psychiatry	1
15	Pulmonology	1
16	Radiology	1
17	Rehabilitation	1
18	Rheumatology	1
19	Surgery	1
20	Urology	1

- **Function**: Counts the treatments provided, grouped by doctor specialty.
- **Purpose**: Evaluates the workload of each department or specialty.

Join Queries

10. **Retrieve all patients along with their contact numbers.**

```sql
-- 10. Retrieve all patients along with their contact numbers.
SELECT Patient.FullName, PatientContact.ContactNumber
FROM Patient
JOIN PatientContact ON Patient.PatientID = PatientContact.PatientID;
```

177 %

Results | Messages

	FullName	ContactNumber
1	John Smith	196183079
2	Mary Johnson	183828492
3	Robert Brown	145827931
4	Jennifer Davis	193782640
5	Michael Wilson	134576981
6	Emily Moore	168297430
7	Daniel Taylor	192874930
8	Sarah Anderson	148273092
9	David Thomas	172839400
10	Laura Jackson	193284762
11	James White	182763492
12	Linda Harris	147293860
13	Andrew Martin	192873620
14	Elizabeth Thompson	183920740
15	Christopher Garcia	195847201
16	Patricia Martinez	132846970
17	Matthew Robinson	174839205
18	Barbara Clark	183729405
19	Joshua Lewis	172839560
20	Karen Hall	198273640

- **Function**: Lists all patients with their contact numbers.
- **Purpose**: Quickly accesses patient contact information.

11. **Retrieve all appointments along with the doctors and patients involved.**

```sql
-- 11. Retrieve all appointments along with the doctors and patients involved.
SELECT Appointment.DateTime, Patient.FullName AS PatientName, Doctor.Specialty AS DoctorSpecialty
FROM Appointment
JOIN Patient ON Appointment.PatientID = Patient.PatientID
JOIN Doctor ON Appointment.DoctorID = Doctor.DoctorID;
```

177 %

Results | Messages

	DateTime	PatientName	DoctorSpecialty
1	2023-01-15 10:00:00.000	John Smith	Cardiology
2	2023-02-20 11:00:00.000	Mary Johnson	Neurology
3	2023-03-25 12:00:00.000	Robert Brown	Orthopedics
4	2023-04-10 13:00:00.000	Jennifer Davis	Radiology
5	2023-05-05 14:00:00.000	Michael Wilson	Pediatrics
6	2023-06-12 15:00:00.000	Emily Moore	Oncology
7	2023-07-19 16:00:00.000	Daniel Taylor	Emergency
8	2023-08-15 17:00:00.000	Sarah Anderson	Dermatology
9	2023-09-20 18:00:00.000	David Thomas	Gastroenterology
10	2023-10-25 09:00:00.000	Laura Jackson	Endocrinology
11	2023-11-30 10:30:00.000	James White	Urology
12	2023-12-15 11:30:00.000	Linda Harris	Ophthalmology
13	2024-01-05 12:30:00.000	Andrew Martin	Psychiatry
14	2024-02-10 13:30:00.000	Elizabeth Thompson	Surgery
15	2024-03-15 14:30:00.000	Christopher Garcia	Anesthesiology
16	2024-04-20 15:30:00.000	Patricia Martinez	Nephrology
17	2024-05-25 16:30:00.000	Matthew Robinson	Pulmonology
18	2024-06-30 17:30:00.000	Barbara Clark	Hematology
19	2024-07-15 18:30:00.000	Joshua Lewis	Rheumatology
20	2024-08-20 09:30:00.000	Karen Hall	Rehabilitation

- **Function**: Shows the time of appointments, patient names, and doctor specialties.
- **Purpose**: Visualizes appointment participants and schedules.

12. **Retrieve all treatments along with the medications used.**

```sql
-- 12. Retrieve all treatments along with the medications used.
SELECT Treatment.TreatmentDescription, Medication.Name AS MedicationName
FROM TreatMedication
JOIN Treatment ON TreatMedication.TreatmentID = Treatment.TreatmentID
JOIN Medication ON TreatMedication.MedicationID = Medication.MedicationID;
```

177 %

Results | Messages

	TreatmentDescription	MedicationName
1	Routine Checkup	Paracetamol
2	MRI Analysis	Amoxicillin
3	Fracture Fixation	Ibuprofen
4	Radiology Review	Ciprofloxacin
5	Vaccination	Metformin
6	Oncology Screening	Atorvastatin
7	Emergency Response	Omeprazole
8	Skin Treatment	Losartan
9	Cardiology Consultation	Aspirin
10	Neurology Test	Clopidogrel
11	Joint Repair	Levothyroxine
12	Radiology Scan	Cetirizine
13	Pediatrics Follow-up	Loratadine
14	Cancer Diagnosis	Prednisone
15	Surgery Preparation	Albuterol
16	Dermatology Consultation	Insulin
17	Heart Monitoring	Metoprolol
18	Brain Scan	Gabapentin
19	Orthopedics Evaluation	Hydrochlorothiazide
20	Radiology Test	Fluoxetine

- **Function**: Lists treatments with the medications used for each treatment.
- **Purpose**: Tracks the relationship between treatments and medications.

13. **Retrieve the list of patients who have undergone lab tests.**

```sql
-- 13. Retrieve the list of patients who have undergone lab tests.
SELECT DISTINCT Patient.FullName AS PatientName
FROM LabTestInstances
JOIN Patient ON LabTestInstances.PatientID = Patient.PatientID;
```

177 %

Results | Messages

	PatientName
1	Andrew Martin
2	Barbara Clark
3	Christopher Garcia
4	Daniel Taylor
5	David Thomas
6	Elizabeth Thompson
7	Emily Moore
8	James White
9	Jennifer Davis
10	John Smith
11	Joshua Lewis
12	Karen Hall
13	Laura Jackson
14	Linda Harris
15	Mary Johnson
16	Matthew Robinson
17	Michael Wilson
18	Patricia Martinez
19	Robert Brown
20	Sarah Anderson

- **Function**: Displays a unique list of patients who have completed lab tests.
- **Purpose**: Understands the coverage of lab testing among patients.

Conditional and Advanced Queries

14. **Retrieve patients who have unpaid bills.**

```sql
-- 14. Retrieve patients who have unpaid bills.
SELECT Patient.FullName, Bill.Amount, Bill.PaymentStatus
FROM Bill
JOIN Patient ON Bill.PatientID = Patient.PatientID
WHERE Bill.PaymentStatus = 'Pending';
```

177 %

Results | Messages

	FullName	Amount	PaymentStatus
1	Mary Johnson	300.00	Pending
2	Jennifer Davis	450.00	Pending
3	Emily Moore	350.00	Pending
4	Sarah Anderson	400.00	Pending
5	Laura Jackson	300.00	Pending
6	Linda Harris	150.00	Pending
7	Elizabeth Thompson	330.00	Pending
8	Patricia Martinez	390.00	Pending
9	Barbara Clark	410.00	Pending
10	Karen Hall	230.00	Pending

- **Function**: Lists patients with unpaid bills, including the bill amount and status.
- **Purpose**: Identifies overdue payments for follow-up.

15. **Retrieve all patients who visited a specific department.**

```sql
-- 15. Retrieve all patients who visited a specific department.
SELECT DISTINCT Patient.FullName AS PatientName, Department.DepartmentName
FROM Appointment
JOIN Patient ON Appointment.PatientID = Patient.PatientID
JOIN Doctor ON Appointment.DoctorID = Doctor.DoctorID
JOIN Staff ON Doctor.StaffID = Staff.StaffID
JOIN Department ON Staff.DepartmentID = Department.DepartmentID
WHERE Department.DepartmentName = 'Cardiology';
```

77 %

Results | Messages

	PatientName	DepartmentName
1	David Thomas	Cardiology
2	John Smith	Cardiology
3	Matthew Robinson	Cardiology

- **Function**: Lists patients who visited a specific department (e.g., Cardiology).
- **Purpose**: Tracks patient volume for a specific department.

16. **Retrieve the most recent appointment for each patient.**

```sql
-- 16. Retrieve the most recent appointment for each patient.
SELECT PatientID, MAX(DateTime) AS MostRecentAppointment
FROM Appointment
GROUP BY PatientID;
```

177 %

Results | Messages

	PatientID	MostRecentAppointment
1	1	2023-01-15 10:00:00.000
2	2	2023-02-20 11:00:00.000
3	3	2023-03-25 12:00:00.000
4	4	2023-04-10 13:00:00.000
5	5	2023-05-05 14:00:00.000
6	6	2023-06-12 15:00:00.000
7	7	2023-07-19 16:00:00.000
8	8	2023-08-15 17:00:00.000
9	9	2023-09-20 18:00:00.000
10	10	2023-10-25 09:00:00.000
11	11	2023-11-30 10:30:00.000
12	12	2023-12-15 11:30:00.000
13	13	2024-01-05 12:30:00.000
14	14	2024-02-10 13:30:00.000
15	15	2024-03-15 14:30:00.000
16	16	2024-04-20 15:30:00.000
17	17	2024-05-25 16:30:00.000
18	18	2024-06-30 17:30:00.000
19	19	2024-07-15 18:30:00.000
20	20	2024-08-20 09:30:00.000

- **Function**: Displays the most recent appointment time for each patient.
- **Purpose**: Tracks the latest interactions for each patient.

17. **Retrieve the total revenue generated by each department.**

```sql
-- 17. Retrieve the total revenue generated by each department.
SELECT Department.DepartmentName, SUM(Bill.Amount) AS TotalRevenue
FROM Bill
JOIN Patient ON Bill.PatientID = Patient.PatientID
JOIN Appointment ON Patient.PatientID = Appointment.PatientID
JOIN Doctor ON Appointment.DoctorID = Doctor.DoctorID
JOIN Staff ON Doctor.StaffID = Staff.StaffID
JOIN Department ON Staff.DepartmentID = Department.DepartmentID
WHERE Bill.PaymentStatus = 'Paid'
GROUP BY Department.DepartmentName;
```

177 %

Results | Messages

	DepartmentName	TotalRevenue
1	Cardiology	1100.00
2	Emergency	480.00
3	Orthopedics	1100.00
4	Pediatrics	820.00

- **Function**: Calculates the total revenue for each department based on paid bills.
- **Purpose**: Evaluates the financial contribution of each department.

18. **Retrieve the number of appointments scheduled for each month.**

```sql
-- 18. Retrieve the number of appointments scheduled for each month.
SELECT MONTH(DateTime) AS Month, COUNT(*) AS AppointmentCount
FROM Appointment
GROUP BY MONTH(DateTime);
```

177 %

Results | Messages

	Month	AppointmentCount
1	1	2
2	2	2
3	3	2
4	4	2
5	5	2
6	6	2
7	7	2
8	8	2
9	9	1
10	10	1
11	11	1
12	12	1

- **Function**: Counts the number of appointments grouped by month.
- **Purpose**: Analyzes monthly trends in appointment scheduling.

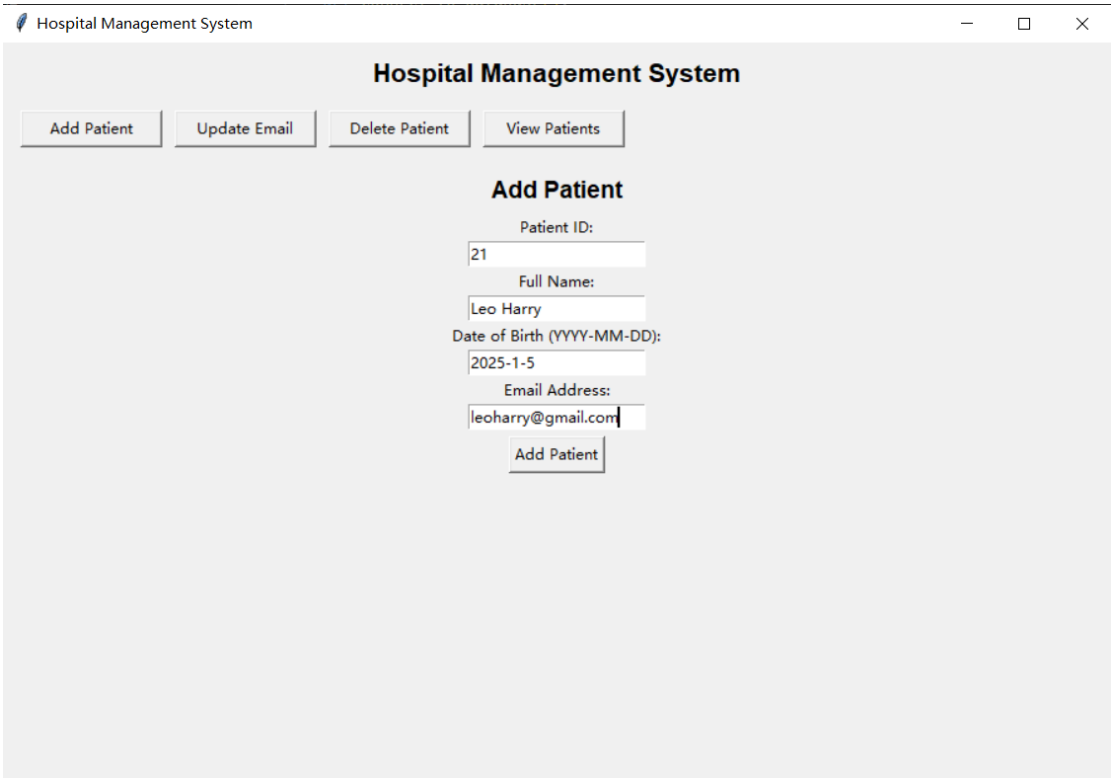
3.User Manual for Hospital Management System

Introduction
This application is a simple Hospital Management System designed for managing patient data efficiently. It provides a graphical user interface (GUI) for performing basic database operations such as adding, updating, deleting, and viewing patient records.

Main Features
1. **Add Patient**: Add a new patient record to the database.
2. **Update Email:** Update the Email Address of an existing patient.
3. **Delete Patient**: Delete a patient record from the database.
4. **View Patients**: View all patient records or search for a specific patient by ID.

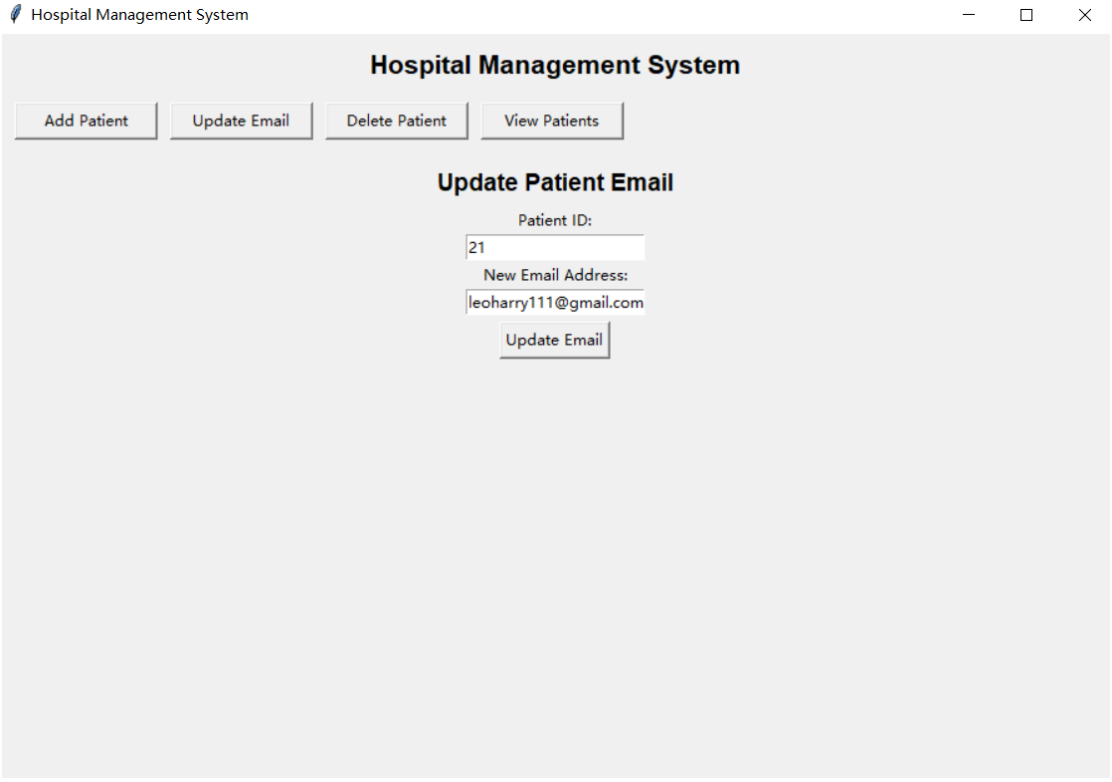
How to Use the Application

1. Add Patient



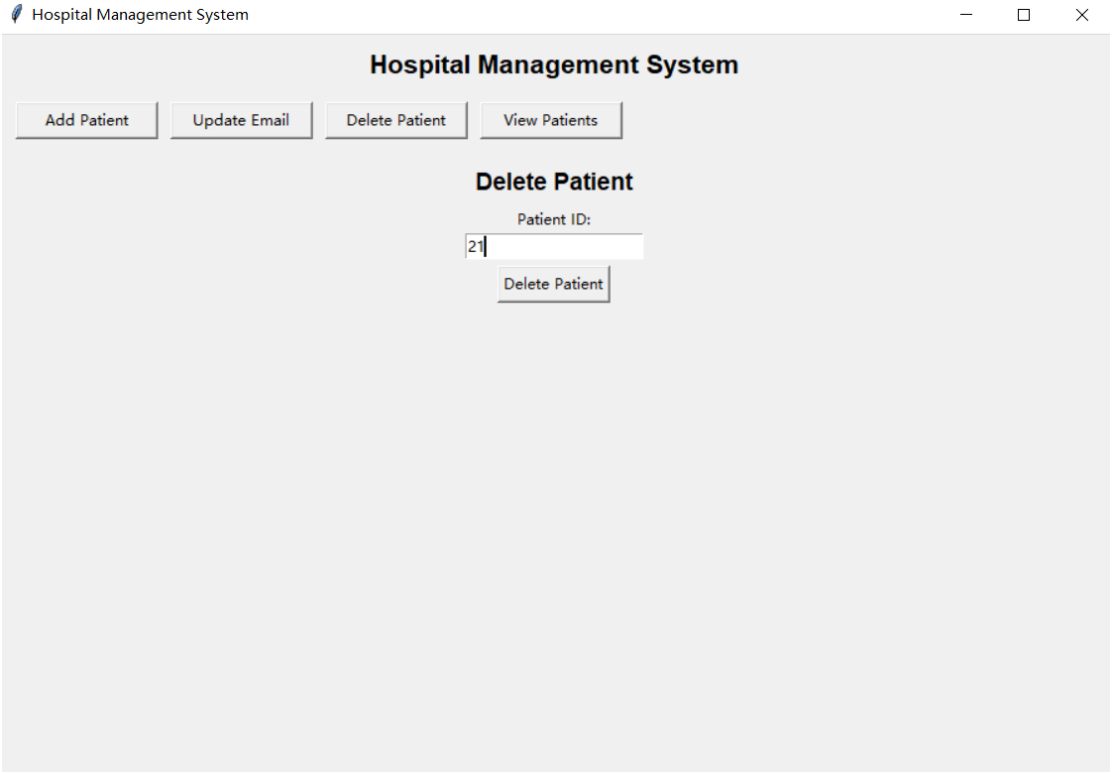
- **Steps to Add a Patient**:
    1. Click on the **"Add Patient"** button at the top of the screen.
    2. Fill in the required details in the form:
        - **Patient ID**: A unique identifier for the patient.
        - **Full Name**: The full name of the patient.
        - **Date of Birth**: Enter the date in YYYY-MM-DD format.
        - **Email Address**: Enter the patient's email address.
    3. Click the **"Add Patient"** button below the textboxes to save the record.
    4. If successful, a confirmation message will appear.

2. Update Email



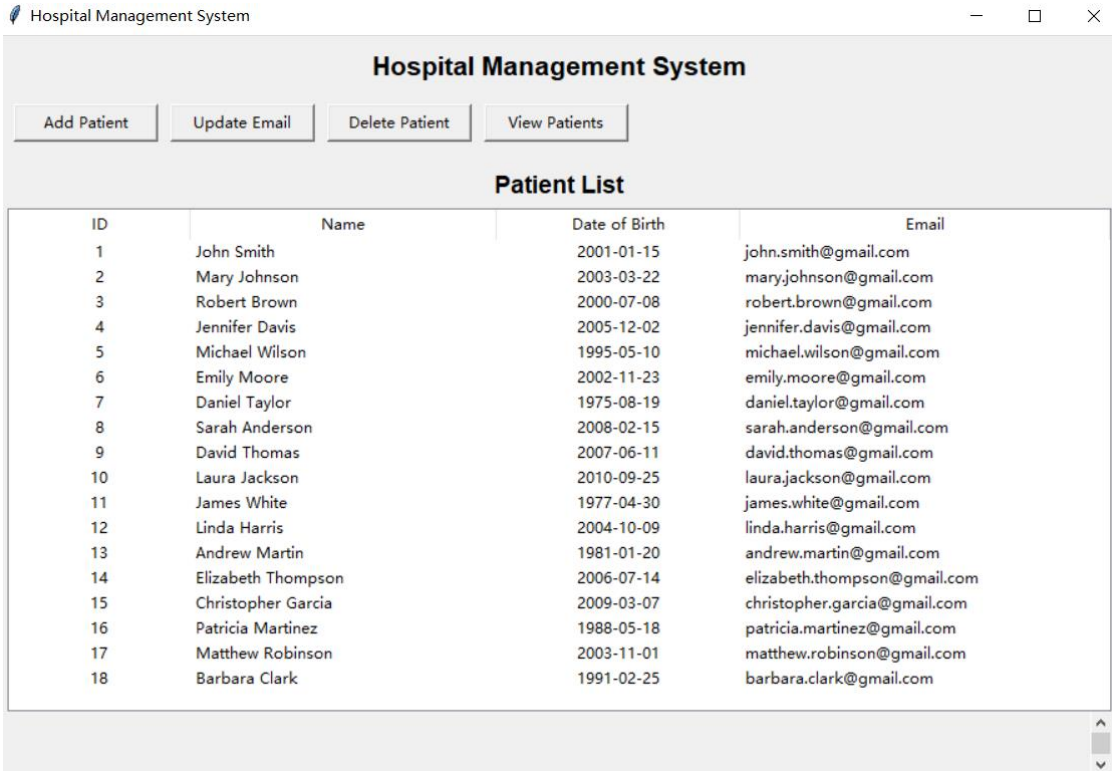
- **Steps to Update a Patient's Email**:
    1. Click on the **"Update Email"** button at the top of the screen.
    2. Enter the following details:
        - **Patient ID**: The ID of the patient whose contact details need to be updated.
        - **New Email Address**: Enter the updated email address.
    3. Click the **"Update Email"** button below the textboxes.
    4. A confirmation message will appear if the update is successful.

3. Delete Patient



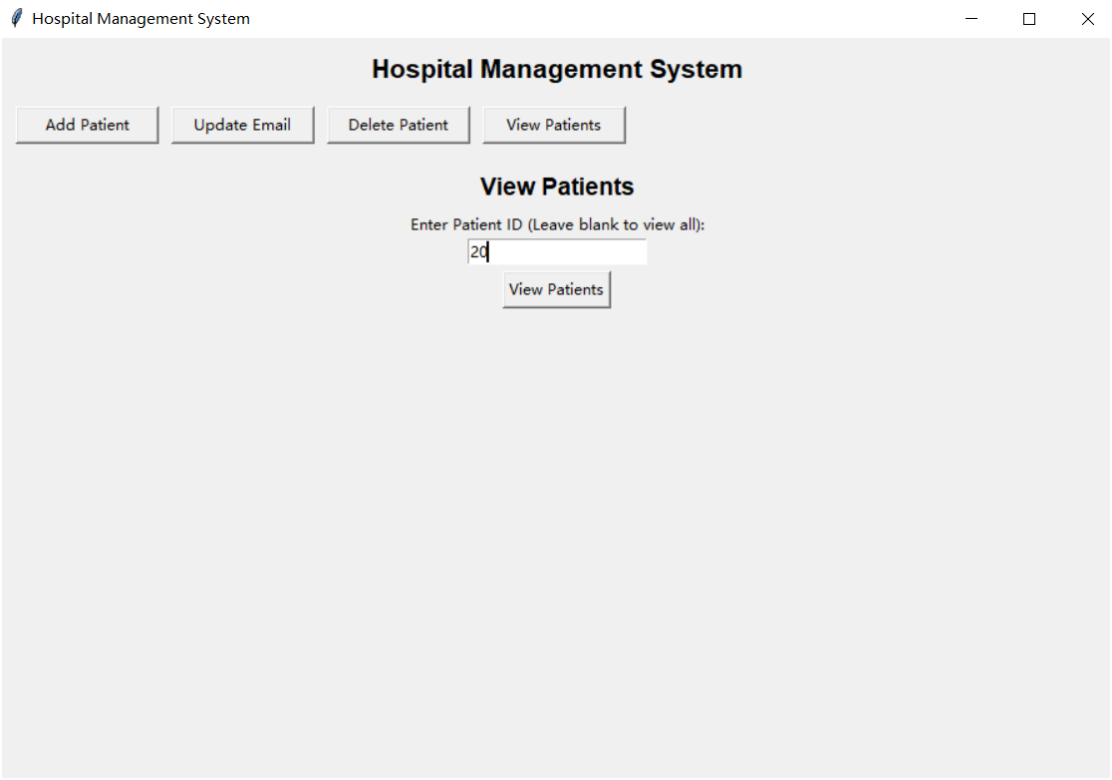
- **Steps to Delete a Patient**:
    1. Click on the **"Delete Patient"** button at the top of the screen.
    2. Enter the **Patient ID** of the record to be deleted.
    3. Click the **"Delete Patient"** button below the textbox.
    4. A confirmation message will appear if the deletion is successful.

4. View Patients

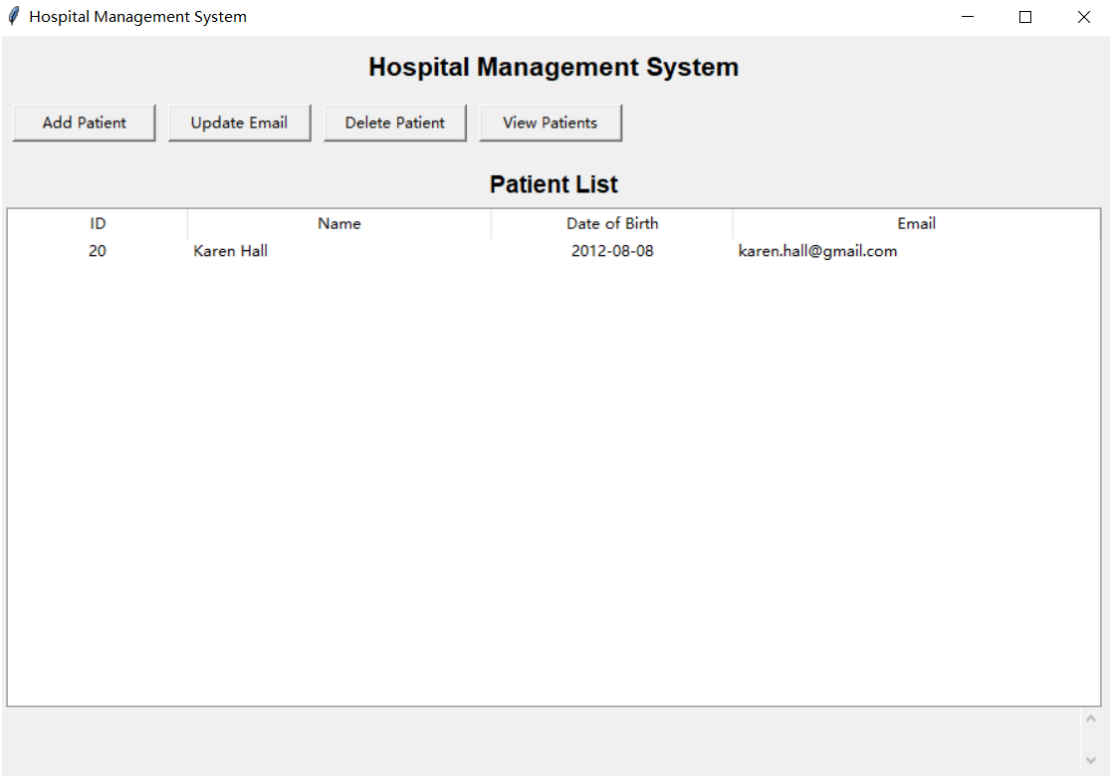


ID	Name	Date of Birth	Email
1	John Smith	2001-01-15	john.smith@gmail.com
2	Mary Johnson	2003-03-22	mary.johnson@gmail.com
3	Robert Brown	2000-07-08	robert.brown@gmail.com
4	Jennifer Davis	2005-12-02	jennifer.davis@gmail.com
5	Michael Wilson	1995-05-10	michael.wilson@gmail.com
6	Emily Moore	2002-11-23	emily.moore@gmail.com
7	Daniel Taylor	1975-08-19	daniel.taylor@gmail.com
8	Sarah Anderson	2008-02-15	sarah.anderson@gmail.com
9	David Thomas	2007-06-11	david.thomas@gmail.com
10	Laura Jackson	2010-09-25	laura.jackson@gmail.com
11	James White	1977-04-30	james.white@gmail.com
12	Linda Harris	2004-10-09	linda.harris@gmail.com
13	Andrew Martin	1981-01-20	andrew.martin@gmail.com
14	Elizabeth Thompson	2006-07-14	elizabeth.thompson@gmail.com
15	Christopher Garcia	2009-03-07	christopher.garcia@gmail.com
16	Patricia Martinez	1988-05-18	patricia.martinez@gmail.com
17	Matthew Robinson	2003-11-01	matthew.robinson@gmail.com
18	Barbara Clark	1991-02-25	barbara.clark@gmail.com

- **Steps to View All Patients**:
    1. Leave the textbox blank and click on the **"View Patients"** button below the textbox.
    2. All patient records will be displayed in a table format.
    3. Each record includes:
        - **ID**: Patient ID
        - **Name**: Patient's Name
        - **Date of Birth**: Patient's Date of Birth
        - **Email**: Patient's Email Address
- **Steps to Search for a Specific Patient**:



1. Enter the **Patient ID** in the search field.
2. Click the **"View Patients"** button.
3. The record of the specific patient will be displayed.



Error Handling

1. **Missing Required Fields**:
   - If any required field is left blank, an error message will be displayed.
   - Ensure all fields are filled before submitting the form.
2. **Invalid Input**:
   - Ensure the **Patient ID** is unique and matches the database record when updating or deleting.
   - Use the correct format for the **Date of Birth** (YYYY-MM-DD) and valid phone numbers/email addresses.
3. **Database Connection Error**:
   - If the application cannot connect to the database, check the server connection and retry.