-0A PROJECT REPORT ON

LP-III

BLOCKCHAIN TECHNOLOGY

“Develop a Blockchain based application for health related medical records”

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

BACHELOR OF ENGINEERING

COMPUTER ENGINEERING

SUBMITTED BY

Student name: Aarti Shivaji Sathe Exam No: B1905504316

Under The Guidance of

Guide Name

DEPARTMENT OF COMPUTER ENGINEERING
(NBA ACCREDITED)

NUTAN MAHARASHTRA INSTITUTE OF ENGINNERING TECHNOLOGY

TALEGAON DABHADE TAL.MAVAL, PUNE, 410507

SAVITRIBAI PHULE PUNE UNIVERSITY

2024-2025

CERTIFICATE

This is to certify that the Project Entitled

“Develop a Blockchain based application for health related medical records”

Submitted by

Student name: Aarti Shivaji Sathe Exam No: B1905504316

is a bonafide student of this institute and the work has been carried out by him/her under the supervision of **Prof. Pritam Ahire** and it is approved for the partial fulfillment of the requirement of **Savitribai Phule Pune University**, for the award of the degree of Bachelor of Engineering in Computer Engineering.

Prof. Pritam Ahire	Dr. Prasad Dhore	
Project Guide	**H.O.D.**	**Principal**
Dept. of Computer Engg.	**Dept. of Computer Engg.**	**NMIET Pune**

External Examiner
Sign

Nutan Maharashtra Institute of Engineering and Technology, Pune – 07.

Place: Pune

Date:

Acknowledgments

It gives us great pleasure in presenting the preliminary LP-III project report on ***'Develop a Blockchain based application for health related medical records".***

I would like to take this opportunity to thank my internal project guide ***Prof. Pritam Ahire*** *for his able guidance and support in completing this project.*

I am also grateful to ***Dr. Prasad Dhore****, Head of Computer Engineering Department, Nutan Maharashtra institute of Engineering and Technology, Pune for his indispensable support, suggestions.*

In the end our special thanks to ***Principal****, Nutan Maharashtra instituteof Engineering and Technology, Pune for providing healthy environment, resources such as laboratory with all needed software platforms, tools etc... which helped us to successfullycomplete our project.*

Also, I would like to take this opportunity to thank my family members & supporters, withoutthem it could not have been done effectively in such a short period of time. I cannot forget their love & support.

Aarti Sathe(B1905504316)

Student Name (Exam Seat No.)

(B.E. Computer Engineering)

CONTENT

1.INTRODUCTION AND AIMS/MOTIVATION AND OBJECTIVES

Creating a blockchain-based application for health-related medical records involves several steps, including design, development, and deployment. Below is a structured approach that outlines the architecture, components, and implementation steps for such an application.

Key Features

1. Patient Data Management:
   - Secure storage of medical records (e.g., patient history, lab results, prescriptions).
   - Ability to update records by authorized healthcare providers.

2. Access Control:
   - Role-based access control to ensure only authorized users can view or modify records.
   - Patients can grant or revoke access to their medical records.

3. Audit Trail:
   - Immutable record of all access and changes made to medical records.
   - Traceability of data modifications for accountability.

4. Interoperability:
   - Ability to integrate with existing health systems and databases.
   - Standardized data formats (e.g., HL7, FHIR) for compatibility.

5. Data Privacy:
   - Encryption of sensitive data both in transit and at rest.
   - Use of pseudonymous identifiers to protect patient identities.

Architecture

The architecture consists of several components:

1. Blockchain Layer:
   - Blockchain Network: A permissioned blockchain network (e.g., Hyperledger Fabric, Ethereum) to store medical records securely.
   - Smart Contracts: Define rules for data access, sharing, and modification.

2. Web Application:
   - A user interface for patients, healthcare providers, and administrators.
   - Technologies: HTML, CSS, JavaScript (React, Angular, or Vue.js).

2. PROBLEM STATEMENT

When approaching a problem, especially in the context of data analysis and prediction, it’s essential to have a structured methodology. Here’s a comprehensive approach to solving problems like predicting stock prices using historical market data:

1. Define the Problem
- Identify the Objective: Clearly outline what you want to achieve. For instance, predicting future stock price returns based on past data.
- Scope: Determine the scope of the analysis (e.g., specific stocks, timeframes, market conditions).

2. Data Collection
- Gather Data: Collect historical stock price data, which may include:
- Daily opening, closing, high, and low prices.
- Trading volume.
- Macroeconomic indicators (inflation rate, GDP growth, etc.).
- Source: Use reliable data sources like stock exchanges, financial websites, or APIs (e.g., Yahoo Finance, Alpha Vantage).

3. Data Preprocessing
- Cleaning: Remove any missing or erroneous data points.
- Normalization: Scale the data if necessary, especially when dealing with different ranges of numerical values.
- Feature Engineering: Create additional features that might help in predictions, such as:
- Moving averages (e.g., 50-day, 200-day).
- Technical indicators (e.g., RSI, MACD).
- Lagged variables (previous days’ prices).

4. Exploratory Data Analysis (EDA)

3. PROJECT REQUIREMENTS

Developing a blockchain-based application for managing health-related medical records is an exciting project that can significantly enhance data security, patient privacy, and interoperability in the healthcare sector. Below is a comprehensive approach to implementing such a project:

1. Define the Problem

- Objective: Develop a secure, decentralized, and transparent system to store, manage, and share medical records, ensuring data privacy, immutability, and easy access for patients and authorized healthcare providers.
- Scope: Determine the scope of the application, such as whether it will focus on patient records, medical prescriptions, test results, or a combination of these. Also, decide whether to target specific healthcare sectors (hospitals, clinics) or be open to general use.

2. Stakeholder Analysis

- Patients: The primary owners of their medical data.
- Healthcare Providers: Hospitals, clinics, doctors, and other health professionals who need access to patient data.
- Insurers: Insurance companies that might need verified medical information for claims.
- Regulatory Bodies: Compliance with healthcare regulations like HIPAA (USA), GDPR (Europe), or local health data regulations is essential.

3. Technology Stack Selection

- Blockchain Platform: Choose an appropriate blockchain platform based on security, scalability, and cost considerations. Some popular platforms include:

- Ethereum: Offers smart contract functionality but has issues with scalability and gas fees.
- Hyperledger Fabric: Suitable for private blockchain networks where only authorized participants can access the data.
- Solana: A high-performance blockchain offering fast and scalable transactions.
- Polkadot: Offers interoperability between different blockchain networks.

4. Data Collection & Storage Design

- Data Types: Identify the types of medical records you want to store, such as:
  - Patient demographics (name, age, medical history).
  - Clinical data (diagnoses, medications, lab results).
  - Imaging data (X-rays, MRIs, etc.).
  - Insurance information.
- Off-Chain vs. On-Chain Storage:
  - Due to the large size of medical data, it’s not practical to store entire medical records on-chain.
  - On-Chain: Store references to the medical records (e.g., metadata, access controls, audit logs) to ensure transparency and immutability.
  - Off-Chain: Store the actual medical records on a secure off-chain storage system (e.g., IPFS, cloud storage with encryption) and reference these records from the blockchain.

5. Smart Contracts for Record Management

- Smart Contract Design: Create smart contracts that define the rules for accessing, updating, and sharing medical records. Key functionalities include:
  - Access Control: Only authorized entities (e.g., healthcare providers, patients) can access specific records.

1. LITERATURE SURVEY

A literature survey is an essential part of any project development process, as it provides insights into existing research, technologies, methodologies, and trends relevant to the topic. Below is a literature survey for developing a blockchain-based application for health-related medical records:

1. Introduction to Blockchain Technology

Blockchain technology, first introduced as the underlying architecture for Bitcoin in 2008 by Satoshi Nakamotocentralized, distributed ledger system that ensures immutability, security, and transparency. Each block in a blockchain contains a cryptographic hash of the previous block, ensuring a secure chain of data. Since blockchain enables data to be shared across multiple participants in a secure and verifiable way, it has gained significant attention for applications beyond cryptocurrency, including supply chains, healthcare, and identity management.

Key Literature:

- Nakamoto, S. (2008): "Bitcoin: A Peer-to-Peer Electronic Cash System"introduced the core concepts of blockchain technology, decentralization, and cryptographic consensus .
- Sw15): "Blockchain: Blueprint for a New Economy"explores blockchain beyond finance and highlights its potential in various industries, including healthcare .

2. Blockealthcare

Blockchain technology has garnered attention in healthcare due to its ability to address longstanding issues such as data fragmentation, security vulnerabilities, and lack of interoperability between healthcare systems. Literature emphasizes blockchain's potential in securely managing electronic health records (EHR), ensuring data integrity, and enhancing patient privacy and control.

5. SYSTEM ANALYSIS

System analysis is a critical phase in the development of any application, including a blockchain-based medical record system. The purpose of system analysis is to thoroughly understand the system requirements, functionalities, stakeholders, and any challenges or constraints that need to be addressed. Here's a detailed system analysis for a Blockchain-Based Health Record Management System:

1. Problem Definition

Managing medical records in healthcare is often problematic due to:

- Data Fragmentation: Medical records are often scattered across different healthcare providers, making it hard for patients and doctors to access a complete medical history.
- Data Security and Privacy: Storing sensitive patient data in centralized databases raises concerns about data breaches, unauthorized access, and potential misuse.
- Lack of Control by Patients: Patients typically have limited control over their own medical data.
- Interoperability: Healthcare providers often use different systems that do not communicate well with each other, making data sharing inefficient.
- Regulatory Compliance: Healthcare systems must comply with stringent privacy regulations like HIPAA (USA), GDPR (Europe), and other local laws.

2. System Objectives

The objectives of the blockchain-based medical record system are:

- Decentralization: Eliminate the need for a central authority by using blockchain for storing medical records and ensuring that no single entity has full control over the data.
- Patient Empowerment: Allow patients to fully control their medical data, including giving and revoking access to specific healthcare providers.
- Data Security: Use blockchain’s cryptographic features to ensure that medical data is stored securely and cannot be tampered with.

- Interoperability: Facilitate the seamless exchange of medical records between different healthcare institutions and providers.
- Compliance: Ensure that the system meets relevant privacy regulations, including the right to patient data privacy and security.

3. Stakeholders

- Patients: The primary users of the system, who will store and manage their medical data securely.
- Healthcare Providers: Doctors, hospitals, and clinics that will need access to patients' medical data for diagnosis, treatment, and follow-up care.
- Insurance Providers: May need verified medical records to process claims.
- Regulatory Bodies: Ensure that the system complies with legal requirements like HIPAA and GDPR.
- System Administrators: Responsible for maintaining the blockchain network, updating smart contracts, and ensuring the proper functioning of the system.

4. Functional Requirements

# 4.1. User Authentication and Authorization

- Patients should be able to register and log in securely to the system using secure authentication methods (e.g., multi-factor authentication).
- Healthcare providers should authenticate and request access to patient records based on the patient’s consent.

# 4.2. Medical Record Management

- Patients should be able to upload their medical records to the system.
- Patients should have full control to give or revoke access to their records to healthcare providers.

6. PROPOSED ARCHITECTURE

Proposed Architecture for Blockchain-Based Health Record Management System

The proposed architecture of the blockchain-based health record management system is designed to ensure security, privacy, interoperability, and scalability while giving patients full control over their medical data. This architecture combines blockchain technology with off-chain storage and encryption mechanisms to manage medical records efficiently.

1. Overview of the Proposed Architecture

The architecture is divided into several layers, each responsible for a distinct function. Below is an outline of the components and how they interact with each other:

- Blockchain Layer: Stores metadata and provides the logic for smart contracts, ensuring access control, data integrity, and auditability.
- Off-Chain Storage Layer: Stores large medical data securely and links it to blockchain records.
- Encryption Layer: Provides data security and privacy by encrypting medical records before storage and sharing.
- User Interface (UI) Layer: Interfaces through which patients and healthcare providers interact with the system.
- API Layer: Interfaces that allow external systems, such as hospital systems and insurance companies, to integrate with the blockchain network.

2. Layered Architecture Design

# 2.1. Blockchain Layer
The blockchain is the core of the system, handling medical data references, patient permissions, and access control.

- Blockchain Type: A permissioned blockchain like Hyperledger Fabric is preferred for this architecture to ensure only authorized participants (hospitals, clinics, patients) have access to the network.

- Smart Contracts:
  - Access Control Contracts: These smart contracts define rules for access to the medical records. Patients can grant or revoke access to healthcare providers.
  - Data Sharing Contracts: These govern how data is shared between different healthcare providers and institutions.
  - Audit Trail Contracts: Every interaction with the medical records (viewing, adding, or updating) is immutably logged on the blockchain.

- On-Chain Data:
  - Hash of Medical Records: Only a cryptographic hash of medical records is stored on-chain for verification purposes. The actual data resides off-chain.
  - Patient Permissions: Stores access permissions and consent given by the patient to various healthcare providers.
  - Audit Logs: Records of all activities performed on the data, such as data sharing or updates.

7. PROJECT PLAN

Project Plan for Blockchain-Based Health Record Management System

The project plan outlines the stages, timelines, key deliverables, and resources necessary for the successful development of the Blockchain-Based Health Record Management System. This system will leverage blockchain technology to secure, manage, and share medical records with an emphasis on patient control, data privacy, and compliance with healthcare regulations.

1. Project Scope

# Objective:
To build a decentralized health record management system using blockchain technology that enables secure storage, sharing, and management of medical records with patient-controlled access.

# Key Features:
- Patient-driven access control to medical records.
- Secure and decentralized storage of medical data using off-chain solutions.
- Immutable audit trails for all data access and modifications.
- Integration with existing healthcare systems (Electronic Health Records).
- Compliance with regulations (HIPAA, GDPR).

2. Project Phases and Tasks

# Phase 1: Project Initiation and Requirement Analysis

Duration: 4 weeks

Key Activities:

- Define the objectives and scope of the system.
- Conduct stakeholder meetings (patients, healthcare providers, IT teams) to gather system requirements.
- Identify regulatory requirements (e.g., HIPAA, GDPR).
- Create a Requirements Specification Document covering:
  - Functional requirements (e.g., patient record management, access control).
  - Non-functional requirements (e.g., security, scalability, compliance).
- Deliverables: Requirements specification document, initial project plan, and timeline approval.

# Phase 2: System Architecture and Design

Duration: 6 weeks

Key Activities:

- Design the overall system architecture, including:
  - Blockchain layer (smart contracts, access control mechanisms).
  - Off-chain storage (IPFS/cloud storage integration).
  - API layer for integration with EHR systems.
- Design the user interface (UI) for both patient and healthcare provider portals.
- Ensure the system complies with security and privacy regulations.
- Prepare a detailed technical design document outlining:
  - Data flow.
  - Smart contract structure.
  - Encryption strategies for sensitive medical data.

9.ALGORITHM, PROTOCOLS USED

Here’s a simplified pseudocode example for a smart contract that manages medical records:

~~~
```solidity
pragma solidity ^0.8.0;
contract MedicalRecords {
    struct Record {
        uint id;
        string patientName;
        string dataHash; // Hash of the medical record
        address doctor;
        bool exists;
    }
    mapping(uint => Record) public records;
    mapping(address => bool) public authorizedDoctors;

    event RecordAdded(uint id, string patientName, address doctor);
    event AccessGranted(address doctor);
    event AccessRevoked(address doctor);
    modifier onlyAuthorized() {
        require(authorizedDoctors[msg.sender], “Not authorized”);
        _;
    }
    function addRecord(uint _id, string memory _patientName, string memory _dataHash)
public onlyAuthorized {
        records[_id] = Record(_id, _patientName, _dataHash, msg.sender, true);
        emit RecordAdded(_id, _patientName, msg.sender);
    }

    function grantAccess(address _doctor) public {
        authorizedDoctors[_doctor] = true;
        emit AccessGranted(_doctor);
    }

    function revokeAccess(address _doctor) public {
        authorizedDoctors[_doctor] = false;
        emit AccessRevoked(_doctor);
    }
}
~~~

10. MATHEMATICAL MODEL

“Mathematical Models for Blockchain-Based Health Record Management System”

In a “blockchain-based health record management system”, several mathematical models and techniques are relevant for different aspects of system design, data security, performance optimization, and network stability. Below are key areas where mathematical models can be applied:

“1. Cryptographic Models”

“1.1. Hash Functions”
A “cryptographic hash function”ensures the integrity of medical records stored off-chain by generating a unique hash for each data record. The hash is stored on the blockchain and used to verify the integrity of the data.

- “Hash Function (SHA-256)”:
  - Let the medical record data be represented as \(D \).
  - The hash function \(H(x) \) transforms the data into a fixed-size hash:

  \[

  h = H(D)

  \]

  where \(h \) is a unique hash stored on-chain, and any change in \(D \) will result in a different hash \(h' \), thus providing data integrity verification.

“1.2. Asymmetric Encryption”
“Public key cryptography”ensures secure data sharing between patients and healthcare providers.

- “RSA Encryption”:
  - Data is encrypted using the recipient's public key \(K_{\text{pub}} \) and decrypted using their private key \(K_{\text{priv}} \).
  - The encryption of a medical record \(D \) is modeled as:
  \[
  C = E(K_{\text{pub}}, D)
  \]
  where \(C \) is the ciphertext that can only be decrypted by the owner of \(K_{\text{priv}} \):
  \[
  D = D(K_{\text{priv}}, C)
  \]

“1.3. Symmetric Encryption”
For encrypting large files (e.g., X-rays or MRI scans), “symmetric encryption”(like AES-256) is used.

- “AES-256 Encryption”:
  - Given a medical record \(D \) and a secret key \(K \), the encryption is modeled as:
  \[
  C = E(K, D)
  \]
  The ciphertext \(C \) can only be decrypted by the corresponding key \(K \), ensuring data confidentiality.

11.SYSTEM IMPLEMENTATION – CODE DOCUMENTATION

```
import hashlib
import json
from time import time
from flask import Flask, request, jsonify

class Blockchain:
    def __init__(self):
        self.chain = []
        self.current_records = []
        self.create_block(previous_hash='1', proof=100)

    def create_block(self, proof, previous_hash):
        block = {
            'index': len(self.chain) + 1,
            'timestamp': time(),
            'records': self.current_records,
            'previous_hash': previous_hash,
        }
        self.current_records = []
        self.chain.append(block)
        return block

    def add_record(self, patient_name, data_hash):
        self.current_records.append({
            'patient_name': patient_name,
            'data_hash': data_hash
        })
        return self.last_block['index'] + 1

    @property
    def last_block(self):
        return self.chain[-1]

    def hash(self, block):
        block_string = json.dumps(block, sort_keys=True).encode()
        return hashlib.sha256(block_string).hexdigest()

# Flask web application
app = Flask(__name__)

# Create a blockchain instance
```

```
blockchain = Blockchain()

@app.route('/add_record', methods=['POST'])
def add_record():
    data = request.get_json()
    patient_name = data['patient_name']
    data_hash = data['data_hash']

    block_index = blockchain.add_record(patient_name, data_hash)
    response = {
        'message': 'Record will be added to Block',
        'block_index': block_index,
    }
    return jsonify(response), 201

@app.route('/mine_block', methods=['GET'])
def mine_block():
    previous_block = blockchain.last_block
    previous_hash = blockchain.hash(previous_block)
    block = blockchain.create_block(proof=100, previous_hash=previous_hash)

    response = {
        'message': 'Block Mined!',
        'index': block['index'],
        'records': block['records'],
        'previous_hash': block['previous_hash'],
    }
    return jsonify(response), 200

@app.route('/chain', methods=['GET'])
def full_chain():
    response = {
        'chain': blockchain.chain,
        'length': len(blockchain.chain),
    }
    return jsonify(response), 200

if __name__ == '__main__':
    app.run(debug=True)
```

OUTPUT

```
{
   “chain”: [
      {
         “index”: 1,
         “previous_hash”: “1”,
         “records”: [],
         “timestamp”: 1692011551.123456
      },
      {
         “index”: 2,
         “previous_hash”: “some_previous_hash”,
         “records”: [
            {
               “patient_name”: “John Doe”,
               “data_hash”: “abc123hashvalue”
            }
         ],
         “timestamp”: 1692011552.654321
      }
   ],
   “length”: 2
}
```

12. CONCLUSIONS

In conclusion, developing a blockchain-based application for medical records addresses key healthcare challenges like “data security”, “privacy”, and “interoperability”. By granting patients ownership of their data while enabling healthcare providers to access and update records securely, blockchain technology ensures transparency and tamper-proof record-keeping.

With proper implementation, this system can lead to more efficient, secure, and trustworthy medical data management, ultimately improving patient care and healthcare system performance.

13.APPENDIX

a. **Tools used**

“ Tools Used for Developing a Blockchain-Based Health Records Application

1. “Blockchain Platform”:
   - “Ethereum”: A widely-used platform that supports smart contracts, which can be used to manage permissions and data access.
   - “Hyperledger Fabric”: A permissioned blockchain platform tailored for enterprise use, offering privacy and scalability.

2. “Smart Contracts”:
   - “Solidity”: A popular programming language for writing smart contracts on Ethereum.
   - “Chaincode”: Used in Hyperledger Fabric to define business logic and rules for blockchain transactions.

3. “Front-End Development”:
   - “React.js” or “Angular.js”: Popular frameworks for building user-friendly web interfaces.
   - “HTML5, CSS3, JavaScript”: Standard web development tools for structuring and styling the application.

4. “Backend Development”:
   - “Node.js”: A JavaScript runtime for developing the backend, facilitating interaction between the blockchain and the web application.
   - “Express.js”: A framework to manage API routes and requests.

5. “Databases (Optional for off-chain storage)”:

- “IPFS (InterPlanetary File System)”: For storing large medical records off-chain and linking to the blockchain using a hash.
- “MongoDB” or “PostgreSQL”: For off-chain data storage, if required, such as metadata or non-critical health data.

6. “APIs”:

- “Web3.js”: JavaScript library to interact with Ethereum blockchain.
- “Hyperledger SDK”: Provides API functionalities to interact with the Hyperledger blockchain.

7. “Encryption & Security”:

- “AES/RSA Encryption”: For securing data transmission and storage.
- “OAuth 2.0”: For secure authentication.

8. “Testing & Development”:

- “Truffle”: A development framework for Ethereum that helps in smart contract compilation, deployment, and testing.
- “Ganache”: Local Ethereum blockchain for testing purposes.
- “Postman”: API testing tool to validate data flow between frontend, backend, and blockchain.

9. “Cloud & DevOps Tools”:

- “AWS” or “Azure Blockchain Service”: For hosting blockchain nodes and related services.
- “Docker/Kubernetes”: For containerization and deployment of the application.
- “CI/CD Pipelines”: Automated deployment, testing, and monitoring of the application.

b. References

- ***Blockchain Platforms***:
- *Ethereum: https://ethereum.org/en/*
- *Hyperledger Fabric: https://www.hyperledger.org/use/fabric*
- ***Smart Contracts***:
- *Solidity Documentation: https://soliditylang.org/docs/*
- *Chaincode for Hyperledger: https://hyperledger-fabric.readthedocs.io/en/release-2.2/chaincode.html*
- ***IPFS***:
- *IPFS Documentation: https://docs.ipfs.io/*
- ***Web3.js***:
- *Web3.js Documentation: https://web3js.readthedocs.io/*
- ***React.js***:
- *React.js Documentation: https://reactjs.org/*
- ***Testing Frameworks***:
- *Truffle Framework: https://trufflesuite.com/*
- *Ganache: https://trufflesuite.com/ganache/*