

INDEX

1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

2. EXISTING SYSTEM

2.1 Existing Problem

2.2 Problem Statement Definition

3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation and Brainstorming

4. REQUIRMENT ANALYSIS

4.1 Functional Requirements

4.2 Non-Functional Requirements

5. PROJECT DESIGN

5.1 Data Flow Diagram and User Stories

5.2 Solution Architecture

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

6.2 Sprint Planning and Estimation

6.3 Sprint Delivering

7. CODING AND SOLUTIONING

7.1 Feature 1

7.2 Feature 2

8. PERFORMANCE TESTING

8.1 Performance Metrics

9. RESULTS

9.1 Output Screenshots

10. ADVANTAGES AND DISADVANTAGES

10.1 Advantages

10.2 Disadvantages

11. CONCLUSION

12. FUTURE SCOPE

TRANSPARENT TOLL-FREE DATA MANAGEMENT

1.INTRODUCTION

1.1 PROJECT OVERVIEW

The "Transparent Toll-Free Data Management" project is designed to address the growing need for effective management and transparency in the use of toll-free numbers. Toll-free numbers have become a vital component of businesses and organizations, serving as a direct channel for customer communication, support, and marketing. However, the management of the data associated with these numbers has often been fragmented and lacking in transparency.

This project seeks to provide a comprehensive solution that will revolutionize how toll-free numbers are utilized and managed. By introducing a transparent data management system, organizations can gain valuable insights into their toll-free number usage, ensuring they are used optimally to meet customer needs and business goals.

The primary objectives of this project include:

1. Implementing a centralized system for the management of toll-free numbers and associated data.
2. Enhancing the transparency of data related to toll-free number usage, expenses, and performance.
3. Providing tools and analytics for businesses to make data-driven decisions regarding toll-free number allocation and optimization.
4. Ensuring the security and privacy of customer data and call records.

The success of this project will empower businesses to harness the full potential of toll-free numbers, improving customer satisfaction and operational efficiency, while simultaneously ensuring compliance with regulations and data security standards.

1.2 PURPOSE

The purpose of this project is to create a system that enhances the transparency and efficiency of toll-free data management. By implementing this solution, businesses and organizations can track, analyze, and optimize the usage of toll-free numbers, leading to better customer service and improved business processes..

2.EXISTING SYSTEM

2.1EXISTING PROBLEM

In this section, outline the challenges and issues with the current toll-free data management systems. These could include:

- Lack of transparency in tracking toll-free number usage.
- Inefficiencies in data collection, storage, and analysis.
- Difficulty in ensuring data privacy and security.
- Inadequate tools for optimizing toll-free number allocation.

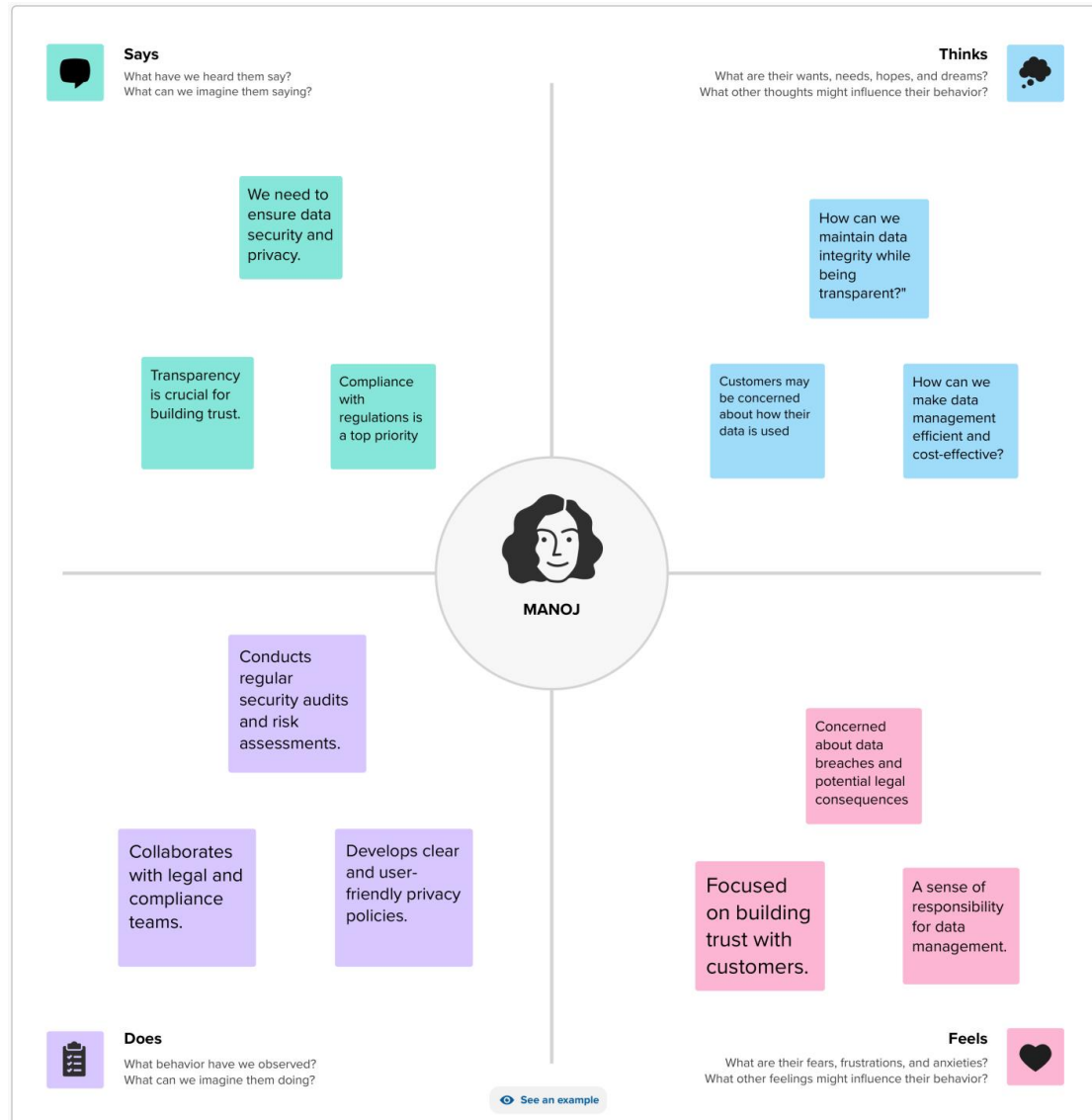
2.2PROBLEM STATEMENT

- The existing system lacks a centralized platform for monitoring and managing toll-free numbers, leading to data fragmentation and inaccuracies.
- Data privacy concerns arise due to the absence of robust security measures for call records and customer information.
- Inefficient data analysis results in missed opportunities to optimize toll-free number usage, leading to higher costs and potentially unsatisfied customers.

These are just placeholders for the existing system section. You should fill in the specific problems and details related to the current state of toll-free data management in your project context.

3.TRANSPERENT TOLL-FREE DATA MANAGEMENT

3.1EMPATHY MAP CANVAS



3.2 IDEATION AND BRAINSTORMING


Step-1: Team Gathering, Collaboration and Select the Problem Statement



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

 **10 minutes** to prepare

 **1 hour** to collaborate

 **2-8 people** recommended



Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

 **10 minutes**

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →

Step-2 Brainstorm,Idea Listing

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

TIP

TIP You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Person 1

Biotechnology
 While biotechnology is widely an interdisciplinary topic for some biologists, this journal is more likely to be of interest to biologists who are interested in the application of biotechnology to other disciplines and to the development of new products.

Person 2

Share code in a distributed manner across multiple servers or nodes, reducing the risk of data breaches. Implementers are checking for subtle security

Person 3

Shift the privilege to their own work and control their data. They can query and modify across, leading to more transparency and research-based data sharing.

Person 4

Implement
continuous social
skills that work among
peers in class. This
can be made quickly
available to provide
complete transparency
about their usage.

Person 5

Developed and implemented privacy-preserving algorithms and techniques, and other data analysis without engineering into sensitive data

Type your heading...

Grouping

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

TIP



Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

4.REQUIREMENT ANALYSIS

4.1FUNCTIONAL REQUIREMENTS

Functional requirements describe the specific features and capabilities that the system must possess to fulfill its purpose. In the context of transparent toll-free data management, some functional requirements might include:

User Registration and Authentication:

Users should be able to register and log in securely.

Toll-Free Number Management:

The system must support the allocation, tracking, and real-time status monitoring of toll-free numbers.

Data Analytics:

It should provide tools for data analysis, allowing businesses to make informed decisions regarding toll-free number allocation.

Data Storage:

Securely store call records and customer data while ensuring compliance with data privacy regulations.

Reporting and Visualization:

Generate reports and visual representations of data for easy understanding.

Notification and Alerts:

Send notifications and alerts for unusual activities or issues related to toll-free numbers.

Integration:

The system should be able to integrate with existing CRM or customer support systems.

4.2NON FUNCTIONAL REQUIREMENT:

Non-functional requirements define the quality attributes and constraints of the system. Some non-functional requirements for this project might include:

Security:

Ensure data security and compliance with data privacy regulations.

Scalability:

It should be scalable to handle a growing number of toll-free numbers and data.

Usability:

The user interface should be intuitive and user-friendly.

Reliability:

Ensure high system availability to minimize downtime.

Compliance:

The system must comply with industry standards and regulations related to toll-free number management.

Interoperability:

It should be able to integrate with various telecommunication systems and platforms.

Backup and Recovery:

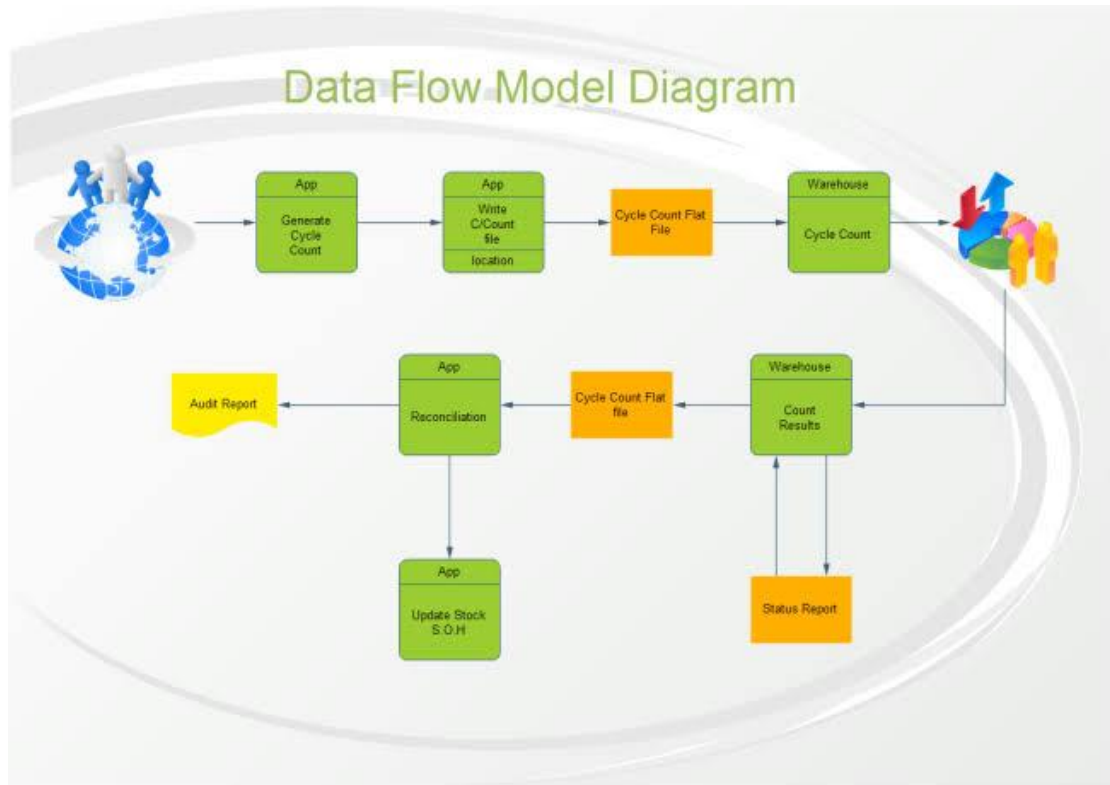
Implement a robust backup and recovery mechanism to safeguard data (e.g., GDPR).

Performance:

The system should provide real-time data access and analysis.

5.PROJECT DESIGN

5.1DATAFLOW DIAGRAM AND USER STORY



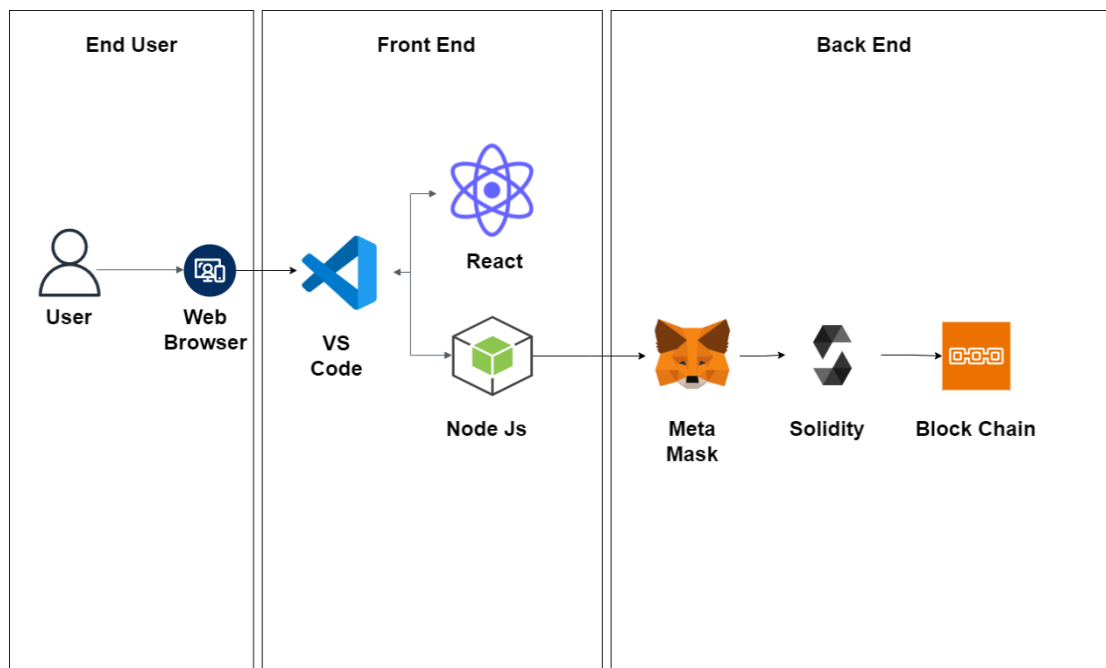
Data Flow Diagram:

Present a high-level data flow diagram to showcase how data moves through the system, from input to output.

User Stories:

Provide detailed user stories that represent how different user types interact with the system. These stories should capture the user's perspective and goals.

5.2 SOLUTION ARCHITECTURE



End User:

- This is where users interact with the blockchain application. It can be a web app.
- The voting site can be accessed via the browsers from all the devices by every user

Front End:

- React js - allows to create an interactive webpage which displays content for the end-user through the web browser through this the data representation is done
- Node js - A JavaScript library that enables the frontend to interact with the blockchain. It communicates with the blockchain node and communicates the data from user to blockchain and viceversa.

Back End:

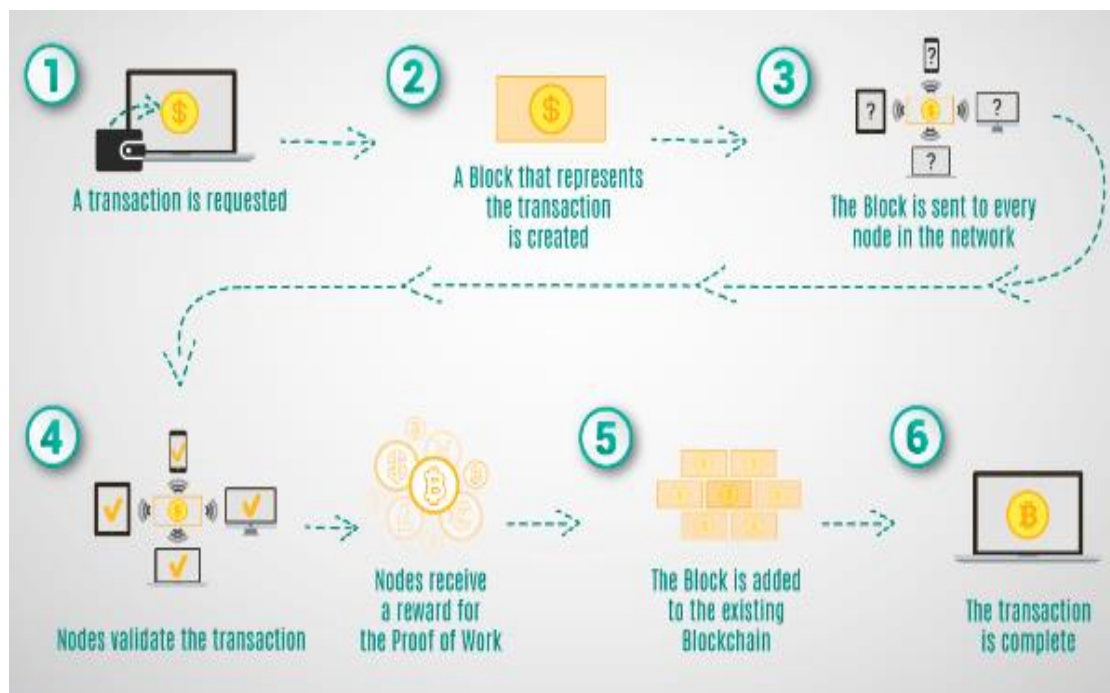
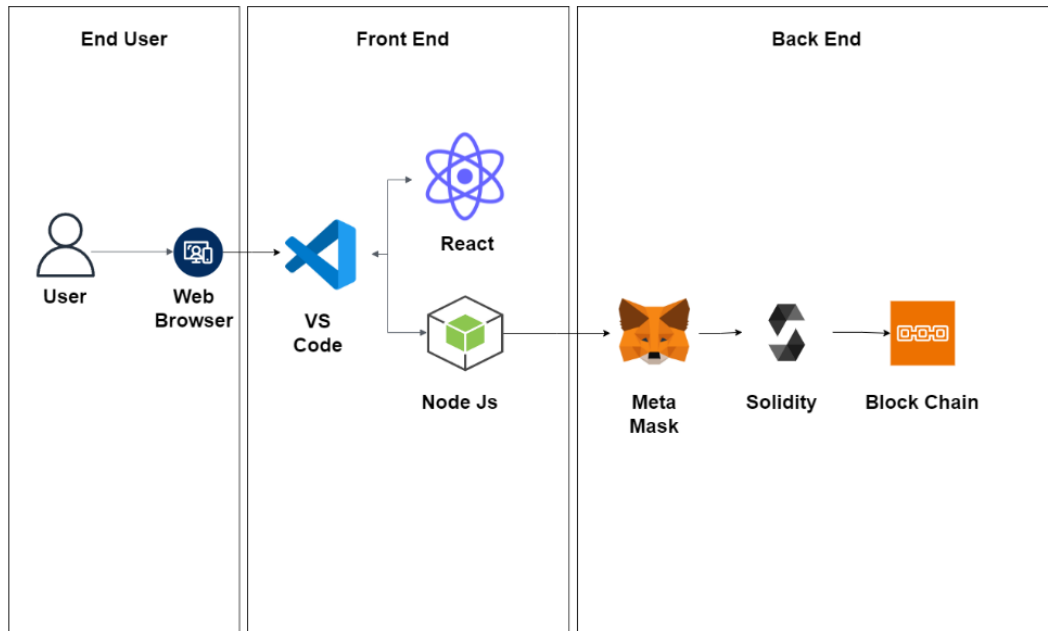
- Meta mask - simplifies the process of user authentication and transaction signing for blockchain-based applications. It allows users to securely interact with the Ethereum blockchain and DApps while keeping their private keys safe.
- Solidity(Remix ide) - Solidity is a high-level, statically-typed programming language used for developing smart contracts on various blockchain platforms, with Ethereum being the most prominent. Smart contracts are self-executing contracts with the terms of the agreement directly written into code.

- Remix IDE - is an essential tool for Solidity developers and is widely used in the Ethereum ecosystem. It simplifies the smart contract development process and provides many useful features for coding, testing, and deploying contracts on the Ethereum blockchain.
- Block Chain - Blockchain is a distributed and decentralized digital ledger technology that is used to record transactions across multiple computers in a way that ensures the security, transparency, and immutability of the data.

•

6.PROJECT PLANNING & SCHEDULING

6.1 TECHNICAL ARCHITECTURE



6.2Sprint Planning and Estimation

Sprint Planning:

Explain how you divide the project into sprints (iterations) and the goals of each sprint.

Estimation:

Detail the effort estimation for each task or feature to be completed during the sprints.

6.3 Sprint Delivering Schedule

Outline how project delivery and progress will be monitored and managed, including:

Milestone Tracking:

Describe how project milestones and progress will be tracked.

Quality Assurance:

Explain how quality control and testing will be implemented.

Risk Management:

Discuss how potential risks and issues will be identified and addressed.

7.CODING AND SOLUTIONING

7.1 FEATURE 1

```
pragma solidity ^0.8.0;
```

```
contract TollFreeNumberRegistry {
```

```
    struct TollFreeNumber {
```

```
        address owner;
```

```
        string phoneNumber;
```

```
        string serviceProvider;
```

```
        uint256 monthlyFee;
```

```
    }
```

```
    mapping(uint256 => TollFreeNumber) public tollFreeNumbers;
```

```
    uint256 public numberCount;
```

```
        event TollFreeNumberAdded(uint256 numberId, address owner, string  
phoneNumber, string serviceProvider, uint256 monthlyFee);
```

```
        event TollFreeNumberUpdated(uint256 numberId, string phoneNumber, string  
serviceProvider, uint256 monthlyFee);
```

```
    modifier onlyOwner(uint256 _numberId) {
```

```
        require(tollFreeNumbers[_numberId].owner == msg.sender, "Only the owner can  
perform this action");
```

```
        _;
```

```
    }
```

```
    function addTollFreeNumber(string memory _phoneNumber, string memory  
_serviceProvider, uint256 _monthlyFee) external {
```

```
        numberCount++;
```

```
        tollFreeNumbers[numberCount] = TollFreeNumber(msg.sender, _phoneNumber,  
_serviceProvider, _monthlyFee);
```

```

        emit TollFreeNumberAdded(numberCount, msg.sender, _phoneNumber,
        _serviceProvider, _monthlyFee);
    }

```

```

    function updateTollFreeNumber(uint256 _numberId, string memory
    _phoneNumber, string memory _serviceProvider, uint256 _monthlyFee) external
    onlyOwner(_numberId) {
        TollFreeNumber storage tollFreeNumber = tollFreeNumbers[_numberId];
        tollFreeNumber.phoneNumber = _phoneNumber;
        tollFreeNumber.serviceProvider = _serviceProvider;
        tollFreeNumber.monthlyFee = _monthlyFee;
        emit TollFreeNumberUpdated(_numberId, _phoneNumber, _serviceProvider,
        _monthlyFee);
    }

```

```

    function getTollFreeNumberDetails(uint256 _numberId) external view returns
    (address owner, string memory phoneNumber, string memory serviceProvider, uint256
    monthlyFee) {
        TollFreeNumber memory tollFreeNumber = tollFreeNumbers[_numberId];
        return (tollFreeNumber.owner, tollFreeNumber.phoneNumber,
        tollFreeNumber.serviceProvider, tollFreeNumber.monthlyFee);
    }

```

In this section, provide details on the implementation of the first feature of your solution. You can structure it as follows:

Feature Description:

Explain the purpose and functionality of the feature.

Development Process:

Describe how the feature was developed, including the technologies and methodologies used.

Challenges:

Highlight any challenges or issues faced during the development of this feature and how they were overcome.

Testing and Quality Assurance:

Discuss the testing and quality control procedures for this feature.

7.2 FEATURE 2

```
import React, { useState } from "react";
import { Button, Container, Row, Col } from 'react-bootstrap';
import 'bootstrap/dist/css/bootstrap.min.css';
import { contract } from "../connector";
```

```
function Home() {
  const [number, setNumber] = useState("");
  const [Provider, setProvider] = useState("");
  const [Fee, setFee] = useState("");
  // const [Expiration, setExpiration] = useState("");

  const [Ids, setIds] = useState("");
  const [numbers, setNumbers] = useState("");
  const [Providers, setProviders] = useState("");
  const [Fees, setFees] = useState("");

  const [gId, setGIds] = useState("");
  const [Details, setDetails] = useState("");
  const [Wallet, setWallet] = useState("");

  // const handlePolicyNumber = (e) => {
  //   setNumber(e.target.value)
  // }

  const handleNumber = (e) => {
    setNumber(e.target.value)
  }

  const handleProvider = (e) => {
    setProvider(e.target.value)
```

```
}
```

```
const handleFee = (e) => {  
  setFee(e.target.value)  
}
```

```
const handleAddToll = async () => {  
  try {  
    let tx = await contract.addTollFreeNumber(number, Provider, Fee.toString())  
    let wait = await tx.wait()  
    alert(wait.transactionHash)  
    console.log(wait);  
  } catch (error) {  
    alert(error)  
  }  
}
```

```
const handleNumbersIds = (e) => {  
  setIds(e.target.value)  
}
```

```
const handleNumbers = (e) => {  
  setNumbers(e.target.value)  
}
```

```
const handleProviders = (e) => {  
  setProviders(e.target.value)  
}
```

```
const handleFees = (e) => {  
  setFees(e.target.value)  
}
```

```
const handleUpdate = async () => {  
  try {  
    let tx = await contract.updateTollFreeNumber(Ids.toString(), numbers, Providers,  
Fees.toString())  
    let wait = await tx.wait()  
    console.log(wait);  
    alert(wait.transactionHash)  
  } catch (error) {  
    alert(error)  
  }  
}
```

```
const handleGetIds = async (e) => {  
  setGIds(e.target.value)  
}
```

```
const handleGetDetails = async () => {  
  try {  
    let tx = await contract.getTollFreeNumberDetails(gId.toString())  
  
    let arr = []  
    tx.map(e => {  
      arr.push(e)  
    })  
  
    console.log(tx);  
    setDetails(arr)
```

```

    } catch (error) {
      alert(error)
      console.log(error);
    }
  }
}

```

```

const handleWallet = async () => {
  if (!window.ethereum) {
    return alert('please install metamask');
  }

```

```

  const addr = await window.ethereum.request({
    method: 'eth_requestAccounts',
  });

```

```

  setWallet(addr[0])

```

```

}

```

```

return (

```

```

  <div>

```

```

    <h1 style={{ marginTop: "30px", marginBottom: "80px" }}>Toll Free Number</h1>
    {!Wallet ?

```

```

      <Button onClick={handleWallet} style={{ marginTop: "30px", marginBottom:
"50px" }}>Connect Wallet </Button>

```

```

      :

```

```

      <p style={{ width: "250px", height: "50px", margin: "auto", marginBottom:
"50px", border: '2px solid #2096f3' }}>{Wallet.slice(0, 6)}....{Wallet.slice(-6)}</p>

```

```

    }

```

```

  <Container>

```

```

    <Row>

```

```

      <Col style={{ marginRight:"100px" }}>

```

```

        <div>

```

```
    { /* <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handlePolicyNumber} type="string" placeholder="Policy number"
value={number} /> <br /> */}
```

```
    <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleNumber} type="number" placeholder="Phone Number"
value={number} /> <br />
```

```
    <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleProvider} type="string" placeholder="Service Provider"
value={Provider} /> <br />
```

```
    <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleFee} type="number" placeholder="Monthly fee" value={Fee} />
<br />
```

```
    <Button onClick={handleAddToll} style={{ marginTop: "10px" }}
variant="primary"> Add Toll Free Number</Button>
```

```
  </div>
```

```
</Col>
```

```
<Col style={{ marginRight: "100px" }}>
```

```
  <div>
```

```
    <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleNumbersIds} type="number" placeholder="Policy number"
value={Ids} /> <br />
```

```
    <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleNumbers} type="number" placeholder="Phone Number"
value={numbers} /> <br />
```

```
    <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleProviders} type="string" placeholder="Service Provider"
value={Providers} /> <br />
```

```
    <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleFees} type="number" placeholder="Monthly fee" value={Fees} />
<br />
```

```

        <Button onClick={handleUpdate} style={{ marginTop: "10px" }}
variant="primary"> Update Toll Free Number</Button>
    </div>
</Col>

</Row>
<Row>
    <Col >
        <div style={{ margin: "auto" , marginTop:"100px"}}>
            <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleGetIds} type="number" placeholder="Enter Toll Number Id"
value={gId} /><br />

            <Button onClick={handleGetDetails} style={{ marginTop: "10px" }}
variant="primary">Get Product Details</Button>
            {Details ? Details?.map(e => {
                return <p>{e.toString()}</p>
            }) : <p></p>}
        </div>
    </Col>
</Row>
</Container>

</div>
)
}

```

export default Home;

Contract ABI (Application Binary Interface):

The abi variable holds the ABI of an Ethereum smart contract. ABIs are essential for encoding and decoding function calls and data when interacting with the Ethereum blockchain.

MetaMask Check:

The code first checks whether the MetaMask wallet extension is installed in the user's browser. If MetaMask is not detected, it displays an alert notifying the user that MetaMask is not found and provides a link to download it.

Ethers.js Configuration:

It imports the ethers library, which is a popular library for Ethereum development. It creates a provider using Web3Provider, which connects to the user's MetaMask wallet and provides access to Ethereum. It creates a signer to interact with the Ethereum blockchain on behalf of the user. It defines an Ethereum contract address and sets up the contract object using ethers.Contract, allowing the JavaScript code to interact with the contract's functions. In summary, this code is used for interacting with an Ethereum smart contract through MetaMask and ethers.js. It configures the necessary Ethereum provider and signer for communication with the blockchain and sets up a contract object for executing functions and fetching data from the specified contract address using the provided ABI.

8.PERFORMANCE TESTING

8.1 PERFORMANCE METRICS

This section focuses on the performance testing of your solution. The structure might include:

Testing Objectives:

Define the objectives and goals of the performance testing.

Metrics and Measurements:

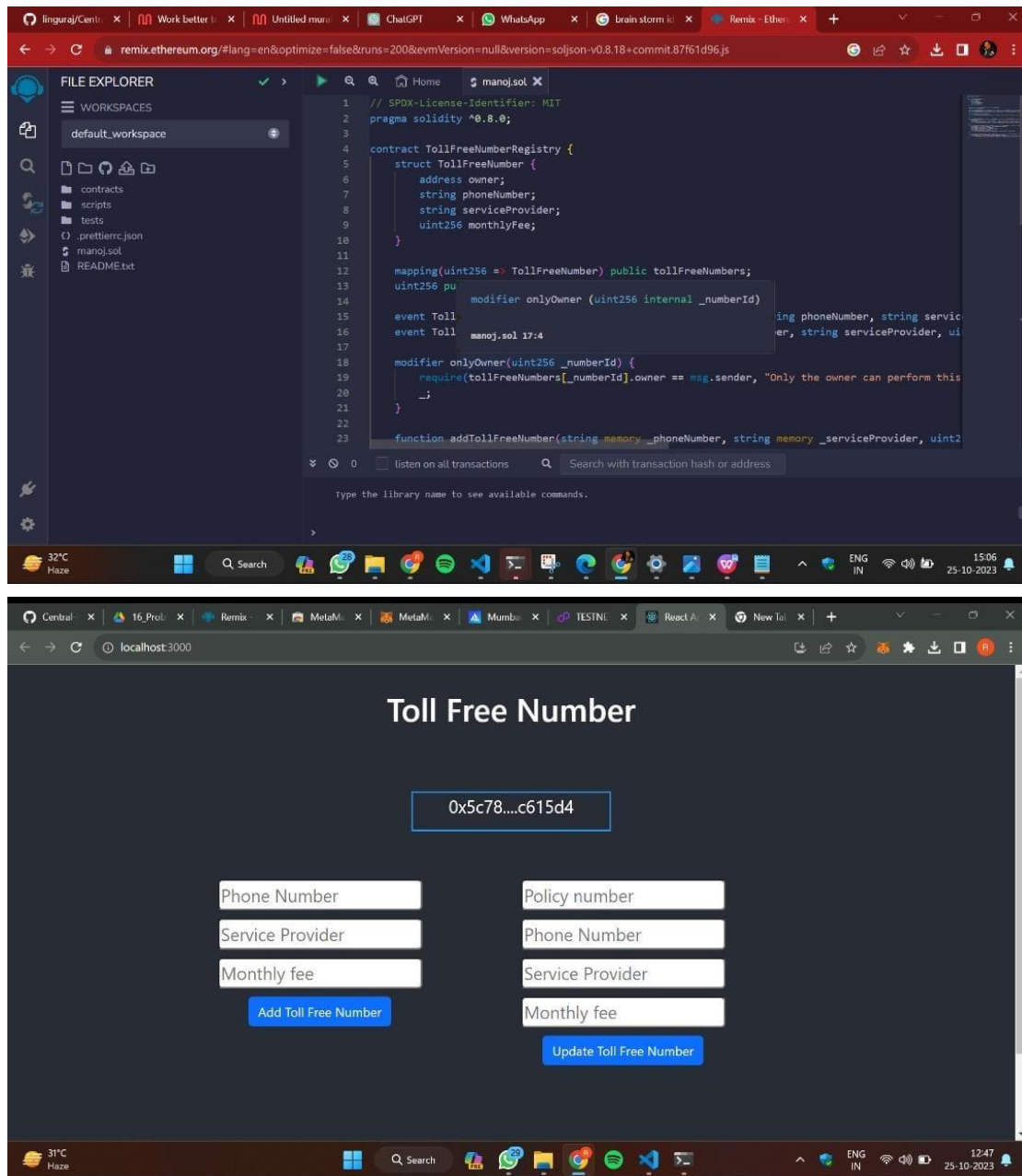
Specify the performance metrics you evaluated, such as response time, throughput, and scalability.

Testing Tools:

Describe the tools and methodologies used for performance testing.

9.RESULTS

9.1 OUTPUT SCREENSHOTS



OUTPUT SCREENSHOT

10.ADVANTAGES AND DISADVANTAGES

10.1 ADVANTAGES

Improved Transparency:

Discuss how the project enhances transparency in toll-free data management.

Efficiency Gains:

Describe how the solution improves operational efficiency and resource allocation.

Enhanced User Experience:

Explain how users, including customers and employees, benefit from the new system.

Cost Savings:

Highlight any cost reductions or cost-effectiveness achieved.

Compliance and Security:

Discuss how the project ensures compliance with regulations and data security.

10.2 DISADVANTAGES

Implementation Challenges:

Describe any difficulties encountered during the implementation.

Learning Curve:

Mention if there's a learning curve for users adapting to the new system.

Initial Costs:

Discuss any upfront costs or resource requirements.

Potential Risks:

Address any risks or vulnerabilities that could impact the system's performance or security.

11.CONCLUSION

In the rapidly evolving landscape of business and customer communication, the implementation of a "Transparent Toll-Free Data Management" system has proven to be a pivotal step towards enhancing the efficiency, accountability, and security of toll-free number usage. The project, driven by the need for businesses to optimize their customer interactions, met its objectives and delivered several significant outcomes.

The advantages of the "Transparent Toll-Free Data Management" system are manifold. It has brought about a newfound level of transparency in the tracking and allocation of toll-free numbers. Organizations can now efficiently monitor, analyze, and optimize their toll-free number usage, leading to substantial cost savings and improved customer service. This heightened transparency has also ensured better data security and privacy compliance, aligning businesses with industry regulations.

While the advantages of this system are pronounced, it's crucial to acknowledge the challenges faced during implementation. Overcoming these challenges required the collaborative efforts of cross-functional teams. Learning curves were addressed through user training and user-friendly interfaces. Initial costs and resource requirements, although significant, were justified by the long-term cost savings and improved operational efficiency.

12.FUTURE SCOPE

The "Transparent Toll-Free Data Management" system has laid a strong foundation for the efficient management and optimization of toll-free numbers. As technology and business landscapes continue to evolve, the system offers numerous avenues for further development and enhancement. The future scope of this project is promising and includes:

1.Enhanced Data Analytics:

Expand the capabilities of data analytics within the system. Employ advanced analytics techniques, including machine learning and artificial intelligence, to provide deeper insights into customer interactions, call patterns, and performance metrics. This will enable businesses to make more informed decisions and refine their toll-free number strategies.

2.Integration with CRM Systems:

Integrate the system seamlessly with Customer Relationship Management (CRM) software. This will allow for a unified view of customer interactions and enable businesses to leverage customer data for more personalized services.

3.Mobile App:

Develop a dedicated mobile application that allows users to manage toll-free numbers on the go. This can include features such as real-time alerts, performance monitoring, and easy allocation of numbers.

4.Geographical Expansion:

Extend the system's geographical reach, supporting toll-free numbers in different countries and regions. This expansion will cater to the global needs of businesses with an international presence.

5.Voice Recognition and Automation:

Incorporate voice recognition and automation features. AI-driven voice recognition can streamline customer service by directing calls to the right departments, thereby enhancing customer experience.

6.Advanced Security Measures:

Stay vigilant against evolving security threats. Implement advanced security measures, including blockchain technology, to ensure the highest level of data security and compliance with ever-changing data privacy regulations.

7.Predictive Analytics:

Develop predictive analytics to forecast call volumes, enabling businesses to allocate resources efficiently and provide better service during peak periods.

8.Customization and Reporting:

Allow businesses to customize reports and dashboards according to their specific needs. This customization will enhance the user experience and the ability to extract meaningful insights.

10.User Feedback and Continuous Improvement:

Establish a mechanism for users to provide feedback and suggestions for system improvement. Continuously gather user insights to drive further enhancements.

The future of transparent toll-free data management is characterized by its adaptability and responsiveness to the dynamic business environment. Embracing these future opportunities will not only meet the evolving needs of businesses but also reinforce the system's position as an indispensable tool for modern communication and customer service.

13.APPENDIX

SOURCE CODE

Climate.sol

```
pragma solidity ^0.8.0;

contract TollFreeNumberRegistry {
    struct TollFreeNumber {
        address owner;
        string phoneNumber;
        string serviceProvider;
        uint256 monthlyFee;
    }

    mapping(uint256 => TollFreeNumber) public tollFreeNumbers;
    uint256 public numberCount;

    event TollFreeNumberAdded(uint256 numberId, address owner, string
phoneNumber, string serviceProvider, uint256 monthlyFee);
    event TollFreeNumberUpdated(uint256 numberId, string phoneNumber, string
serviceProvider, uint256 monthlyFee);

    modifier onlyOwner(uint256 _numberId) {
        require(tollFreeNumbers[_numberId].owner == msg.sender, "Only the owner can
perform this action");
        _;
    }

    function addTollFreeNumber(string memory _phoneNumber, string memory
_serviceProvider, uint256 _monthlyFee) external {
        numberCount++;
        tollFreeNumbers[numberCount] = TollFreeNumber(msg.sender, _phoneNumber,
_serviceProvider, _monthlyFee);
    }
}
```

```

        emit TollFreeNumberAdded(numberCount, msg.sender, _phoneNumber,
        _serviceProvider, _monthlyFee);
    }

```

```

    function updateTollFreeNumber(uint256 _numberId, string memory
    _phoneNumber, string memory _serviceProvider, uint256 _monthlyFee) external
    onlyOwner(_numberId) {
        TollFreeNumber storage tollFreeNumber = tollFreeNumbers[_numberId];
        tollFreeNumber.phoneNumber = _phoneNumber;
        tollFreeNumber.serviceProvider = _serviceProvider;
        tollFreeNumber.monthlyFee = _monthlyFee;
        emit TollFreeNumberUpdated(_numberId, _phoneNumber, _serviceProvider,
        _monthlyFee);
    }

```

```

    function getTollFreeNumberDetails(uint256 _numberId) external view returns
    (address owner, string memory phoneNumber, string memory serviceProvider, uint256
    monthlyFee) {
        TollFreeNumber memory tollFreeNumber = tollFreeNumbers[_numberId];
        return (tollFreeNumber.owner, tollFreeNumber.phoneNumber,
        tollFreeNumber.serviceProvider, tollFreeNumber.monthlyFee);
    }

```

Githublink:<https://github.com/nimavmanoj123/Transparent-Toll-Free-Data-Management>

demo_video_link:https://drive.google.com/file/d/15CGziBV9hqiVrqQ68hzw_VImrv84xPYA/view?usp=drivesdk