Catalyst Africa
Town Hall

Cardano Smart Contract Development for Beginners

TABLE OF CONTENT

Cardano Smart Contract Development for Beginners: A Week-Long Guide

This syllabus outlines a week-long program designed to equip you with the foundational knowledge to embark on your journey as a Cardano smart contract developer. Each day will cover specific topics, building your skills progressively.

Learning Objectives:

- Gain a fundamental understanding of blockchain technology and Cardano's architecture.
- Grasp the concept of smart contracts and their role within blockchain applications.
- Explore different smart contract development tools available on Cardano.
- Build a basic smart contract using Marlowe, a beginner-friendly language.
- Learn essential testing and debugging practices for smart contracts.
- Identify security best practices for developing secure smart contracts.
- Explore advanced topics and resources for further development on Cardano.

Day 1: Understanding Blockchain Fundamentals

This first day lays the foundation for your journey towards Cardano smart contract development. We will spend 3 hours diving deep into the world of blockchain technology, focusing specifically on Cardano's unique architecture.

Module 1.1: Introduction to Blockchain Technology

Demystifying blockchain is essentially about breaking down the complex technology of blockchain into simpler, easier-to-understand terms and concepts. Here are the key points that are typically covered in our guide:

1. **Distributed ledgers and decentralization:** Blockchain utilizes a distributed ledger, which means that data is stored and synchronized across multiple locations or nodes. This decentralized approach removes the need for a central authority, making the system more transparent, secure, and resilient to tampering.

2. **Immutability and data integrity:** Immutability refers to the idea that once data is recorded on the blockchain, it cannot be altered or deleted. This feature ensures the integrity and trustworthiness of the data stored on the blockchain, making it highly reliable for various applications.

3. **Consensus mechanisms:** Consensus mechanisms are algorithms or protocols that ensure all nodes in a blockchain network agree on the validity of transactions. Proof of Stake is one such consensus mechanism where validators are chosen to create new blocks based on the amount of cryptocurrency they hold. This helps in validating transactions securely and efficiently.

4. **Benefits and limitations:** Blockchain technology offers numerous benefits such as increased security, transparency, efficiency, and

reduced costs in various industries like Cardano. However, it also has limitations like scalability issues, high energy consumption, and regulatory challenges that need to be addressed for widespread adoption. Fortunately, Cardano has already overcome some challenges while other alternative solutions are being developed.

5. **Real-world applications:** Blockchain has applications beyond cryptocurrencies, including supply chain management, healthcare data management, voting systems, smart contracts, and digital identity verification. These real-world use cases demonstrate the versatility and potential impact of blockchain technology across different sectors.

Exploring Different Blockchain Platforms : We'll briefly compare and contrast some popular blockchain platforms like Bitcoin, Ethereum, and Hyperledger Fabric to highlight Cardano's unique features and approach.

1. Bitcoin:

- Bitcoin is the first and most well-known blockchain platform, primarily used as a decentralized digital currency.
- It uses a proof-of-work consensus algorithm, which consumes a lot of computational power.
- Bitcoin's scripting language is limited, mainly focused on transactions.
- Limited smart contract capabilities compared to platforms like Ethereum or Cardano.

2. Ethereum:

- Ethereum is a decentralized platform that enables programmable smart contracts and decentralized applications (dApps).
- It uses a proof-of-stake consensus algorithm, transitioning from proof-of-work to improve scalability and energy efficiency.
- Ethereum introduced the concept of smart contracts, allowing developers to create custom applications on the blockchain.
- Ethereum is widely used for decentralized finance (DeFi) applications and token creation.

3. Hyperledger Fabric:

- Hyperledger Fabric is an open-source enterprise-focused blockchain platform developed by the Linux Foundation.
- It is designed for private, permissioned blockchains, catering to businesses and enterprises.
- Hyperledger Fabric provides modular architecture, allowing for customization of various components to meet specific business requirements.
- Supports private channels, enabling confidential transactions between parties.

Unique Features of Cardano:

- Cardano is a blockchain platform that focuses on sustainability, scalability, and security.
- Utilizes a proof-of-stake consensus algorithm called Ouroboros, which is energy-efficient and secure.
- Cardano's approach includes heavy emphasis on peer-reviewed research and scientific rigor in its development process.

- It aims to provide a scalable and interoperable platform that can support various decentralized applications and systems.

In summary, while Bitcoin paved the way for blockchain technology, Ethereum brought smart contracts to the forefront, and Hyperledger Fabric targeted enterprise use cases, Cardano sets itself apart with its focus on sustainability, rigorous research, and scalability to provide a secure and inclusive blockchain platform for decentralized applications (DApps).

Module 1.2: Unveiling Cardano's Architecture

The Cardano Vision

Cardano aspires to be a secure, scalable, and sustainable blockchain platform that can support a wide range of decentralized applications (dApps). Unlike some other blockchains that prioritize one aspect over the others, Cardano strives for a balanced approach.

Scalability: Cardano is designed to handle a large number of transactions per second (TPS) without compromising security. This is crucial for widespread adoption.

Security: Maintaining a secure network that is resistant to attacks is paramount for any blockchain platform. Cardano uses a proof-of-stake consensus mechanism to achieve this.

Sustainability: The energy consumption of proof-of-work blockchains like Bitcoin has raised environmental concerns. Cardano's proof-of-stake approach is significantly more energy-efficient.

A Layered Approach

Cardano's innovative architecture separates the blockchain into two distinct layers:

Cardano Settlement Layer (CSL): This layer is responsible for recording transactions on the ledger and maintaining the overall state of the network. It ensures the security and integrity of the Cardano blockchain.

Cardano Computation Layer (CCL): This layer is responsible for running smart contracts and processing computations. It provides flexibility for developers to build dApps on Cardano.

The separation of these layers offers several advantages:

Improved Scalability: The CCL can be scaled independently of the CSL, allowing for more efficient transaction processing.

Enhanced Security: A security breach in the CCL wouldn't compromise the CSL, protecting the core network.

Greater Flexibility: Developers have more freedom to experiment and innovate on the CCL without affecting the stability of the CSL.

Getting to Know Ouroboros

Ouroboros is Cardano's unique proof-of-stake (PoS) consensus mechanism. Unlike proof-of-work (PoW) which relies on miners solving complex puzzles to validate transactions, Ouroboros uses a staking mechanism to secure the network. Here's a quick rundown of Ouroboros :

Security Guarantees: Ouroboros is demonstrably secure, meaning its security properties can be mathematically proven. This is a significant advantage compared to many other PoS protocols.

Energy Efficiency: Ouroboros consumes significantly less energy than PoW blockchains. This makes Cardano a more sustainable blockchain platform.

Impact on Transaction Processing: Ouroboros can potentially achieve higher transaction throughput compared to PoW because it eliminates the computational overhead of mining.

Smart Contracts on Cardano

Smart contracts are self-executing contracts that reside on a blockchain. They can be used to automate various agreements and processes. Cardano supports smart contracts written in Plutus (and another variety of well-known programming languages), a powerful and secure language designed specifically for blockchain development.

Here are some benefits of using Cardano for smart contract development :

Security: Cardano's layered architecture and Ouroboros consensus mechanism contribute to a secure environment for smart contracts.

Scalability: The layered architecture allows for efficient smart contract execution.

Flexibility: Plutus is a versatile language that enables developers to create complex smart contracts.

Wrap up

In this first day, you already have a solid foundation on the fundamentals of blockchain and a clear understanding of the architecture of Cardano. This knowledge will be crucial as we explore smart contract development in the coming days.

Day 2: Smart Contract Fundamentals

Module 2.1: Demystifying Smart Contracts

What are Smart Contracts?

Smart contracts are self-executing contracts with the terms of the agreement directly written into code. They are automatically executed and enforced when predetermined conditions are met, without the need for intermediaries like lawyers or arbitrators. Smart contracts are typically built on blockchain technology, such as Ethereum, and enable secure and transparent transactions without the need for trust between parties. They have the potential to revolutionize industries by automating and improving the efficiency of contractual agreements.

Anatomy of a Smart Contract

Here are some components that make up the anatomy of a smart **contract:**

1. **Terms and Conditions:** The rules, obligations, and conditions of the contract are encoded in the smart contract.
2. **Codebase:** Smart contracts are written in programming languages supported by the blockchain platform. Common languages include Solidity for Ethereum, Chaincode for Hyperledger Fabric, Michelson for Tezos and Plutus for Cardano.
3. **Transaction Logic:** The logic that controls how the contract executes and under what conditions payments or actions are triggered.
4. **Addresses:** Smart contracts have an address on the blockchain network, which is used to interact with and deploy the contract.

5. **State Variables:** These are variables that store data within the smart contract, such as balances, ownership information, and other relevant information.
6. **Functions:** These are pieces of code within the smart contract that govern how the contract can be interacted with. Functions can be called by users to perform specific actions.
7. **Events:** Events can be emitted during the execution of the smart contract to notify external applications or users about specific state changes.
8. **Modifiers:** Modifiers are used to change the behavior of functions in a declarative way, enforcing conditions before the function is executed.
9. **Inheritance:** Smart contracts can inherit properties and functions from other contracts, enabling code reuse and modularity.
10. **Encryption** and Security: Smart contracts often utilize encryption techniques to secure sensitive data and ensure the integrity and confidentiality of the contract.

Benefits of Smart Contracts:

In the Smart Contract use here are some key benefits to consider:

- **Increased security and trust:** Because smart contracts are stored on a blockchain like Cardano, they are tamper-proof and transparent. This reduces the risk of fraud and errors.
- **Reduced transaction costs:** Smart contracts can automate many manual tasks involved in traditional contracts, streamlining the process and potentially lowering costs.
- **Faster execution:** Smart contracts can execute automatically when predetermined conditions are met, expediting the completion of agreements.

- **Improved accuracy:** By automating tasks, smart contracts can minimize the potential for human error in contract execution.

Module 2.2: Limitations and the Future of Smart Contracts (1 hour)

Limitations of Smart Contracts:

Smart contracts have many advantages, such as increased security, automation of processes, cost savings, and transparency. However, they also have some limitations that may need to be addressed for their widespread adoption and success in the future. Some of the limitations of smart contracts include:

1. **Complexity:** Smart contracts can be complex to create and require a good understanding of programming languages and blockchain technology. This can be a barrier for individuals and organizations without the necessary technical expertise.
2. **Immutable Code:** Once a smart contract is deployed on a blockchain, it is immutable and cannot be changed. This means that any bugs or errors in the code can have serious consequences and may be difficult to rectify. This lack of flexibility may be a limitation for certain use cases.
3. **Security Concerns:** While smart contracts are designed to be secure, they are not immune to vulnerabilities and attacks. There have been instances of smart contract hacks and security breaches in the past, highlighting the need for robust security measures and audits.
4. **Oracles:** Smart contracts rely on external data sources known as oracles to trigger and execute certain conditions. The accuracy and reliability of these oracles are crucial for the overall functionality

and integrity of smart contracts. However, there are concerns about the trustworthiness of oracles and potential manipulation of data.

5. **Regulatory Challenges:** Smart contracts operate in a decentralized and often cross-border environment, which can pose regulatory challenges for governments and authorities. Ensuring compliance with existing laws and regulations, especially in areas such as data privacy and consumer protection, may be a hurdle for the widespread adoption of smart contracts.
6. **Scalability:** As blockchain networks grow and more transactions are processed, scalability becomes a concern for smart contracts. Improving scalability without compromising security and decentralization is a key challenge that needs to be addressed for the future of smart contracts.

Despite these limitations, ongoing research and development efforts are focused on addressing these challenges and improving the functionality of smart contracts. As the technology continues to evolve and mature, smart contracts are expected to play a significant role in reshaping various industries and driving innovation in the future.

The Future of Smart Contracts

Smart contracts are undoubtedly set to revolutionize the way transactions are conducted in various industries. By utilizing blockchain technology, smart contracts enable self-executing agreements between parties without the need for intermediaries. This not only reduces transaction costs but also increases transparency and security.

In the future, smart contracts are expected to be more widely adopted, especially in industries like finance, real estate, supply chain management, and even governance. With advances in blockchain technology, smart contracts will become more efficient, scalable, and versatile, allowing for more complex and customized agreements to be created.

Furthermore, as more people and businesses start using smart contracts, we can expect to see the development of standardized templates and libraries that will make it easier for non-experts to create and deploy smart contracts. This will democratize access to this technology and open up new opportunities for innovation and collaboration.

Overall, the future of smart contracts looks bright, and they are poised to transform the way we engage in agreements and transactions, making them more secure, efficient, and accessible to a wider audience.

Day 3: Choosing Your Tools

Module 3.1: Unveiling Cardano's Development Landscape

In this module, we will explore the tools available for building smart contracts on the Cardano blockchain. Here are some of the key tools you can consider when developing smart contracts on Cardano:

Plutus is the smart contract platform for Cardano, allowing developers to write secure and robust smart contracts in the functional programming language Haskell. Plutus is designed to provide safety and security guarantees for smart contracts on the blockchain.

Marlowe is a domain-specific language for financial contracts on the Cardano blockchain. It enables non-programmers to create financial contracts using a simple and intuitive interface, making it easier for individuals and businesses to engage in smart contract transactions on Cardano.

The Plutus Playground is an online environment where developers can write, compile, and test Plutus smart contracts. It provides a sandboxed environment to experiment with smart contract code before deploying it on the Cardano blockchain.

The Cardano Node is the core component of the Cardano blockchain network, responsible for validating transactions, executing smart contracts, and maintaining the integrity of the blockchain. Developers can interact with the Cardano Node to deploy and manage smart contracts on the network.

By leveraging these tools, developers can create powerful and innovative smart contracts on the Cardano blockchain, contributing to the growth and adoption of decentralized applications on the network.

Module 3.2: Mastering Marlowe & Plutus Playground

Marlowe and Plutus are both tools related to smart contract development on the Cardano blockchain, but they serve different purposes:

Marlowe

- **Focuses on ease of use:** Designed to be more approachable, even for those with little coding experience.
- **Visual development:** Offers a drag-and-drop interface using Blockly, allowing you to build contracts without writing complex code.
- **Good for starting out:** If you're new to smart contract development, Marlowe can be a great way to get started by visualizing the logic behind your contracts.

Plutus

- **More powerful and flexible:** Provides a general-purpose smart contract language based on Haskell, a powerful functional programming language.
- **Requires coding knowledge:** Writing smart contracts in Plutus requires proficiency in Haskell.
- **Suitable for complex applications:** Offers greater control and flexibility for building intricate smart contracts.

Marlowe Playground

This web-based tool allows you to write, test, and simulate Marlowe contracts before deploying them to the Cardano blockchain.

Day 4: Building Your First Smart Contract (Marlowe)

Module 4.1: Deep Dive into Marlowe Concepts

Marlowe: key concepts to understand for building your smart contract

- **Roles:** Define the participants involved in the contract, such as buyer, seller, or escrow agent.
- **Choices:** Represent the different actions each role can take within the contract, like sending payment or delivering goods.
- **Promises:** Specify the obligations and expectations of each role, ensuring all parties fulfill their part of the agreement.
- **Collateral:** Act as a security deposit held by the contract to incentivize participants to follow the agreed-upon terms.
- **Conditions:** Establish the criteria that need to be met before a specific action can occur within the contract.

Building Blocks for Your Contract

When building your Marlowe contract, consider these elements:

1. **Identify Roles:** Determine the participants interacting with your contract.
2. **Define Choices:** Specify the actions each role can perform at various stages of the contract.
3. **Establish Promises:** Outline the obligations of each role to ensure a successful transaction.
4. **Set Conditions:** Define the criteria that need to be met for each choice to be executed.

5. **Incorporate Collateral (Optional):** Include a security deposit if necessary to enforce adherence to the agreement.

Marlowe Playground Walkthrough

The Marlowe Playground offers a visual interface to experiment with Marlowe concepts. For a general walkthrough the Playground here.

Find other features in the chart below:

N⁰	What?	How?
1.	**Explore the Interface**	Familiarize yourself with the drag-and-drop blocks representing roles, choices, promises, and conditions.
2.	**Build Your Contract**	Start by defining the roles involved in your smart contract.
3.	**Connect Choices and Promises**	Establish the actions each role can take and the corresponding obligations they must fulfill.
4.	**Set Conditions**	Specify the criteria that need to be met before certain actions can be executed.
5.	**Test and Refine**	Run simulations within the Playground to test your contract's logic and identify any potential issues.
6.	**Marlowe Code Generation**	Once satisfied, the Playground can generate the Marlowe code representing your smart contract.

Module 4.1. Ideas for simple smart contracts

Ideas for simple smart contracts suitable for beginners on Cardano, focusing on Marlowe due to its user-friendly approach:

1. Escrow Contract

An escrow contract acts as a trusted third party holding funds or assets until certain conditions are met. This can be a great way to learn about roles, promises, and conditions in Marlowe.

- **Roles:** Buyer, Seller, Escrow Agent (played by the contract)
- **Scenario:** Buyer wants to purchase an item from Seller. Buyer sends funds to the contract (Escrow Agent). Seller sends proof of ownership/delivery of the item to the contract. Upon verification, the contract releases the funds to the Seller.
- **Promises:** Buyer promises to pay. Seller promises to deliver the item.
- **Conditions:** The contract only releases funds to the Seller when it receives proof of delivery from the Seller.

2. Multi-Signature Wallet

This contract simulates a multi-signature wallet where multiple parties (e.g., two family members) need to approve a transaction before funds are released.

- **Roles:** Owner 1, Owner 2 (controlled by different wallets)
- **Scenario:** Both Owner 1 and Owner 2 need to approve a transaction for funds to be released from the contract.
- **Promises:** Both Owners promise to approve the transaction if it's legitimate.

- **Conditions:** The contract only releases funds when it receives approval signatures from both Owners.

Benefits of these examples:

- **Relatively simple logic:** These contracts involve straightforward conditions and promises, making them easier to grasp for beginners.
- **Focus on core concepts:** They allow you to practice defining roles, choices, promises, and conditions, which are fundamental building blocks for any Marlowe contract.

Learning Resources:

- **Marlowe Playground:** https://play.marlowe.iohk.io/ - Experiment with these concepts visually in the Marlowe Playground.

Marlowe Getting Started Tutorial: Search for "Marlowe getting started tutorial" here https://www.youtube.com/watch?v=r71ZZmMzdno.

Remember, these are just starting points. As you gain confidence, you can explore more complex Marlowe contracts or consider learning Plutus for even greater flexibility.

Day 5: Testing and Debugging

Module 5.1: The Importance of Rigorous Testing (1 hour)

Test is just as crucial for smart contracts as it is for traditional software. Here's why rigorous testing is vital for smart contracts:

Why Rigorous Testing Matters for Smart Contracts

- **Immutability:** Smart contracts are deployed on the blockchain and are immutable. This means any errors or vulnerabilities in the code become permanent and cannot be fixed after deployment. Rigorous testing helps identify and eliminate issues before they reach the live blockchain.
- **Financial Impact:** Smart contracts often deal with real-world assets or tokens. Bugs can lead to significant financial losses for users if funds are sent incorrectly or security vulnerabilities are exploited.
- **Security Concerns:** Smart contracts are a prime target for hackers due to the potential for financial gain. Thorough testing helps uncover security weaknesses and prevent malicious attacks.

Types of Smart Contract Testing

- **Unit Testing:** Test individual functions or components of the smart contract to ensure they behave as expected under various conditions. This helps isolate and fix logic errors within the contract.
- **Integration Testing:** Verify how the smart contract interacts with other blockchain components like wallets or oracles. This ensures seamless data flow and proper communication between different parts of the system.
- **Fuzz Testing:** Involves feeding the smart contract with unexpected or random data inputs to identify potential edge cases or vulnerabilities that might not be caught with traditional testing methods.

- **Security Audits:** Engage professional security experts to conduct in-depth analyses of the smart contract code to identify security flaws and potential attack vectors.

Effective Testing Strategies for Smart Contracts

- **Start Early:** Integrate testing throughout the development process, right from the design phase. This allows for early detection and correction of errors.
- **Clear Requirements:** Clearly define the expected behavior and functionalities of the smart contract. This serves as a baseline for test case creation.
- **Code Coverage:** Aim for high code coverage with your test suite to ensure most parts of the contract logic are exercised and tested.
- **Formal Verification (Optional):** For highly critical smart contracts, consider formal verification techniques using mathematical methods to prove the contract's correctness under specific conditions.

Benefits of Rigorous Testing

- **Increased Security:** Reduces the risk of vulnerabilities and exploits that could lead to financial losses.
- Enhanced Reliability: Ensures the smart contract functions as intended under various scenarios.
- **Improved User Confidence:** Provides users with greater confidence in the security and reliability of the smart contract.

The implementation of a rigorous testing strategy that incorporates different testing methods can develop more secure, reliable, and trustworthy smart contracts.

Module 5.2: Leveraging Plutus Playground for Testing

Leveraging Plutus Playground for Testing

The Plutus Playground is a valuable tool for testing and debugging Plutus smart contracts in a safe, simulated environment before deployment on the Cardano blockchain. Here's how you can leverage it for effective testing:

Benefits of Using Plutus Playground for Testing:

- **Safe Environment:** The Playground provides a sandboxed environment where you can experiment with your smart contract code without risk of affecting the real blockchain. This allows you to test various scenarios and identify issues before deployment.
- **Interactive Interface:** The Playground offers a user-friendly interface for writing, simulating, and debugging your Plutus code. You can step through the contract execution, inspect intermediate states, and identify the root cause of errors.
- **Built-in Test Cases:** The Playground allows you to define unit tests for your smart contract functions. This helps ensure each function behaves as expected under different input conditions.
- **Rapid Feedback:** You can quickly iterate on your code, make changes, and retest them within the Playground, accelerating the development and debugging process.

Testing Strategies with Plutus Playground:

1. Unit Testing:

- Define unit tests for individual functions within your smart contract.
- Use the Playground's test case functionality to specify inputs and expected outputs for each function.

- Run the tests to verify if the functions produce the correct results under various scenarios.

2. Scenario Testing:

- Craft test cases that simulate real-world use cases of your smart contract.
- This can involve setting up transactions, manipulating wallet UTXOs, and observing the contract's behavior.
- The Playground allows you to interact with simulated wallets and tokens to create these test scenarios.

3. Error Handling:

- Write test cases that trigger potential error conditions within your contract.
- Verify if the contract handles errors gracefully, such as by reverting invalid transactions or providing informative error messages.

4. Edge Case Testing:

o Design test cases with unexpected or extreme input values to identify potential edge cases or vulnerabilities in your contract logic.

o The Playground allows for flexibility in crafting these test cases to explore the boundaries of your contract's behavior.

Additional Tips:

Start Simple: Begin with writing unit tests for individual functions before moving on to more complex scenario testing.

- **Test Early and Often:** Integrate testing throughout the development process, not just at the end.
- **Clear Documentation:** Document your test cases and expected outcomes for future reference and collaboration.
- **Consider Formal Verification (Optional):** For highly critical smart contracts, explore formal verification tools alongside the Playground for added assurance.

Leveraging the functionalities of the Plutus Playground, you can establish a robust testing strategy for your Plutus smart contracts. This helps ensure they are secure, reliable, and function as intended before interacting with real users and assets on the Cardano blockchain.

Resources

Marlowe

- Marlowe Website: https://marlowe.iohk.io/

- Marlowe Playground: https://play.marlowe.iohk.io/

Plutus

- Plutus Playground: https://play.marlowe.iohk.io/ (While it's the Marlowe Playground, information about testing Plutus contracts can be found there)

Plutus Pioneer Program:

https://iohk.io/en/blog/posts/2022/11/18/what-iog-has-delivered-for-cardano/ (This program offers a lesson on the Marlowe Playground which can be applied to Plutus testing as well)

- Cardano Documentation: https://github.com/cardano-foundation (This is the official Cardano documentation website which includes resources on Plutus)

General Smart Contract Testing

Community Resources

- Cardano Forum: https://forum.cardano.org/

- Cardano Developers Blog: https://iohk.io/en/blog/posts/page-1/