NAAN MUDHALVAN PROJECT REPORT

TRANSPARENT TOLL-FREE DATA MANAGEMENT

DATE	30 October 2023
TEAM ID	NM2023TMID05495
PROJECT NAME	TRANSPARENT TOLL-FREE DATA MANAGEMENT

TEAM MEMBERS

ELUMALAI M	**421620114008**
YAKESH S	**421620114025**
GOPINATH P	**421620114009**
ARAVINDAN P	**421620114002**
DINAGARAJ J	**421620114006**

Mailam Engineering College , Tindivanam , Mailam – 604304
Villupuram District

Table of Contents

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:

- Implementing a centralized system for the management of toll-free numbers and associated data.
- Enhancing the transparency of data related to toll-free number usage, expenses, and performance.
- Providing tools and analytics for businesses to make data-driven decisions regarding toll-free number allocation and optimization.

- 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

Transparency in toll-free data management refers to the practice of making toll-free phone number data and related information openly accessible and understandable to relevant stakeholders, such as the organization that owns the toll-free numbers, regulatory authorities, and the public.

The purpose of transparency in toll-free data management includes:

- **Accountability:** By being transparent about how toll-free numbers are managed and utilized, organizations can be held accountable for their use. This helps prevent misuse or fraud associated with toll-free numbers.

- **Regulatory Compliance:** Transparency ensures that organizations comply with regulations and standards set by authorities like the Federal Communications Commission (FCC) in the United States. Accessible data allows regulators to monitor and enforce compliance.

- **Public Trust:** When the public can access information about toll-free numbers, it builds trust. People can verify the legitimacy of organizations using toll-free numbers for customer service, support, or other purposes.

- **Preventing Fraud:** Transparent data management can help identify and prevent fraudulent activities, such as spam and phishing, which often involve toll-free numbers.

- **Efficient Resource Allocation:** Organizations can better allocate toll-free resources when they have transparency into how these numbers are used. This can lead to cost savings and improved service quality.

- **Data Analysis:** Access to transparent data allows organizations to analyze call traffic, identify patterns, and make informed decisions regarding their toll-free number strategies.

In summary, transparency in toll-free data management is essential to maintain trust, regulatory compliance, and efficient use of toll-free numbers. It supports fair and accountable practices while preventing misuse and fraud associated with these valuable communication tools.

2. LITERATURE SURVEY

- **Challenges in Toll-Free Data Management:** Discuss the challenges and inefficiencies in traditional toll data management systems, such as manual data handling and lack of transparency**.**

- **Blockchain and Transparency:** Explore how blockchain technology, with its inherent transparency and immutability, is being adopted to address these challenges in toll data management. Cite specific projects or case studies that demonstrate the benefits.

- **Smart Contracts and Automation:** Go into more detail on the application of smart contracts in toll collection and management. Discuss how automation through smart contracts can enhance efficiency and reduce fraud.

- **Security Measures:** Elaborate on the security measures employed in toll-free data management systems. Discuss encryption, access control, and data privacy measures to protect sensitive information.

- **Regulatory and Legal Aspects:** Explore the regulatory and legal aspects of toll data management, especially concerning data privacy, compliance with government regulations, and how blockchain impacts these factors.

- **Use Cases:** Provide real-world examples of toll-free data management systems that have successfully incorporated transparency and blockchain technology.

- **Future Directions:** Discuss potential future developments and trends in transparent toll-free data management, including emerging technologies, industry standards, and areas for further research.

- **Challenges and Limitations:** Address any challenges and limitations of implementing transparent toll-free data management systems, such as scalability, interoperability, and public acceptance.

2.1 EXISTING 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.

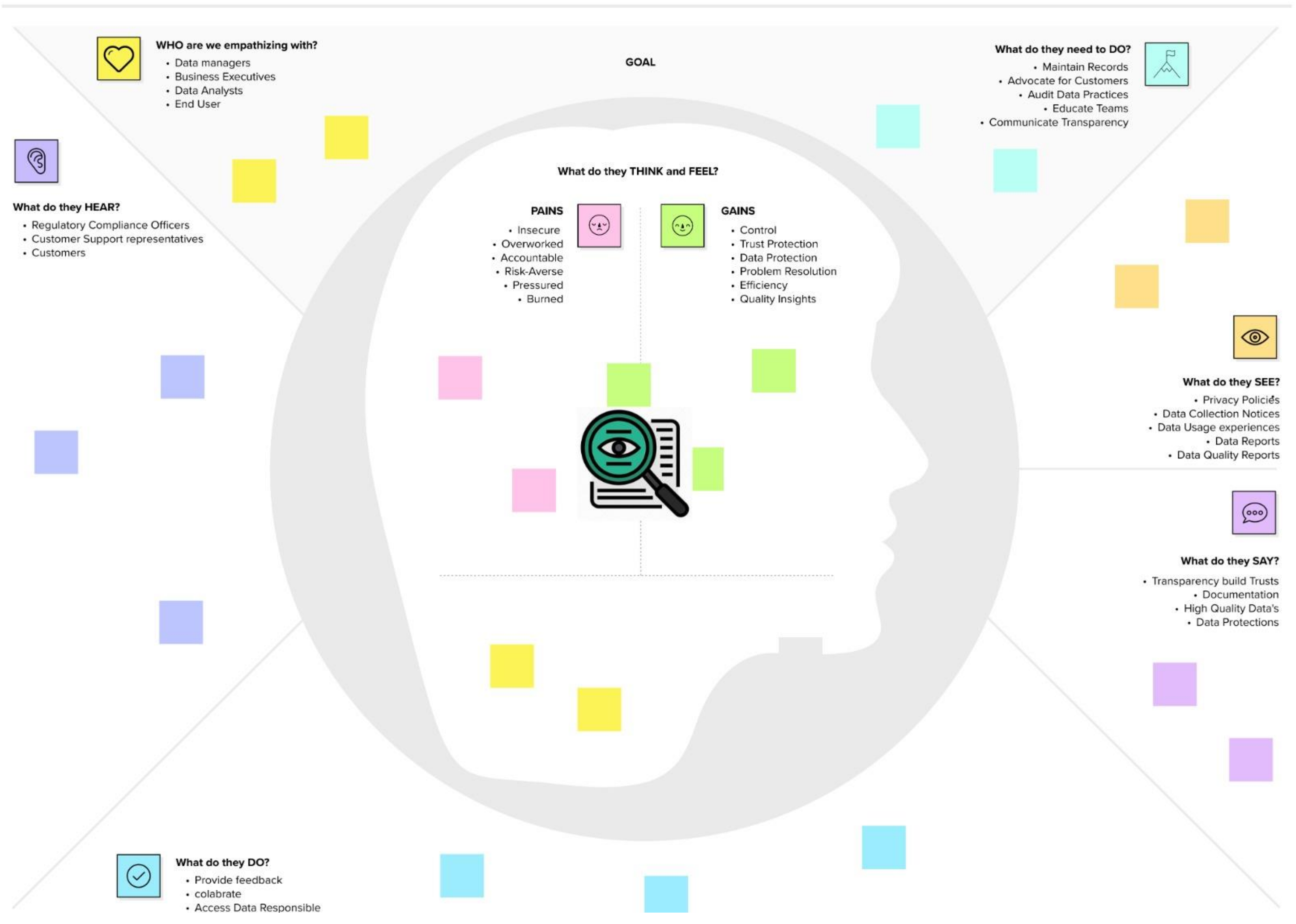
2.2 PROBLEM STATEMENT

- 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 place holders 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. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION AND BRAINSTORMING

Creating an electronic voting system using blockchain technology is a complex task that involves a range of technical, security, and usability challenges. Ideation and brainstorming are crucial to generate innovative solutions. Here's a structured approach to ideate and brainstorm for a blockchain-based electronic voting system **:**

1. Define the Problem:

Start by defining the specific challenges and requirements for the electronic voting system. For example, ensure transparency, security, accessibility, and scalability.

2. Assemble a Diverse Team:

Gather a cross-functional team that includes blockchain developers, cybersecurity experts, election officials, accessibility advocates, and legal experts.

3. Research and Benchmarking:

Research existing blockchain-based voting systems, their strengths, and weaknesses. Benchmark against traditional voting systems.

4. Ideation Techniques:

Employ brainstorming techniques to generate innovative ideas. Here are some approaches:

- **Blockchain Features Brainstorm:** Brainstorm specific blockchain features and how they can enhance the voting system's security and transparency. For example, consider immutability, cryptographic proofs, and distributed ledger technology.

- **User-Centered Design:** Put voters at the center of the design process. Brainstorm user-friendly interfaces and accessibility features for all citizens, including those with disabilities.

- **Layered Security:** Brainstorm layers of security to protect against various threats, including identity verification, cryptographic methods, and auditing mechanisms.

5. Constraints and Requirements:

Define constraints and requirements for the system, such as legal compliance, budget limitations, and scalability.

6. Prototyping:

Develop low-fidelity prototypes or conceptual models to visualize how the blockchain-based voting system might work.

7. Test and Gather Feedback:

Collect feedback from potential users, security experts, and election officials. Conduct usability testing and security audits.

8. Risk Assessment:

Brainstorm potential risks and vulnerabilities, and develop mitigation strategies.

9. Iteration:

Iterate through the ideation and brainstorming process, revising and refining ideas based on feedback and emerging technologies.

10. Evaluate Feasibility:

Assess the technical feasibility of the proposed ideas, considering factors like blockchain platform selection, scalability, and interoperability.

11. Select the Best Ideas:

Choose the most promising ideas based on feasibility, impact, and alignment with project objectives.

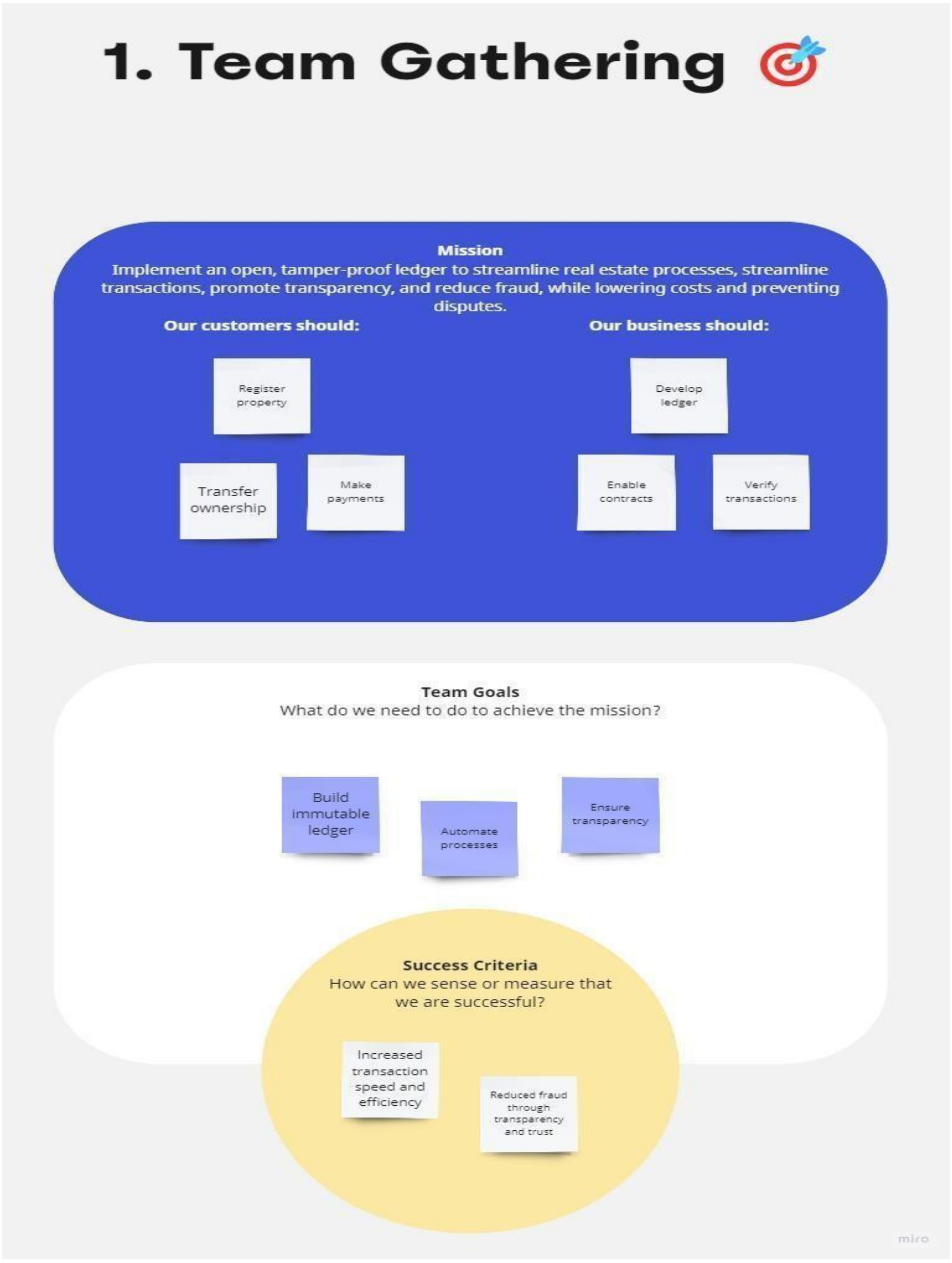
12. Detailed Plans:

Create detailed plans for the selected ideas, including technical specifications, budgeting, project timelines, and resource requirements.

13. Pilot Programs:

Consider conducting small-scale pilot programs to validate the ideas and gather real-world feedback.

Step-1: Team Gathering And Collaboration



1. Team Gathering

Mission

Implement an open, tamper-proof ledger to streamline real estate processes, streamline transactions, promote transparency, and reduce fraud, while lowering costs and preventing disputes.

Our customers should:

- Register property
- Transfer ownership
- Make payments

Our business should:

- Develop ledger
- Enable contracts
- Verify transactions

Team Goals

What do we need to do to achieve the mission?

- Build immutable ledger
- Automate processes
- Ensure transparency

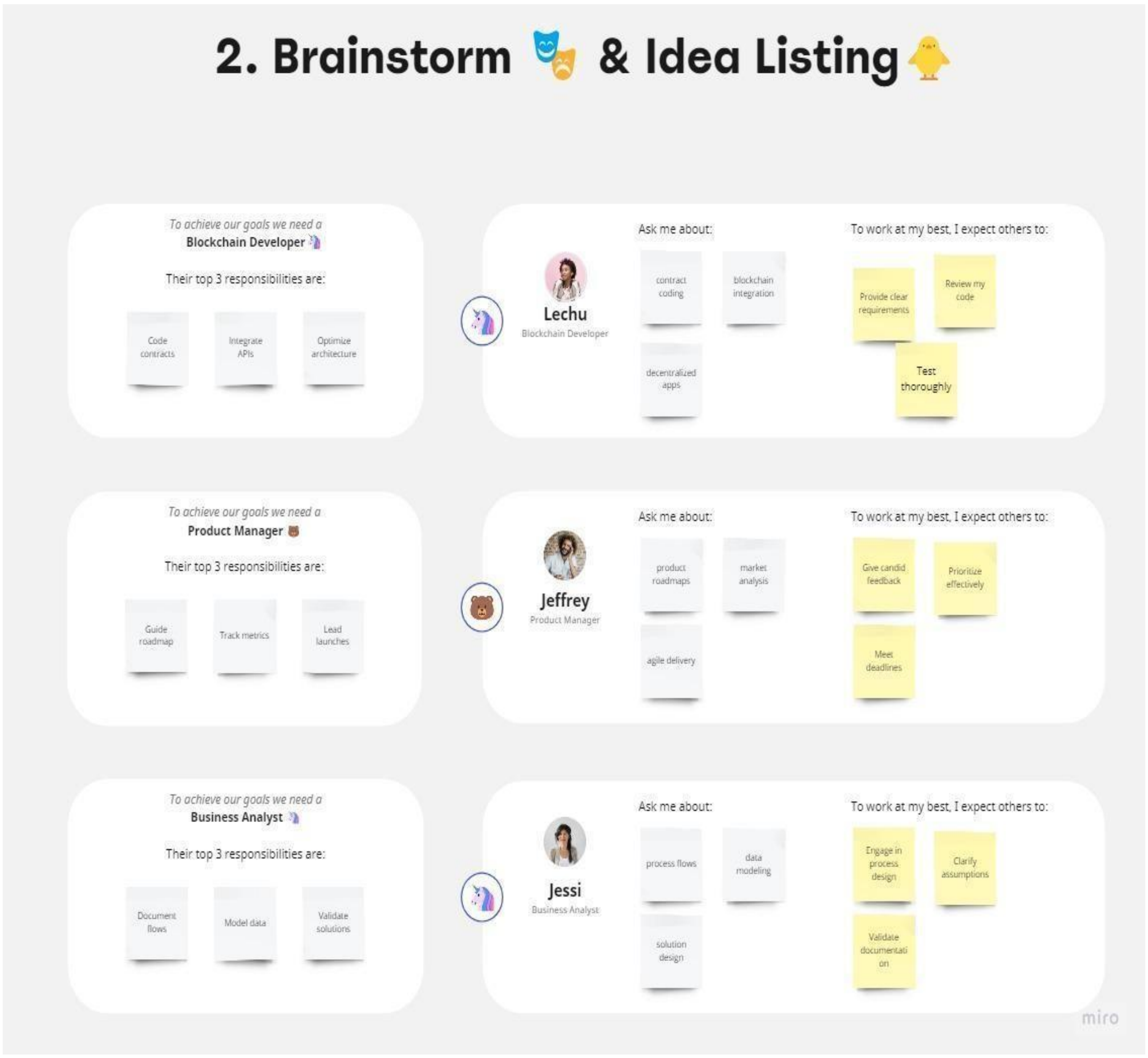
Success Criteria

How can we sense or measure that we are successful?

- Increased transaction speed and efficiency
- Reduced fraud through transparency and trust

miro

<u>Step-2:</u> Brainstorm, Idea Listing and Grouping



2. Brainstorm 🎭 & Idea Listing 🐥

To achieve our goals we need a
Blockchain Developer 🦄

Their top 3 responsibilities are:

Code contracts
Integrate APIs
Optimize architecture

Lechu
Blockchain Developer

Ask me about:
contract coding
blockchain integration
decentralized apps

To work at my best, I expect others to:
Provide clear requirements
Review my code
Test thoroughly

To achieve our goals we need a
Product Manager 🐻

Their top 3 responsibilities are:

Guide roadmap
Track metrics
Lead launches

Jeffrey
Product Manager

Ask me about:
product roadmaps
market analysis
agile delivery

To work at my best, I expect others to:
Give candid feedback
Prioritize effectively
Meet deadlines

To achieve our goals we need a
Business Analyst 🦄

Their top 3 responsibilities are:

Document flows
Model data
Validate solutions

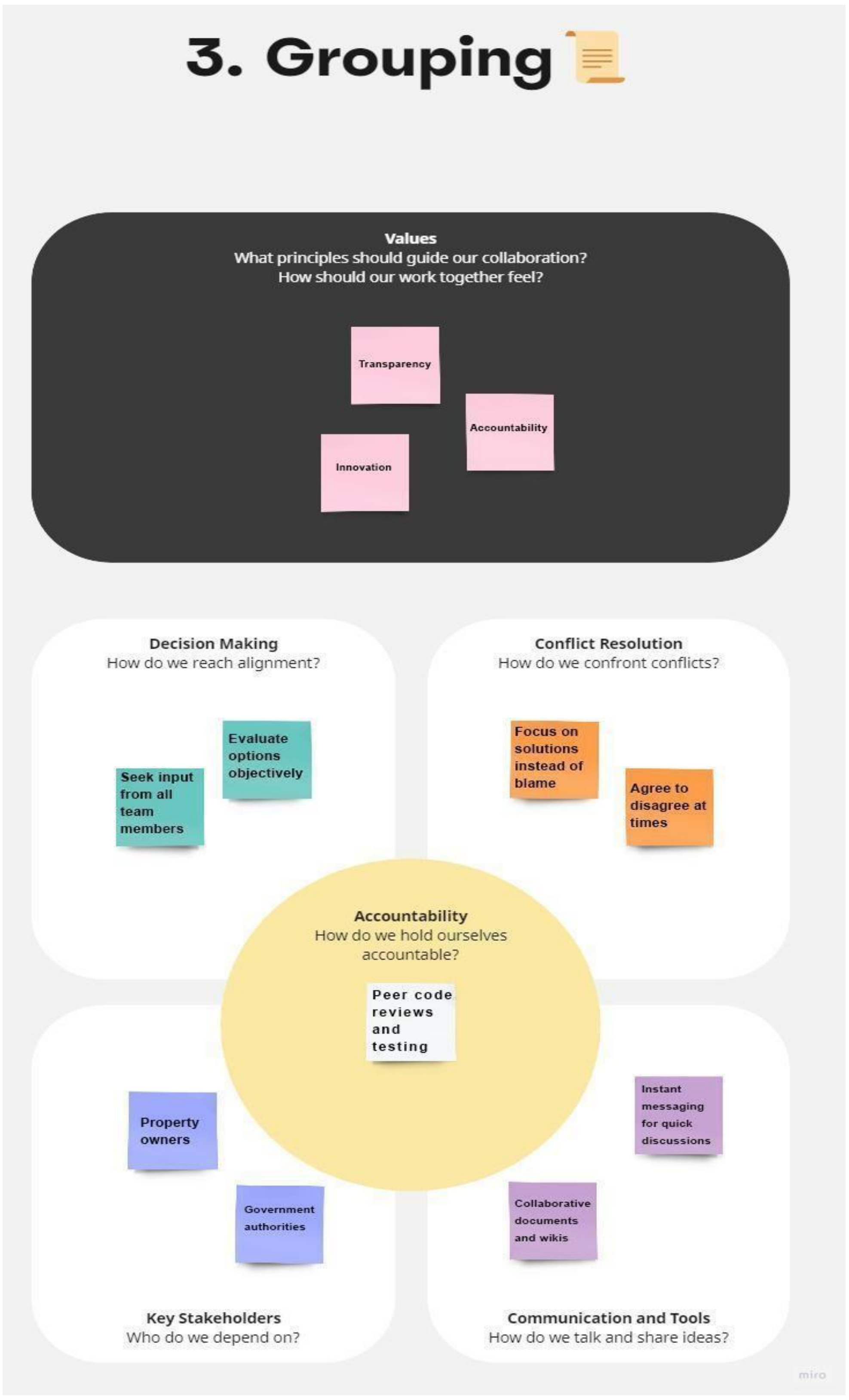
Jessi
Business Analyst

Ask me about:
process flows
data modeling
solution design

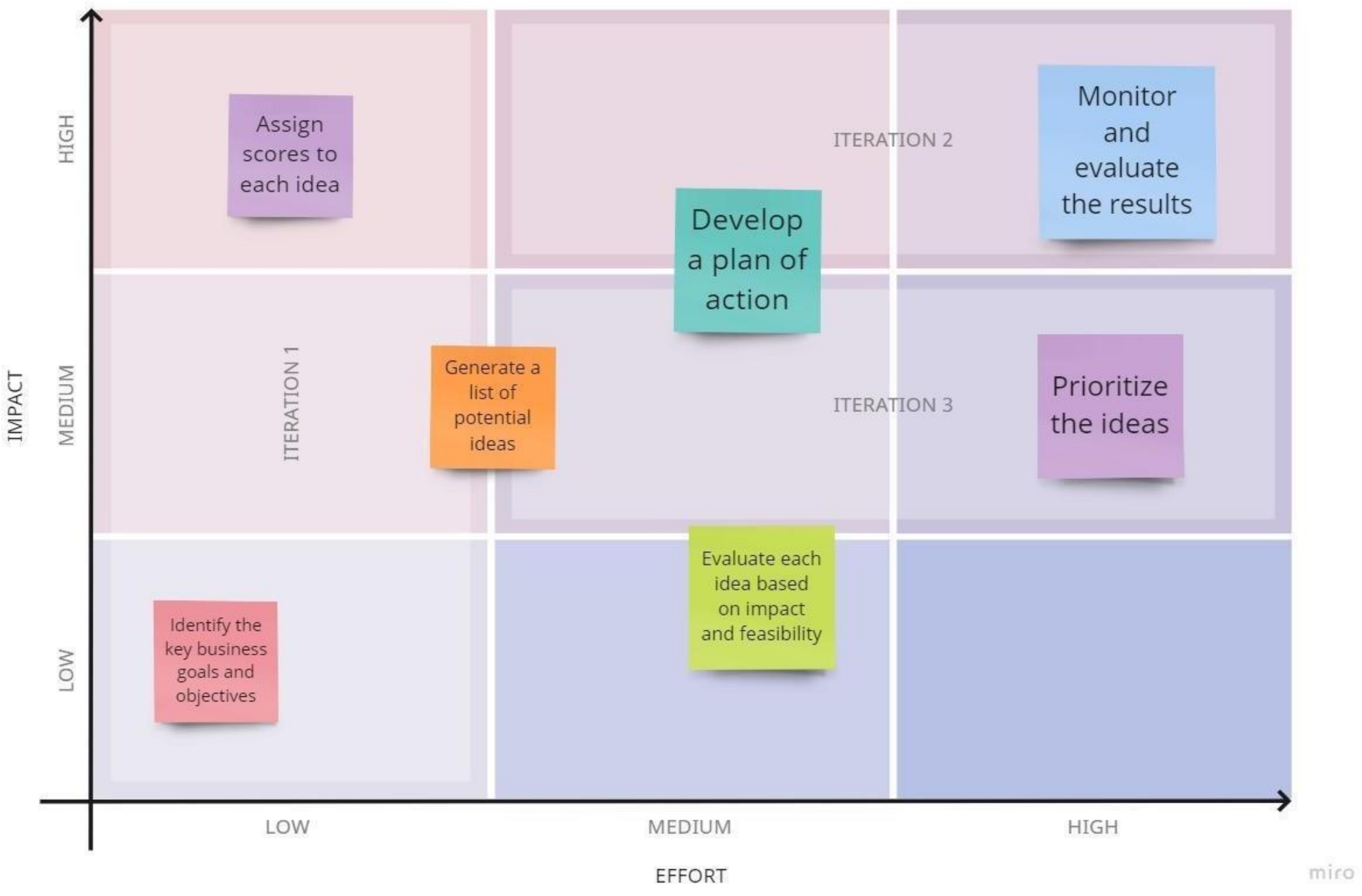
To work at my best, I expect others to:
Engage in process design
Clarify assumptions
Validate documentation

miro

Step-3: Grouping


3. Grouping
Values
What principles should guide our collaboration?
How should our work together feel?
Transparency
Accountability
Innovation
Decision Making
How do we reach alignment?
Evaluate options objectively
Seek input from all team members
Conflict Resolution
How do we confront conflicts?
Focus on solutions instead of blame
Agree to disagree at times
Accountability
How do we hold ourselves accountable?
Peer code reviews and testing
Property owners
Government authorities
Instant messaging for quick discussions
Collaborative documents and wikis
Key Stakeholders
Who do we depend on?
Communication and Tools
How do we talk and share ideas?
miro

Step-4: Idea Prioritization



Assign scores to each idea
ITERATION 2
Monitor and evaluate the results
Develop a plan of action
ITERATION 1
Generate a list of potential ideas
ITERATION 3
Prioritize the ideas
Evaluate each idea based on impact and feasibility
Identify the key business goals and objectives
IMPACT
HIGH
MEDIUM
LOW
EFFORT
LOW
MEDIUM
HIGH
miro

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL 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.

4.2 NON 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:**

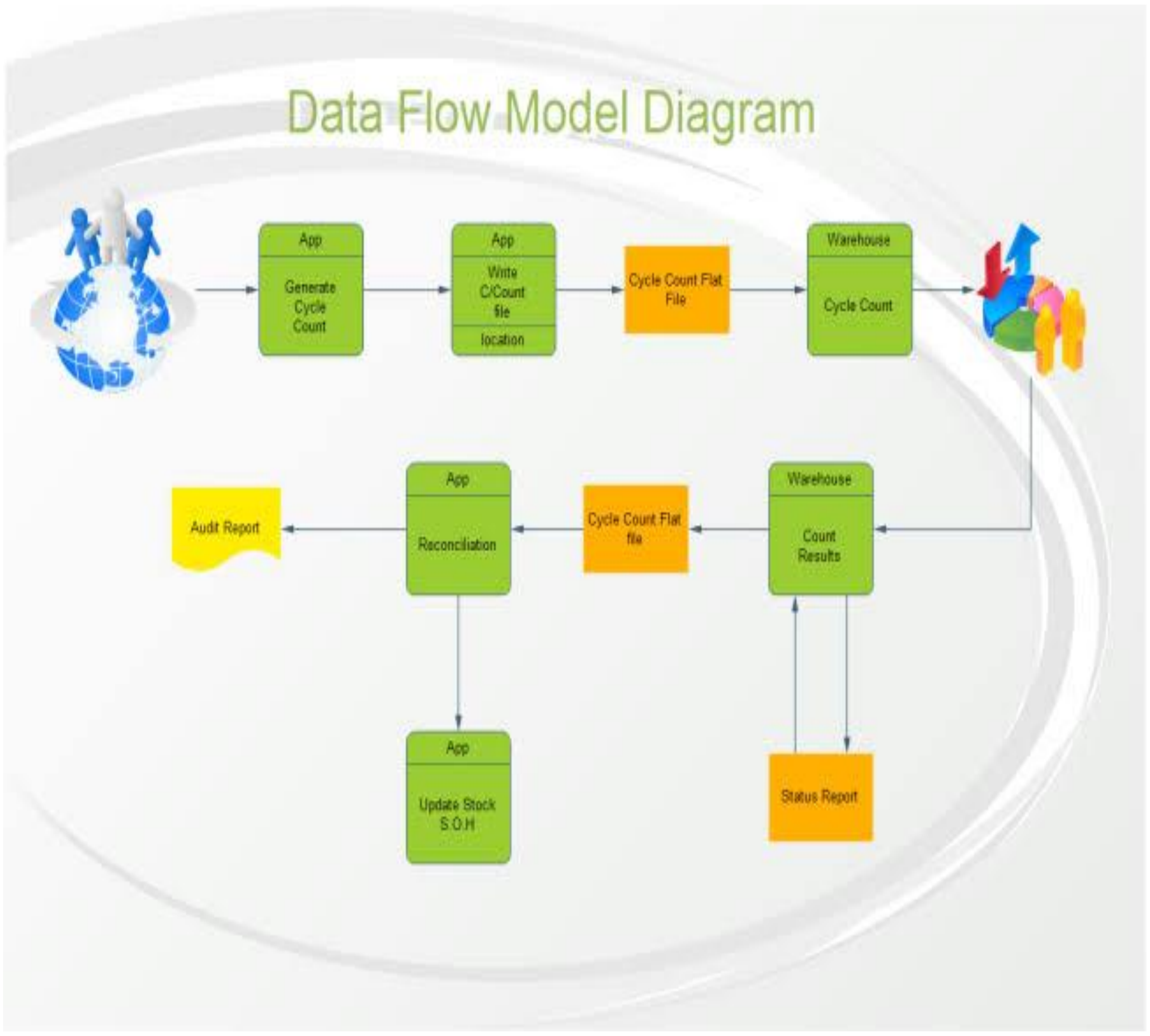
  Implement a robust backup and recovery mechanism to safeguard data (e.g., GDPR).

- **Performance:**

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

5. PROJECT DESIGN

5.1 DATA FLOW-DIAGRAM AND USER STORY



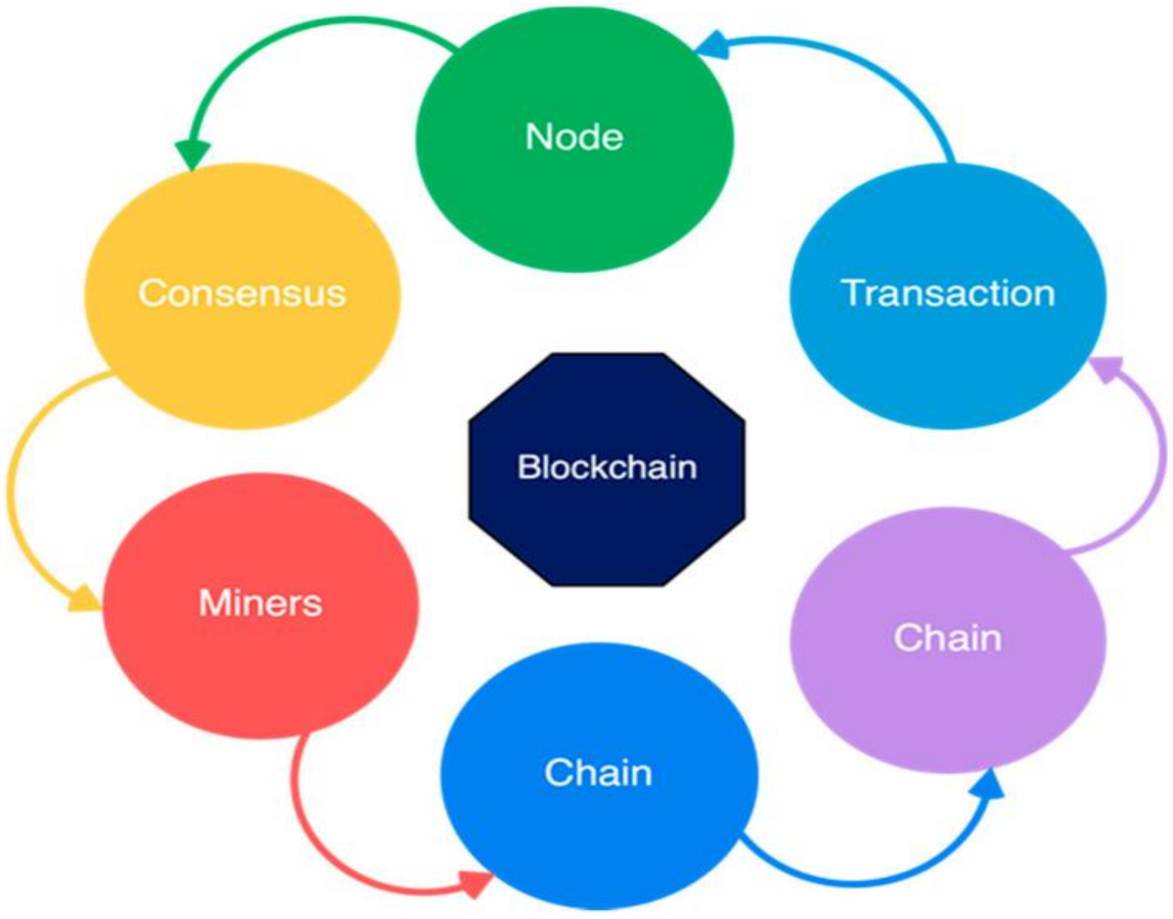
Data Flow Diagram:

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.

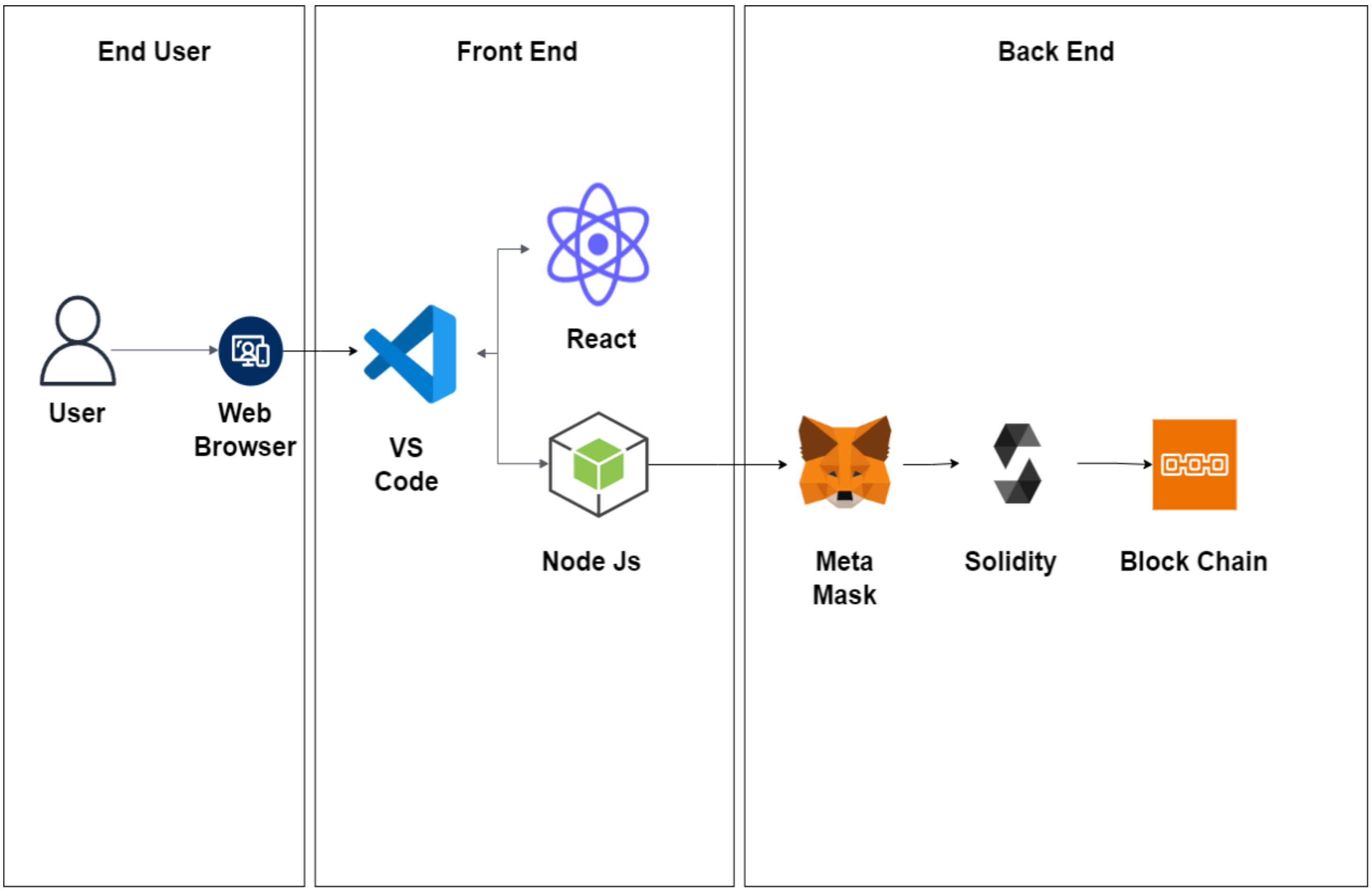
- **Solidity(Remix ide)** - Solidity is a high-level, statically-typed programming language used for developing smart contracts on various blockchain platforms, with Ethereum being the most prominent. Smart contracts are self-executing contracts with the terms of the agreement directly written into code.

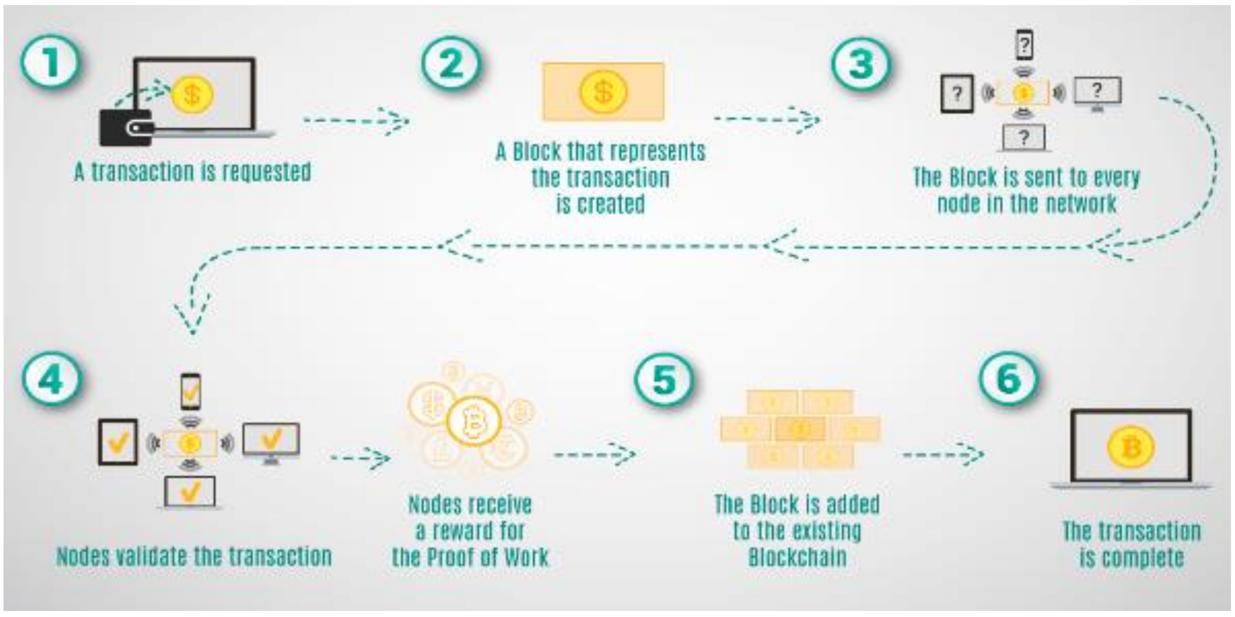
- **Remix IDE -** is an essential tool for Solidity developers and is widely used in the Ethereum ecosystem. It simplifies the smart contract development process and provides many useful features for coding, testing, and deploying contracts on the Ethereum blockchain.

- **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 :




1
A transaction is requested
2
A Block that represents the transaction is created
3
The Block is sent to every node in the network
4
Nodes validate the transaction
Nodes receive a reward for the Proof of Work
5
The Block is added to the existing Blockchain
6
The transaction is complete

6.2 Sprint 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

Transparent Toll-Free Data Management (Solidity)

```
// SPDX-License-Identifier: MIT
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);
  }
}
```

This Solidity contract implements basic functions of Transparent Toll-Free Data Management properties on the blockchain.

Contract ABI (Application Binary Interface):

The abi variable holdstheABI 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 Web3 Provider, which connects to the user'sMetaMask 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

Here are some key performance metrics for performance testing in this context:

1. Response Time Metrics:

- **Average Response Time:** Measure the average time taken for the system to respond to user requests, such as data retrieval or transaction processing.

- **Peak Response Time:** Identify the maximum response time observed during high load periods.

- **95th and 99th Percentile Response Times:** Evaluate the response times at the 95th and 99th percentiles, which indicate the performance for most user interactions.

2. Throughput Metrics:

- **Transactions per Second (TPS):** Determine the number of transactions (data entries, retrievals, smart contract executions) the system can handle per second under different load levels.

- **Requests per Second (RPS):** Calculate the rate at which user requests are processed by the system.

3. Scalability Metrics:

- **Scalability Limits:** Identify the maximum user load or data volume the system can handle without performance degradation.

- **Response Time Scalability:** Measure how response times change as user load increases or decreases.

4. Error Metrics:

- **Error Rate:** Calculate the percentage of failed transactions or erroneous responses under load.

- **Error Handling Time:** Evaluate the time it takes to detect and handle errors or exceptions in the system.

5. Resource Utilization Metrics:

- **CPU Utilization:** Monitor CPU usage to assess the system's processing capacity.

- **Memory Utilization:** Evaluate RAM usage to ensure efficient memory management.

- **Network Bandwidth:** Measure network usage to identify potential bottlenecks

6. Concurrency Metrics:

- **Concurrent User Load:** Determine the maximum number of concurrent users the system can support while maintaining acceptable performance.

- **Transaction Concurrency:** Assess the system's ability to handle simultaneous data entries, retrievals, and smart contract executions.

7. Stress Testing Metrics:

- **Breakpoint:** Identify the load level at which the system fails or experiences significant degradation.

- **Recovery Time:** Measure how quickly the system recovers after a stress point.

8. Smart Contract Metrics:

- **Smart Contract Execution Time:** Assess the time it takes for smart contracts to execute under different loads.

- **Gas Usage:** Monitor the gas consumption of smart contracts to optimize cost-efficiency.

9. Data Storage Metrics:

- **Data Retrieval Time:** Measure the time it takes to retrieve data from storage.

- **Data Insertion Time:** Evaluate the time it takes to insert new data.

10. Compliance Metrics:

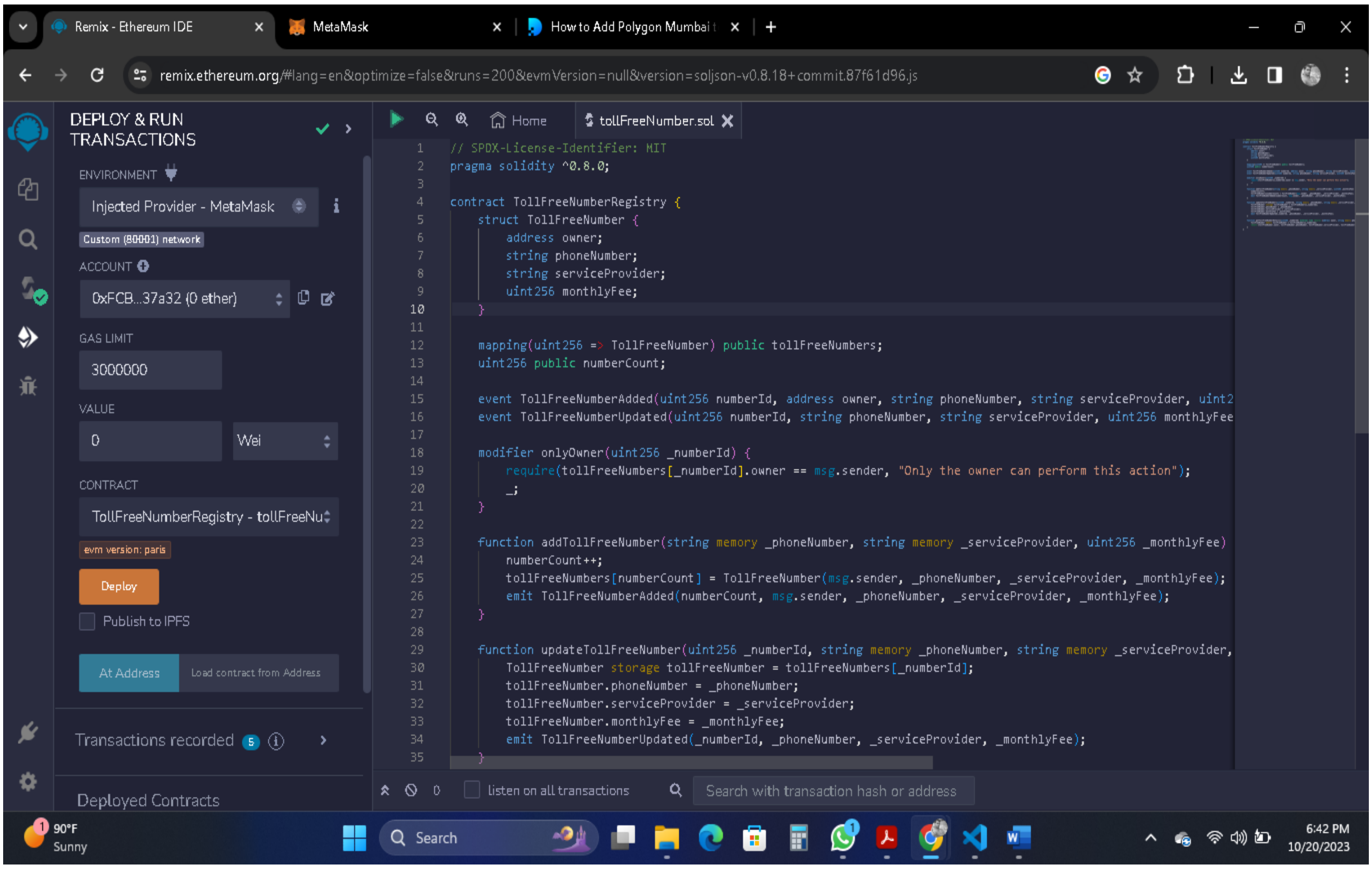
- **Regulatory Compliance:** Ensure that the system complies with data privacy regulations and legal requirements.

- **Data Protection:** Assess the security measures in place for protecting sensitive data.

By defining and monitoring these performance metrics, we can ensure that Transparent Toll-Free Data Management system operates efficiently, maintains transparency and accountability, and meets regulatory standards while handling user loads effectively.

9. RESULTS

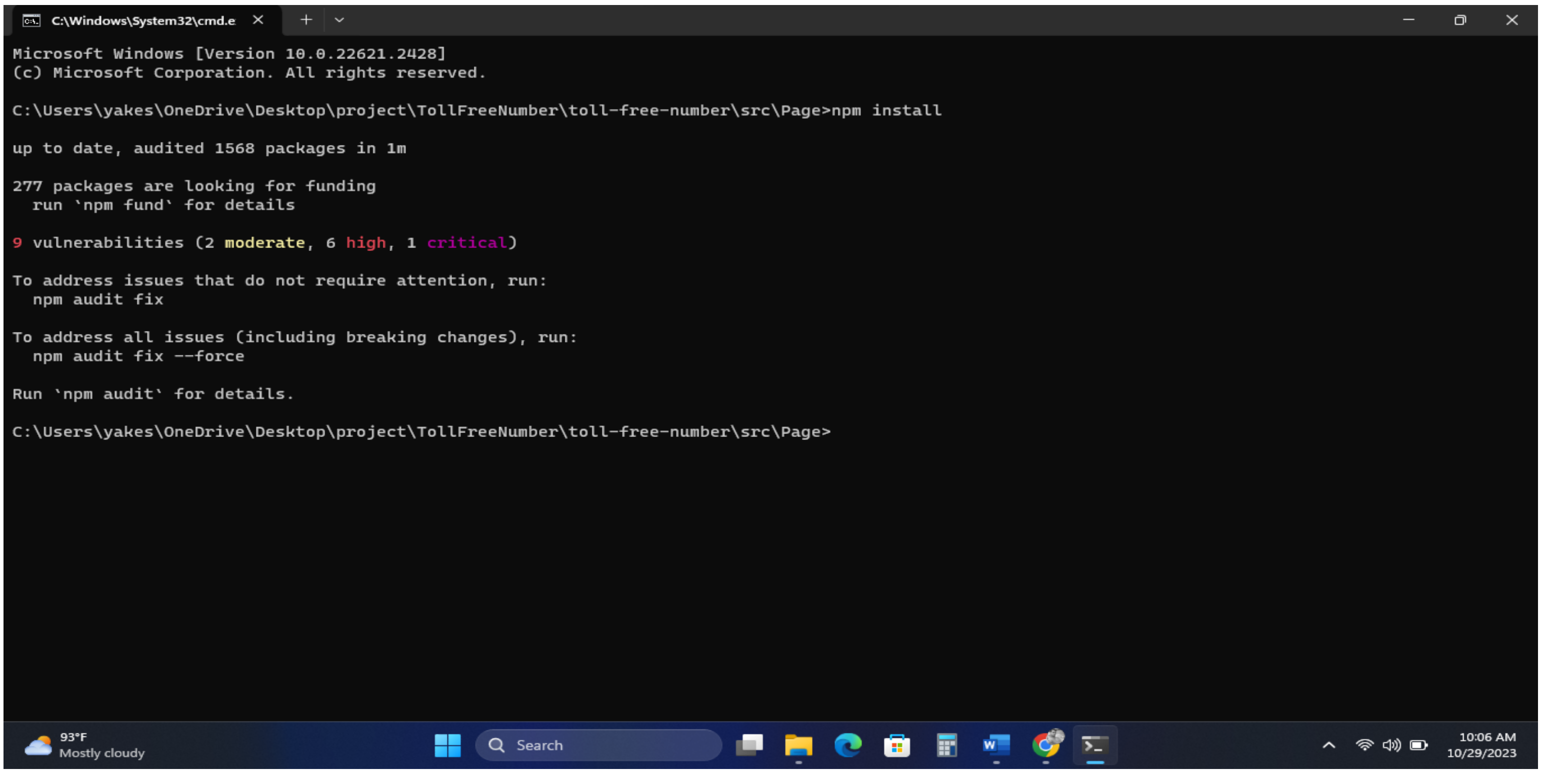
9.1 OUTPUT SCREENSHOTS


Remix - Ethereum IDE
MetaMask
How to Add Polygon Mumbai t
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js
DEPLOY & RUN TRANSACTIONS
Home
tollFreeNumber.sol
listen on all transactions
Search with transaction hash or address
Transactions recorded 5
Deployed Contracts
90°F
Sunny
Search
6:42 PM
10/20/2023

ENVIRONMENT: Injected Provider - MetaMask
Custom (80001) network
ACCOUNT: 0xFCB...37a32 (0 ether)
GAS LIMIT: 3000000
VALUE: 0 Wei
CONTRACT: TollFreeNumberRegistry - tollFreeNu
evm version: paris
Deploy
Publish to IPFS
At Address
Load contract from Address

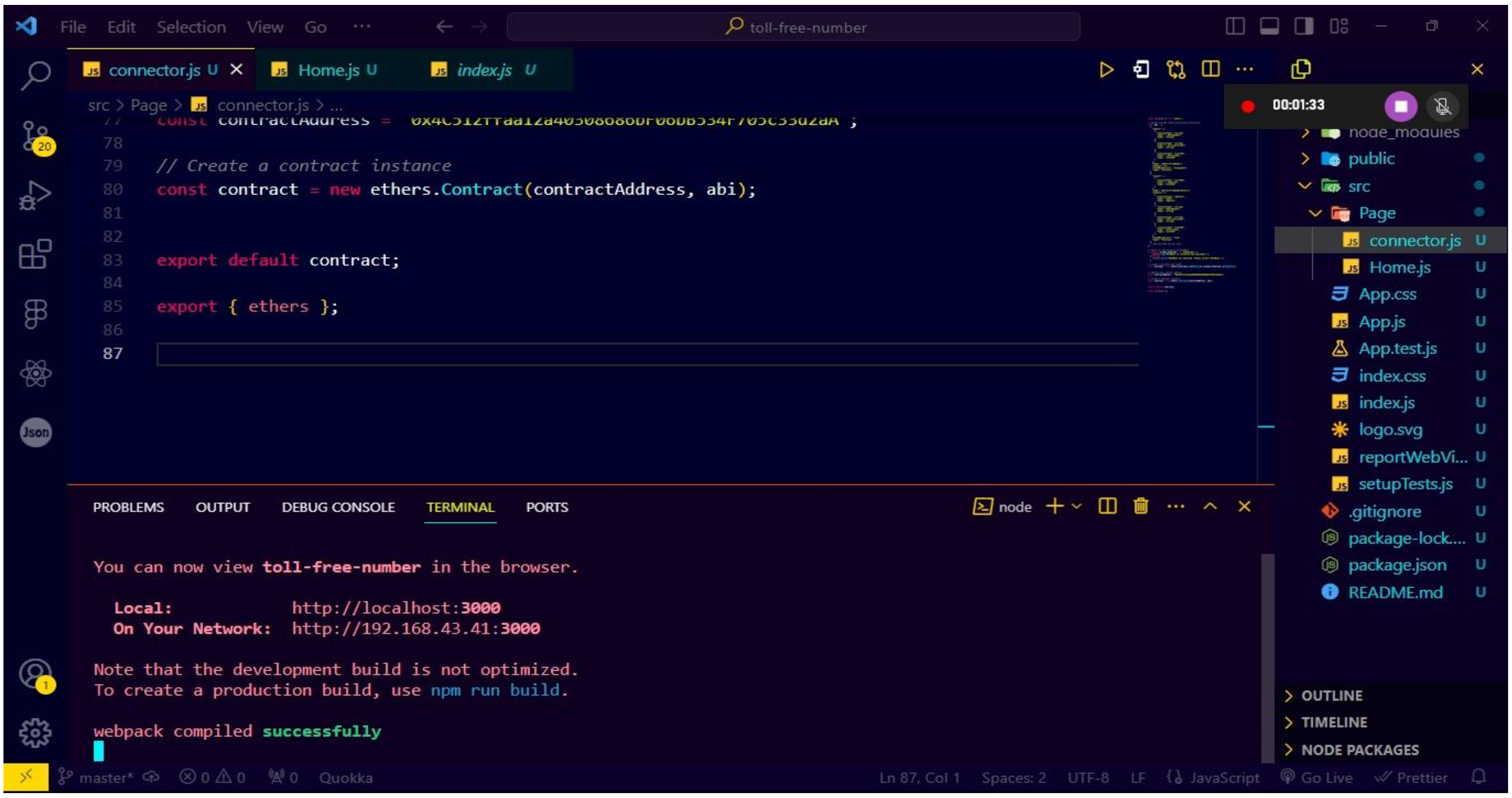
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

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, uint2
    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)
        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,
        TollFreeNumber storage tollFreeNumber = tollFreeNumbers[_numberId];
        tollFreeNumber.phoneNumber = _phoneNumber;
        tollFreeNumber.serviceProvider = _serviceProvider;
        tollFreeNumber.monthlyFee = _monthlyFee;
        emit TollFreeNumberUpdated(_numberId, _phoneNumber, _serviceProvider, _monthlyFee);
    }
```

CREATING A SMART CONTACT



```
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\yakes\OneDrive\Desktop\project\TollFreeNumber\toll-free-number\src\Page>npm install

up to date, audited 1568 packages in 1m

277 packages are looking for funding
  run `npm fund` for details

9 vulnerabilities (2 moderate, 6 high, 1 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

C:\Users\yakes\OneDrive\Desktop\project\TollFreeNumber\toll-free-number\src\Page>
```

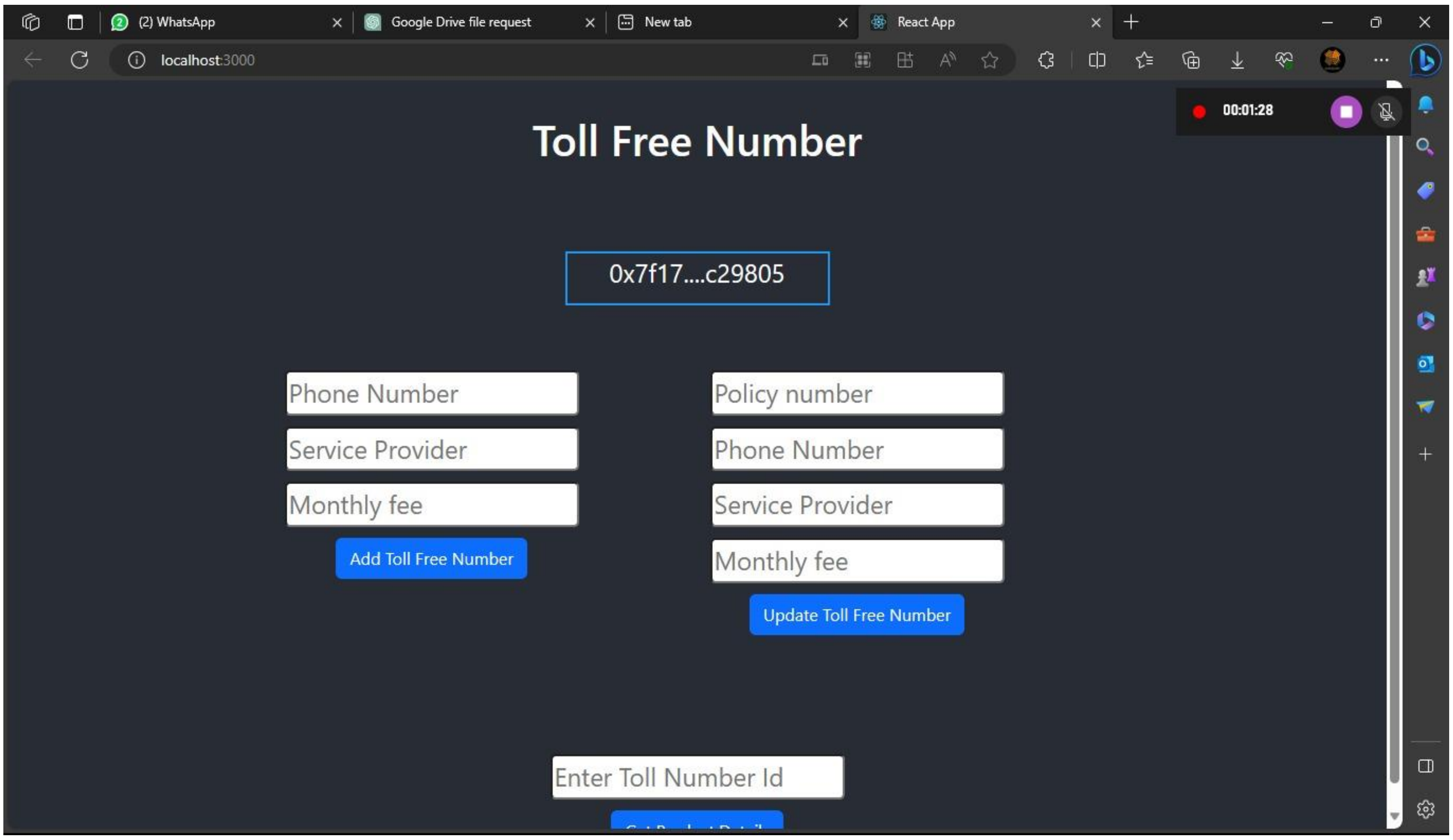
INSTALLING DEPENDENCIES



```javascript
// Create a contract instance
const contract = new ethers.Contract(contractAddress, abi);


export default contract;

export { ethers };

```

```
You can now view toll-free-number in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.43.41:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

HOSTING THE SITE LOCALLY



Toll Free Number

0x7f17....c29805

Phone Number
Service Provider
Monthly fee
Add Toll Free Number

Policy number
Phone Number
Service Provider
Monthly fee
Update Toll Free Number

Enter Toll Number Id

OUTPUT SCREEN

10. ADVANTAGES AND DISADVANTAGES

10.1 ADVANTAGES

- **Data Integrity and Immutability:** Data stored on a blockchain is tamper-proof. Once recorded, data cannot be altered or deleted without consensus, ensuring data integrity.

- **Transparency and Accountability:** Blockchain provides a transparent and auditable ledger. Every data transaction is recorded and visible to authorized parties, fostering accountability.

- **Security:** Strong cryptographic techniques and decentralized architecture make blockchains highly secure. Access to data is controlled through private keys.

- **Automation with Smart Contracts:** Smart contracts enable automation of toll collection and data management processes, reducing manual intervention and associated errors.

- **Efficiency:** The automation, transparency, and real-time data access offered by blockchain improve efficiency in toll-free data management.

- **User Trust:** The transparency and security of blockchain instill trust in users. They can verify transactions, monitor their data, and expect data integrity.

- **Regulatory Compliance:** The auditability of blockchain and data protection measures support compliance with regulatory requirements

10.2 DISADVANTAGES

- **Scalability Challenges:** Blockchain networks can face scalability issues, particularly public blockchains. High transaction volumes can lead to slower processing times and increased costs.

- **Complexity:** Implementing blockchain technology requires a deep understanding of the technology and can be complex to set up and maintain.

- **Energy Consumption:** Proof-of-work blockchains, like Bitcoin and Ethereum, consume significant energy, which can be an environmental concern.

- **Costs:** Developing, deploying, and maintaining a blockchain system can be expensive. Initial setup costs and ongoing operational expenses can be prohibitive.

- **Lack of Regulation:** The regulatory environment for blockchain technology is still evolving, and it can be unclear how existing laws apply to specific use cases.

- **User Education:** Users may need to be educated about how to interact with a blockchain-based system and the associated security measures.

- **Data Privacy Concerns:** While blockchain provides security, it also makes data more permanent. This could raise concerns about personal data stored on the blockchain, especially if privacy regulations change.

- **Resistance to Change:** Introducing blockchain to an existing system may face resistance from stakeholders who are accustomed to traditional data management methods.

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:

- **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.

- **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.

- **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.

- **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.

- **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.

- **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.

- **Predictive Analytics:**

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

- **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.

- **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

```
// SPDX-License-Identifier: MIT
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);
  }
}
```

Connector.js :

```
const { ethers } = require("ethers");

const abi = [
 {
  "anonymous": false,
  "inputs": [
   {
    "indexed": false,
    "internalType": "uint256",
    "name": "numberId",
    "type": "uint256"
   },
   {
    "indexed": false,
    "internalType": "address",
    "name": "owner",
    "type": "address"
   },
   {
    "indexed": false,
    "internalType": "string",
    "name": "phoneNumber",
    "type": "string"
   },
   {
    "indexed": false,
    "internalType": "string",
    "name": "serviceProvider",
    "type": "string"
   },
   {
    "indexed": false,
    "internalType": "uint256",
    "name": "monthlyFee",
    "type": "uint256"
   }
  ],
  "name": "TollFreeNumberAdded",
  "type": "event"
 },
 {
  "anonymous": false,
  "inputs": [
   {
    "indexed": false,
   "internalType": "uint256",
   "name": "numberId",
   "type": "uint256"
  },
  {
   "indexed": false,
   "internalType": "string",
   "name": "phoneNumber",
   "type": "string"
  },
  {
   "indexed": false,
   "internalType": "string",
   "name": "serviceProvider",
   "type": "string"
  },
  {
   "indexed": false,
   "internalType": "uint256",
   "name": "monthlyFee",
   "type": "uint256"
  }
 ],
 "name": "TollFreeNumberUpdated",
 "type": "event"
},
{
 "inputs": [
  {
   "internalType": "string",
   "name": "_phoneNumber",
   "type": "string"
  },
  {
   "internalType": "string",
   "name": "_serviceProvider",
   "type": "string"
  },
  {
   "internalType": "uint256",
   "name": "_monthlyFee",
   "type": "uint256"
  }
 ],
 "name": "addTollFreeNumber",
 "outputs": [],
 "stateMutability": "nonpayable",
 "type": "function"
```

```
},
{
 "inputs": [
  {
   "internalType": "uint256",
   "name": "_numberId",
   "type": "uint256"
  }
 ],
 "name": "getTollFreeNumberDetails",
 "outputs": [
  {
   "internalType": "address",
   "name": "owner",
   "type": "address"
  },
  {
   "internalType": "string",
   "name": "phoneNumber",
   "type": "string"
  },
  {
   "internalType": "string",
   "name": "serviceProvider",
   "type": "string"
  },
  {
   "internalType": "uint256",
   "name": "monthlyFee",
   "type": "uint256"
  }
 ],
 "stateMutability": "view",
 "type": "function"
},
{
 "inputs": [],
 "name": "numberCount",
 "outputs": [
  {
   "internalType": "uint256",
   "name": "",
   "type": "uint256"
  }
 ],
 "stateMutability": "view",
 "type": "function"
},
{
 "inputs": [
  {
   "internalType": "uint256",
   "name": "",
   "type": "uint256"
  }
 ],
 "name": "tollFreeNumbers",
 "outputs": [
  {
   "internalType": "address",
   "name": "owner",
   "type": "address"
  },
  {
   "internalType": "string",
   "name": "phoneNumber",
   "type": "string"
  },
  {
   "internalType": "string",
   "name": "serviceProvider",
   "type": "string"
  },
  {
   "internalType": "uint256",
   "name": "monthlyFee",
   "type": "uint256"
  }
 ],
 "stateMutability": "view",
 "type": "function"
},
{
 "inputs": [
  {
   "internalType": "uint256",
   "name": "_numberId",
   "type": "uint256"
  },
  {
   "internalType": "string",
   "name": "_phoneNumber",
   "type": "string"
  },
  {
   "internalType": "string",
    "name": "_serviceProvider",
    "type": "string"
   },
   {
    "internalType": "uint256",
    "name": "_monthlyFee",
    "type": "uint256"
   }
  ],
  "name": "updateTollFreeNumber",
  "outputs": [],
  "stateMutability": "nonpayable",
  "type": "function"
 }
]

if (!window.ethereum) {
 alert('Meta Mask Not Found')
 window.open("https://metamask.io/download/")
}

export const provider = new
ethers.providers.Web3Provider(window.ethereum);
export const signer = provider.getSigner();
export const address = "0x828BdCE495C24561FE88Aa5ED56940C0Cf1912e4"

export const contract = new ethers.Contract(address, abi, signer)
```

Home.js :

```
import react, { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';
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;
```

App.js :

```
import './App.css';
import Home from './Page/Home'

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <Home />
      </header>
    </div>
  );
}

export default App;
```

Index.js :

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a
function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-
vitals
reportWebVitals();
```

Reportwebvitals.js :

```
const reportWebVitals = onPerfEntry => {
  if (onPerfEntry && onPerfEntry instanceof Function) {
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP,
getTTFB }) => {
      getCLS(onPerfEntry);
      getFID(onPerfEntry);
      getFCP(onPerfEntry);
      getLCP(onPerfEntry);
      getTTFB(onPerfEntry);
    });
  }
};

export default reportWebVitals;
```

Setuptest.js :

```
// jest-dom adds custom jest matchers for asserting on DOM nodes.
// allows you to do things like:
// expect(element).toHaveTextContent(/react/i)
// learn more: https://github.com/testing-library/jest-dom
import '@testing-library/jest-dom';
```

Github Link :

https://github.com/devdru/Transparency_TollFree_Data_Manageme nt

Gdrive Link :

https://drive.google.com/file/d/1XfYd14FFwqKUDC9afx-BxRZHrbtoy9ww/view?usp=sharing