VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

FOUNDATIONS OF BLOCKCHAIN TECHNOLOGY

BCSE324L

Dr. Malathi D.

FOUNDATIONS OF BLOCKCHAIN TECHNOLOGY

Course Objectives

- To understand building blocks of Blockchain.
- To significance of Distributed Ledger Technology and Smart Contract.
- To exploit applications of Blockchain in real world scenarios and their impacts.

Expected Outcomes

- Understand Blockchain ecosystem and its services in real world sceneries
- Apply and Analyze the requirement of Distributed Ledger Technology and Smart Contract
- Design and Demonstrate end-to-end decentralized applications
- Acquaint the protocol and assess their computational requirements

Smart Contracts

Smart Contracts

- **Anatomy of a Smart Contracts**
- **Life Cycle**
- **Usage Patterns**
- **DLT-based smart contracts**
- **Use Cases**
    - **Healthcare Industry**
    - **Property Transfer**

Smart Contracts

Smart Contract

- A smart contract is a **self-executing computer program** stored on a blockchain that **automatically carries out the terms of an agreement once predetermined conditions are met**.
- Smart contracts were introduced in the 1990s by cryptographer Nick Szabo, who described them as "**a set of promises, specified in digital form, including protocols within which the parties perform on these promises.**“
- This technology allows transactions to be conducted **without the need for intermediaries**, ensuring that **all participants can verify the outcome immediately, without delays**.
- In simpler terms, smart contracts are **digital agreements in which the terms between a buyer and seller are written directly into code**. These contracts operate on blockchain networks, making the transactions **traceable, transparent, and irreversible**.
- This automation **enhances the accuracy, speed, and efficiency** of transactions.
- Smart contracts are finding their way into **numerous applications** from **automating financial transactions to managing supply chains**, as more people recognize their potential.

Smart Contracts

Traditional Contracts vs. Smart Contracts

- **Traditional contracts require third-party oversight for enforcement and dispute resolution.**
    - **Example**: a tenant must pay rent monthly, and disputes may require legal intervention.
- **Smart contracts eliminate intermediaries by automatically enforcing terms when conditions are met.**
    - **Example**: vending machine - insert a dollar, and you get a Coke.
    - The machine **follows built-in rules**, **triggering the agreed action** (dispensing a Coke) once an **obligation** (inserting a dollar) **is fulfilled**.
    - This automation ensures **agreements are executed precisely and without third-party involvement**.

Smart Contracts

Why Are Smart Contracts Useful?

- Elimination of Middlemen: **Reduce costs** by removing intermediaries like brokers or lawyers.
- Accuracy, Speed and Efficiency: **Automated execution** ensures fast, **disruption-free transactions**.
- Trust and Transparency: The **system is secure and transparent**, eliminating the need to trust other parties. Since smart contracts are free from third-party interference, there is **no risk of tampering** for personal gain. **Encrypted transaction logs are shared** among participants, **ensuring transparency**.
- Immutable and Secure: Blockchain technology ensures **data integrity and security**, as transactions are **encrypted and unchangeable**.
- Security: Blockchain's **encryption** makes transaction records **highly secure and resistant to hacking**. **Modifying a single record would require altering the entire chain**, which is nearly impossible.
- Cost Savings: By eliminating intermediaries, smart contracts **reduce the time delays and fees** associated with traditional contracts.

Smart Contracts

How Do Smart Contracts Work?

- A smart contract is a binding agreement between two parties, just like any traditional contract. However, it operates using code that leverages blockchain technology, making the process more efficient, transparent, and secure.
- **Smart contracts are executed through simple "if/when…then…" statements that are written into the blockchain.**

Steps Involved in Smart Contract Operation:

1. **Agreement**

- The parties involved need to **agree on the terms and conditions** of the transaction or exchange.
- They must also **decide how the smart contract will work**, including the **specific criteria that need to be met for the contract to be fulfilled**.

2. Contract Creation

- The contract is then created by **coding the agreed-upon terms into a programming language**.
- Done by the parties **themselves or through a smart contract service provider**.
- At this stage, it is **crucial to thoroughly verify the contract's security**.

Smart Contracts

3. Deployment

- **Once finalized**, the smart contract is **uploaded to the blockchain**, similar to how regular crypto transactions are processed.
- Code is embedded in the blockchain, **once verified**, the **contract becomes active and cannot be altered**.

4. Monitoring Conditions

- The smart contract **continuously monitors the blockchain** or another reliable source **for specific conditions or triggers**.
- These triggers can be **anything that can be verified digitally**, such as a payment being made or a specific date being reached.

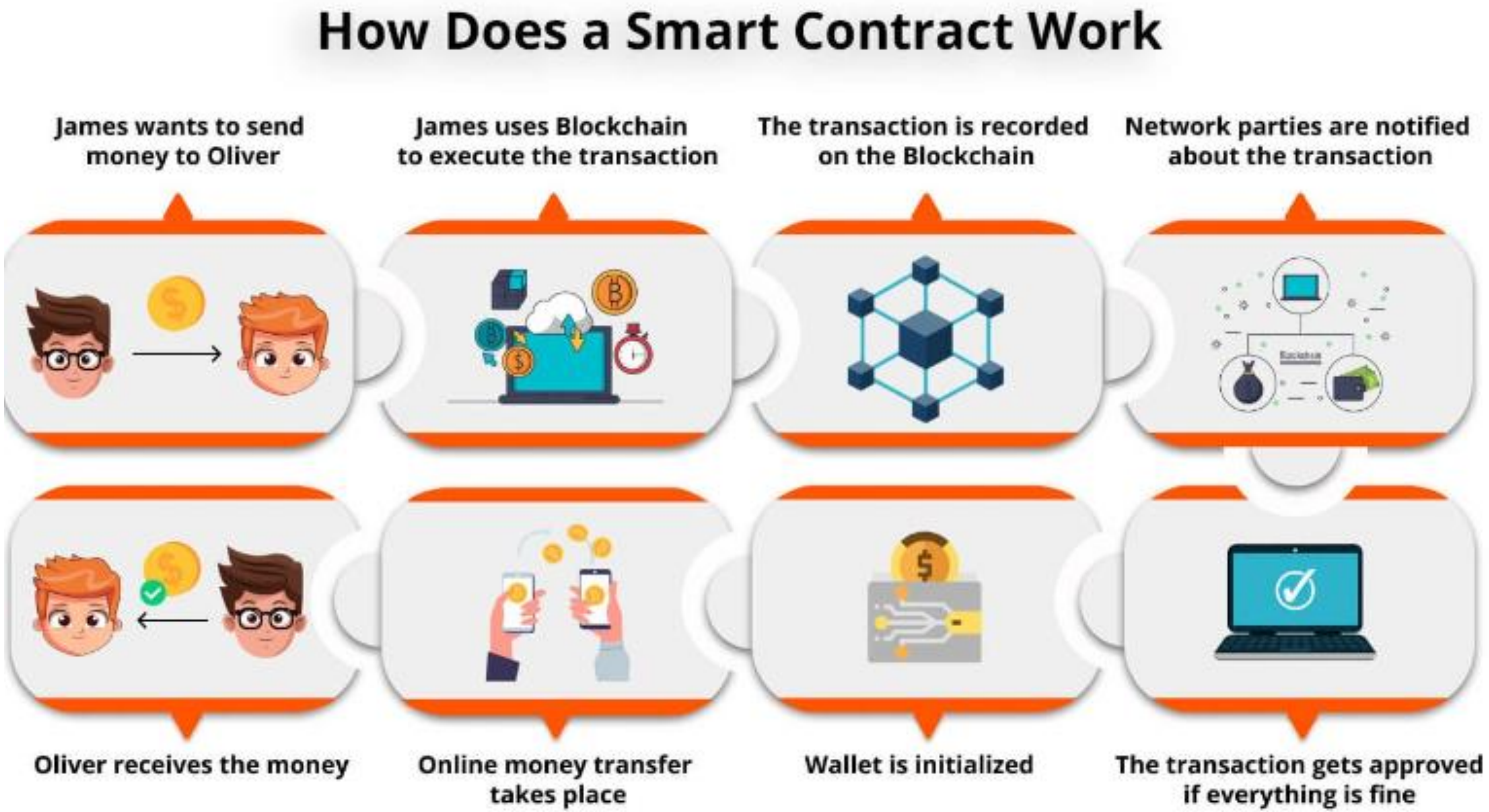
5. Execution

- When the conditions are met, the smart contract **automatically executes the agreed-upon actions**, following the "**if/when…then…**" logic.
- This could **involve transferring funds to a seller or updating ownership records** for a buyer.

6. Recording

- After execution, the **results are immediately recorded on the blockchain**.
- The system **verifies the actions taken**, **logs the transaction**, and **stores the completed contract** on the blockchain, **ensuring it is always accessible**.

Smart Contracts



How Does a Smart Contract Work

James wants to send money to Oliver

James uses Blockchain to execute the transaction

The transaction is recorded on the Blockchain

Network parties are notified about the transaction

The transaction gets approved if everything is fine

Wallet is initialized

Online money transfer takes place

Oliver receives the money

Smart Contracts

Types of Smart Contracts: Smart contracts can be categorized into three main types:

1. **Smart Legal Contracts**

- These are **legally binding contracts that follow the structure of traditional legal agreements**: "If this happens, then that will happen.“
- Since they are stored on the blockchain and cannot be changed, they offer **more transparency**.
- These contracts can be **executed automatically**, for example, by **making a payment once a specific condition is met. Failure to comply can result in legal consequences**.

2. Decentralized Autonomous Organizations (DAOs)

- DAOs are **groups governed by a smart contract** that **grants voting rights to its members**.
- These organizations **operate without a central leader**, with rules and funding decisions controlled by the code embedded in the blockchain.
- **Example: VitaDAO**, where a community uses smart contracts to collaborate on scientific research.

3. Application Logic Contracts (ALCs)

- ALCs involve **application-based code** that interacts with other blockchain contracts or devices, such as those in the Internet of Things (IoT).
- Unlike other smart contracts, these are **not agreements between people or organizations but between machines and other contracts**.

Smart Contracts

Benefits of Smart Contracts

1. **Single Source of Truth**

- All parties have **access to the same data at all times**, **reducing the risk of exploitation**.
- This **enhances trust and security**, as **contract-related information is consistently available** and cannot be altered.

2. Reduction in Human Effort

- Smart contracts **eliminate the need for third-party verification or human oversight**, allowing for faster processes and cost savings.

3. Prevention of Errors

- Smart contracts ensure that **every detail is recorded accurately**, reducing the risk of errors that could lead to serious issues later on.

4. Zero-Trust by Default

- Smart contracts **operate on a decentralized network**, **eliminating the need to trust other parties** in the transaction. This aligns with zero-trust security principles, **ensuring a fair and transparent process**.

5. Built-in Backup

- All essential **transactional details are stored** on the blockchain, making it **easy to retrieve data if needed**.

Smart Contracts

Challenges of Smart Contracts

1. **Rigidity and Inconsistent Support**

- Once a smart contract is deployed, it is **almost impossible to modify**, making **fixing errors difficult and costly**. Additionally, while smart contracts may comply with some national laws, ensuring global adherence can be challenging.

2. Difficulty in Capturing Unquantifiable Data

- Smart contracts **work well with quantifiable data**, but not all industries use measurable metrics, such as creative work that requires subjective evaluation.

3. Conflict with GDPR

- The General Data Protection Regulation (GDPR) **allows individuals to request the deletion of their digital data**. However, if that data is part of a smart contract, it cannot be removed or altered.

4. Skills Shortage

- Developing smart contracts **requires specialized knowledge of programming languages like Solidity**, which are not widely known. This expertise is hard to find.

5. Scalability Issues

- The scalability of blockchain networks can be a problem. For example, while **Visa (Payment processing networks) can process** about **24,000 transactions per second**, **Ethereum**, the largest blockchain for smart contracts, can **only handle about 30 transactions per second**.

Smart Contracts

Top Smart Contract Tools

1. **BoringSolidity:** A **library for developing Solidity smart contracts**, created by ConsenSys Diligence, that aims to **improve code quality and reduce vulnerabilities**.
2. **Chainlink:** A leading Oracle solution that **brings real-world data into smart contracts**, providing **reliable and tamper-proof information** across multiple blockchains.
3. **Ethcode:** A **Visual Studio Code extension** for Ethereum smart contract development, offering a **user-friendly environment for coding, testing, and debugging**.
4. **Octopus:** A **tool for in-depth analysis** of smart contract source code, **helping developers identify and fix errors before deployment**.
5. **OpenZeppelin:** An **open-source framework** that provides a **library of secure smart contracts**, along with audit and **authentication services**.
6. **Solidity:** The **primary language** for creating smart contracts on Ethereum, with a **syntax similar to Python, C++, and JavaScript**. Solidity applications can also run on other blockchains like Polygon and Avalanche.

Smart Contracts

Best Practices for Using Smart Contracts

- When using smart contracts, consider the following best practices:

1. Prioritize Simplicity:

- **Keep the contract logic simple to reduce the chance of errors**.
- Using **pre-written code** can help ensure smooth execution.

2. Update Contracts Regularly

- Regular updates help **identify and fix vulnerabilities**, improving security and user experience.

3. Lock Compiler Versions

- **Explicitly specify compiler versions** in the contract code **to ensure consistency** across different environments.

4. Conduct Rigorous Testing

- Test the contract on a test network **before deploying it on the mainnet** to catch any issues early.

5. Work with Experts on Independent Audits

- Since smart contracts are on a decentralized network, their **code must be secure**.
- Independent audits can **help identify potential vulnerabilities**.

Smart Contracts

Real-World Smart Contract Examples

- **Smart contracts are revolutionizing industries** like Finance, Real Estate, Healthcare, Insurance, and Elections. They **automate tasks, calculate payments, and execute terms** instantly upon meeting conditions, **saving time and enabling multi-party consensus validation**.

1. Trade Finance

- Smart contracts enhance trade finance by automating processes and reducing the need for intermediaries. They ensure transparency, security, and efficiency in transactions.
  - **Automated Payments**: **Trigger payments upon meeting specific conditions**, reducing delays.
  - **Document Verification**: **Automatically verify** and authenticate trade documents.
  - **Shipment Tracking**: **Track goods** through every stage of the supply chain.
  - **Dispute Resolution**: **Resolve disputes efficiently** through predefined rules.
- These capabilities **mitigate risks, lower costs, and improve the speed** of trade finance operations.

Smart Contracts

2. Real Estate

- Smart contracts can **record property ownership and optimize transaction speed** by reducing the need for lawyers or brokers.
  - **Property Transactions**: **Execute sales automatically when conditions like payment confirmation are met**, reducing intermediaries.
  - **Lease Agreements**: **Automate rental agreements and payments**, ensuring timely transactions and reducing disputes.
  - **Escrow Services**: Securely **hold funds until all terms are fulfilled**, **releasing them automatically upon completion.**
  - **Title Management**: Record and transfer titles securely on the blockchain, enhancing transparency and reducing fraud.
- Smart contracts provide a **transparent and cost-effective alternative** to traditional property management and **streamline processes** in the mortgage industry.

Smart Contracts

3. Healthcare

- Smart contracts can **securely store patient data** on a blockchain, **accessible only with the patient's private key**. This ensures that **medical providers have the information they need while keeping it secure**.
  - **Clinical Trials**: Automate trial protocols, ensuring data integrity, streamlining patient consent, and automating payments to participants.
  - **Billing and Insurance Claims**: Automate the submission and processing of claims, reducing administrative costs and improving efficiency.
- Smart contracts protect patient data, enhance the trial process, and streamline billing in healthcare.

4. Elections

- Blockchain voting systems could be the future of elections, **making voting safer and more accessible**.
  - **Voter Identity Validation**: **Prevent multiple votes** with smart contracts that validate voters' identities.
  - **Voter Registration and Vote Counting**: Streamline voter registration, verify identities, and count votes in real-time, leading to **faster and more reliable results**.
- Blockchain technology could **reduce election costs by up to 90%**, making the process more economical and accessible.

Smart Contracts

5. Insurance

- Smart contracts strengthen claim processing by **automating error checks** and policy management, **reducing fraud, and shortening processing times**.
  - **Policy Management**: Automate the issuance and renewal of policies, ensuring compliance and reducing administrative tasks.
  - **Fraud Detection**: Smart contracts detect and prevent fraud more efficiently.
- Implementing blockchain technology in insurance could save the industry up to $10 billion annually by improving efficiency and reducing fraud.

6. Legal Contracts

- Smart contracts are revolutionizing traditional contracts by automatically executing transactions when specific conditions are met, eliminating intermediaries and reducing legal fees.

7. Fan Engagement and Rewards

- In the Web3 economy, smart contracts create new monetization opportunities for creators.
- For example, artists can issue NFTs, ensuring they receive royalties for their work through automated payments.

Smart Contracts

8. Music Rights and Revenue

- Smart contracts can revolutionize the music industry by automating and streamlining royalty payments and rights management.
- This ensures artists get paid instantly and fairly, fostering a more direct relationship between artists and their audience.

9. Retail and Small Business Operations

- Smart contracts enhance administrative efficiency by automating payment processes, digitizing payroll administration, and improving supply chain visibility.
- Platforms like XMoney enable seamless cryptocurrency payments for retailers, reducing transaction fees and attracting tech-savvy customers.

10. Digital Identities

- Smart contracts securely manage digital identities, enhancing security, privacy, and efficiency across multiple platforms.
- The blockchain identity management market is expected to grow significantly, highlighting the increasing adoption of this technology.

Anatomy of a Smart Contracts

What is a Smart Contract?

- It is a **self-executing agreement with the terms and conditions** directly written into code.
- Smart contracts operate on a blockchain, which makes them **transparent, easy to trace, and irreversible**.
- This means once a smart contract is deployed, it **can't be changed**, ensuring the terms are automatically executed as soon as all conditions are met.

Key Components of a Smart Contract

- Smart contracts are made up of several essential elements:
  - **Participants**: These are the **entities interacting with the contract**. Participants could be **individuals, computer systems**, or even other smart contracts.
  - **State**: This represents the **current status or condition of the contract**. As participants take actions, the **contract's state changes** to reflect new information.
  - **Functions**: Functions define the **operations the contract can perform**. Participants trigger these functions, and they can **alter the state of the contract**.
  - **Rules**: These are the **conditions that govern how the contract operates**. Rules are embedded in the contract's code and **must be satisfied for specific functions to execute**.

Anatomy of a Smart Contracts

Structure of a Smart Contract

- Although smart contracts can be designed for various purposes, most of them follow a similar basic structure:
  - **Preamble**: This part **contains introductory details** like the contract's name, version, and sometimes a description of its purpose.
  - **State Variables**: These **variables store the contract's current information**. For example, in a contract for a sale, state variables might include details like the **buyer, seller, price, and the current status** of the item.
  - **Functions**: This section **includes the actions the contract can perform**. Examples of functions in a sales contract might be actions like **initiating a sale, confirming payment, or delivering the item** to the buyer.
  - **Modifiers**: These are conditions that must be **met before certain functions can be executed**. For example, a sale function might only execute if the **item status is set as 'for sale.'**
  - **Events**: Events are **actions that log updates to the contract's state**. They are recorded on the blockchain, creating a transparent and traceable record of all actions taken under the contract.

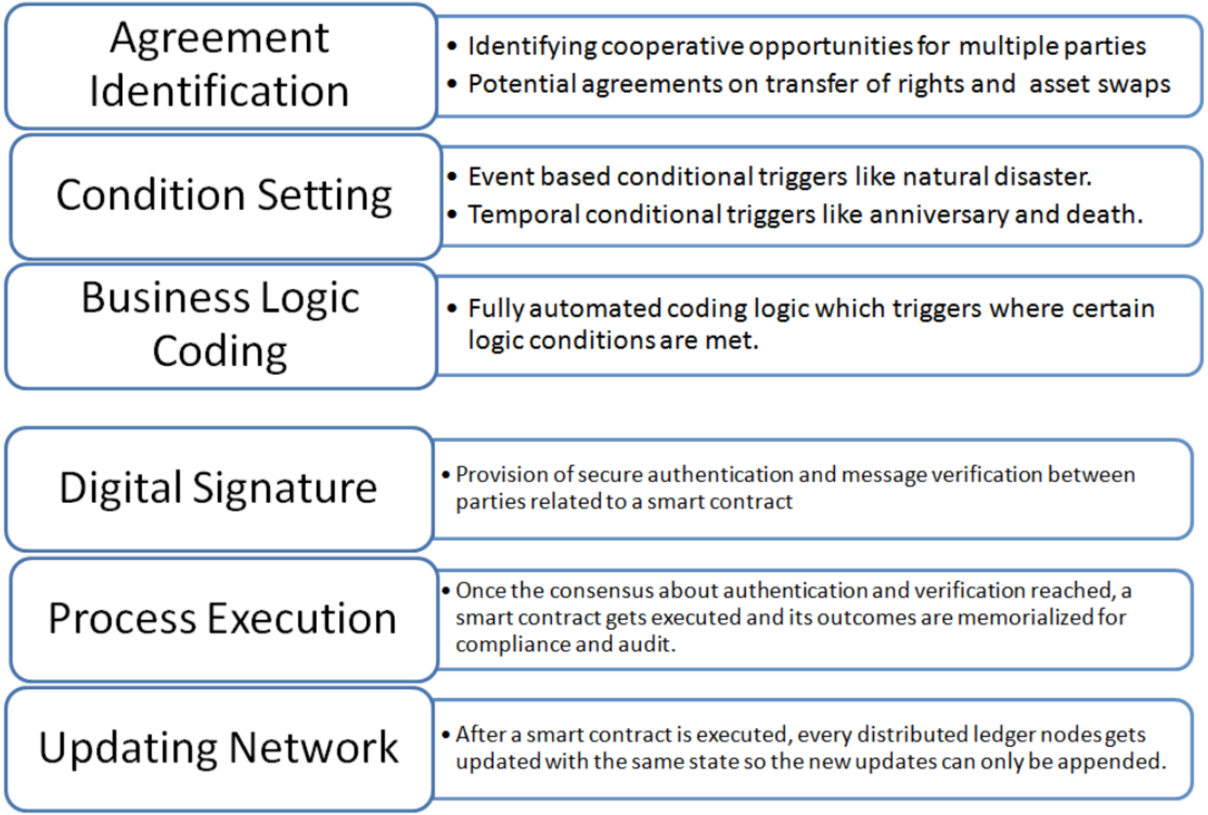
Anatomy of a Smart Contracts

Example:

- **Participants**: The **buyer and seller** are the key participants.
- **State Variables**: Variables such as **buyer, seller, price, and item status** (e.g., ‘for sale’ or ‘sold’) represent the current state of the contract.
- **Functions**: Key functions might include **‘initiateSale’** (to start the sale), **‘confirmPayment’** (to confirm the buyer’s payment), and **‘deliverItem’** (to finalize the delivery).
- **Rules**: The contract might have rules such as `**confirmPayment**` only being executable if the item is still marked as **‘for sale,’** or `**deliverItem**` being executable only after the item has been marked as **‘sold.’**
- **Events**: Actions like `**SaleInitiated**`, `**PaymentConfirmed**`, **and** `**ItemDelivered**` are logged as events. These events **update the state** of the contract (such as changing the item status from **‘for sale’ to ‘sold’**).

Anatomy of a Smart Contracts

Six-point anatomy to formalize a smart contract design



Agreement Identification

- Identifying cooperative opportunities for multiple parties
- Potential agreements on transfer of rights and asset swaps

Condition Setting

- Event based conditional triggers like natural disaster.
- Temporal conditional triggers like anniversary and death.

Business Logic Coding

- Fully automated coding logic which triggers where certain logic conditions are met.

Digital Signature

- Provision of secure authentication and message verification between parties related to a smart contract

Process Execution

- Once the consensus about authentication and verification reached, a smart contract gets executed and its outcomes are memorialized for compliance and audit.

Updating Network

- After a smart contract is executed, every distributed ledger nodes gets updated with the same state so the new updates can only be appended.

Life Cycle of a Smart Contract

- Smart contracts are programs stored on a blockchain that **execute automatically when predetermined conditions are met**, making the process **faster, more secure, and efficient**.
- Their primary purpose is to **enforce an agreement without the need for intermediaries**.
- Smart contracts are commonly written in high-level programming languages like **Solidity and Vyper** and have broad applications across industries such as **finance, gaming, healthcare, and real estate**.
- **Example**: You're buying a house, but you want to avoid the hassle of lawyers, stacks of paperwork, and waiting weeks for things to be finalized.

Smart Contract Life Cycle

- The life cycle of a Smart Contract on the blockchain **involves four key stages**, each **transforming the contract from an idea to a fully executed agreement**.
  - Create
  - Freeze
  - Execute
  - Finalize

Life Cycle of a Smart Contract

1. Create: Laying the Foundation

- This phase involves **drafting and negotiating the terms of the smart contract**, similar to traditional contract negotiation.
- **Both parties must agree on the contract's content & goals**, which can be done either online or offline.
- O**nce the terms are finalized, they are converted into code** that will be stored & executed on a blockchain.
- **Key tasks in the creation phase include**
  - **Negotiation**: Multiple parties agree on the terms.
  - **Design & implementation**: The agreed-upon terms are turned into a smart contract.
  - **Validation**: The contract is verified for accuracy and correctness before being deployed.
- **All participants must have digital wallets** on the blockchain where the contract will be hosted.
- **Example**: Musician and a Music streaming platform come together to **negotiate a royalty agreement**.
  - They decide that the **musician will receive a percentage of the revenue** generated by the streams of their songs.
  - Once they agree on the terms, those details are **converted into a coded Smart Contract**.
  - This phase ensures that **both sides understand the agreement and are ready to proceed**.

Life Cycle of a Smart Contract

2. Freeze: Securing the Agreement

- Smart Contract is now **deployed on the blockchain**, and it becomes publicly visible on a digital ledger.
- **Miners or nodes** across the blockchain network **validate the contract and its transactions**.
- A small fee, often referred to as a "**gas fee**," **is paid to miners** to ensure the **system isn't overwhelmed with unnecessary smart contracts**.
- During this phase, the **digital assets of the participants are "frozen"**—locked in their respective digital wallets—until the smart contract's conditions are fulfilled.
- The blockchain **nodes act as a governing body**, verifying whether the preconditions for contract execution have been met.
- **Key tasks in the freeze phase include**
  - **Storage**: The smart contract is stored on the blockchain and made visible to all.
  - **Asset freezing**: Digital assets from both parties are locked in their wallets to ensure they are only transferred once the contract conditions are met.
- **Example:**
  - During this phase, the **digital assets** (**money from the platform and the royalty rights for the musician**) are "frozen."

Life Cycle of a Smart Contract

3. Execute: The Agreement Comes to Life

- Once the **preconditions for the smart contract are met**, the execution phase begins.
- Blockchain nodes, or "miners," **evaluate the contract and verify its integrity**.
- The smart contract's **execution engine reads the contract code**, and if the conditions are satisfied, the **contract is automatically executed**.
- For example, if the contract involves a financial transaction, **once payment is confirmed**, the contract will automatically trigger the **release of the agreed-upon assets or goods**.
- The consensus mechanism—a process where the **majority of nodes agree—ensures that the transaction is valid** and in accordance with the contract's terms.
- **Key tasks in the execution phase include**
  - **Condition evaluation**: The blockchain checks whether the contract's conditions are met.
  - **Automatic execution**: Once conditions are met, the smart contract is executed automatically.
- **Example**:
  - Once the **streaming platform generating revenue from the musician's songs**—the contract enters the Execute phase. The Smart Contract's engine **automatically calculates the musician's share of the revenue and sends the payment** directly to their digital wallet.
  - No human intervention is needed—the process is automatic and transparent.

Life Cycle of a Smart Contract

4. Finalize: Wrapping Things Up

- After the contract is executed, the **new states of the involved parties are updated**.
- For example, if assets were transferred, the **blockchain updates the records** to reflect this.
- The **assets that were frozen during the "freeze" phase are now unfrozen** and allocated to the receiving party. The **transaction is recorded** on the blockchain, **ensuring transparency and immutability**.
- **Key tasks in the finalize phase include**
  - **State update**: The contract's result is added to the blockchain, and the participants' new states are recorded.
  - **Asset unfreezing**: The digital assets are released to the receiving party, completing the transaction.
- **Example**:
  - The **musician's wallet is updated** with the payment, and the **new transaction is recorded** on the blockchain for transparency. The **frozen assets are now unfrozen**, and both parties can access their funds.
  - The **Smart Contract is closed**, and the assets are unfrozen.
  - The contract's **final state is stored permanently** on the blockchain.

Life Cycle of a Smart Contract

Bonus: Ending the Contract

- Both the musician and the streaming platform **have the option to end the royalty service** **by calling functions like cancelSubscription or endSubscription**.
- This **stops any future royalty payments**.
- When the **contract is canceled**, it **enters the Finalize stage**, **updating the contract's state** to reflect the end of the agreement.
- **What happens here?**
  - The contract can be **ended either by the musician or the streaming platform**.
  - The **royalty payments stop**, and the **Smart Contract is closed**.

- And the life of a Smart Contract, transforming from a simple idea to a fully functioning agreement that flies through the blockchain, automating processes and making life easier for everyone.
- Whether it's paying royalties to musicians or buying a house, **Smart Contracts make complex transactions simple and efficient**.

Text Book and Reference Books

Text Book

- Dhillon, V., Metcalf, D., and Hooper, M, Blockchain enabled applications, 2017, 1st Edition, CA: Apress, Berkeley.

Reference Books

- Diedrich, H., Ethereum: Blockchains, digital assets, smart contracts, decentralized autonomous organizations, 2016, 1st Edition, Wildfire publishing, Sydney.
- Wattenhofer, R. P, Distributed Ledger Technology: The Science of the Blockchain (Inverted Forest Publishing), 2017, 2nd Edition, Createspace Independent Pub, Scotts Valley, California, US.