



reLIFE Scheduler - Application Documentation

Supervised by:

Mr. Ako

TISHK INTERNATIONAL UNIVERSITY

May 31, 2025

Department of Computer Engineering

Authors: Ahmed Emad & Natheer Nihad & Bavin Kawa

Part I: User Documentation

1. Prerequisites & Development Setup

This section outlines the necessary software, tools, and libraries required to compile, run, and further develop the reLIFE Scheduler application from its source code.

1.1. Software Requirements:

- **Microsoft Visual Studio:** Version [e.g., 2022] or later, with the ".NET desktop development" workload installed.
- **.NET SDK:** Version [8.0] (matching the project's target framework). This is typically included with Visual Studio.
- **Microsoft SQL Server:** A running instance of SQL Server (SQL Server Express [2019/2022], Developer Edition, or Standard/Enterprise). The database schema provided in this documentation needs to be created on this instance.
- **SQL Server Management Studio (SSMS) (Recommended):** For database management, schema creation, and direct data inspection.

1.2. Key NuGet Packages:

The project relies on several NuGet packages for its functionality. These packages should be restored automatically by Visual Studio when the solution is opened, or they can be manually installed via the NuGet Package Manager.

- **Microsoft.Data.SqlClient:**
 - **Project(s) Used In:** reLIFE.BusinessLogic
 - **Purpose:** The primary ADO.NET data provider for connecting to and interacting with Microsoft SQL Server. It is used by all Repository classes for executing SQL queries.
 - **Version (Approximate):** [5.1.x or latest stable]
- **MaterialSkin.2 (or similar name for the specific MaterialSkin library used):**
 - **Project(s) Used In:** reLIFE.WinFormsUI
 - **Purpose:** Provides Material Design theming and custom controls (e.g., MaterialForm, MaterialButton, MaterialTextBox2, MaterialCheckedListBox) for the Windows Forms user interface.

- **Version (Approximate):** [2.3.x or latest stable]
- **Microsoft.Extensions.Configuration:**
 - **Project(s) Used In:** reLIFE.BusinessLogic (specifically by DbHelper for reading appsettings.json manually) or reLIFE.WinFormsUI (if using a more integrated configuration host).
 - **Purpose:** Provides core abstractions for application configuration.
 - **Version (Approximate):** [8.0.x or version compatible with your .NET target]
- **Microsoft.Extensions.Configuration.Json:**
 - **Project(s) Used In:** reLIFE.BusinessLogic (by DbHelper) or reLIFE.WinFormsUI.
 - **Purpose:** Enables reading configuration settings from JSON files (i.e., appsettings.json).
 - **Version (Approximate):** [8.0.x or version compatible with your .NET target]
- **Microsoft.Extensions.Configuration.Binder (Optional but often included):**
 - **Project(s) Used In:** reLIFE.BusinessLogic or reLIFE.WinFormsUI.
 - **Purpose:** Allows binding configuration sections to strongly-typed C# objects.
 - **Version (Approximate):** [8.0.x or version compatible with your .NET target]

1.3. Project Setup from Source:

1. **Clone/Download Source Code:** Obtain the project source files.
2. **Open Solution:** Open reLIFE.sln in Microsoft Visual Studio.
3. **Restore NuGet Packages:** Visual Studio should automatically prompt to restore packages. If not, right-click the solution in Solution Explorer and select "Restore NuGet Packages."
4. **Database Setup:**
 - Ensure a SQL Server instance is accessible.

- Create a new database (e.g., reLifeDB).
- Execute the SQL schema script (provided in Appendix or Section 3.2) against the created database to generate the necessary tables.

5. Configure Connection String:

- Open the appsettings.json file in the reLIFE.WinFormsUI project.
- Update the DefaultConnection string under ConnectionStrings to point to your SQL Server instance and the reLifeDB database, using appropriate credentials (Windows Authentication or SQL Login).

6. Build Solution: Select "Build" > "Rebuild Solution" from the Visual Studio menu. Address any compilation errors.

7. Run Application: Set the reLIFE.WinFormsUI project as the startup project (if not already) and run the application (F5 or Start button).

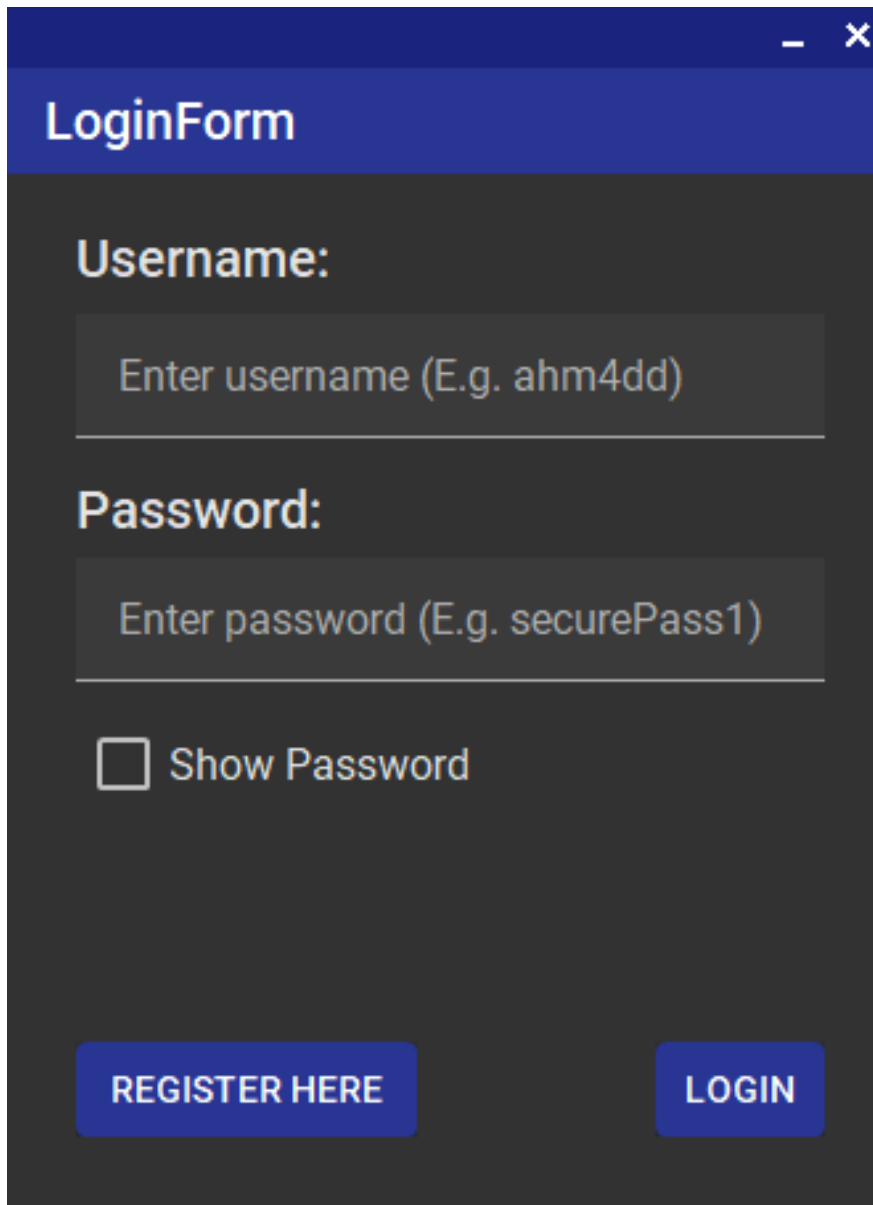
2.4. End-User Prerequisites (Running the compiled application):

For an end-user running a pre-compiled version of reLIFE Scheduler:

- **.NET Desktop Runtime:** The appropriate version of the .NET Desktop Runtime corresponding to the project's target framework (e.g., .NET 8.0 Desktop Runtime) must be installed on the user's machine.
- **(Implicit) Database Connectivity:** While not a user installation step, the application requires connectivity to the SQL Server database as configured during development/deployment. For a standalone single-user setup, this often means SQL Server Express running on the same machine.

2. Get Started

2.1. Launching the Application



The image shows a screenshot of a web application window titled "LoginForm". The window has a dark blue header bar with the title "LoginForm" in white. Below the header, the background is dark gray. The form contains two input fields: "Username:" and "Password:". The "Username:" field has a placeholder text "Enter username (E.g. ahm4dd)". The "Password:" field has a placeholder text "Enter password (E.g. securePass1)". Below the password field, there is a checkbox labeled "Show Password". At the bottom of the form, there are two buttons: "REGISTER HERE" and "LOGIN".

LoginForm

Username:

Enter username (E.g. ahm4dd)

Password:

Enter password (E.g. securePass1)

☐ Show Password

REGISTER HERE **LOGIN**

2.2. User Registration

— ×

RegistrationForm

Username:

Enter username (E.g. ahm4dd)

Email:

Enter email (E.g. AJ@gmail.com)

Password:

Enter password (E.g. securePass1)

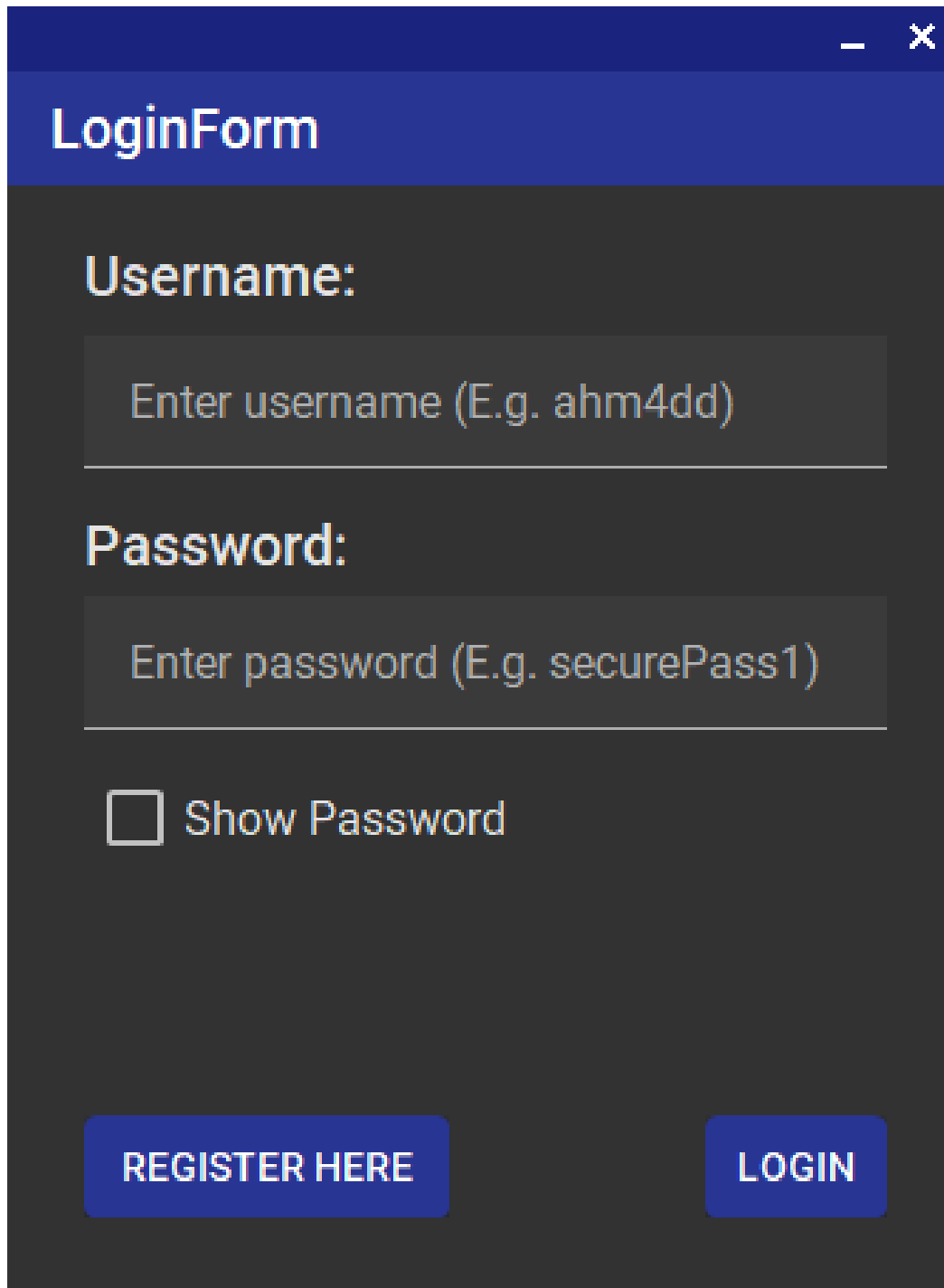
Confirm Password:

Re-enter your password

CANCEL

REGISTER

2.3. Logging In



A screenshot of a web application window titled "LoginForm". The window has a dark blue header bar with the title "LoginForm" in white. Below the header, the background is dark gray. The form contains two input fields: "Username:" and "Password:". The "Username:" field has a placeholder text "Enter username (E.g. ahm4dd)". The "Password:" field has a placeholder text "Enter password (E.g. securePass1)". Below the password field, there is a checkbox labeled "Show Password". At the bottom of the form, there are two buttons: "REGISTER HERE" and "LOGIN".

LoginForm

Username:

Enter username (E.g. ahm4dd)

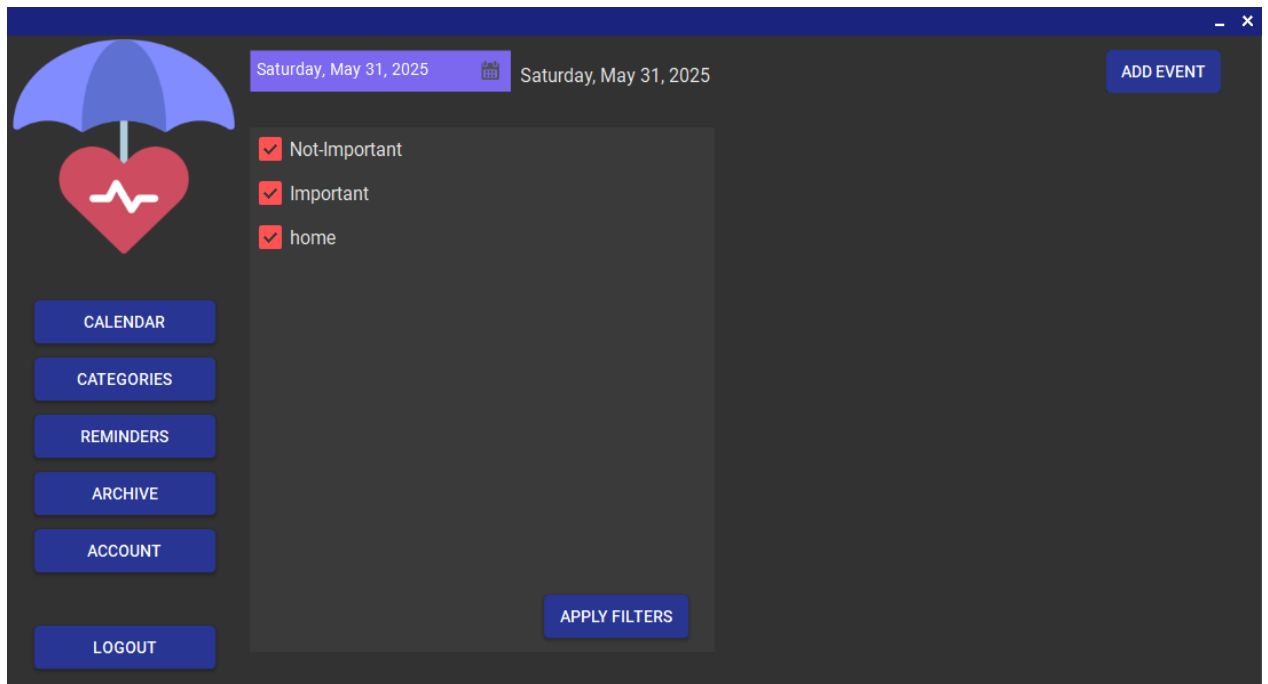
Password:

Enter password (E.g. securePass1)

☐ Show Password

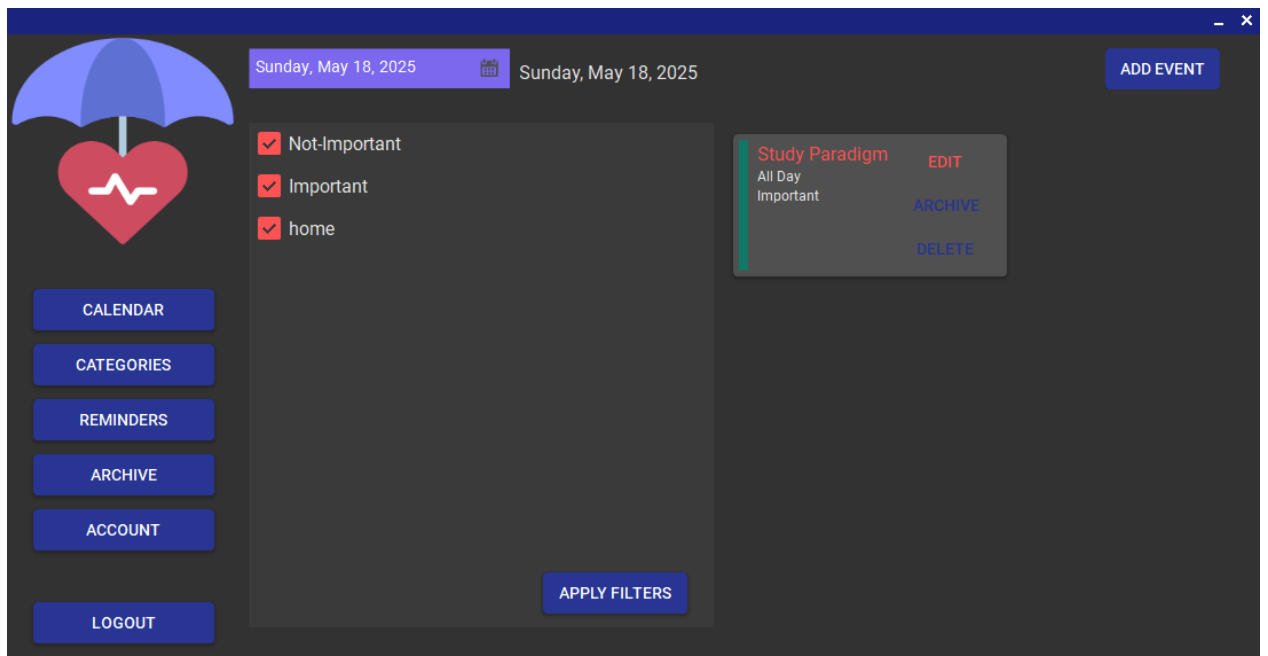
REGISTER HERE **LOGIN**

2.4. Main Dashboard Overview

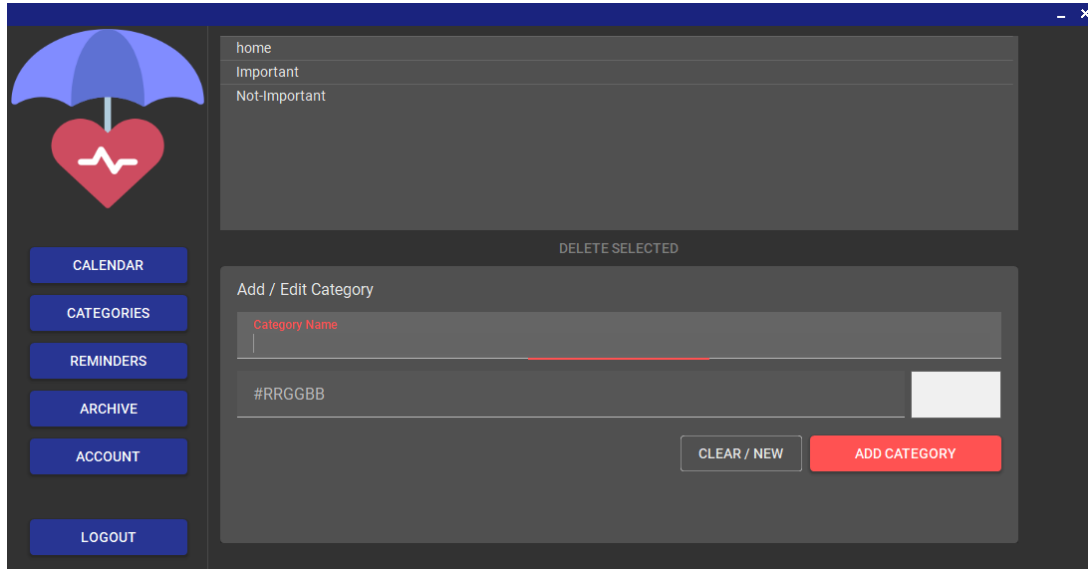


3. Using reLIFE Scheduler - Key Features:

3.1. Calendar View (CalendarViewForm)

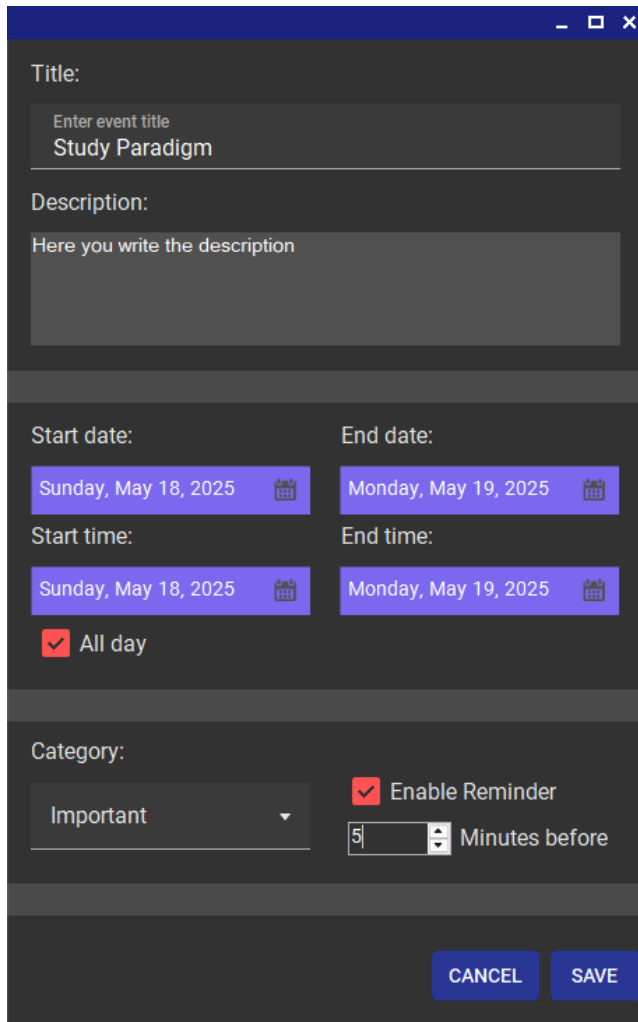


3.2. Managing Categories (CategoryManagerForm)



The screenshot shows the CategoryManagerForm interface. On the left is a sidebar with a logo (an umbrella over a heart with a pulse line) and a list of navigation buttons: CALENDAR, CATEGORIES, REMINDERS, ARCHIVE, ACCOUNT, and LOGOUT. The main area has a header with 'home', 'Important', and 'Not-Important' links. Below this is a 'DELETE SELECTED' button. The central form is titled 'Add / Edit Category' and contains two input fields: 'Category Name' (with a red error message) and a color picker labeled '#RRGGBB'. At the bottom right of the form are two buttons: 'CLEAR / NEW' and 'ADD CATEGORY'.

3.3. Event Form (EventForm - Add/Edit)



The screenshot shows the EventForm - Add/Edit interface. It features a 'Title:' field with the placeholder 'Enter event title' and the text 'Study Paradigm'. Below is a 'Description:' field with the placeholder 'Here you write the description'. The form is divided into two columns for date and time selection. The 'Start date:' is 'Sunday, May 18, 2025' and the 'End date:' is 'Monday, May 19, 2025'. The 'Start time:' is 'Sunday, May 18, 2025' and the 'End time:' is 'Monday, May 19, 2025'. There is a checkbox for 'All day' which is checked. Below this is a 'Category:' dropdown menu showing 'Important'. To the right of the dropdown is a checkbox for 'Enable Reminder' which is checked, followed by a spinner box containing the number '5' and the text 'Minutes before'. At the bottom right are two buttons: 'CANCEL' and 'SAVE'.

3.4. Managing Reminders (ReminderListViewForm)

The screenshot shows a web application window titled "ReminderListViewForm". On the left is a sidebar with a logo (an umbrella over a heart with a pulse line) and a vertical menu of buttons: CALENDAR, CATEGORIES, REMINDERS, ARCHIVE, ACCOUNT, and LOGOUT. The main content area displays a single reminder card for "Going to the Gym". The card shows the reminder is due on 5/31/2025 at 11:30 PM, with the event starting on 6/1/2025 at 12:00 AM (30 minutes before). There are "VIEW" and "DELETE" links on the right side of the card.

Going to the Gym [VIEW](#)

Reminder due: 5/31/2025 11:30 PM

Event starts: 6/1/2025 12:00 AM
(30 minutes before) [DELETE](#)

CALENDAR

CATEGORIES

REMINDERS

ARCHIVE

ACCOUNT

LOGOUT

3.5. Managing Archived Events (ArchiveViewForm)

The screenshot shows a web application window titled "ArchiveViewForm". The sidebar is identical to the previous form. The main content area features a date range filter at the top: "Archived From: Saturday, May 31, 2025" and "Archived To: Saturday, May 31, 2025", with a "FILTER BY DATE" button. Below this is a "Filter by Category:" section with three checked options: "Not-Important", "Important", and "home". An "APPLY CATEGORY FILTER" button is at the bottom of this section. The main area displays two archived event cards. The first card is for "aeidaesrnt", originally scheduled for 5/27/2025 (All Day), archived on 5/26/2025 at 7:12 AM, with categories "home" and "sreanitdnediat". The second card is for "Go to the gym", originally scheduled for 5/27/2025 (All Day), archived on 5/26/2025 at 7:11 AM, with categories "Important" and "Chest day". Both cards have "VIEW", "RETRIEVE", and "DELETE" links.

Archived From: Saturday, May 31, 2025 Archived To: Saturday, May 31, 2025 [FILTER BY DATE](#)

Filter by Category:

☒ Not-Important

☒ Important

☒ home

[APPLY CATEGORY FILTER](#)

aeidaesrnt [VIEW](#)

Originally: 5/27/2025 (All Day)

Archived: 5/26/2025 7:12 AM

home [RETRIEVE](#)

sreanitdnediat [DELETE](#)

Go to the gym [VIEW](#)

Originally: 5/27/2025 (All Day)

Archived: 5/26/2025 7:11 AM

Important [RETRIEVE](#)

Chest day [DELETE](#)

CALENDAR

CATEGORIES

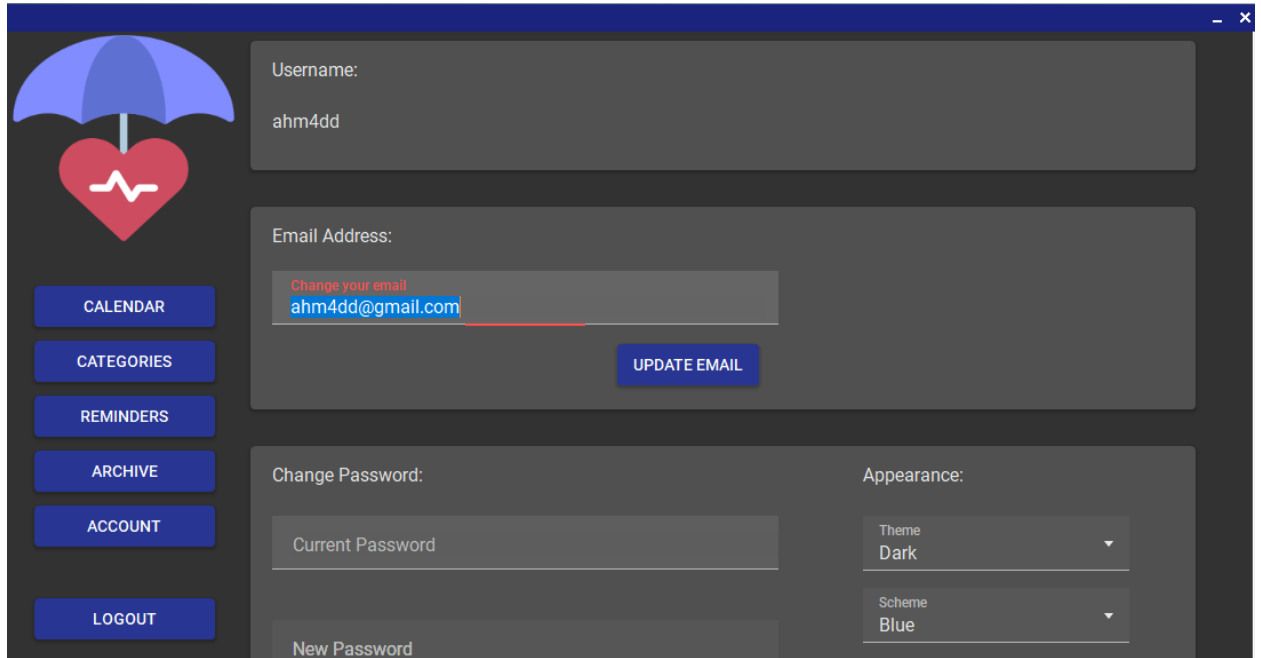
REMINDERS

ARCHIVE

ACCOUNT

LOGOUT

3.6. Account Settings (AccountSettingsForm)



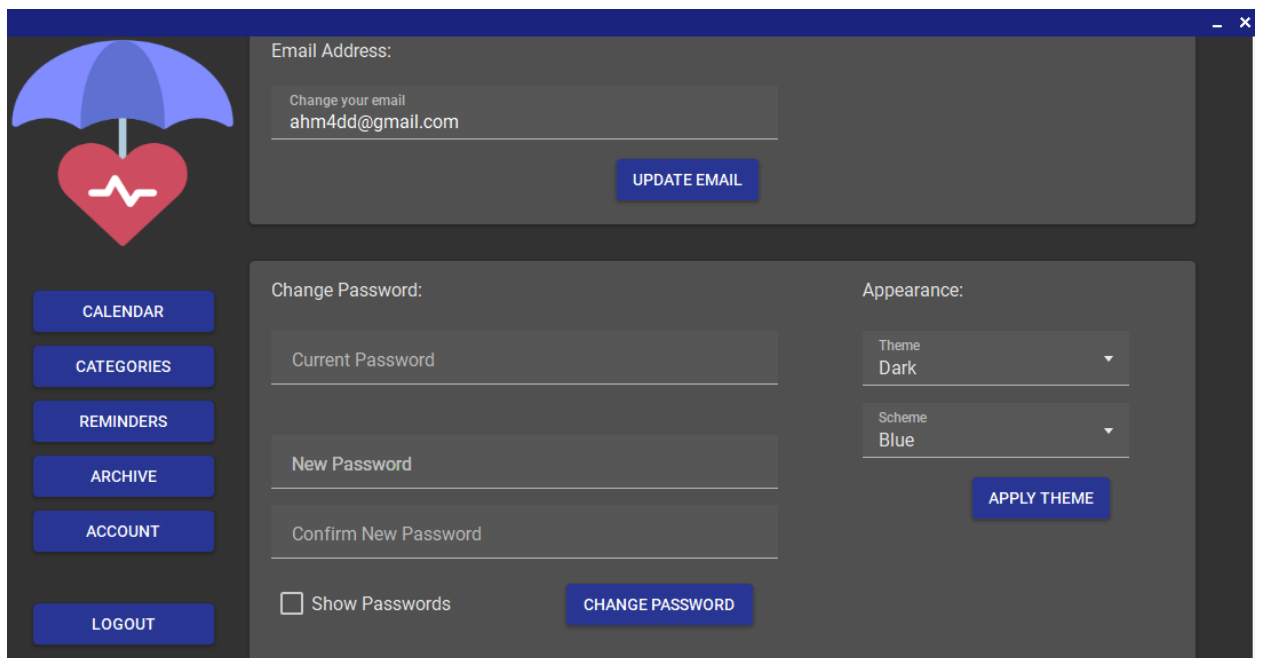
This screenshot shows the 'Account Settings' form with the 'ACCOUNT' tab selected. The form includes a sidebar with navigation links: CALENDAR, CATEGORIES, REMINDERS, ARCHIVE, ACCOUNT, and LOGOUT. The main content area has three sections: 'Username' (ahm4dd), 'Email Address' (ahm4dd@gmail.com) with a 'CHANGE your email' link and an 'UPDATE EMAIL' button, and 'Change Password' (Current Password, New Password) and 'Appearance' (Theme: Dark, Scheme: Blue).

Username: ahm4dd

Email Address: [Change your email](#)
ahm4dd@gmail.com **UPDATE EMAIL**

Change Password: Current Password New Password

Appearance: Theme Dark Scheme Blue



This screenshot shows the 'Account Settings' form with the 'ARCHIVE' tab selected. The sidebar remains the same. The main content area has two sections: 'Email Address' (ahm4dd@gmail.com) with a 'CHANGE your email' link and an 'UPDATE EMAIL' button, and 'Change Password' (Current Password, New Password, Confirm New Password) and 'Appearance' (Theme: Dark, Scheme: Blue). The 'Show Passwords' checkbox is unchecked, and there is an 'APPLY THEME' button.

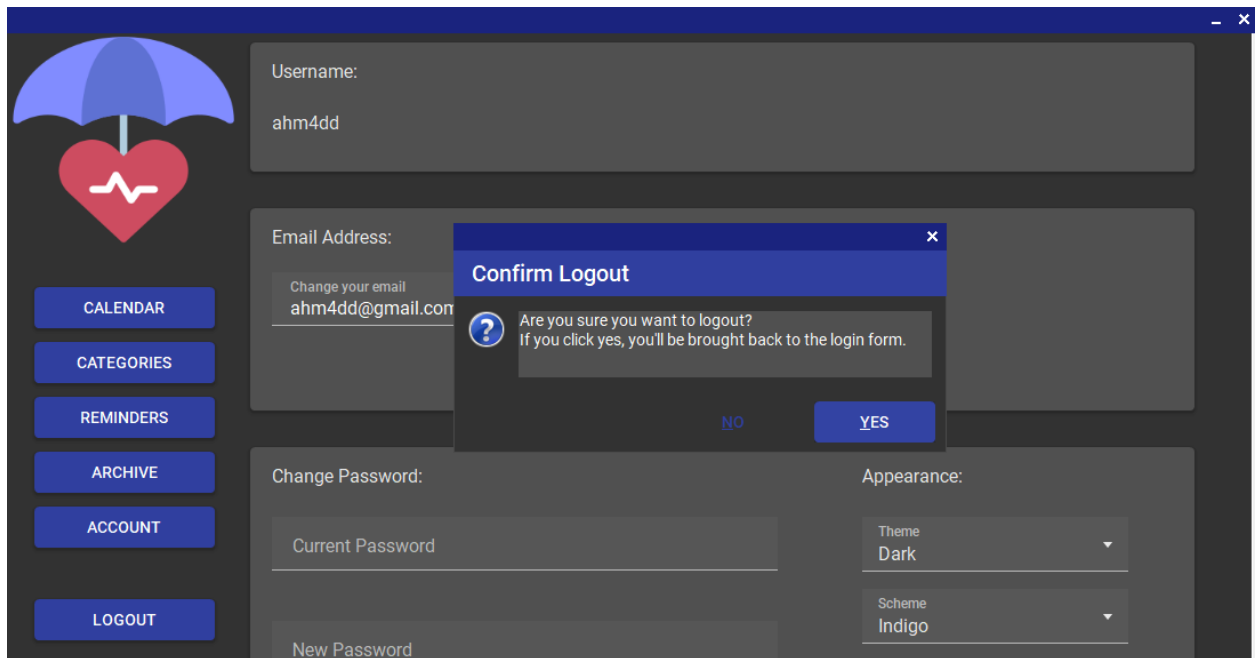
Email Address: [Change your email](#)
ahm4dd@gmail.com **UPDATE EMAIL**

Change Password: Current Password New Password Confirm New Password

Appearance: Theme Dark Scheme Blue **APPLY THEME**

☐ Show Passwords **CHANGE PASSWORD**

3.7. Logging Out



Part II: Technical Documentation

1. Project Overview & Architecture

Purpose: reLIFE Scheduler is a C# WinForms desktop application for personal event management, utilizing MaterialSkin for UI theming and ADO.NET for direct SQL Server database interaction.

Architecture (Layered):

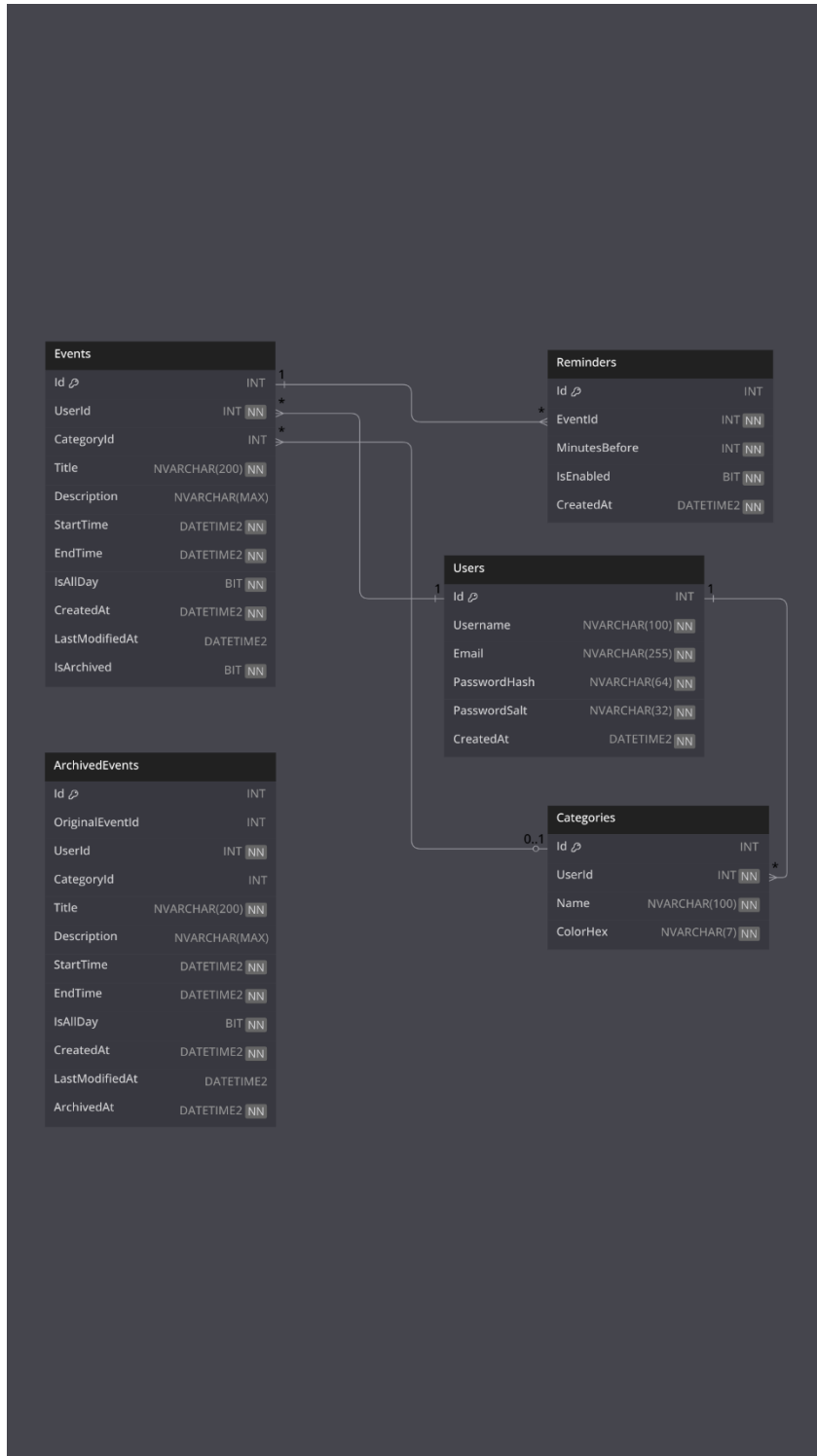
The application follows a layered architecture:

1. **Presentation (reLIFE.WinFormsUI):** Handles user interface (Forms like MainForm, CalendarViewForm, EventForm) and user input.
2. **Business Logic (reLIFE.BusinessLogic):** Contains services (EventService, AuthService, etc.) that implement application rules and coordinate data operations. Repositories (EventRepository, etc.) within this layer execute SQL queries.
3. **Core (reLIFE.Core):** Defines data models (User, Event, Category, etc.) as simple C# classes.

4. **Database (SQL Server):** Persistent storage of all application data.

2. Database Design

2.1. Enhanced Entity Diagram (EER)



The EER diagram visually represents the tables (Users, Categories, Events, ArchivedEvents, Reminders) and their relationships, including primary and foreign keys, and cardinalities.

2.2. Table Schemas (DDL Snippets)

- **Users Table:**

- CREATE TABLE Users (
 - Id INT PRIMARY KEY IDENTITY(1,1),
 - Username NVARCHAR(100) NOT NULL UNIQUE,
 - Email NVARCHAR(255) NOT NULL UNIQUE,
 - PasswordHash NVARCHAR(64) NOT NULL,
 - PasswordSalt NVARCHAR(32) NOT NULL,
 - CreatedAt DATETIME2 NOT NULL DEFAULT GETDATE()
-);

- **Categories Table:**

- CREATE TABLE Categories (
 - Id INT PRIMARY KEY IDENTITY(1,1),
 - UserId INT NOT NULL,
 - Name NVARCHAR(100) NOT NULL,
 - ColorHex NVARCHAR(7) NOT NULL,
 - CONSTRAINT FK_Categories_User FOREIGN KEY (UserId) REFERENCES Users(Id) ON DELETE CASCADE,
 - CONSTRAINT UQ_Categories_User_Name UNIQUE (UserId, Name)
-);
- CREATE INDEX IX_Categories_UserId ON Categories(UserId);

- **Events Table:**

- CREATE TABLE Events (
- Id INT PRIMARY KEY IDENTITY(1,1),
- UserId INT NOT NULL,
- CategoryId INT NULL,
- Title NVARCHAR(200) NOT NULL,
- Description NVARCHAR(MAX) NULL,
- StartTime DATETIME2 NOT NULL,
- EndTime DATETIME2 NOT NULL,
- IsAllDay BIT NOT NULL DEFAULT 0,
- IsArchived BIT NOT NULL DEFAULT 0,
- CreatedAt DATETIME2 NOT NULL DEFAULT GETDATE(),
- LastModifiedAt DATETIME2 NULL,
- CONSTRAINT FK_Events_User FOREIGN KEY (UserId) REFERENCES Users(Id) ON DELETE CASCADE,
- CONSTRAINT FK_Events_Category FOREIGN KEY (CategoryId) REFERENCES Categories(Id) ON DELETE SET NULL
-);
- CREATE INDEX IX_Events_UserId_StartTime ON Events(UserId, StartTime);

- **ArchivedEvents Table:**

- CREATE TABLE ArchivedEvents (
- Id INT PRIMARY KEY, -- Original Event.Id
- UserId INT NOT NULL,
- CategoryId INT NULL,
- Title NVARCHAR(200) NOT NULL,
- Description NVARCHAR(MAX) NULL,
- StartTime DATETIME2 NOT NULL,

- EndTime DATETIME2 NOT NULL,
- IsAllDay BIT NOT NULL DEFAULT 0,
- CreatedAt DATETIME2 NOT NULL, -- Original creation
- LastModifiedAt DATETIME2 NULL, -- Original modification
- ArchivedAt DATETIME2 NOT NULL DEFAULT GETDATE()
-);
- CREATE INDEX IX_ArchivedEvents_UserId ON ArchivedEvents(UserId);
- CREATE INDEX IX_ArchivedEvents_ArchivedAt ON ArchivedEvents(ArchivedAt);

- **Reminders Table:**

- CREATE TABLE Reminders (
- Id INT PRIMARY KEY IDENTITY(1,1),
- EventId INT NOT NULL,
- MinutesBefore INT NOT NULL,
- IsEnabled BIT NOT NULL DEFAULT 1,
- CreatedAt DATETIME2 NOT NULL DEFAULT GETDATE(),
- CONSTRAINT FK_Reminders_Event FOREIGN KEY (EventId) REFERENCES Events(Id) ON DELETE CASCADE
-);
- CREATE INDEX IX_Reminders_EventId ON Reminders(EventId);

2.3. Key Relationships & Integrity:

Referential integrity is maintained via foreign keys. ON DELETE CASCADE (e.g., Users to Events) and ON DELETE SET NULL (e.g., Events to Categories) are used. Indexes on primary keys, foreign keys, and frequently searched columns optimize query performance.

2.4. Database Interaction (ADO.NET):

The application uses ADO.NET (Microsoft.Data.SqlClient) for direct SQL Server

interaction. Repositories build and execute parameterized SQL queries. DbHelper provides the connection string.

```
// Example: UserRepository.GetUserByUsername (Simplified)

public User? GetUserByUsername(string username)
{
    User? user = null;

    const string sql = "SELECT Id, Username, Email, PasswordHash, PasswordSalt,
CreatedAt FROM Users WHERE Username = @Username;";

    using (var connection = new SqlConnection(_connectionString))
    using (var command = new SqlCommand(sql, connection))
    {
        command.Parameters.AddWithValue("@Username", username);
        connection.Open();
        using (var reader = command.ExecuteReader())
        {
            if (reader.Read())
            {
                user = new User { /* Map fields from reader */
                    Id = reader.GetInt32(reader.GetOrdinal("Id")),
                    Username = reader.GetString(reader.GetOrdinal("Username")),
                    Email = reader.GetString(reader.GetOrdinal("Email")),
                    PasswordHash = reader.GetString(reader.GetOrdinal("PasswordHash")),
                    PasswordSalt = reader.GetString(reader.GetOrdinal("PasswordSalt")),
                    CreatedAt = reader.GetDateTime(reader.GetOrdinal("CreatedAt"))
                };
            }
        }
    }
}
```

```

    }

    return user;
}

```

2.5. Security:

User passwords are not stored directly. They are hashed using SHA256 with a unique, per-user salt via the PasswordHasher class.

```

// Conceptual: PasswordHasher.cs

public class PasswordHasher
{
    public string HashPassword(string password, out string salt)
    {
        // 1. Generate random salt bytes

        // 2. Convert password to bytes

        // 3. Combine salt and password bytes

        // 4. Compute SHA256 hash of combined bytes

        // 5. Convert salt and hash to hex strings

        // Placeholder

        salt =
Convert.ToHexString(System.Security.Cryptography.RandomNumberGenerator.GetBytes(
16));

        // Simplified hashing for example:

        using var sha256 = System.Security.Cryptography.SHA256.Create();

        byte[] hashBytes =
sha256.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password + salt));

        return Convert.ToHexString(hashBytes);
    }
}

```

```

public bool VerifyPassword(string password, string storedHash, string storedSalt)
{
    // 1. Convert storedSalt (hex) back to bytes

    // 2. Combine entered password and retrieved salt bytes

    // 3. Hash the combination using SHA256

    // 4. Compare computed hex hash with storedHash (fixed-time comparison is ideal)

    // Placeholder

    using var sha256 = System.Security.Cryptography.SHA256.Create();

    byte[] hashBytes =
sha256.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password + storedSalt));

    return Convert.ToHexString(hashBytes).Equals(storedHash,
StringComparison.OrdinalIgnoreCase);
}
}

```

3. Key Workflows (Illustrative)

3.1. Adding a New Event:

User input from EventForm is passed to EventService.AddEvent. The service validates and calls EventRepository.AddEvent, which executes a parameterized INSERT INTO Events (...) OUTPUT INSERTED.Id, ... VALUES (...) SQL query. Reminders are handled separately via ReminderService.

3.2. Displaying Active Reminders (ReminderListViewForm):

ReminderListViewForm calls ReminderService.GetActiveUpcomingReminderInfos. This service method, in turn, calls ReminderRepository.GetActiveUpcomingReminderInfosForUser.

// In ReminderRepository.cs - GetActiveUpcomingReminderInfosForUser SQL core

```

public List<ReminderInfo> GetActiveUpcomingReminderInfosForUser(int userId,
DateTime? upcomingLimit = null)
{
    // ...

    string sql = @"
        SELECT r.Id AS ReminderId, r.EventId, e.Title AS EventTitle,
            e.StartTime AS EventStartTime, r.MinutesBefore, r.IsEnabled
        FROM Reminders r
        INNER JOIN Events e ON r.EventId = e.Id
        WHERE e.UserId = @UserId AND r.IsEnabled = 1 AND e.IsArchived = 0 AND
e.StartTime > GETDATE()"

        + (upcomingLimit.HasValue ? " AND DATEADD(minute, -r.MinutesBefore,
e.StartTime) <= @UpcomingLimit" : "")

        + " ORDER BY DATEADD(minute, -r.MinutesBefore, e.StartTime) ASC;";

    // ... ADO.NET execution and mapping to List<ReminderInfo> ...

    return new List<ReminderInfo>(); // Placeholder for actual return
}

```

The form then uses the ReminderInfo DTOs (which include event details from the join) to create MaterialCard displays.

4. Future Work

- Advanced recurring event features.
- Pop-up notifications for reminders.
- Search functionality.
- User settings persistence for theme and defaults.
-

Part III: References

- **Microsoft Corporation. (Date Varies). ADO.NET Guide.** Microsoft Docs.
 - Retrieved from: <https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/>
 - *Purpose: Core documentation for database interaction using ADO.NET, including SqlConnection, SqlCommand, SqlDataReader, and parameterized queries.*
- **Microsoft Corporation. (Date Varies). SQL Server Documentation.** Microsoft Docs.
 - Retrieved from: <https://docs.microsoft.com/en-us/sql/sql-server/>
 - *Purpose: Official documentation for SQL Server features, T-SQL syntax, database design principles, and management.*
- **Ignace Maes et al. MaterialSkin 2 for .NET.** GitHub Repository.
 - Retrieved from: <https://github.com/material-components/material-components-web-components> (Note: This seems to be the web components one. You'll need to find the **correct GitHub link for the WinForms MaterialSkin 2 library you used**. A common one is by "leocb/MaterialSkin" or "IgnaceMaes/MaterialSkin" - please verify the exact one.)
 - *Purpose: The UI library providing Material Design components and theming for the Windows Forms application.*
- **Microsoft Corporation. (Date Varies). System.Security.Cryptography Namespace.** Microsoft Docs.
 - Retrieved from: <https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography>
 - *Purpose: Documentation for cryptographic services, including SHA256 and RandomNumberGenerator used for password hashing and salt generation.*
- **Microsoft Corporation. (Date Varies). JSON configuration provider in .NET.** Microsoft Docs.
 - Retrieved from: <https://docs.microsoft.com/en-us/dotnet/core/extensions/configuration-providers#json-configuration-provider>
 - *Purpose: Information on using appsettings.json and the Microsoft.Extensions.Configuration libraries for application configuration.*