SVKM'S NMIMS Deemed to be UNIVERSITY

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING

Capstone Project (2023)

Interim Report

Project Title:

Decentralised Data and Resource-Provisioning System with End-Point Security

Group:

Yash Burshe K007

Ridhima Mishra K035

Arushi Rai K047

B. Tech Computer Science Engineering

(Specialization: Cybersecurity)

Faculty Mentor: Dr. Pintu Shah

Industry Mentor: Mr. Godwin Jathanna

Table of Contents:

Statement of Purpose:

In the current data management practices still being implemented at leading organizations that process a plethora of data across thousands of employees, vendors, and customers, the **manual allotment** of data access and permissions introduces significant vulnerabilities, **making the system prone to single points of failure and susceptible to various risks**. For instance, human errors, disgruntled employees, "whistle-blowers", etc. with unauthorized access pose serious threats to data security and integrity. These factors can lead to data breaches, leakage of sensitive information, and compromised data privacy, resulting in severe consequences for organizations.

To mitigate these risks and enhance data security, our project proposes a **decentralized file storage system, employing blockchain technology** to revolutionize data access control. We want to replace the manual allocation process with a **consensus-based system, controlled and maintained by smart contracts.** The goal is to create a more secure, transparent, and tamper-resistant data management solution.

Our proposed system utilizes a **Proof of Authority mechanism**, and the smart contracts are pre-defined based on company policies, laws, guidelines, and regulations. This system ensures that only authorized entities are granted access to specific data based on their roles within the organization hierarchy. Each user's reputation is stored on the blockchain in line with that user's position in the organization hierarchy, establishing a trust-based approach for data access and permission granting.

By leveraging blockchain's decentralized nature and immutability, the system **minimizes the risks associated with centralized authority and reduces the chances of data breaches due to human factors**. The smart contracts act as self-executing protocols, **eliminating the need for intermediaries**, and ensuring that access control decisions are **automated, transparent, and resistant to manipulation.**

The proposed system brings greater accountability and transparency, as all data access and permission changes are recorded on the blockchain, making it **auditable and traceable**. Any attempt to tamper with the data or modify access permissions without proper authorization will be immediately detected and flagged, ensuring data integrity and mitigating insider threats.

Scope and Objectives:

The main objectives of this project are to develop a secure and efficient blockchain and smart contracts-controlled data sharing platform with multi-user access and role-based authentication. The platform aims to address the vulnerabilities in current data management practices by providing fine-grained access control, persistent metadata management, and robust end-point security measures.

1. **Blockchain-Controlled Data Sharing**: Our primary goal is to create a decentralized data sharing platform leveraging blockchain technology. This will ensure a distributed and tamper-resistant network, eliminating single points of failure and unauthorized access. We will utilize smart contracts to automate data sharing and resource-provisioning processes, to promote transparency and accountability.

2. **Multi-User Access and Role-Based Authentication**: This project will implement a comprehensive user authentication system with role-based access control (RBAC). Different user roles within the organization hierarchy will be granted specific access permissions, enhancing data security and minimizing the risk of unauthorized data exposure. The roles of individuals in the company will also be used to facilitate a Proof of Authority consensus mechanism.

3. **Persistent Metadata Management**: Our platform will incorporate a robust metadata management system to facilitate policy-compliant data classification and access control. Metadata will store vital information related to data ownership, access permissions, update records, and usage policies, and much more, to ensure compliance with company policies and regulatory requirements.

4. **End-Point Security Measures**: To enhance user interactions' security, our project will implement end-point security measures. Secure communication protocols will be employed to safeguard data transmission between clients and the platform.

5. **Encryption with Integrity Controls**: To safeguard multi-user data, we will use encryption with integrity controls. Advanced cryptographic techniques will be used to ensure data confidentiality and integrity, protecting sensitive information from unauthorized access and tampering.

6. **Fine-Grained File Sharing**: The project will analyze and implement access control mechanisms to enable fine-grained file sharing between various components of the organization hierarchy.

Attribute-based access control (ABAC) will be explored to handle complex access control policies efficiently.

7. **Notifications for Policy Non-Compliance or Violations**: An essential aspect of the platform will be the incorporation of a notification system to alert relevant users and administrators in case of policy non-compliance or access control violations. Real-time notifications will be sent through emails, messages, and in-app alerts to promptly address any security breaches or policy violations.

Literature Survey:

BDSS-FA: A Blockchain-Based Data Security Sharing Platform With Fine-Grained Access Control: The paper presents BDSS-FA, a blockchain-based data security sharing platform with fine-grained access control for the Internet of Things (IoT). The platform is designed to address the challenge of establishing an effective data sharing mechanism between different organizations for IoT. BDSS-FA is based on blockchain and HABE, which provide a secure data sharing mechanism with traceability and fine-grained access control. The system framework includes Key Generation Center (KGC), Data Owner (DO), P2P based data distribution platform, IPFS Cluster, Fabric Blockchain and Data Consumer (DC). The paper also describes the system initialization process, the functions of the six entities in the system framework, and the deployment of the Fabric blockchain. BDSS-FA can provide a secure and efficient data sharing mechanism for IoT. Further research is needed to explore the integration of BDSS-FA with existing data management systems and the mechanisms by which BDSS-FA facilitates decentralized data and resource-provisioning.

Blockchain- Enabled Data Sharing in Supply Chains: Model, Operationalization, and Tutorial: This paper is about Blockchain-Enabled Data Sharing in Supply Chains. It discusses the challenges of data sharing in supply chains and how blockchain technology can be used to create a secure and trustworthy data-sharing mechanism. The paper proposes a new data-valuation and data-pricing mechanism called usage-based valuation, where the value of the data is determined based on its intended use. The paper also provides a step-by-step guide for setting up a blockchain-enabled data-sharing marketplace using Hashgraph. Additionally, the growing theoretical literature on data valuation and provides references to related studies is reviewed. The aim is to help readers overcome the challenges of data sharing in supply chains using blockchain technology.

A Medical Data Sharing Scheme Based on Attribute Cryptosystem and Blockchain Technology: This research paper proposes a medical data sharing scheme based on attribute

cryptosystem and blockchain technology. The scheme addresses security risks associated with existing medical data sharing schemes and ensures the authenticity of data sources. The encrypted medical data is stored in the cloud, and the storage address and medical- related information are written into the blockchain, which ensures efficient storage and eliminates the possibility of irreversible modification of the data. The proposed scheme combines attribute-based encryption (ABE) and attribute- based signature (ABS), which achieves the sharing of medical data in many-to-many communications. The ABE achieves data privacy and fine-grained access control, and the ABS verifies the authenticity of the source of the medical data while protecting the signer's identity. The scheme also reduces the computational burden by outsourcing most of the operations of medical data ciphertext decryption to the cloud service provider (CSP). The paper concludes that the proposed scheme satisfies the requirements for confidentiality and unforgeability in the random oracle model and offers higher computational performance than other similar schemes.

Project Outline:

(Gantt Chart)

Certainly, here's the project plan in the format you requested:

Week 1 (31st August – 6th September):
- Pick a name for the application.
- Define project scope, objectives, and stakeholders.
- Write up the company security policy.
- Set up communication channels and tools.
- Identify functional and non-functional requirements.
- Gather user stories and use cases for different roles.

Week 2 (7th September – 13th September):
- Design the overall system architecture.
- Plan the integration of blockchain, smart contracts, and authentication components.

Week 3 (14th September – 20th September):
- Set-up blockchain network and nodes.
- Implement data storage and retrieval on the blockchain.

Week 4 (21st September – 27th September):
- Design and code smart contracts for data sharing and access control.
- Test and deploy smart contracts on the blockchain.

Week 5 (28th September – 4th October):
- Design user registration and authentication processes.
- Implement role-based access control (RBAC) mechanisms.

Week 6 (5th October – 11th October):
- Design metadata structure for data ownership and access control.
- Implement metadata storage and retrieval functionalities.

Week 7 (12th October – 18th October):
- Backlog Clearing.
- Choose and implement secure communication protocols.

Week 8 (19th October – 25th October):
- Set up SSL/TLS certificates for data transmission.
- Choose encryption algorithms and methods.

Week 9 (26th October – 1st November):
- Implement data encryption and integrity checks.
- Design notification system architecture.

Week 10 (2nd November – 8th November):
- Create a user manual, technical documentation, and guide.
- Document architecture, codebase, and deployment procedures.

Applicability:

For the scope of this project, we have developed this web application "on contract" for an event management company called Eventopolis. We are referring to the software as 'Sentinel Share' and the policies and use cases we have developed are in accordance with the same context.

This policy applies to all personnel within the organization hierarchy, including any individuals/groups/organizations serving under a contract with Eventopolis:

1. **The CEO**
2. **Other C-level Executives** - CTO, CISO
3. **Horizontal Heads** (Hospitality, Public Relations)
4. **Project Managers** (specific for each new event)
5. **Vertical Heads** (specific for each of the departments for every event being planned)
6. **Third-party vendors** (being contracted by the company)
7. **Clients**

Access to the general public is regulated solely through the deficit of any data security and privacy controls, as will be illustrated in the Data Classification section of this document.

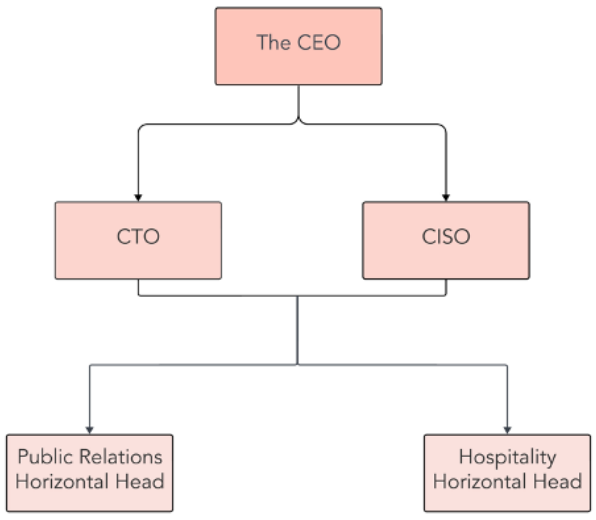


Diagram 1: General Company Hierarchy

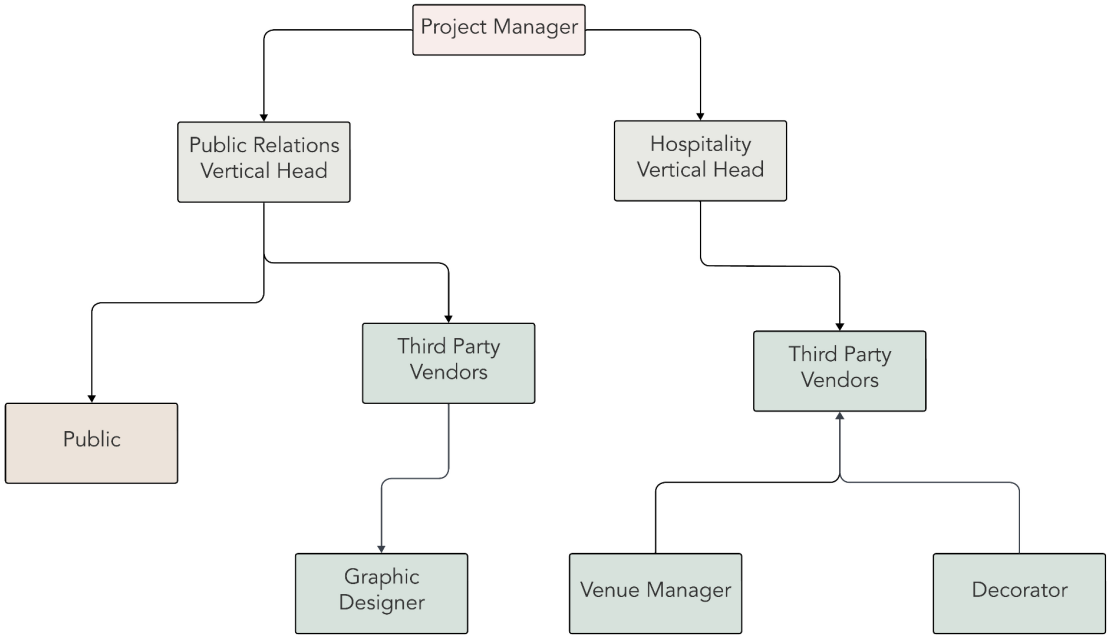


Diagram 2: Hierarchy established for the duration of a project

Refer to Diagram 3 to establish the distinction between the Levels of the Hierarchy and the Levels for Data Classification.

Data Classification:

Data in the company is classified into the following levels:

Secret Data:

Secret data is classified as highly sensitive and valuable information, very critical within Eventopolis. It includes information that, if exposed, could cause significant harm to Eventopolis or its interests. Access to secret data is granted to individuals who have a legitimate need to know and have been authorized to access this level of information within Eventopolis. Strict security measures are in place to protect secret data from unauthorized access.

Confidential Data:

Confidential data represents sensitive information that, while not as critical as secret or top-secret data, still holds considerable value and should be protected from unauthorized disclosure within Eventopolis. This category often includes proprietary business data, financial information, and customer data. Access to confidential data is typically limited to employees or stakeholders within Eventopolis who require it for their job responsibilities.

Internal Data:

Internal data refers to information that is essential for Eventopolis' day-to-day operations and decision-making but is not as sensitive as confidential, secret, or top-secret data. It may include internal communications, reports, and non-sensitive Eventopolis documents. Access to internal data is generally granted to Eventopolis employees and individuals who need it to perform their duties under a specific event project.

Restricted Data:

Restricted data includes information that is sensitive and should not be made publicly available but is not as critical as confidential, secret, or top-secret data within Eventopolis. This category often includes data shared with third-party vendors, partners, or specific stakeholders on a need-to-know basis. Access to restricted data within Eventopolis is tightly controlled, and individuals are given access based on their specific roles and responsibilities.

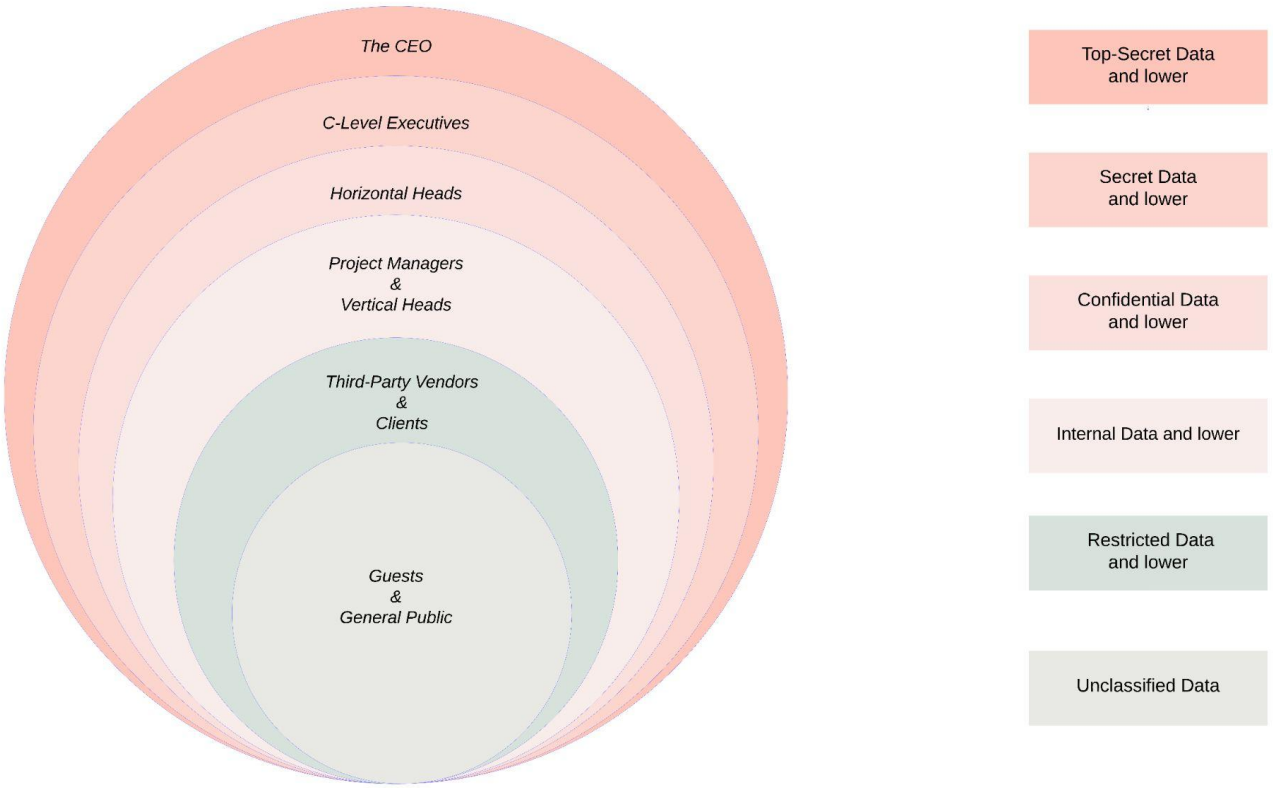
Unclassified Data: Unclassified data is information that poses little to no risk if disclosed to the general public or individuals without a specific need to know within Eventopolis. This category includes publicly available information, such as marketing materials, public-facing Eventopolis

websites, and non-sensitive public records. Unclassified data is typically accessible to anyone within Eventopolis and does not require special authorization or clearance to access.

Each level corresponds to specific roles within the organization, as detailed in the next section.

Authority and Access Control:

Access to data and resources is structured as follows:
- **All C-level Executives:** Access to secret level data and below.
- **Horizontal Heads:** Access to confidential level data and below.
- **Project Managers and Vertical Heads**: Access to internal level data and below.
- **Third-party vendors and Clients**: Access to restricted data and below.
- **General public**: Access to unclassified data.



Use Cases:

All data usage and handling actions will be logged on the SentinelShare decentralized database located on a private Ethereum test-net.

The specific functions, with instances, and associated metadata, which will be encrypted per Sentinel Share's standards are enlisted below:

{Use Cases created to limit the Scope of Work for this project}

Use Case 1:

Login to Portal

Step 1 – Login ID, Password

Step 2 – OTP

Step 3 – Authenticate and Provide Access

Use Case 2:

New Project from Client

Step 1 – Use Case 1

Step 2 – 'Request New Event' Button

Step 3 – Client enters the following information:

Event Date,

Event Time,

Event Type (formal/ casual/ party/ wedding),

Theme Type (dark/ warm/ light/ pastels/ monochrome),

Venue Type (small/ medium/ big/ large),

Guest List (yes -> [Guest Name/ Guest Phone/ Guest Email]/ no)

Step 4 – Submit

Use Case 3:

Vertical Head (Hospitality) Assigning Work to Vendors (Venue Manager/ Decorator)

Step 1 – Vertical Head (Hospitality): Use Case 1

Step 2 – Vertical Head (Hospitality) sees the following on their dashboard:

Event Date,
Event Time,
Event Type (formal/ casual/ party/ wedding),
Theme Type (dark/ warm/ light/ pastels/ monochrome),
Venue Type (small/ medium/ big/ large)

Step 3 – 'Assign to Venue Manager'/'Assign to 'Decorator' respectively

Step 4 – Submit

Step 5 – Vendor: Use Case 1

Step 6 – Vendor sees the following on their dashboard:

If Vendor.Type == 'Decorator':
Event Date,
Event Time,
Event Type (formal/ casual/ party/ wedding),
Theme Type (dark/ warm/ light/ pastels/ monochrome),
Venue Type (small/ medium/ big/ large)

Else If Vendor.Type == 'Venue Manager':
Event Date,
Event Time,
Venue Type (small/ medium/ big/ large)

Step 7 – Vendor adds the following information on their dashboard:

If Vendor.Type == 'Decorator':
'Approved'/ 'Rejected'

Else If Vendor.Type == 'Venue Manager':
'Approved' -> 'Add Venue Address'/ 'Rejected'

Step 8 – Submit

Step 9 – Vertical Head (Hospitality): Use Case 1

Step 10 – (Step 2) + Vendor Response 'Event Venue Address'

Step 11 – 'Approve'

Step 12 – Submit

Step 13 – Client: Use Case 1

Step 14 – Client sees contents from (Step 2) and Event Venue Address

Step 15 – 'Approve'

Use Case 4:

Vertical Head (Public Relations) Assigning Work to Vendor (Graphic Designer)

Step 1 – Vertical Head (Public Relations): Use Case 1

Step 2 – Vertical Head (Public Relations) sees the following on their dashboard:

Event Date,
Event Time,
Event Type (formal/ casual/ party/ wedding),
Theme Type (dark/ warm/ light/ pastels/ monochrome),
Venue Type (small/ medium/ big/ large)
Event Venue Address

Step 3 – 'Assign to Graphic Designer'

Step 4 – Submit

Step 5 – Vendor: Use Case 1

Step 6 – Vendor sees the following on their dashboard:

Event Date,
Event Time,
Event Type (formal/ casual/ party/ wedding),
Theme Type (dark/ warm/ light/ pastels/ monochrome),
Event Venue Address

Step 7 – Vendor adds the design files in .png/.jpg/.jpeg/.ai/.ps/.zip format on their dashboard

Step 8 – Submit

Step 9 – Vertical Head (Public Relations): Use Case 1

Step 10 – (Step 2) + Vendor Response files of 'Invitation Design'

Step 11 – 'Approve'

Step 12 – Submit

Step 13 – Client: Use Case 1

Step 14 – Client sees contents from (Step 2) and 'Invitation Design'

Step 15 – 'Approve'

Use Case 5:

Graphic Designer Attempting to Escalate Privilege by Requesting Additional Data

Use Case 4's Steps 1 to 6

Step 7 – Vendor fills a 'Data Requisition' form on another tab on the dashboard, and enters the following:

Data Required = "Guest Names"

Reason = "To put on the invitations, so that they seem more customized and appropriate for such a personal event."

Step 8 – Submit Data Requisition Form

Step 9 – Vertical Head (Hospitality) sees the Data Requisition form on their dashboard

Step 10 – Vertical Head (Hospitality) clicks on 'Approve'/'Deny'

Step 11 – Submit

Step 12 – Project Manager: Use Case 1

Step 12 – Project Manager sees the Data Requisition form submitted by the vendor on their dashboard along with the Vertical Head (Hospitality)'s approval or denial of the request.

Step 13 – Project Manager clicks 'Approve'/'Deny'

Step 14 – Vendor: Use Case 1

Step 15 – Vendor sees the approval or denial of the request basis this algorithm:

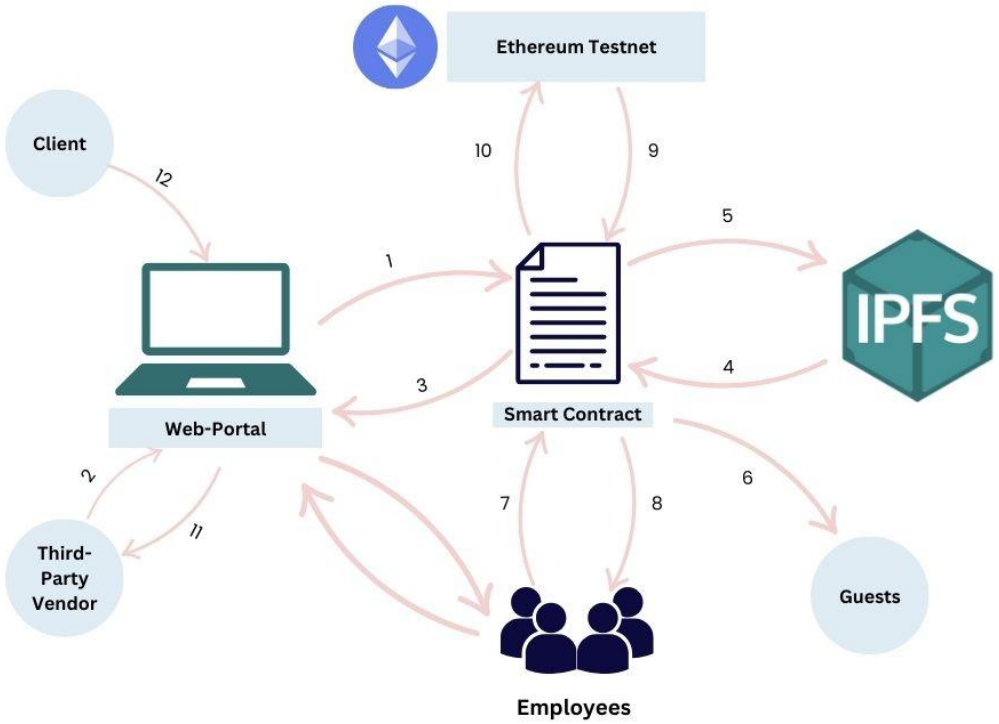
```
If Step 10 == 'Approve'
        If Step 13 == 'Deny':
                'Denied'
        Else
                'Approved' and Guest List table visible.

Else if Step 10 == 'Deny'
        If Step 13 == 'Approve':
                'Approved' and Guest List table visible.
        Else
                'Denied'
```

Step 16 – Use Case 4 Step 7 onwards.

Data handling adheres to data protection regulations, backup requirements, and network security standards. Encryption protocols are used for secure data communication. Policies cover data creation, classification, updates, deletions, and provisioning actions logged on the blockchain.

System Architecture:



The above diagram allows us to visualize the data flow in the software.

The following table contains information about all technologies employed to make/communicate with the individual entities in our entire system application, including all human actors and computer systems:

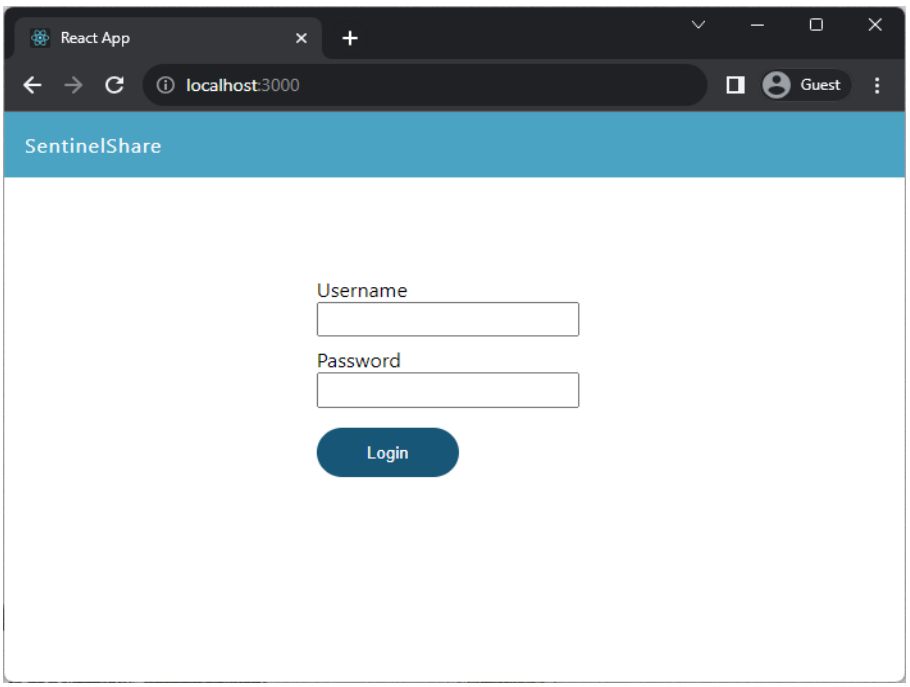
The Web Portal	HTML, CSS, JS, Node.js, React.js
Smart Contracts	Truffle, Remix IDE for Solidity
Inter-Planetary File System	Helia, IPFS CLI, MongoDB

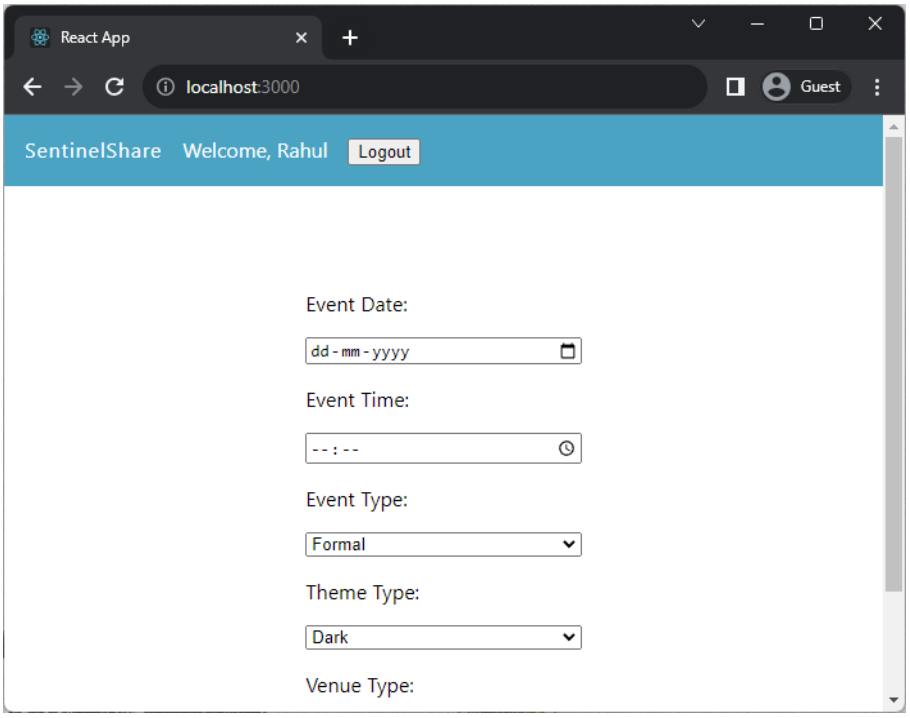
Ethereum Testnet	Sepolia, Metamask
Employees of Eventopolis	Communicating via the Web Portal Dashboard and Email/WhatsApp messages for 2FA
Clients of Eventopolis	Communicating via the Web Portal Dashboard and Email/WhatsApp messages for 2FA
Third-Party Vendors contracted by Eventopolis	Communicating via the Web Portal Dashboard and Email/WhatsApp messages for 2FA
General Public/ Guests of the events being organized by Eventopolis	Communicated to via Email/WhatsApp messages for invites and publicity purposes.

Practical Implementation:

Frontend

The frontend employs the use of ReactJS to create a single page application. It allows for organization users to login and view, add, or approve file access for requesting users (eg. third party clients). The project is deployed through Docker and maintained using a CI/CD pipeline setup on Github Actions.





Backend

Our backend is built using three main components, a Blockchain, InterPlanetary FileSystem and NodeJs. Data is stored in IPFS blocks and the unique Content Identifiers (CIDs) for the data (given by IPFS) is stored in smart contracts on the Ethereum Testnet 'Sepolia'. Using NodeJs, we call the smart contract's functions and retrieve the CID which we then pass to IPFS and get our data. Smart contracts, IPFS and NodeJs are all automated using an automation pipeline.

```
function getDatabaseCID() {
  //Call getHash method of contract and log value (This contains the
IPFS CID)
  contract.methods
    .getHash()
    .call()
    .then((value) => {
      console.log(`AuthDB CID: ${value}`);
      console.log("Fetch successful!\n");
    })
    .catch((error) => console.error(error));
}
```

In function getDatabaseCID(), we use web3.js's powerful web3.eth.Contract function to call our Smart Contract's function: getHash().

```
function getHash() public view returns (string memory) {
        return ipfsHash;
    }
```

This code is a part of the Smart Contract written in Solidity. The function simply returns a value ipfsHash that has been previously stored in it while executing the contract.
Once we get our CID, we can get our data from IPFS using Helia's .get() function by passing the CID and getting a return value, i.e. our data stored on that IPFS block.

```
import '@testing-library/jest-dom';

import { createHelia } from 'helia'
import { strings } from '@helia/strings'
import { json } from '@helia/json'




const helia = await createHelia()
const s = strings(helia)


const myImmutableAddress = await s.add('hello world')
console.log(typeof myImmutableAddress)


console.log(await s.get(myImmutableAddress))




const j = json(helia)


const myImmutableAddress2 = await j.add({ hello: 'world' })


console.log(await j.get(myImmutableAddress2))
```

Helia is an implementation of IPFS on JavaScript. This makes it easier to interact with our IPFS implementation without manual intervention. The code creates two immutable strings and an immutable JSON object using the Helia library. The createHelia() function creates a new Helia instance, which is a container for immutable data. The strings() and json() functions return objects that can be used to create and manipulate immutable strings and JSON objects, respectively. The add() method on the strings() and json() objects takes the data as input and returns an immutable address for that data. The get() method on the strings() and json() objects takes the immutable address as input and returns the data stored at that address.

Key Features:

1. Smart contracts: self-executing protocols, eliminates the need for intermediaries
2. Resistant to collusion and manipulation
3. Increases immunity to social engineering
4. Enhances auditability and traceability
5. Ensures policy and regulatory compliance
6. More prompt notifications in case of suspicious activity
7. Scalability and performance
8. Future scope in IoT-related data sharing

Challenges Up Till Now:

1. Integration of IPFS with our Smart Contracts (data sharing between the files is changing the data type to string from an object)
2. Automation pipeline building to remove any manual intervention required on the Truffle Dashboard and Metamask
3. Sending retrieved data from IPFS for authentication to the React.js frontend