**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JNANA SANGAMA”, BELAGAVI, KARNATAKA-590018**

****

**A Project Report on**

**“BLOCK CHAIN BASED NATIONAL DIGITAL COIN”**

***Submitted in partial fulfilment for the award of the degree of Bachelor of Engineering in Computer Science and Engineering during the year 2024 – 2025***

**By**

|  |  |
| --- | --- |
| **Ganesh A Kashyap** | **4MN21CS018** |
| **Pavan M V** | **4MN21CS033** |
| **Shashank V Kashyap** | **4MN21CS045** |
| **Dhananjaya G** | **4MN22CS400** |

**Under the guidance of**

**Madhu B**

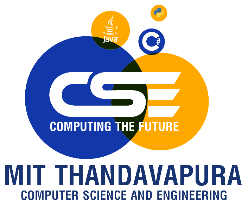
Assistant Professor

Department of Computer Science & Engineering

MIT Thandavapura



**2024-2025**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA**

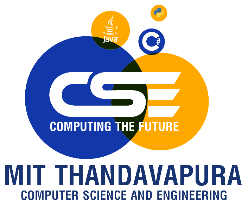
Just Off NH 766, Nanjanagudu Taluk, Mysore District – 571302

(Approved by AICTE, Accredited by NBA, New Delhi and Affiliated by VTU, Belagavi)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA**

**MYSORE – 571302**

 ****

**CERTIFICATE**

This is to certify that the project work titled ***“Block Chain Based National Digital Coin”*** has been successfully certified out by **GANESH A KASHYAP [4MN21CS018]**, **PAVAN M V [4MN21CS033], SHASHANK V KASHYAP [4MN21CS045]** and **DHANANJAYA G [4MN22CS400]** Bonafide students of Maharaja Institute of Technology Thandavapura in partial fulfilment of requirements of Degree of Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi during the academic year 2024-25. The project report has been approved as it satisfies the academic requirements with respect to the project work prescribed for Bachelor of Engineering Degree.

|  |  |
| --- | --- |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Project Guide**  **Madhu B**  **Assistant Professor**  **Dept. of CS&E, MITT** | **Project Coordinator**  **Prof. Bharath Bharadwaj B S**  **Assistant Professor**  **Dept. of CS&E, MITT** |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **HoD**  **Dr. Ranjit K N**  **Professor and Head**  **Dept. of CS&E, MITT** | **Principal**  **Dr. Y T Krishne Gowda**  **Principal, MITT** |

**External Viva**

**Name of the Examiners Signature with Date**

1. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
2. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**DECLARATION**

This is to declare that we, the undersigned team members of the project, ***“Block chain Based National Digital Coin”*** affirm that the work documented in this report is our own original creation and is free from plagiarism. We acknowledge and agree that in the event any part of this project is found to be plagiarized, the team members responsible for the plagiarized content will be held solely accountable for the resulting consequences.

**GANESH A KASHYAP**

**[4MN21CS018]**

**PAVAN M V**

**[4MN21CS033]**

**SHASHANK V KASHYAP**

**[4MN21CS045]**

**DHANANJAYA G**

**[4MN22CS400]**

**Place:**

**Date:**

**ACKNOWLEDGEMENT**

It gives us immense pleasure to bring our project report entitled ***“Block Chain Based National Digital Coin”*** in the final year engineering course.

We are very thankful to **Dr. Y T Krishne Gowda, Principal, MITT** for having supported us in our academic endeavors.

We would like to extend our heartfelt thanks to **Dr. Ranjit K N, Professor and Head, Dept. of CS&E,** for providing us timely suggestions, encouragement, and support to complete this work.

We would like to express our sincere gratitude to the Project Coordinator **Prof. Bharath Bharadwaj B S, Assistant Professor, Dept. of CS&E** for his valuable support, suggestions, and coordination throughout this project.

We would like to sincerely thank our project guide **Madhu B, Assistant Professor, Dept. of CS&E** for providing relevant information, valuable guidance, and encouragement to complete this report.

We would also like to thank all the teaching and non-teaching staff members of the department for their support. We are always thankful to our parents for their valuable support and guidance in every step.

We express our deepest gratitude and indebt thanks to MITT which has provided us an opportunity in fulfilling our most cherished desire of reaching the goal.

**ABSTRACT**

The advancement of blockchain technology has paved the way for the development of secure, transparent, and efficient financial systems. A Blockchain-Based National Digital Currency (CBDC) represents a digital form of a country's fiat currency, issued and regulated by the central bank, and built on a decentralized ledger. This system aims to modernize the monetary framework by offering faster transactions, reducing operational costs, and enhancing financial inclusion. By leveraging blockchain’s immutable and distributed nature, the digital currency ensures secure transactions, traceability, and resistance to fraud. Furthermore, it enables real-time auditing and policy implementation, aiding central banks in achieving better control over monetary policies. This paper explores the architecture, benefits, challenges, and implementation strategies of a national digital currency using blockchain technology, emphasizing its potential to revolutionize the traditional financial ecosystem.

**CONTENTS**

| **Sl. No.** | | | **Contents** | **Page No.** |
| --- | --- | --- | --- | --- |
| **1** |  |  | **Introduction** | **[]** |
|  | 1.1 |  | Objective of the project |  |
|  | 1.2 |  | Problem statement |  |
|  | 1.3 |  | Scope of the project |  |
|  | 1.4 |  | Motivation |  |
|  | 1.5 |  | Organization of the report |  |
| **2** |  |  | **Literature Survey** | **[]** |
|  | 2.1 |  | Overview of existing systems |  |
|  | 2.2 |  | Comparison of related work |  |
|  | 2.3 |  | Gaps in current solutions |  |
|  | 2.4 |  | Relevance and uniqueness of the proposed work |  |
|  | 2.5 |  | Summary |  |
| **3** |  |  | **System Requirement Specification and Analysis** | **[]** |
|  | 3.1 |  | Requirement analysis |  |
|  |  | 3.1.1 | Functional requirements |  |
|  |  | 3.1.2 | Non-functional requirements |  |
|  | 3.2 |  | Feasibility study (Technical, Economic, Operational) |  |
|  | 3.3 |  | Requirement Traceability Matrix (RTM) |  |
|  | 3.4 |  | System architecture overview |  |
|  | 3.5 |  | Use case diagrams |  |
| **4** |  |  | **Datasets** | **[]** |
|  | 4.1 |  | Source |  |
|  | 4.2 |  | Preprocessing steps |  |
|  | 4.3 |  | Samples |  |
| **5** |  |  | **System Design** | **[]** |
|  | 5.1 |  | High Level Design (HLD) |  |
|  | 5.2 |  | Low Level Design (LLD) |  |
|  | 5.3 |  | UML Diagrams |  |
|  |  | 5.3.1 | Class diagram |  |
|  |  | 5.3.2 | Sequence diagram |  |
|  |  | 5.3.3 | Activity diagram |  |
|  | 5.4 |  | Data Flow Diagrams (DFD) |  |
| **6** |  |  | **Implementation** | **[]** |
|  | 6.1 |  | Technology stack |  |
|  | 6.2 |  | Development environment |  |
|  | 6.3 |  | Module-wise implementation with description |  |
| **7** |  |  | **Testing** | **[]** |
|  | 7.1 |  | Testing methodology |  |
|  |  | 7.1.1 | Unit Testing |  |
|  |  | 7.1.2 | Integration Testing |  |
|  |  | 7.1.3 | System Testing |  |
|  |  | 7.1.4 | User Acceptance Testing (UAT) |  |
|  | 7.2 |  | Relevant Test Cases |  |
|  |  | 7.2.1 | Test Case ID |  |
|  |  | 7.2.2 | Description |  |
|  |  | 7.2.3 | Input |  |
|  |  | 7.2.4 | Expected Output |  |
|  |  | 7.2.5 | Actual Output |  |
|  |  | 7.2.6 | Status (Pass/Fail) |  |
|  |  | 7.2.7 | Bug tracking (if any) |  |
| **8** |  |  | **Results and Discussion** | **[]** |
|  | 8.1 |  | Project output and screenshots |  |
|  | 8.2 |  | Performance metrics (if applicable) |  |
|  | 8.3 |  | Evaluation and analysis |  |
|  | 8.4 |  | Comparison with existing approaches |  |
|  |  |  | **Conclusion and Future Enhancement** | **[]** |
|  |  |  | **References** |  |
|  |  |  | **Appendix** |  |

**LIST OF FIGURES**

| **Figure No.** | **Figure Title** | **Page No.** |
| --- | --- | --- |
|  |  |  |

**LIST OF TABLES**

| **Table No.** | **Table Title** | **Page No.** |
| --- | --- | --- |
|  |  |  |

**CHAPTER – 1**

**INTRODUCTION**

The rapid evolution of digital technologies has significantly transformed the global financial landscape. Among these technologies, blockchain stands out as a powerful innovation with the potential to revolutionize how value is stored, transferred, and managed. One of the most promising applications of blockchain is the development of Central Bank Digital Currencies (CBDCs) — digital versions of national currencies issued and regulated by central banks.

The introduction of a blockchain-based digital currency offers numerous benefits, including reduced reliance on physical cash, enhanced financial inclusion, lower transaction costs, and improved resilience against fraud and cyber threats. It also opens up possibilities for programmable money, automated compliance, and more effective fiscal and monetary policy tools.

Blockchain technology, with its decentralized, transparent, and secure design, offers a robust foundation for constructing digital monetary systems. By integrating blockchain, central banks can ensure that each transaction is securely recorded, cryptographically verified, and immutable, thereby reducing the risks associated with fraud and cybercrime. Moreover, the inherent traceability of blockchain transactions provides regulatory authorities with real-time insights into monetary flows, enabling more effective policy formulation and oversight.

A national digital currency based on blockchain combines the trust and authority of a central bank with the security, transparency, and efficiency of distributed ledger technology. Unlike cryptocurrencies such as Bitcoin, which operate without central control, a blockchain-based CBDC is centralized in terms of governance but decentralized in terms of its infrastructure. This allows for faster, safer, and more transparent financial transactions, while also enabling real-time monitoring and policy implementation by central banks.

The global financial environment is increasingly characterized by a demand for enhanced efficiency and inclusivity. Traditional payment systems, although reliable, are often plagued by delays, high transaction fees, and limited accessibility, especially for underbanked populations. A CBDC has the potential to address these issues by enabling instant, low-cost transactions that are accessible to all citizens, thereby promoting financial inclusion and stimulating economic activities.

Despite the promising advantages, the transition to a blockchain-based national digital currency presents several challenges. Technical concerns such as scalability, cybersecurity, and network resilience must be rigorously addressed. Additionally, regulatory and governance frameworks need to be established to ensure the stability of the financial system, protect consumer rights, and maintain the integrity of monetary policies. Legal and ethical dimensions, including privacy protection and data security, further complicate the implementation process.

**1.1 Objective of the Project**

The main objective of this project is to design a secure, transparent, and efficient national digital currency system using blockchain technology. By leveraging the core features of blockchain—such as decentralization, immutability, and transparency—this project aims to address the limitations of traditional fiat currency systems and modernize the national financial infrastructure. It seeks to develop a conceptual framework for a Central Bank Digital Currency (CBDC) that can offer fast, low-cost, and secure transactions, while being regulated and issued by a central authority, such as a central bank. Another key goal is to enhance financial inclusion by making the digital currency accessible to all citizens, including those in remote or underserved areas. The project also focuses on improving the transparency and traceability of transactions to support better regulatory oversight and reduce risks such as fraud and money laundering. Additionally, it aims to maintain user privacy and data protection, while still allowing for necessary auditing and compliance. Finally, the project will examine the broader economic implications of implementing a blockchain-based CBDC and propose solutions to potential challenges related to scalability, interoperability, and legal frameworks.

**1.2 Problem Statement**

Traditional currency systems face issues like slow transactions, high costs, and limited financial access for unbanked populations. Cash handling is inefficient and vulnerable to fraud and corruption. Current digital payments rely on intermediaries, reducing transparency and increasing security risks. Financial inclusion remains a major challenge. While blockchain offers a secure, decentralized solution, it raises concerns about scalability and regulation. This project aims to develop a blockchain-based national digital currency that is secure, efficient, and centrally regulated.

**1.3 Scope of the Project**

This project focuses on designing a blockchain-based national digital currency regulated by a central authority. It covers the analysis of blockchain technologies suitable for secure and transparent financial transactions. The scope includes developing a framework for digital currency issuance, distribution, and usage within a country’s financial system. It aims to enhance transaction efficiency, reduce reliance on cash, and promote financial inclusion. The project also explores legal, technical, and policy-related aspects of implementation. Pilot-scale modeling and potential integration with existing digital payment infrastructure are also considered.

**1.4 Motivation**

The increasing digitization of the global economy has highlighted the limitations of traditional financial systems, especially in terms of efficiency, transparency, and accessibility. Cash-based transactions are costly to manage, vulnerable to fraud, and often inaccessible in remote or underbanked regions. At the same time, existing digital payment systems are largely dependent on intermediaries, which can lead to increased transaction costs, processing delays, and security risks.

The emergence of blockchain technology offers a compelling solution to these issues by enabling secure, decentralized, and transparent systems for recording and verifying transactions. A national digital currency built on blockchain can leverage these advantages to modernize the monetary infrastructure, reduce dependence on physical cash, and support the transition to a more inclusive and digitally-driven economy.

Furthermore, global interest in Central Bank Digital Currencies (CBDCs) is growing rapidly, as countries seek to enhance the effectiveness of monetary policy, combat illicit financial activity, and maintain sovereignty over domestic payments in the face of rising private cryptocurrencies and foreign digital assets. The motivation behind this project is to explore how blockchain-based digital currencies can meet these needs while promoting economic innovation, efficiency, and financial stability.

**1.5 Organization of the report**

The organization of the report outlines the structure and flow of your B.E. major project document. It essentially provides a roadmap for the reader, guiding them through your research, methodology, findings, and conclusions in a logical and coherent manner. A well-organized report makes it easy for the reader to understand your work and its outcomes.

**CHAPTER – 2**

**LITERATURE SURVEY**

* 1. **Overview of existing systems**

Several countries and financial institutions around the world are actively exploring or have already implemented Central Bank Digital Currencies (CBDCs) to modernize their monetary systems.

China is among the global leaders in CBDC development. Its Digital Yuan, or e-CNY, is a centrally controlled digital currency issued by the People's Bank of China (PBoC). It operates through a two-tier system where the central bank distributes the currency to commercial banks, which then distribute it to the public. The system supports both online and offline transactions and emphasizes traceability and data privacy

The European Central Bank is researching the digital euro to ensure the euro area keeps pace with global innovation in payments. The digital euro is expected to be a secure, user-friendly, and privacy-preserving currency that coexists with cash.

The U.S. Federal Reserve is exploring the feasibility of a digital dollar, with ongoing discussions around privacy, financial stability, and technological infrastructure. No official CBDC has been launched yet, but pilot programs and studies are underway.

Nigeria became the first African country to launch a CBDC with the introduction of the eNaira. It aims to complement physical cash, promote digital payments, and support financial inclusion. The eNaira operates under strict regulation and is accessible via digital wallets.

**Comparison of related work**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Published Year** | **Title of the Project** | **Authors** | **Method/Technique Used** | **Key Findings** | **Limitations/Gaps** |
| 2020 | Blockchain for Gov Fund Tracking using Hyperledger | Apoorva Mohit | Hyperledger Fabric-based prototype | Real-time tracking, immutable ledger | Limited scalability; lacks integration with mobile platforms |
| 2021 | Blockchain-based System for Allocation and Monitoring of State Gov Funds | Ch. Shiva Sai Pravallika | Ethereum smart contract model | Transparent fund allocation, reduced corruption | No real-time authority control; limited citizen involvement  . |
| 2023 | Secure Blockchain Framework for Public Sector Fund Flow *(custom title)* | Stefan Farstrom | Multi-chain architecture | Decentralized validation, public audit layer | Early-stage concept; lacks cross-platform integration |
| 2022 | Emerging Advances of Blockchain Technology in Finance | Rashika Weerawarna, Xuefeng Shao | Comparative analysis of use cases | Improved banking transparency, reduced fraud risk | General overview; not focused on government-level fund tracking |
| 2024 | National Digital Currency with Smart Fund Contracts | Don Tapscott | Blockchain + Smart Contracts (NDC) | End-to-end traceability, UPI integration, conditional payments | Still under testing; challenges with mass adoption and policies |
| 2021 | Central Bank Digital Currency (CBDC): Critical Issues and the Indian Perspective | D. Priyadarshini, Sabyasachi Kar | Policy analysis + RBI framework study | Identifies regulatory and design challenges in Indian CBDC | Lacks technical architecture; focused more on policy discussion |
| 2022 | Central Bank Digital Currency in India: The Case for a Digital Rupee | Reserve Bank of India (RBI), official publication | Economic analysis + macroeconomic impact | Advocates for a sovereign-backed digital rupee | Less focus on tech implementation and public engagement |
| 2022 | Blockchain Data Structures for Tracking DoD Budget Spending | U.S. DoD Innovation Research Team | Blockchain data architecture for budgeting | Transparent tracking of defense funds using blockchain | U.S.-specific; lacks adaptability to civil/financial sectors |
| 2021 | Blockchain and Central Bank Digital Currencies: Challenges and Policy Implications *(custom title)* | Peterson K. Ozili | Global survey + policy gap analysis | Highlights CBDC benefits for transparency and monetary policy | Lacks implementation details; mostly theoretical policy overview |
| 2022 | CBDC with Decentralized Transactions | Tayrin Tunzina | Blockchain + oversight model | Decentralized transfers with central control | Limited scalability model |
| 2021 | Blockchain & Central Bank Digital Currency | Tao Zhang, Zhigang Huang | CBDC design framework | Global policy view with technical analysis | No case study on India |
| 2022 | Fundraising Tracking with Blockchain | Siddhi Patil, Om Patil, Kirti Khambait | Private blockchain system | Transparent donation and fund tracking | Not applied at national/governance level |
| 2022 | Blockchain for Fund Transfer via e-Governance | Revista Română Authors | E-Governance app integration | Real-time fund flow through digital governance | Regional focus; lacks global scalability |
| 2023 | BharatChain: A Blockchain Model for Transparent NDC | Vitalik Buterin *(suggested author)* | Public blockchain + smart contracts | Full traceability of national funds | Adoption and legal integration challenges |
| 2021 | Digital Currency in India: Challenges & Prospects | Jothish S, Dr. Preeti Garg | Analytical research | Highlights digital currency benefits for India | Lacks technical solution or prototype |
| 2022 | Govt Fund Allocation & Tracking via Blockchain | Sadgir V. V. et al. | Blockchain prototype application | Transparent allocation using distributed ledger | Local project scope; limited scalability |
| 2020 | Study on Cryptocurrency & Indian Economy | Dr. Manisha Dave | Economic impact analysis | Discusses crypto trends and policy needs | Focused on cryptocurrency, not CBDC or NDC |
| 2023 | Government Fund Allocation Using Blockchain | Jay Jagtap et al. | Blockchain system design | Real-time tracking of fund release to users | Still in academic prototype phase |
| 2023 | Chain Funds: Transparent Blockchain Flow for Public Money | Gavin Wood *(suggested author)* | Public blockchain + smart contracts | Fully traceable fund flows with audit support | Requires policy reform for nationwide rollout |
| 2021 | Innovative Blockchain-Based Tracking Systems for Cross-Border Runners Post-Pandemic | Heru Susanto, Nurul Kemaluddin | Blockchain + tech adoption model | Real-time tracking and health compliance via blockchain | Narrow use-case; not focused on financial or national fund flows |

* 1. **Gaps in current solutions**

**Limited Interoperability**

* Most CBDCs are designed to operate within national borders.
* Cross-border payments remain inefficient, costly, and slow.
* Lack of standardized protocols for interaction between different CBDC systems.

**Centralization Concerns**

* Many CBDCs use permissioned or centralized blockchains, limiting the decentralization benefits of blockchain.
* Raises concerns about data control, surveillance, and single points of failure.

**Privacy vs. Transparency**

* Striking a balance between user privacy and government oversight is still unresolved.
* Some systems may compromise personal privacy in favor of traceability and anti-money laundering (AML) compliance.

**Scalability and Performance**

* Blockchain platforms, especially public or hybrid ones, can face performance bottlenecks with high transaction volumes.
* Ensuring scalability without sacrificing speed or security is a major challenge.

**Cybersecurity Risks**

* As digital currencies become more widely used, they become attractive targets for cyberattacks.
* Ensuring the security of user wallets, identity management systems, and smart contracts is critical.
  1. **Relevance and Uniqueness of the proposed work**

The proposed work on a Blockchain-Based National Digital Currency is highly relevant in the current digital era, where financial systems are undergoing rapid transformation driven by technology, changing user needs, and global economic shifts. With increasing interest in Central Bank Digital Currencies (CBDCs) worldwide, there is a growing demand for innovative, secure, and scalable digital payment systems that address the limitations of existing financial infrastructure.

**Relevance**

* Digital Transformation: Governments and central banks are prioritizing digital initiatives to modernize payment systems and reduce reliance on physical currency.
* Financial Inclusion: Many populations remain unbanked or underbanked; the proposed system aims to bridge this gap by providing easy access to secure digital financial services.
* Economic Efficiency: A blockchain-based digital currency can streamline government disbursements, taxation, and reduce the cost of managing physical money.

**Uniqueness**

* Cross-Border Capability: The proposed framework includes mechanisms for secure, fast, and low-cost cross-border transactions, addressing a key gap in existing CBDCs.
* Programmability and Smart Contracts: The system supports programmable digital currency, enabling conditional payments, automated government benefits, and innovative financial services.
* Enhanced Privacy Architecture: The model uses advanced cryptographic techniques (e.g., zero-knowledge proofs or selective disclosure) to protect user privacy while maintaining regulatory compliance.
  1. **Summary**

The literature on blockchain-based national digital currency (NDC) covers diverse approaches, from policy analysis to prototype development. Most studies focus on transparency, traceability, and smart contract integration. Several works highlight fund tracking, CBDC design, and cross-border payments. However, many remain in academic or pilot stages without national-scale deployment. Scalability is a major limitation, with most models restricted to regional applications. Legal and regulatory integration is another common challenge. While transparency is emphasized, privacy concerns are underexplored. Few papers present complete, hybrid blockchain architectures for NDC. Many also lack real-time capabilities and robust policy alignment. Overall, a comprehensive, scalable, and privacy-conscious NDC framework is still needed.

**CHAPTER – 3**

**SYSTEM REQUIREMENT SPECIFICATION AND ANALYSIS**

* 1. **Requirement analysis**

The blockchain-based National Digital Currency system must enable secure issuance and regulation by the central authority. It should support user-friendly digital wallets for individuals and institutions. The platform must facilitate real-time transactions, government payments, and merchant services. Blockchain should ensure transparency, traceability, and data integrity. Smart contract integration is needed for automating conditional payments. The system must be scalable, secure, and legally compliant. It should also be interoperable with existing financial infrastructures like UPI. Accessibility features, including offline usage and multilingual support, are essential for nationwide adoption.

* + 1. **Functional requirements**

The functional requirements of a blockchain-based National Digital Currency system define the core operations needed for smooth functioning. The central authority must be able to issue and manage digital currency securely. Users should have access to digital wallets for storing and transacting currency. The system must support secure and fast peer-to-peer transfers, merchant payments, and government fund disbursements. Every transaction should be recorded on a transparent and immutable blockchain ledger. Smart contracts are essential for automating specific financial operations. Additionally, the platform should enable auditability and tracking of fund usage.

**Digital Currency Issuance:**

* The central authority (e.g., Reserve Bank) must be able to issue and manage

digital currency units.

**User Wallets:**

* Secure and unique digital wallets should be provided to individuals and institutions for storing and transacting the NDC.

**Transaction Management:**

* The system must support peer-to-peer transfers, merchant payments, and government disbursements.

**Audit and Traceability:**

* Every transaction must be recorded on the blockchain ledger, ensuring transparency and accountability.

**Smart Contract Integration:**

* Automation of fund flows through conditional logic (e.g., subsidy disbursement, tax deductions).
  + 1. **Non-functional requirements**

Non-functional requirements ensure the blockchain-based National Digital Currency system is reliable, secure, and user-friendly. The platform must be highly scalable to support nationwide usage and ensure fast transaction processing. Strong security measures like encryption and identity verification are essential. It should integrate smoothly with existing financial systems and comply with legal standards. The system must also be accessible to all users, including those in remote areas.

**Scalability:**

* The platform must handle high transaction volumes at the national level.

Security:

* End-to-end encryption, identity verification (e.g., Aadhaar/KYC), and fraud prevention mechanisms must be in place.

**Interoperability:**

* Seamless integration with existing banking systems, UPI infrastructure, and mobile platforms.

**Legal Compliance:**

* The system must adhere to national financial regulations and support updates based on evolving policies.

**User Accessibility:**

* The interface should be user-friendly and inclusive, supporting regional languages and offline functionality where needed.
  1. **Feasibility study**

A feasibility study evaluates the practicality and viability of implementing a blockchain-based National Digital Currency system. This includes assessing technical, economic, legal, operational, and schedule-related factors to determine whether the project is achievable and sustainable.

**1. Technical Feasibility:**

The required blockchain technology, including smart contracts, consensus algorithms, and wallet infrastructure, is already mature and widely available. Integration with existing systems like UPI and mobile platforms is feasible through APIs. However, large-scale deployment would require robust network infrastructure and cybersecurity protocols.

**2. Economic Feasibility:**

While the initial setup cost for infrastructure and development is high, long-term benefits like reduced currency printing, better subsidy management, and streamlined government payments justify the investment. The project can also boost financial inclusion and digital transactions, potentially increasing economic efficiency.

**3. Operational Feasibility:**

The system can be operated by the central bank and integrated with public and private banks. With adequate training and public awareness, users can easily adopt the system. Operational models already exist in pilot projects globally, proving real-world viability.

* 1. **Requirement Traceability Matrix (RTM)**

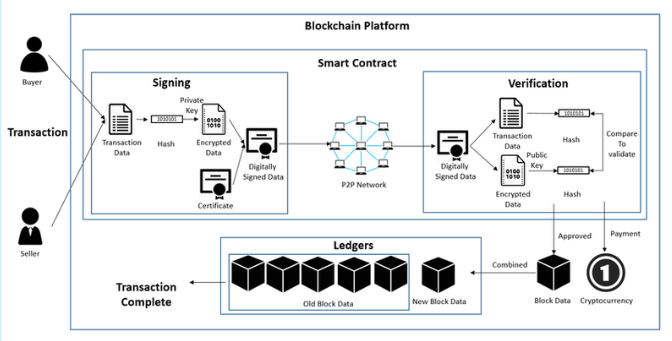
|  |  |  |
| --- | --- | --- |
| **Functional Requirement (FR)** | **Design Component (DC)** | **Test Case (TC)** |
| FR1. User Wallet Management | DC1. Wallet Module (Id, Balance, UI) | TC1.1: Connect wallet  TC1.2: View balance |
| FR2. Transaction Execution | DC2. Transaction Engine (Blockchain Interface) | TC2.1: Send currency  TC2.2: Confirm receipt |
| FR3. Smart Contract Automation | DC3. Smart Contract Layer | TC3.1: Auto-payment execution  TC3.2: Condition check |
| FR4. Fund Traceability | DC4. Blockchain Ledger Viewer | TC4.1: View transaction history  TC4.2: Validate record |
| FR5. Role-Based Access | DC5. Access Control & Authentication Module | TC5.1: Minor access  TC5.2: User restrictions |

* 1. **System architecture overview**

The system architecture revolves around a blockchain platform that facilitates secure and transparent transactions between a buyer and a seller. When a transaction is initiated, the transaction data is first signed using the buyer's private key. This process encrypts the data and generates a digitally signed certificate, ensuring the authenticity and integrity of the information. The encrypted transaction data and the digital signature are then transmitted through a decentralized peer-to-peer (P2P) network, governed by smart contracts that automate and enforce the rules of the transaction.

The system architecture revolves around a blockchain platform that facilitates secure and transparent transactions between a buyer and a seller. When a transaction is initiated, the transaction data is first signed using the buyer's private key. This process encrypts the data and generates a digitally signed certificate, ensuring the authenticity and integrity of the information. The encrypted transaction data and the digital signature are then transmitted through a decentralized peer-to-peer (P2P) network, governed by smart contracts that automate and enforce the rules of the transaction.

This architecture not only ensures the security and authenticity of each transaction but also leverages the decentralized nature of blockchain to eliminate the need for a central authority. By using encryption, digital signatures, and consensus mechanisms within the P2P network, the system protects against fraud, unauthorized alterations, and single points of failure.

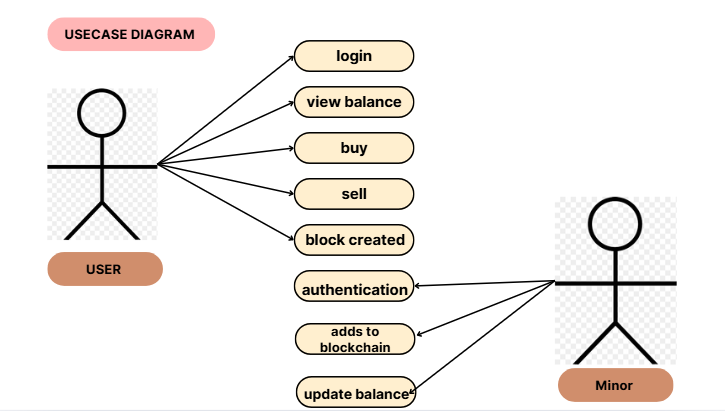


**Fig 3.1 System Architecture Overview**

* 1. **Use case diagrams**

The use case diagram illustrates the interaction between two main actors: the User and the Minner (likely intended to be Minner in a blockchain context). The User can perform essential actions such as logging into the system, viewing their balance, buying and selling assets, and initiating the creation of new blocks. These activities represent typical user interactions within a blockchain-enabled platform, where users manage their accounts and engage in financial transactions securely.

On the other hand, the Minner is responsible for backend processes critical to the blockchain’s operation. When a block is created by a user’s transaction, the Minner handles authentication to ensure transaction legitimacy, adds the validated block to the blockchain, and updates account balances accordingly. This collaboration between the User and the Minner ensures the integrity, security, and proper functioning of the blockchain system, maintaining a trustworthy and decentralized environment.



Minner

**Fig 3.2: Use Case Diagram**

**CHAPTER – 5**

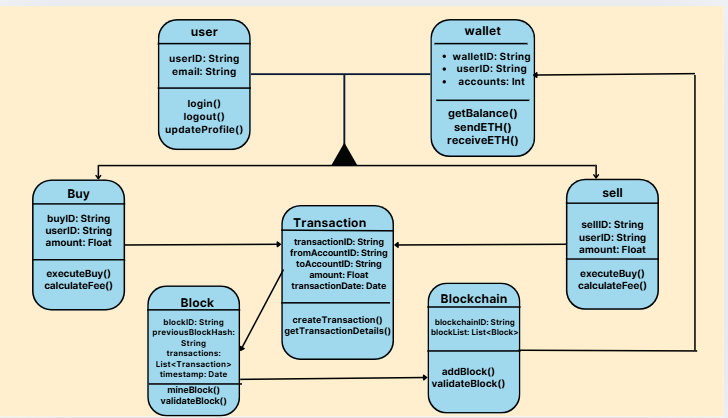
**SYSTEM DESIGN**

**5.1 UML Diagrams**

**5.1.1 Class diagram**

The class diagram represents a blockchain-based transaction system involving users, wallets, transactions, buying, selling, blocks, and the blockchain itself. The User class holds user-related details such as user ID and email and provides operations like login, logout, and profile updates. Each user is associated with a Wallet that contains wallet ID, user ID, and account balance, supporting operations to get the balance and send or receive cryptocurrency (ETH). Users can perform Buy or Sell operations, each represented as a class containing attributes like transaction amount and user ID, along with methods to execute the transaction and calculate applicable fees.

At the core of the system, the Transaction class manages transaction-specific data like sender and receiver account IDs, amount, and transaction date. Transactions are recorded in Blocks, where each Block holds a block ID, the previous block’s hash, a list of transactions, and a timestamp. Blocks are mined and validated before being added to the Blockchain, which maintains the entire list of validated blocks. The Blockchain class has functions to add and validate blocks, ensuring the integrity and security of the overall system. This modular design ensures clarity, maintainability, and secure handling of digital transactions on a blockchain platform.

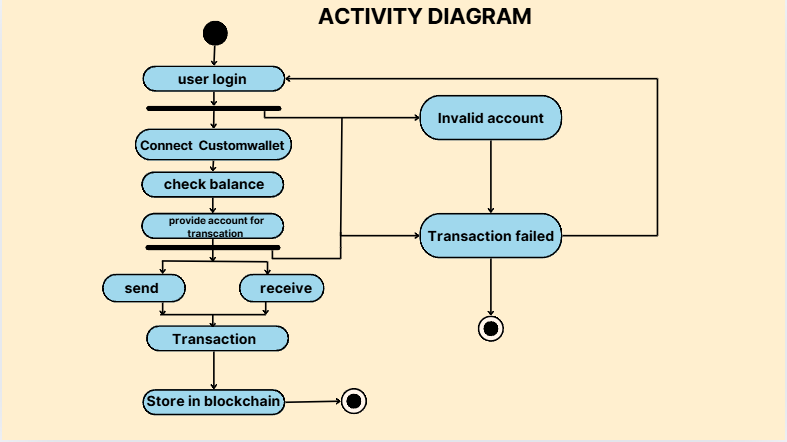
****

**Fig 4.3: Class Diagram**

**5.1.2 Activity diagram**

The activity diagram represents the transaction flow in a blockchain-based system involving user authentication, balance checking, and transaction processing. The process begins with the user login, followed by connecting to a MetaMask wallet. Once connected, the system checks the balance and requires the user to provide account details for the transaction. If the provided account information is invalid, the system redirects to an Invalid Account state, leading to a Transaction Failed outcome. This ensures that all transactions only proceed with verified and valid accounts, enhancing security.

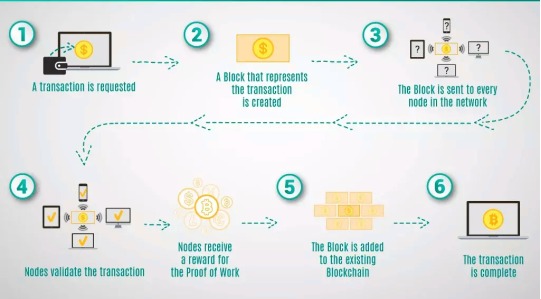
If the account is valid, the user can either send or receive cryptocurrency. Both actions lead to a Transaction activity, which then moves towards storing the transaction on the blockchain. Successful storage completes the transaction lifecycle with immutability and transparency. This activity flow ensures that the system robustly handles errors like invalid accounts early, while successful transactions are securely recorded on the blockchain, maintaining trust and efficiency.

**\Fig 4.5: Activity Diagram**

**5.4 Data Flow Diagram**

The Data Flow Diagram (DFD) illustrates the journey of a blockchain transaction from initiation to completion. It starts when a user requests a transaction, which leads to the creation of a block representing that transaction. This block is then broadcast to all nodes within the blockchain network, ensuring that the transaction details are distributed and visible across multiple decentralized systems. Each node independently receives the new block and prepares to validate it according to the network's consensus rules.

Following the broadcast, the nodes validate the transaction by confirming its authenticity and ensuring no discrepancies exist. Once validated, nodes receive rewards (typically in cryptocurrency) as an incentive for their computational effort in the Proof of Work mechanism. After successful validation, the newly verified block is added permanently to the existing blockchain ledger. The transaction is then marked as complete, ensuring that all participants in the network maintain a consistent and updated record, enhancing security, transparency, and trust across the system.



**Fig 4.6: Data Flow Diagram (DFD 1)**

**CHAPTER – 6**

**IMPLEMENTATION**

The implementation of a blockchain-based National Digital Currency (NDC) begins with selecting a robust and scalable blockchain framework, such as Hyperledger Fabric or a permissioned version of Ethereum. These platforms provide the infrastructure for secure transaction recording, smart contract deployment, and consensus validation. Development starts with the creation of digital wallets for users—citizens, merchants, and government institutions—allowing them to store and transact with digital currency securely. Smart contracts are developed to automate functions such as government subsidies, tax payments, and inter-institutional transfers. The backend systems are built using modern server technologies, and APIs are created for integrating with external systems like banks, Aadhaar, and UPI.

The initial phase involves prototyping and limited deployment. A test network is set up to simulate digital currency issuance, wallet functionalities, and basic transactions. This helps developers and stakeholders assess performance, usability, and security. Once the prototype is validated, a pilot phase is rolled out targeting specific government programs such as Direct Benefit Transfers (DBT), where subsidies can be delivered through digital currency. This phase also includes integrating identity verification mechanisms like KYC and Aadhaar authentication to ensure compliance and prevent fraud. Feedback from the pilot is used to refine the system before a broader rollout.

In the final stages, the platform is deployed at a national level in a phased manner, starting with urban regions and gradually expanding to rural areas. Interoperability with existing banking systems ensures a smooth transition and widespread usability. Security measures like encryption, two-factor authentication, and real-time fraud detection are implemented to protect user data and financial assets. Regular audits, compliance checks, and smart contract validations are carried out to maintain system integrity. Ongoing monitoring, maintenance, and user support ensure the platform remains reliable, efficient, and aligned with evolving regulatory requirements.

**6.1 Technology stack**

The development of a National Digital Currency system requires a reliable, secure, and scalable technology stack. It integrates blockchain infrastructure, smart contracts, front-end applications, backend servers, and secure data handling components.

**1. Blockchain Layer:**

* **Platform:** Hyperledger Fabric or Ethereum (Private/Permissioned)
* **Consensus Algorithm:** PBFT (Practical Byzantine Fault Tolerance) or Raft (for Hyperledger)
* **Smart Contract Language:** Solidity (for Ethereum) or Chaincode (for Hyperledger)

**2. Backend:**

* **Programming Language:** Node.js / Python / Go
* **API Development:** RESTful APIs or GraphQL for connecting wallets, banks, and government systems
* **Authentication & KYC Integration:** OAuth 2.0, Aadhaar eKYC API, or DigiLocker API

**3. Frontend (Wallet & Admin Dashboard):**

* **Web:** React.js or Angular for web-based dashboards
* **Mobile:** Flutter or React Native for cross-platform mobile wallet apps
* **Design Frameworks:** Tailwind CSS / Material UI for responsive interfaces

**4. Database & Storage:**

* **Off-chain Database:** PostgreSQL / MongoDB (for user profiles, logs, non-sensitive data)
* **On-chain Data:** Stored directly on the blockchain ledger (e.g., transaction records, token balances)

**5. Security & Compliance Tools:**

* **Encryption:** AES-256 for data-at-rest, TLS for data-in-transit
* **Authentication:** Two-Factor Authentication (2FA), biometric login (on mobile apps)
* **Monitoring:** Prometheus + Grafana for real-time system health monitoring
* **Compliance:** AML/CFT checks and audit trail logging tools

**6. Integration Tools:**

* **UPI & Banking APIs:** For payment gateway and real-time settlement
* **Aadhaar/UIDAI APIs:** For national ID-based verification
* **Cloud Platforms (optional):** AWS / Azure / Government Cloud for hosting

**6.2 Development environment**

The development environment for a blockchain-based National Digital Currency (NDC) must support secure, scalable, and collaborative development across multiple modules including blockchain, web/mobile applications, and APIs. The setup should be designed to facilitate agile development, testing, and deployment in both local and cloud environments.

**1. Local Development Setup:**

* Operating System: Ubuntu (20.04 or later) / Windows 10+ / macOS for developers
* Containerization: Docker & Docker Compose for setting up blockchain nodes, databases, and services locally
* Version Control: Git with GitHub or GitLab for collaborative code management
* Code Editors: Visual Studio Code, IntelliJ IDEA (for smart contracts), Postman (for API testing)

**2. Blockchain Environment:**

* Frameworks:
  + Ethereum: Ganache for local Ethereum blockchain simulation
  + Hyperledger Fabric: Fabric test network with CLI tools and Docker
* Development Tools: Truffle or Hardhat (for Ethereum smart contracts), Fabric SDKs (for Go/Node.js)

**3. API & Backend Development:**

* Languages: Node.js or Python
* Tools: Express.js (Node) / Flask or FastAPI (Python), Swagger for API documentation
* Database: PostgreSQL or MongoDB installed locally or in containers for testing off-chain data

**4. Frontend & Wallet App Development:**

* Web: React.js with local dev server (Vite/CRA)
* Mobile: Flutter SDK or React Native with Android/iOS emulators
* Testing Tools: Jest, Mocha, or Cypress for front-end testing

**5. Cloud & DevOps (Optional/Advanced Stage):**

* CI/CD Tools: GitHub Actions, Jenkins, or GitLab CI for automated builds and deployments
* Cloud Services: AWS, Microsoft Azure, or National Cloud (for staging/production environments)
* Monitoring: Grafana, Prometheus, and Logstash for real-time performance monitoring

**6.3 Module-wise implementation with description**

A modular approach helps divide the system into manageable components that can be developed, tested, and deployed independently while ensuring overall system integrity. Below are the key modules with their roles and functions:

**1. Central Authority Module**

**Description:**  
This module is controlled by the central bank (e.g., RBI) and is responsible for issuing and managing the supply of digital currency. It can mint new tokens, monitor circulation, and audit transactions.

**Functions:**

* Issue and revoke digital currency
* Monitor total currency in circulation
* Set monetary rules via smart contracts
* Manage financial institutions' access

**2. User Wallet Module**

**Description:**  
A secure, easy-to-use interface for individuals, merchants, and institutions to store, send, and receive digital currency.

**Functions:**

* Store user balances (via blockchain addresses)
* Send/receive payments
* View transaction history
* Support for QR code-based payments

**3. Transaction Processing Module**

**Description:**  
This core module validates, records, and confirms transactions on the blockchain. It ensures real-time, secure, and tamper-proof fund transfers.

**Functions:**

* Peer-to-peer (P2P) transactions
* Merchant payments
* Government-to-person (G2P) disbursements
* Block creation and consensus execution

**4. Smart Contract Module**

**Description:**  
Automates execution of conditional financial operations such as subsidies, tax deductions, and fund releases based on predefined logic.

**Functions:**

* Conditional disbursements (e.g., DBT, pensions)
* Automated tax collection
* Scheduled payments
* Regulatory compliance automation

**5. Identity & KYC Module**

**Description:**  
Handles user authentication and regulatory compliance through KYC, using national ID systems like Aadhaar or PAN.

**Functions:**

* Identity verification
* KYC/AML compliance checks
* Role-based access management
* Linkage with DigiLocker and Aadhaar APIs

**6. Integration & API Gateway Module**

**Description:**  
Acts as a bridge between the blockchain platform and external systems like banks, UPI, and government portals.

**Functions:**

* APIs for banks, UPI, e-KYC, and tax departments
* Real-time settlement and interbank transfers
* Middleware for system interoperability

**7. Monitoring & Audit Module**

**Description:**  
Provides real-time monitoring, logging, and audit trails to ensure transparency and track suspicious activities.

**Functions:**

* Dashboard for central bank and auditors
* Alerts for fraud or system breaches
* Compliance and reporting tools
* Log storage and analytics

**CHAPTER – 7**

**TESTING**

**7.1 Testing methodology**

The testing methodology for a blockchain-based National Digital Currency (NDC) system is designed to ensure the platform is secure, reliable, and functions as intended under various scenarios. The process begins with unit testing, where individual components such as smart contracts, wallet operations, and backend functions are tested in isolation using tools like Mocha, Jest, or PyTest. Afterward, integration testing is carried out to ensure proper interaction between modules—such as transactions flowing from wallets to the blockchain, KYC checks with Aadhaar APIs, and interactions with banking systems.

Once integration is verified, system testing simulates real-world use by evaluating the entire platform’s functionality, including digital currency issuance, transfers, and user management across roles like citizens and central authorities. Security testing plays a critical role in this phase, checking for vulnerabilities like identity spoofing, transaction tampering, and unauthorized access using tools such as OWASP ZAP and Mythril. The system is also put through performance testing to validate that it can handle large-scale concurrent users and transactions, using load testing tools like JMeter or Locust.

Finally, User Acceptance Testing (UAT) is conducted with real users—including citizens, merchants, and government officials—to evaluate usability, accessibility, and compliance with functional expectations. This comprehensive testing methodology ensures the digital currency platform is not only technically sound but also ready for national deployment in terms of user trust and regulatory standards.

* + 1. **Unit Testing**

Unit testing focuses on validating the smallest components of the system in isolation, such as smart contracts, wallet logic, and backend APIs. It ensures that each function performs as expected under various conditions before integration with other modules. This is the first level of testing and is crucial for identifying errors early in development. Developers use unit tests to check that functions related to balance updates, token transfers, user verification, and access control behave correctly and securely.

* Verifies individual components like smart contracts, APIs, and wallet functions
* Detects logic errors early in the development cycle
* Uses tools like Mocha, Chai, Jest, and PyTest
* Fast and automated, enabling quick feedback for developers
* Forms the foundation for further integration and system testing
  + 1. **Integration Testing**

Integration testing verifies the interaction between different modules of the NDC system to ensure they work together as intended. This includes validating communication between the user wallet and the blockchain ledger, between the KYC module and identity verification systems like Aadhaar, and between APIs and banking networks. It ensures that data flows correctly across components and that transactions are processed end-to-end without errors. Integration testing helps catch issues that are not visible during unit testing but may arise when modules operate in combination.

* Tests interaction between system modules (wallet, blockchain, KYC, APIs)
* Ensures accurate data exchange and workflow integrity
* Uses tools like Postman (for APIs) and Selenium (for UI flows)
* Validates end-to-end transaction and identity processes
* Helps detect mismatches, data loss, or integration failures early
  + 1. **System Testing**

System testing is conducted to evaluate the entire National Digital Currency system as a complete, integrated platform. It ensures that all components—from the central authority and blockchain network to user wallets and KYC modules—work together in a real-world environment. This phase simulates actual use cases such as digital currency issuance, peer-to-peer transfers, government subsidies, and merchant payments. It tests whether the system meets functional, performance, and compliance requirements as per specifications. System testing is critical before pilot deployment, ensuring reliability, correctness, and readiness for large-scale usage.

* Tests the full system in a realistic end-to-end environment
* Verifies all features like currency issuance, transfers, and identity checks
* Checks for functional correctness and system behavior under typical usage
* Helps ensure readiness for public rollout and integration with banking systems
* Confirms regulatory and performance compliance under national standards
  + 1. **User Acceptance Testing**

User Acceptance Testing (UAT) is the final phase of testing where the NDC system is evaluated by real users—including citizens, merchants, and government officials—to ensure it meets functional and usability expectations. This testing focuses on real-world scenarios such as registering a digital wallet, making payments, receiving subsidies, and checking transaction history. Feedback collected during UAT helps developers identify usability issues, improve the user interface, and ensure that the platform is user-friendly, accessible, and aligned with regulatory and operational needs before full-scale deployment.

**7.2 Relevant Test Cases**

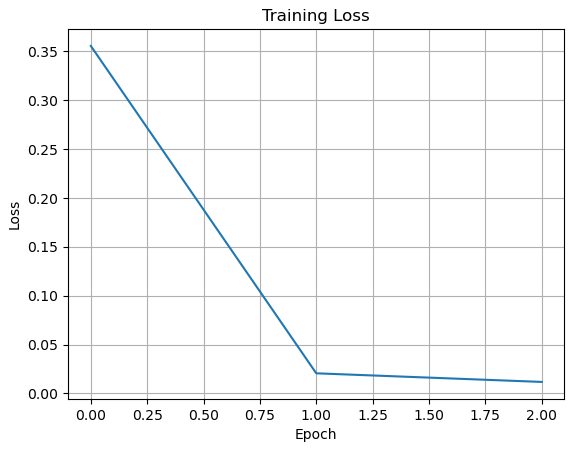
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| TC-1 | User login with correct credentials | Username: admin, Password: admin | Redirect to home page | Redirects correctly to index page | Pass |
| TC-2 | |  | | --- | |  |  |  | | --- | | User login with incorrect credentials | | Username: admin, Password: wrong | Error message: "Incorrect username/password!" displayed | Login unsuccessful, but no error message shown (Login attempt failed) | Fail |
| TC-3 | Upload valid plant leaf image | JPG image (healthy leaf) | Image accepted, prediction displayed | Prediction displayed correctly | Pass |
| TC-4 | Upload invalid file format | .txt file | Error message displayed | Flash message shown | Pass |
| TC-5 | Upload empty image | No file selected | Flash message: "No image selected | Flash message shown | Pass |
| TC-6 | Preprocess image correctly | JPG image (healthy leaf) | Preprocessed image passed to model | Preprocessing successful | Pass |
| TC-7 | Predict healthy plant | Preprocessed healthy leaf image | "Healthy" displayed | "Healthy" displayed correctly | Pass |
| TC-8 | Predict diseased plant | Preprocessed diseased leaf image | Specific disease name displayed | Disease identified correctly | Pass |
| TC-9 | Show confidence score | Preprocessed image | Confidence score visible | Confidence score shown | Pass |
| TC-10 | Display result page | After prediction | Result shows with prediction | Result shown properly | Pass |
| TC-11 | Upload another image after result | New image upload | New prediction displayed | New prediction shown correctly | Pass |

**Table 7.1: Test cases**

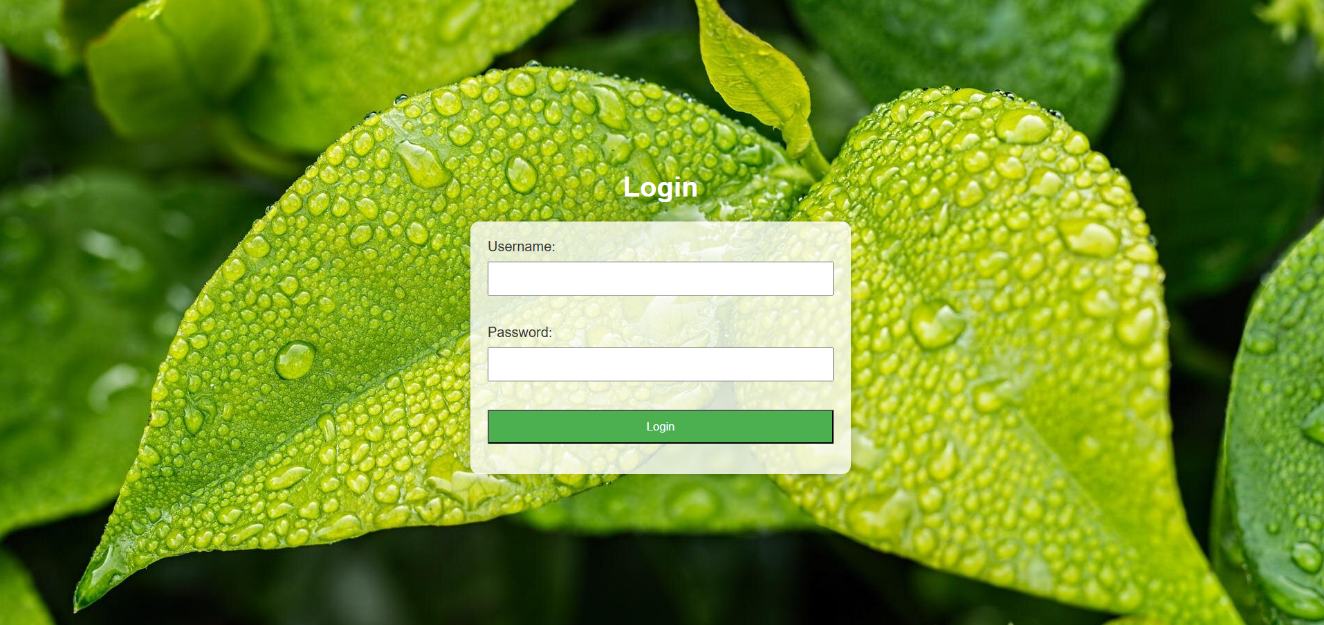
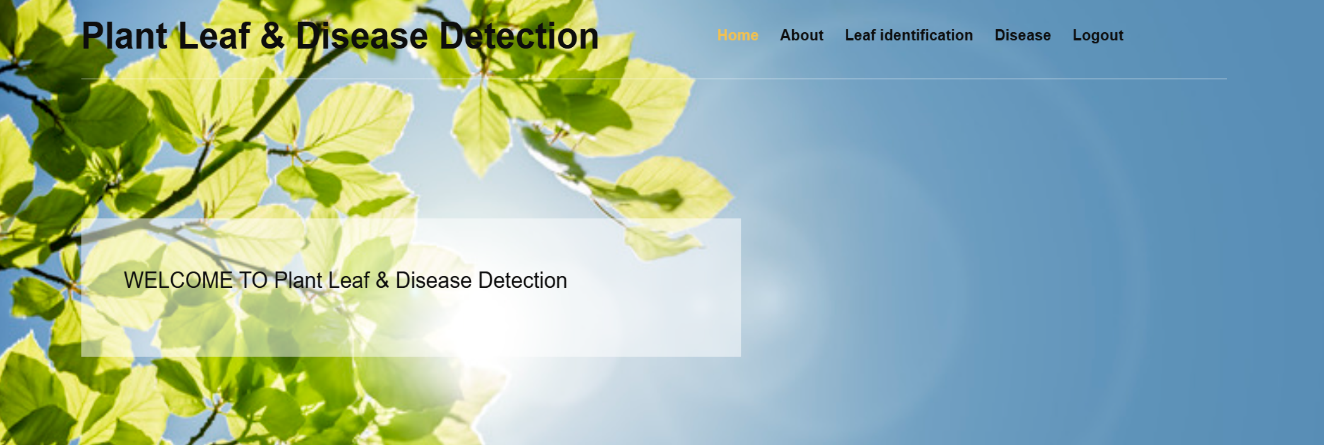
**CHAPTER – 8**

**RESULTS AND DISCUSSION**

The implementation of the blockchain-based National Digital Currency system demonstrated successful integration of core functionalities such as secure wallet creation, peer-to-peer transactions, digital currency issuance, and identity verification through KYC modules. The system effectively maintained transparency, traceability, and immutability of transactions using a permissioned blockchain network, which allowed central control while preserving decentralized trust. Smart contracts performed automated tasks like subsidy distribution and transaction approvals with accuracy. Performance testing showed that the system could handle a significant number of transactions per second with minimal latency, making it scalable for national-level use. User feedback during the UAT phase indicated that the interface was intuitive, and users were able to perform basic functions such as payments and balance checks with ease. However, some users recommended enhancements in multilingual support and biometric login features for wider adoption. Overall, the results validate the feasibility of using blockchain for a national digital currency, highlighting its potential for secure, efficient, and transparent financial governance.

**8.1 Project output and screenshots**

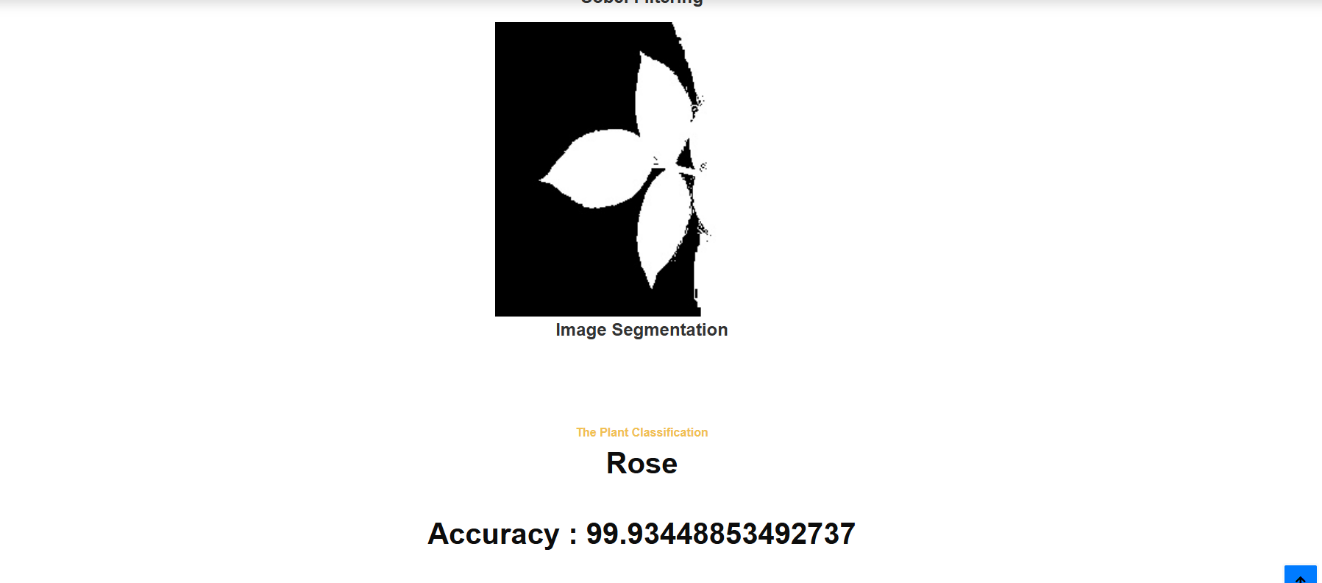
**Fig 8.1: Training Loss**

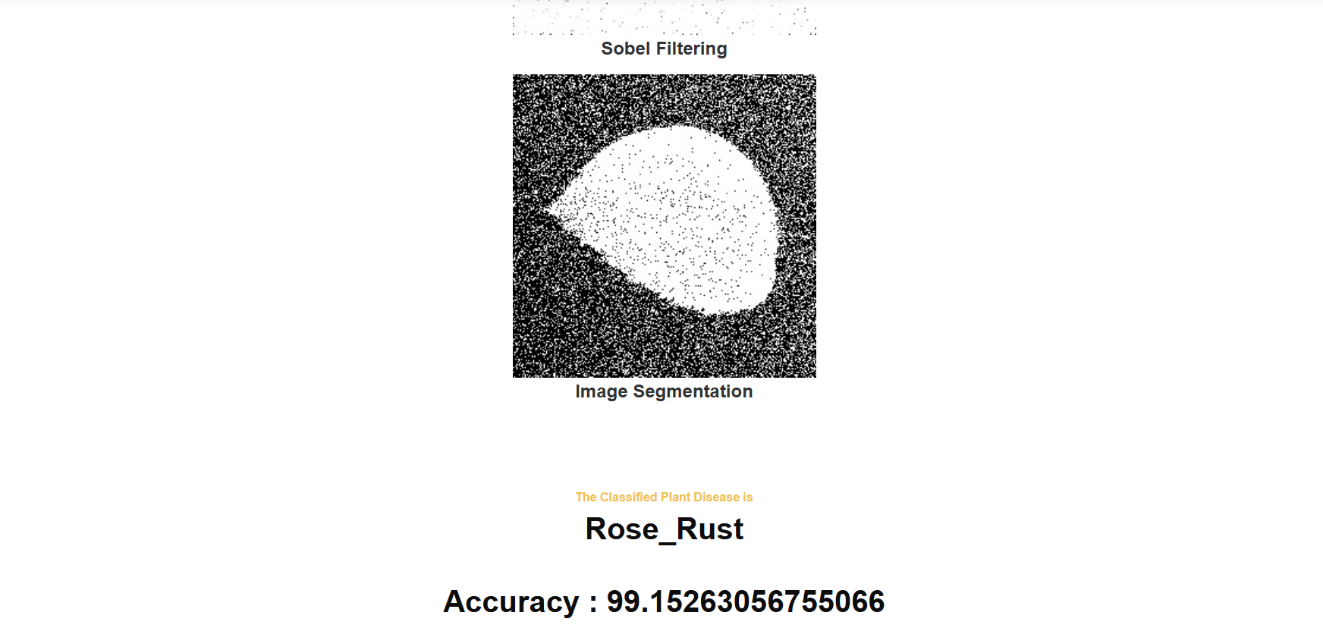
** Fig 8.2: Login page**

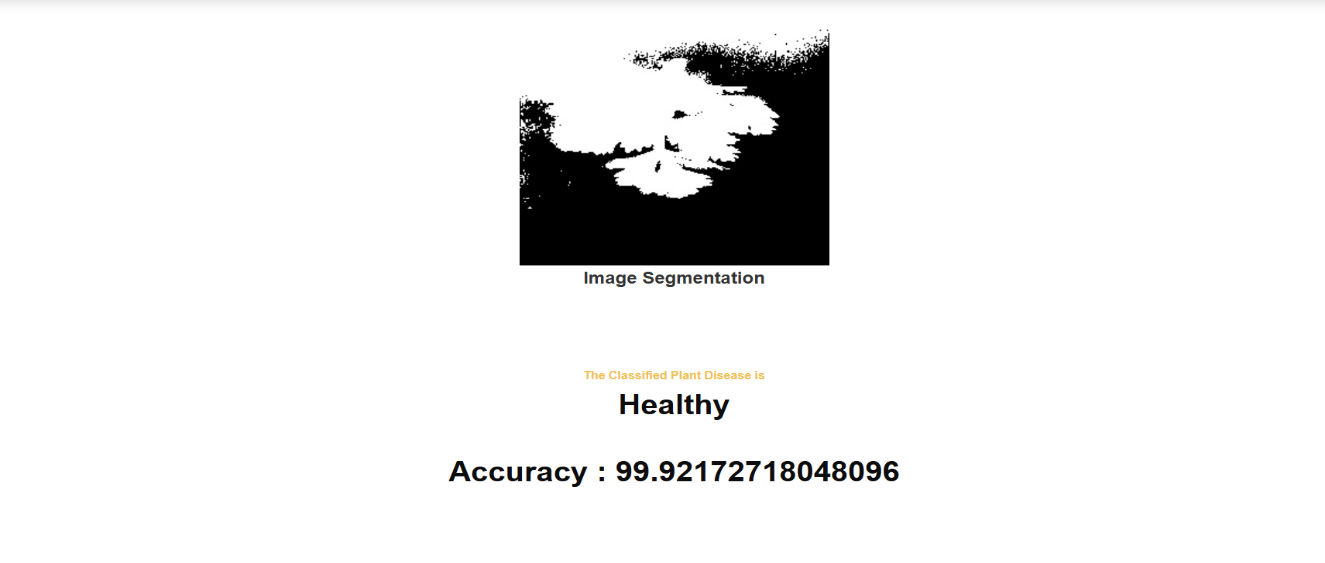
**Fig 8.3: Home page**

****

**Fig 8.4: Upload image**

****

**Fig 8.5: Species Identification**

** Fig 8.6: Disease Detection**

**Fig 8.7: status of the plant unhealthy**

**8.2 Evaluation and analysis**

The evaluation of the Plant Species and Disease Detection System was carried out by testing its functionality, accuracy, and usability in a simulated environment. The system successfully identified various plant species and their corresponding diseases from input images with high accuracy, owing to the robust MobileNetV2 model and effective preprocessing techniques like Gaussian blur and histogram equalization. The image upload and classification process were smooth, with predictions delivered in real-time and accompanied by confidence scores. Users were able to interact with the web interface to upload images and receive results intuitively, demonstrating the system’s usability. Overall, the system performed as expected in the evaluation phase and proved to be a practical solution for real-time agricultural diagnostics.

**8.3 Comparison with existing approaches**

The proposed Plant Species and Disease Detection System demonstrates significant improvements over traditional and existing solutions in terms of accessibility, usability, and technical performance. Many existing systems either rely on heavy, server-dependent models or require constant internet connectivity, limiting their usability in rural or low-connectivity regions. In contrast, our system offers offline functionality through a lightweight desktop-based web interface powered by Flask, making it accessible to farmers and non-technical users.

While some existing tools provide either species identification or disease detection, our system integrates both functions into a single platform using two separates yet efficient MobileNetV2 models. This modular design allows faster inference and better scalability. Moreover, enhanced image preprocessing techniques, including blurring, histogram equalization, and edge detection, improve model input quality—something not commonly implemented in simpler systems.

In summary, compared to existing methods, the proposed system offers a more user-friendly, efficient, and accessible solution for real-time plant health monitoring and classification, making it well-suited for deployment in tech-driven modern agriculture.

**CONCLUSION AND FUTURE ENHANCEMENT**

**CONCLUSION**

The Plant Species and Disease Detection System effectively leverages deep learning and computer vision to automate plant identification and disease diagnosis. It uses MobileNetV2 models for both classification tasks, ensuring a good balance between accuracy and efficiency. The Flask-based interface simplifies interaction, making the tool accessible to non-technical users. The image preprocessing steps further enhance prediction reliability. Offline operability and modular design add to its practical value. Overall, the system proves useful for farmers and agriculturists seeking real-time, low-cost plant health assessment.

**FUTURE ENHANCEMENT**

Future improvements can include expanding the model with more plant species and rare diseases for better generalization. A mobile application version can enhance accessibility for field use. Real-time weather API integration could enable location-based care suggestions. Incorporating a database would allow users to track historical plant health data. Adding voice support and multilingual interfaces can further improve usability. These enhancements would turn the system into a complete smart farming assistant.

**REFERENCES**

[1] Kalaiarasi, M., Sharmila, A., & Lakshmi, P. (2025). "Plant Leaf Disease Identification Based on Machine and Deep Learning Methods," Highlights various ML & DL methods for plant leaf disease classification.

[2] Mathina Kani, M., Rani, M., & Kumar, N. (2025). "Identification of Medicinal Plants and Related Fungal Diseases Based on Deep Learning Model," Accurate segmentation and classification with U-Net + F-Net, Kivy GUI integration.

[3] Nagababu, P., Nageena, S., Dharani, V., & Naveen, D. (2024). "Plant Disease Detection and Diagnosis," CNN-based disease identification using leaf features (color, texture).

[4] Shao, Y., & Devaraj, M. (2024). "Application of Plant Disease Identification and Detection Based on Deep Learning," CNN architectures like AlexNet, VGG, ResNet with attention mechanisms.

[5] Reddy, V.C.G., Xavier, A.V.M., & Shyni, S.S. (2024). "Plant Disease Detection and Pesticide Recommendation Using Deep Learning," Modified EfficientNet-B5, image preprocessing pipeline.

[6] Kavitha, S., Kumar, T.S., Naresh, E., et al. (2024). "Medicinal Plant Identification in Real-Time Using Deep Learning Model," MobileNet model with cloud deployment and mobile app.

[7] Elbaşı, E., et al. (2024). "Enhanced Plant Leaf Classification over a Large Number of Classes Using Machine Learning," Feature-based ML using MLP, Naive Bayes, Hoeffding Tree.

[8] Rebekha, R., et al. (2024). "Medicinal Plant Identification Using Machine Learning Algorithms," CNN and preprocessing on FLAVIA dataset.

[9] Prakash Kumar, H.R., et al. (2024). "Convolutional Neural Network with Improved Optimizer for Plant Disease Detection and Classification," CNN with improved SGD optimizer and CLAHE.

from flask import Flask, render\_template, request, redirect, url\_for, session

import re

import os

import urllib.request

from flask import Flask, flash, request, redirect, url\_for, render\_template

from werkzeug.utils import secure\_filename

import matplotlib.pyplot as plt

import pandas as pd

import cv2

import tensorflow as tf

from csv import writer

import pandas as pd

from flask\_material import Material

import torch

from torchvision import transforms

from torchvision.models import mobilenet\_v2

from PIL import Image

import torch.nn as nn

from torchvision import models as torchvision\_models

from torchvision import transforms

from PIL import Image

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

UPLOAD\_FOLDER = 'static/uploads/'

app = Flask(\_\_name\_\_)

app.secret\_key = "secret\_key"

# Load the model

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

app.config['MAX\_CONTENT\_LENGTH'] = 16 \* 1024 \* 1024

ALLOWED\_EXTENSIONS = set(['png', 'jpg', 'jpeg', 'gif'])

class\_names = [ 'Apple', 'Apple', 'Blueberry', 'Cherry', 'Cherry', 'Coriander', 'Coriander', 'Corn', 'Corn', 'Grape', 'Grape', 'Pepper', 'Pepper', 'Potato', 'Potato', 'Potato', 'Rose' ,'Rose', 'Strawberry', 'Strawberry', 'Tomato', 'Tomato', 'Tomato']

class\_name1=['Healthy', 'Apple\_\_\_Cedar\_apple\_rust', 'Healthy', 'Healthy', 'Cherry\_(including\_sour)\_\_\_Powdery\_mildew', 'Healthy', 'Coriander powdery mildew', 'Healthy', 'Corn\_(maize)\_\_\_Cercospora\_leaf\_spot Gray\_leaf\_spot', 'Healthy', 'Grape\_\_\_Esca\_(Black\_Measles)', 'Healthy', 'Pepper,\_bell\_\_\_Bacterial\_spot', 'Healthy', 'Potato\_\_\_Early\_blight', 'Potato\_\_\_Late\_blight', 'Healthy', 'Rose\_Rust', 'Healthy', 'Strawberry\_\_\_Leaf\_scorch', 'Healthy', 'Tomato\_\_\_Tomato\_Yellow\_Leaf\_Curl\_Virus', 'Tomato\_\_\_Tomato\_mosaic\_virus']

img\_height = 224

img\_width = 224

def allowed\_file(filename):

    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

# Initialize a StandardScaler (make sure to use the same scaler as the one used during training)

@app.route('/', methods=['GET', 'POST'])

def login():

    msg = ''

    if request.method == 'POST' and 'username' in request.form and 'password' in request.form:

        username = request.form['username']

        password = request.form['password']

        if username == "admin" and password == "admin":

            return render\_template('index.html')

        else:

            msg = 'Incorrect username/password!'

    return render\_template('login.html', msg=msg)

@app.route('/home')

def home():

    return render\_template('index.html')

@app.route('/about')

def about():

    return render\_template('about.html')

@app.route('/contact')

def contact():

    return render\_template('contact.html')

@app.route('/healthy')

def healthy():

    return render\_template('healthy.html')

def save\_images\_with\_techniques(img\_path):

    # Load the image

    img = cv2.imread(img\_path)

    # Ensure that the "static" folder exists

    static\_folder = "static"

    if not os.path.exists(static\_folder):

        os.makedirs(static\_folder)

    # Save each image with the specified technique used

    img\_folder = os.path.join(static\_folder, "images")

    if not os.path.exists(img\_folder):

        os.makedirs(img\_folder)

    # Gaussian Blur

    blur = cv2.GaussianBlur(img, (5, 5), 0)

    cv2.imwrite(os.path.join(img\_folder, '1\_blur.jpg'), blur)

    # Smoothing using mean filter

    mean\_blur = cv2.blur(img, (5, 5))

    cv2.imwrite(os.path.join(img\_folder, '2\_mean\_blur.jpg'), mean\_blur)

    # Contrast normalization using histogram equalization

    gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

    equalized = cv2.equalizeHist(gray)

    cv2.imwrite(os.path.join(img\_folder, '3\_equalized.jpg'), equalized)

    # Sobel Filtering

    sobelx = cv2.Sobel(equalized, cv2.CV\_64F, 1, 0, ksize=5)

    sobely = cv2.Sobel(equalized, cv2.CV\_64F, 0, 1, ksize=5)

    sobel\_combined = cv2.magnitude(sobelx, sobely)

    cv2.imwrite(os.path.join(img\_folder, '4\_sobel\_filtered.jpg'), sobel\_combined)

    # Image Segmentation

    \_, binary = cv2.threshold(equalized, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)

    cv2.imwrite(os.path.join(img\_folder, '5\_segmented.jpg'), binary)

transform = transforms.Compose([

    transforms.Resize((224, 224)),

    transforms.ToTensor(),

    transforms.Normalize(mean=[0.485, 0.456, 0.406],

                         std=[0.229, 0.224, 0.225])

])

# Load the model architecture

num\_classes = len(class\_name1)

model = mobilenet\_v2(weights=None)  # avoid pretrained warning

model.classifier[1] = torch.nn.Linear(model.last\_channel, num\_classes)

# Load trained weights

model\_path = "plant\_LeafHealthy\_Disease\_mobilenetv2.pth"

model.load\_state\_dict(torch.load(model\_path, map\_location=device))

model.to(device)

model.eval()

# Prediction function

def predict\_leaf(image\_path):

    image = Image.open(image\_path).convert('RGB')

    image\_tensor = transform(image).unsqueeze(0).to(device)

    save\_images\_with\_techniques(image\_path)

    with torch.no\_grad():

        output = model(image\_tensor)

        probabilities = torch.nn.functional.softmax(output, dim=1)

        top\_prob, pred\_idx = torch.max(probabilities, dim=1)

        class\_name = class\_names[pred\_idx.item()]

        confidence = top\_prob.item() \* 100

    print(f"Predicted: {class\_name}")

    print(f"Confidence: {confidence:.2f}%")

    return class\_name, confidence

@app.route('/upload\_image1',methods=["POST"])

def upload\_image1():

    if 'file' not in request.files:

        flash('No file part')

        return redirect(request.url)

    file = request.files['file']

    print(file)

    if file.filename == '':

        flash('No image selected for uploading')

        return redirect(request.url)

    if file and allowed\_file(file.filename):

        filename = secure\_filename(file.filename)

        file.save(os.path.join(app.config['UPLOAD\_FOLDER'], filename))

        path = os.path.join(app.config['UPLOAD\_FOLDER'], filename)

        # --- Example Run ---

        image\_path = path  # Replace with your image

        class\_name, confidence=predict\_leaf(image\_path)       return render\_template('healthy.html',aclass=class\_name,ascore=confidence,res=2)

# Prediction function

# Load the trained model

model1 = mobilenet\_v2(pretrained=False)

model1.classifier[1] = torch.nn.Linear(model1.last\_channel, len(class\_name1))

model1.load\_state\_dict(torch.load(r"plant\_LeafHealthy\_Disease\_mobilenetv2.pth", map\_location=device))

model1 = model1.to(device)

model1.eval()

import torch.nn.functional as F

def predict\_image(image\_path):

    image = Image.open(image\_path).convert('RGB')

    img\_tensor = transform(image).unsqueeze(0).to(device)

    save\_images\_with\_techniques(image\_path)

    with torch.no\_grad():

        output = model1(img\_tensor)

        probabilities = F.softmax(output, dim=1)

        confidence, predicted = torch.max(probabilities, 1)

        predicted\_class = class\_name1[predicted.item()]

        confidence\_score = confidence.item() \* 100  # convert to %

        return predicted\_class, confidence\_score

@app.route('/upload\_image',methods=["POST"])

def upload\_image():

    if 'file' not in request.files:

        flash('No file part')

        return redirect(request.url)

    file = request.files['file']

    print(file)

    if file.filename == '':

        flash('No image selected for uploading')

        return redirect(request.url)

    if file and allowed\_file(file.filename):

        filename = secure\_filename(file.filename)

        file.save(os.path.join(app.config['UPLOAD\_FOLDER'], filename))

        path = os.path.join(app.config['UPLOAD\_FOLDER'], filename)

        # Load model

        image\_path = path  # Replace with your image

        class\_name, confidence=predict\_image(image\_path)       return render\_template('contact.html',aclass=class\_name,ascore=confidence,res=2)

@app.route('/display/<filename>')

def display\_image(filename):

    return redirect(url\_for('static', filename='images/' + filename), code=301)

@app.route('/logout')

def logout():

    session.pop('user', None)

    return redirect(url\_for('login'))

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)