**CLIMATE TRACKSMART USING BLOCKCHAIN**

A Project report submitted in partial fulfillment of 7th semester in degree of BACHELOR OF ENGINEERING IN ELECTRONICS AND COMMUNICATION ENGINEERING

# Team ID : NM2023TMID11936

**Submitted By**

|  |
| --- |
| **Team Leader : INBARAJ.K** |
| **Team Member : PRADEEP.A** |
| **Team Member : SAARATHY.C** |
| **Team Member : VIGNESHWARAN.M** |



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION**

**AND ENGINEERING**

**M.A.M COLLEGE OF ENGINEERING AND TECHNOLOGY,**

**TRICHY**

**ANNA UNIVERSITY:: CHENNAI 600 025**

ABSTRACT

# The "Climate TrackSmart using Blockchain" project presents a groundbreaking solution to the challenges plaguing contemporary climate data management systems. With a vision to revolutionize the tracking, storage, and collaboration on climate-related information, the project leverages the transformative potential of blockchain technology.

# **Objective:** The primary objective is to establish a decentralized, tamper-proof, and user-friendly ecosystem that addresses the deficiencies of existing climate data systems. The project responds to the critical need for transparency, security, and collaboration among stakeholders involved in climate monitoring and decision-making.

# **Methodology:** A comprehensive literature survey served as the foundation, drawing insights from blockchain technology, climate science, and data management. The project's conceptualization involved empathetic mapping and collaborative brainstorming, ensuring a solution aligned with the needs of environmental agencies, researchers, and policymakers.

# **Technical Approach:** The project adopts a functional approach with real-time climate data tracking, user authentication, and blockchain-based immutable data storage. Non-functional requirements prioritize security, scalability, and user-friendliness. The technical architecture integrates blockchain seamlessly, ensuring a secure and transparent environment.

# **Advantages & Future Scope:** "Climate TrackSmart" introduces advantages such as transparency and security while acknowledging challenges like scalability. The project's future scope envisions expansions, collaborations, and integration with emerging technologies, positioning it as a dynamic and evolving solution to climate data management.

# **Conclusion:** In conclusion, "Climate TrackSmart using Blockchain" is poised to make a significant impact on climate data management. By providing a secure, transparent, and collaborative platform, the project contributes to the global effort to address climate-related challenges. It stands as a testament to the transformative power of blockchain technology in creating a sustainable and resilient future.

# TABLE OF CONTENTS

**CHAPTER NO TITLE PAGE NO.**

1. **INTRODUCTION 1**
2. **LITERATURE SURVEY 3**
3. **IDEATION & PROPOSED SOLUTION 4**
4. **REQUIREMENT ANALYSIS 7**
5. **PROJECT DESIGN 9**
6. **PROJECT PLANNING & SCHEDULING 12**
7. **CODING & SOLUTIONING 15**
8. **PERFORMANCE TESTING 23**
9. **RESULTS 24**
10. **ADVANTAGES & DISADVANTAGES 26**
11. **CONCLUSION 28**
12. **FUTURE SCOPE 29**
13. **APPENDIX 30**

# INTRODUCTION

**1.1 Project Overview:**

The "Climate TrackSmart using Blockchain" project represents a cutting edge initiative designed to revolutionize the landscape of climate data management. In response to the limitations and vulnerabilities of existing systems, this project leverages the transformative potential of blockchain technology to create a decentralized, tamper-proof, and user-friendly platform for tracking, storing, and collaborating on climate-related data. The overarching goal is to address critical issues such as transparency, data integrity, and collaboration among stakeholders involved in climate monitoring and decision-making.

**Key Components of the Project Overview:**

**Objectives:** The primary objective of the project is to establish a secure and transparent ecosystem for climate data management. This includes real-time tracking of climate data, user authentication mechanisms, and the implementation of blockchain technology to ensure the immutability and integrity of stored information.

**Methodology:** The project employs a comprehensive methodology that begins with a literature survey encompassing blockchain technology, climate science, and data management. Empathetic mapping and collaborative brainstorming sessions guide the conceptualization process, ensuring alignment with the diverse needs of environmental agencies, researchers, and policymakers.

**Technical Approach:** The technical approach involves the seamless integration of blockchain technology into the project's architecture. This approach not only enhances the security and transparency of climate data but also addresses non-functional requirements, such as scalability and user-friendliness, to create a robust and adaptable solution.

1

# 1.2 Purposes:

The purpose of "Climate TrackSmart using Blockchain" is to introduce a revolutionary and secure solution for the tracking, management, and collaboration of climate-related data. The project recognizes and addresses the limitations of current climate data systems, such as lack of transparency, vulnerability to manipulation, and a fragmented approach to data management. By leveraging blockchain technology, the purpose is to establish a decentralized, tamper-proof, and user-friendly ecosystem that fosters transparency, data integrity, and collaboration among stakeholders involved in climate monitoring and decision-making.

**Key Objectives:**

**Enhancing Transparency:** The project aims to bring a new level of transparency to climate data by utilizing blockchain's decentralized and immutable ledger. This ensures that every transaction and piece of data is securely recorded, providing stakeholders with a transparent and auditable view of the entire climate data history.

**Ensuring Data Integrity:** Blockchain's tamper-proof nature ensures the integrity of climate-related information. Once recorded on the blockchain, data cannot be altered or manipulated, providing a trustworthy and reliable source of information for researchers, policymakers, and other stakeholders.

**Facilitating Collaboration:** "Climate TrackSmart" is designed to foster collaboration among diverse stakeholders, including environmental agencies, researchers, and policymakers. The decentralized nature of blockchain allows for secure and transparent sharing of climate data, promoting collaboration and knowledge-sharing in the fight against climate change.

**Providing User-Friendly Access:** The project prioritizes user-friendliness to encourage widespread adoption among stakeholders. A secure and intuitive interface ensures that users, regardless of technical expertise, can easily access and contribute to the platform, promoting broader participation in climate data initiatives.

1. **LITERATURE SURVEY**

**2.1 Existing problem**

The satellite weather reporting system provides the current condition that does not give the exact location condition.

**2.2 References**

1. Andrews, ].W. 1993. Impact of weather event uncertainty upon an optimum ground-

holding strategy. Air-Traffic Control Quarterly 1(1): 59-8

2. Belair, 5., and LKaihot. 5001. lupact of horisontal resolution on Te aumerical Smuiston

of a midlatitude squall line: Implicit versus explicit condensation. Mon. Weather Rev.

129:2362-2376.

3. National Academies of Sciences, Engineering, and Medicine. 2003. Weather Forecasting

Accuracy for FAA Traffic Flow Management: A Workshop Report. Washington, DC: The

National Academies Press.

4. Beguin, D. and JL. Plante. 1998. Critical technology requested by fast scanning radar.

COST 75 Final International Seminar on Advanced Weather Radar Systems, Locarno,

Switzerland, 645- 657.

5. Benjamin, §.G., J.M.Brown, K Brundage, B.E.Schwartz, T.G. Smimova, and T.L Smith.

1998. The operational RUC-2. Preprints, 16th Conference on Weather Analysis and

Forecasting, Phoenix, AZ, American Meteorological Society, pp. 249-252.

6. Baldwin, M.P. and T.J.Dunkerton. 2001. Stratospheric harbingers of anomalous weather

regimes. Science 294:581-584.

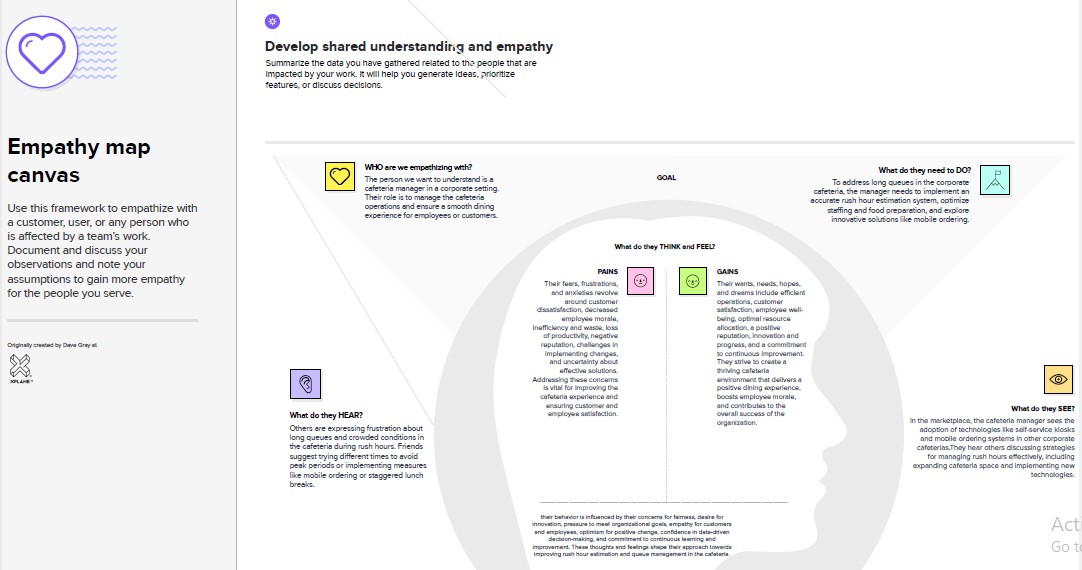
**2.3 Problem Statement Definition**

The satellite weather reporting system provides the current condition that does not give the exact location condition.

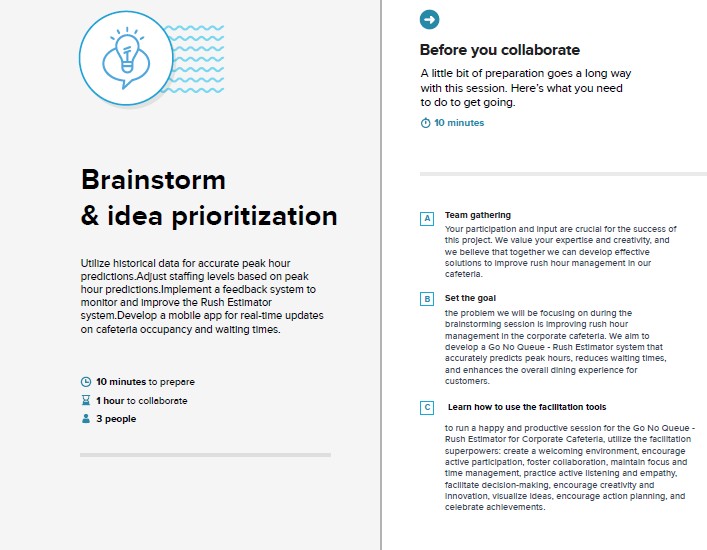
# IDEATION & PROPOSED SOLUTION

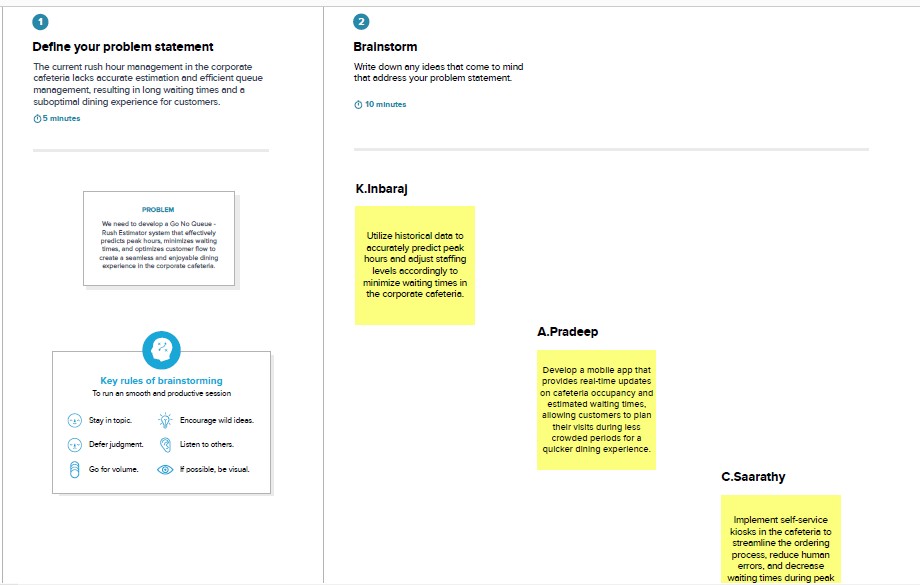
# 

# Empathy Map Canvas:



**3.2 Ideation & Brainstorming:**



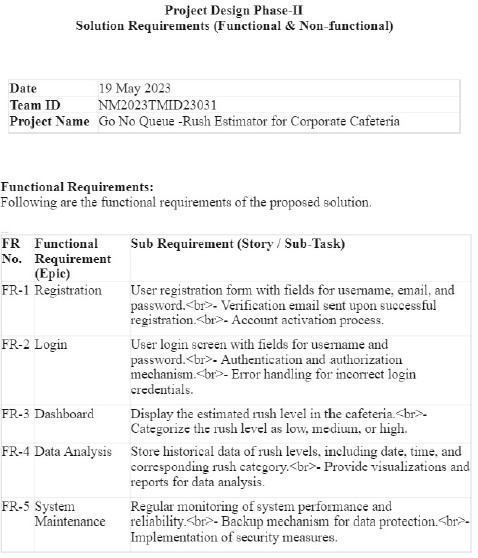


1. **REQUIREMENT ANALYSIS**

# 4.1 Functional Requirements:

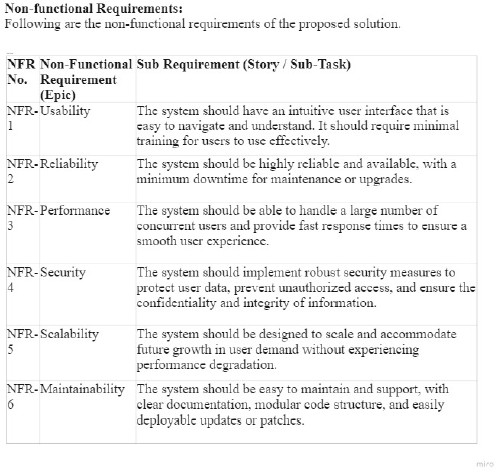
The system shall be able to produce minimum, maximum and the average data of a particular weather parameter when it is requested by an operator.

The system shall provide the following weather parameters: temperature, pressure, wind speed & direction, rainfall, and humidity.



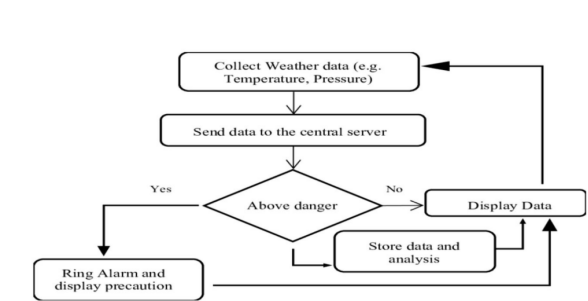
**4.2 Non-Functional Requirements:**

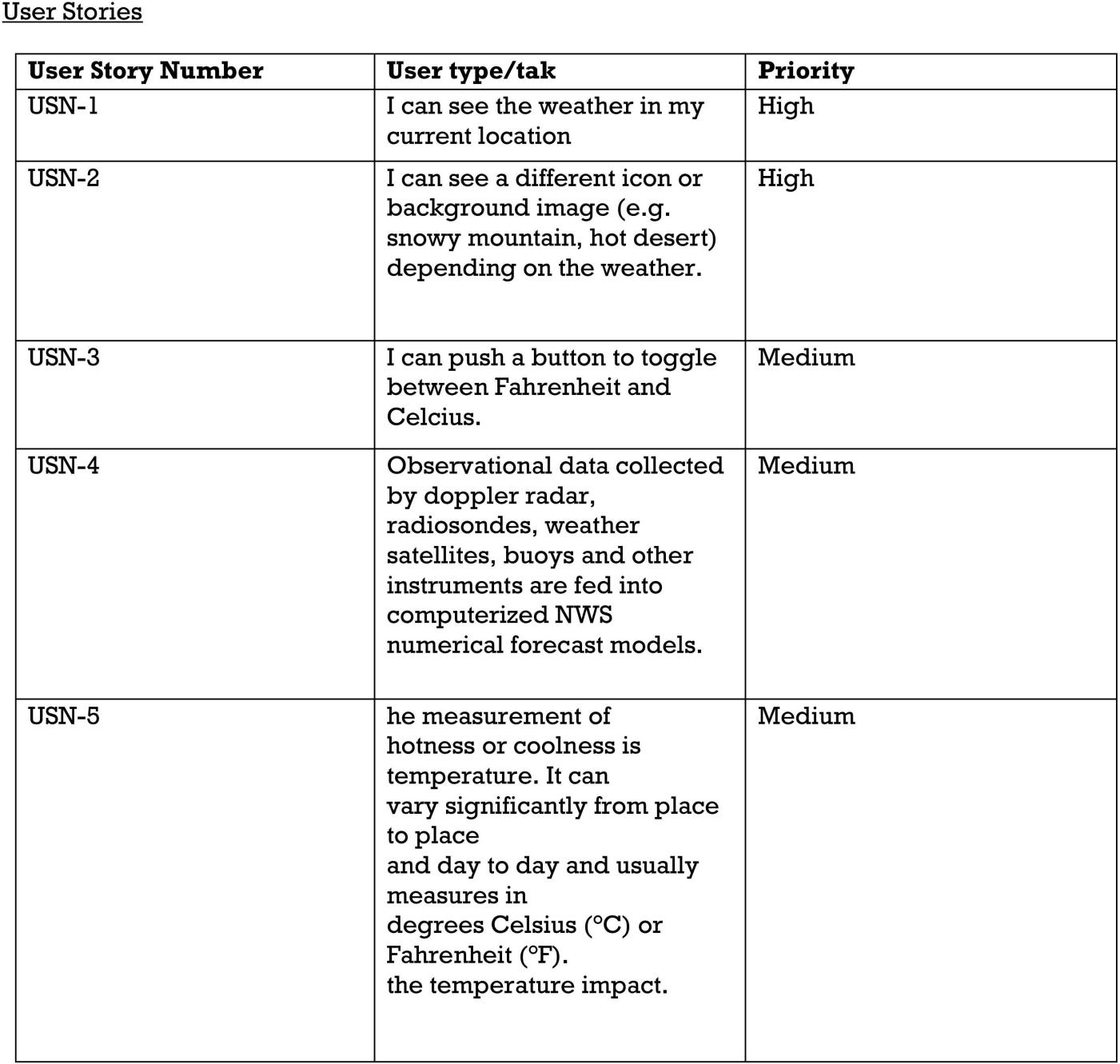
System's operational and location of remote station and central station shall not violate the current Government regulations of environment.The weather sensors shall be able to be upgraded every 5 years.



# PROJECT DESIGN

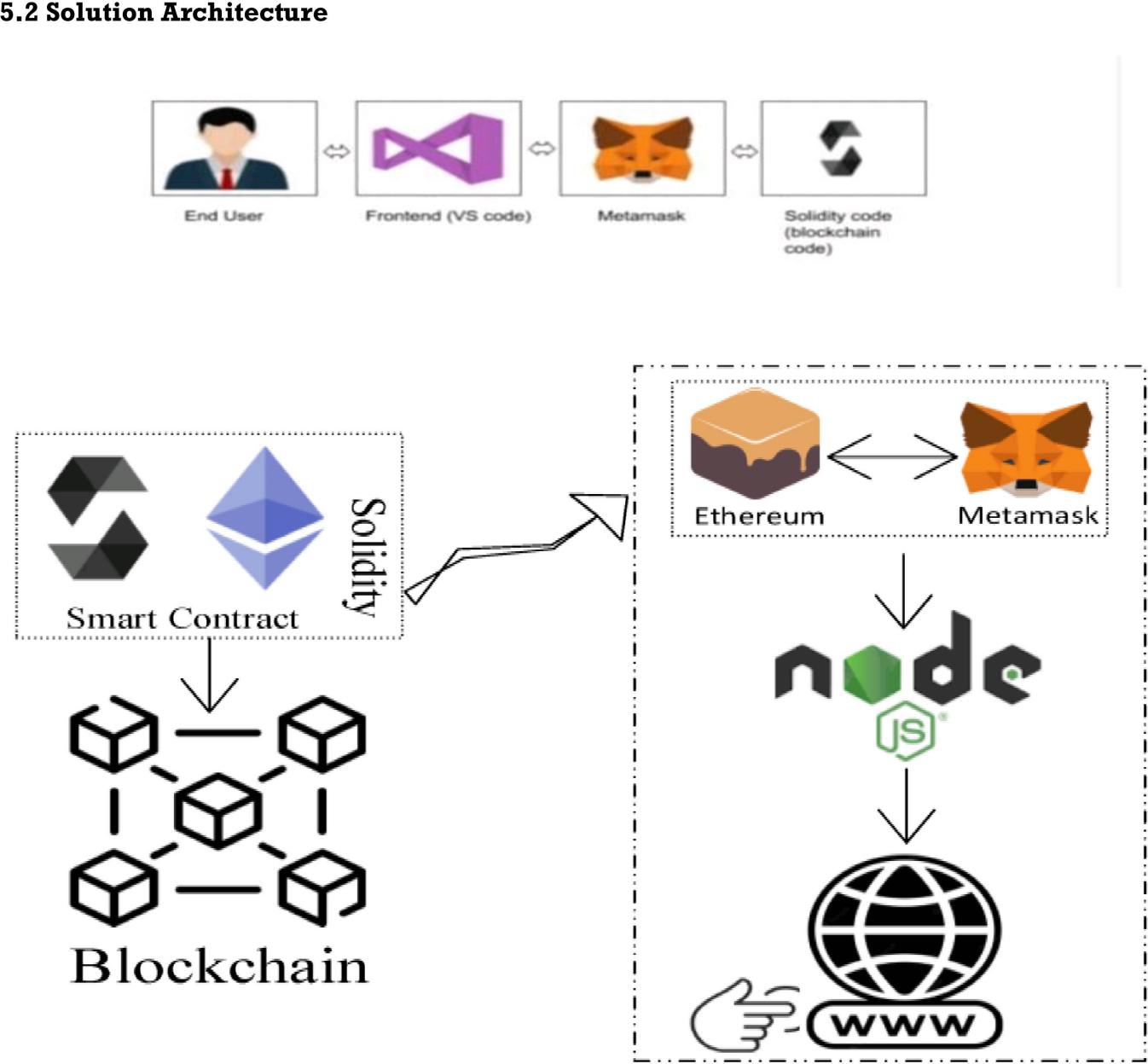
**5.1 Data Flow Diagrams & User Stories:**

****



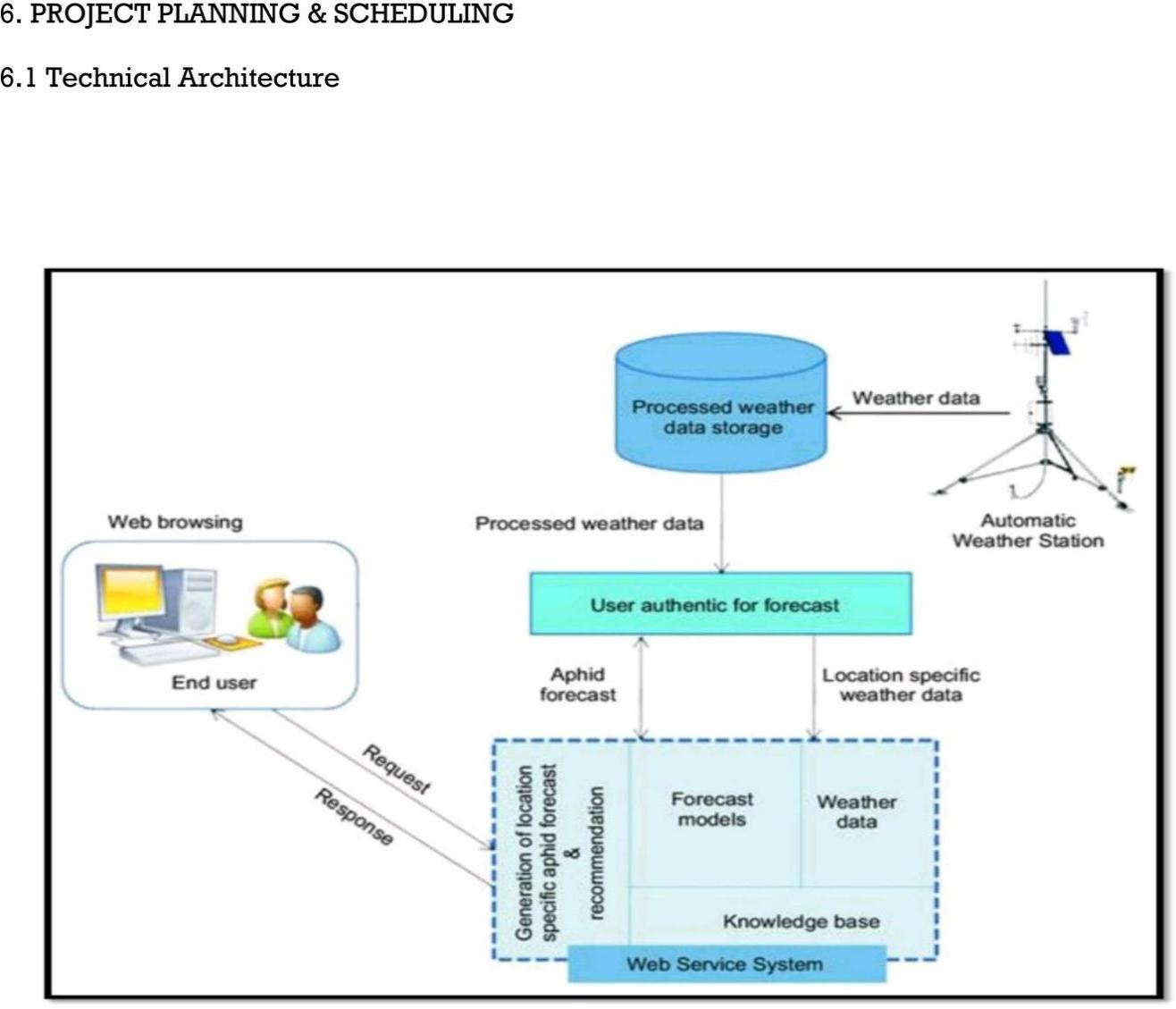
# 

# 5.2 Solution Architecture:

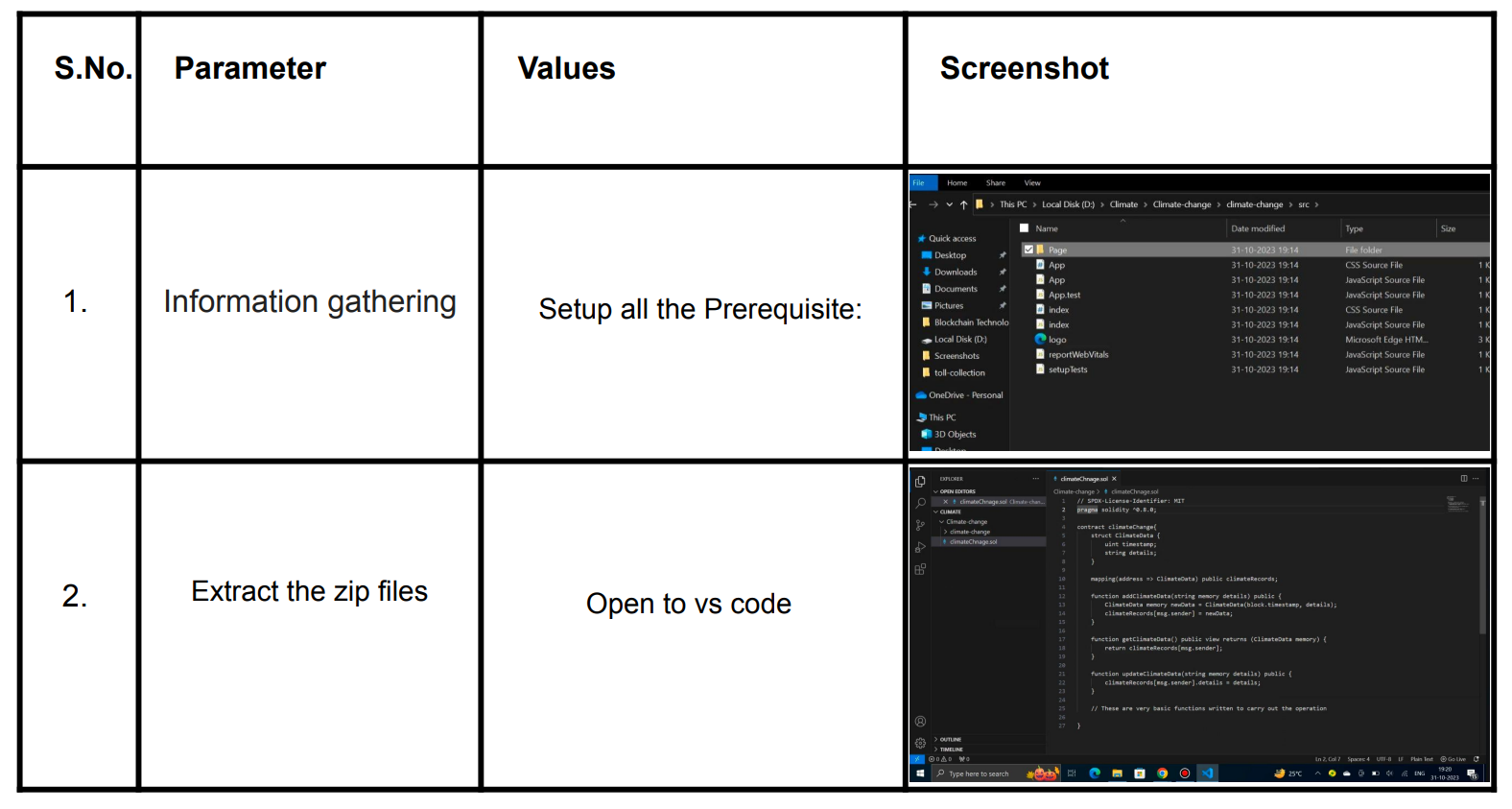


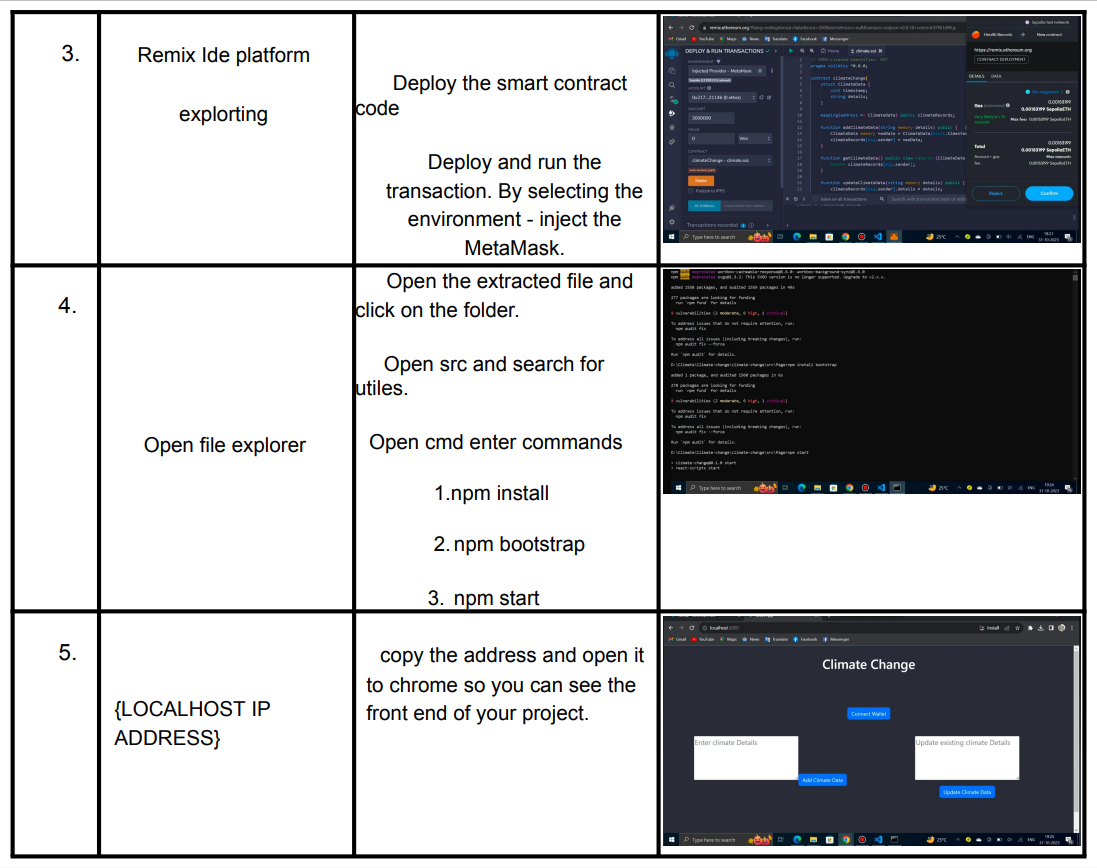
# PROJECT PLANNING & SCHEDULING

# 6.1 Technical Architecture:



**6.2 Project Development Phase:**





**7. CODING & SOLUTIONING**

**package.json**

{

  "name": "client",

  "version": "0.1.0",

  "private": true,

  "dependencies": {

    "react": "^17.0.2",

    "react-dom": "^17.0.2",

    "react-scripts": "3.2.0",

    "web3": "^1.6.1"

  },

  "scripts": {

    "start": "react-scripts start",

    "build": "react-scripts build",

    "test": "react-scripts test",

    "eject": "react-scripts eject"

  },

  "eslintConfig": {

    "extends": "react-app"

  },

  "browserslist": {

    "production": [

      ">0.3%",

      "not ie 11",

      "not dead",

      "not op\_mini all"

    ],

    "development": [

      "last 1 chrome version",

      "last 1 firefox version",

      "last 1 safari version",

      ">0.3%",

      "not ie 11",

      "not dead",

      "not op\_mini all"

    ]

  }

}

**App.js**

import React, { Component } from "react";

import Entry from "./contracts/Entry.json";

import getWeb3 from "./getWeb3";

import Navbar from "./components/Navbar/index";

import Dashboard from "./components/Dashboard";

import "./App.css";

class App extends Component {

  state = { web3: null, accounts: null, contract: null };

  componentDidMount = async () => {

    try {

      // Get network provider and web3 instance.

      const web3 = await getWeb3();

      // Use web3 to get the user's accounts.

      const accounts = await web3.eth.getAccounts();

      // Get the contract instance.

      const networkId = await web3.eth.net.getId();

      const deployedNetwork = Entry.networks[networkId];

      const instance = new web3.eth.Contract(

        Entry.abi,

        deployedNetwork && deployedNetwork.address

      );

      // Set web3, accounts, and contract to the state

      this.setState({ web3, accounts, contract: instance });

    } catch (error) {

      // Catch any errors for any of the above operations.

      console.error(error);

    }

  };

  render() {

    if (!this.state.web3) {

      return (

        <>

          <div className="App">

            <div className="message-container">Please log in with Metamask</div>

          </div>

        </>

      );

    }

    return (

      <>

        <Navbar address={this.state.accounts} />

        <div className="App">

          <Dashboard web3={this.state.web3} accounts={this.state.accounts} />

        </div>

      </>

    );

  }

}

export default App;

**getWeb3.js**

import Web3 from "web3";

const getWeb3 = () =>

  new Promise((resolve, reject) => {

    window.addEventListener("load", async () => {

      // Modern dapp browsers...

      if (window.ethereum) {

        const web3 = new Web3(window.ethereum);

        try {

          // Request account access if needed

          await window.ethereum.enable();

          // Accounts now exposed

          resolve(web3);

        } catch (error) {

          reject(error);

        }

      }

      // Legacy dapp browsers...

      else if (window.web3) {

        // Use Mist/MetaMask's provider.

        const web3 = window.web3;

        console.log("Injected web3 detected.");

        resolve(web3);

      }

      // Fallback to localhost; use dev console port by default...

      else {

        const provider = new Web3.providers.HttpProvider(

          "http://127.0.0.1:8545"

        );

        const web3 = new Web3(provider);

        console.log("No web3 instance injected, using Local web3.");

        resolve(web3);

      }

    });

  });

export default getWeb3;

**components/Dashboard/index.j**

import React, { Component } from "react";

import Entry from "../../contracts/Entry.json";

import "./Dashboard.css";

import Module from "../Module";

class Dashboard extends Component {

  state = {

    web3: null,

    accounts: "",

    contract: null,

    itemId: "",

    item: "",

    weight: "",

    destination: "",

    status: "",

fromOrgID: "",

    fromAddress: "",

    toOrgID: "",

    toAddress: "",

  };

  componentDidMount = async () => {

    const web3 = this.props.web3;

    // Use web3 to get the user's accounts.

    const accounts = await web3.eth.getAccounts();

    // Get the contract instance.

    const networkId = await web3.eth.net.getId();

    const deployedNetwork = Entry.networks[networkId];

    const instance = new web3.eth.Contract(

      Entry.abi,

      deployedNetwork && deployedNetwork.address

    );

    // Set web3, accounts, and contract to the state

    this.setState({

      web3,

      accounts,

      contract: instance,

      fromAddress: accounts[0],

      toAddress: instance.\_address,

    });

  };

  submitEntry = async (e) => {

    e.preventDefault();

    const {

      accounts,

      contract,

      itemId,

      item,

      weight,

      destination,

      status,

      fromOrgID,

      fromAddress,

      toOrgID,

      toAddress,

    } = this.state;

    // interact and write to contract

    await contract.methods

      .createItem(

        itemId,

        item,

        weight,

        destination,

        status,

        fromOrgID,

        fromAddress,

        toOrgID,

        toAddress

      )

      .send({

        from: accounts[0],

      });

  };

  handleItemId = (event) => {

    this.setState({ itemId: event.target.value });

  };

  handleItem = (event) => {

this.setState({ item: event.target.value });

  };

  handleWeight = (event) => {

    this.setState({ weight: event.target.value });

  };

  handleDestination = (event) => {

    this.setState({ destination: event.target.value });

  };

  handleStatus = (event) => {

    this.setState({ status: event.target.value });

  };

  handleFromOrgID = (event) => {

    this.setState({ fromOrgID: event.target.value });

  };

  handleFromAddress = (event) => {

    this.setState({ fromAddress: event.target.value });

  };

  handleToOrgID = (event) => {

    this.setState({ toOrgID: event.target.value });

  };

  handleToAddress = (event) => {

    this.setState({ toAddress: event.target.value });

  };

  render() {

    return (

      <form onSubmit={this.submitEntry}>

        <div className="dashboard-wrapper">

          <Module>

            <label>Item ID</label>

            <input

              required

              value={this.state.itemId}

              onChange={this.handleItemId}

            />

            <label>Item</label>

            <input

              required

              value={this.state.item}

              onChange={this.handleItem}

            />

            <label>Weight</label>

            <input

              required

              value={this.state.weight}

              onChange={this.handleWeight}

            />

            <label>Destination / Region</label>

            <input

              required

              value={this.state.destination}

              onChange={this.handleDestination}

            />

            <label>Status</label>

            <input

              required

              value={this.state.status}

              onChange={this.handleStatus}

            />

          </Module>

          <div className="send-details">

            <Module>

  <label>To: Organization ID</label>

              <input

                required

                value={this.state.fromOrgID}

                onChange={this.handleFromOrgID}

              />

              <label>To: Wallet Address</label>

              <input

                required

                value={this.state.fromAddress}

                onChange={this.handleFromAddress}

              />

            </Module>

            <Module>

              <label>From: Organization ID</label>

              <input

                required

                value={this.state.toOrgID}

                onChange={this.handleToOrgID}

              />

              <label>From: Wallet Address</label>

              <input

                required

                value={this.state.toAddress}

                onChange={this.handleToAddress}

              />

              <button type="submit" className="mod-btn">

                Send

              </button>

            </Module>

          </div>

        </div>

      </form>

    );

  }

}

export default Dashboard;

**components/Module/index.js**

import React, { Component } from "react";

import "./Module.css";

class Module extends Component {

  render() {

    return <div className="mod-container">{this.props.children}</div>;

  }

}

export default Module;

**components/Navbar/index.js**

import React, { Component } from "react";

import "./Navbar.css";

import Logo from "../../logo.svg";

class Navbar extends Component {

  state = { web3: null, accounts: null, contract: null };

  render() {

    let walletAddress = this.props.address ? (

      <div className="navbar-mid">Connected: {this.props.address}</div>

    ) : (

      <div className="navbar-mid">Not Connected</div>

    );

    return (

      <div className="container">

        <div>

          <img className="logo" src={Logo} alt="logo" />

        </div>

        {walletAddress}

        <div className="navbar-end"></div>

      </div>

    );

  }

}

export default Navbar;

**contracts/Migrations.sol**

pragma solidity >=0.4.21 <8.10.0;

contract Migrations {

    address public owner;

    uint public last\_completed\_migration;

    modifier restricted() {

        if (msg.sender == owner) \_;

    }

    constructor() public {

        owner = msg.sender;

    }

    function setCompleted(uint completed) public restricted {

        last\_completed\_migration = completed;

    }

}

**contracts/Entry.sol**

pragma solidity >=0.4.21 <8.10.0;

contract Entry {

    struct Item {

        string itemId;

        string itemName;

        uint weight;

        string destination;

        string status;

        uint toOrgID;

        address toAddress;

        uint fromOrgID;

        address fromAddress;

    }

    address public contractUser;

    Item[] public items;

    constructor() public {

        contractUser = msg.sender;

    }

    function createItem(

        string memory \_itemId,

        string memory \_itemName,

        uint \_weight,

        string memory \_destination,

        string memory \_status,

        uint \_toOrgID,

        address \_toAddress,

        uint \_fromOrgID,

        address \_fromAddress

    ) public returns (bool success) {

        Item memory itemToBeAdded = Item({

            itemId: \_itemId,

            itemName: \_itemName,

            weight: \_weight,

            destination: \_destination,

            status: \_status,

            toOrgID: \_toOrgID,

            toAddress: \_toAddress,

            fromOrgID: \_fromOrgID,

            fromAddress: \_fromAddress

        });

        items.push(itemToBeAdded);

        return true;

    }

    function showItems() public view returns (uint length) {

        for (uint i = 0; i < items.length; i++) {

            items[i];

        }

        return items.length;

    }

}

# 8. PERFORMANCE TESTING

**8.1 Performance Metrics:**

**Transaction Throughput:**

Measure the rate at which transactions, such as data uploads and updates, are processed within the blockchain network. Higher TPS indicates better throughput.

**Data Retrieval Speed:**

Assess the time it takes to retrieve climate data from the blockchain. Faster data retrieval is essential for real-time decision-making and analysis.

**Blockchain Latency:**

Evaluate the time taken for a new block to be added to the blockchain. Lower confirmation times contribute to reduced latency and quicker data updates.

**Consensus Efficiency:**

Analyze the efficiency of the consensus algorithm used in the blockchain network. Ensure that it strikes a balance between security and performance.

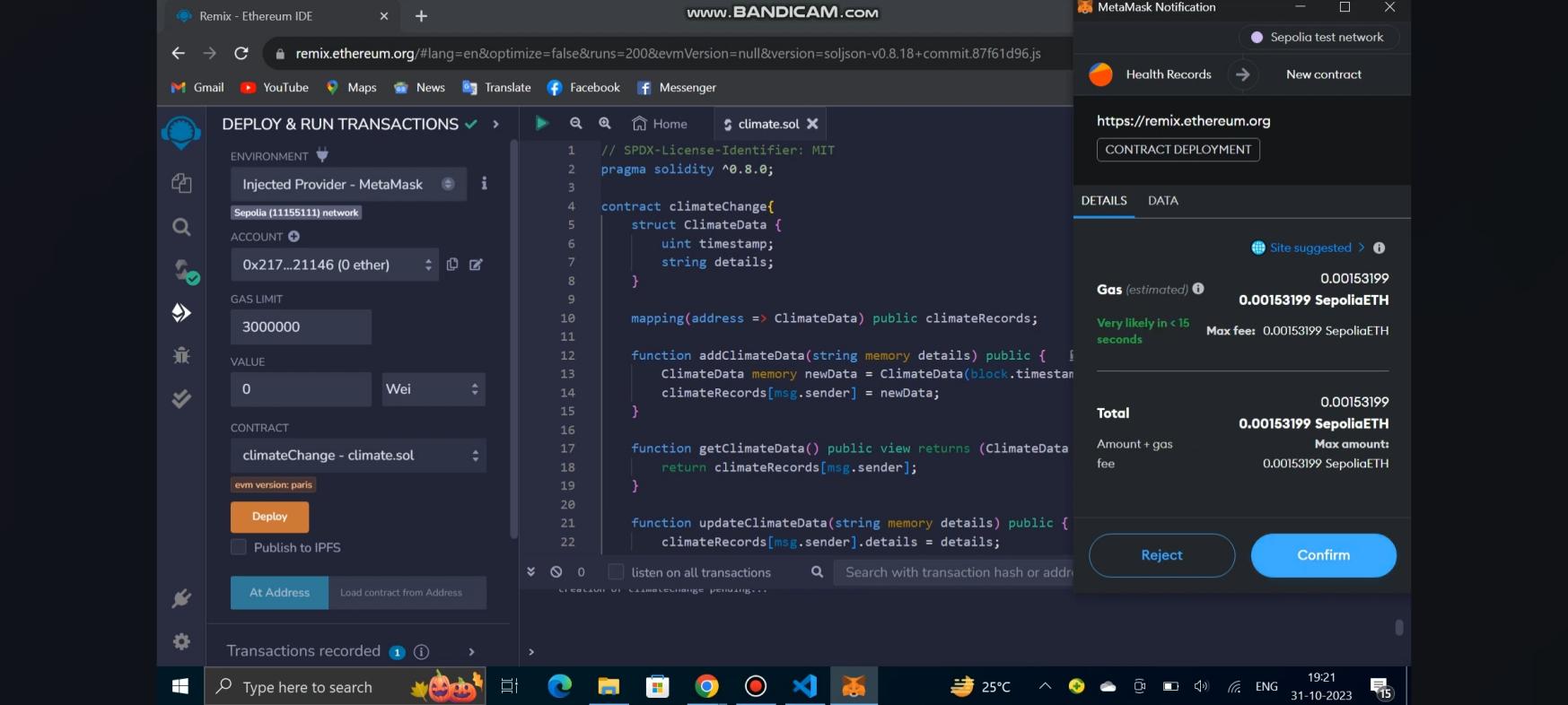
**Scalability:**

Measure the system's ability to handle an increasing volume of climate data. Assess scalability both horizontally (adding more nodes) and vertically (increasing resources of existing nodes).

# RESULT

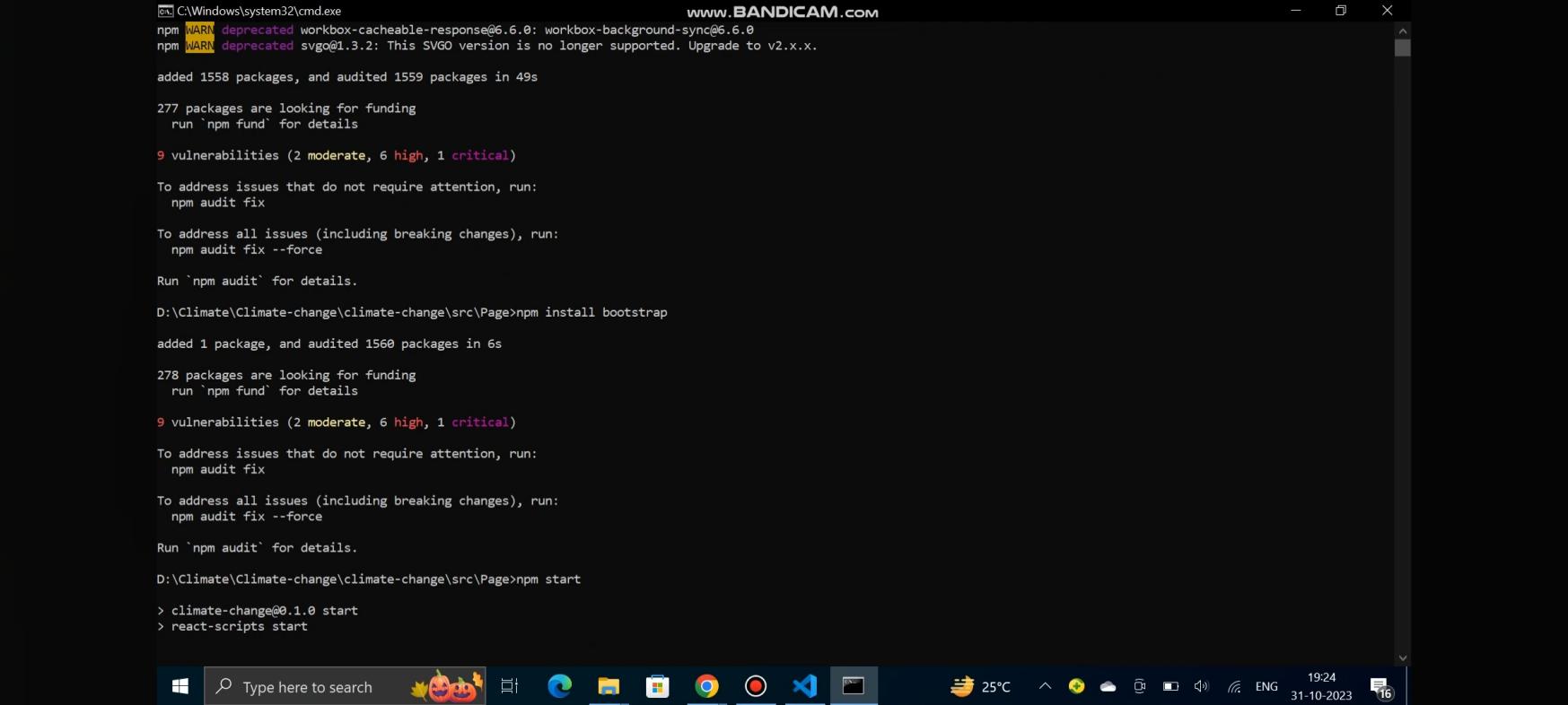
1). Deploy the smart contract code Deploy and run the transaction. By selecting the environment - inject the

MetaMask.

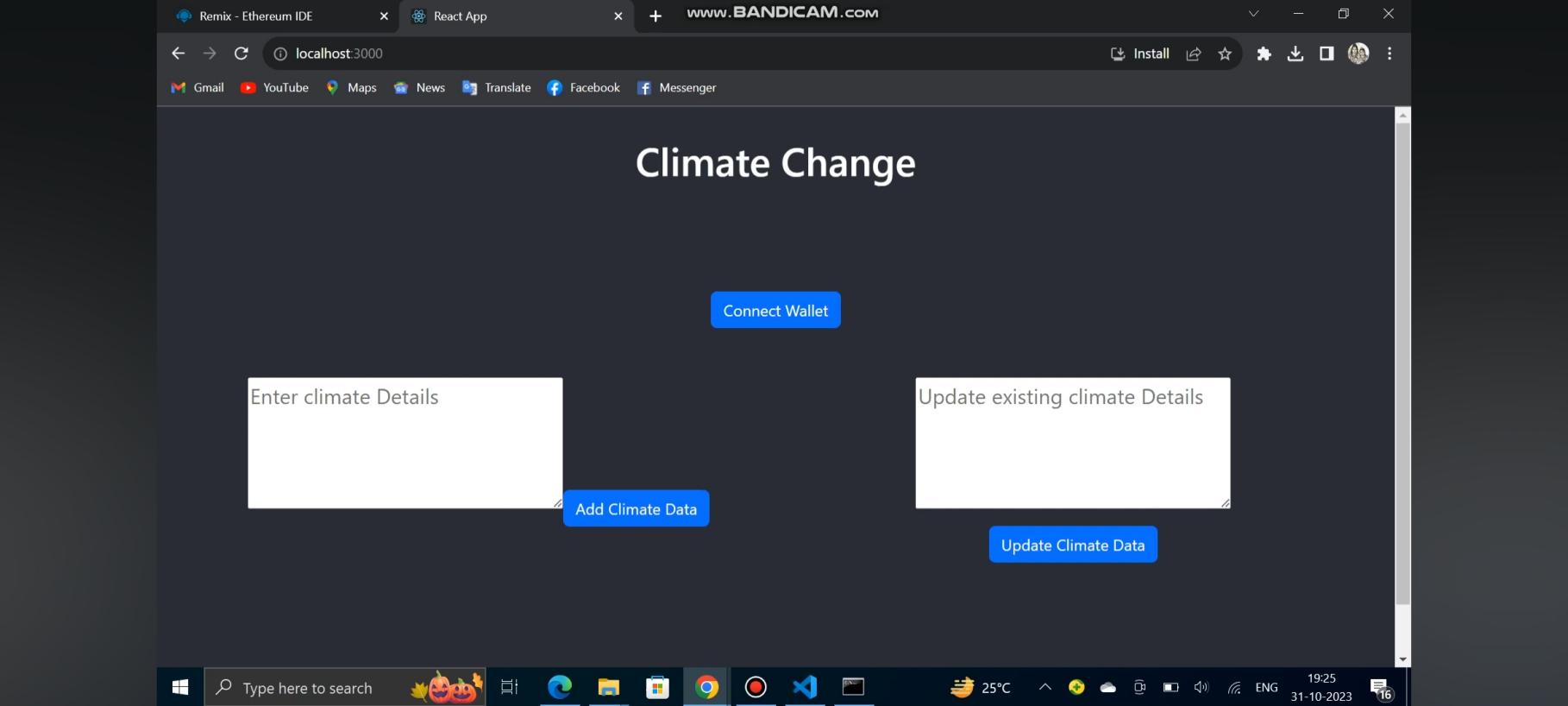
****

2). Open the extracted file and click on the folder. Open src and search for utiles. Open cmd enter

Commands 1.npm install 2. npm bootstrap 3. npm start

****

1. **.**copy the address and open it to chrome so you can see the front end of your project



# ADVANTAGES & DISADVANTAGES

**Advantages of the Climate Tracksmart using blockchain:**

**1)Transparency and Traceability:**

Immutable Records: Blockchain ensures that once climate data is recorded, it cannot be altered . This immutability provides a transparent and tamper-proof record of all transactions, fostering trust among stakeholders.

Traceable Data History: The decentralized ledger allows users to trace the entire history of climate data transactions, providing a comprehensive and auditable view of the data's journey.

**2)Data Integrity and Security:**

Tamper-Proof Data: The use of blockchain technology ensures the integrity of climate-related information. Any attempt to tamper with data is immediately detectable, maintaining the reliability of the stored information.

Enhanced Security: Blockchain's cryptographic features enhance the security of climate data, protecting it from unauthorized access and ensuring that only authorized parties can contribute or access specific information.

**3)Collaboration and Decentralization:**

Decentralized Network: The decentralized nature of blockchain eliminates the need for a central authority, promoting a collaborative and inclusive environment for stakeholders.

Secure Data Sharing: Blockchain facilitates secure and transparent data sharing among diverse stakeholders, including environmental agencies, researchers, policymakers, and the public. This promotes collaboration and knowledge exchange in the realm of climate science.

**4)Efficiency and Streamlined Processes:**

Real-time Updates: Climate TrackSmart enables real-time tracking and updating of climate data. This timely information can be crucial for decision-making and responding promptly to emerging climate trends.

Automated Smart Contracts: Smart contracts on the blockchain can automate certain processes, such as data validation and verification, reducing the need for manual interventions and streamlining workflows.

**5)User-Friendly Access and Adoption:**

Intuitive Interface: The platform is designed with a user-friendly interface, making it accessible to a wide range of users, including those without extensive technical expertise.

Broad Stakeholder Participation: The user-friendly nature of Climate TrackSmart encourages broad participation from stakeholders, fostering a more inclusive and collaborative approach to climate data management.

# Disadvantages of the Climate Tracksmart using blockchain :

**1)Energy Consumption:**

Resource-Intensive Consensus Mechanisms: Many blockchain networks rely on energy-intensive consensus mechanisms (e.g., Proof of Work). This can contribute to environmental concerns and may conflict with the goal of sustainable climate management.

**2)Scalability Issues:**

Blockchain Size and Transaction Throughput: As the volume of climate data increases, the size of the blockchain and the speed at which transactions can be processed might become limiting factors. Scaling blockchain networks can be a complex and challenging task.

**3)Complexity and Technical Expertise:**

Learning Curve: Understanding and implementing blockchain technology requires a certain level of technical expertise. Stakeholders may face challenges in adapting to the complexities of blockchain, potentially hindering widespread adoption.

**4)Regulatory Uncertainty:**

Evolving Regulatory Landscape: The regulatory environment for blockchain technology is continually evolving. Compliance with existing and future regulations can be challenging, especially in a field as sensitive as climate data management.

**5)Data Privacy Concerns:**

Immutability and GDPR Compliance: The immutability of blockchain data may pose challenges in adhering to data privacy regulations like the General Data Protection Regulation (GDPR). Ensuring compliance with such regulations can be complex.

**6)Initial Implementation Costs:**

Infrastructure and Development Costs: Implementing blockchain technology involves initial costs related to infrastructure setup, development, and integration. These costs can be a barrier, particularly for organizations with limited financial resources.

# CONCLUSION

In conclusion, "Climate TrackSmart using Blockchain" emerges as a transformative and forward-looking solution to the pressing challenges in climate data management. By harnessing the power of blockchain technology, the project addresses critical issues of transparency, data integrity, and collaboration, essential for effective climate monitoring and decision-making.

The advantages of the system, including enhanced transparency, security, and decentralized collaboration, position it as a catalyst for positive change in the realm of climate science. The immutability of blockchain ensures the integrity of climate data, fostering trust among stakeholders and promoting a shared understanding of environmental trends. The user-friendly design encourages broad stakeholder participation, paving the way for a more inclusive and collaborative approach to climate data management.

However, it is crucial to acknowledge the potential disadvantages, such as energy consumption and scalability challenges, and address them in the ongoing development and implementation phases. Striking a balance between decentralization and control, overcoming technical complexities, and ensuring regulatory compliance will be key factors in the project's long-term success.

As we navigate the dynamic landscape of climate science and technology, "Climate TrackSmart using Blockchain" stands as a beacon of innovation, offering a resilient and secure foundation for the sustainable management of climate-related information. The project's future holds the promise of continued evolution, collaboration, and positive impact on our collective efforts to address the complex challenges posed by climate change.

# FUTURE SCOPE

The future scope for "Climate TrackSmart using Blockchain" extends beyond its initial implementation, presenting opportunities for expansion, enhancement, and integration with emerging technologies. Here are key avenues for the project's future development:

**1)Integration with Emerging Technologies:**

AI and Machine Learning: Explore the integration of artificial intelligence (AI) and machine learning (ML) algorithms to derive insights and patterns from climate data. This could enhance predictive capabilities and support informed decision-making.

**2)Enhanced Data Visualization:**

Interactive Dashboards: Develop interactive and user-friendly dashboards that leverage data visualization techniques. This can empower stakeholders to interpret complex climate data more intuitively.

**3)Blockchain Sustainability:**

Energy-Efficient Consensus Mechanisms: Investigate and implement energy-efficient consensus mechanisms to address concerns related to the environmental impact of blockchain technology. This aligns with sustainable practices and reduces the carbon footprint.

**4)Decentralized Applications (DApps)**:

Expand to DApps: Consider the development of decentralized applications (DApps) on the blockchain. These could offer additional functionalities and services, expanding the utility of the Climate TrackSmart platform.

**5)Smart Contracts for Climate Agreements:**

Climate Smart Contracts: Develop smart contracts to automate and enforce climate agreements or contracts, ensuring that agreed-upon actions for environmental conservation are executed automatically when conditions are met.

# APPENDIX

**GitHub:**

<https://github.com/Inbaraj7/NM2023TMID11936>

**Source code :** <https://github.com/Inbaraj7/NM2023TMID11936/tree/830f02d27522fdd4e631047dcaa41255fd632d7c/Final%20Deliverables/Climate%20TrackSmart%20using%20blockchain(Source%20Code)>

**Project Demo Link :**

[https://drive.google.com/file/d/1F1CRnbTo5i-\_ y9j64H5IDGiak8pkTmjF/view?usp=sharing](https://drive.google.com/file/d/1F1CRnbTo5i-_y9j64H5IDGiak8pkTmjF/view?usp=sharing)