**Team 6**

**Veetrag, Sam, Amitesh**

**Assignment – Design**

[This file contains both DDL (Page 1) and Stored Procedures (Page 12)]

**DDL**

-- Create Database

CREATE DATABASE IF NOT EXISTS HospitalManagementSystem;

USE HospitalManagementSystem;

-- Table: Role

CREATE TABLE Role (

RoleID INT PRIMARY KEY AUTO\_INCREMENT,

RoleName VARCHAR(50) NOT NULL UNIQUE,

Description TEXT

);

-- Table: User

CREATE TABLE User (

UserID INT PRIMARY KEY AUTO\_INCREMENT,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

Username VARCHAR(50) UNIQUE NOT NULL,

Password VARCHAR(255) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL,

RoleID INT,

AccountStatus ENUM('Active', 'Inactive', 'Suspended') DEFAULT 'Active',

RegistrationDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

LastLogin TIMESTAMP NULL,

FOREIGN KEY (RoleID) REFERENCES Role(RoleID) ON DELETE CASCADE

);

-- Table: LoginAttempt

CREATE TABLE LoginAttempt (

LoginID INT PRIMARY KEY AUTO\_INCREMENT,

UserID INT,

LoginTime TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

SuccessStatus BOOLEAN,

IPAddress VARCHAR(45),

FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE

);

-- Table: PasswordReset

CREATE TABLE PasswordReset (

ResetToken VARCHAR(255) PRIMARY KEY,

UserID INT,

ExpirationDate TIMESTAMP NOT NULL,

ResetStatus ENUM('Pending', 'Used', 'Expired') DEFAULT 'Pending',

FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE

);

-- Table: Patient

CREATE TABLE Patient (

PatientID INT PRIMARY KEY AUTO\_INCREMENT,

UserID INT NOT NULL,

DateOfBirth DATE NOT NULL,

Gender ENUM('Male', 'Female', 'Other'),

Address TEXT,

ContactNumber VARCHAR(15),

EmergencyContact VARCHAR(100),

BloodType VARCHAR(5),

FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE

);

-- Table: MedicalRecord

CREATE TABLE MedicalRecord (

RecordID INT PRIMARY KEY AUTO\_INCREMENT,

PatientID INT NOT NULL,

DateCreated TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

LastUpdated TIMESTAMP NULL,

MedicalHistory TEXT,

Allergies TEXT,

FOREIGN KEY (PatientID) REFERENCES Patient(PatientID) ON DELETE CASCADE

);

-- Table: Department

CREATE TABLE Department (

DepartmentID INT PRIMARY KEY AUTO\_INCREMENT,

DepartmentName VARCHAR(100) NOT NULL UNIQUE,

Location VARCHAR(100),

ManagerID INT NULL,

FOREIGN KEY (ManagerID) REFERENCES User(UserID) ON DELETE SET NULL

);

-- Table: Provider

CREATE TABLE Provider (

ProviderID INT PRIMARY KEY AUTO\_INCREMENT,

UserID INT NOT NULL,

Specialization VARCHAR(100),

LicenseNumber VARCHAR(50) UNIQUE,

DepartmentID INT,

FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE,

FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID) ON DELETE SET NULL

);

-- Table: DutyRoster

CREATE TABLE DutyRoster (

RosterID INT PRIMARY KEY AUTO\_INCREMENT,

ProviderID INT NOT NULL,

ShiftDate DATE NOT NULL,

ShiftStart TIME NOT NULL,

ShiftEnd TIME NOT NULL,

Status ENUM('Scheduled', 'On-Duty', 'Completed', 'Cancelled') DEFAULT 'Scheduled',

FOREIGN KEY (ProviderID) REFERENCES Provider(ProviderID) ON DELETE CASCADE

);

-- Table: Room

CREATE TABLE Room (

RoomID INT PRIMARY KEY AUTO\_INCREMENT,

RoomNumber VARCHAR(10) NOT NULL UNIQUE,

RoomType VARCHAR(50),

Floor INT,

Status ENUM('Available', 'Occupied', 'Maintenance') DEFAULT 'Available'

);

-- Table: AppointmentType

CREATE TABLE AppointmentType (

TypeID INT PRIMARY KEY AUTO\_INCREMENT,

TypeName VARCHAR(50) NOT NULL UNIQUE,

Duration INT, -- Duration in minutes

Description TEXT

);

-- Table: Appointment

CREATE TABLE Appointment (

AppointmentID INT PRIMARY KEY AUTO\_INCREMENT,

PatientID INT NOT NULL,

ProviderID INT NOT NULL,

RoomID INT,

TypeID INT,

AppointmentDate DATE NOT NULL,

StartTime TIME NOT NULL,

Status ENUM('Scheduled', 'Checked-In', 'Completed', 'Cancelled') DEFAULT 'Scheduled',

Notes TEXT,

FOREIGN KEY (PatientID) REFERENCES Patient(PatientID) ON DELETE CASCADE,

FOREIGN KEY (ProviderID) REFERENCES Provider(ProviderID) ON DELETE CASCADE,

FOREIGN KEY (RoomID) REFERENCES Room(RoomID) ON DELETE SET NULL,

FOREIGN KEY (TypeID) REFERENCES AppointmentType(TypeID) ON DELETE SET NULL

);

-- Table: Medication

CREATE TABLE Medication (

MedicationID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100) NOT NULL UNIQUE,

GenericName VARCHAR(100),

Category VARCHAR(50),

Manufacturer VARCHAR(100),

UnitPrice DECIMAL(10, 2) NOT NULL

);

-- Table: Inventory

CREATE TABLE Inventory (

InventoryID INT PRIMARY KEY AUTO\_INCREMENT,

MedicationID INT NOT NULL,

BatchNumber VARCHAR(50),

StockLevel INT NOT NULL,

ExpiryDate DATE NOT NULL,

Location VARCHAR(50),

FOREIGN KEY (MedicationID) REFERENCES Medication(MedicationID) ON DELETE CASCADE

);

-- Table: Prescription

CREATE TABLE Prescription (

PrescriptionID INT PRIMARY KEY AUTO\_INCREMENT,

PatientID INT NOT NULL,

ProviderID INT NOT NULL,

PrescriptionDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

Status ENUM('Active', 'Completed', 'Cancelled') DEFAULT 'Active',

Notes TEXT,

FOREIGN KEY (PatientID) REFERENCES Patient(PatientID) ON DELETE CASCADE,

FOREIGN KEY (ProviderID) REFERENCES Provider(ProviderID) ON DELETE CASCADE

);

-- Table: PrescriptionDetail

CREATE TABLE PrescriptionDetail (

DetailID INT PRIMARY KEY AUTO\_INCREMENT,

PrescriptionID INT NOT NULL,

MedicationID INT NOT NULL,

Dosage VARCHAR(50),

Frequency VARCHAR(50),

Duration INT, -- Duration in days

Quantity INT,

Instructions TEXT,

FOREIGN KEY (PrescriptionID) REFERENCES Prescription(PrescriptionID) ON DELETE CASCADE,

FOREIGN KEY (MedicationID) REFERENCES Medication(MedicationID) ON DELETE CASCADE

);

-- Table: LabType

CREATE TABLE LabType (

LabTypeID INT PRIMARY KEY AUTO\_INCREMENT,

TypeName VARCHAR(100) NOT NULL UNIQUE,

Description TEXT,

ProcessingTime INT -- Processing time in hours

);

-- Table: LabTest

CREATE TABLE LabTest (

LabTestID INT PRIMARY KEY AUTO\_INCREMENT,

PatientID INT NOT NULL,

ProviderID INT NOT NULL,

LabTypeID INT NOT NULL,

OrderDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

Status ENUM('Ordered', 'Sample-Collected', 'Processing', 'Completed', 'Cancelled') DEFAULT 'Ordered',

Priority ENUM('Routine', 'Urgent', 'Emergency') DEFAULT 'Routine',

FOREIGN KEY (PatientID) REFERENCES Patient(PatientID) ON DELETE CASCADE,

FOREIGN KEY (ProviderID) REFERENCES Provider(ProviderID) ON DELETE CASCADE,

FOREIGN KEY (LabTypeID) REFERENCES LabType(LabTypeID) ON DELETE CASCADE

);

-- Table: LabResult

CREATE TABLE LabResult (

ResultID INT PRIMARY KEY AUTO\_INCREMENT,

LabTestID INT NOT NULL,

ResultDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

ResultValue TEXT,

ReferenceRange VARCHAR(100),

Interpretation TEXT,

TechnicianID INT,

FOREIGN KEY (LabTestID) REFERENCES LabTest(LabTestID) ON DELETE CASCADE,

FOREIGN KEY (TechnicianID) REFERENCES Provider(ProviderID) ON DELETE CASCADE

);

-- Table: Insurance

CREATE TABLE Insurance (

InsuranceID INT PRIMARY KEY AUTO\_INCREMENT,

ProviderName VARCHAR(100) NOT NULL UNIQUE,

ContactInfo TEXT,

PolicyDetails TEXT

);

-- Table: PatientInsurance

CREATE TABLE PatientInsurance (

PatientInsuranceID INT PRIMARY KEY AUTO\_INCREMENT,

PatientID INT NOT NULL,

InsuranceID INT NOT NULL,

PolicyNumber VARCHAR(50) NOT NULL UNIQUE,

StartDate DATE NOT NULL,

EndDate DATE,

FOREIGN KEY (PatientID) REFERENCES Patient(PatientID) ON DELETE CASCADE,

FOREIGN KEY (InsuranceID) REFERENCES Insurance(InsuranceID) ON DELETE CASCADE

);

-- Table: Bill

CREATE TABLE Bill (

BillID INT PRIMARY KEY AUTO\_INCREMENT,

PatientID INT NOT NULL,

GeneratedDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

DueDate DATE NOT NULL,

TotalAmount DECIMAL(10, 2) NOT NULL,

Status ENUM('Pending', 'Paid', 'Overdue', 'Cancelled') DEFAULT 'Pending',

FOREIGN KEY (PatientID) REFERENCES Patient(PatientID) ON DELETE CASCADE

);

-- Table: PaymentMethod

CREATE TABLE PaymentMethod (

MethodID INT PRIMARY KEY AUTO\_INCREMENT,

MethodName VARCHAR(50) NOT NULL UNIQUE,

Description TEXT

);

-- Table: Payment

CREATE TABLE Payment (

PaymentID INT PRIMARY KEY AUTO\_INCREMENT,

BillID INT NOT NULL,

Amount DECIMAL(10, 2) NOT NULL,

PaymentDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

MethodID INT NOT NULL,

TransactionReference VARCHAR(100) UNIQUE,

Status ENUM('Pending', 'Completed', 'Failed') DEFAULT 'Pending',

FOREIGN KEY (BillID) REFERENCES Bill(BillID) ON DELETE CASCADE,

FOREIGN KEY (MethodID) REFERENCES PaymentMethod(MethodID) ON DELETE CASCADE

);

-- Table: NotificationType

CREATE TABLE NotificationType (

TypeID INT PRIMARY KEY AUTO\_INCREMENT,

TypeName VARCHAR(50) NOT NULL UNIQUE,

Description TEXT,

Template TEXT

);

-- Table: Notification

CREATE TABLE Notification (

NotificationID INT PRIMARY KEY AUTO\_INCREMENT,

TypeID INT NOT NULL,

UserID INT NOT NULL,

Title VARCHAR(100) NOT NULL,

Message TEXT NOT NULL,

SentDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

ReadDate TIMESTAMP NULL,

Status ENUM('Pending', 'Sent', 'Read', 'Failed') DEFAULT 'Pending',

FOREIGN KEY (TypeID) REFERENCES NotificationType(TypeID) ON DELETE CASCADE,

FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE

);

-- Table: Survey

CREATE TABLE Survey (

SurveyID INT PRIMARY KEY AUTO\_INCREMENT,

Title VARCHAR(100) NOT NULL UNIQUE,

Description TEXT,

StartDate DATE NOT NULL,

EndDate DATE NOT NULL,

Status ENUM('Draft', 'Active', 'Closed') DEFAULT 'Draft'

);

-- Table: SurveyResponse

CREATE TABLE SurveyResponse (

ResponseID INT PRIMARY KEY AUTO\_INCREMENT,

SurveyID INT NOT NULL,

PatientID INT NOT NULL,

ResponseDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

Responses TEXT NOT NULL,

FOREIGN KEY (SurveyID) REFERENCES Survey(SurveyID) ON DELETE CASCADE,

FOREIGN KEY (PatientID) REFERENCES Patient(PatientID) ON DELETE CASCADE

);

-- Table: Event

CREATE TABLE Event (

EventID INT PRIMARY KEY AUTO\_INCREMENT,

EventType ENUM('Reminder', 'Follow-up', 'Survey', 'Alert') NOT NULL,

RelatedID INT, -- Can reference other table IDs like AppointmentID or SurveyID

UserID INT NOT NULL,

EventDate TIMESTAMP NOT NULL,

Description TEXT,

Status ENUM('Scheduled', 'Triggered', 'Completed', 'Cancelled') DEFAULT 'Scheduled',

FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE

);

**Stored Procedures**

-- Add a new Role

DELIMITER //

CREATE PROCEDURE AddRole (

IN p\_RoleName VARCHAR(50),

IN p\_Description TEXT

)

BEGIN

INSERT INTO Role (RoleName, Description)

VALUES (p\_RoleName, p\_Description);

END //

DELIMITER ;

-- Retrieve a Role by ID

DELIMITER //

CREATE PROCEDURE GetRoleByID (

IN p\_RoleID INT

)

BEGIN

SELECT \* FROM Role WHERE RoleID = p\_RoleID;

END //

DELIMITER ;

-- Update an existing Role

DELIMITER //

CREATE PROCEDURE UpdateRole (

IN p\_RoleID INT,

IN p\_RoleName VARCHAR(50),

IN p\_Description TEXT

)

BEGIN

UPDATE Role

SET RoleName = p\_RoleName,

Description = p\_Description

WHERE RoleID = p\_RoleID;

END //

DELIMITER ;

-- Delete a Role

DELIMITER //

CREATE PROCEDURE DeleteRole (

IN p\_RoleID INT

)

BEGIN

DELETE FROM Role WHERE RoleID = p\_RoleID;

END //

DELIMITER ;

-- Add a new User

DELIMITER //

CREATE PROCEDURE AddUser (

IN p\_FirstName VARCHAR(50),

IN p\_LastName VARCHAR(50),

IN p\_Username VARCHAR(50),

IN p\_Password VARCHAR(255),

IN p\_Email VARCHAR(100),

IN p\_RoleID INT

)

BEGIN

INSERT INTO User (FirstName, LastName, Username, Password, Email, RoleID, AccountStatus, RegistrationDate)

VALUES (p\_FirstName, p\_LastName, p\_Username, p\_Password, p\_Email, p\_RoleID, 'Active', CURRENT\_TIMESTAMP);

END //

DELIMITER ;

-- Retrieve a User by ID

DELIMITER //

CREATE PROCEDURE GetUserByID (

IN p\_UserID INT

)

BEGIN

SELECT \* FROM User WHERE UserID = p\_UserID;

END //

DELIMITER ;

-- Update an existing User

DELIMITER //

CREATE PROCEDURE UpdateUser (

IN p\_UserID INT,

IN p\_FirstName VARCHAR(50),

IN p\_LastName VARCHAR(50),

IN p\_Email VARCHAR(100),

IN p\_RoleID INT

)

BEGIN

UPDATE User

SET FirstName = p\_FirstName,

LastName = p\_LastName,

Email = p\_Email,

RoleID = p\_RoleID

WHERE UserID = p\_UserID;

END //

DELIMITER ;

-- Delete a User

DELIMITER //

CREATE PROCEDURE DeleteUser (

IN p\_UserID INT

)

BEGIN

DELETE FROM User WHERE UserID = p\_UserID;

END //

DELIMITER ;

-- Add a Login Attempt

DELIMITER //

CREATE PROCEDURE AddLoginAttempt (

IN p\_UserID INT,

IN p\_SuccessStatus BOOLEAN,

IN p\_IPAddress VARCHAR(45)

)

BEGIN

INSERT INTO LoginAttempt (UserID, LoginTime, SuccessStatus, IPAddress)

VALUES (p\_UserID, CURRENT\_TIMESTAMP, p\_SuccessStatus, p\_IPAddress);

END //

DELIMITER ;

-- Retrieve Login Attempts by User ID

DELIMITER //

CREATE PROCEDURE GetLoginAttemptsByUserID (

IN p\_UserID INT

)

BEGIN

SELECT \* FROM LoginAttempt WHERE UserID = p\_UserID ORDER BY LoginTime DESC;

END //

DELIMITER ;

-- Add a Password Reset Request

DELIMITER //

CREATE PROCEDURE AddPasswordReset (

IN p\_UserID INT,

IN p\_ResetToken VARCHAR(255),

IN p\_ExpirationDate TIMESTAMP

)

BEGIN

INSERT INTO PasswordReset (ResetToken, UserID, ExpirationDate, ResetStatus)

VALUES (p\_ResetToken, p\_UserID, p\_ExpirationDate, 'Pending');

END //

DELIMITER ;

-- Retrieve Password Reset by Token

DELIMITER //

CREATE PROCEDURE GetPasswordResetByToken (

IN p\_ResetToken VARCHAR(255)

)

BEGIN

SELECT \* FROM PasswordReset WHERE ResetToken = p\_ResetToken;

END //

DELIMITER ;

-- Update Password Reset Status

DELIMITER //

CREATE PROCEDURE UpdatePasswordResetStatus (

IN p\_ResetToken VARCHAR(255),

IN p\_ResetStatus ENUM('Pending', 'Used', 'Expired')

)

BEGIN

UPDATE PasswordReset

SET ResetStatus = p\_ResetStatus

WHERE ResetToken = p\_ResetToken;

END //

DELIMITER ;

-- Add a new Patient

DELIMITER //

CREATE PROCEDURE AddPatient (

IN p\_UserID INT,

IN p\_DateOfBirth DATE,

IN p\_Gender ENUM('Male', 'Female', 'Other'),

IN p\_Address TEXT,

IN p\_ContactNumber VARCHAR(15),

IN p\_EmergencyContact VARCHAR(100),

IN p\_BloodType VARCHAR(5)

)

BEGIN

INSERT INTO Patient (UserID, DateOfBirth, Gender, Address, ContactNumber, EmergencyContact, BloodType)

VALUES (p\_UserID, p\_DateOfBirth, p\_Gender, p\_Address, p\_ContactNumber, p\_EmergencyContact, p\_BloodType);

END //

DELIMITER ;

-- Retrieve a Patient by ID

DELIMITER //

CREATE PROCEDURE GetPatientByID (

IN p\_PatientID INT

)

BEGIN

SELECT \* FROM Patient WHERE PatientID = p\_PatientID;

END //

DELIMITER ;

-- Update an existing Patient

DELIMITER //

CREATE PROCEDURE UpdatePatient (

IN p\_PatientID INT,

IN p\_DateOfBirth DATE,

IN p\_Gender ENUM('Male', 'Female', 'Other'),

IN p\_Address TEXT,

IN p\_ContactNumber VARCHAR(15),

IN p\_EmergencyContact VARCHAR(100),

IN p\_BloodType VARCHAR(5)

)

BEGIN

UPDATE Patient

SET DateOfBirth = p\_DateOfBirth,

Gender = p\_Gender,

Address = p\_Address,

ContactNumber = p\_ContactNumber,

EmergencyContact = p\_EmergencyContact,

BloodType = p\_BloodType

WHERE PatientID = p\_PatientID;

END //

DELIMITER ;

-- Delete a Patient

DELIMITER //

CREATE PROCEDURE DeletePatient (

IN p\_PatientID INT

)

BEGIN

DELETE FROM Patient WHERE PatientID = p\_PatientID;

END //

DELIMITER ;

-- Add a new Medical Record

DELIMITER //

CREATE PROCEDURE AddMedicalRecord (

IN p\_PatientID INT,

IN p\_MedicalHistory TEXT,

IN p\_Allergies TEXT

)

BEGIN

INSERT INTO MedicalRecord (PatientID, DateCreated, MedicalHistory, Allergies)

VALUES (p\_PatientID, CURRENT\_TIMESTAMP, p\_MedicalHistory, p\_Allergies);

END //

DELIMITER ;

-- Retrieve a Medical Record by ID

DELIMITER //

CREATE PROCEDURE GetMedicalRecordByID (

IN p\_RecordID INT

)

BEGIN

SELECT \* FROM MedicalRecord WHERE RecordID = p\_RecordID;

END //

DELIMITER ;

-- Update an existing Medical Record

DELIMITER //

CREATE PROCEDURE UpdateMedicalRecord (

IN p\_RecordID INT,

IN p\_MedicalHistory TEXT,

IN p\_Allergies TEXT

)

BEGIN

UPDATE MedicalRecord

SET MedicalHistory = p\_MedicalHistory,

Allergies = p\_Allergies,

LastUpdated = CURRENT\_TIMESTAMP

WHERE RecordID = p\_RecordID;

END //

DELIMITER ;

-- Delete a Medical Record

DELIMITER //

CREATE PROCEDURE DeleteMedicalRecord (

IN p\_RecordID INT

)

BEGIN

DELETE FROM MedicalRecord WHERE RecordID = p\_RecordID;

END //

DELIMITER ;

-- Add a new Department

DELIMITER //

CREATE PROCEDURE AddDepartment (

IN p\_DepartmentName VARCHAR(100),

IN p\_Location VARCHAR(100),

IN p\_ManagerID INT

)

BEGIN

INSERT INTO Department (DepartmentName, Location, ManagerID)

VALUES (p\_DepartmentName, p\_Location, p\_ManagerID);

END //

DELIMITER ;

-- Retrieve a Department by ID

DELIMITER //

CREATE PROCEDURE GetDepartmentByID (

IN p\_DepartmentID INT

)

BEGIN

SELECT \* FROM Department WHERE DepartmentID = p\_DepartmentID;

END //

DELIMITER ;

-- Update an existing Department

DELIMITER //

CREATE PROCEDURE UpdateDepartment (

IN p\_DepartmentID INT,

IN p\_DepartmentName VARCHAR(100),

IN p\_Location VARCHAR(100),

IN p\_ManagerID INT

)

BEGIN

UPDATE Department

SET DepartmentName = p\_DepartmentName,

Location = p\_Location,

ManagerID = p\_ManagerID

WHERE DepartmentID = p\_DepartmentID;

END //

DELIMITER ;

-- Delete a Department

DELIMITER //

CREATE PROCEDURE DeleteDepartment (

IN p\_DepartmentID INT

)

BEGIN

DELETE FROM Department WHERE DepartmentID = p\_DepartmentID;

END //

DELIMITER ;

-- Add a new Provider

DELIMITER //

CREATE PROCEDURE AddProvider (

IN p\_UserID INT,

IN p\_Specialization VARCHAR(100),

IN p\_LicenseNumber VARCHAR(50),

IN p\_DepartmentID INT

)

BEGIN

INSERT INTO Provider (UserID, Specialization, LicenseNumber, DepartmentID)

VALUES (p\_UserID, p\_Specialization, p\_LicenseNumber, p\_DepartmentID);

END //

DELIMITER ;

-- Retrieve a Provider by ID

DELIMITER //

CREATE PROCEDURE GetProviderByID (

IN p\_ProviderID INT

)

BEGIN

SELECT \* FROM Provider WHERE ProviderID = p\_ProviderID;

END //

DELIMITER ;

-- Update an existing Provider

DELIMITER //

CREATE PROCEDURE UpdateProvider (

IN p\_ProviderID INT,

IN p\_Specialization VARCHAR(100),

IN p\_LicenseNumber VARCHAR(50),

IN p\_DepartmentID INT

)

BEGIN

UPDATE Provider

SET Specialization = p\_Specialization,

LicenseNumber = p\_LicenseNumber,

DepartmentID = p\_DepartmentID

WHERE ProviderID = p\_ProviderID;

END //

DELIMITER ;

-- Delete a Provider

DELIMITER //

CREATE PROCEDURE DeleteProvider (

IN p\_ProviderID INT

)

BEGIN

DELETE FROM Provider WHERE ProviderID = p\_ProviderID;

END //

DELIMITER ;

-- Add a Duty Roster

DELIMITER //

CREATE PROCEDURE AddDutyRoster (

IN p\_ProviderID INT,

IN p\_ShiftDate DATE,

IN p\_ShiftStart TIME,

IN p\_ShiftEnd TIME,

IN p\_Status ENUM('Scheduled', 'On-Duty', 'Completed', 'Cancelled')

)

BEGIN

INSERT INTO DutyRoster (ProviderID, ShiftDate, ShiftStart, ShiftEnd, Status)

VALUES (p\_ProviderID, p\_ShiftDate, p\_ShiftStart, p\_ShiftEnd, p\_Status);

END //

DELIMITER ;

-- Retrieve a Duty Roster by ID

DELIMITER //

CREATE PROCEDURE GetDutyRosterByID (

IN p\_RosterID INT

)

BEGIN

SELECT \* FROM DutyRoster WHERE RosterID = p\_RosterID;

END //

DELIMITER ;

-- Update a Duty Roster

DELIMITER //

CREATE PROCEDURE UpdateDutyRoster (

IN p\_RosterID INT,

IN p\_ShiftDate DATE,

IN p\_ShiftStart TIME,

IN p\_ShiftEnd TIME,

IN p\_Status ENUM('Scheduled', 'On-Duty', 'Completed', 'Cancelled')

)

BEGIN

UPDATE DutyRoster

SET ShiftDate = p\_ShiftDate,

ShiftStart = p\_ShiftStart,

ShiftEnd = p\_ShiftEnd,

Status = p\_Status

WHERE RosterID = p\_RosterID;

END //

DELIMITER ;

-- Delete a Duty Roster

DELIMITER //

CREATE PROCEDURE DeleteDutyRoster (

IN p\_RosterID INT

)

BEGIN

DELETE FROM DutyRoster WHERE RosterID = p\_RosterID;

END //

DELIMITER ;

-- Add a new Room

DELIMITER //

CREATE PROCEDURE AddRoom (

IN p\_RoomNumber VARCHAR(10),

IN p\_RoomType VARCHAR(50),

IN p\_Floor INT,

IN p\_Status ENUM('Available', 'Occupied', 'Maintenance')

)

BEGIN

INSERT INTO Room (RoomNumber, RoomType, Floor, Status)

VALUES (p\_RoomNumber, p\_RoomType, p\_Floor, p\_Status);

END //

DELIMITER ;

-- Retrieve a Room by ID

DELIMITER //

CREATE PROCEDURE GetRoomByID (

IN p\_RoomID INT

)

BEGIN

SELECT \* FROM Room WHERE RoomID = p\_RoomID;

END //

DELIMITER ;

-- Update an existing Room

DELIMITER //

CREATE PROCEDURE UpdateRoom (

IN p\_RoomID INT,

IN p\_RoomNumber VARCHAR(10),

IN p\_RoomType VARCHAR(50),

IN p\_Floor INT,

IN p\_Status ENUM('Available', 'Occupied', 'Maintenance')

)

BEGIN

UPDATE Room

SET RoomNumber = p\_RoomNumber,

RoomType = p\_RoomType,

Floor = p\_Floor,

Status = p\_Status

WHERE RoomID = p\_RoomID;

END //

DELIMITER ;

-- Delete a Room

DELIMITER //

CREATE PROCEDURE DeleteRoom (

IN p\_RoomID INT

)

BEGIN

DELETE FROM Room WHERE RoomID = p\_RoomID;

END //

DELIMITER ;

-- Add a new Appointment Type

DELIMITER //

CREATE PROCEDURE AddAppointmentType (

IN p\_TypeName VARCHAR(50),

IN p\_Duration INT,

IN p\_Description TEXT

)

BEGIN

INSERT INTO AppointmentType (TypeName, Duration, Description)

VALUES (p\_TypeName, p\_Duration, p\_Description);

END //

DELIMITER ;

-- Retrieve an Appointment Type by ID

DELIMITER //

CREATE PROCEDURE GetAppointmentTypeByID (

IN p\_TypeID INT

)

BEGIN

SELECT \* FROM AppointmentType WHERE TypeID = p\_TypeID;

END //

DELIMITER ;

-- Update an existing Appointment Type

DELIMITER //

CREATE PROCEDURE UpdateAppointmentType (

IN p\_TypeID INT,

IN p\_TypeName VARCHAR(50),

IN p\_Duration INT,

IN p\_Description TEXT

)

BEGIN

UPDATE AppointmentType

SET TypeName = p\_TypeName,

Duration = p\_Duration,

Description = p\_Description

WHERE TypeID = p\_TypeID;

END //

DELIMITER ;

-- Delete an Appointment Type

DELIMITER //

CREATE PROCEDURE DeleteAppointmentType (

IN p\_TypeID INT

)

BEGIN

DELETE FROM AppointmentType WHERE TypeID = p\_TypeID;

END //

DELIMITER ;

-- Add a new Appointment

DELIMITER //

CREATE PROCEDURE AddAppointment (

IN p\_PatientID INT,

IN p\_ProviderID INT,

IN p\_RoomID INT,

IN p\_TypeID INT,

IN p\_AppointmentDate DATE,

IN p\_StartTime TIME,

IN p\_Status ENUM('Scheduled', 'Checked-In', 'Completed', 'Cancelled'),

IN p\_Notes TEXT

)

BEGIN

INSERT INTO Appointment (PatientID, ProviderID, RoomID, TypeID, AppointmentDate, StartTime, Status, Notes)

VALUES (p\_PatientID, p\_ProviderID, p\_RoomID, p\_TypeID, p\_AppointmentDate, p\_StartTime, p\_Status, p\_Notes);

END //

DELIMITER ;

-- Retrieve an Appointment by ID

DELIMITER //

CREATE PROCEDURE GetAppointmentByID (

IN p\_AppointmentID INT

)

BEGIN

SELECT \* FROM Appointment WHERE AppointmentID = p\_AppointmentID;

END //

DELIMITER ;

-- Update an existing Appointment

DELIMITER //

CREATE PROCEDURE UpdateAppointment (

IN p\_AppointmentID INT,

IN p\_PatientID INT,

IN p\_ProviderID INT,

IN p\_RoomID INT,

IN p\_TypeID INT,

IN p\_AppointmentDate DATE,

IN p\_StartTime TIME,

IN p\_Status ENUM('Scheduled', 'Checked-In', 'Completed', 'Cancelled'),

IN p\_Notes TEXT

)

BEGIN

UPDATE Appointment

SET PatientID = p\_PatientID,

ProviderID = p\_ProviderID,

RoomID = p\_RoomID,

TypeID = p\_TypeID,

AppointmentDate = p\_AppointmentDate,

StartTime = p\_StartTime,

Status = p\_Status,

Notes = p\_Notes

WHERE AppointmentID = p\_AppointmentID;

END //

DELIMITER ;

-- Delete an Appointment

DELIMITER //

CREATE PROCEDURE DeleteAppointment (

IN p\_AppointmentID INT

)

BEGIN

DELETE FROM Appointment WHERE AppointmentID = p\_AppointmentID;

END //

DELIMITER ;

-- Add a new Medication

DELIMITER //

CREATE PROCEDURE AddMedication (

IN p\_Name VARCHAR(100),

IN p\_GenericName VARCHAR(100),

IN p\_Category VARCHAR(50),

IN p\_Manufacturer VARCHAR(100),

IN p\_UnitPrice DECIMAL(10, 2)

)

BEGIN

INSERT INTO Medication (Name, GenericName, Category, Manufacturer, UnitPrice)

VALUES (p\_Name, p\_GenericName, p\_Category, p\_Manufacturer, p\_UnitPrice);

END //

DELIMITER ;

-- Retrieve a Medication by ID

DELIMITER //

CREATE PROCEDURE GetMedicationByID (

IN p\_MedicationID INT

)

BEGIN

SELECT \* FROM Medication WHERE MedicationID = p\_MedicationID;

END //

DELIMITER ;

-- Update an existing Medication

DELIMITER //

CREATE PROCEDURE UpdateMedication (

IN p\_MedicationID INT,

IN p\_Name VARCHAR(100),

IN p\_GenericName VARCHAR(100),

IN p\_Category VARCHAR(50),

IN p\_Manufacturer VARCHAR(100),

IN p\_UnitPrice DECIMAL(10, 2)

)

BEGIN

UPDATE Medication

SET Name = p\_Name,

GenericName = p\_GenericName,

Category = p\_Category,

Manufacturer = p\_Manufacturer,

UnitPrice = p\_UnitPrice

WHERE MedicationID = p\_MedicationID;

END //

DELIMITER ;

-- Delete a Medication

DELIMITER //

CREATE PROCEDURE DeleteMedication (

IN p\_MedicationID INT

)

BEGIN

DELETE FROM Medication WHERE MedicationID = p\_MedicationID;

END //

DELIMITER ;

-- Add a new Inventory Item

DELIMITER //

CREATE PROCEDURE AddInventory (

IN p\_MedicationID INT,

IN p\_BatchNumber VARCHAR(50),

IN p\_StockLevel INT,

IN p\_ExpiryDate DATE,

IN p\_Location VARCHAR(50)

)

BEGIN

INSERT INTO Inventory (MedicationID, BatchNumber, StockLevel, ExpiryDate, Location)

VALUES (p\_MedicationID, p\_BatchNumber, p\_StockLevel, p\_ExpiryDate, p\_Location);

END //

DELIMITER ;

-- Retrieve an Inventory Item by ID

DELIMITER //

CREATE PROCEDURE GetInventoryByID (

IN p\_InventoryID INT

)

BEGIN

SELECT \* FROM Inventory WHERE InventoryID = p\_InventoryID;

END //

DELIMITER ;

-- Update an existing Inventory Item

DELIMITER //

CREATE PROCEDURE UpdateInventory (

IN p\_InventoryID INT,

IN p\_MedicationID INT,

IN p\_BatchNumber VARCHAR(50),

IN p\_StockLevel INT,

IN p\_ExpiryDate DATE,

IN p\_Location VARCHAR(50)

)

BEGIN

UPDATE Inventory

SET MedicationID = p\_MedicationID,

BatchNumber = p\_BatchNumber,

StockLevel = p\_StockLevel,

ExpiryDate = p\_ExpiryDate,

Location = p\_Location

WHERE InventoryID = p\_InventoryID;

END //

DELIMITER ;

-- Delete an Inventory Item

DELIMITER //

CREATE PROCEDURE DeleteInventory (

IN p\_InventoryID INT

)

BEGIN

DELETE FROM Inventory WHERE InventoryID = p\_InventoryID;

END //

DELIMITER ;

-- Add a new Prescription

DELIMITER //

CREATE PROCEDURE AddPrescription (

IN p\_PatientID INT,

IN p\_ProviderID INT,

IN p\_Status ENUM('Active', 'Completed', 'Cancelled'),

IN p\_Notes TEXT

)

BEGIN

INSERT INTO Prescription (PatientID, ProviderID, Status, Notes)

VALUES (p\_PatientID, p\_ProviderID, p\_Status, p\_Notes);

END //

DELIMITER ;

-- Retrieve a Prescription by ID

DELIMITER //

CREATE PROCEDURE GetPrescriptionByID (

IN p\_PrescriptionID INT

)

BEGIN

SELECT \* FROM Prescription WHERE PrescriptionID = p\_PrescriptionID;

END //

DELIMITER ;

-- Update an existing Prescription

DELIMITER //

CREATE PROCEDURE UpdatePrescription (

IN p\_PrescriptionID INT,

IN p\_PatientID INT,

IN p\_ProviderID INT,

IN p\_Status ENUM('Active', 'Completed', 'Cancelled'),

IN p\_Notes TEXT

)

BEGIN

UPDATE Prescription

SET PatientID = p\_PatientID,

ProviderID = p\_ProviderID,

Status = p\_Status,

Notes = p\_Notes

WHERE PrescriptionID = p\_PrescriptionID;

END //

DELIMITER ;

-- Delete a Prescription

DELIMITER //

CREATE PROCEDURE DeletePrescription (

IN p\_PrescriptionID INT

)

BEGIN

DELETE FROM Prescription WHERE PrescriptionID = p\_PrescriptionID;

END //

DELIMITER ;

-- Add a Prescription Detail

DELIMITER //

CREATE PROCEDURE AddPrescriptionDetail (

IN p\_PrescriptionID INT,

IN p\_MedicationID INT,

IN p\_Dosage VARCHAR(50),

IN p\_Frequency VARCHAR(50),

IN p\_Duration INT,

IN p\_Quantity INT,

IN p\_Instructions TEXT

)

BEGIN

INSERT INTO PrescriptionDetail (PrescriptionID, MedicationID, Dosage, Frequency, Duration, Quantity, Instructions)

VALUES (p\_PrescriptionID, p\_MedicationID, p\_Dosage, p\_Frequency, p\_Duration, p\_Quantity, p\_Instructions);

END //

DELIMITER ;

-- Retrieve Prescription Details by ID

DELIMITER //

CREATE PROCEDURE GetPrescriptionDetailByID (

IN p\_DetailID INT

)

BEGIN

SELECT \* FROM PrescriptionDetail WHERE DetailID = p\_DetailID;

END //

DELIMITER ;

-- Update a Prescription Detail

DELIMITER //

CREATE PROCEDURE UpdatePrescriptionDetail (

IN p\_DetailID INT,

IN p\_PrescriptionID INT,

IN p\_MedicationID INT,

IN p\_Dosage VARCHAR(50),

IN p\_Frequency VARCHAR(50),

IN p\_Duration INT,

IN p\_Quantity INT,

IN p\_Instructions TEXT

)

BEGIN

UPDATE PrescriptionDetail

SET PrescriptionID = p\_PrescriptionID,

MedicationID = p\_MedicationID,

Dosage = p\_Dosage,

Frequency = p\_Frequency,

Duration = p\_Duration,

Quantity = p\_Quantity,

Instructions = p\_Instructions

WHERE DetailID = p\_DetailID;

END //

DELIMITER ;

-- Delete a Prescription Detail

DELIMITER //

CREATE PROCEDURE DeletePrescriptionDetail (

IN p\_DetailID INT

)

BEGIN

DELETE FROM PrescriptionDetail WHERE DetailID = p\_DetailID;

END //

DELIMITER ;

-- Add a new Lab Type

DELIMITER //

CREATE PROCEDURE AddLabType (

IN p\_TypeName VARCHAR(100),

IN p\_Description TEXT,

IN p\_ProcessingTime INT

)

BEGIN

INSERT INTO LabType (TypeName, Description, ProcessingTime)

VALUES (p\_TypeName, p\_Description, p\_ProcessingTime);

END //

DELIMITER ;

-- Retrieve a Lab Type by ID

DELIMITER //

CREATE PROCEDURE GetLabTypeByID (

IN p\_LabTypeID INT

)

BEGIN

SELECT \* FROM LabType WHERE LabTypeID = p\_LabTypeID;

END //

DELIMITER ;

-- Update an existing Lab Type

DELIMITER //

CREATE PROCEDURE UpdateLabType (

IN p\_LabTypeID INT,

IN p\_TypeName VARCHAR(100),

IN p\_Description TEXT,

IN p\_ProcessingTime INT

)

BEGIN

UPDATE LabType

SET TypeName = p\_TypeName,

Description = p\_Description,

ProcessingTime = p\_ProcessingTime

WHERE LabTypeID = p\_LabTypeID;

END //

DELIMITER ;

-- Delete a Lab Type

DELIMITER //

CREATE PROCEDURE DeleteLabType (

IN p\_LabTypeID INT

)

BEGIN

DELETE FROM LabType WHERE LabTypeID = p\_LabTypeID;

END //

DELIMITER ;

-- Add a new Lab Test

DELIMITER //

CREATE PROCEDURE AddLabTest (

IN p\_PatientID INT,

IN p\_ProviderID INT,

IN p\_LabTypeID INT,

IN p\_Status ENUM('Ordered', 'Sample-Collected', 'Processing', 'Completed', 'Cancelled'),

IN p\_Priority ENUM('Routine', 'Urgent', 'Emergency')

)

BEGIN

INSERT INTO LabTest (PatientID, ProviderID, LabTypeID, Status, Priority)

VALUES (p\_PatientID, p\_ProviderID, p\_LabTypeID, p\_Status, p\_Priority);

END //

DELIMITER ;

-- Retrieve a Lab Test by ID

DELIMITER //

CREATE PROCEDURE GetLabTestByID (

IN p\_LabTestID INT

)

BEGIN

SELECT \* FROM LabTest WHERE LabTestID = p\_LabTestID;

END //

DELIMITER ;

-- Update an existing Lab Test

DELIMITER //

CREATE PROCEDURE UpdateLabTest (

IN p\_LabTestID INT,

IN p\_PatientID INT,

IN p\_ProviderID INT,

IN p\_LabTypeID INT,

IN p\_Status ENUM('Ordered', 'Sample-Collected', 'Processing', 'Completed', 'Cancelled'),

IN p\_Priority ENUM('Routine', 'Urgent', 'Emergency')

)

BEGIN

UPDATE LabTest

SET PatientID = p\_PatientID,

ProviderID = p\_ProviderID,

LabTypeID = p\_LabTypeID,

Status = p\_Status,

Priority = p\_Priority

WHERE LabTestID = p\_LabTestID;

END //

DELIMITER ;

-- Delete a Lab Test

DELIMITER //

CREATE PROCEDURE DeleteLabTest (

IN p\_LabTestID INT

)

BEGIN

DELETE FROM LabTest WHERE LabTestID = p\_LabTestID;

END //

DELIMITER ;

-- Add a new Lab Result

DELIMITER //

CREATE PROCEDURE AddLabResult (

IN p\_LabTestID INT,

IN p\_ResultValue TEXT,

IN p\_ReferenceRange VARCHAR(100),

IN p\_Interpretation TEXT,

IN p\_TechnicianID INT

)

BEGIN

INSERT INTO LabResult (LabTestID, ResultValue, ReferenceRange, Interpretation, TechnicianID)

VALUES (p\_LabTestID, p\_ResultValue, p\_ReferenceRange, p\_Interpretation, p\_TechnicianID);

END //

DELIMITER ;

-- Retrieve a Lab Result by ID

DELIMITER //

CREATE PROCEDURE GetLabResultByID (

IN p\_ResultID INT

)

BEGIN

SELECT \* FROM LabResult WHERE ResultID = p\_ResultID;

END //

DELIMITER ;

-- Update an existing Lab Result

DELIMITER //

CREATE PROCEDURE UpdateLabResult (

IN p\_ResultID INT,

IN p\_ResultValue TEXT,

IN p\_ReferenceRange VARCHAR(100),

IN p\_Interpretation TEXT,

IN p\_TechnicianID INT

)

BEGIN

UPDATE LabResult

SET ResultValue = p\_ResultValue,

ReferenceRange = p\_ReferenceRange,

Interpretation = p\_Interpretation,

TechnicianID = p\_TechnicianID,

ResultDate = CURRENT\_TIMESTAMP

WHERE ResultID = p\_ResultID;

END //

DELIMITER ;

-- Delete a Lab Result

DELIMITER //

CREATE PROCEDURE DeleteLabResult (

IN p\_ResultID INT

)

BEGIN

DELETE FROM LabResult WHERE ResultID = p\_ResultID;

END //

DELIMITER ;

-- Add a new Insurance

DELIMITER //

CREATE PROCEDURE AddInsurance (

IN p\_ProviderName VARCHAR(100),

IN p\_ContactInfo TEXT,

IN p\_PolicyDetails TEXT

)

BEGIN

INSERT INTO Insurance (ProviderName, ContactInfo, PolicyDetails)

VALUES (p\_ProviderName, p\_ContactInfo, p\_PolicyDetails);

END //

DELIMITER ;

-- Retrieve an Insurance by ID

DELIMITER //

CREATE PROCEDURE GetInsuranceByID (

IN p\_InsuranceID INT

)

BEGIN

SELECT \* FROM Insurance WHERE InsuranceID = p\_InsuranceID;

END //

DELIMITER ;

-- Update an existing Insurance

DELIMITER //

CREATE PROCEDURE UpdateInsurance (

IN p\_InsuranceID INT,

IN p\_ProviderName VARCHAR(100),

IN p\_ContactInfo TEXT,

IN p\_PolicyDetails TEXT

)

BEGIN

UPDATE Insurance

SET ProviderName = p\_ProviderName,

ContactInfo = p\_ContactInfo,

PolicyDetails = p\_PolicyDetails

WHERE InsuranceID = p\_InsuranceID;

END //

DELIMITER ;

-- Delete an Insurance

DELIMITER //

CREATE PROCEDURE DeleteInsurance (

IN p\_InsuranceID INT

)

BEGIN

DELETE FROM Insurance WHERE InsuranceID = p\_InsuranceID;

END //

DELIMITER ;

-- Add a new Patient Insurance

DELIMITER //

CREATE PROCEDURE AddPatientInsurance (

IN p\_PatientID INT,

IN p\_InsuranceID INT,

IN p\_PolicyNumber VARCHAR(50),

IN p\_StartDate DATE,

IN p\_EndDate DATE

)

BEGIN

INSERT INTO PatientInsurance (PatientID, InsuranceID, PolicyNumber, StartDate, EndDate)

VALUES (p\_PatientID, p\_InsuranceID, p\_PolicyNumber, p\_StartDate, p\_EndDate);

END //

DELIMITER ;

-- Retrieve a Patient Insurance by ID

DELIMITER //

CREATE PROCEDURE GetPatientInsuranceByID (

IN p\_PatientInsuranceID INT

)

BEGIN

SELECT \* FROM PatientInsurance WHERE PatientInsuranceID = p\_PatientInsuranceID;

END //

DELIMITER ;

-- Update an existing Patient Insurance

DELIMITER //

CREATE PROCEDURE UpdatePatientInsurance (

IN p\_PatientInsuranceID INT,

IN p\_PatientID INT,

IN p\_InsuranceID INT,

IN p\_PolicyNumber VARCHAR(50),

IN p\_StartDate DATE,

IN p\_EndDate DATE

)

BEGIN

UPDATE PatientInsurance

SET PatientID = p\_PatientID,

InsuranceID = p\_InsuranceID,

PolicyNumber = p\_PolicyNumber,

StartDate = p\_StartDate,

EndDate = p\_EndDate

WHERE PatientInsuranceID = p\_PatientInsuranceID;

END //

DELIMITER ;

-- Delete a Patient Insurance

DELIMITER //

CREATE PROCEDURE DeletePatientInsurance (

IN p\_PatientInsuranceID INT

)

BEGIN

DELETE FROM PatientInsurance WHERE PatientInsuranceID = p\_PatientInsuranceID;

END //

DELIMITER ;

-- Add a new Bill

DELIMITER //

CREATE PROCEDURE AddBill (

IN p\_PatientID INT,

IN p\_GeneratedDate TIMESTAMP,

IN p\_DueDate DATE,

IN p\_TotalAmount DECIMAL(10, 2),

IN p\_Status ENUM('Pending', 'Paid', 'Overdue', 'Cancelled')

)

BEGIN

INSERT INTO Bill (PatientID, GeneratedDate, DueDate, TotalAmount, Status)

VALUES (p\_PatientID, p\_GeneratedDate, p\_DueDate, p\_TotalAmount, p\_Status);

END //

DELIMITER ;

-- Retrieve a Bill by ID

DELIMITER //

CREATE PROCEDURE GetBillByID (

IN p\_BillID INT

)

BEGIN

SELECT \* FROM Bill WHERE BillID = p\_BillID;

END //

DELIMITER ;

-- Update an existing Bill

DELIMITER //

CREATE PROCEDURE UpdateBill (

IN p\_BillID INT,

IN p\_PatientID INT,

IN p\_GeneratedDate TIMESTAMP,

IN p\_DueDate DATE,

IN p\_TotalAmount DECIMAL(10, 2),

IN p\_Status ENUM('Pending', 'Paid', 'Overdue', 'Cancelled')

)

BEGIN

UPDATE Bill

SET PatientID = p\_PatientID,

GeneratedDate = p\_GeneratedDate,

DueDate = p\_DueDate,

TotalAmount = p\_TotalAmount,

Status = p\_Status

WHERE BillID = p\_BillID;

END //

DELIMITER ;

-- Delete a Bill

DELIMITER //

CREATE PROCEDURE DeleteBill (

IN p\_BillID INT

)

BEGIN

DELETE FROM Bill WHERE BillID = p\_BillID;

END //

DELIMITER ;

-- Add a new Payment Method

DELIMITER //

CREATE PROCEDURE AddPaymentMethod (

IN p\_MethodName VARCHAR(50),

IN p\_Description TEXT

)

BEGIN

INSERT INTO PaymentMethod (MethodName, Description)

VALUES (p\_MethodName, p\_Description);

END //

DELIMITER ;

-- Retrieve a Payment Method by ID

DELIMITER //

CREATE PROCEDURE GetPaymentMethodByID (

IN p\_MethodID INT

)

BEGIN

SELECT \* FROM PaymentMethod WHERE MethodID = p\_MethodID;

END //

DELIMITER ;

-- Update an existing Payment Method

DELIMITER //

CREATE PROCEDURE UpdatePaymentMethod (

IN p\_MethodID INT,

IN p\_MethodName VARCHAR(50),

IN p\_Description TEXT

)

BEGIN

UPDATE PaymentMethod

SET MethodName = p\_MethodName,

Description = p\_Description

WHERE MethodID = p\_MethodID;

END //

DELIMITER ;

-- Delete a Payment Method

DELIMITER //

CREATE PROCEDURE DeletePaymentMethod (

IN p\_MethodID INT

)

BEGIN

DELETE FROM PaymentMethod WHERE MethodID = p\_MethodID;

END //

DELIMITER ;

-- Add a new Payment

DELIMITER //

CREATE PROCEDURE AddPayment (

IN p\_BillID INT,

IN p\_Amount DECIMAL(10, 2),

IN p\_PaymentDate TIMESTAMP,

IN p\_MethodID INT,

IN p\_TransactionReference VARCHAR(100),

IN p\_Status ENUM('Pending', 'Completed', 'Failed')

)

BEGIN

INSERT INTO Payment (BillID, Amount, PaymentDate, MethodID, TransactionReference, Status)

VALUES (p\_BillID, p\_Amount, p\_PaymentDate, p\_MethodID, p\_TransactionReference, p\_Status);

END //

DELIMITER ;

-- Retrieve a Payment by ID

DELIMITER //

CREATE PROCEDURE GetPaymentByID (

IN p\_PaymentID INT

)

BEGIN

SELECT \* FROM Payment WHERE PaymentID = p\_PaymentID;

END //

DELIMITER ;

-- Update an existing Payment

DELIMITER //

CREATE PROCEDURE UpdatePayment (

IN p\_PaymentID INT,

IN p\_BillID INT,

IN p\_Amount DECIMAL(10, 2),

IN p\_PaymentDate TIMESTAMP,

IN p\_MethodID INT,

IN p\_TransactionReference VARCHAR(100),

IN p\_Status ENUM('Pending', 'Completed', 'Failed')

)

BEGIN

UPDATE Payment

SET BillID = p\_BillID,

Amount = p\_Amount,

PaymentDate = p\_PaymentDate,

MethodID = p\_MethodID,

TransactionReference = p\_TransactionReference,

Status = p\_Status

WHERE PaymentID = p\_PaymentID;

END //

DELIMITER ;

-- Delete a Payment

DELIMITER //

CREATE PROCEDURE DeletePayment (

IN p\_PaymentID INT

)

BEGIN

DELETE FROM Payment WHERE PaymentID = p\_PaymentID;

END //

DELIMITER ;

-- Add a new Notification Type

DELIMITER //

CREATE PROCEDURE AddNotificationType (

IN p\_TypeName VARCHAR(50),

IN p\_Description TEXT,

IN p\_Template TEXT

)

BEGIN

INSERT INTO NotificationType (TypeName, Description, Template)

VALUES (p\_TypeName, p\_Description, p\_Template);

END //

DELIMITER ;

-- Retrieve a Notification Type by ID

DELIMITER //

CREATE PROCEDURE GetNotificationTypeByID (

IN p\_TypeID INT

)

BEGIN

SELECT \* FROM NotificationType WHERE TypeID = p\_TypeID;

END //

DELIMITER ;

-- Update an existing Notification Type

DELIMITER //

CREATE PROCEDURE UpdateNotificationType (

IN p\_TypeID INT,

IN p\_TypeName VARCHAR(50),

IN p\_Description TEXT,

IN p\_Template TEXT

)

BEGIN

UPDATE NotificationType

SET TypeName = p\_TypeName,

Description = p\_Description,

Template = p\_Template

WHERE TypeID = p\_TypeID;

END //

DELIMITER ;

-- Delete a Notification Type

DELIMITER //

CREATE PROCEDURE DeleteNotificationType (

IN p\_TypeID INT

)

BEGIN

DELETE FROM NotificationType WHERE TypeID = p\_TypeID;

END //

DELIMITER ;

-- Add a new Notification

DELIMITER //

CREATE PROCEDURE AddNotification (

IN p\_TypeID INT,

IN p\_UserID INT,

IN p\_Title VARCHAR(100),

IN p\_Message TEXT,

IN p\_Status ENUM('Pending', 'Sent', 'Read', 'Failed')

)

BEGIN

INSERT INTO Notification (TypeID, UserID, Title, Message, Status)

VALUES (p\_TypeID, p\_UserID, p\_Title, p\_Message, p\_Status);

END //

DELIMITER ;

-- Retrieve a Notification by ID

DELIMITER //

CREATE PROCEDURE GetNotificationByID (

IN p\_NotificationID INT

)

BEGIN

SELECT \* FROM Notification WHERE NotificationID = p\_NotificationID;

END //

DELIMITER ;

-- Update an existing Notification

DELIMITER //

CREATE PROCEDURE UpdateNotification (

IN p\_NotificationID INT,

IN p\_TypeID INT,

IN p\_UserID INT,

IN p\_Title VARCHAR(100),

IN p\_Message TEXT,

IN p\_Status ENUM('Pending', 'Sent', 'Read', 'Failed'),

IN p\_ReadDate TIMESTAMP

)

BEGIN

UPDATE Notification

SET TypeID = p\_TypeID,

UserID = p\_UserID,

Title = p\_Title,

Message = p\_Message,

Status = p\_Status,

ReadDate = p\_ReadDate

WHERE NotificationID = p\_NotificationID;

END //

DELIMITER ;

-- Delete a Notification

DELIMITER //

CREATE PROCEDURE DeleteNotification (

IN p\_NotificationID INT

)

BEGIN

DELETE FROM Notification WHERE NotificationID = p\_NotificationID;

END //

DELIMITER ;

-- Add a new Survey

DELIMITER //

CREATE PROCEDURE AddSurvey (

IN p\_Title VARCHAR(100),

IN p\_Description TEXT,

IN p\_StartDate DATE,

IN p\_EndDate DATE,

IN p\_Status ENUM('Draft', 'Active', 'Closed')

)

BEGIN

INSERT INTO Survey (Title, Description, StartDate, EndDate, Status)

VALUES (p\_Title, p\_Description, p\_StartDate, p\_EndDate, p\_Status);

END //

DELIMITER ;

-- Retrieve a Survey by ID

DELIMITER //

CREATE PROCEDURE GetSurveyByID (

IN p\_SurveyID INT

)

BEGIN

SELECT \* FROM Survey WHERE SurveyID = p\_SurveyID;

END //

DELIMITER ;

-- Update an existing Survey

DELIMITER //

CREATE PROCEDURE UpdateSurvey (

IN p\_SurveyID INT,

IN p\_Title VARCHAR(100),

IN p\_Description TEXT,

IN p\_StartDate DATE,

IN p\_EndDate DATE,

IN p\_Status ENUM('Draft', 'Active', 'Closed')

)

BEGIN

UPDATE Survey

SET Title = p\_Title,

Description = p\_Description,

StartDate = p\_StartDate,

EndDate = p\_EndDate,

Status = p\_Status

WHERE SurveyID = p\_SurveyID;

END //

DELIMITER ;

-- Delete a Survey

DELIMITER //

CREATE PROCEDURE DeleteSurvey (

IN p\_SurveyID INT

)

BEGIN

DELETE FROM Survey WHERE SurveyID = p\_SurveyID;

END //

DELIMITER ;

-- Add a new Survey Response

DELIMITER //

CREATE PROCEDURE AddSurveyResponse (

IN p\_SurveyID INT,

IN p\_PatientID INT,

IN p\_Responses TEXT

)

BEGIN

INSERT INTO SurveyResponse (SurveyID, PatientID, ResponseDate, Responses)

VALUES (p\_SurveyID, p\_PatientID, CURRENT\_TIMESTAMP, p\_Responses);

END //

DELIMITER ;

-- Retrieve a Survey Response by ID

DELIMITER //

CREATE PROCEDURE GetSurveyResponseByID (

IN p\_ResponseID INT

)

BEGIN

SELECT \* FROM SurveyResponse WHERE ResponseID = p\_ResponseID;

END //

DELIMITER ;

-- Update an existing Survey Response

DELIMITER //

CREATE PROCEDURE UpdateSurveyResponse (

IN p\_ResponseID INT,

IN p\_SurveyID INT,

IN p\_PatientID INT,

IN p\_Responses TEXT

)

BEGIN

UPDATE SurveyResponse

SET SurveyID = p\_SurveyID,

PatientID = p\_PatientID,

Responses = p\_Responses,

ResponseDate = CURRENT\_TIMESTAMP

WHERE ResponseID = p\_ResponseID;

END //

DELIMITER ;

-- Delete a Survey Response

DELIMITER //

CREATE PROCEDURE DeleteSurveyResponse (

IN p\_ResponseID INT

)

BEGIN

DELETE FROM SurveyResponse WHERE ResponseID = p\_ResponseID;

END //

DELIMITER ;

-- Add a new Event

DELIMITER //

CREATE PROCEDURE AddEvent (

IN p\_EventType ENUM('Reminder', 'Follow-up', 'Survey', 'Alert'),

IN p\_RelatedID INT,

IN p\_UserID INT,

IN p\_EventDate TIMESTAMP,

IN p\_Description TEXT,

IN p\_Status ENUM('Scheduled', 'Triggered', 'Completed', 'Cancelled')

)

BEGIN

INSERT INTO Event (EventType, RelatedID, UserID, EventDate, Description, Status)

VALUES (p\_EventType, p\_RelatedID, p\_UserID, p\_EventDate, p\_Description, p\_Status);

END //

DELIMITER ;

-- Retrieve an Event by ID

DELIMITER //

CREATE PROCEDURE GetEventByID (

IN p\_EventID INT

)

BEGIN

SELECT \* FROM Event WHERE EventID = p\_EventID;

END //

DELIMITER ;

-- Update an existing Event

DELIMITER //

CREATE PROCEDURE UpdateEvent (

IN p\_EventID INT,

IN p\_EventType ENUM('Reminder', 'Follow-up', 'Survey', 'Alert'),

IN p\_RelatedID INT,

IN p\_UserID INT,

IN p\_EventDate TIMESTAMP,

IN p\_Description TEXT,

IN p\_Status ENUM('Scheduled', 'Triggered', 'Completed', 'Cancelled')

)

BEGIN

UPDATE Event

SET EventType = p\_EventType,

RelatedID = p\_RelatedID,

UserID = p\_UserID,

EventDate = p\_EventDate,

Description = p\_Description,

Status = p\_Status

WHERE EventID = p\_EventID;

END //

DELIMITER ;

-- Delete an Event

DELIMITER //

CREATE PROCEDURE DeleteEvent (

IN p\_EventID INT

)

BEGIN

DELETE FROM Event WHERE EventID = p\_EventID;

END //

DELIMITER ;