

VETRI VINAYAHA COLLEGE OF ENGINEERING AND TECHNOLOGY



ANNA UNIVERSITY : CHENNAI
OCT -2023

SUBMITTED BY

PRAVEENA	R	623320104016
AARTHY	S	623320104001
BOOMIKA	V	623320104007
BARATH	S	623320104006

Guided by
MR.A.SURYA
AP/CSE

PROJECT TITLE

**ELECTRONIC HEALTH
RECORD**

DOMAIN: BLOCK CHAIN TECHNOLOGY

1.1 INTRODUCTION

1.1 PROJECT OVERVIEW:

The digitization of healthcare has ushered in a new era of patient record management, with Electronic Health Records (EHRs) at the forefront of this transformation. EHRs offer numerous advantages, such as improved accessibility, efficiency, and the potential for real-time data analysis, enhancing the quality of patient care. However, the adoption of EHRs has also exposed significant challenges, including data security, privacy concerns, and interoperability issues.

The sensitive and personal nature of healthcare data demands the highest standards of security and integrity. Data breaches, unauthorized access, and tampering with patient records can have profound consequences on patient trust and healthcare outcomes. Additionally, the siloed nature of EHR systems among different healthcare providers has hampered efficient data sharing and continuity of care.

1.2 PURPOSE:

The primary purpose of this project is to investigate the application of blockchain technology in Electronic Health Records (EHRs) with the aim of addressing critical issues in healthcare data management and security. The specific purposes include:

To assess how blockchain can improve the security of healthcare data by providing an immutable and decentralized ledger for storing EHRs. This technology has the potential to significantly reduce the risk of data breaches, unauthorized access, and data tampering.

2. LITERATURE SURVEY

2.1 Existing problem:

Traditional EHR systems typically rely on centralized data repositories managed by healthcare institutions, such as hospitals and clinics. Patient health records, including medical history, treatment plans, test results, and prescriptions, are stored in these centralized databases. HealthCare data is often soloed, meaning that patient records are stored in different formats and locations, making it challenging to share information across different healthcare providers and institutions.

This fragmentation can lead to inefficiencies, medical errors, and difficulties in providing continuity of care. While EHRs offer advantages in terms of digital accessibility and ease of data retrieval, they are susceptible to security breaches and unauthorized access.

2.2 References:

- Journals such as the "Journal of Medical Internet Research," "Health Informatics Journal," and "Blockchain in Healthcare Today" often publish research on the intersection of blockchain and healthcare.
- Consult resources from government health agencies like the U.S. Department of Health and Human Services (HHS) for insights into healthcare data management and regulations.
- Look for books on blockchain in healthcare or electronic health records in academic libraries or online retailers.
- Databases like PubMed, IEEE Xplore, Google Scholar, and ResearchGate can be valuable sources for academic papers, articles, and conference proceedings.

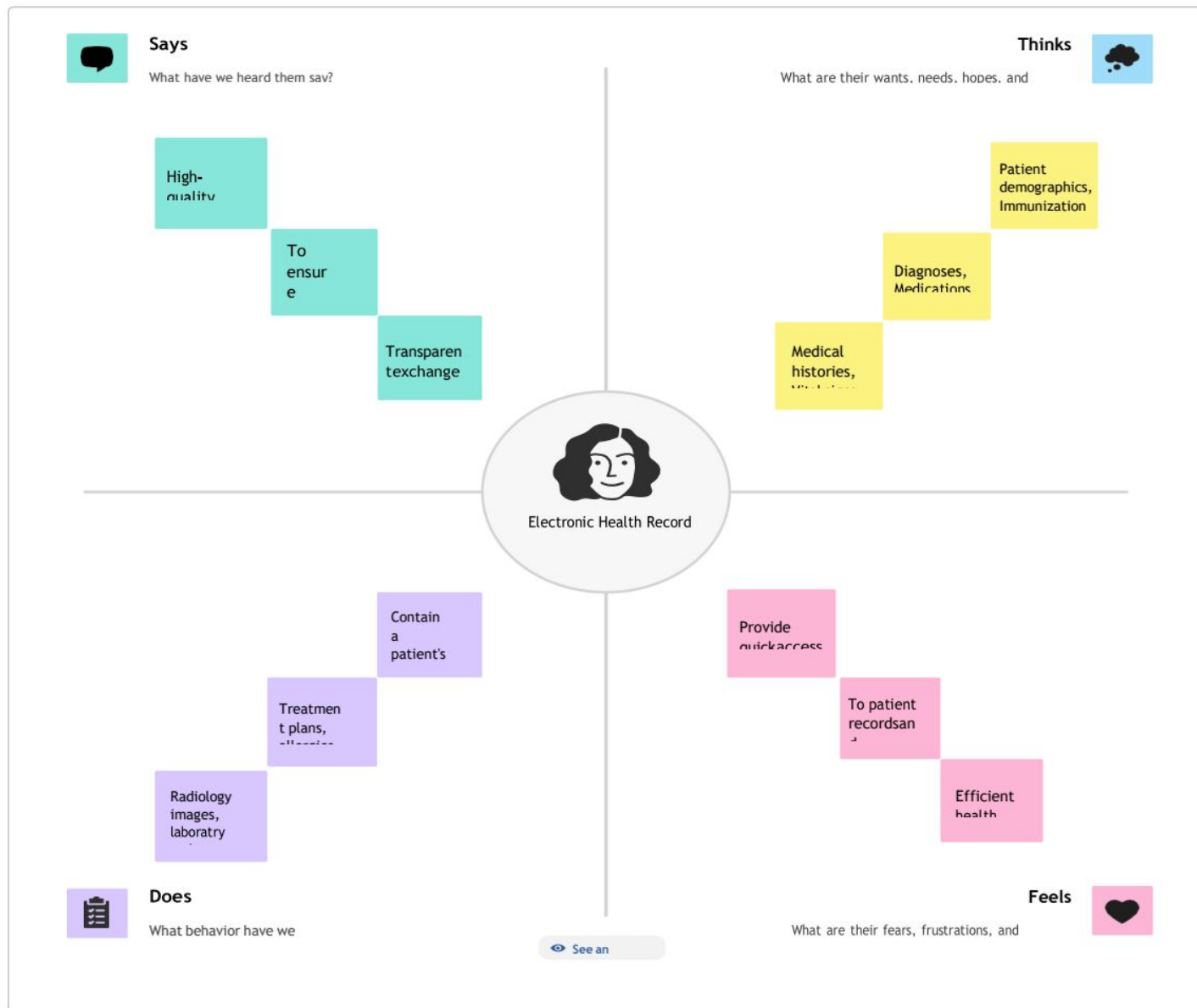
2.3 Problem Statement Definition:

A problem statement is a clear and concise description of an issue, challenge, or situation that needs to be addressed or resolved. It serves as a critical component in the early stages of problem-solving, research, or project development by providing a precise understanding of what the problem is and why it's important. A well-crafted problem statement typically includes the following elements This section outlines the problem's nature and context. It explains the specific issue or challenge that needs attention and provides a clear understanding of what is not working or what is causing concern.

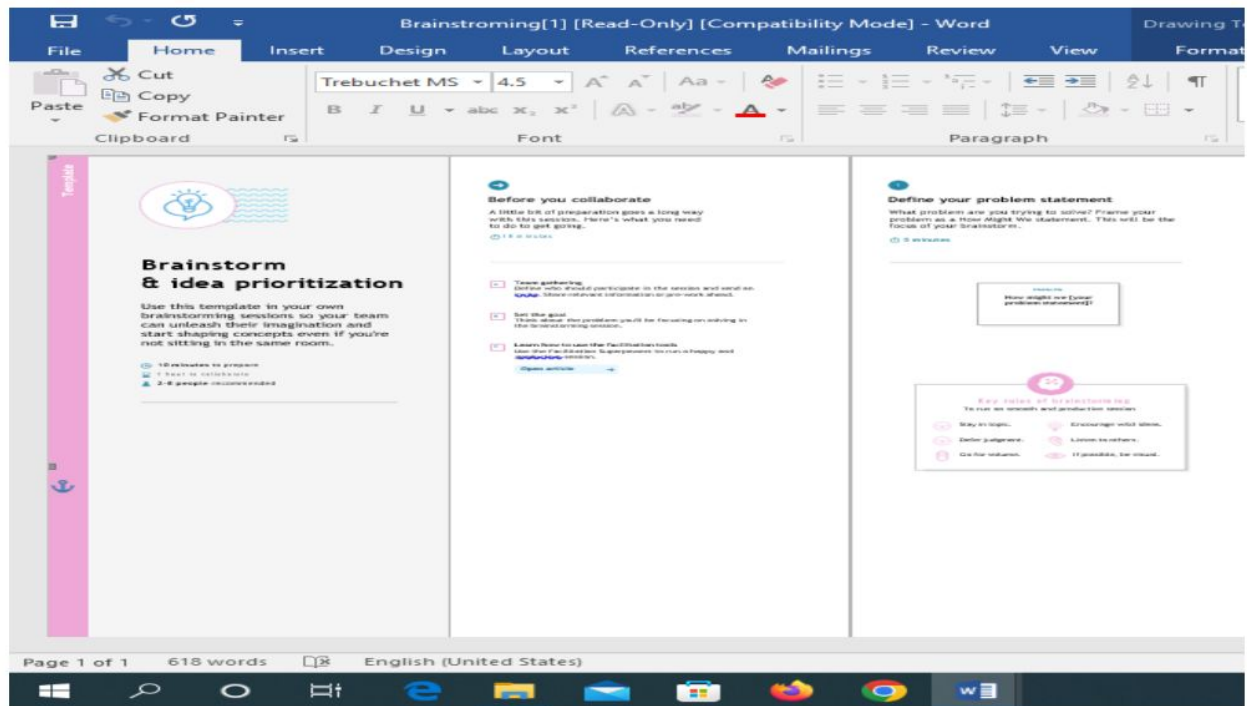
The problem statement should highlight why the problem is significant and what consequences or effects it has. This helps to convey the importance of addressing the problem. It's important to define the scope of the problem, which outlines the limits or boundaries of the issue being addressed. This helps in focusing the problem-solving efforts and avoiding overly broad statements. Identify the individuals, groups, or organizations that are affected by or have a stake in the problem. Understanding the stakeholders and their interests is crucial for finding a solution that considers all relevant perspectives.

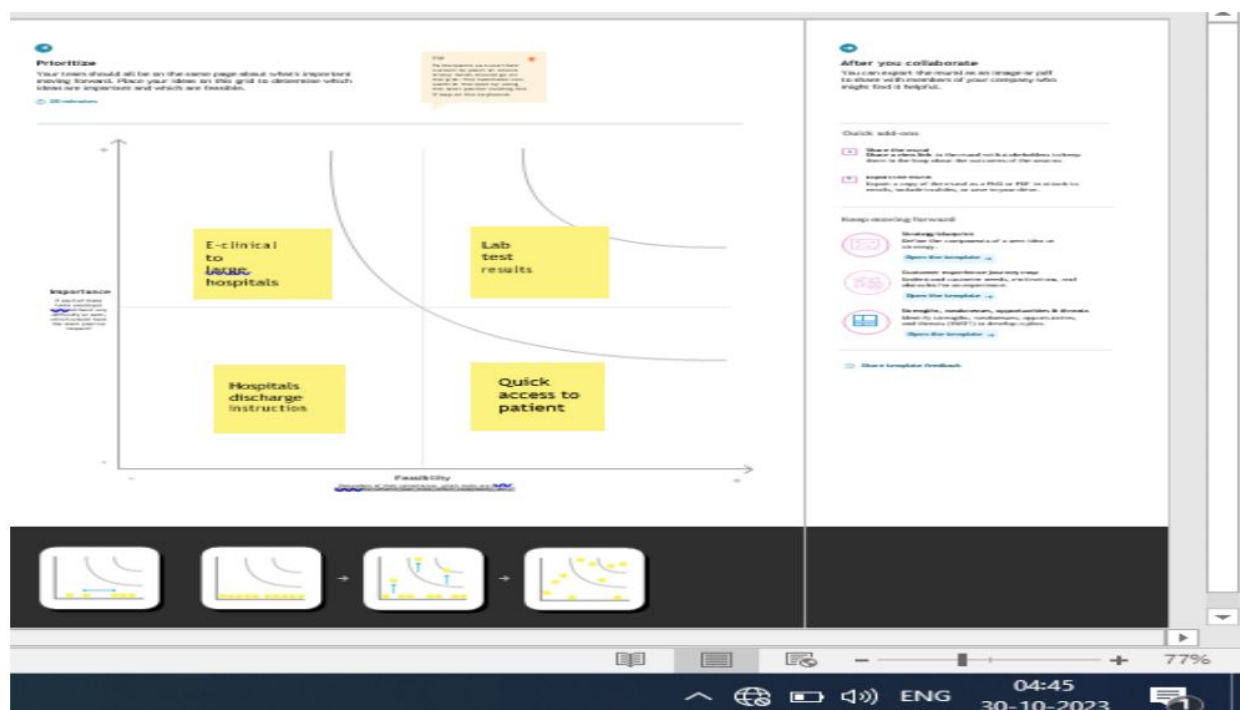
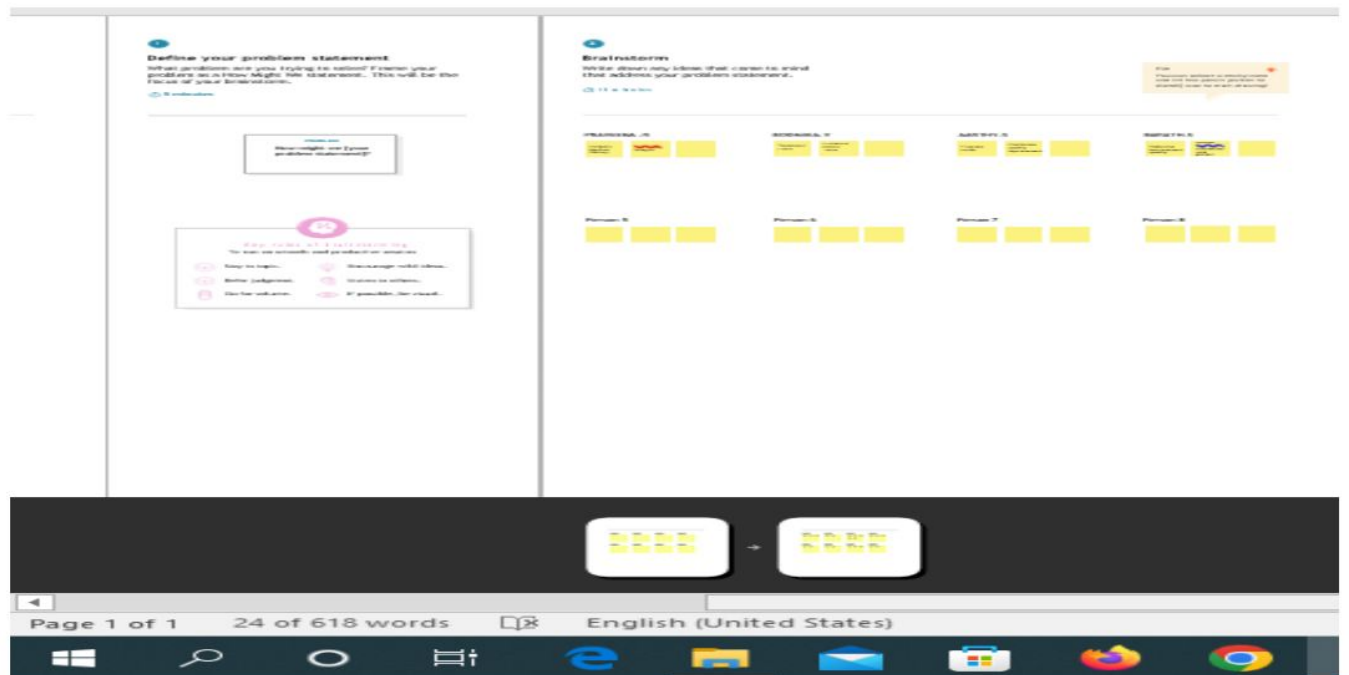
3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas:



3.2 Ideation & Brainstorming:





4. REQUIREMENT ANALYSIS

4.1 Functional requirement:

1. **System Features:** List the features that the system should provide. These features can be as simple as user authentication or as complex as data analytics tools, depending on the system's purpose.
2. **Use Cases:** Describe the various use cases or scenarios in which the system will be used. These are detailed descriptions of how users interact with the system to accomplish specific tasks.
3. **User Interfaces:** Specify the user interfaces, including the layout, design, and navigation. This includes details like buttons, menus, forms, and any user interactions.
4. **Data Handling:** Explain how the system should handle data. This involves data input, storage, retrieval, processing, and output.
5. **External Interfaces:** Describe any external systems or components that the system must interact with, including APIs, databases, and hardware.
6. **Performance Requirements:** Specify performance metrics, such as response times, throughput, and resource usage that the system must meet to be considered functional.
7. **Security Requirements:** Detail the security features and measures the system should have to protect data, user privacy, and system integrity.
8. **Scalability and Reliability:** Outline how the system should scale to accommodate increased loads or users and how it should ensure reliability and uptime.
9. **Error Handling:** Describe how the system should handle errors, exceptions, and edge cases. This may include error messages and recovery mechanisms.
10. **Reporting and Logging:** Specify the types of reports and logs the system should generate. This is important for monitoring and troubleshooting.
11. **User Roles and Permissions:** Define different user roles and their access levels. Determine what each role is allowed to do within the system.
12. **Regulatory Compliance:** If the system must adhere to specific regulations or standards, outline the compliance requirement.

4.2. Non-Functional Requirements:

1. Performance:

Specifies the maximum allowable time for a system to respond to a user request. Defines the number of transactions or operations a system should handle within a given time period. Describes how well the system can handle increased load or user activity by adding resources (e.g., hardware or servers).

2. Reliability:

Specifies the percentage of time the system should be operational and accessible to users. Describes the system's ability to continue functioning in the presence of hardware or software failure. Specifies the time it takes for the system to recover from a failure and return to normal operation.

3. Security:

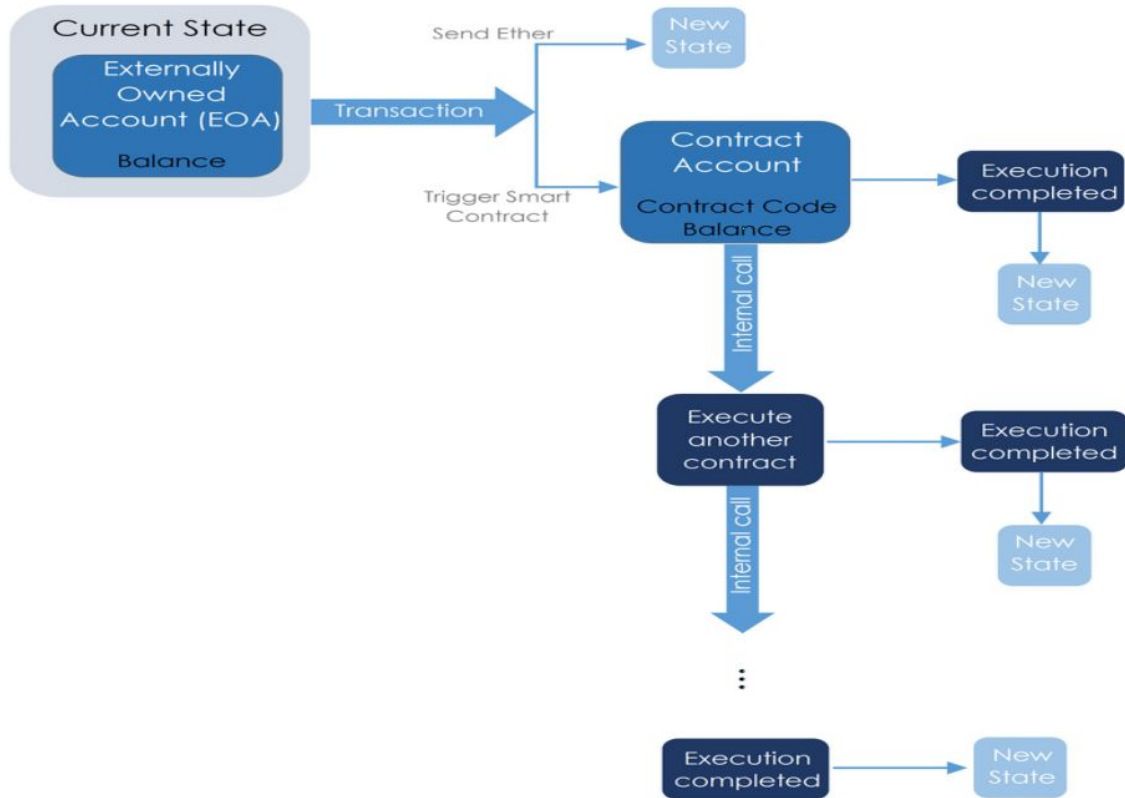
Specifies requirements for user authentication and access control. Describes the encryption requirements for sensitive data in transit and at rest. Defines the level of auditing and logging required for tracking and monitoring system activities.

4. Usability:

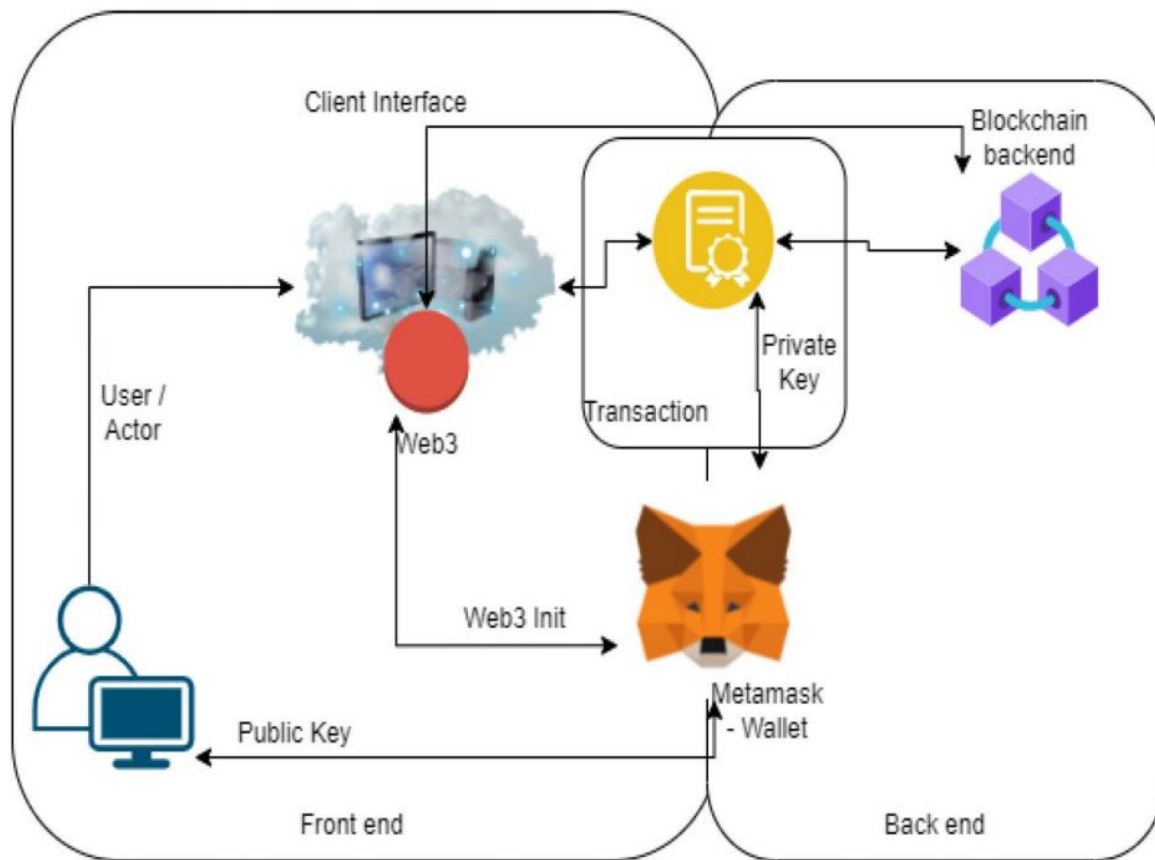
Describes the user interface's aesthetics, intuitiveness, and ease of use. Specifies compliance with accessibility standards to ensure that users with disabilities can access and use the system. Requires the provision of user guides and documentation to assist users in using the system effectively.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories:

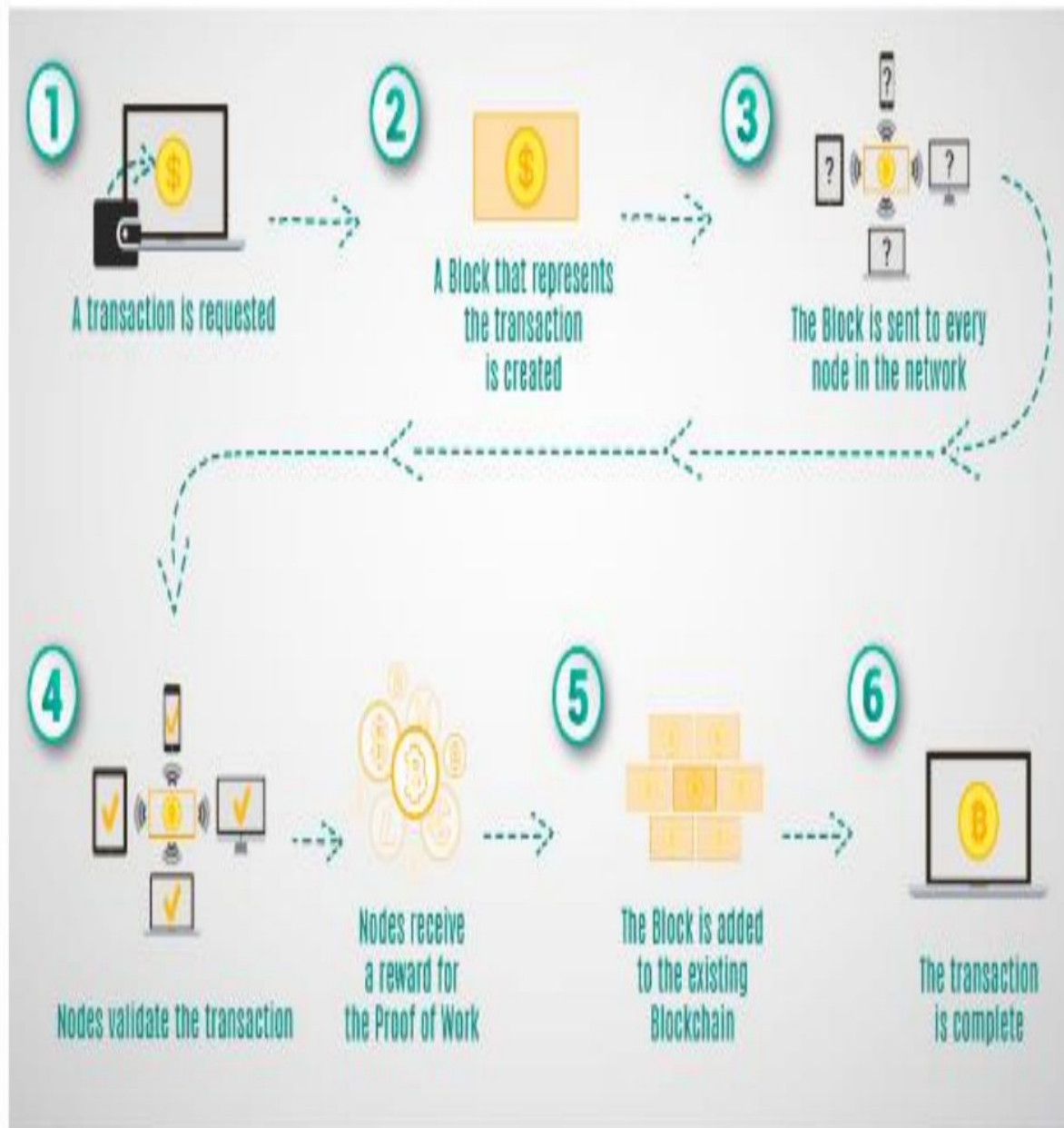


5.2 Solution Architecture:



6. PROJECT PLANNING

6.1 Technical Architecture:



7. CODING & SOLUTIONING:

SPDX-License-Identifier: This specifies the license under which the code is distributed. In this case, it's using the MIT license.

Pragma solidity 0.8.0 This indicates that the contract is designed to work with Solidity version 0.8.0 or higher. It ensures compatibility with the specified version.

1. Choose a Blockchain Platform:

Start by selecting a blockchain platform or framework suitable for your project. Common platforms for healthcare applications include Ethereum, Hyperledger Fabric, and Corda. The choice depends on factors like scalability, privacy, and the specific use case.

2. Design the Data Structure:

Define the structure of the blockchain for storing EHRs. Each block in the chain should contain a set of records. Consider how patient data will be structured and encrypted for privacy.

3. Data Encryption:

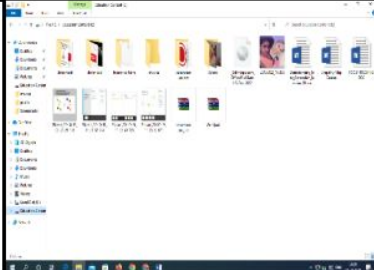

Encrypt EHRs and ensure data privacy. Use encryption techniques to secure the data in transit and at rest. Patient consent and control over data sharing are crucial considerations.




4. Smart Contracts:

Develop smart contracts to automate processes such as consent management, data sharing, and access control. These contracts can be written in Solidity (for Ethereum) or another suitable language for your chosen blockchain platform.

8. PERFORMANCE TESTING:

8.1 Performance Metrics:

S.No.	Parameter	Values	Screenshot
1.	Information gathering	Setup all the Prerequisite:	
2.	Extract the zip files	Open to vs code	

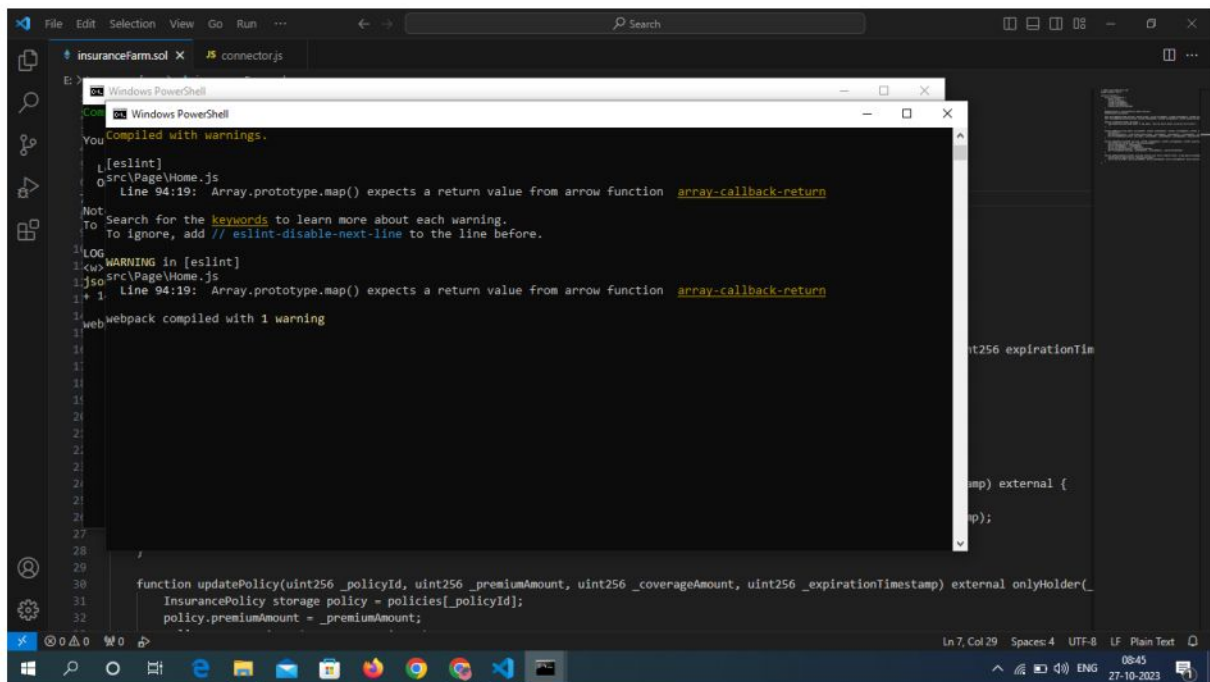
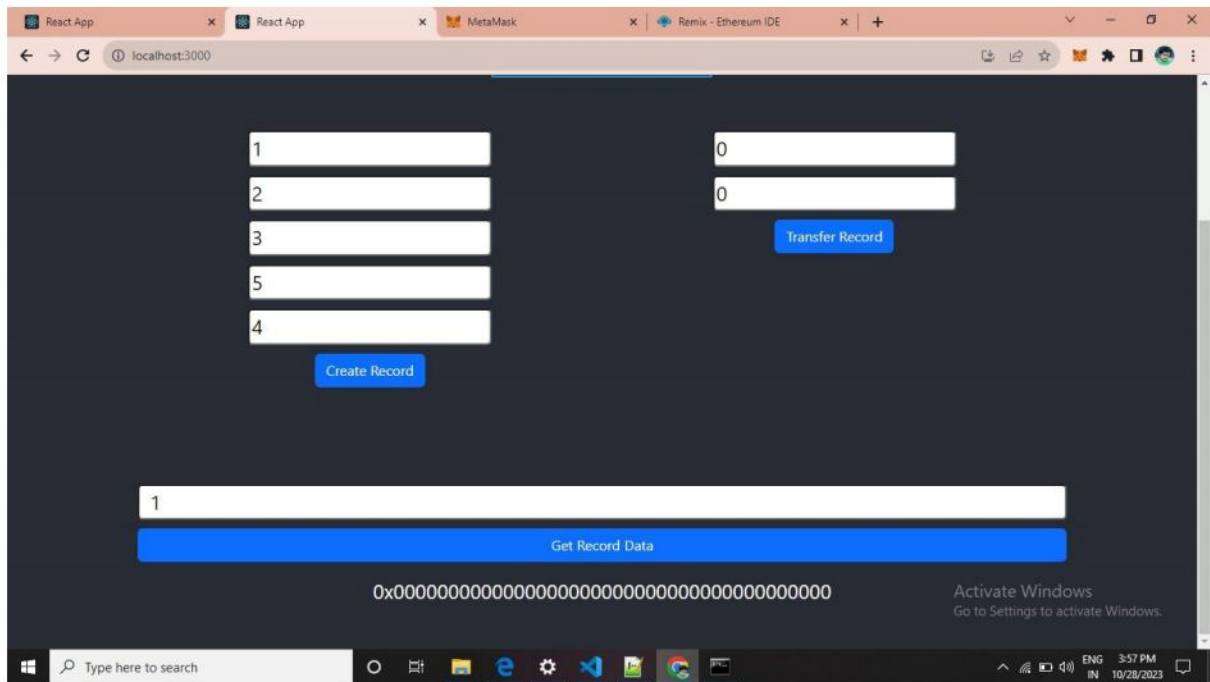
3.	Remix Ide platform exploring	Deploy the smart contract code Deploy and run the transaction. Byselecting the environment - inject the MetaMask.	
4	Open file explorer	Open the extracted file and click onthe folder. Open src, and search for utiles. Open cmd enter commands 1.npm install 2.npm install bootstrap 3. npm start	
5	{LOCALHOST ADDRESS I PADDRESS	copy the address and open it to chrome so you can see the front end of your project.	

9. RESULTS

9.1 OUTPUT SCREENSHOTS:

```
1 const { ethers } = require("ethers");
2
3 const abi = [
4   {
5     "anonymous": false,
6     "inputs": [
7       {
8         "indexed": true,
9         "internalType": "uint256",
10        "name": "recordId",
11        "type": "uint256"
12      },
13      {
14        "indexed": true,
15        "internalType": "address",
16        "name": "patientAddress",
17        "type": "address"
18      }
19    ],
20    "name": "RecordCreated",
21    "type": "event"
22  },
23  {
24    "anonymous": false,
25    "inputs": [
26      {
27        "indexed": true,
28        "internalType": "uint256",
29        "name": "recordId",
30        "type": "uint256"
31      },
32      {
33        "indexed": true,
34        "internalType": "address",
35        "name": "from",
36        "type": "address"
37      },
38      {
39        "indexed": true,
40        "internalType": "address",
41        "name": "to",
42        "type": "address"
43      }
44    ],
45    "name": "RecordTransferred",
46    "type": "event"
47  }
48 ]
```

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract HealthRecords {
5
6   struct PatientRecord {
7     string Name;
8     address patientAddress;
9     string diseases;
10    string contactInfo;
11  }
12
13  mapping(uint256 => PatientRecord) public records;
14
15  event RecordCreated(uint256 indexed recordId, address indexed patientAddress);
16  event RecordTransferred(
17    uint256 indexed recordId,
18    address indexed from,
19    address indexed to
20  );
21
22  modifier onlyOwner(uint256 recordId) {
23    require(records[recordId].patientAddress == msg.sender, "Only contract owner can call this");
24  }
25 }
```



10. ADVANTAGE & DISADVANTAG

10.1 ADVANTAGE:

1. **Data Security:** Blockchain's decentralized and immutable nature ensures that patient data is secure and tamper-resistant. Data is stored in a distributed network of nodes, making it extremely challenging for unauthorized parties to alter or access patient records without proper authorization.
2. **Patient Privacy:** Blockchain technology allows patients to have greater control over their EHRs. Patients can grant and revoke access to their records, ensuring that their data is only shared with authorized healthcare providers. This control enhances patient privacy and consent management.
3. **Interoperability:** Blockchain can facilitate data interoperability among different healthcare providers and systems. It uses standardized data formats and ensures that information is consistent and readily accessible, improving the exchange of healthcare data and patient care continuity.
4. **Efficient Data Sharing:** Healthcare professionals can access patient records more efficiently and securely, reducing the time and effort required to obtain relevant information. This can lead to faster and more accurate diagnosis and treatment.
5. **Reduced Data Redundancy:** Blockchain eliminates the need for multiple copies of patient records in different healthcare institutions. This reduces data redundancy and minimizes storage costs while improving data consistency.

10.2 DISADVANTAGE:

1. **Scalability Challenges:** Blockchain networks, especially public blockchains like Ethereum, face scalability issues. As the number of transactions and data size grows, the network may experience slower transaction processing times, which can be problematic for real-time healthcare applications.
2. **Energy Consumption:** Some blockchain networks, like Bitcoin, consume significant amounts of energy due to the proof-of-work consensus mechanism. This environmental impact may not align with sustainable practices in the healthcare industry.
3. **Complexity and Learning Curve:** Developing and managing blockchain-based EHR systems can be complex, requiring specialized knowledge and expertise. Healthcare professionals may need training to effectively use and interact with blockchain systems.
4. **Integration Challenges:** Integrating blockchain with existing EHR systems and healthcare infrastructure can be challenging. Legacy systems may not seamlessly communicate with blockchain networks, leading to data migration and compatibility issues.
5. **Regulatory Compliance:** While blockchain can enhance security, it may not fully address all regulatory and legal requirements in healthcare, such as HIPAA compliance in the United States. Ensuring that blockchain-based EHRs adhere to healthcare regulations is essential.
6. **Data Privacy Concerns:** While blockchain provides strong data security, it also creates a permanent and immutable record of data. This can be a concern if sensitive information needs to be expunged or updated due to legal or medical reasons.

11. CONCLUSION

In conclusion, the integration of blockchain technology into Electronic Health Records (EHRs) represents a promising path forward in the healthcare sector. Blockchain's unique features, including decentralization, immutability, and data security, offer solutions to many of the persistent challenges in EHR management. However, the adoption of blockchain in healthcare is not without its complexities and considerations.

The successful implementation of blockchain in EHRs requires a thoughtful and well-planned approach. It demands a balance between the benefits and challenges, addressing scalability with scalable solutions, minimizing energy consumption, ensuring regulatory compliance, and designing user-friendly interfaces for healthcare professionals and patients.

12. FUTURE SCOPE

The future scope of using blockchain technology for Electronic Health Records (EHRs) is highly promising. As the healthcare industry continues to evolve, blockchain can play a pivotal role in shaping the way patient data is managed, shared, and secured. Here are some key areas of future scope for blockchain in EHRs. Blockchain will continue to drive improvements in data sharing and interoperability across different healthcare providers, institutions, and systems. Standardized data formats and smart contracts will make it easier to share patient data seamlessly, ultimately improving patient care continuity.

13. APPENDIX

13.1 SOURCE CODE:

```
// SPDX-License-Identifier: MIT
```

```
pragma solidity ^0.8.0;
```

```
contract HealthRecords {
```

```
    struct PatientRecord {
```

```
        string Name;
```

```
        address patientAddress;
```

```
        string diseases;
```

```
        string contactInfo;
```

```
    }
```

```
    mapping(uint256 => PatientRecord) public records;
```

```
    event RecordCreated(uint256 indexed recordId, address indexed  
patientAddress);
```

```
    event RecordTransferred(
```

```
        uint256 indexed recordId,
```

```
        address indexed from,

        address indexed to

    );
```

```
    modifier onlyOwner(uint256 recordId) {

        require(msg.sender == records[recordId].patientAddress,"Only contract
owner can call this");

        _;

    }
```

```
    function createRecord(

        uint256 recordId,

        string memory name, address _patientAddress, string memory _diseases,
string memory _contactInfo

    ) external {

        records[recordId].Name = name;

        records[recordId].patientAddress = _patientAddress;

        records[recordId].diseases = _diseases;

        records[recordId].contactInfo = _contactInfo;
```



```

        emit RecordCreated(recordId, _patientAddress);

    }

    function transferRecord(uint256 recordId, address newOwner) external
    onlyOwner(recordId) {

        //require(records[recordId].patientAddress == newOwner, "New Owner
        should have different Address");

        require(records[recordId].patientAddress == msg.sender, "Only record
        owner can transfer");

        records[recordId].patientAddress = newOwner;

        emit RecordTransferred(recordId, records[recordId].patientAddress,
        newOwner);

    }

    function getRecordData(

        uint256 recordId

    ) external view returns (string memory, address, string memory,string
    memory) {

        return (records[recordId].Name,

            records[recordId].patientAddress,

```

```
        records[recordId].dieses,  
        records[recordId].contactInfo);  
    }  
}
```

```
function getRecordOwner(uint256 recordId) external view returns (address)  
{  
    return records[recordId].patientAddress;  
}
```

13.2 GitHub & Project Demo Link: