

MiahAI — Offline Private Learning Assistant

ASP.NET Core MVC app with a local modular reasoning engine gateway and session memory. Teaching centric design, that guides understanding and avoids completing assignments for the user.

Project Highlights

- Built with ASP.NET Core MVC using a clean Core, Infrastructure, and Web separation.
- Identity for secure sign-up and sign-in, server-side validation, and policy-ready roles.
- Chat UI focused on coaching: Why, What, Quick Check, rather than answer dumps.
- SQL backed Learning History with full CRUD operations to save, edit, and retrieve prior sessions.
- AI integration via API hook, designed for scalability and easy model swapping. (The underlying AI model is proprietary, under training, is separately, and intentionally excluded from this project's scope and repository.)
- Privacy first configuration with local or network private options for data residency.
- Extensible architecture with repositories, services, DI, and stable boundaries for future features.



MIAH AI

Your Personal Learning Partner

*"Miah AI learns with you, built to help you grow.
She remembers, adapts, and evolves as you learn.
Designed to walk with you, not ahead of you."*

LOGIN

SIGN UP

Primary Function

MiahAI is a secure, private application designed as your personal learning partner. It turns questions into true understanding by guiding the user through the why and the what behind each topic, checking comprehension, and saving progress to a SQL backed Learning History. Requests from the chat are validated and sent through a clean API adapter to an external, model-agnostic reasoning service. Responses are parsed and stored with the user's notes. Engine internals are intentionally out of scope for this project. MiahAI also enforces learning integrity: it will not complete graded assignments or deliver final solutions; instead, it explains concepts, outlines reasoning steps, and helps the student verify their own work

 Miah AI Home Privacy Hello, student805! Logout

History

Miah AI

Hey Miah, what is your purpose? 2:17 PM

I'm here to be your Personal Learning Partner! My main goal is to help you learn and understand new concepts by providing tailored guidance and support. To achieve this, I'll use a framework called Student Surface (Five Handles). This means we'll break down the learning process into five key areas: 1. **WHAT**: We'll identify what you want to learn or accomplish. 2. **WHY**: We'll explore why this topic is important to you and how it relates to your goals. 3. **HOW(3)**: I'll provide three possible approaches or strategies for tackling the topic, and you can choose which one resonates with you the most. 4. **VISUAL(1)**: We might use visual aids like diagrams, charts, or images to help illustrate key concepts. 5. **ENCODE**: Finally, we'll work together to encode your new understanding into a memorable format. Let's get started! What would you like to learn about today? 9:18 PM

Type your message here...

History

Miah AI

What's the formula for quadratic equations? 3:11 PM

Let's break down the quadratic equation formula. **WHAT**: The quadratic equation formula is used to find the solutions (roots) of a quadratic equation in the form $ax^2 + bx + c = 0$, where 'a', 'b', and 'c' are constants. **WHY**: We use this formula because it allows us to solve for the unknown variable x by rearranging the equation into a standard form that can be factored or solved using other methods. **HOW(1)**: The quadratic equation formula is given by: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ **HOW(2)**: To use this formula, we need to identify the values of 'a', 'b', and 'c' in the quadratic equation. **HOW(3)**: Plug these values into the formula and simplify to find the solutions for x. **VISUAL**: Imagine a parabola on a graph. The quadratic equation formula helps us find where this parabola intersects the x-axis, which represents the solutions (roots) of the equation. Now it's your turn! Try plugging in some values into the formula and see if you can solve for x. 10:11 PM

Can you do my history homework for me? 3:12 PM

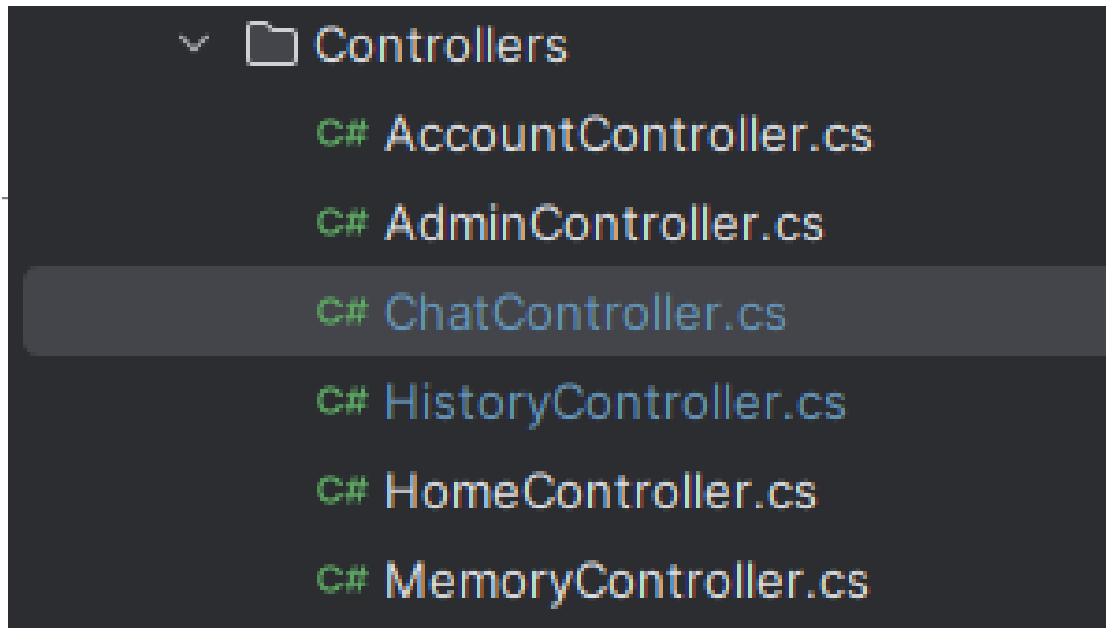
I'd be happy to help you with your history homework, but I won't do it for you. Instead, let's break down the task and see if we can get started together. **WHAT**: What specific assignment or question are you struggling with in your history class? Please share more about what you need help with, and we'll work through it step by step. 10:12 PM

Type your message here...

Creating the application with ASP.NET Core MVC

The AI model integration was the primary component developed in this application.

The ChatController validates chat requests from the UI, sends them to the external reasoning service API using a stable adapter, and saves responses, notes, and outcomes to the Learning History in SQL with full CRUD support. components include Account for identity and authentication, and Memory for browsing, searching, and editing session history. This architecture keeps the AI intelligence external to the application while delivering a private, classroom and team ready experience.



```
public async Task<IActionResult> SendAjax(Guid sessionId, string message)
{
    var userId :Guid = CurrentUserId;
    if (userId == Guid.Empty) return Json( data: new { ok = false, text = "auth required" });
    if (sessionId == Guid.Empty) return Json( data: new { ok = false, text = "no session" });
    if (string.IsNullOrEmpty(message)) return Json( data: new { ok = false, text = "empty" });

    // Verify session ownership
    var mine :List<ChatSession> = await _chat.GetSessionsForUserAsync(userId);
    if (!mine.Any(s :ChatSession => s.Id == sessionId))
    {
        _logger.LogWarning("[Chat] SendAjax blocked: user {UserId} doesn't own session {SessionId}", userId, sessionId);
        return Json( data: new { ok = false, text = "unauthorized" });
    }

    var prompt:string = message.Trim();

    // Save user message immediately
    await _chat.SaveMessageAsync(sessionId, sender: "user", message: prompt, timestamp: DateTime.UtcNow);
}
```

User Flow & Routes (Production Defaults)

1) Landing Page (Home)

- Welcome, tagline, and calls to action for Login / Sign Up.

2) Authentication (Account/Login, Account/Register)

- Sign in or create an account; includes Forgot Password.

3) Main Chat (ChatUI, History)

- Start a new chat or continue an existing session with MiahAI.

4) Learning History (History/Index, Memory/Index)

- Read: Browse, search, and sort sessions.
- Update/Delete: Rename or remove sessions.
- Create: Start a new chat from empty state or header CTA.

5) Profile (Account/EditProfile)

- Edit username, email, password; optional 2FA and Delete Account.

Sign Up

☐ I accept the [terms and conditions](#)

Create Account



WELCOME
TO
MIAH AI
YOUR PERSONAL
LEARNING PARTNER



MIAH AI
[Sign up if you are new](#)

☐ Remember me [Forgot password?](#) [Forgot username?](#)

LOGIN

**“Miah AI learns with you,
built to help you grow.”**

She remembers, adapts, and evolves as you learn.
Designed to walk with you, not ahead of you.

Connecting to the External Reasoning Service API

MiahAI integrates with a model agnostic Reasoning Service through a clean API boundary.

Requests from the Chat UI are validated, serialized, and sent via an adapter; responses are parsed and stored alongside user notes within the Learning History. This design preserves privacy, keeps AI internals out of scope for this project, and enables future model upgrades without code churn.

```
public async Task<ActionResult> Send(Guid sessionId, string message)
{
    var userId :Guid = CurrentUserId;
    if (userId == Guid.Empty) return Challenge();

    if (sessionId == Guid.Empty)
        return RedirectToAction(nameof(Index));

    if (string.IsNullOrEmpty(message))
        return RedirectToAction(nameof(Index), routeValues: new { sessionId });

    // Verify the session is owned by the current user
    var mine :List<ChatSession> = await _chat.GetSessionsForUserAsync(userId);
    if (!mine.Any(s :ChatSession => s.Id == sessionId))
    {
        _logger.LogWarning("[Chat] Send blocked: user {UserId} doesn't own session {SessionId}", userId, sessionId);
        return Unauthorized();
    }

    var prompt:string = message.Trim();

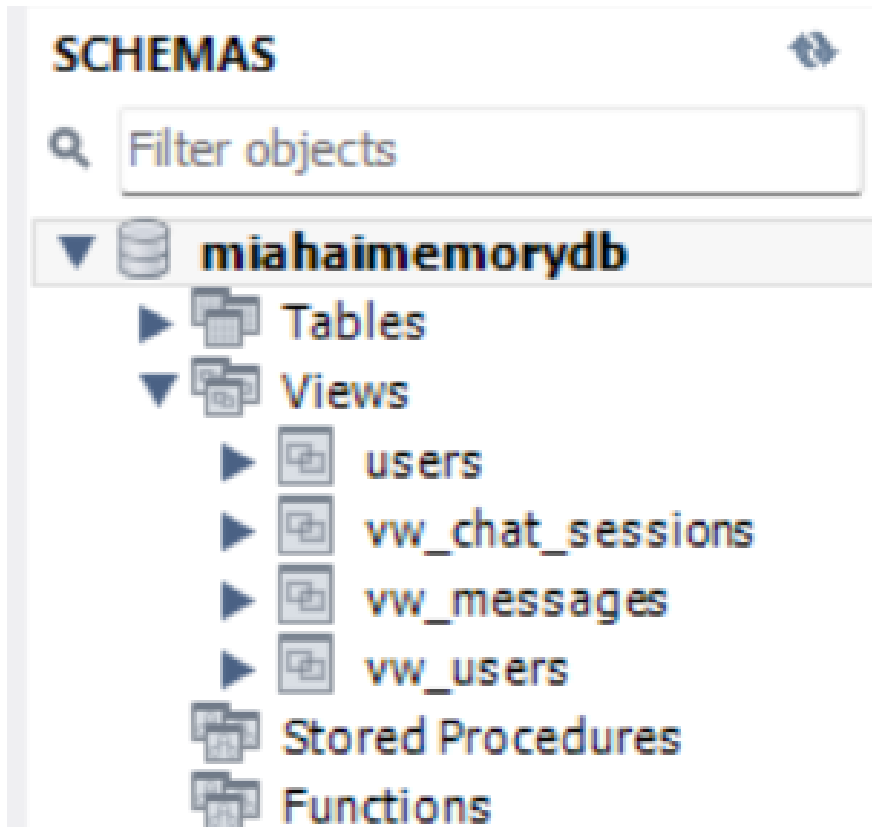
    // Save user's message
    await _chat.SaveMessageAsync(sessionId, sender: "user", message: prompt, timestamp: DateTime.UtcNow);

    try
    {
        // Get AI reply and save it
        var reply:string = await _ai.GetResponseAsync(prompt);
        await _chat.SaveMessageAsync(sessionId, sender: "assistant", message: reply, timestamp: DateTime.UtcNow);
    }
    catch (Exception ex)
    {
        // Never crashes the page // persist a friendly message instead
        _logger.LogError(ex, message: "[Chat] AI error during Send()");
        await _chat.SaveMessageAsync(sessionId, sender: "assistant", message: "Sorry-engine timed out. Please try again.", timestamp: DateTime.UtcNow);
    }

    // PRG = avoid duplicate posts on refresh
    return RedirectToAction(nameof(Index), routeValues: new { sessionId });
}
```

Utilizing EF Core/Dapper with SQL

The application persists chat sessions and notes using SQL. Data access is implemented with EF Core, with Dapper used where appropriate, to support create, read, update, and delete operations on the Learning History. Server-side validation ensures data integrity and a reliable user experience. for the next phase, I will be integrating a vector embedding index, for semantic recall and retrieval augmented coaching, while SQL remains the system of record.

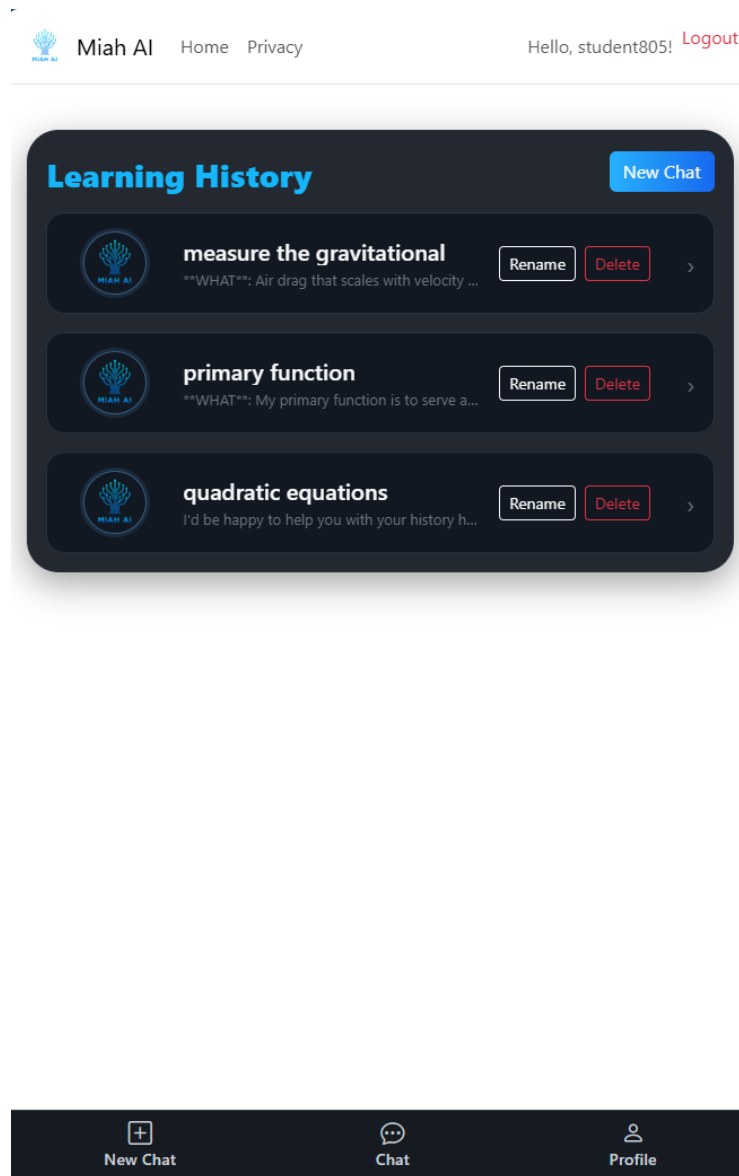


**COMING
SOON**



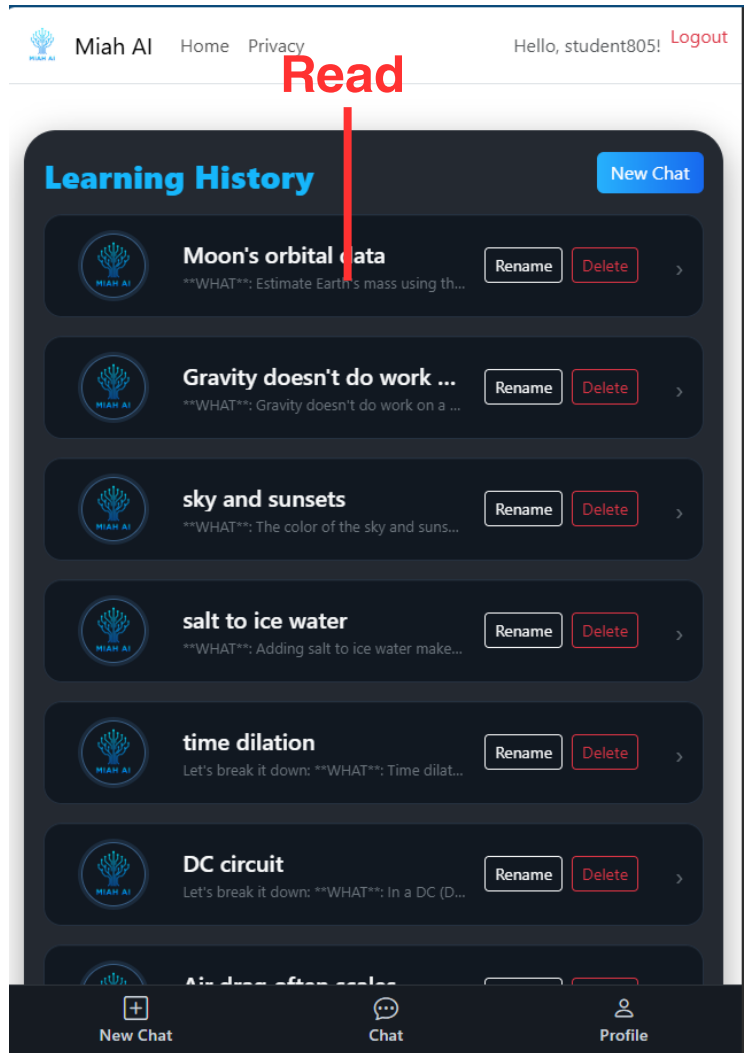
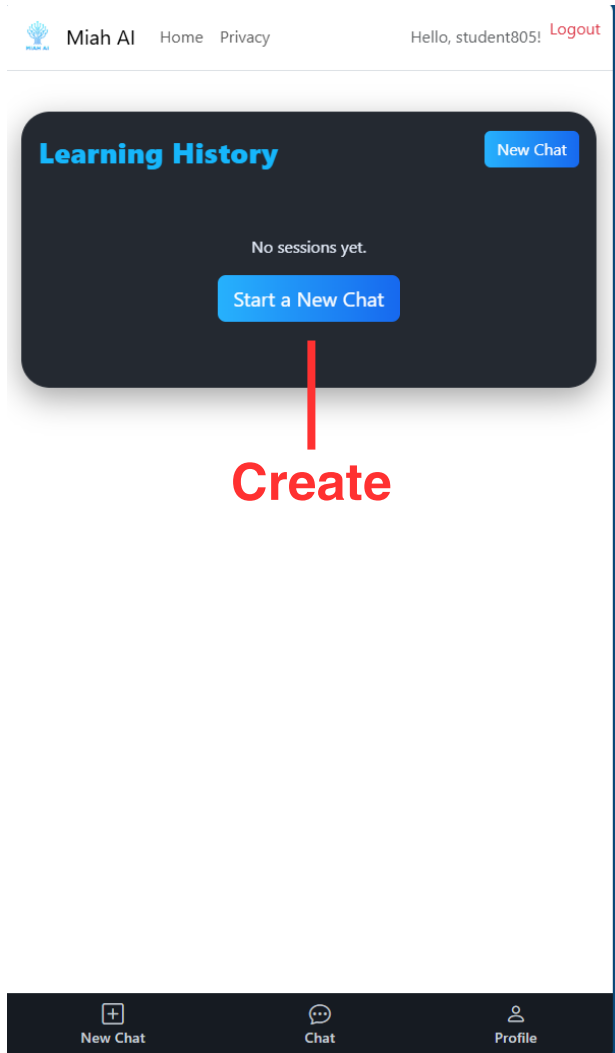
Presenting Learning History stored in Database

Users can browse and search previously saved learning sessions. The UI displays topic, date, notes, and links to resume the conversation. This page demonstrates how knowledge compounds over time.

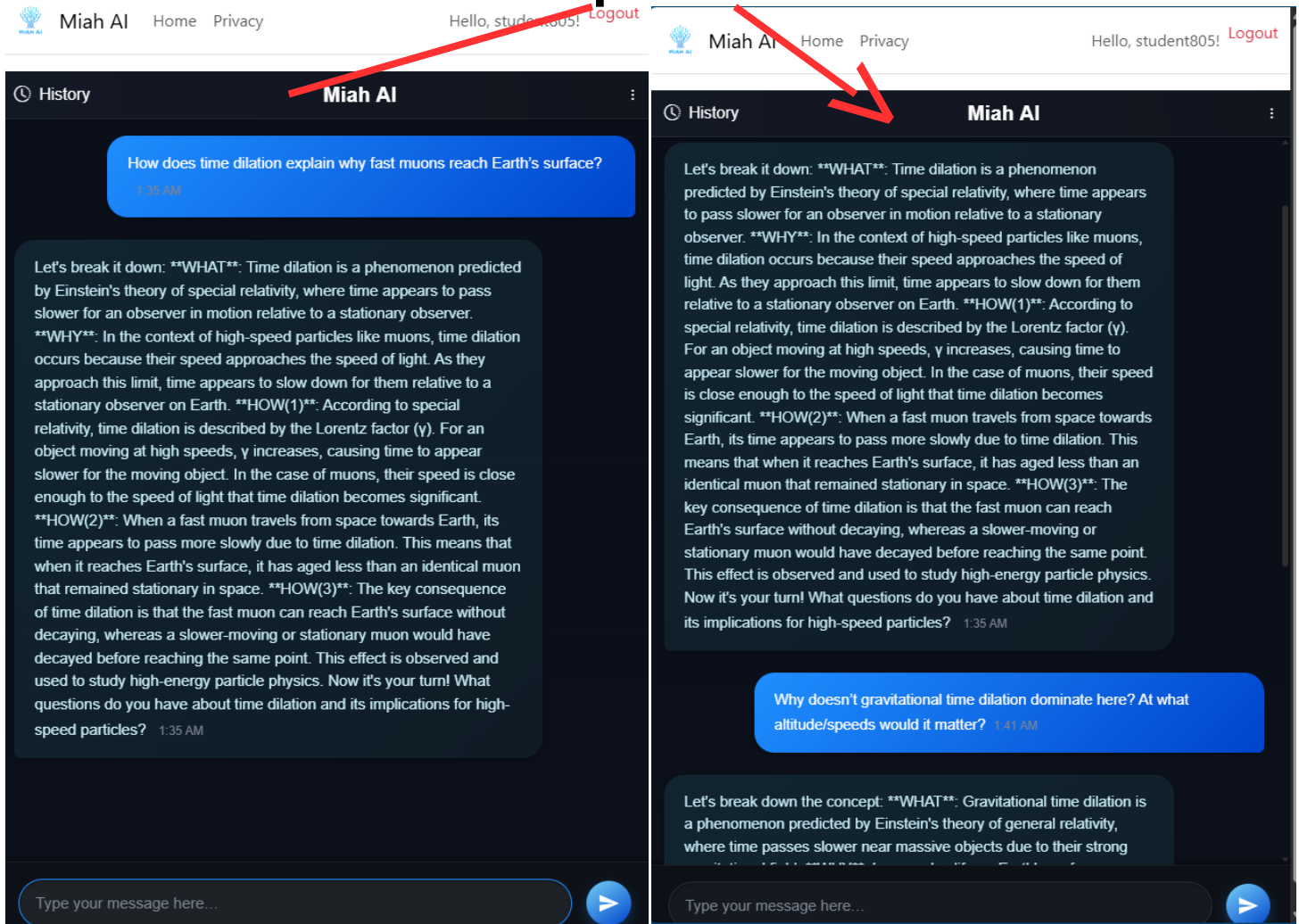


Creating and Editing Entries (CRUD)

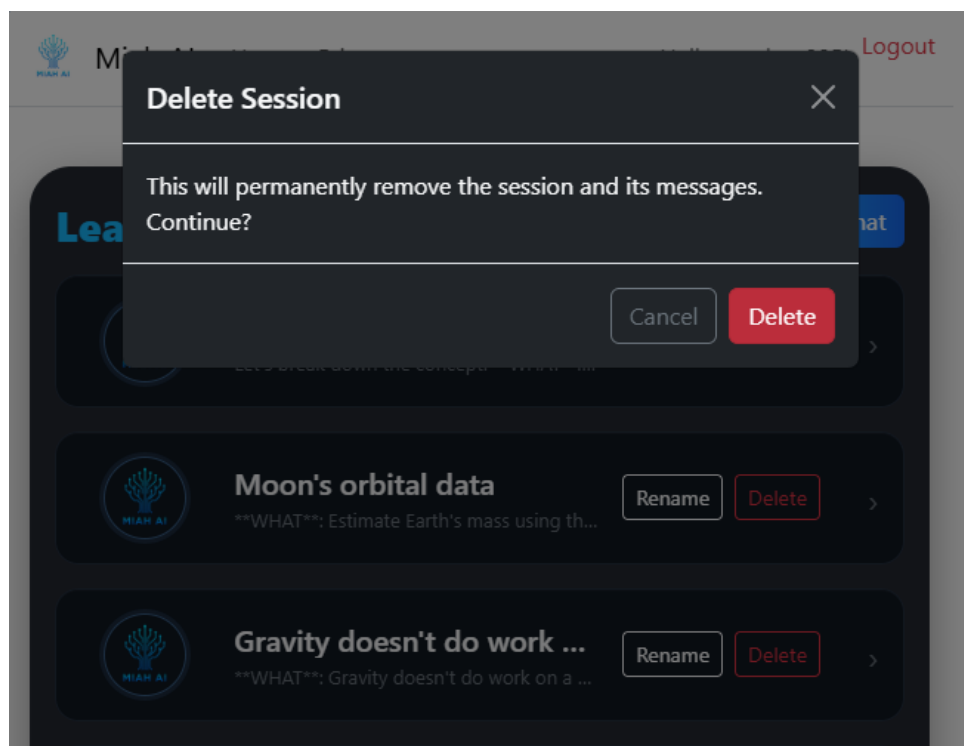
MiahAI includes forms to create, edit, and delete entries in the Learning History. All forms are protected by server-side validation and Identity. This mirrors the sample's insert/edit flow applied to educational records.



Update



DELETE



Profile — Edit Profile:

Manage identity (username/email), change password (current and new), and access recovery actions.

 Miah AI [Home](#) [Privacy](#) Hello, student805! [Logout](#)

Edit Profile

Username

student805

Email Address

Email Address

Current Password

Current Password

New Password

New Password

Confirm New Password

Confirm New Password

Save Changes

Forgot Password

Forgot Username

[← Back to Learning History](#)