CURRICULUM FOR DAPPS & SOLIDITY COURSE

MODULE 1: SMART CONTRACTS, ETHEREUM VM AND DAPPS

Students begin week three by delving deeper into the Ethereum network and DApps. They consider DApp architecture, use cases, GHOST protocol and Solidity. Students use geth to build or join an Ethereum network. Students write, compile, deploy and run contracts using Remix and MetaMask on the Ropsten testnet. During a live coding session, students deploy a token on the Ethereum Ropsten testnet.

MODULE 2: SOLIDITY BASICS

In this module, students continue to develop, in detail, their understanding of Solidity programming including: contracts, functions, data types, variables, conditionals, loops, maps, etc. They use the Remix IDE to compile, deploy, run and debug smart contracts. In a live coding session, students write and test Solidity contracts. During a live coding session, students write Solidity contracts and test simple token implementation.

MODULE 3: SOLIDITY ADVANCED

Focusing on writing smart contracts in Solidity, this module takes students through, coding their own assets. During live coding sessions, student write payable contracts, receive funds, write contract emitting events, and create a simple timed auction contract and a company shares contract.

MODULE 4: BUILDING DAPPS WITH ETHEREUM

This session focuses on using the Web3 API and other libraries to interact with the Ethereum network. Students are familiarized with Ganache and Truffle. Using Truffle to create an app, students test-run a contract in Ganache. Students begin their second and final practical project, focused on building a DApp with Solidity and Web3. Students create a certificate repository DApp during a live coding session.

MODULE 5: DAPPS - CONCEPTS

This session highlights core concepts in DApp development, including DApp architectures, decentralized storage and IPFS, the Web3 API and Metamask. Students learn about multi-signature wallets, test coding client-side wallets, publish files and folders in IPFS and utilize Web3, Ethers.js and Metamask to create sample DApps.

MODULE 6: WORKING ON THE DAPP PRACTICAL PROJECT: ARCHITECTURE AND UI

Students will start working on their DApp practical project: from the concept to architecture, UI design and writing the smart contracts for the DApp.

MODULE 7: SERVER-SIDE WEB3 API

In this module, students extend their knowledge in DApp development by learning how to access the Ethereum network from the server-side using JavaScript, Infura, Ether. js, C#, Python and Java APIs. During live coding sessions, students test various types of contracts in the Ropsten testnet. Students also explore client-side wallets, including wallet generation, and signing and sending transactions.

MODULE 8: BLOCKCHAIN AND SMART CONTRACT SECURITY

This session highlights blockchain security, wallet security, cold wallets, smart contract security, and explains the DAO hack, Parity hacks, Tether token hack, Bitcoin Gold scam and other blockchain hacks as case studies in crypto-security. Students are introduced to various vectors of attack and inspect suspicious sites for scams. Students then consider smart-contract security tools and audit processes and practice a simulated hack on a vulnerable contract.

MODULE 9: OTHER BLOCKCHAIN PLATFORMS

In this module, we look at consortium blockchain platforms based on Solidity and EVM and alternative to the Ethereum network like Qtum and RSK. Students code and run smart contracts in Qtum and RSK during a live coding session.

MODULE 10: PRACTICAL PROJECT - SMART CONTRACT AND DAPP UI

In this module students work on their second practical project: building a DApp using Solidity smart contract, written in Truffle framework, deployed on the Ropsten testnet, accessed from client-side UI or server-side code through Web3 or other Ethereum library, involving more complicated elements like decentralized storage and unit testing.

PRACTICAL PROJECT DEVELOPMENT & SUPPORT

Students will develop two practical projects during the course. In Full Immersion mode, the two projects will be developed during the four weeks following the face-to-face training. In Hybrid Immersion mode, students will complete a project in the final two weeks of each course. The training team will provide technical assistance daily through online chats and discussions.