BDCPC301 - **BLOCKCHAIN FUNDAMENTALS**

BDCPC301 - **Apply Fundamentals of Blockchain**

Competence

**RQF Level: 5 Learning Hours**

**100**

**Credits: 10**

**Sector: ICT AND MULTIMEDIA**

**Trade: SOFTWARE DEVELOPMENT**

**Module Type: Specific Module**

**Curriculum: CTSWD5003 TVET CERTIFICATE V IN SOFTWARE DEVELOPMENT**

**Copyright: © Rwanda TVET Board, 2024**

**Issue Date: February 2024**

x

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Purpose statement** | This specific module describes the skills, knowledge and attitude required to Apply Fundamentals of Blockchain. This module is intended to prepare students pursuing TVET Level 5 in software development. Upon completion of this module, the learner will be able to Design Blockchain system architecture, Apply Solidity Basics, Develop Smart contracts system and Apply Frontend Integration | | | | | |
| **Learning assumed to be in place** | Develop frontend using react JS | | | | | |
| **Delivery modality** | **Training delivery** | | **100%** | **Assessment** | | **Total 100%** |
| Theoretical content | | 30% | Formative assessment | 30% | 50% |
| Practical work: | | 70% | 70% |
| Group project and presentation | 20 % |
| Individual project /Work | 50 % |
|  | | Summative Assessment | | | 50% |

**Elements of competence and Performance Criteria**

|  |  |
| --- | --- |
| **Elements of competence** | **Performance criteria** |
| 1. **Design blockchain system architecture** | 1.1 System requirements are properly identified based on project purpose |
| 1.2. Blockchain technologies are effectively selected based on project requirements. |
| 1.3. Architecture of blockchain application is clearly designed based on specific requirements |
| 1. **Apply Solidity Basics** | 2.1 Environment is properly prepared based on development tools standards |
| 2.2 Solidity concepts is clearly applied based on solidity principles |
| 2.3 Solidity functions are properly created based system requirements |
| 2.4 Function interaction is accurately implemented based on blockchain technology |
| 2.5 Gas Costs are accurately optimized according to function definition |
| 1. **Develop Smart contracts** **system** | 3.1 Smart contracts are properly created based on blockchain technology |
| 3.2. Tokens are properly created based on token standards |
| 3.3 Security of Smart contracts are properly Applied based on specific requirements |
| 3.4. Smart contracts are accurately deployed based on specific requirements |
| 1. **Apply frontend Integration** | 4.1 Web3 dependencies is clearly installed based on versions |
| 4.2 Smart contract is properly connected based deployed version |
| 4.3 Functions are clearly used based on smart contract definitions |

### 

|  |  |  |
| --- | --- | --- |
| **Knowledge** | **Skills** | **Attitudes** |
| * Identification of blockchain requirements. * Description of blockchain key concepts * Description of types of Consensus mechanism * Description of the types of attacks and vulnerabilities of blockchain * Description of blockchain architecture * Description of Solidity programming language. * Description of smart contracts. * Description of wallet | * Application of blockchain use cases. * Designing the architecture of blockchain application * Implementation of function Interaction * Optimization of Gas Costs * Creation of smart contracts * Writing smart contract * Creation of Tokens * Applying of security of smart contracts. * Deployment of smart contracts * Configuration of contract network * Connection of smart contract. * Deployment of web3 frontend. | * Being Detail oriented * Being a good observer * Being Creative * Having critical thinking * Being Problem solver * Being self-motivated * Being Continuous learner * Having Adaptability * Being Innovative * Teamwork * Hardworking |

**Course content**

|  |  |  |
| --- | --- | --- |
| **Learning outcome** | | **At the end of the module the learner will be able to:**  1. Design blockchain system architecture  2. Apply Solidity Basics.  3. Develop Smart contracts system.  4. Apply frontend Integration |
| **Learning outcome 1: Design blockchain system architecture** | | **Learning hours: 10 hours** |
| **Indicative content** | | |
| * **Identification blockchain requirements** * Introduction to blockchain * Define * blockchain * cryptography * History of blockchain * Types of Blockchain * Blockchain Principles * Functionalities of blockchain * Pros and Cons of blockchain * Blockchain Company solutions * Description of blockchain key concepts * Essential components of wallet (Private Keys, Public Keys, Addresses) * Transactions, Merkle Trees, and Blocks * Hierarchical Deterministic Wallets, Mnemonic Seeds and Smart Contracts * Working of Blockchain Transaction * Apply blockchain use cases * **Selecting Blockchain Technologies** * Description of Blockchain technology stack principles * Consensus Layer (PoW,PoS, etc) * Network Layer (Ethereum's Peer-to-Peer Network) * Protocol Layer (Ethereum's EVM -Ethereum Virtual Machine) * Smart Contracts Layer (Decentralized Finance (DeFi) Platforms) * Application Layer (CryptoKitties) * Storage Layer (IPFS - InterPlanetary File System) * Identity and Access Management (SelfKey) * Security and Encryption (Public and Private Key Encryption) * Interoperability Layer (Polkadot) * Scalability Solutions (Lightning Network for Bitcoin) * Governance Mechanisms (Tezos) * User Interfaces (MetaMask) * Describe types of Consensus mechanism * Proof of Work * Proof of Stake * Delegated Proof of Stake * Proof of Authority * Proof of Weight * Use appropriate Consensus Mechanism (Proof of Work, Proof of Stake) * Identify the types of attacks and vulnerabilities of blockchain * Attack in consensus mechanism * Sybil Attack (spamming the network, disrupt communication among nodes) * Double Spending * Eclipse Attack * Smart Contract Vulnerabilities (re-entrancy attacks, integer overflow/underflow). * DDoS Attack (Distributed Denial of Service) * Blockchain Spamming * Long-Range Attack * Selfish Mining * Routing Attacks * Transaction Malleability * Consensus Manipulation * **Designing the architecture of blockchain application** * Description of blockchain architecture * Components * Connection * Instance relation * Designing system architecture * Design Blockchain based Systems * Designing the Blockchain Network * Design Smart Contract * Drawing blockchain architecture * Identify the Use Case * Identify third party Integration * Identify the Consensus Mechanism * Identify the Platform * Design the Blockchain Instance * Design the Architecture | | |
| Resources required for the learning outcome | | |
| **Equipment** | * Computer * Projector | |
| **Materials** | * Whiteboard and markers * Internet | |
| **Tools** | * V**S**Code * Remix, Truffle, and Hardhat * Web3.js and ether.js | |
| **Facilitation techniques or Learning activity** | * Brainstorming * Group discussion * Demonstration * Practical exercise * Trainer guided | |
| **Formative assessment methods /(CAT)** | * Oral assessment * Written assessment * Practical assessment | |

|  |  |  |
| --- | --- | --- |
| **Learning outcome 2: Apply Solidity Basics** | | **Learning hours: 20 hours** |
| **Indicative content** | | |
| * **Preparation of environment** * Description of key terms * Solidity * Syntax * Data types * Variables * identifiers * Arrays * Struct * Functions * Control structures * State variables * Modifiers (conditions) * smart contract * Visibility and Access Control * Ethereum * Ethereum Virtual Machine (EVM) * Set up solidity environment * Installing Code editor (remix, visual studio code) * Installing node.js and npm (Node Package Manager) for package management. * Installing Solidity compiler (solc) and Ethereum development tools (e.g., Truffle, Hardhat). * **Applying solidity concepts** * Data types and variables * Use of functions * Control structures * arrays and structs * Events and logging * Error handling * **Implementing function Interaction** * Connect to wallet * Metamask Wallet * Trust wallet * Access the Contract Address * Use a Blockchain Explorer * Perform function operations * Read only operations * Write operation * **Optimizing Gas Costs** * Proper analysis of Gas cost * Calculating the cost of Ethereum transfer * Heavy and Light functions * Block limit * Opcode Gas cost * Non-payable functions * Elaboration of Storage * Smaller Integers, Unchanged Storage Values, Arrays * Refunds and Setting to Zero * ERC20 Transfers * Storage Cost for Files * Structs and Strings, Variable Packing, Array Length * Optimization of Memory cost * Memory vs Call data * Mappings vs Arrays * Freeing Up Unused Storage * immutable and constant * Access Modifier * Indexed Events * Minimizing On-Chain Data | | |
| Resources required for the indicative content | | |
| **Equipment** | * Computer * Projector | |
| **Materials** | * Whiteboard and markers * Internet | |
| **Tools** | * Visual Studio Code, IntelliJ IDEA for coding and testing * Remix, Truffle, and Hardhat for smart contract development * Web3 libraries | |
| **Facilitation techniques or Learning activity** | * Brainstorming * Group discussion * Trainer guided * Demonstration * Practical exercise | |
| **Formative assessment methods /(CAT)** | * Oral assessment * Written assessment * Practical assessment | |

|  |  |  |
| --- | --- | --- |
| **Learning outcome 3: Develop smart contracts system** | | **Learning hours: 45 hours** |
| ● **Creating smart contracts**   * Applying of Solidity programming language * Mapping, Arrays, Structs and Error handling * Use of modifiers(conditions), Interfaces, Events, and Inheritance * Contracts composition * Storage locations * Compiling * Test with hardhat (chai, mocha) * Writing smart contract   **● Creating Tokens**   * Implementation of Fungible token (FT) standards * ERC20 Token Standard * Writing an ERC20 Token in Solidity * Implementation of Non-Fungible Token standards (NFT) * ERC721 standard * Write NTF smart contracts using ERC721 standard * ERC1155 Multi Token Smart Contract   ● **Applying security of smart contracts**   * Protection of smart contracts against Re-entrancy Attack * Securing smart contract using Escrow Service Contract * Usage of third-party libraries * OpenZeppelin (includes safemath) * Chainlink   ● **Deploying smart contracts**   * Selection of development blockchain network * Local network (Ganache) * Public network (e.g mainnet, testnet) * Create infrastructure services for blockchain applications * Alchemy * Infura * Deploy contract * Truffle * Hardhat | | |
| **Indicative content** | | |
| Resources required for the indicative content | | |
| **Equipment** | * Computer * Projector | |
| **Materials** | * Whiteboard and marker**s** * Internet | |
| **Tools** | * Studio Code, IntelliJ IDEA * Remix, Truffle, and Hardhat * Web3 libraries | |
| **Facilitation techniques or Learning activity** | * Brainstorming * Group discussion * Trainer guided * Demonstration * Practical exercise | |
| **Formative assessment methods /(CAT)** | * Oral assessment * Written assessment * Practical assessment | |

|  |  |  |
| --- | --- | --- |
| **Learning outcome 4: Apply frontend integration** | | **Learning hours: 25 hours** |
| **Indicative content** | | |
| * **Installing web3 dependencies** * Configure contract network * Install contract extension in browser for development (eg. Metamask) * Create wallet * Load enough balance in wallet(faucet) * Connect to smart contract wallet using frontend * Install web3 libraries (e.g ether.js,web3.js) * Connect to smart contract using keys(contract address, Application Binary Interface - ABI) * **Connecting smart contract** * Consume smart contract functions based on defined functionalities * Create instance of smart contract * **Use of function** * Implement operations based on smart contract predefined functions * Deploy web3 frontend based on specific requirements * Test web application * Build production bundles * Configure keys on production environment variables * Deploy production builds | | |
| Resources required for the indicative content | | |
| **Equipment** | * Computer * Projector | |
| **Materials** | * Internet * Whiteboard * markers | |
| **Tools** | * Visual Studio Code * Remix, Truffle, and Hardhat * Web3 libraries | |
| **Facilitation techniques or Learning activity** | * Brainstorming * Group discussion * Demonstration * Practical exercise * Trainer guided | |
| **Formative assessment methods /(CAT)** | * Oral assessment * Written assessment * Practical assessment | |

**Integrated/Summative assessment (For specific module)**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CareSphere is a healthcare tech start-up company based in Kigali city, Gasabo District, specializing in secure patient record storage. Our platform enables patients, doctors, and medical facilities to securely store, access, and exchange medical records while preserving privacy. Leveraging blockchain technology, we aim to revolutionize healthcare data management with immutable and transparent record-keeping.  As a blockchain developer at CareSphere, your role involves designing and implementing a decentralized healthcare records management platform. Utilizing blockchain for secure storage and writing smart contracts, you will develop a user-friendly frontend and a robust backend (smart contracts) to facilitate record interactions.  **Project Tasks:**   1. Requirement Gathering and Analysis 2. Blockchain Selection and Configuration 3. Smart Contract Development 4. Frontend and Backend integration 5. Security and Privacy Measures 6. User Testing and Verification 7. Documentation and Training   **Resources**   |  |  | | --- | --- | | **Tools** | * Integrated Development Environment (IDE) for smart contract development (e.g., Remix, Hardhat, Truffle Suite) * Web3.js or similar libraries for frontend smart contracts communication with the blockchain * pre-designed frontend | | **Equipment** | * Servers for frontend hosting * Secure key management system for user authentication | | **Materials/ Consumables** | * SSL certificates for secure communication * Hosting services for web deployment |   **This task will be performed within 8 hours.**  **NB:  all materials, tools and equipment are provided.** | | | | | |
| Assessable outcomes | Assessment criteria (Based on performance criteria) | Indicator | Observation | | Marks allocation |
| Yes | No |
| **1. Design blockchain architecture** | 1.1. System requirements are properly identified | Ind1: Functional and non-functional requirements are identified |  |  | **3** |
| Ind2: Acceptance criteria of each requirement are defined |  |  | **3** |
| 1.2. Blockchain technologies are effectively selected | Ind 1: blockchain technologies are described |  |  | **3** |
| Ind 2: Consensus mechanism are Appropriately used |  |  | **5** |
| 1.3. Architecture is clearly designed | Ind1: Technologies and Tools are selected |  |  | **3** |
| Ind2: Use cases are identified |  |  | **5** |
| Ind 3: Architecture is designed |  |  | **5** |
| **3. Develop Smart Contracts** | 3.1 Smart Contract is properly created | Ind1: Solidity language is used |  |  | **5** |
| Ind2: smart contract is well written |  |  | **5** |
| 3.2. smart contract is correctly secured. | **Ind1.** Smart contract is accurately protected |  |  | **8** |
| **Ind2.** Third-party libraries are used |  |  | **5** |
| 3.3. Smart contract is accurately deployed | **Ind 1.** blockchain network is Selected. |  |  | 3 |
| Ind 2. Infrastructure services of blockchain is created. |  |  | 5 |
| **Ind3.**Smart contract is deployed |  |  | 8 |
| **4. Front end Integration** | 4.1 Web3 dependencies are clearly installed. | **Ind1:**contract network is well configured. |  |  | **6** |
| **Ind2:** smart contract wallet are well connected using front end |  |  | **6** |
| 4.2**.** Smart contract is successful connected | **Ind 1:** Smart contract functions are consumed based defined functionalities**.** |  |  | **5** |
| **Ind 2.** Instance of smart contract is Created. |  |  | **7** |
| 4.3**.** smart contractFunctions are well applied | **Ind 1.** Smart contract predefinedfunctions are well implemented |  |  | **5** |
|  | Ind 2. Web3 front-end is deployed based on specific requirements |  |  | **5** |
| Total marks | |  | | | |
| Percentage Weightage | | 100% | | | |
| Minimum Passing line % (Aggregate): 70% | | | | | |

References:

1. Smart Contracts: Building Blocks for Digital Markets by Nick Szabo. Published:1996
2. Mastering Ethereum: Building Smart Contracts and DApps by Andreas M. Antonopoulos and Gavin Wood. Published:December 23, 2018
3. Ethereum Smart contract Development: Build Decentralized Applications with Solidity by Mayukh Mukhopadhyay. Published: February 23, 2018.
4. Web3: A Radical Guide by Chris Dixon and Naval Ravikant. Published:October 30, 2021
5. Mastering Bitcoin: Unlocking Digital Cryptocurrencies by Andreas M. Antonopoulos, published: July 1,2014
6. Blockchain Basics: A Non-Technical Introduction in 25 Steps by Daniel Drescher,published: March 14, 2017
7. Blockchain Revolution: How the Technology Behind Bitcoin and Other Cryptocurrencies is Changing the World by Don Tapscott and Alex Tapscott. Published: June 12, 2018
8. Blockchain Applications: A Hands-On Approach by Arshdeep Bahga and Vijay Madisetti. Published: January 31, 2017

# Bibliography

web3. (2016). *web3.js - Ethereum JavaScript API*. Retrieved 04 28, 2024, from web3js.readthedocs.io: https://web3js.readthedocs.io/en/v1.10.0/