AI CERTs™

Blockchain+ Developer™ (5 Days)

Program Detailed Curriculum

Executive Summary

The Blockchain+ Developer certification offers a comprehensive journey into blockchain technology and smart contracts. Beginning with the origin and structure of blockchain, participants explore consensus mechanisms and the concept of smart contracts, delving into Ethereum Virtual Machine (EVM) and Solidity basics. Advanced topics cover Solidity structures, tokenization, and non-fungible tokens (NFTs). Development tools like Truffle and Ganache are introduced, along with testing techniques and DApp integration. Participants then explore private blockchain with Hyperledger Fabric, learning its architecture, Docker setup, and Golang programming. The course culminates in chaincode development, REST API integration, and chaincode auditing

Course Prerequisites

- Familiarity with general programming concepts like data structures, algorithms and networks
- Understanding of at least one legacy programming stack (e.g. Python, JavaScript, Java or similar)
- Fundamental knowledge to use command line consoles on any operating system
- Ability to understand developer concepts like SDKs, APIs, application development tools etc.
- Experience with building end to end tiered applications

Module 1

Introduction to Blockchain and Smart Contracts

1.1 Origin of Blockchain

- **Historical Development of Blockchain Technology:** The historical development of blockchain technology can be traced back to various cryptographic and decentralized concepts, including public-key cryptography and distributed computing
- **Evolution and Impact of Blockchain Technology:** Since its inception, blockchain technology has evolved rapidly, expanding beyond its original use case as the underlying technology for cryptocurrencies like Bitcoin

1.2 What is Blockchain?

- **Definition of Blockchain:** Blockchain is a decentralized, tamper-resistant ledger technology that records transactions across a network of computers
- **Cryptographic Security:** Cryptographic security in blockchain involves the use of hashing, digital signatures, and encryption to guarantee the integrity, confidentiality, and authenticity of data

1.3 Consensus Mechanisms

- **Proof of Work (PoW) and Proof of Stake (PoS):** PoW requires participants to perform computationally intensive tasks to validate transactions and create new blocks. It's known for its security but consumes significant energy
- PoS selects validators based on the amount of cryptocurrency they hold and are willing to "stake" as collateral, promoting energy efficiency but sometimes criticized for potential centralization

- **Alternative Consensus Mechanisms:** This category encompasses various other consensus mechanisms like Delegated Proof of Stake (DPoS), where token holders vote for delegates to validate transactions

1.4 What are Smart Contracts?

- **Smart Contract Definition:** Self-executing contracts with predefined rules and conditions encoded on the blockchain and execution environment: Smart contracts run on blockchain platforms, providing decentralized and tamper-proof execution
- **Implementation and Management:** programming languages for smart Contracts are the languages like Solidity and Vyper are used to write smart contracts on platforms like Ethereum

1.5 Bitcoin Blockchains

- **Blockchain Structure and Operations:** Blockchain operates as a decentralized ledger where data is stored in blocks, linked together chronologically using cryptographic hashes, ensuring immutability and integrity through consensus mechanisms like proof of work or proof of stake
- **Network and User Interaction:** Blockchain enables peer-to-peer transactions without intermediaries, with users interacting through interfaces like wallets and DApps

Module 2

Ethereum Virtual (EVM) and Solidity Basics

2.1 What is an EVM and Ethereum?

- **Ethereum Infrastructure and Components:** Ethereum's infrastructure comprises a decentralized network of nodes running the Ethereum Virtual Machine (EVM), which executes smart contracts
- **Smart Contracts and Development Environment:** Smart contracts are self-executing agreements with code stored on the blockchain, automating contract execution and enforcement. Ethereum's development environment includes tools like Solidity for smart contract programming

2.2 Wallets Introduction and Creation

- **Ethereum Wallets:** Ethereum wallets are digital tools that allow users to store, manage, and interact with Ethereum tokens and assets on the Ethereum blockchain
- **Ethereum Account Management:** Ethereum account management involves creating, securing, and accessing accounts on the Ethereum blockchain. Each Ethereum account consists of a pair of cryptographic keys

2.3 Introduction to Remix Editor with Metamask

- **Remix IDE:** Remix IDE is an open-source web-based integrated development environment (IDE) specifically designed for Ethereum smart contract development
- **Metamask Integration:** Metamask is a browser extension and mobile application that serves as an Ethereum wallet and gateway to decentralized applications (DApps) on the Ethereum blockchain

2.4 Smart Contract Basic Structure

- **Solidity Smart Contract Structure:** Solidity, a programming language specifically designed for writing smart contracts on the Ethereum blockchain, follows a structured format
- **Interaction with Smart Contracts:** Interacting with smart contracts on the Ethereum blockchain involves sending transactions to their respective addresses, triggering the execution of specific functions defined within the contract code

2.5 Variables, If/Else, Strings, Loops, Arrays, Test Tokens

- **Basic Solidity Programming Constructs:** Solidity, the programming language used for Ethereum smart contract development, encompasses various fundamental constructs
- **Control Flow and Iteration in Solidity:** Control flow and iteration in Solidity enable developers to implement conditional statements and loop structures to control the flow of execution within smart contracts

Module 3

Advanced Solidity and Structures

3.1 Libraries, Interfaces, Modifiers

- **Solidity Advanced Constructs:** Solidity, the programming language for Ethereum smart contracts, offers advanced constructs to facilitate complex contract development
- **Solidity Contract Management:** Solidity provides features for managing contracts on the Ethereum blockchain, including deployment, interaction, and upgradability

3.2 Structures, Enums, ABI, Calldata, Events, and Transfers

- **Solidity Data Structures:** Solidity, the programming language used for Ethereum smart contract development, supports various data structures to organize and manipulate data efficiently
- **Solidity Data Handling and Communication:** Solidity provides mechanisms for handling data and communication within smart contracts on the Ethereum blockchain

3.3 Contract-to-Contract Calls

- **Contract to Contract calls:** In Solidity, the programming language for Ethereum smart contracts, contract-to-contract calls enable communication and interaction between different smart contracts deployed on the Ethereum blockchain

3.4 Address and Address Payable

- **Address and address payable:** The address data type is a 20-byte value that represents an Ethereum address. It is used to store Ethereum addresses and interact with external accounts or contracts on the Ethereum blockchain

3.5 Receive and Fallback Functions

- **Receive and Fallback functions:** It was Introduced in Solidity version 0.6.0, the receive function is a special function that is automatically called when a contract receives Ether without any data or when it is sent Ether explicitly using a simple transfer operation
- **Solidity Data Handling and Communication:** Solidity provides mechanisms for handling data and communication within smart contracts on the Ethereum blockchain

3.6 Upgradeable Contracts

- **Receive and Fallback functions:** In Solidity, the receive and fallback functions are special functions used to handle incoming Ether transactions in a contract

3.7 Openzepplin Libraries

- **Openzepplin Libraries:** OpenZeppelin is a leading open-source framework for building secure and audited smart contracts on Ethereum and other blockchain platforms

Module 4

Tokenization and NFTs

4.1 ERC20 Token Creation

- **ERC Standards and Token Contract Creation:** ERC standards, or Ethereum Request for Comments standards, are sets of rules and guidelines defining various functionalities and interfaces for Ethereum-based tokens and smart contracts
- **Token Contract Structure and Functionality:** The token contract structure and functionality are designed to provide a standardized interface for interacting with tokens while also allowing for customization and extensibility based on the specific requirements of the token issuer

4.2 NFT, NFT Minting, IPFS, Security, and Pinata Cloud

- **Introduction to ERC721:** ERC721 is a standard interface for non-fungible tokens (NFTs) on the Ethereum blockchain
- **NFT Minting and Transfer Processes:** The minting and transfer processes for ERC721 tokens involve the creation and ownership transfer of unique digital assets represented by NFTs

Module 5

Development Tools and Techniques

5.1 Truffle, Ganache, and Hardhat

- **Development Frameworks and Tools:** Development frameworks and tools play a crucial role in streamlining the process of building, testing, and deploying blockchain applications
- **Local Blockchain Networks:** Local blockchain networks are private or semi-private blockchain networks deployed locally on developer machines or within development environments for testing and experimentation purposes

5.2 Metamask Wallet

- **Metamask Wallet:** Metamask is a browser extension and mobile application that serves as an Ethereum wallet, allowing users to manage Ethereum accounts, securely store private keys

5.3 Remix Development Environment

- **Remix Development Environment:** Remix is a web-based integrated development environment (IDE) specifically designed for Ethereum smart contract development

5.3 Localnet, and Testnet Deployment

- **Private Ethereum:** Private Ethereum refers to a customized Ethereum blockchain network set up for private or internal use, allowing organizations or developers to create and test decentralized applications
- **Geth - go lang:** Geth, short for "Go Ethereum," is the official Go implementation of the Ethereum protocol

Module 6

DApp Integration and Testing

6.1 Web3.0 Integration with JS

- **Introduction to Web3.0 Integration:** Web3.0 integration refers to the incorporation of Web3 technologies into web applications, enabling direct interaction with blockchain networks and decentralized protocols
- **Contract Interaction and Transaction Execution:** Contract interaction and transaction execution in the context of blockchain development involve communicating with and executing functions on smart contracts deployed on blockchain networks

6.2 Wallet Creation and Sending Transactions

- **Wallet Creation and Transaction Management:** Wallet creation involves generating a unique public-private key pair, allowing users to securely store and manage their digital assets
- **MetaMask Integration:** MetaMask integration enables users to interact with Ethereum-based decentralized applications (DApps) directly from their web browsers

Module 7

Introduction to Private Blockchains - Hyperledger Fabric

7.1 Public Vs Private vs. Consortium Blockchain Frameworks

- **Public and Private Blockchain Frameworks:** Public blockchain frameworks like Ethereum and Bitcoin are open, permissionless networks where anyone can participate, view transactions, and interact with the blockchain
- **Consortium Blockchain Frameworks:** Consortium blockchain frameworks, like Quorum and R3 Corda, are semi-decentralized networks where multiple organizations collaborate to maintain the blockchain

7.2 Introduction to the Hyperledger Fabric

- **Hyperledger Fabric Architecture and Components:** Hyperledger Fabric architecture consists of several key components including a peer-to-peer network, smart contracts
- **Setup Hyperledger Fabric Business Network:** Setting up a Hyperledger Fabric business network involves configuring network components, defining organizations and their roles, deploying smart contracts (chaincode).

7.3 Hyperledger Projects

- **Hyperledger Blockchain Frameworks:** Hyperledger is an umbrella project under the Linux Foundation that hosts a variety of open-source blockchain frameworks and tools aimed at facilitating enterprise-grade blockchain development

Module 8

Deep Dive into the Hyperledger Fabric

8.1 Basic Concepts of HLF

- **Distributed Ledger Technology (DLT):** Distributed Ledger Technology (DLT) is a decentralized database system that records and stores transactions across multiple locations
- **Smart Contracts and Chaincode:** Smart contracts, also known as chaincode in some blockchain frameworks, are self-executing contracts with pre-defined rules and conditions stored on a blockchain

8.2 Docker Introduction

- **Introduction and Installation:** Docker is a platform that enables developers to develop, deploy, and run applications using containers
- **Docker Compose and Docker Images:** Docker Compose is a tool for defining and running multi-container Docker applications

8.3 Commands and Setup

- **Prerequisites and Installation:** Installation involves setting up the required development environment, including installing development tools, frameworks, and libraries specific to the chosen blockchain platform
- **Chaincode Development and Deployment:** Chaincode development involves writing the smart contracts (chaincode) that define the business logic and rules for interacting with the blockchain network
- **Wallets and Connection Profiles:** Wallets are software applications used to securely store and manage cryptographic keys (e.g., private keys) required for interacting with blockchain networks

Module 9

Golang Programming for Hyperledger Fabric

9.1 Installation and Path Setup

- **Golang Installation:** Golang, or Go, installation involves downloading and installing the Go programming language compiler and tools from the official Golang website or package manager of the operating system
- **Setting up Go Environment Variables:** Setting up Go environment variables involves configuring the system's environment variables like PATH and GOPATH

9.2 VS Code Plugin Setup, Variables, Strings, Conditional Statements, and Loops

- **Code Plugin Setup:** Code plugin setup involves installing and configuring plugins or extensions for code editors or integrated development environments (IDEs) such as Visual Studio Code (VS Code) to enhance the development experience when working with specific programming languages or frameworks

9.3 Basics of the Language

- **Basic Concepts in Golang:** Basic concepts in Golang include understanding the language syntax, data types, variables, constants, control flow (if statements, loops), functions, packages, and error handling
- **Programming Constructs in Golang:** Programming constructs in Golang encompass various language features and constructs used to structure and control the flow of Go programs, including conditionals (if, else), loops (for, range).

Module 10

Chaincode Structure and Error Handling

10.1 Chaincode Explanation using Fabric Samples, Test-network Explanation using Linux Scripting

- **Fabric Samples Overview:** Fabric Samples provide a collection of sample applications, smart contracts (chaincode), and network configurations to showcase various features and capabilities of Hyperledger Fabric
- **Chaincode and Test Network Explanation:** In Hyperledger Fabric, chaincode refers to the smart contracts deployed on the blockchain network, defining the business logic and rules for updating the ledger state through transactions

10.2 Error Handling

- **Error Handling in Chaincode:** Error handling in chaincode involves implementing mechanisms to gracefully handle and propagate errors that occur during the execution of smart contracts (chaincode).
- **Best Practices and Considerations:** Use Explicit Error Returns, Error Wrapping, Error Propagation, Graceful Failure etc

10.3 Error Codes and Messages

- **Error Codes and Messages:** Understanding common error codes and messages, troubleshooting techniques, and best practices for effective error handling in software development.

10.4 Logging Errors

- **Logging Errors:** Learn best practices for logging errors efficiently, debugging code, and improving system stability.

10.5 Handling Panics

- **Handling Panics:** Discover techniques to mitigate and recover from panics in blockchain systems, ensuring stability and resilience. Practical strategies for implementation.

Module 11

Custom Chaincode

11.1 Extending the Default Chaincode

- **Default Chaincode Overview and Extension:** The default chaincode in Hyperledger Fabric provides basic functionalities for asset management and transaction processing
- **Identifying Extension Points and Implementing Custom Functions:** Identifying extension points involves analyzing the default chaincode to determine areas where custom logic or additional functionalities can be integrated seamlessly
- **Testing and Validation:** Testing and validation are crucial steps in the extension process to ensure the reliability, correctness, and performance of the extended chaincode

11.2 Chaincode Deployment

- **Chaincode Packaging, Installation, and Instantiation:** In Hyperledger Fabric, chaincode is packaged as a compressed file containing the source code and metadata required for deployment
- **Lifecycle Management and Approval/Commit Process:** Chaincode lifecycle management involves the process of updating, upgrading, and retiring chaincode versions on the Hyperledger Fabric network

11.3 REST API Integration with Front End

- **REST API Architecture and Chaincode Interaction Endpoints:** The REST API architecture for Hyperledger Fabric provides endpoints for interacting with chaincode deployed on the blockchain network
- **Data Serialization and Deserialization, Error Handling, and Response Codes:** Data serialization and deserialization are processes of converting complex data structures into a format suitable for transmission over the network and vice versa

Module 12

Smart Contract Auditing and Tools Hyperledger Fabconnect, and Firefly

12.1 Why Smart Contract Audits are Necessary

- **Introduction to Tools for Smart Contract Development:** Tools for smart contract development are software applications, libraries, or frameworks designed to assist developers in creating, testing, debugging, and deploying smart contracts on blockchain platforms like Ethereum
- **Role of Slither and Solhint:** Slither and Solhint are tools used in the development of smart contracts written in Solidity, the programming language for Ethereum smart contracts

12.2 Introduction to Firefly, Fabconnect, and Blockchain Explorer

- **Introduction to Tools for Blockchain Development:** Tools for blockchain development encompass a wide range of software applications, libraries, frameworks, and platforms designed to assist developers in creating, deploying, and managing blockchain-based applications and networks
- **Functionality and Use Cases:** The functionality and use cases of tools for blockchain development vary depending on their specific purpose and target audience