SOUTHERN CROSS UNIVERSITY

ASSIGNMENT COVER SHEET

Student Name:	**SAIKIRAN REDDY ANUGULA**
Student ID No.:	**24784524**
Unit Name:	**Blockchain Technology and Applications**
Unit Code:	**INFO6001**
Tutor's name:	**Dr. Seyed Lajevardi**
Assignment No.:	**Assessment 2**
Assignment Title:	**Practical**
Due date:	**04/08/2025**
Date submitted:	**04/08/2025**

Declaration:

I have read and understand the Rules Relating to Awards (Rule 3 Section 18 – Academic Misconduct Including Plagiarism) as contained in the SCU Policy Library. I know the penalties for plagiarism and agree to be bound by these rules. The work I am submitting electronically is entirely my work.

Signed: SAIKIRAN REDDY ANUGULA
(Please type your name.)

Date: 04/08/2025

ASSESSMENT 2

INFO6001

BY SAIKIRAN REDDY ANUGULA

TABLE OF CONTENTS

INTRODUCTION

This report outlines the design and prototype implementation of a blockchain-based system aimed at enhancing traffic data sharing within smart city applications. Built using Solidity smart contracts and deployed in Remix IDE, the solution provides secure, transparent, and decentralized management of traffic-related information contributed by various entities such as sensors and citizens.

IMPORTANCE OF BLOCKCHAIN TECHNOLOGY IN SMART CITY SCENARIOS

Blockchain technology is highly valuable for smart city traffic data sharing because it offers a decentralized, transparent, and tamper-resistant way to collect and distribute real-time traffic information from multiple sources, such as sensors and citizens. By using blockchain, cities can ensure that once traffic data is recorded, it cannot be secretly changed or deleted, increasing trust among all stakeholders.

Additionally, smart contracts can automatically enforce that only verified entities can submit new reports, which helps prevent false or unauthorized data entries. Overall, blockchain provides enhanced data integrity, security, and openness, which are essential for efficient, reliable, and collaborative urban traffic management.

Current Challenges in Traffic Data Management

Traditional traffic data systems typically rely on centralized databases controlled by single authorities or third-party providers. This centralized model presents multiple limitations:

- Data Integrity Risks: Centralized systems are vulnerable to data tampering or accidental errors without full transparency.
- Limited Trust and Accessibility: Stakeholders, including citizens and third-party services, often lack direct access to verify data authenticity.
- Single Point of Failure: System outages or cyber-attacks can disrupt traffic data availability, affecting timely decision-making.

Advantages of Blockchain Technology

Blockchain provides a decentralized ledger maintained across a distributed network, which ensures:

- Transparency: All registered participants have access to immutable traffic reports, fostering trust in the data.
- Security: Data entries are cryptographically secured and cannot be altered retroactively.

- Access Control: Smart contracts enforce registration of entities and restrict data submission to authorized participants only, mitigating the risk of false reports.
- Real-Time Sharing: Data from multiple sources (e.g., sensors, citizens) can be collated honestly and immediately, improving traffic management responsiveness.

Therefore, blockchain presents an effective approach to overcome challenges in urban traffic data systems by enabling an inclusive, secure, and reliable data-sharing platform.

PROTOTYPE OVERVIEW

I have tried to implement the traffic data sharing using two smart contracts.

1. EntityManager Contract
2. TrafficDataChain Contract

The EntityManager contract serves as the foundation for the system by managing the registration of various participants in the smart city network. Entities such as citizens, sensors, or officials can register themselves by providing basic information including their name, type, and Ethereum address. The contract enforces rules to prevent duplicate registrations, ensuring that each address corresponds to only one registered entity. It maintains a mapping of addresses to entity details and stores a list of all registered addresses for easy retrieval. Additionally, the contract provides functions to verify if an entity is registered and to access detailed information about any registered user, thereby supporting identity management and controlled participation within the system.

Building on this, the TrafficDataChain contract extends the functionality of the EntityManager by enabling registered entities to submit traffic-related data, such as accident reports or congestion updates. Each traffic data entry records the reporter's address, a description of the event, the type of data, and a timestamp marking when it was submitted. Importantly, the contract restricts data submissions to only those addresses previously registered, enhancing data integrity and trustworthiness. Submitted reports are stored sequentially in an array and emit events upon creation, effectively mimicking the concept of adding blocks to a blockchain. The contract also provides methods to retrieve the total number of traffic reports and to access individual entries by index, facilitating transparent data sharing and real-time monitoring within the smart city environment.

DEMONSTRATION OF SMART CONTRACTS IMPLEMENTATION

The prototype built for smart city traffic data sharing involves several key functions that align with real-world business requirements, such as registering participants, submitting verified traffic data, validating transactions, and retrieving information for transparency. The following demonstration outlines these core functionalities through practical scenarios, illustrating how the system behaves under typical interactions.

Entity Registration

The system begins by allowing various participants—such as traffic sensors, city officials, or regular citizens—to register themselves as entities within the network. Each entity registers by providing a name and a type (for example, "Sensor" or "Citizen"). This registration step is crucial because only registered entities can participate in submitting traffic data.

To showcase this, suppose two participants, "Sensor 1" and "Alice", want to join. When Sensor 1 calls the registration function from its own Ethereum address (simulated in Remix by selecting Account #1), the contract records their details, prevents duplicate registrations, and confirms the registration by emitting an event. Similarly, Alice registers from a different Ethereum address (Account #2) and her information is stored in the system. If Alice attempts to register a second time from the same address, the system gracefully rejects this attempt with a clear error message stating she is already registered. Throughout this process, functions like getAllEntities allow anyone to retrieve a list of all registered addresses, while getEntity returns detailed information for a given entity, such as their name, type, and registration status.

Traffic Data Submission

Once entities are registered, they can contribute traffic reports to the system, representing real-time data flowing into the network. For example, Sensor 1 may submit a report indicating an accident at a certain location, while Alice might report congestion in a busy downtown area. Both these data submissions include the type of incident (“Accident” or “Congestion”), details describing the event, the reporter’s address, and the exact timestamp of submission.

The system enforces strict control to prevent unregistered participants from submitting traffic data. If an unregistered user (for instance, from Account #3) attempts to contribute a traffic report, the contract immediately reverts the transaction and displays an error such as “Not a registered entity.” This protects the network from unauthorized or potentially false data entries, thereby maintaining data integrity.

Each successful submission emits a TrafficDataSubmitted event containing the reporter’s address, index of the new traffic report, and the data type, providing transparency and traceability. Users can verify the number of traffic reports with the getTrafficDataCount function and access specific entries by their index

via getTrafficFeedAt. These retrieval functions return comprehensive details about each report, ensuring that the traffic data is easily accessible and verifiable by any participant.

Querying Registered Entities and Traffic Data

The system provides comprehensive methods to retrieve and inspect both entities and traffic reports. For example, running getAllEntities returns an array of addresses registered in the network, allowing a city authority or public interface to present a directory of all authorized data reporters. Additionally, the getEntity function reveals detailed information about any participant, facilitating transparency about which users contribute data and their associated metadata.

For the reported traffic data, getTrafficDataCount offers a straightforward way to determine how many events have been logged, while getTrafficFeedAt retrieves detailed descriptions, reporter identities, and timestamps for specific entries. This combination allows for real-time monitoring and historical auditing of traffic events, essential for informed decision-making in urban management.

Data Validation

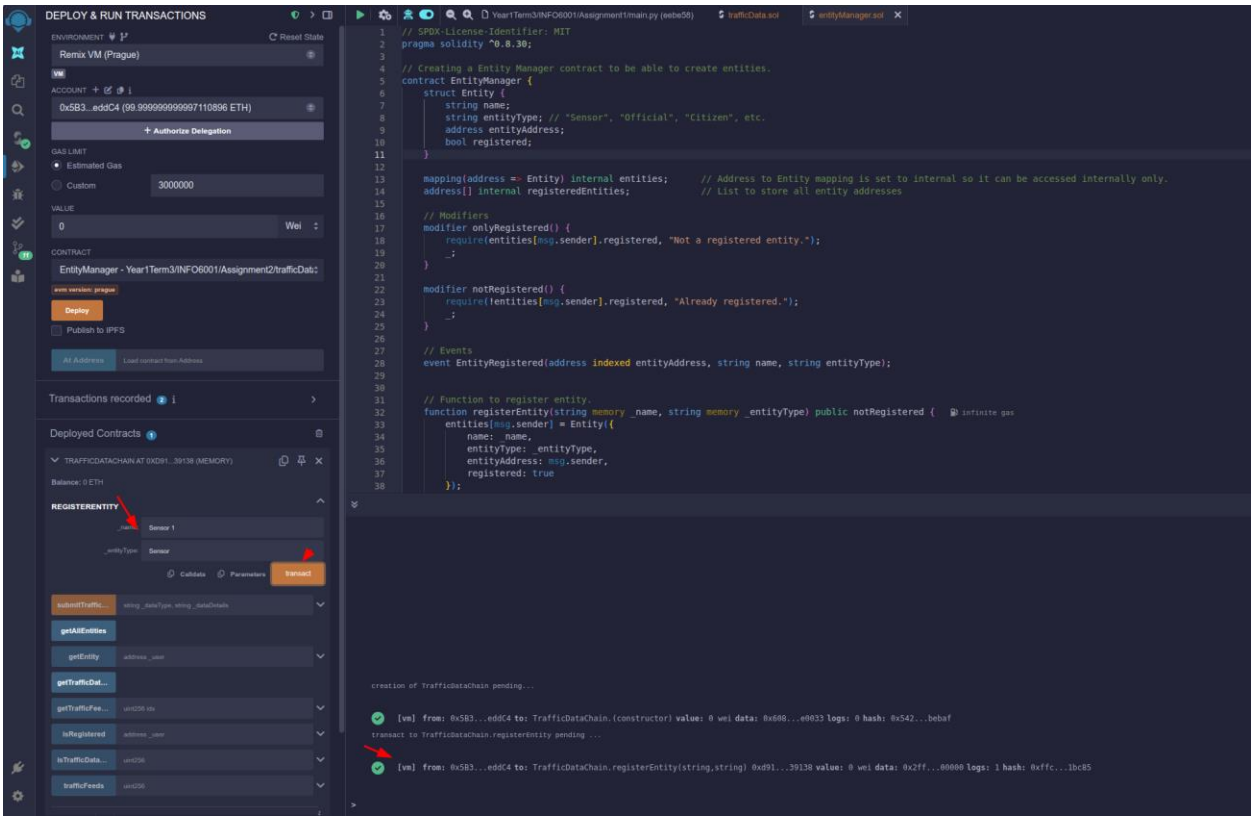
While my current implementation includes an isTrafficDataValid function acting as a placeholder that always returns true, its presence illustrates the system's extensibility for advanced validation. I have not been able to validate the data using the function that could enforce additional logic, such as verifying reporter credentials, filtering spam, or cross-referencing data consistency.

I would be implementing the above functionality before assignment3.

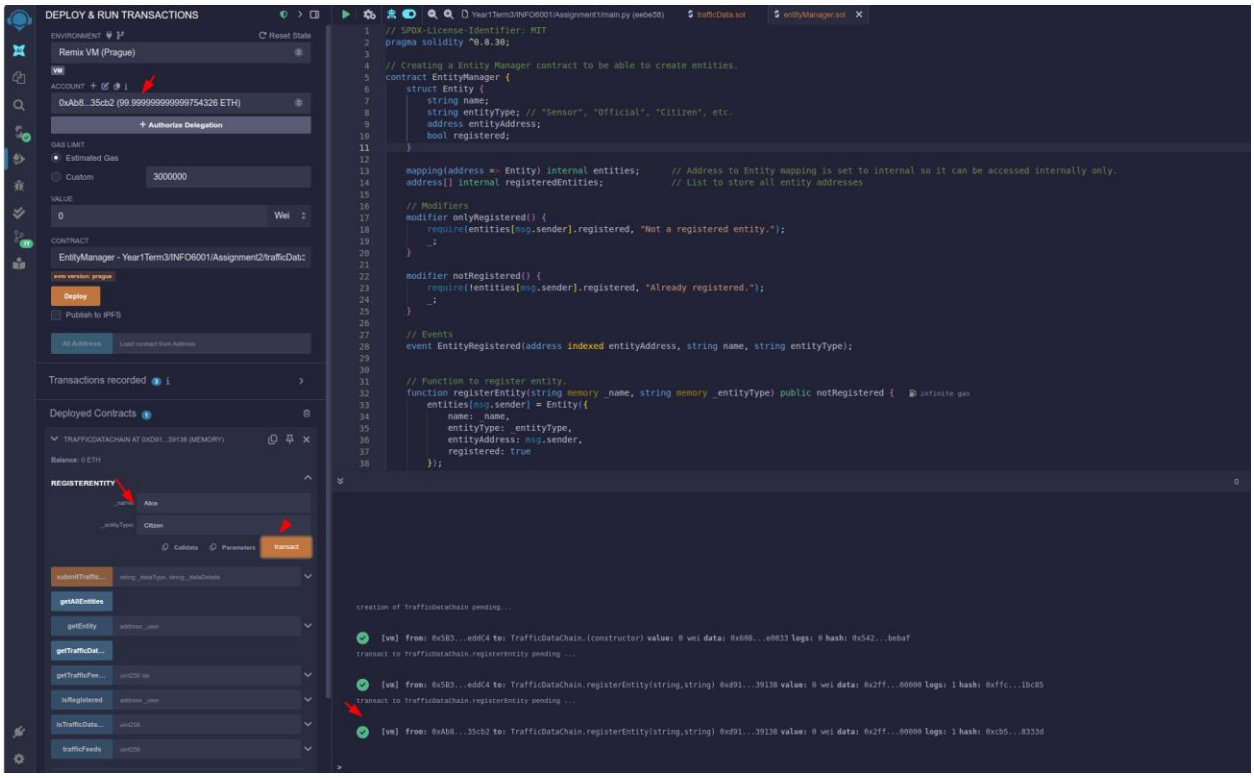
DEMO BY DEPLOYING SMART CONTRACTS

I have added a sensor entity by deploying the contract trafficData because it already inherits all the functionality of the entityManager contract.

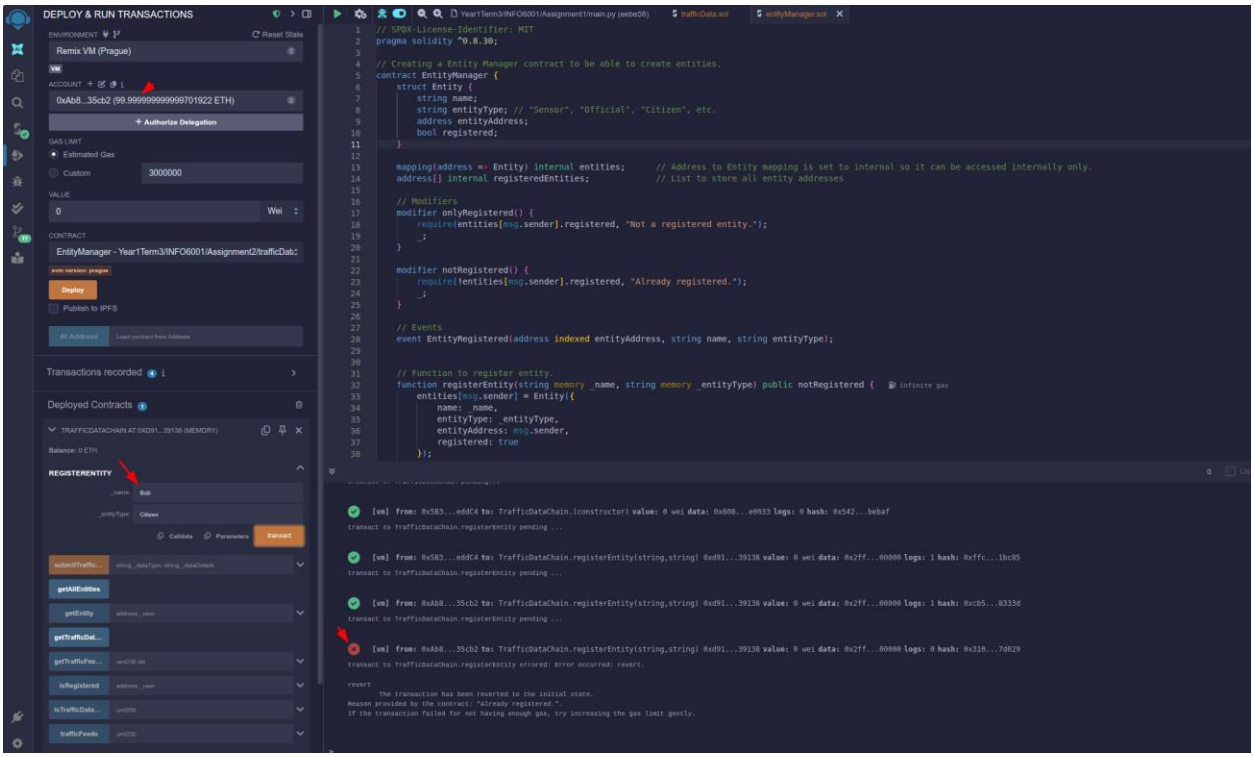
A Sensor 1 entity with type as Sensor and address 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 has been added to the blockchain.



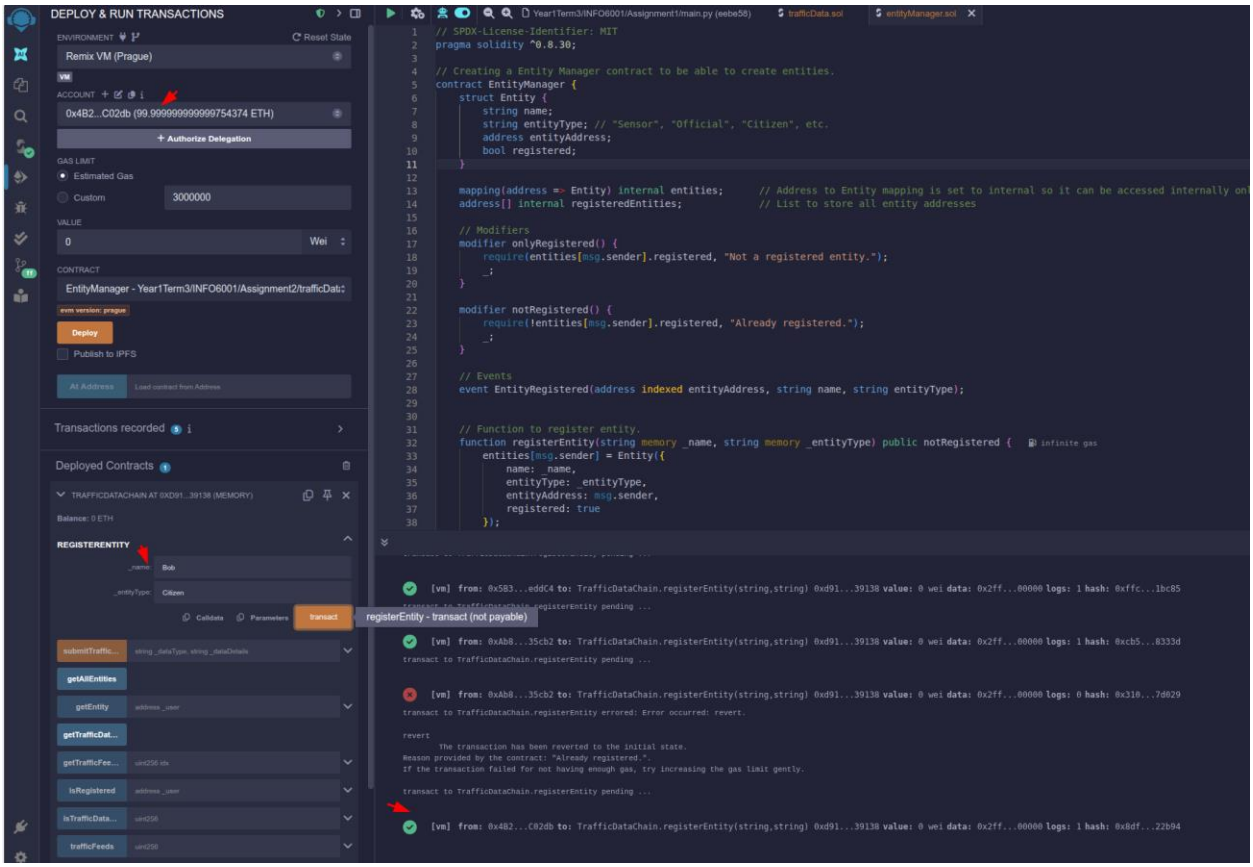
I have changed the Ethereum address from the dropdown of Account and added the entity as ‘Alice’ with entity type as ‘Citizen’ and address 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 and made the transaction.



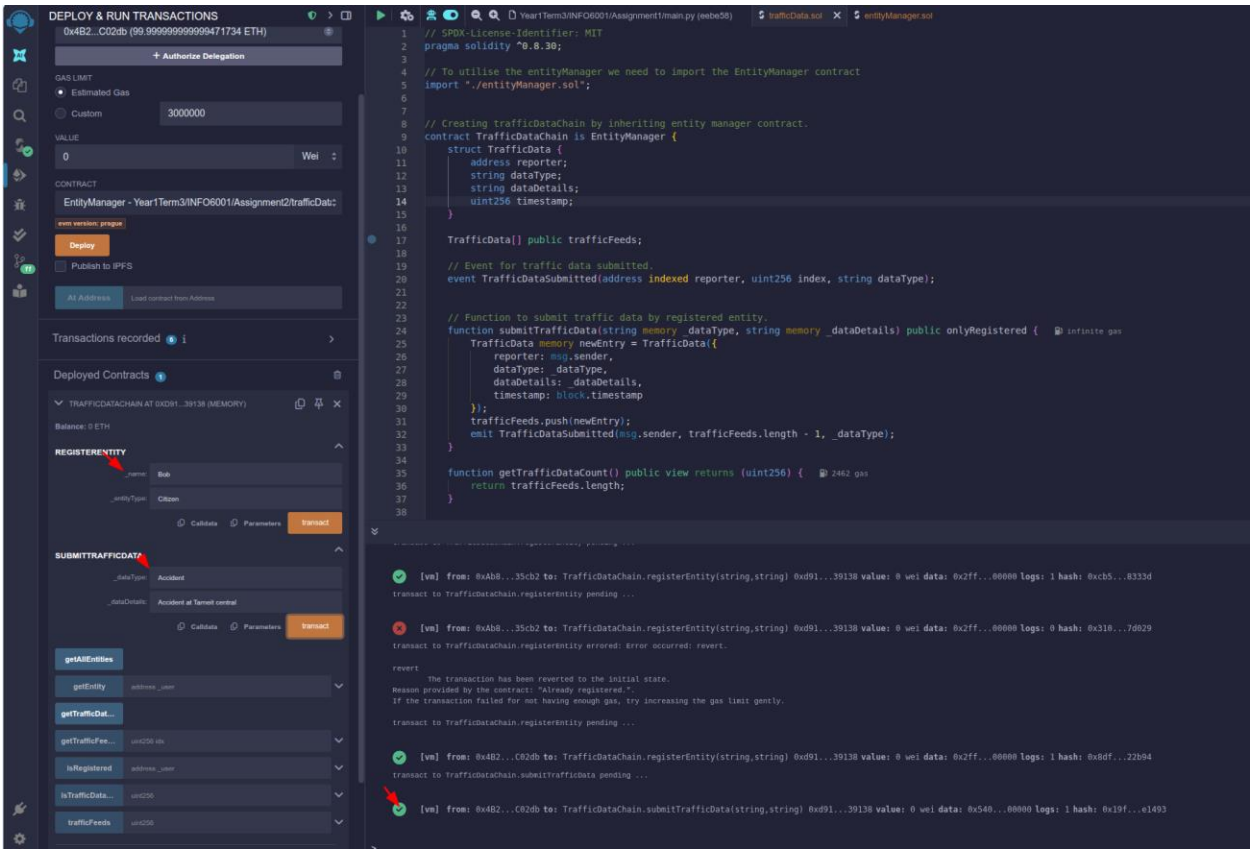
Now if I try to add another entity using the same address 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 the smart contract doesn't let me create one.



I have changed the Ethereum account address and created a new entity ‘Bob’ with type ‘Citizen’ with address 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db.



Entity ‘Bob’ creates a new traffic data submission as type ‘Accident’ and details ‘Accident at Tarneit Central’ and hits transact. It will now be added to the blockchain.



By Clicking on the getAllEntities we get the addresses of all the entities we have just created.

To get the information for a specific entity we can use the getEntity function all by passing it the address, lets pass address for ‘Bob’ 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db.

We could see that only the information of entity ‘Bob’ has been returned.

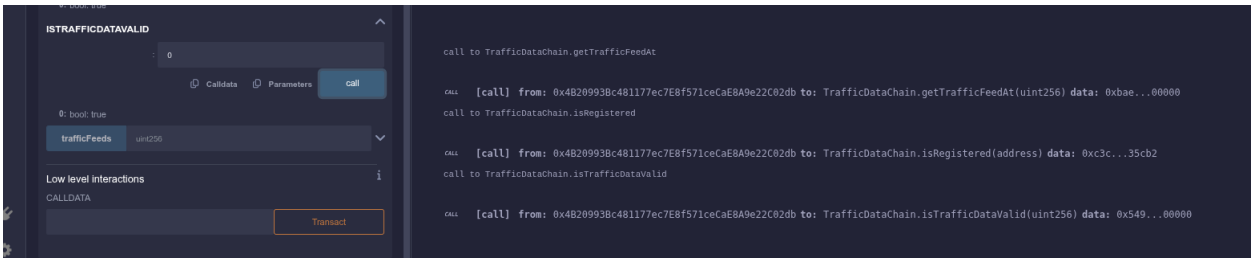
Because we have only added traffic data from entity ‘Bob’ we get only 1 result when the getTrafficData is called.

When we call the getTrafficData function by passing the index of the trafficDataSubmitted (in this case we have only submitted one data from entity ‘Bob’). So, passing in ‘0’ gives us the traffic Data submitted by entity ‘Bob’.

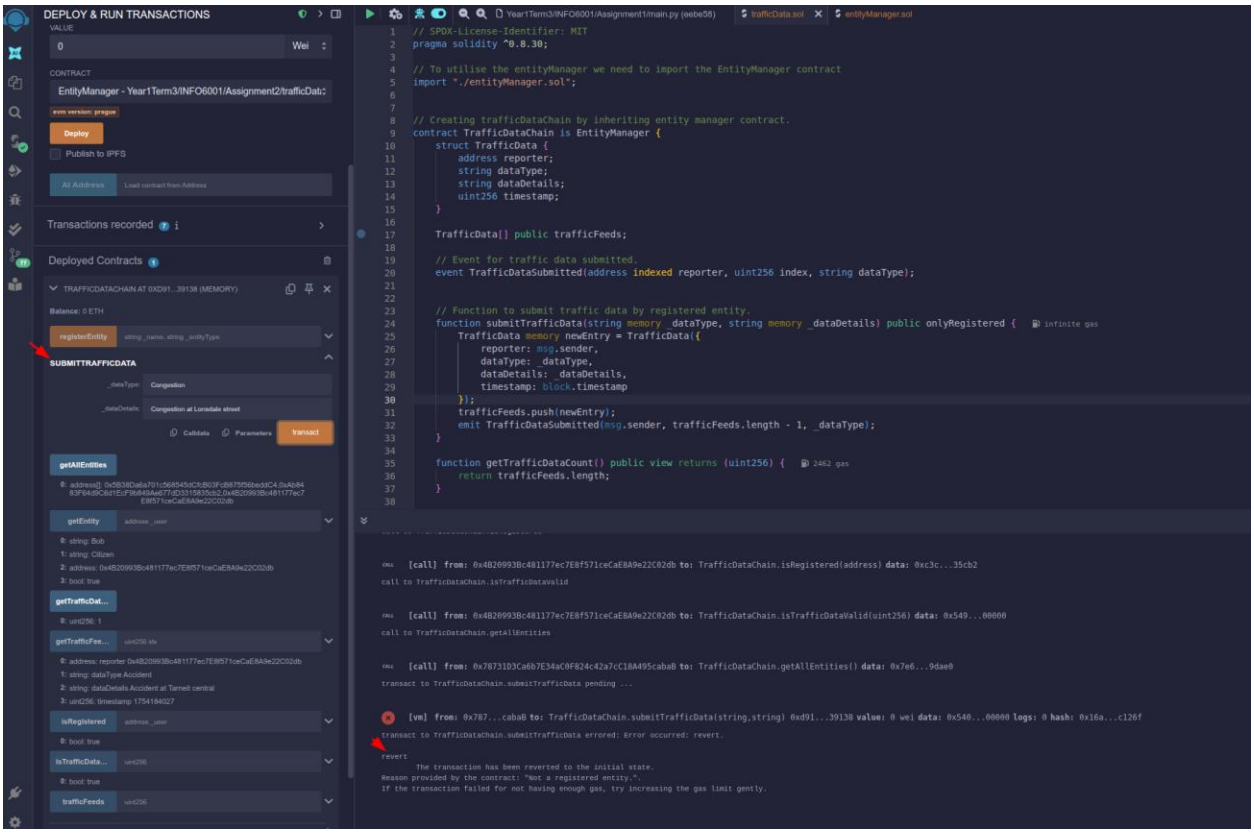
Let’s validate if the entity ‘Alice’ whom we created earlier is properly registered by passing the address of ‘Alice’ 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to isRegistered function.

We can see that the function returns a boolean value ‘true’ which indicates the entity ‘Alice’ has been registered.

I'm still working on the is traffic data valid function which needs to check if the data submitted is valid. For now, the function returns true for any index value submitted to it.

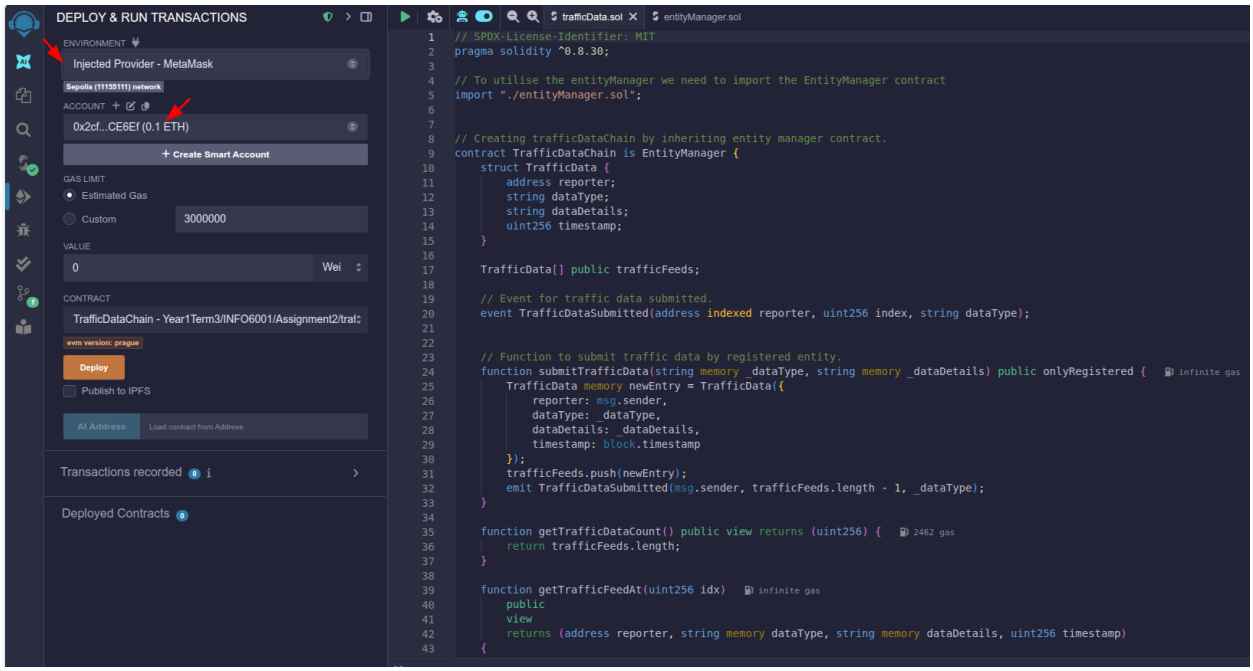


I have changed the address and then tried to submit a new traffic event called ‘Congestion’ with details ‘Congestion at lonsdale street’ but the smart contract wouldn’t allow me to transact and throws out and error saying, “not a registered entity’.

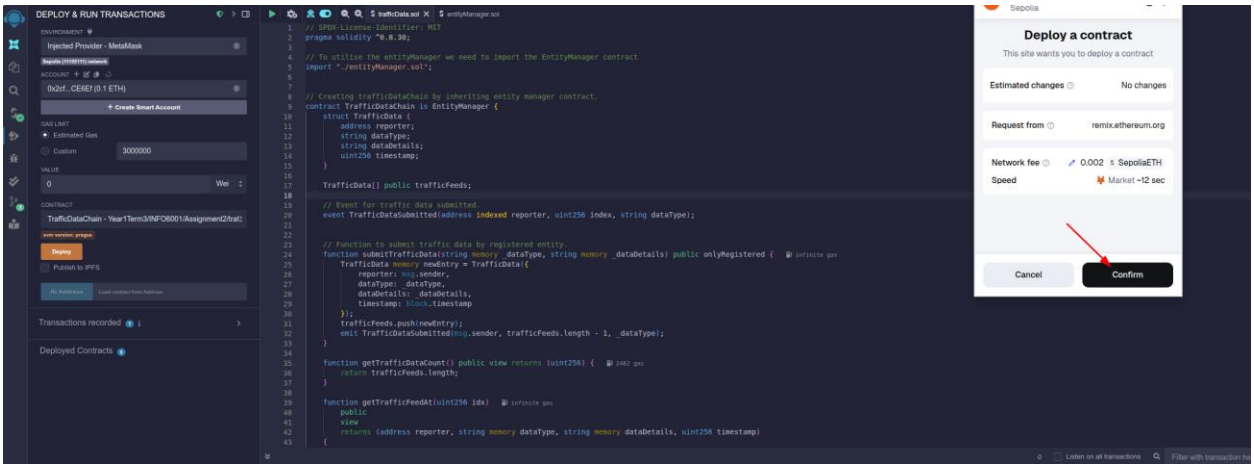


Deploying on Sepolia Network

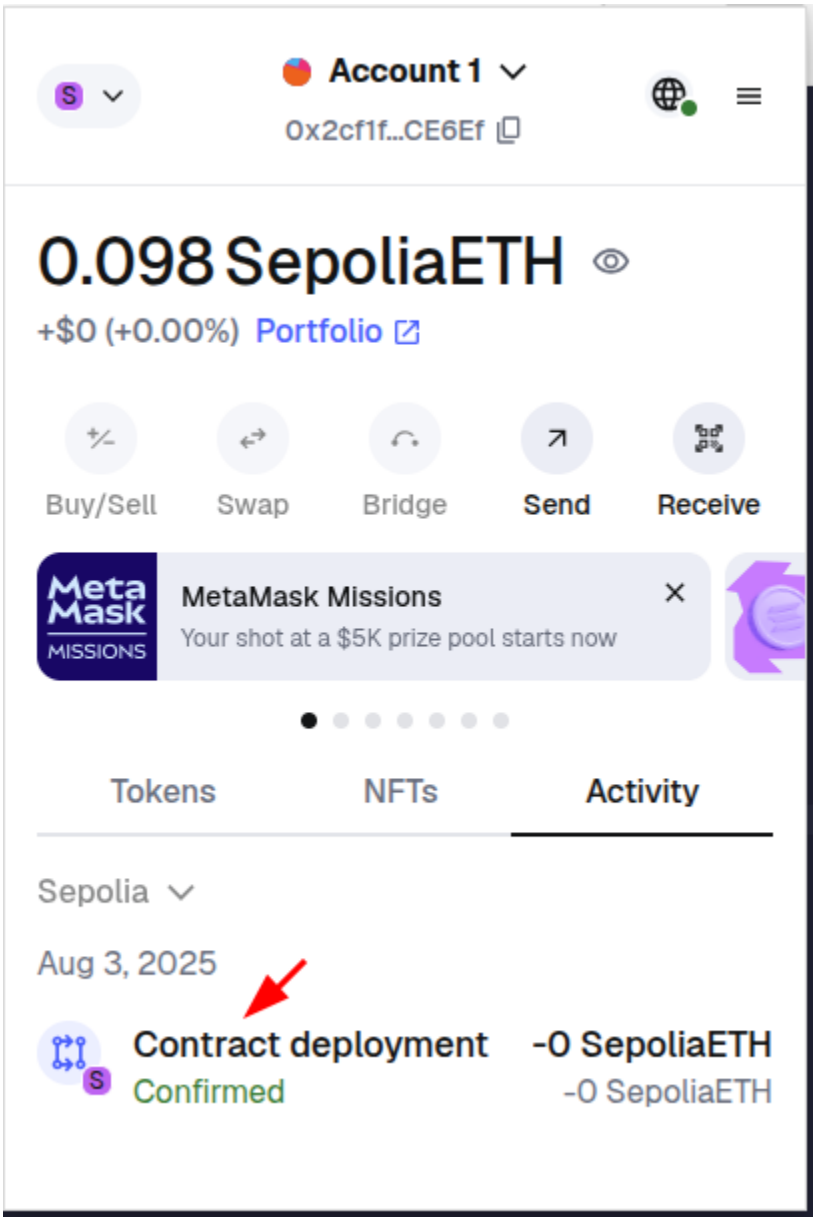
I have connected my workspace in remix ide to MetaMask and then selected Sepolia as the network.



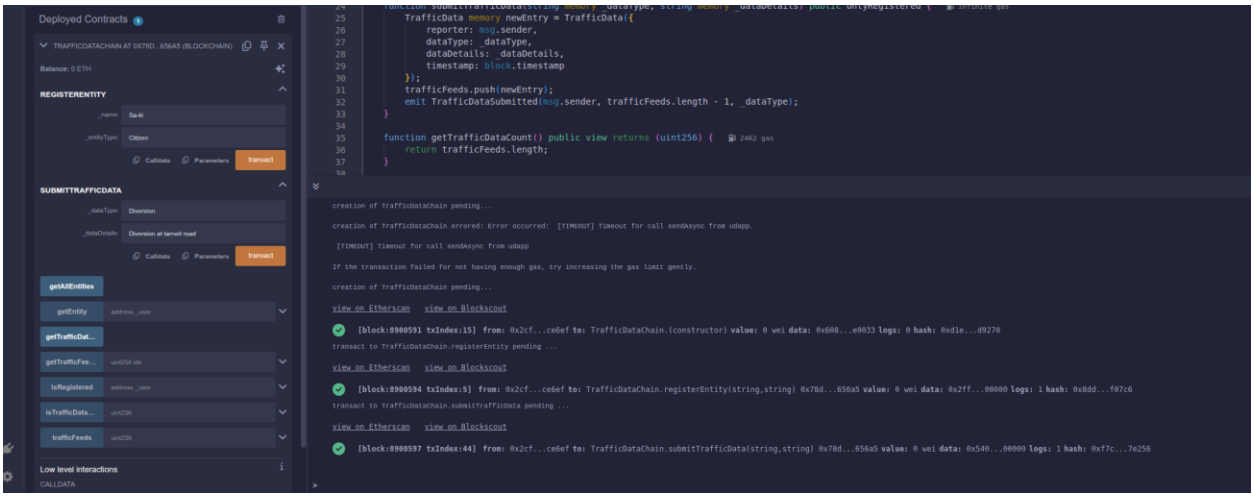
I will now deploy the smart contract on sepolia network by clicking on deploy, can the approve the transaction on metamask.

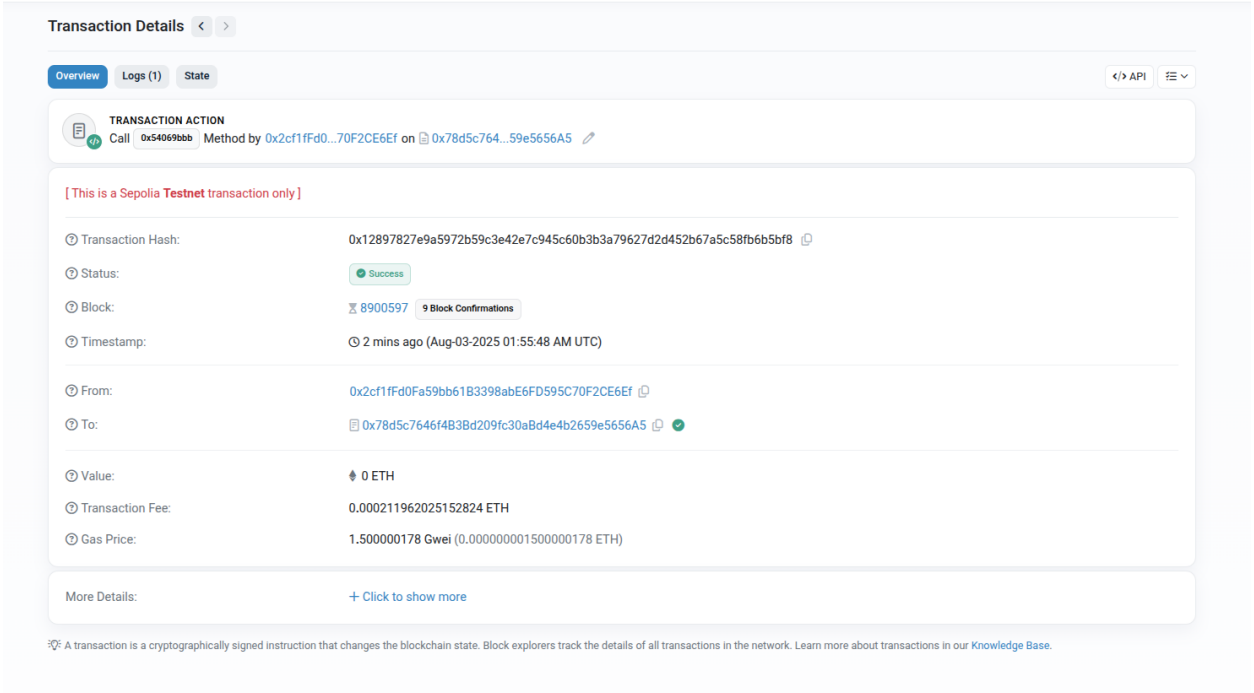


The transaction has been approved, and we can view it on the Etherscan page.



I have then created an entity ‘Sa-ki’ as ‘Citizen’ and updated a new traffic data entry as ‘Diversion’ at ‘Diversion at tarneit road’ and completed the transaction which can be verified on the Etherscan.


Deployed Contracts
REGISTERENTITY
SUBMITTRAFFICDATA
transact
getAllEntities
Low level interactions
TrafficData memory newEntry = TrafficData({
reporter: msg.sender,
dataType: _dataType,
dataDetails: _dataDetails,
timestamp: block.timestamp
});
trafficFeeds.push(newEntry);
emit TrafficDataSubmitted(msg.sender, trafficFeeds.length - 1, _dataType);
function getTrafficDataCount() public view returns (uint256) {
return trafficFeeds.length;
creation of TrafficDataChain pending...
creation of TrafficDataChain errored: Error occurred: [TIMEOUT] Timeout for call sendAsync from udapp.
[TIMEOUT] Timeout for call sendAsync from udapp
If the transaction failed for not having enough gas, try increasing the gas limit gently.
view on Etherscan
view on Blockscout
transact to TrafficDataChain.registerEntity pending ...
transact to TrafficDataChain.submitTrafficData pending ...


Transaction Details
Overview
Logs (1)
State
API
TRANSACTION ACTION
Call 0x54069bbb Method by 0x2cf1fFd0...70F2CE6Ef on 0x78d5c764...59e5656A5
[This is a Sepolia Testnet transaction only]
Transaction Hash:
0x12897827e9a5972b59c3e42e7c945c60b3b3a79627d2d452b67a5c58fb6b5bf8
Status:
Success
Block:
8900597
9 Block Confirmations
Timestamp:
2 mins ago (Aug-03-2025 01:55:48 AM UTC)
From:
0x2cf1fFd0Fa59bb61B3398abE6FD595C70F2CE6Ef
To:
0x78d5c7646f4B3Bd209fc30aBd4e4b2659e5656A5
Value:
0 ETH
Transaction Fee:
0.000211962025152824 ETH
Gas Price:
1.500000178 Gwei (0.000000001500000178 ETH)
More Details:
+ Click to show more
A transaction is a cryptographically signed instruction that changes the blockchain state. Block explorers track the details of all transactions in the network. Learn more about transactions in our Knowledge Base.

GenAI Use Declaration

I acknowledge that I have used GenAI tools to complete this assessment. I used Perplexity tool to understand how to work on the report template for demonstrating the working process of my trafficDataSharing smart contracts within the parameters outlined in the Assessment Brief and by the Unit Assessor.

Saikiran Reddy Anugula

24784524