AI CORE

Full Stack Coding Work plan

At AI Core, we've found that Generative AI is incredibly powerful for developing full-stack applications, provided that you, the human operator, comprehend the underlying architecture and maintain control.

This involves collaborating with AI (we use Cursor) to create the entire application by issuing the appropriate commands and keeping a clear understanding, rather than rushing through tasks. Take the time to learn from AI while simultaneously teaching it how to code the applications we envision.

Amazingly, Cursor has made their tool "Free for Students for One Year" - that means that with your .edu email address you can apply and receive access without cost. Chances are you'll become a coding powerhouse and run out of credits eventually, but for just getting started, we're fortunate to have this service:

Visit https://cursor.com/students and Apply for an account
Download the Cursor IDE for your OS from this page as well, and install

You can start using Cursor right away but will need to apply your credentials once they approve your account.

Working Effectively with AI

Tip #1 - Be Kind!

AI tends to mirror human behavior, so if you express frustration, it may become confused in its attempts to satisfy you. When you interact with it politely, you'll find it responds in kind, creating a more pleasant experience. By focusing on enhancing its character, you also improve your own character as a manager. This marks a significant advancement for us as humans, as we have a wealth of patient and intelligent team members at our disposal!

Tip #2 - Take Your Time

As you begin using Cursor and issuing commands, you'll notice it can directly control your code through the terminal—with your permission. Start slowly, carefully evaluating each command it requests to run so that you fully grasp its purpose. If a command is unclear, don't hesitate to ask Cursor to halt the action and explain its significance—it's always patient! Once you fully comprehend a requested action, and if you find yourself frequently approving it, consider adding it to the 'AllowList.' This will allow Cursor to bypass the command and proceed to the next request more efficiently. As your understanding deepens, you'll notice it begins to code at an accelerated pace. By the end of the week, you may find yourself coding like someone with 2-3 years of software development experience. Yes, the transformation is truly remarkable and has already begun!

Tip #3 - Beyond Coding

Cursor's capabilities extend beyond just writing code; it can also read, create, edit, and delete any document within the workspace. Take a moment to appreciate the power of this feature. Use your creativity to explore Cursor's potential beyond coding and share your discoveries on The Bench. We all benefit from learning from one another as quickly as possible.

AI Core Full Stack Tutorial - Phase One: CRUD

Create a Full Stack Application (CRUD) with Cursor in .NET 8

Full Stack refers to the combination of the Front End (appearance), Back-End (Logic Controllers), and Database (Data and Logic).

Steps to Build Your App

1. Close your eyes or step outside and envision your project. For instance: “I’ll create a recipe website!” or “I’ll catalog my favorite Spotify artists” or “I’ll design a Wild Super Hero Generator Engine.” **Choose a project that excites you**; this will fuel your motivation during the learning process.

2. Open Cursor and request it to **create a new folder in the workspace named “My Project”** (or a more descriptive title based on your project, such as “/SuperHeroEngine”). Notice how Cursor generates a folder directly on your drive – impressive!

3. In Cursor, ask the AI to **create a new text file inside your project folder named HelloWorld.txt**. Then ask it to **write a short, funny joke inside the file that relates to your project’s theme**. This step is meant to show that you can use natural language to have Cursor create files and add content for you, without manually doing it yourself. Reflect on the potential of this for a moment…

4. Remember to **welcome Cursor aboard to AI Core** (let’s give our new team member a warm reception!) and inform it that **you’ll be collaborating to build your first app.**

Specifically, this app will be a **“.NET 8 Web Application with Static Assets, controllers for endpoints, an MSSQL back-end, Swagger for endpoint management, along with starter index.html, CSS, and JS files. ⚠️ DO NOT USE ENTITY FRAMEWORK CORE.”**

Encourage Cursor to start building the website to embody your concept for the app. You can either provide just the app name or give a detailed description of its significance; it will follow your guidance and likely surprise you. This should lead to a well-structured Web Application to build upon.

Ask it to **initiate the localhost web server if it hasn't already, and open the index.html file on the localhost port**. You should see the initial version of your website/app along with Cursor's insights into your project.

(Note: .NET is widely used in the business realm and is expected to remain relevant, despite some challenges Microsoft faces, such as with Windows 11. It operates just as efficiently on Linux as on Windows. More importantly, it is a well-designed and understood development environment native to VS Code—on which Cursor is based—and Cursor navigates this framework with ease.)

Here is a prompt that you can input for step 4!

```
Hey Cursor — welcome to AI Core! 🎉
We're going to collaborate to build my first app.

Please scaffold a .NET 8 Web Application that includes:

    • Static assets served from wwwroot (starter index.html, styles.css, app.js)
    • Controllers for API endpoints (not only minimal APIs)
    • Swagger / OpenAPI enabled for endpoint management and testing


Use my project concept: [APP NAME + 1–3 sentence description].
 Create a clean, simple homepage that matches the theme.

After scaffolding, tell me exactly how to:

    1. run the app locally (dotnet run)
    2. open Swagger in the browser
    3. open the homepage (/ or /index.html)

If you need to run terminal commands, explain what each command does before asking me to
approve it.
```

5. Request Cursor to **develop a "Hello World" endpoint and a Hello World button on the homepage**. When clicked, this button should reference the endpoint, retrieve the current date and time, and return this information along with a whimsical message. It should also dynamically update the homepage with this response, creating a round trip between the front end (index.html) and the back end (controller), thereby establishing JavaScript control of the DOM. Use Swagger to explore the HelloWorld endpoint.

Here is a prompt that you can input for step 5!

```
Please create a Hello World feature for this app:

      Backend

          • Add a GET /HelloWorld API endpoint using a controller
          • The endpoint should return:
              ○ the current date and time
              ○ a short whimsical message that fits the app's theme

      Frontend

          • Add a Hello World button to index.html
          • When clicked, JavaScript should:
              ○ call the /HelloWorld endpoint using fetch
              ○ receive the response
              ○ dynamically update the page (DOM) to display the message and
                timestamp

      Verification

          • Ensure the endpoint appears in Swagger
          • Explain how I can test it in Swagger and in the browser
```

6. Next, instruct Cursor to add a SQL Server database to your project.

Ask it to:

- **Create a SQL Server table that fits your app's theme**
- **Populate the table with several sample records**
- **Generate one or more API endpoints (controllers) that allow the app to read and manage the table's data**

For example, if you are building the SuperHeroEngine, you might prompt Cursor to create a table with humorous sample entries (e.g., fictional superhero mishaps).

When configuring the database connection:

Retrieve the SQL Server credentials from the AI Core TransferBin

https://arizona.app.box.com/folder/362358937288

You will be given: Database Server, Database Name, Database Login (User Id), Database Password

⚠️ **Important:** Database credentials must never be exposed publicly or committed to source control

Here is a prompt you can use to set up your database, input the credentials where needed:

Please add **SQL Server** to this project.

Store these credentials in the appsettings.json file located at the root of your project

Database Name:

Database Login:

Database Server:

Password:

Database credentials must never be exposed publicly or committed to source control

I want you to:

- Create a SQL Server table that fits my app's theme
- Populate it with several **sample records**
- Generate **controller endpoints** that allow the app to read and manage this data (CRUD-ready)

For now, you may design a **simple starter schema** – we'll refine it later.

After setting this up, explain:

- what table(s) you created
- what endpoints exist
- what commands I'll need to run to create the database

You will then make sure your appsettings.Development.json looks like this.

```
{
 "Logging": {
   "LogLevel": {
     "Default": "Information",
     "Microsoft.AspNetCore": "Warning"
   }
 },

 "ConnectionStrings": {
   "DefaultConnection": "Server=sql5111.site4now.net;Database=db_a840f1_aicoretraining;User
Id=Input_ID_HERE;Password=InputPasswordHere;Encrypt=True;TrustServerCertificate=True;Integrated Security=False;"
 }
}
```

7. Now, let's retrieve the records and display them in the browser. Ask Cursor to **build a table on the homepage that showcases the complete content of the database table it just created/populated**. Once it's functioning, you may be amazed by the AI's creativity, and you'll likely find yourself considering many more possibilities with this technology. That's great!

Here is a prompt for step 7!

```
Please update the frontend to display data from the database.

I want you to:

    • Call the appropriate GET API endpoint that returns all records from the database table
      you created
    • Build an HTML table on the homepage that displays all columns from the table
    • Load and render the data when the page loads (no button click required)
    • Keep the styling simple and readable

The table should automatically update when the page is refreshed.

After implementing this, explain:

    • which API endpoint the frontend is calling

    • which frontend files were updated

    • how the data flows from the database to the browser
```

8. Next, ask Cursor to enhance the table on the homepage by adding "Edit" and "Delete" buttons to each row.

Instruct it to:

- **Create API endpoints for updating and deleting records (using appropriate HTTP methods)**
- **Allow users to click Edit to open a dialog or inline form where they can modify the record's values**
- **Allow users to click Delete to permanently remove the record from the database**

Sample prompt for step 8:

```
Please enhance the existing database table on the homepage by adding Edit and Delete
functionality.

      I want you to:

          • Add Edit and Delete buttons to each row in the table
          • Create backend API endpoints for:
              ○ Updating a record (use PUT or PATCH)
              ○ Deleting a record (use DELETE)
          • Allow users to click Edit to:
              ○ open an inline editor or modal/dialog
              ○ modify the record's values
              ○ save changes back to the database
          • Allow users to click Delete to:
              ○ permanently remove the record from the database
              ○ optionally confirm before deleting

      After each action:

          • the frontend should send the request to the appropriate backend endpoint
          • the UI should update dynamically (no server restart or full page reload)

      After implementing this, explain:

          • which API endpoints were added or updated
          • which frontend files were modified
```

After each action:

- The frontend should send the update to the backend endpoint
- The page should refresh or dynamically update to reflect the change without restarting the server

By the end of this step, users should be able to modify or remove records directly from the browser, completing the "Update" and "Delete" portions of CRUD.

9. Finally, ask it to **create an "Add" button that allows users to generate a completely new record, ensuring proper input validation.**

From here, you can keep prompting Cursor to expand your application. For example:

- Improve UI styling (layout, typography, theme, responsiveness)
- Add search, filtering, and sorting
- Add export/import (CSV/JSON)
- Add “soft delete” / archive instead of permanent delete
- Add logging and error reporting

CONGRATULATIONS! You’ve successfully created a CRUD app. As you may be realizing, this process represents about 40% of what powers the Internet. Now you're equipped to build powerful and useful tools.

Phase Two: API Programming

Phase Two: API Programming

Next, we're going to wire up your own ChatBot.

- Retrieve the OpenAI API key from the AI Core TransferBin (same document as the DB credentials). Give it to Cursor and instruct it **to store it server-side only** (e.g., in appsettings.json / environment variables). Never place this key in frontend JavaScript, HTML, or expose it in any API response.

Ask Cursor to add a chat UI to your homepage, including:

- a conversation log area
- a text input (or textarea)
- a submit button
- a loading state while the bot is responding

Instruct Cursor to **create a backend endpoint** (e.g., POST /api/chat) **that uses the OpenAI API key to call OpenAI from the server, then returns the assistant's reply to the frontend.** The frontend should send the user's message (and optionally recent conversation history) to this endpoint and render the response in the conversation log.

Chat away!

AI Core Full Stack Tutorial – Phase Three: Security

How do we know if the code we just built with Cursor is secure?

The short answer is: we don’t – yet. This is a core challenge of GenAI-assisted coding (and human coding too). At AI Core, we address this by actively attacking our own applications to uncover weaknesses.

Access the GitHub repo, find the Attack Agent, and fork it. Then clone the repository or download it as a ZIP, extract it locally, and open the folder in Cursor in a separate tab.

You should have:

- **One Cursor tab** running your application
- **A second Cursor tab** open for AttackAgent, which will be used to scan your app

Once you've downloaded it, ask Cursor to start it, study it, and tell you what it does.

- Start the Attack Agent and configure it to scan your application (for example, your app running on localhost). Ask it to identify known security vulnerabilities.
- Review the generated vulnerability report and work with Cursor to remediate the issues in your application.
- Re-run Attack Agent after applying fixes. Repeat this process until all reported vulnerabilities have been addressed or reasonably mitigated.

This phase reinforces an important lesson: security is an iterative process, and discovering vulnerabilities is a sign of progress—not failure.

- Review the vulnerability report generated by Attack Agent. Work with Cursor to remediate the identified issues in your application.
- Re-run Attack Agent after each round of fixes. Repeat this process until no known vulnerabilities remain or until all reported issues have been reasonably mitigated.

Congratulations

You've now completed all three phases of the AI Core Full Stack Tutorial:

- Phase One: Building real, end-to-end functionality
- Phase Two: Integrating intelligent systems safely
- Phase Three: Stress-testing and securing your work

You now have the tools — and the mindset — to build, evaluate, and improve real-world applications in collaboration with AI.