EMRge System Documentation

EMRge is a comprehensive software solution designed to efficiently manage the operations of a hospital or healthcare facility. It integrates electronic medical records (EMR) management with various administrative, clinical, and financial functionalities to streamline hospital operations, enhance patient care, and improve overall efficiency.

Running the Application

To run the EMRge application, follow these steps:

1. **Setup Database Connection:**
    - Ensure that you have a SQL Server instance available.
    - Modify the **EMRgeDB.connString** property to point to your SQL Server instance in the **EMRgeDB** class (not provided in this code snippet).
2. **Build the Application:**
    - Create a new C# project in your preferred development environment (e.g., Visual Studio).
    - Copy and paste the desired class into your project's namespace.
    - Configure the project to reference necessary assemblies (e.g., System, System.Data.SqlClient).
3. **Database Setup:**
    - Set up the required database schema and tables
    - Implement the necessary stored procedures, such as **GetAllEmployees**, **GenerateEmployeeID**, **InsertMedicationRecord**, **InsertLaboratoryRecord**, etc
4. **Interact with Employee Class:**
    - Instantiate the **desired** class to manage employee data.
    - Utilize methods to retrieve and manipulate employee records.
5. **Running the Application:**
    - Compile and run the application.
    - Use appropriate UI or console interfaces to interact with the methods.

Employee Class

This documentation covers the **Employee** class, a crucial part of the EMRge application. The **Employee** class encapsulates employee information and provides methods for database interaction.

The **Employee** class represents an employee in the EMRge system. It contains properties to store various details about an employee, such as their ID, name, department, contact information, salary, and more.

Properties

- **ID**: Employee's unique identifier.
- **Department**: Employee's department within the organization.
- **FirstName**: First name of the employee.
- **MiddleName**: Middle name of the employee.
- **LastName**: Last name of the employee.
- **Gender**: Gender of the employee.
- **DateOfBirth**: Date of birth of the employee.
- **Email**: Email address of the employee.
- **PhoneNumber**: Contact phone number of the employee.
- **Address**: Address of the employee.
- **Nationality**: Nationality of the employee.
- **Branch**: Branch or location where the employee works.
- **Salary**: Salary of the employee.
- **DateJoined**: Date when the employee joined the organization.
- **CurrentlyWorking**: Boolean indicating whether the employee is currently employed.
- **ECName**: Name of the employee's emergency contact person.
- **ECRelation**: Relationship of the emergency contact person to the employee.
- **ECPhoneNumber**: Phone number of the emergency contact person.
- **ECEmail**: Email address of the emergency contact person.
- **IsDeleted**: Boolean indicating if the employee record is deleted (default: false).

Methods

1. **GetAllEmployees**
    - Parameters: Search criteria (SearchKey, Branch, Department, MinSalary, MaxSalary, SortBy, Ascending)

- Returns: List of **Employee** objects
- Description: Retrieves a list of employees based on specified search criteria.

2. **GenerateEmployeeID**
    - Returns: Newly generated employee ID as a string
    - Description: Generates a new unique employee ID.
3. **Insert**
    - Description: Inserts the current employee instance into the database.
    - Note: This method interacts with the database using the **InsertEmployee** stored procedure.
4. **GetCredentials**
    - Parameters: Employee ID
    - Returns: Array of integers representing employee's credentials
    - Description: Retrieves an array of credentials associated with the specified employee.
5. **SetDefaultCredentials**
    - Description: Sets default credentials for the employee based on their department.
    - Note: This method interacts with the database using the **EmployeeCredentials** table.
6. **UpdateCredentials**
    - Parameters: Array of integers representing updated credentials
    - Description: Updates the credentials for the employee in the database.
    - Note: This method interacts with the database using the **EmployeeCredentials** table.

ExaminationRecord Class

This documentation covers the **ExaminationRecord** class, a key component of the application. The **ExaminationRecord** class encapsulates examination record information and provides methods for database interaction.

The **ExaminationRecord** class represents a medical examination record within the system. It contains properties to store details about an examination, such as the examination ID, date and time, patient information, physician information, diagnosis, and removal status from the examination queue.

Properties

- **ID**: Unique identifier for the examination record.

- **DateAndTime**: Date and time when the examination took place.
- **PatientID**: ID of the patient associated with the examination.
- **PhysicianID**: ID of the physician who conducted the examination.
- **Diagnosis**: Diagnosis provided during the examination.
- **IsRemovedFromQueue**: Boolean indicating whether the examination record is removed from the examination queue (default: false).

Methods

1. **Insert**
    - Description: Inserts a new examination record into the database.
    - Note: This method interacts with the database using the **InsertExaminationRecord** stored procedure.
    - Parameters: None
    - Returns: None
2. **GetExaminationRecords**
    - Description: Retrieves a list of examination records that are not removed from the queue from the database.
    - Note: This method interacts with the database using the **ExaminationRecord** table.
    - Parameters: None
    - Returns: List of **ExaminationRecord** objects

LaboratoryTestRecord Class

This documentation covers the **LaboratoryTestRecord** class, a crucial component of the application. The **LaboratoryTestRecord** class encapsulates laboratory test record information and provides methods for database interaction.

The **LaboratoryTestRecord** class represents a laboratory test record within the system. It contains properties to store details about a laboratory test, including the test ID, associated examination record ID, date and time, patient information, lab physician information, test type, result, and removal status from the test queue.

Properties

- **ID**: Unique identifier for the laboratory test record.
- **ExaminationRecordID**: ID of the associated examination record for which the test was performed.

- **DateAndTime**: Date and time when the laboratory test took place.
- **PatientID**: ID of the patient for whom the test was performed.
- **LabPhysicianID**: ID of the lab physician who conducted the test.
- **TestType**: Type of the laboratory test.
- **Result**: Result of the laboratory test.
- **IsRemovedFromQueue**: Boolean indicating whether the test record is removed from the test queue (default: false).

Methods

1. **Insert**
    - Description: Inserts a new laboratory test record into the database.
    - Note: This method interacts with the database using the **InsertLaboratoryRecord** stored procedure.
    - Parameters: None
    - Returns: None
2. **GetLaboratoryTestRecords**
    - Description: Retrieves a list of laboratory test records from the database.
    - Note: This method interacts with the database using the **LaboratoryTestRecord** table.
    - Parameters: None
    - Returns: List of **LaboratoryTestRecord** objects

MedicationRecord Class

This documentation covers the **MedicationRecord** class, a key component of the application. The **MedicationRecord** class encapsulates medication record information and provides methods for database interaction.

The **MedicationRecord** class represents a medication record within the system. It holds properties to store details about a medication record, including the record ID, associated examination record ID, prescribed medication, and an alternative medication if necessary.

Properties

- **ID**: Unique identifier for the medication record.
- **ExaminationRecordID**: ID of the associated examination record for which the medication was prescribed.

- **Medication**: The prescribed medication.
- **AlternativeMedication**: An alternative medication that can be used if needed.

Methods

1. **Insert**
    - Description: Inserts a new medication record into the database.
    - Note: This method interacts with the database using the **InsertMedicationRecord** stored procedure.
    - Parameters: None
    - Returns: None
2. **GetMedicationRecords**
    - Description: Retrieves a list of medication records from the database.
    - Note: This method interacts with the database using the **MedicationRecord** table.
    - Parameters: None
    - Returns: List of **MedicationRecord** objects

Patient Class

This documentation covers the **Patient** class, a key component of the application. The **Patient** class encapsulates patient information and provides methods for database interaction.

The **Patient** class represents a patient within the system. It holds properties to store various details about a patient, including personal information, health information, emergency contact details, and registration details.

Properties

- **ID**: Unique identifier for the patient.
- **FirstName**, **MiddleName**, **LastName**: Patient's name components.
- **Gender**: Gender of the patient.
- **DateOfBirth**: Date of birth of the patient.
- **Email**, **PhoneNumber**: Contact information of the patient.
- **Address**: Address of the patient.
- **Nationality**: Nationality of the patient.
- **DateRegistered**: Date when the patient was registered in the system.

- **BranchRegistered**: The branch where the patient was registered.
- **IsInPatient**: Indicates if the patient is an in-patient.
- **RoomNumber**: Room number for in-patients.
- **ECName**, **ECRelation**, **ECPhoneNumber**, **ECEmail**: Emergency contact information.
- **BloodType**: Blood type of the patient.
- **Height**, **Weight**: Health information about height and weight.
- **IsDeleted**: Indicates if the patient record is deleted.

Methods

1. **GetAllInPatients**
    - Description: Retrieves a list of all in-patient records from the database.
    - Note: This method interacts with the database using the **GetAllInPatients** stored procedure.
    - Parameters: None
    - Returns: List of **Patient** objects representing in-patients.
2. **GetAllOutPatients**
    - Description: Retrieves a list of all out-patient records from the database.
    - Note: This method interacts with the database using the **GetAllOutPatients** stored procedure.
    - Parameters: None
    - Returns: List of **Patient** objects representing out-patients.
3. **GeneratePatientID**
    - Description: Generates a unique patient ID.
    - Note: This method interacts with the database using the **GeneratePatientID** stored procedure.
    - Parameters: None
    - Returns: A unique patient ID.
4. **Insert**
    - Description: Inserts a new patient record into the database.
    - Note: This method interacts with the database using the **InsertPatient** stored procedure.
    - Parameters: None

- Returns: None

5. **Update**
    - Description: Updates an existing patient record in the database.
    - Note: This method interacts with the database using the **UpdatePatient** stored procedure.
    - Parameters: An updated **Patient** object
    - Returns: None
6. **Delete**
    - Description: Deletes a patient record from the database.
    - Note: This method interacts with the database using the **DeletePatient** stored procedure.
    - Parameters: None
    - Returns: None
7. **FindByID**
    - Description: Retrieves a patient record by ID from the database.
    - Note: This method interacts with the database using the **FindPatientByID** function.
    - Parameters: Patient ID
    - Returns: A **Patient** object representing the found patient or null if not found.
8. **FindByPhoneNumber**
    - Description: Retrieves a patient record by phone number from the database.
    - Note: This method interacts with the database using the **FindPatientByPhoneNumber** function.
    - Parameters: Phone number
    - Returns: A **Patient** object representing the found patient or null if not found.
9. **GetRecentExaminationRecord**
    - Description: Retrieves the most recent examination record for the patient.
    - Note: This method interacts with the database to retrieve the latest examination record.
    - Parameters: None
    - Returns: An **ExaminationRecord** object representing the most recent examination or null if not found.

EMRge Database Setup and Tables

1. **Database Creation and Selection:**
    - Create a new database named "EMRge".
    - Set the context to the "EMRge" database.
2. **Patient Table:**
    - This table stores patient information, including demographics, medical history, emergency contacts, and more.
3. **Employee Table:**
    - This table stores employee information, including personal details, department, branch, and salary.
4. **ExaminationRecord Table:**
    - Stores details of patient examinations performed by physicians.
5. **LaboratoryTestRecord Table:**
    - Records laboratory test information for patients, including test type and results.
6. **MedicationRecord Table:**
    - Tracks medication details associated with patient examinations.
7. **EmployeeCredentials Table:**
    - Stores employee credentials and access permissions.
8. **Bill Table:**
    - Records billing information for services provided to patients.

Patient Functions and Procedures

1. **GeneratePatientID Procedure:**
    - Generates a unique patient ID starting with "PAT-" and followed by a random identifier.
2. **GetAllOutPatients Procedure:**
    - Retrieves details of all out-patients (non-admitted patients) who are not marked as deleted.
3. **GetAllInPatients Procedure:**
    - Retrieves details of all in-patients (admitted patients) who are not marked as deleted.
4. **InsertPatient Procedure:**
    - Inserts a new patient record into the Patient table with provided details.

5. **UpdatePatient Procedure:**
    - Updates an existing patient's information based on their ID.
6. **DeletePatient Procedure:**
    - Marks a patient as deleted by setting the IsDeleted flag to 1.
7. **FindPatientByID Function:**
    - Retrieves patient details based on the provided patient ID.
8. **FindPatientByPhoneNumber Function:**
    - Retrieves patient details based on the provided phone number.

Employee Functions and Procedures

1. **GetAllEmployees Procedure:**
    - Retrieves a list of employees based on search criteria like branch, department, salary range, etc.
2. **GenerateEmployeeID Procedure:**
    - Generates a unique employee ID starting with "EMP-" followed by a random identifier.
3. **InsertEmployee Procedure:**
    - Inserts a new employee record into the Employee table with provided details.

Examination, Lab, and Medication Insert Procedures

1. **InsertExaminationRecord Procedure:**
    - Inserts a new examination record into the ExaminationRecord table.
2. **InsertLaboratoryRecord Procedure:**
    - Inserts a new laboratory test record into the LaboratoryTestRecord table.
3. **InsertMedicationRecord Procedure:**
    - Inserts a new medication record into the MedicationRecord table.

Bill Triggers

1. **RegistrationBill Trigger:**
    - Automatically generates a billing record for patient registration.
2. **ExaminationBill Trigger:**
    - Automatically generates a billing record for out-patient examinations.
3. **LaboratoryTestBill Trigger:**

- Automatically generates a billing record for laboratory tests.

4. **MedicationBill Trigger:**
    - Automatically generates a billing record for medications prescribed during examinations.

Instructions for Running on Local Machine

1. **Database Setup:**
    - Execute the SQL script in a SQL Server Management Studio (SSMS) or similar tool.
    - The script will create the "EMRge" database and all the necessary tables, procedures, functions, and triggers.
2. **Using Procedures:**
    - You can call the procedures like "GeneratePatientID," "InsertPatient," "UpdatePatient," "DeletePatient," "GenerateEmployeeID," "InsertEmployee," and the others by providing the required parameters.
3. **Using Functions:**
    - You can use the functions like "FindPatientByID" and "FindPatientByPhoneNumber" in SQL queries to retrieve patient information.
4. **Triggers:**
    - Triggers are automatically executed when certain events occur (e.g., after inserting records).
    - The triggers automatically generate billing records based on specific events.