BLOCKCHAIN BASED CERTIFICATE CREATION AND VALIDATION

Project Submitted to Thiruvalluvar University, Serkkadu, Vellore

In Partial Fulfillment for the award of degree

BACHELOR OF COMPUTER APPLICATION

By

THANSEER BASHA V	**(20922U09193)**
THAWFICK A	**(20922U09194)**
UDHAYAKUMAR S	**(20922U09195)**
UTHRAGIRI K	**(20922U09197)**
VENGATESAN G	**(20922U09198)**

Under the Guidance of

Prof. S. ARUN MCA., M.Phil., B.Ed.,

Assistant Professor,

Department of Computer Application,

Shanmuga Industries Arts and Science College,

Tiruvannamalai

DEPARTMENT OF COMPUTER APPLICATION

SHANMUGA INDUSTRIES ARTS AND SCIENCE COLLEGE

MARCH - 2025

BLOCKCHAIN BASED CERTIFICATE CREATION AND VALIDATION

Bonafied Workdone by

THANSEER BASHA V	**(20922U09193)**
THAWFICK A	**(20922U09194)**
UDHAYAKUMAR S	**(20922U09195)**
UTHRAGIRI K	**(20922U09197)**
VENGATESAN G	**(20922U09198)**

Project work submitted by in partial fulfillment of the requirement for the award of degree

BACHELORE OF COMPUTER APPLICATION

To

DEPARTMENT OF COMPUTER APPLICATION

SHANMUGA INDUSTRIES ARTS AND SCIENCE COLLEGE

Accredited with "B+" Grade by NAAC

Certified Under Section 2(f) & 12B of the UGC Act 1956, An ISO 9001:2015 Certified Institution

Permanently Affiliated to Thiruvalluvar University, Vellore – 632 115.

Approved by Govt. of Tamil Nadu & AICTE.

INTERNAL GUIDE **HEAD OF THE DEPARTMENT**

Submitted to the Viva – Voice Examination held on ____________________

Internal Examiner **External Examiner**

DECLARATION

We declare that the project entitled **“BLOCKCHAIN BASED CERTIFICATE CREATION AND VALIDATION”** submitted to Department of Computer Application, Shanmuga Industries Arts and Science College, Tiruvannamalai. In partial fulfillment of the requirement for the award of the degree **Bachelor of computer Application** for Thiruvalluvar University, Vellore – 632 115.

Place :

Date :

Signature of the Candidates

THANSEER BASHA V	**(20922U09193)**
THAWFICK A	**(20922U09194)**
UDHAYAKUMAR S	**(20922U09195)**
UTHRAGIRI K	**(20922U09197)**
VENGATESAN G	**(20922U09198)**

ACKNOWLEDGEMENT

Firstly, we are deeply obliged to the almighty for this opportunity and for leading us to the successful completion of the project. The experience has been very fruitful.

We Wish to acknowledge my indebtedness and deep sense of gratitude to the Honorable Chairman **Mr. N. PALANI B.Sc., LLB.,** Beloved Secretary & Correspondent **Mr. L. VIJAY ANAND, B.Com.,** and Beloved Treasurer **Mr. E. SRIDHAR B.Com.,** Shanmuga Industries Arts and Science College, Tiruvannamalai.

Also we like to express my gratitude to Prof. **Dr. K. ANANDARAJ M.Sc., M.Phil., Ph.D., Principal,** for his constant support and motivation throughout the project.

We were obliged and indebted to thanks **Prof. V. SAKTHIVEL MCA., M.Phil., Professor,** Department of Computer Application Who helped to complete my project successfully.

We are very thankful to my Guide **Prof. S. ARUN MCA., M.Phil., B.Ed., Assistant Professor,** Department of Computer Application who helped me to complete the dissertation successfully. I am grateful to her for her encouragement, invaluable guidance throughout my project.

Last but not the least, we are extremely grateful to all faculties, staff members, family and friends, who have been a constant source of support during the preparation of this project work.

ABSTRACT

This report details a Decentralized application (D App) designed for the creation and verification of certificates utilizing blockchain technology. The project leverages the Ethereum blockchain to ensure the integrity and immutability of certificate records through smart contracts. Educational institutions and certifying bodies can securely generate and issue digital certificates, while verifiers can transparently validate their authenticity. To enhance accessibility, the digital PDF copies of the certificates are stored on the Inter Planetary File System (IPFS), identified by unique content hashes. The technology stack underpinning this project includes Ganache-cli for a local blockchain network, Truffle for smart contract development and deployment, Streamlit for the web application interface, Pinata as an IPFS client, and Docker for containerization. The primary goal of this initiative is to modernize the certificate management process, providing a secure, efficient, and transparent solution that addresses the limitations of traditional paper-based systems

CHAPTER 1:

INTRODUCTION

1.1 Significance of Digital Certificates and Limitations of Traditional Methods:

The increasing digitalization of various aspects of life has amplified the importance of secure electronic communication and data exchange. Digital certificates play a crucial role in this landscape by facilitating the verification of identities and enabling the encryption of online interactions between individuals, systems, and devices. The process of verifying paper certificates is often time-consuming and cumbersome, typically relying on centralized authorities and manual cross-referencing, which can lead to delays and administrative burdens. Sharing these credentials between institutions or individuals can also be complicated, often involving physical documentation and lengthy procedures. In contrast, digital certificates offer immediate access to credentials, enhanced credibility through online verification, portability across devices, seamless integration with digital platforms, cost-effectiveness by reducing printing and administrative overhead, and improved record-keeping through centralized digital management.

The shift towards digital certificates represents a fundamental change driven by the imperative for enhanced security, trust, and efficiency in our increasingly digital world. The inherent limitations of paper-based systems in providing these critical attributes are significant factors propelling the adoption of digital alternatives. Beyond security, digital certificates offer substantial operational advantages, including reduced costs associated with physical document handling, streamlined verification processes, and improved accessibility for individuals and organizations. These benefits collectively position digital certificates as a compelling solution for modern credential management in a landscape where trust and efficiency are paramount.

1.2 Overview of the Blockchain-Based Certificate Verification Dapp:

This project introduces a blockchain-based certificate verification DApp built upon the Ethereum blockchain. The core functionality of this DApp revolves around two primary entities: the Institute, responsible for the secure generation and issuance of certificates, and the Verifier, tasked with validating the authenticity of these credentials. Leveraging the inherent properties of blockchain technology, the DApp ensures the integrity and immutability of all certificate records through the implementation of smart contracts. Furthermore, to facilitate ease of access and sharing, the digital PDF copies of the issued certificates are stored on the InterPlanetary File System (IPFS), a decentralized storage network, and are identified by unique content-based hashes. This architecture combines the security of blockchain with the accessibility of decentralized storage to create a modern and efficient certificate management system.

1.3 Project Goals and Objectives

The overarching **goal** of this project is to establish a secure and transparent system for the issuance and verification of certificates, addressing the shortcomings of traditional methods and leveraging the benefits of blockchain technology. To achieve this goal, several key **objectives** have been identified:

1. **To leverage blockchain's immutability to ensure the integrity of certificate records:** By recording certificate details and their verification status on the blockchain, the system aims to create an unalterable and auditable record, thereby guaranteeing the authenticity

and trustworthiness of the credentials.

2. **To utilize IPFS for decentralized and easily accessible storage of certificate documents:** Storing the actual certificate files on IPFS, a distributed storage network, will ensure their long-term availability and prevent reliance on a single point of failure, while the content-based addressing ensures the integrity of the files.
3. **To streamline the certificate generation and validation process for both issuing institutions and verifying parties:** The DApp is designed to simplify the often-cumbersome processes of certificate creation and verification, making them more efficient and less prone to errors.
4. **To develop a user-friendly web application interface for interacting with the blockchain and IPFS:** A Streamlit-based web application will provide an intuitive interface for both Institutes to manage their certificate issuance and for Verifiers to easily validate credentials, regardless of their technical expertise.

CHAPTER 2:

BLOCKCHAIN TECHNOLOGY AND ITS APPLICATION

2.1 Fundamental Concepts of Blockchain Technology: Immutability and Decentralization

At its core, blockchain technology offers a paradigm shift in how data is stored and managed, with two fundamental concepts driving its transformative potential: immutability and decentralization. **Immutability** in the context of blockchain signifies that once information is recorded on the ledger, it becomes exceedingly difficult, if not practically impossible, to alter, overwrite, or delete it without the consensus of the entire network.

The second fundamental concept is **decentralization**, which refers to the distribution of control and decision-making authority across multiple participants, or nodes, rather than being held by a single central entity. In a decentralized blockchain network, no single organization or individual has unilateral control over the system. Instead, the network operates on a peer-to-peer basis, where multiple computers maintain a copy of the blockchain and collectively validate transactions. Furthermore, decentralization fosters a **trustless environment** among network participants.

The inherent immutability of blockchain technology is particularly vital for a certificate verification system. It provides a strong guarantee that once a certificate is recorded on the blockchain, it cannot be fraudulently modified. This establishes a high level of confidence in the authenticity of the credential, as any attempt to tamper with the record would be immediately apparent due to the broken cryptographic chain.

<u>2.2 Application of Blockchain for Certificate Verification</u>

Blockchain technology offers a compelling solution for the challenges associated with traditional certificate verification methods, providing a **tamper-proof and efficient** means to confirm the authenticity of credentials. For organizations that frequently need to verify qualifications, such as educational institutions and employers, blockchain-based systems offer numerous advantages. These include improved trust in the validity of credentials, a significant reduction in fraudulent certificate claims, a more streamlined and faster verification process, and ultimately, lower verification costs. A key strength of blockchain in this context is its **immutability**, which makes it exceptionally difficult to manipulate recorded data or create false credentials, thereby substantially reducing the risk of fraud [8]. This inherent security is a significant improvement over traditional paper-based systems that are susceptible to forgery and alteration.
Furthermore, blockchain technology provides a **decentralized network** where the authenticity of a certificate can be accessed and verified by anyone with the appropriate permissions, simply by comparing the digital signature recorded on the blockchain. This transparency fosters a greater level of trust among all stakeholders involved in the certification process.

CHAPTER 3:

SYSTEM ANALYSIS AND DESIGN

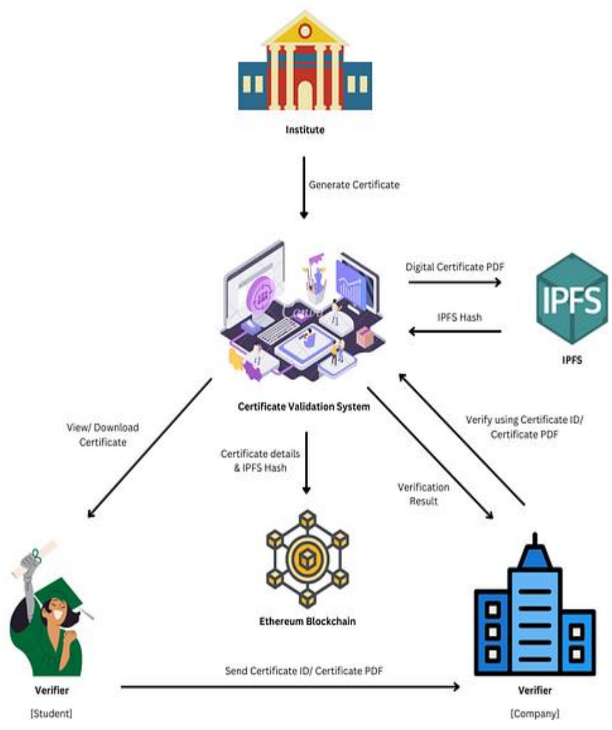
3.1 System Architecture of the Dapp

The blockchain-based certificate verification DApp is structured around a multi-layered architecture to ensure security, accessibility, and efficient operation.

The core components of this system include:

- **Smart Contract Layer:** At the heart of the DApp resides the smart contract, deployed on the Ethereum blockchain. This layer is responsible for storing the cryptographic hashes of the certificates and implementing the core verification logic. The immutability of the blockchain ensures that these records cannot be tampered with once they are recorded.
- **IPFS Storage:** The actual digital PDF copies of the certificates are stored off-chain on the InterPlanetary File System (IPFS). IPFS is a decentralized storage network that utilizes content-based addressing, meaning each file is uniquely identified by its content hash. This ensures that the correct and unaltered certificate is always accessible.
- **Web Application (Streamlit):** The user interface for interacting with the DApp is developed using Streamlit, a Python-based framework for creating web applications with minimal coding. This layer provides intuitive interfaces for both the Institutes to issue and manage certificates and for Verifiers to validate them.
- **Backend Services (Python):** The Streamlit web application communicates with the Ethereum blockchain and the IPFS network through backend services written in Python. These services utilize libraries such as Web3.py to interact with the Ethereum smart contract and Pinata's API to manage files on IPFS.

- **Local Blockchain Network (Ganache-cli):** During the development and testing phases, a local Ethereum blockchain network is emulated using Ganache-cli. This provides a controlled environment for developers to test the smart contracts and the DApp's functionality without incurring costs on a public network.


Institute
Generate Certificate
Digital Certificate PDF
IPFS Hash
IPFS
Certificate Validation System
View/ Download Certificate
Certificate details & IPFS Hash
Verification Result
Verify using Certificate ID/ Certificate PDF
Ethereum Blockchain
Verifier
[Student]
Send Certificate ID/ Certificate PDF
Verifier
[Company]

3.2 Description of Key Entities: Institute and Verifier

The system is designed to accommodate two primary entities, each with distinct roles and functionalities:

3.2.1 Institute

The Institute represents the educational institution, organization, or certification authority that is authorized to generate and issue certificates through the DApp.

- **Role:** Responsible for creating, issuing, and managing digital certificates.

Functionalities:

- **Authentication:** Institutes access the web application through a secure login system powered by Firebase, ensuring that only authorized personnel can issue certificates.

- **Certificate Generation:** The Institute user can input the necessary details for a certificate, such as the student's Unique Identifier (UID), candidate name, course name, and the issuing organization's name. Upon submission, the backend generates a professional-looking PDF certificate using the **report lab** library.
-
- **IPFS Upload:** The generated PDF certificate is then securely uploaded to the IPFS network using the Pinata IPFS client. Pinata returns a unique IPFS hash that serves as a permanent and content-addressed identifier for the certificate.

- **Blockchain Transaction:** The backend calculates a unique Certificate ID (a SHA256 hash of the relevant certificate data) and initiates a transaction to the Ethereum smart contract (**Certification.sol**). This transaction calls the **generateCertificate** function, storing the Certificate ID, the student's details, the organization's name, and the IPFS hash on the blockchain. The smart contract emits a **certificateGenerated** event upon successful creation of the record.

- **Certificate Viewing:** The Institute user has the ability to view the details of the certificates they have issued. By entering the Certificate ID, the system queries the blockchain, retrieves the associated IPFS hash, fetches the certificate PDF from IPFS, and displays it within the web application.

3.2.2 Verifier

The Verifier represents any third party, such as an employer or another educational institution, that needs to validate the authenticity of a certificate issued through the DApp.

- **Role:** Responsible for verifying the validity and authenticity of certificates.

Functionalities:

- **Authentication:** Verifiers also access the web application through Firebase authentication, ensuring a secure and auditable verification process.

- **Verification by PDF Upload:** A Verifier can upload a digital PDF copy of a certificate they wish to validate. The backend extracts relevant information from the PDF, such as the UID, candidate name, course name, and organization name, using the **pdf plumber** library. It then calculates the Certificate ID based on this extracted data and queries the Ethereum smart contract using this ID. The system checks if a certificate with the calculated ID exists on the blockchain. If a record is found, the system confirms the certificate's authenticity.

- **Verification by Certificate ID Input:** Alternatively, a Verifier can directly input the Certificate ID of the certificate they want to verify. The system then queries the Ethereum smart contract using this ID. If a record exists, the system retrieves the associated IPFS hash and fetches the corresponding certificate PDF from IPFS, displaying it to the Verifier. Additionally, the system calls the is Verified function on the smart contract to confirm the existence of the certificate record on the blockchain.

3.3 Data Flow within the System

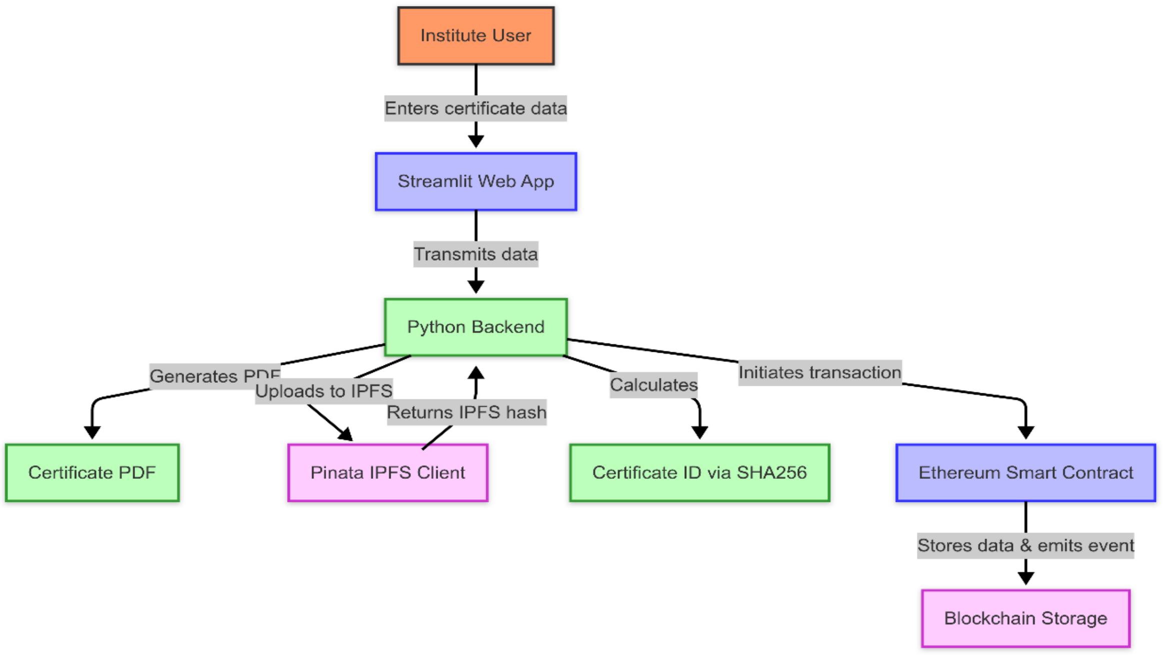


Fig 3.3.1 Certification Creation

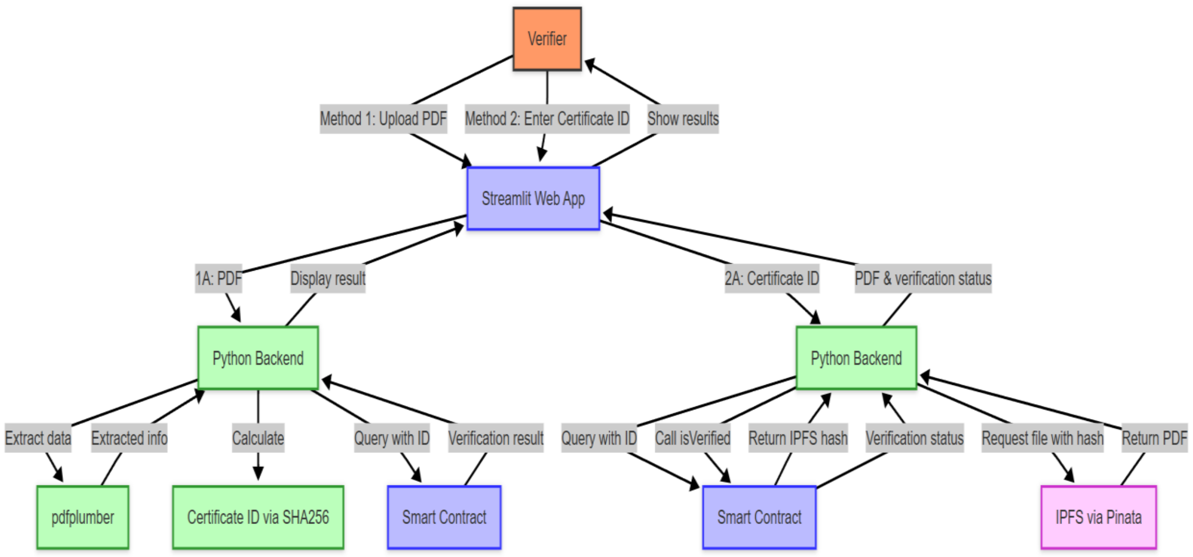


Fig 3.3.2 Certificate Validation

CHAPTER 4:

TECHNOLOGY STACK AND FUNCTIONALITIES

The blockchain-based certificate verification DApp leverages a modern technology stack:

1. **Ethereum Blockchain**: Provides the immutable foundation for certificate verification through smart contracts.

2. **IPFS with Pinata**: Offers decentralized storage for certificate PDFs, using content-based addressing rather than location-based access.

3. **Ganache-cli & Truffle**: Facilitates local blockchain development, testing, and smart contract deployment.

4. **Streamlit**: Powers the Python-based web interface, enabling rapid development of interactive verification tools.

5. **Docker**: Ensures consistency across environments by containerizing the application components.

6. This stack separates certificate content (stored on IPFS) from verification records (on blockchain) for optimal efficiency and security.

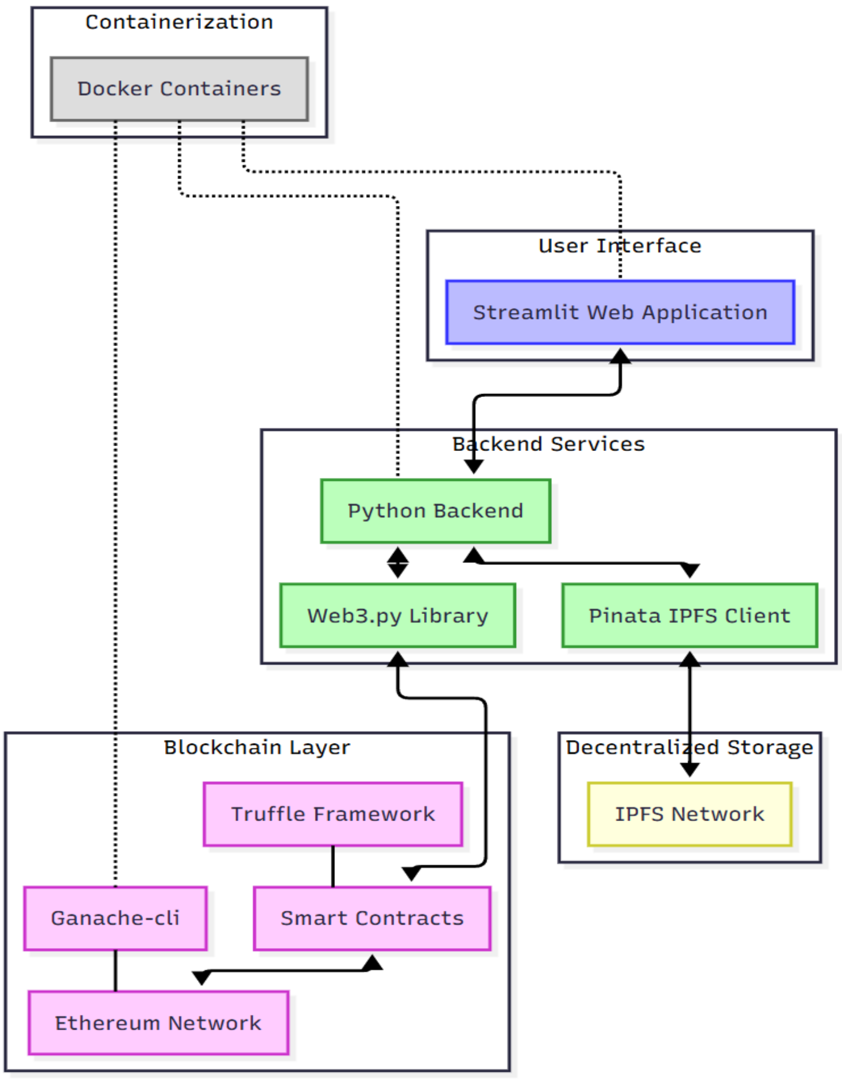
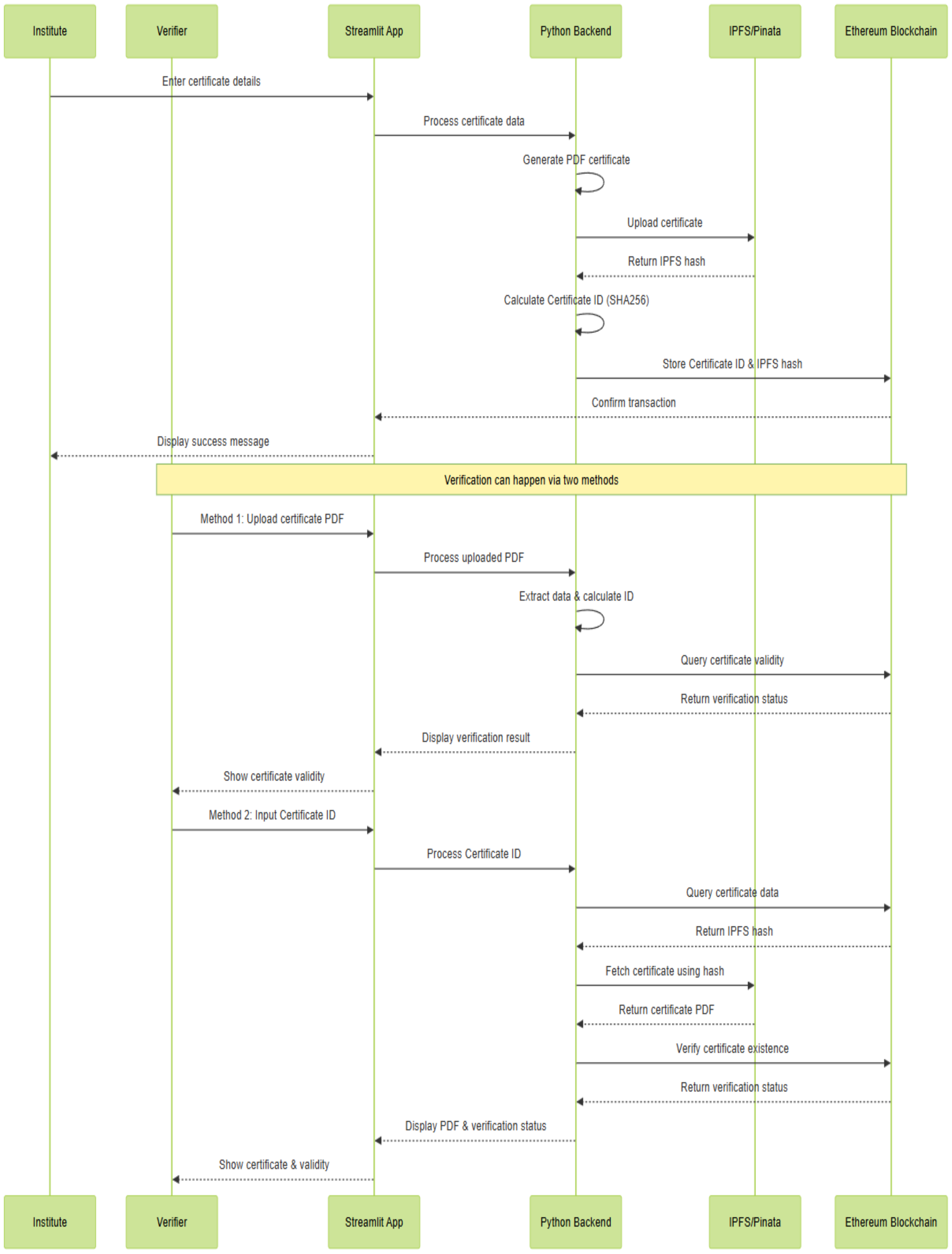


Fig 4.1 Tech Stack Architecture

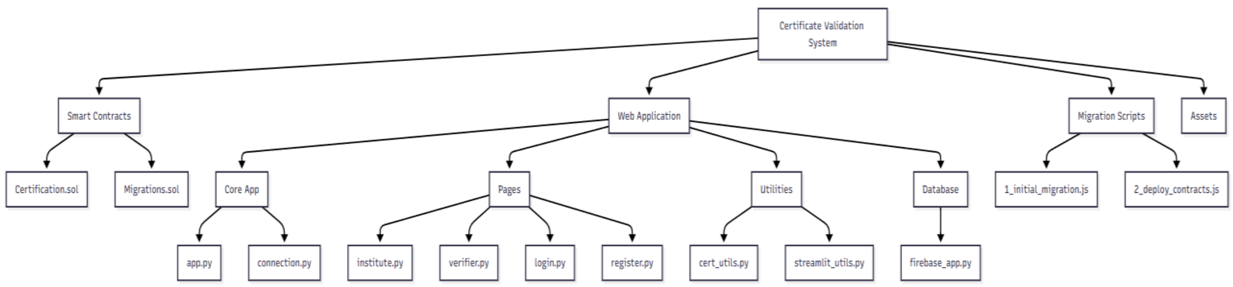

Institute
Verifier
Streamlit App
Python Backend
IPFS/Pinata
Ethereum Blockchain
Enter certificate details
Process certificate data
Generate PDF certificate
Upload certificate
Return IPFS hash
Calculate Certificate ID (SHA256)
Store Certificate ID & IPFS hash
Confirm transaction
Display success message
Verification can happen via two methods
Method 1: Upload certificate PDF
Process uploaded PDF
Extract data & calculate ID
Query certificate validity
Return verification status
Display verification result
Show certificate validity
Method 2: Input Certificate ID
Process Certificate ID
Query certificate data
Return IPFS hash
Fetch certificate using hash
Return certificate PDF
Verify certificate existence
Return verification status
Display PDF & verification status
Show certificate & validity
Institute
Verifier
Streamlit App
Python Backend
IPFS/Pinata
Ethereum Blockchain

CHAPTER 5:

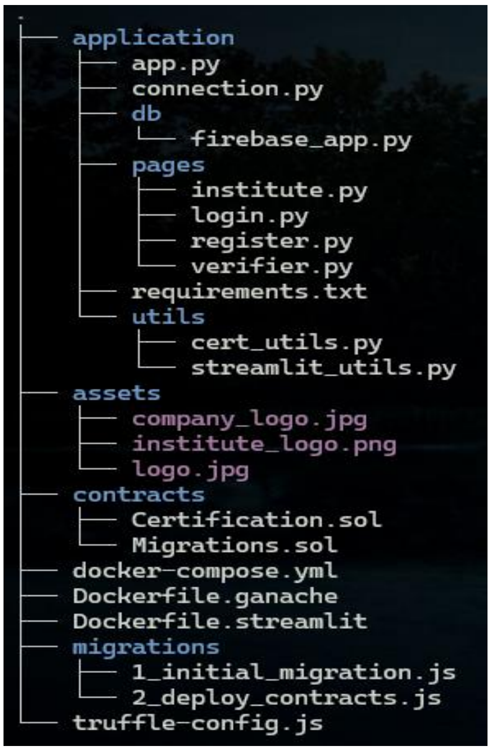
SYSTEM DEVELOPMENT AND IMPLEMENTATION

5.1 Project Setup and Structure

The certificate validation system was implemented using Truffle framework for smart contract development and Streamlit for the web application interface. The project follows a modular structure with clear separation between blockchain components and frontend application.



Tree structure of the project



```
.
├── application
│   ├── app.py
│   ├── connection.py
│   ├── db
│   │   └── firebase_app.py
│   ├── pages
│   │   ├── institute.py
│   │   ├── login.py
│   │   ├── register.py
│   │   └── verifier.py
│   ├── requirements.txt
│   └── utils
│       ├── cert_utils.py
│       └── streamlit_utils.py
├── assets
│   ├── company_logo.jpg
│   ├── institute_logo.png
│   └── logo.jpg
├── contracts
│   ├── Certification.sol
│   └── Migrations.sol
├── docker-compose.yml
├── Dockerfile.ganache
├── Dockerfile.streamlit
├── migrations
│   ├── 1_initial_migration.js
│   └── 2_deploy_contracts.js
└── truffle-config.js
```

Fig 5.1.1 Project Structure

5.2 Smart Contract Implementation

The core of the system is the **Certification.sol** smart contract, which provides functionality for generating, storing, and verifying certificate records on the Ethereum blockchain.

Certification.sol

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

contract Certification {
    struct Certificate {
        string uid;
        string candidate_name;
        string course_name;
        string org_name;
        string ipfs_hash;
    }

    mapping(string => Certificate) public certificates;
    event certificateGenerated(string certificate_id);

    function generateCertificate(
        string memory _certificate_id,
        string memory _uid,
        string memory _candidate_name,
        string memory _course_name,
        string memory _org_name,
        string memory _ipfs_hash
    ) public {
        // Check if certificate with the given ID already exists
        require(
            bytes(certificates[_certificate_id].ipfs_hash).length == 0,
            "Certificate with this ID already exists"
        );

        // Create the certificate
        Certificate memory cert = Certificate({
            uid: _uid,
            candidate_name: _candidate_name,
            course_name: _course_name,
            org_name: _org_name,
            ipfs_hash: _ipfs_hash
        });

        // Store the certificate in the mapping
        certificates[_certificate_id] = cert;

        // Emit an event
        emit certificateGenerated(_certificate_id);
    }

    function getCertificate(
        string memory _certificate_id
    )
        public
        view
        returns (
            string memory _uid,
            string memory _candidate_name,
            string memory _course_name,
            string memory _org_name,
            string memory _ipfs_hash
        )
    {
        Certificate memory cert = certificates[_certificate_id];

        // Check if the certificate with the given ID exists
        require(
            bytes(certificates[_certificate_id].ipfs_hash).length != 0,
            "Certificate with this ID does not exist"
        );

        // Return the values from the certificate
        return (
            cert.uid,
            cert.candidate_name,
            cert.course_name,
            cert.org_name,
            cert.ipfs_hash
        );
    }

    function isVerified(
        string memory _certificate_id
    ) public view returns (bool) {
        return bytes(certificates[_certificate_id].ipfs_hash).length != 0;
    }
}

```

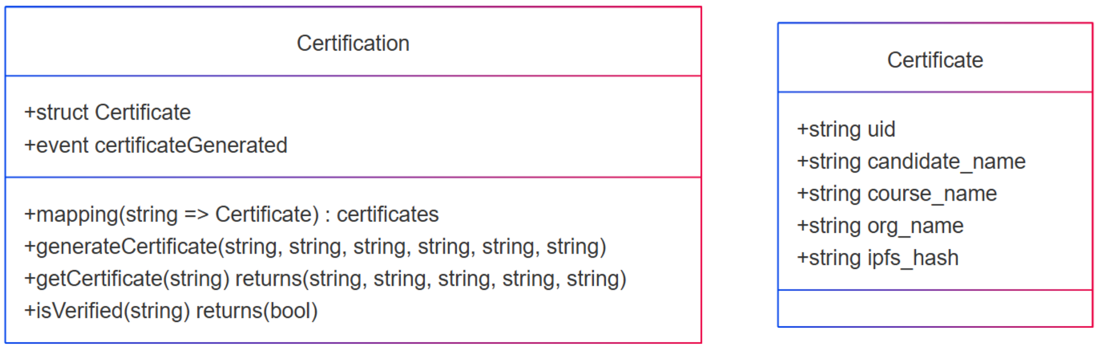

Certification
+struct Certificate
+event certificateGenerated
+mapping(string => Certificate) : certificates
+generateCertificate(string, string, string, string, string, string)
+getCertificate(string) returns(string, string, string, string, string)
+isVerified(string) returns(bool)
Certificate
+string uid
+string candidate_name
+string course_name
+string org_name
+string ipfs_hash

Fig 5.2.1 Smart Contract Functions

5.3 Deployment Process

The system deployment involves setting up a local Ethereum network with Ganache, deploying smart contracts using Truffle, and running the Streamlit application.

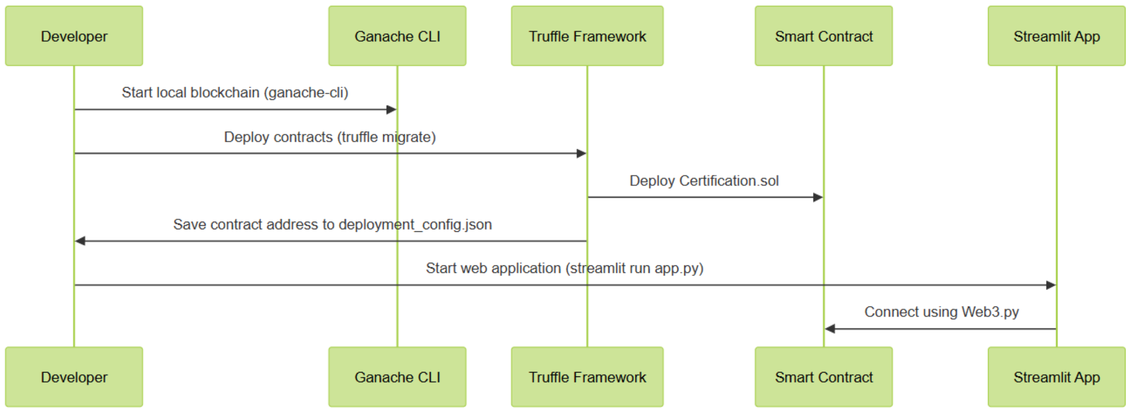


Fig 5.3.1 Deployment Flow

5.4 Web Application Components

The web application is built using Streamlit and consists of several key components: authentication, certificate generation, IPFS storage, and certificate verification.

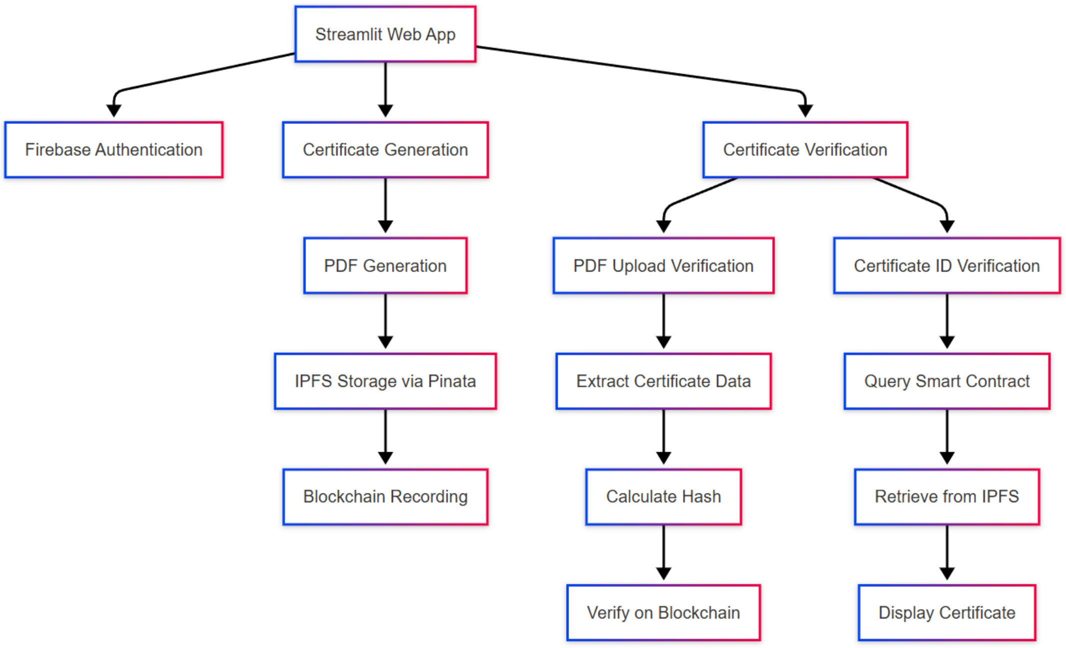


Fig 5.4.1 Application Component Interaction

Firebase_app.py

```python
import pyrebase
from dotenv import load_dotenv
import os

load_dotenv()

config = {
    "apiKey": os.getenv("FIREBASE_API_KEY"),
    "authDomain": os.getenv("FIREBASE_AUTH_DOMAIN"),
    "databaseURL": os.getenv("FIREBASE_DATABASE_URL"),
    "projectId": os.getenv("FIREBASE_PROJECT_ID"),
    "storageBucket": os.getenv("FIREBASE_STORAGE_BUCKET"),
    "messagingSenderId": os.getenv("FIREBASE_MESSAGING_SENDER_ID"),
    "appId": os.getenv("FIREBASE_APP_ID"),
}

firebase = pyrebase.initialize_app(config)
auth = firebase.auth()

def register(email, password):
    try:
        auth.create_user_with_email_and_password(email, password)
        return "success"
    except Exception as e:
        print(f"Error: {e}")
        return "failure"

def login(email, password):
    try:
        auth.sign_in_with_email_and_password(email, password)
        return "success"
    except Exception as e:
        print(f"Error: {e}")
        return "failure"
```

app.py

```python
import streamlit as st
from PIL import Image
from utils.streamlit_utils import hide_icons, hide_sidebar, remove_whitespaces
from streamlit_extras.switch_page_button import switch_page

st.set_page_config(layout="wide", initial_sidebar_state="collapsed")
hide_icons()
hide_sidebar()
remove_whitespaces()

st.title("Certificate Validation System")
st.write("")
st.subheader("Select Your Role")

col1, col2 = st.columns(2)
institite_logo = Image.open("../assets/institute_logo.png")
with col1:
    st.image(institite_logo, output_format="jpg", width=230)
    clicked_institute = st.button("Institute")

company_logo = Image.open("../assets/company_logo.jpg")
with col2:
    st.image(company_logo, output_format="jpg", width=230)
    clicked_verifier = st.button("Verifier")

if clicked_institute:
    st.session_state.profile = "Institute"
    switch_page('login')
elif clicked_verifier:
    st.session_state.profile = "Verifier"
    switch_page('login')

```

Cert_util.py (for certificate generation):

```
from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
import pdfplumber

def generate_certificate(output_path, uid, candidate_name, course_name, org_name,
institute_logo_path):
    # Create a PDF document
    doc = SimpleDocTemplate(output_path, pagesize=letter)

    # Create a list to hold the elements of the PDF
    elements = []

    # Add institute logo and institute name
    if institute_logo_path:
        logo = Image(institute_logo_path, width=150, height=150)
        elements.append(logo)

    # Add institute name
    institute_style = ParagraphStyle(
        "InstituteStyle",
        parent=getSampleStyleSheet()["Title"],
        fontName="Helvetica-Bold",
        fontSize=15,
        spaceAfter=40,
    )
    institute = Paragraph(org_name, institute_style)
    elements.extend([institute, Spacer(1, 12)])

    # Add title
    title_style = ParagraphStyle(
        "TitleStyle",
        parent=getSampleStyleSheet()["Title"],
        fontName="Helvetica-Bold",
        fontSize=25,
        spaceAfter=20,
    )
    title1 = Paragraph("Certificate of Completion", title_style)
    elements.extend([title1, Spacer(1, 6)])

    # Add recipient name, UID, and course name with increased line space
    recipient_style = ParagraphStyle(
        "RecipientStyle",
        parent=getSampleStyleSheet()["BodyText"],
        fontSize=14,
        spaceAfter=6,
        leading=18,
        alignment=1
    )

    recipient_text = f"This is to certify that<br/><br/>\
                     <font color='red'> {candidate_name} </font><br/>\
                     with UID <br/> \
                    <font color='red'> {uid} </font> <br/><br/>\
                     has successfully completed the course:<br/>\
                     <font color='blue'> {course_name} </font>"

    recipient = Paragraph(recipient_text, recipient_style)
    elements.extend([recipient, Spacer(1, 12)])

```

```python
    # Build the PDF document
    doc.build(elements)

    print(f"Certificate generated and saved at: {output_path}")

def extract_certificate(pdf_path):
    with pdfplumber.open(pdf_path) as pdf:
        # Extract text from each page
        text = ""
        for page in pdf.pages:
            text += page.extract_text()
        lines = text.splitlines()

        org_name = lines[0]
        candidate_name = lines[3]
        uid = lines[5]
        course_name = lines[-1]

        return (uid, candidate_name, course_name, org_name)

```

streamlit_utils.py

```python
import streamlit as st
import base64
import requests
import os
from connection import contract

def displayPDF(file):
    # Opening file from file path
    with open(file, "rb") as f:
        base64_pdf = base64.b64encode(f.read()).decode('utf-8')

    # Embedding PDF in HTML
    pdf_display = F'<iframe src="data:application/pdf;base64,{base64_pdf}" width="700" height="1000" type="application/pdf"></iframe>'

    # Displaying File
    st.markdown(pdf_display, unsafe_allow_html=True)

def view_certificate(certificate_id):
    # Smart Contract Call
    result = contract.functions.getCertificate(certificate_id).call()
    ipfs_hash = result[4]

    pinata_gateway_base_url = 'https://gateway.pinata.cloud/ipfs'
    content_url = f"{pinata_gateway_base_url}/{ipfs_hash}"
    response = requests.get(content_url)
    with open("temp.pdf", 'wb') as pdf_file:
        pdf_file.write(response.content)
    displayPDF("temp.pdf")
    os.remove("temp.pdf")

def hide_icons():
    hide_st_style = """
                <style>
                #MainMenu {visibility: hidden;}
                footer {visibility: hidden;}
                </style>"""
    st.markdown(hide_st_style, unsafe_allow_html=True)

def hide_sidebar():
    no_sidebar_style = """
                <style>
                div[data-testid="stSidebarNav"] {visibility: hidden;}
                </style>"""
    st.markdown(no_sidebar_style, unsafe_allow_html=True)
```

```

def remove_whitespaces():
    st.markdown("""
        <style>
               .css-18e3th9 {
                    padding-top: 0rem;
                    padding-bottom: 10rem;
                    padding-left: 5rem;
                    padding-right: 5rem;
                }
               .css-1d391kg {
                    padding-top: 3.5rem;
                    padding-right: 1rem;
                    padding-bottom: 3.5rem;
                    padding-left: 1rem;
                }
        </style>""", unsafe_allow_html=True)


```

5.5 Key Implementation Features

The system implements several important features:

1. **Dual User Roles:** Institute users can generate certificates while verifier users can verify them

2. **Certificate Generation:** Creates PDF certificates with unique identifiers

3. **Decentralized Storage:** Utilizes IPFS via Pinata for tamper-proof certificate storage

4. **Blockchain Verification:** Records certificate metadata on Ethereum for immutable verification

5. **Multiple Verification Methods:** Supports verification via both certificate ID and PDF upload

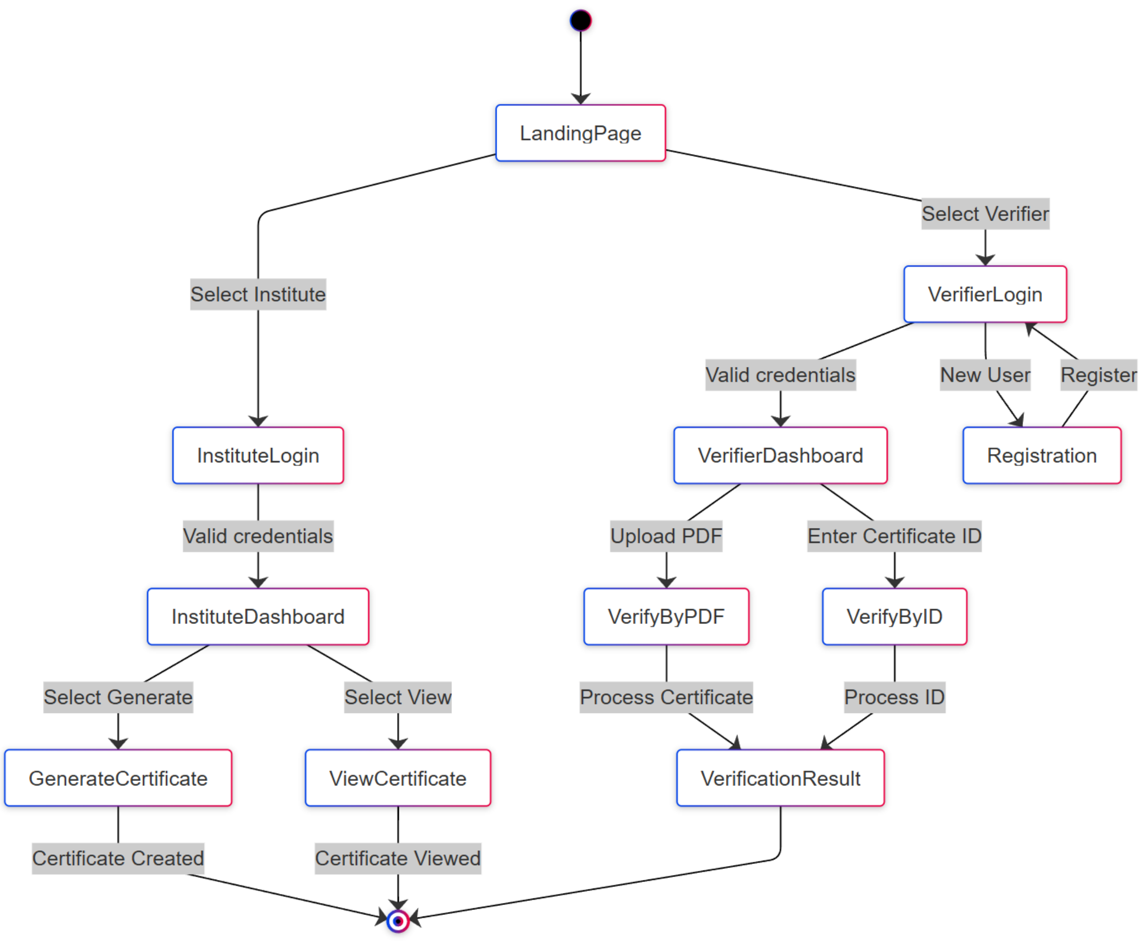
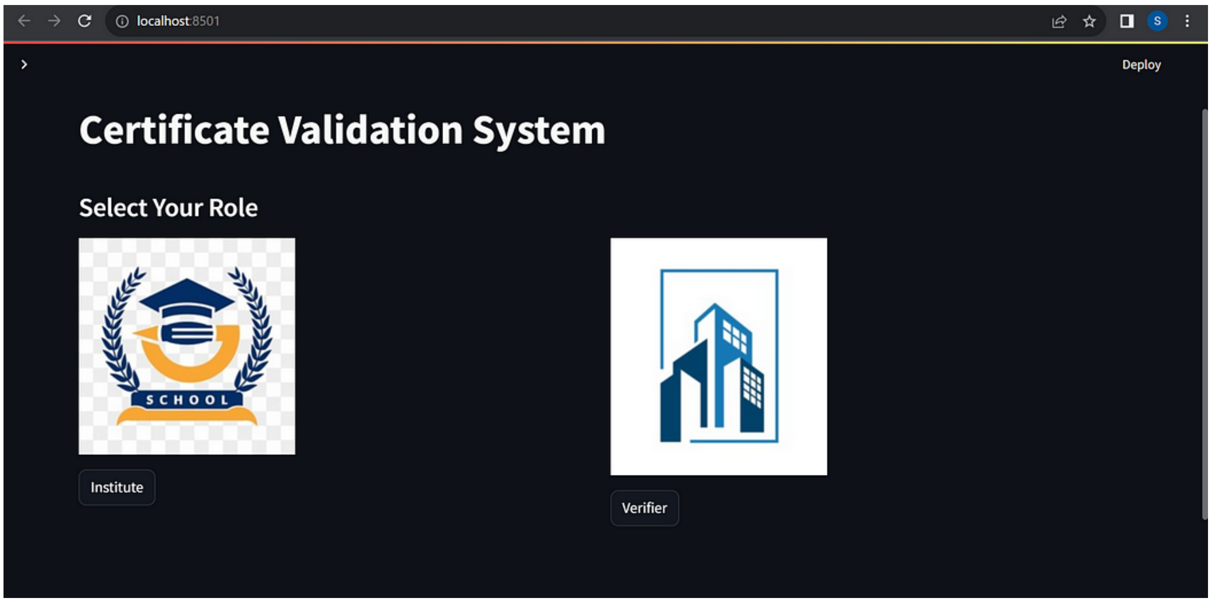


Fig 5.5.1 User Flow

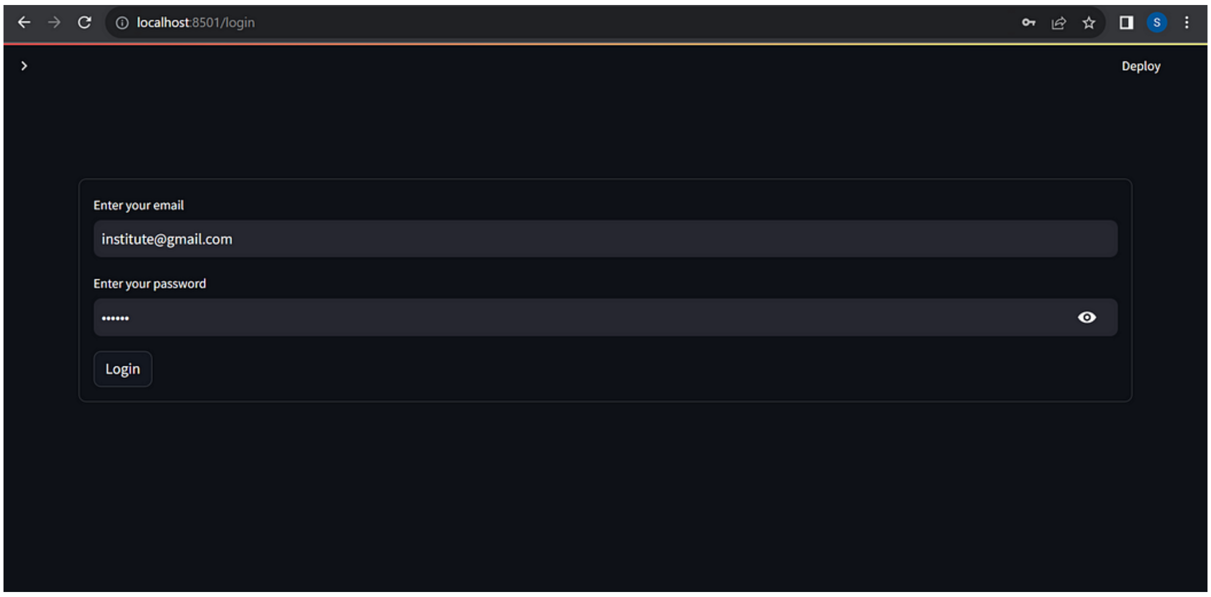


Certificate Validation System

Select Your Role

Institute

Verifier

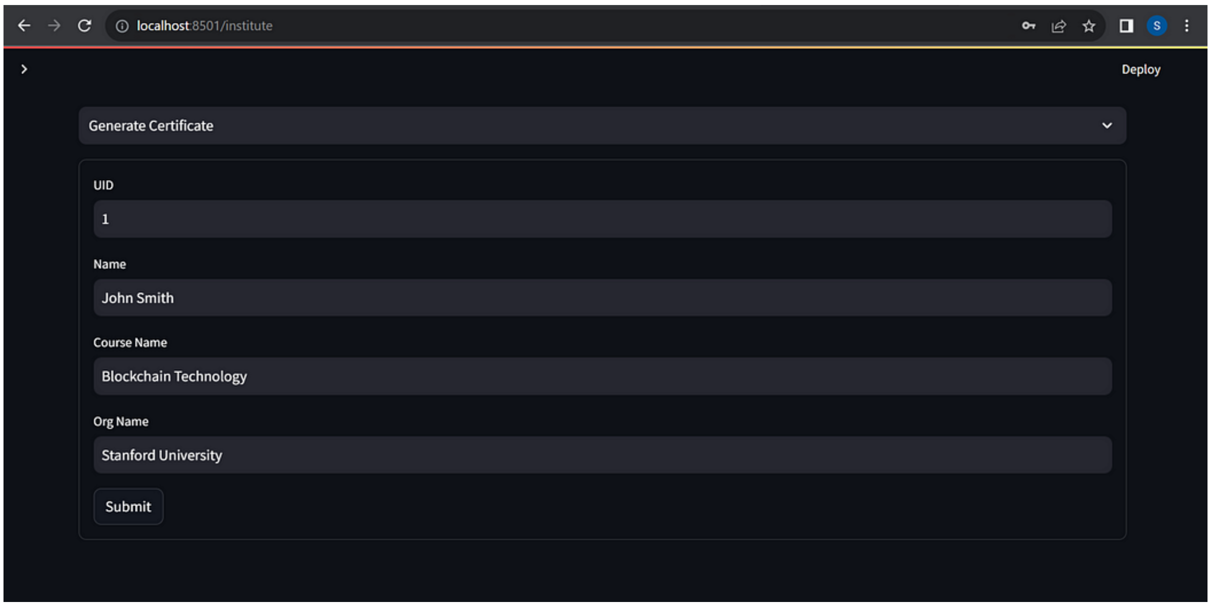


Enter your email

institute@gmail.com

Enter your password

••••••



UID

1

Name

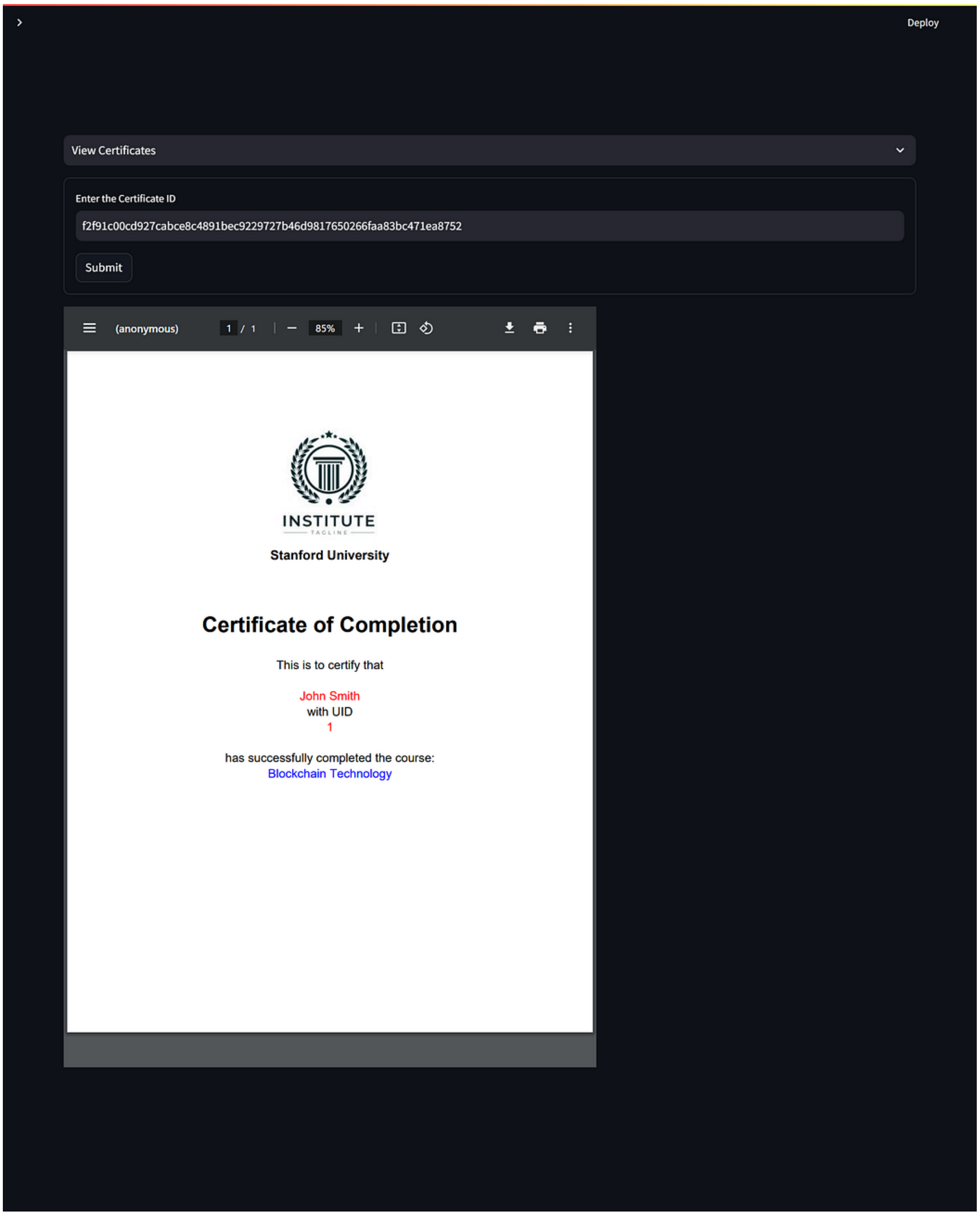
John Smith

Course Name

Blockchain Technology

Org Name

Stanford University



Enter the Certificate ID

f2f91c00cd927cabce8c4891bec9229727b46d9817650266faa83bc471ea8752

INSTITUTE

TAGLINE

Stanford University

Certificate of Completion

This is to certify that

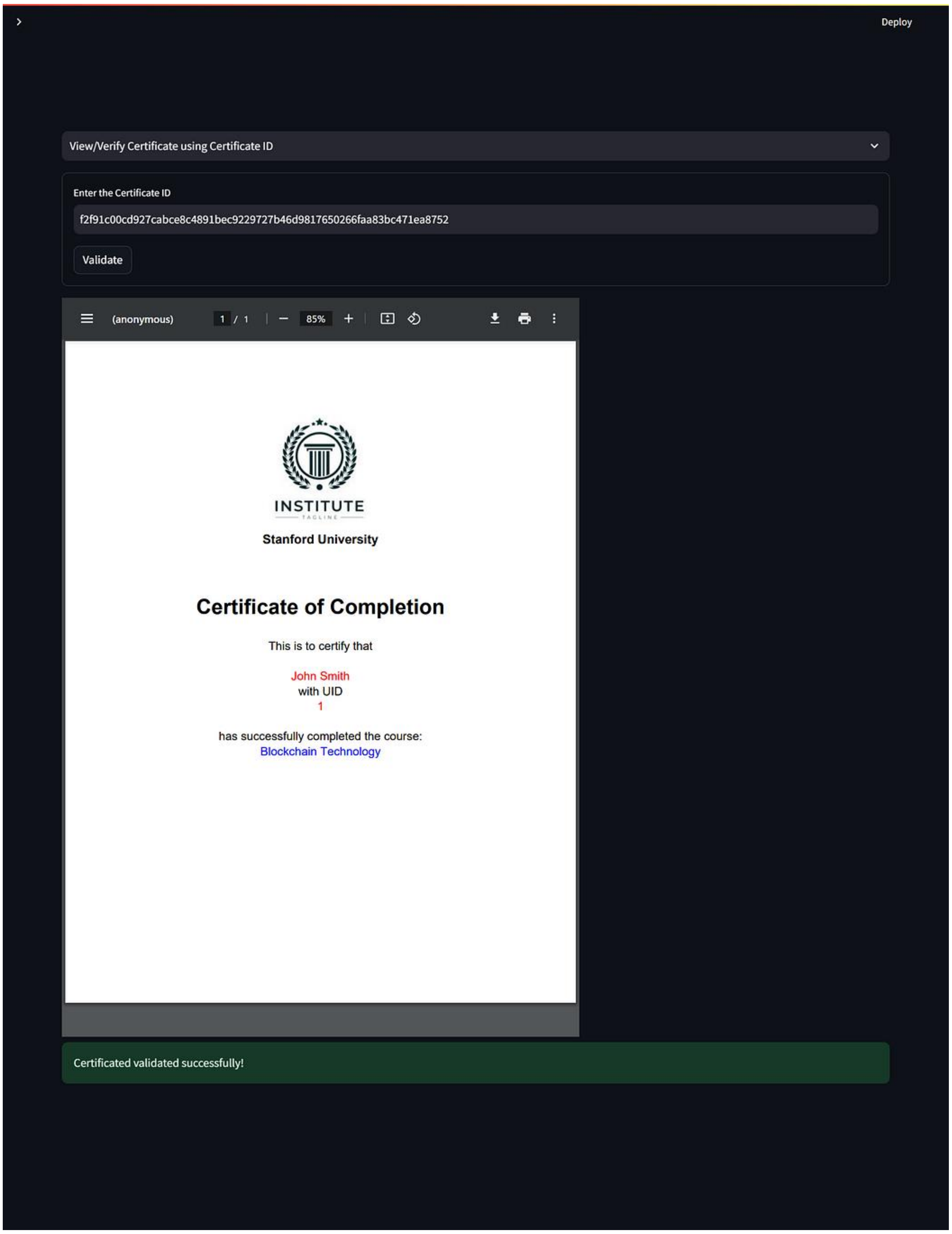
John Smith

with UID

1

has successfully completed the course:

Blockchain Technology



View/Verify Certificate using Certificate ID

Enter the Certificate ID

f2f91c00cd927cabce8c4891bec9229727b46d9817650266faa83bc471ea8752

Validate

INSTITUTE

TAGLINE

Stanford University

Certificate of Completion

This is to certify that

John Smith
with UID
1

has successfully completed the course:
Blockchain Technology

Certificated validated successfully!

CHAPTER 6:

SYSTEM TESTING AND QUALITY ASSURANCE

6.1 Technical Feasibility Analysis

The blockchain-based certificate verification system's technical feasibility is strongly supported by the inherent characteristics of blockchain technology and our carefully selected technology stack. Blockchain's immutable and decentralized nature provides an ideal foundation for certificate verification, effectively eliminating fraud and tampering risks while enhancing transparency and efficiency.

Our technology stack—Ethereum, IPFS, Streamlit, Pinata, Ganache-cli, Truffle, and Docker—creates a synergistic ecosystem where each component addresses specific requirements. While potential challenges exist, particularly around Ethereum's scalability and smart contract security, these can be effectively managed through proper optimization and rigorous testing.

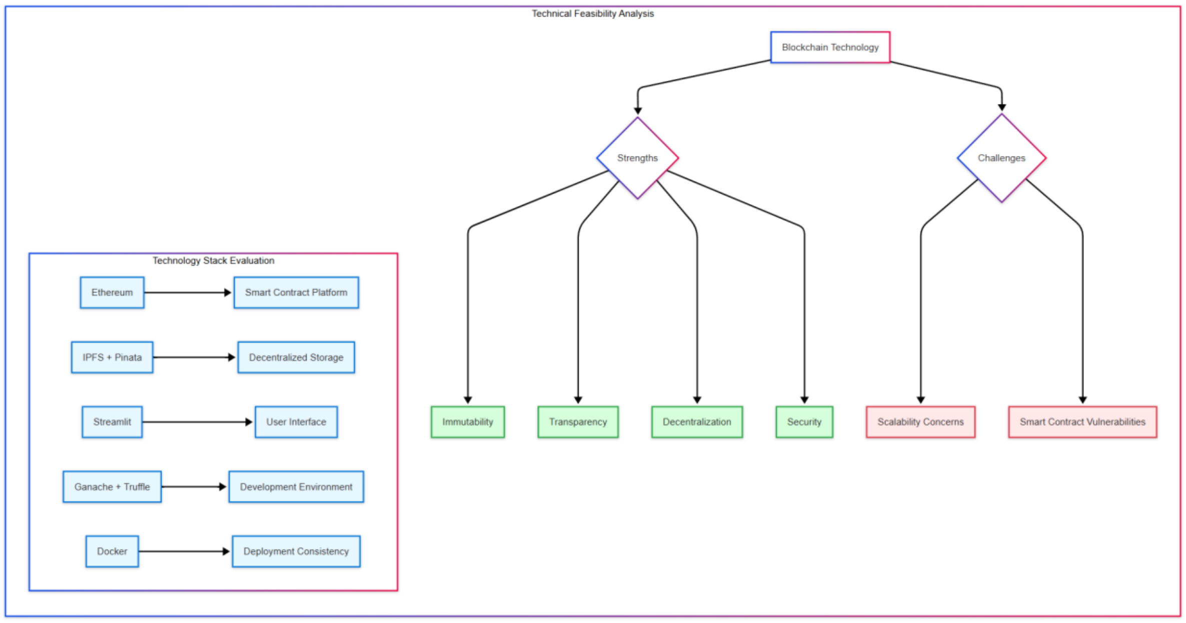


Fig 6.1.1 Technology Stack Feasibility Assessment

6.2 Testing Approaches

A comprehensive testing strategy encompassing multiple approaches ensures the reliability, security, and functionality of our DApp. Our testing methodology follows industry best practices specifically adapted for blockchain applications.

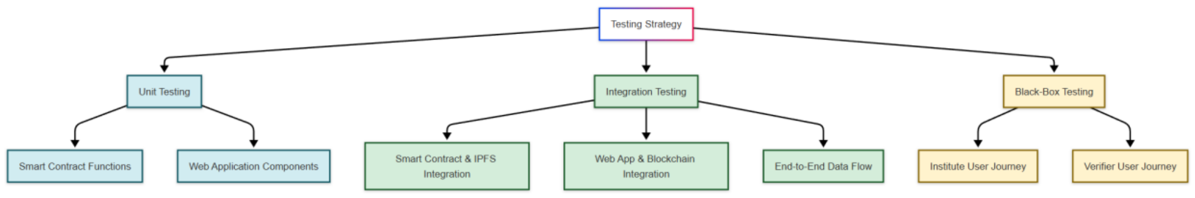


Fig 6.2.1 Testing Methodology Hierarchy

Unit Testing

Unit testing focuses on individual components, particularly the smart contract functions. Given the immutability of deployed contracts, thorough unit testing is critical to identify and eliminate logical errors or vulnerabilities before deployment. Each contract function is tested with various inputs including edge cases to ensure correct behaviour.

Integration Testing

Integration testing verifies interactions between system components—the Certification smart contract, IPFS storage via Pinata, and the Streamlit web application. Tests ensure that data flows correctly from certificate generation through blockchain storage to verification, confirming that all components work together seamlessly.

Black-Box Testing

Black-box testing evaluates functionality from an end-user perspective, focusing on usability and user experience for both Institute and Verifier roles. Using Streamlit's App Test framework, we can automate tests that simulate user interactions with the web interface to ensure intuitive operation and requirement fulfillment.

6.3 Testing Scenarios and Results

A comprehensive testing approach was implemented to ensure system reliability, security, and functionality across all components. The testing strategy encompassed unit testing for smart contracts, integration testing between system components, and black-box testing for the web application interface.

6.3.1 Unit Testing Results

Unit tests were conducted on the Certification smart contract to verify the correctness of individual functions. All critical functions passed their respective test cases, confirming the contract's functional integrity before deployment.

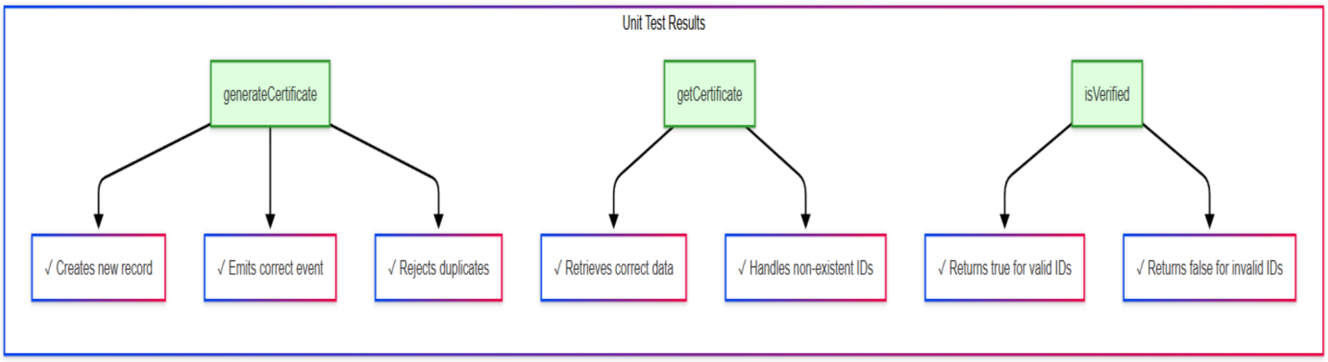


Fig 6.3.1 Smart Contract Unit Tests

6.3.2 Integration Testing Results

Integration tests verified the interaction between system components, particularly focusing on data flow between the web application, blockchain, and IPFS storage.

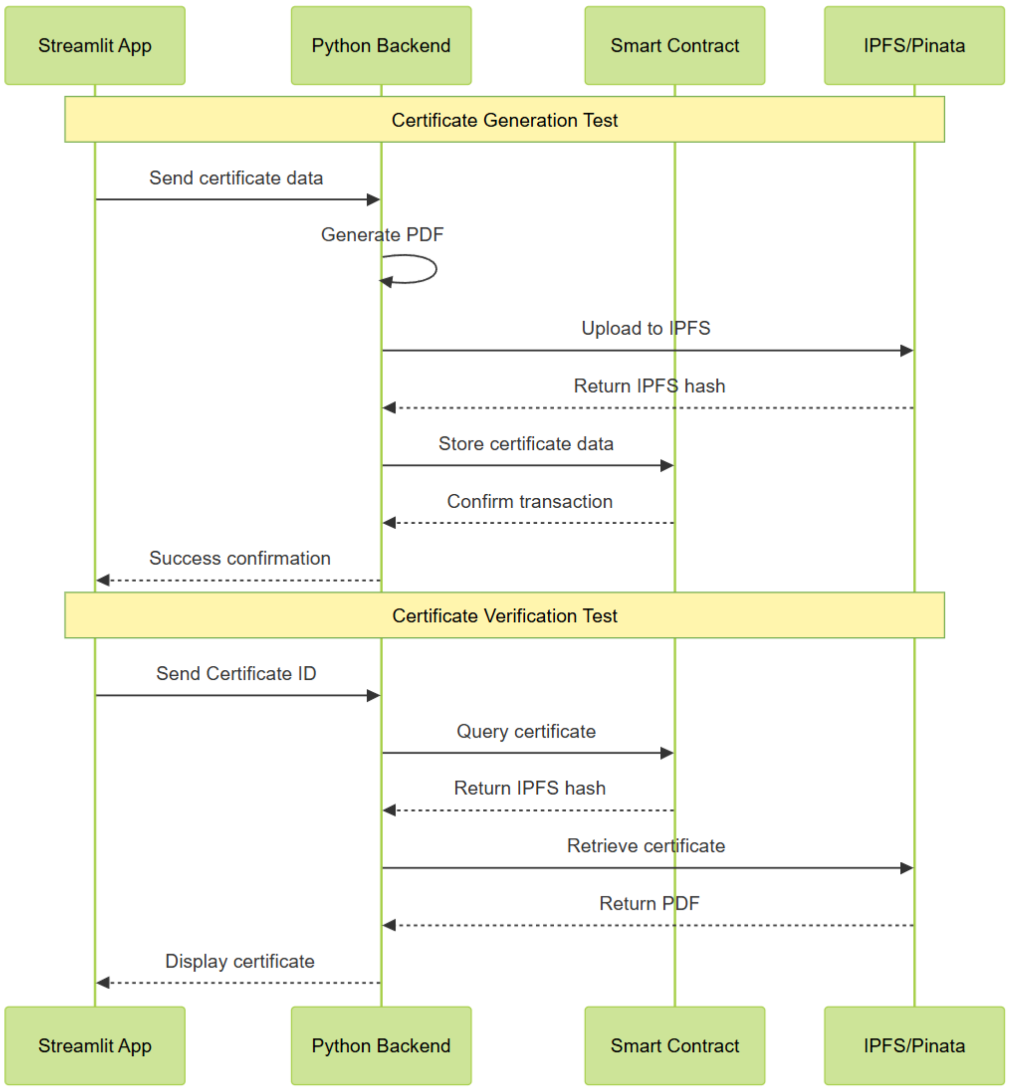


Fig 6.3.2 Integration Testing Flow

Test results confirmed that:

- Certificate data correctly flows from the web interface to the blockchain
- Certificate PDFs are properly stored and retrieved from IPFS
- Certificate verification works through both PDF upload and Certificate ID methods

6.3.3 Black-Box Testing Results

Black-box tests evaluated the system from an end-user perspective, focusing on usability and functionality of the web application.

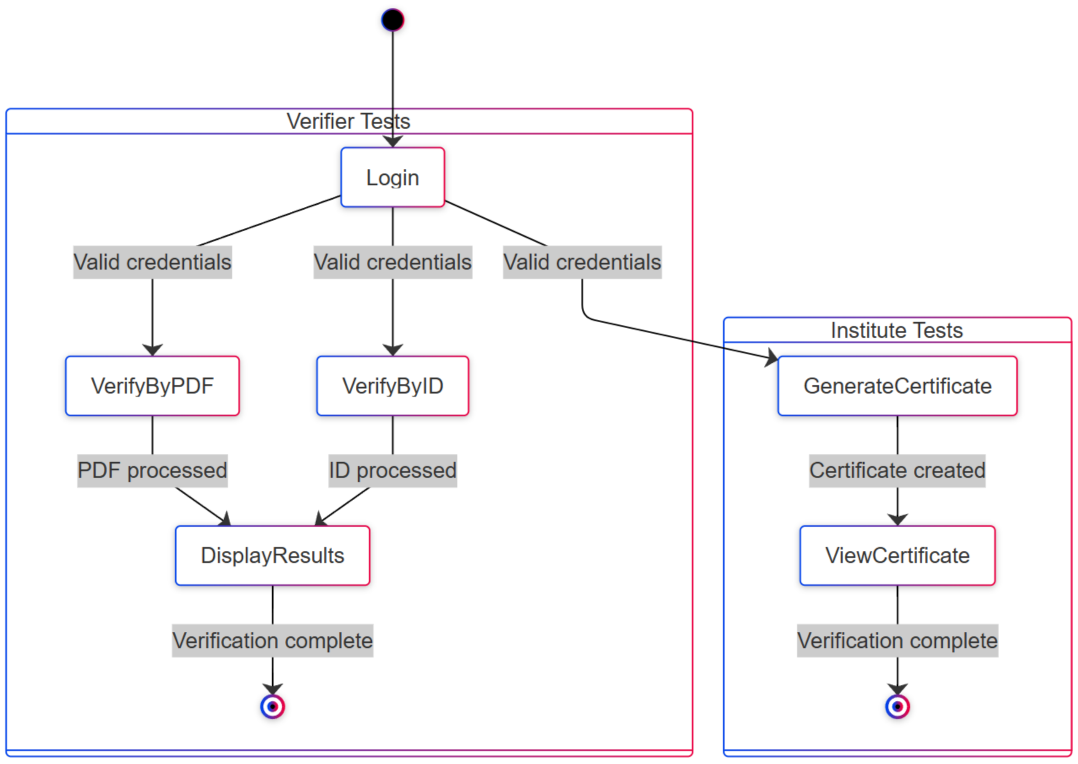


Fig 6.3.3 User Workflow Tests

All major user workflows were tested for both Institute and Verifier roles:

- Login/authentication functions correctly with proper error handling
- Certificate generation process completes successfully with appropriate feedback
- Certificate verification through both methods (PDF upload and Certificate ID) correctly identifies valid and invalid certificates
- Interface provides clear feedback on certificate status across all operations

6.3.4 Test Summary

The testing results confirmed the system's reliability, functionality, and security across all components. Minor issues identified during testing were addressed and resolved, resulting in a stable application ready for deployment. The comprehensive testing approach ensured that the blockchain-based certificate verification system meets its design goals and provides a trustworthy platform for certificate issuance and verification.

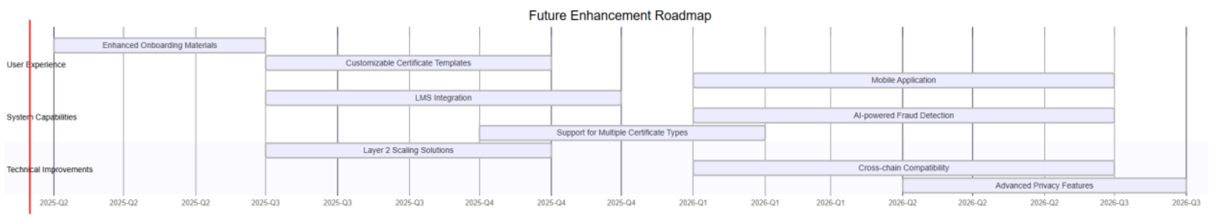
The test-driven development approach significantly reduced the risk of vulnerabilities in the smart contracts, ensuring that the immutable blockchain records maintain their integrity and security throughout the system's lifecycle.

CHAPTER 7:

FUTURE SCOPE AND CONCLUSION

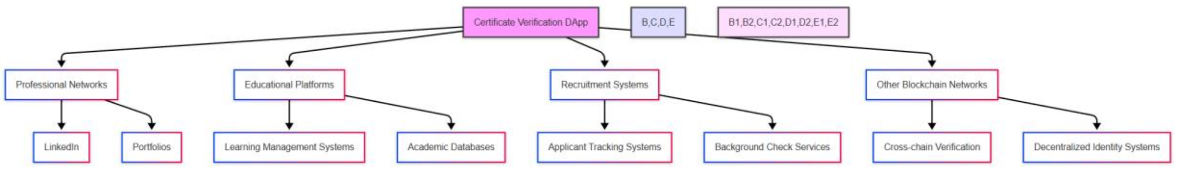
7.1 Potential Future Enhancements

The blockchain-based certificate verification system provides a solid foundation that can be expanded in several directions to increase its utility and effectiveness.



7.2 Integration with External Platforms

The system's utility can be greatly enhanced through strategic integrations with existing platforms and services.


Certificate Verification DApp
B,C,D,E
B1,B2,C1,C2,D1,D2,E1,E2
Professional Networks
Educational Platforms
Recruitment Systems
Other Blockchain Networks
LinkedIn
Portfolios
Learning Management Systems
Academic Databases
Applicant Tracking Systems
Background Check Services
Cross-chain Verification
Decentralized Identity Systems

Key integration opportunities include:

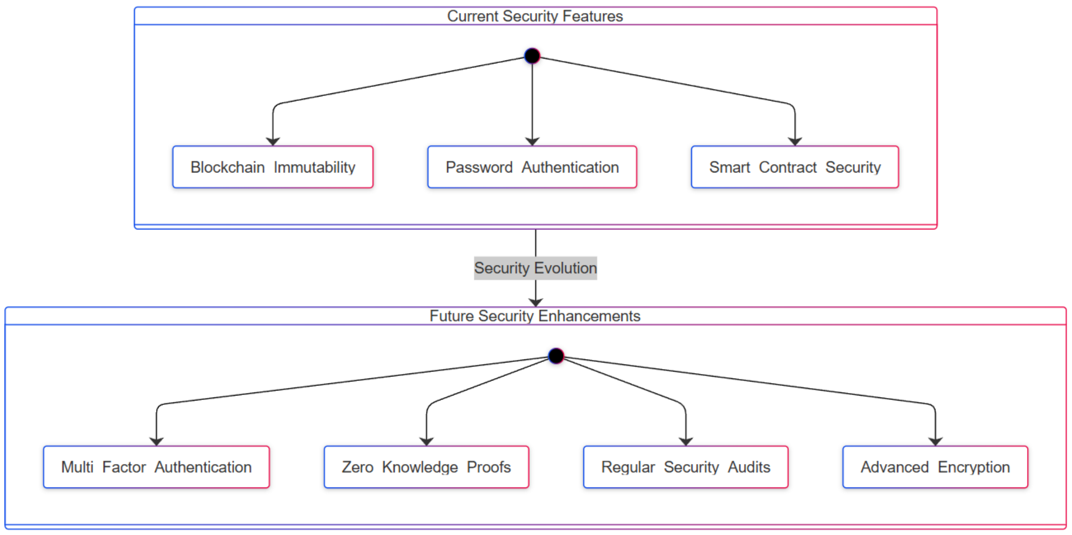
1. **Professional Networks**: Integration with LinkedIn would allow users to display verified credentials directly on their profiles.
2. **Cross-Platform Verification**: Implementing interoperability with other blockchain networks to create a unified verification ecosystem.
3. **Recruitment Systems**: Connecting with applicant tracking systems to streamline credential verification during hiring processes.

7.3 Advanced Security Features

While blockchain provides inherent security benefits, additional measures can further enhance the system's protection against threats.

Proposed security enhancements include:

1. **Multi-Factor Authentication**: Adding additional verification steps beyond passwords for both Institute and Verifier accounts.
2. **Zero-Knowledge Proofs**: Allowing certificate verification without revealing all certificate data, enhancing privacy.
3. **Regular Security Audits**: Implementing scheduled third-party security reviews to identify potential vulnerabilities.



7.4 Scalability Considerations

As system adoption grows, technical optimizations will be essential to maintain performance and cost-effectiveness.

Key scalability approaches include:

1. Layer 2 Solutions: Implementing Ethereum scaling technologies like rollups to reduce transaction costs and increase throughput.

2. Smart Contract Optimization: Refining contract code for gas efficiency to minimize operational costs.
3. Performance Monitoring: Establishing systems to track usage patterns and identify bottlenecks before they impact users.

7.5 Conclusion

The blockchain-based certificate verification system successfully addresses the challenges of traditional credential management by providing:

1. **Enhanced Security**: Utilizing blockchain's immutability to prevent certificate forgery and tampering.
2. **Simplified Verification**: Offering straightforward processes for authenticating credentials without intermediaries.
3. **Decentralized Storage**: Ensuring certificate availability through IPFS distributed file storage.
4. **User-Friendly Interface**: Making advanced technology accessible through an intuitive Streamlit application.

This project demonstrates the practical application of blockchain technology to solve real-world problems in credential verification. By creating a transparent, efficient, and secure platform for managing digital certificates, the system contributes to building trust in educational and professional credentials while reducing administrative overhead.

The modular design of the system provides a solid foundation for future enhancements, integration with other platforms, and adaptation to evolving security and scalability requirements. As blockchain technology continues to mature, this application stands as a tangible example of its potential beyond cryptocurrencies, offering genuine utility in document verification and trust establishment.

CHAPTER 8:

REFERENCE

- https://id4d.worldbank.org/guide/digital-certificates-and-pki
- https://www.adobe.com/acrobat/business/resources/digital-certificate.html
- https://www.digicert.com/faq/trust-and-pki/what-is-a-digital-certificate-and-why-are-digital-certificates-important
- https://www.okta.com/identity-101/digital-certificate/
- https://www.globalsign.com/en/blog/sg/five-beneficial-features-of-digital-certificates
- https://veridoccertificates.com/faq/what-are-the-differences-between-paper-based-and-digital-certificates
- https://www.chaincodeconsulting.com/insights/blog/transforming-credentials-blockchain-based-digital-certificates
- https://sertifier.com/blog/do-you-need-a-digital-diploma/
- https://www.irjmets.com/uploadedfiles/paper/issue_5_may_2023/40154/final/fin_irjmets1685264188.pdf
- https://diplomasafe.com/traditional-vs-digital-certificate/
- https://www.parchment.com/blog/enhancing-learner-success-with-digital-certificates-in-trade-and-vocational-schools/
- https://www.adobe.com/acrobat/business/resources/digital-certificate.html
- https://diplomasafe.com/traditional-vs-digital-certificate/
- https://pinata.cloud/blog/what-is-ipfs-used-for/
- https://filebase.com/blog/5-ipfs-use-cases-you-havent-thought-of-yet/
- https://ipfs.tech/
- https://archive.trufflesuite.com/docs/ganache/reference/cli-options/
- https://github.com/clearmatics/ganache-cli
- https://ethereum.stackexchange.com/questions/78794/creating-workspace-kind-in-ganache-cli

- https://www.truff.com/blogs/the-sauce/what-are-truffles-used-for
- https://www.graalvm.org/latest/graalvm-as-a-platform/language-implementation-framework/
- https://www.softobotics.com/blogs/truffle-a-comprehensive-guide-to-smart-contract-development-on-the-blockchain/
- https://myscale.com/blog/mastering-streamlit-use-cases-ultimate-guide/
- https://techifysolutions.com/blog/introduction-to-streamlit/