**Curriculum**

**Welcome to Full-Stack web3 Development**

Welcome! This course is designed to guide you through building full-stack web3 decentralized applications, or dApps. Throughout this journey, you'll gain the skills needed to create applications similar in concept and complexity to platforms like Aave (app.aave.com) and Uniswap (app.uniswap.org).

At their core, these dApps consist of two primary components:

1. **Frontend:** This is the user-facing interface, typically a website built with standard web technologies (HTML, JavaScript, and often frameworks like React/Next.js, which we'll cover later). Users interact directly with the frontend.
2. **Backend:** Instead of traditional servers, the backend relies on smart contracts deployed onto a blockchain (like Ethereum). The frontend communicates with these smart contracts to trigger actions, read data, and manage state in a decentralized manner.

Clicking a button on the frontend website initiates a transaction or data request that interacts with the smart contracts running on the blockchain backend. Our central resource hub, containing all code, discussions, and the course outline, can be found at our GitHub repository: github.com/Cyfrin/full-stack-web3-cu.

**Our Learning Journey: From Basics to dApps**

We believe in building a strong foundation, which is why this course starts from the absolute basics. Even if you have minimal experience with web development, you're in the right place.

Our approach is progressive. We begin with fundamental web technologies before layering on web3 concepts.

**Section 1: HTML/JS - Buy Me A Coffee**  
Our first project will be a simple "Buy Me A Coffee" dApp. Using only raw HTML and JavaScript, we'll build a minimal interface featuring:

* Buttons like Connect, getBalance, Withdraw, and Buy Coffee.
* An input field for specifying an amount (e.g., ETH Amount).

This initial project serves as a practical introduction to connecting a basic website to the blockchain, fetching data (like wallet balance), and sending simple transactions. As you progress through the course (visible in the GitHub repository's Table of Contents), we'll tackle more complex projects using tools like React/Next.js, building towards sophisticated dApps.

**The Power of Thinking: Our Learning Philosophy**

How do we truly learn and remember? We subscribe to the principle often summarized as: "Memory is the residue of thought." Deep learning and long-term retention don't come from passively watching videos or having answers handed to you. They come from actively engaging with the material, wrestling with problems, and *thinking* things through.

Encountering errors, debugging code, and getting stuck are not obstacles; they are essential parts of the learning process. When you face a challenge, your first step should be to analyze it, try to understand the potential causes, and formulate potential solutions. This active cognitive effort is what cements knowledge in your memory.

This philosophy influences our tool choices. While powerful AI-assisted code editors like Cursor (cursor.com/en) exist, they often provide suggestions or solutions proactively, sometimes before you've even had a chance to fully process the problem or articulate a specific question. While useful for experienced developers, this can hinder learning by short-circuiting the crucial thinking process.

Therefore, for this course, we will primarily use Visual Studio Code (VS Code). It provides excellent features without preempting your thought process. This encourages you to:

1. Encounter and analyze problems independently.
2. Attempt to debug and find solutions yourself.
3. Learn to formulate specific, well-defined questions when you do need help (whether asking an AI, searching online, or posting in discussion forums).

Developing the skill to ask precise questions is invaluable, both for learning and for effectively leveraging AI tools in the future. Embrace the struggle; it's where the real learning happens.

**Setting Up Your Development Environment**

To ensure you can follow along smoothly, it's crucial to set up your development environment correctly. Consistency in tooling minimizes setup friction and lets us focus on learning web3 concepts.

**Code Editor:**

* **Recommended:** Visual Studio Code (VS Code). As discussed, this is the editor used in the course videos and encourages active problem-solving.
* **Alternative:** Cursor is a powerful AI-integrated editor you might explore outside the course context, but we recommend sticking with VS Code while learning the fundamentals here.

**Operating System Considerations (Critical for Windows Users):**

* If you are using Windows, it is **essential** that you use the **Windows Subsystem for Linux (WSL)**.
* **Do not** use the native Windows command prompt (CMD) or PowerShell for the development tasks in this course. Many blockchain development tools are designed primarily for Linux-based environments, and WSL provides the necessary compatibility layer on Windows.

The immediate next step is to get your development environment set up, installing VS Code and configuring WSL if you are on Windows. Following the setup instructions carefully will prepare you for the coding sections ahead.