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 ;
```