Contents

[Chapter 1 Team Information 3](#_Toc147182875)

[Batch 3](#_Toc147182876)

[Members 3](#_Toc147182877)

[Github Repository 3](#_Toc147182878)

[Chapter 2 Project Goals 4](#_Toc147182879)

[Blockchain 4](#_Toc147182880)

[Smart Contract 4](#_Toc147182881)

[Basic 4](#_Toc147182882)

[Advanced 4](#_Toc147182883)

[Chapter 3 Assumptions 5](#_Toc147182884)

[Chapter 4 Deliverables 5](#_Toc147182885)

[Chapter 5 Blockchain node and network setup, solidity contract code, configuration settings. 6](#_Toc147182886)

[AWS Infrastructure & Configuration 6](#_Toc147182887)

[Security Group Configuration 7](#_Toc147182888)

[List of Commands used to create and set up the nodes and accounts. 8](#_Toc147182889)

[Initial Set up 8](#_Toc147182890)

[Nodes & Accounts 9](#_Toc147182891)

[Add as Peers 11](#_Toc147182892)

[Connectivity Check 12](#_Toc147182893)

[MetaMask Configuration 13](#_Toc147182894)

[Solidity code of the contract(s) 14](#_Toc147182895)

[Chapter 6 Overall Design 15](#_Toc147182896)

[Data Structures 15](#_Toc147182897)

[Functions 15](#_Toc147182898)

[Cancellation / Delay Penalties – 16](#_Toc147182899)

[Cancellation by Customer 16](#_Toc147182900)

[Cancellation by Airline - Refund Rules 16](#_Toc147182901)

[Delayed by Airline – Penalty Rules 16](#_Toc147182902)

[Remix – Execution Screenshot 17](#_Toc147182903)

[Remix – Execution Log 17](#_Toc147182904)

# Team Information

## Batch

* August 2022 – November 2022

## Members

* **Group Lead**: Mohan Sami
* Anuradha Kapoor
* Reema Chhetri
* Sachin Ghewde

## Github Repository

<https://github.com/sachin-gg/ACSECapstone>

# Project Goals

Developed a unique blockchain based ticket management system for Eagle Airlines to entice users with transparency, and automated refunds and delay penalties.

## Blockchain

* Develop a Private Ethereum Blockchain implementation, using **geth** nodes running directly on a 3 **AWS EC2** (Ubuntu) servers.
* Use **Clique PoA** (Proof of Authority) as the consensus protocol.
* Develop a base Flight ticket management smart contract in **Solidity**.
* Use **MetaMask** as the wallet for Customers.
* Demonstrate contract behavior via **Remix** connected to the private blockchain.

## Smart Contract

### Basic

* The customer should be able to trigger a cancellation anytime till 2 hours before the flight start time. This should refund money to the customer minus the percentage penalty predefined in the contract by the airlines. The penalty amount should be automatically sent to the airline account.
* Any cancellation triggered by the airline before or after departure time should result in a complete amount refund to the customer.
* The airline should update the status of the flight within 24 hours of the flight’s start time. It can be on-time start, cancelled, or delayed.
* 24 hours after the flight departure time, the customer can trigger a claim function to demand a refund.
  + They should get a complete refund in case of cancellation by the airline.
  + In case of a delay, they should get a predefined percentage amount, and the rest should be sent to the airline.
  + If the airline hasn’t updated the status within 24 hours of the flight departure time, and a customer claim is made, it should be treated as an airline cancellation case by the contract.

### Advanced

* Add support for multiple cancellation penalties in favor of the airline, and delay penalties in favor of the customer, based on various time ranges in the contract.

# Assumptions

1. All flights travel nonstop from origin to destination.
2. All seats will be the same class (like economy) and will be priced the same (10 ETH for domestic or 25 ETH for International).
3. Three Airline (miner) accounts will be created through genesis configuration file and prefunded with 9000 ETH each.
4. Four customer accounts will be created for ease of simulation through genesis configuration file and prefunded with 9000 ETH each.
5. Airline cancellations should result in 100% refund to the customer immediately without any claim process.
6. Customers can claim a refund for cancellation anytime before 2 hours of the flight’s scheduled departure time. Cancellation penalties need to follow the rules below for refund.

|  |  |
| --- | --- |
| Rule: Based on DIFFERENCE of (Scheduled Departure Datetime – Cancellation Datetime) | Refund Amount  (% of Ticket Price) |
| If >= 24 hours | 100% |
| If >= 10 hours and < 24 hours | 80% |
| If >= 2 hour and < 10 hours | 40% |
| If < 2 hour | 0% |

1. Flight delays by airline will make customers eligible for a refund. These rules will apply in 2 scenarios.
   1. Flight lands and the Airline contract automatically closing the ticket.
   2. Flight has not landed, then the customer can claim a refund after 24 hours of scheduled departure time.

|  |  |
| --- | --- |
| Rule: Based on DIFFERENCE of (Scheduled Departure Datetime – Actual Departure Datetime) | Refund Amount  (% of Ticket Price) |
| If < 2 hours | 0% |
| If >= 2 hours and < 10 hours | 10% |
| If >= 10 hours and < 24 hours | 40% |
| If >= 24 hours | 100% |
| If Airline failed to provide any status updates within 24 hours of scheduled departure time. | 100% |

# 

# Deliverables

1. All the code including an aggregated script of commands used to create and set up the nodes and accounts (with comments), the actual solidity code of the contract(s), configuration settings (textual README and/or screenshots of AWS console screen configurations), etc.
2. Architecture diagram showing your entire system architecture.
3. Multiple screenshots highlighting interaction with your system and its behavior.
4. Presentation talking about the process and the output.

# Eaglepoa network Layout

A diagram of a blockchain based ticket management system

Description automatically generated

# AWS Infrastructure and Configuration

A computer screen with a white screen

Description automatically generated

### Security Group Configuration

* Security – Inbound Rules (Required for MetaMask & Remix):

|  |  |  |
| --- | --- | --- |
| * Protocol | * Port | * Source |
| * TCP (listener) | * 9001 | * 0.0.0.0/0 |

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Blockchain node setup

## List of Commands used to set up the nodes and accounts.

Ethereum & Geth **Github**: <https://github.com/sachin-gg/ACSECapstone/blob/main/FinalDocs/readme.md>

### Gensis File (eaglepoa.json)

      "config": {

      "chainId": 80801,

      "homesteadBlock": 0,

      "eip150Block": 0,

      "eip155Block": 0,

      "eip158Block": 0,

      "byzantiumBlock": 0,

      "constantinopleBlock": 0,

      "berlinBlock": 0,

      "clique": {

            "period": 30,

            "epoch": 30000

      }

      },

      "difficulty": "1",

      "gasLimit": "8000000",

      "extradata": "0x0000000000000000000000000000000000000000000000000000000000000000168AaBAc0c700eF95269b0876931F6ECbBbbE5970000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",

      "alloc": {

            "168AaBAc0c700eF95269b0876931F6ECbBbbE597": { "balance": "9000000000000000000000" },

            "c2bdaE1D229f9554A676081FD68916e93C8420c9": { "balance": "9000000000000000000000" },

            "406CC1CF7bADb242eF5fC2d64Af68677F36B5fAb": { "balance": "9000000000000000000000" },

            "d9cAa7ecb5500c5FafcA1F4A954Db75EA34c3361": { "balance": "9000000000000000000000" },

            "A4A19c7b7cAfE09063498F697E647971dC236dca": { "balance": "9000000000000000000000" },

            "E69F4c213c19a6E421869e3a62162f92dDE3050E": { "balance": "9000000000000000000000" },

            "0F97E2dFA64A4ceDcAd9F2Ad5f00578308d73b2e": { "balance": "9000000000000000000000" }

      }

}

### 

### Nodes & Accounts

* Network ID: 80801
* Total Three nodes. No Bootnode
* Total Seven Accounts
  + 1 Contract Owner/Airline Account

"168AaBAc0c700eF95269b0876931F6ECbBbbE597": { "balance": "9000000000000000000000" },

* + 1 Airline Domestic account

"c2bdaE1D229f9554A676081FD68916e93C8420c9": { "balance": "9000000000000000000000" },

* + 1 Airline International account

"406CC1CF7bADb242eF5fC2d64Af68677F36B5fAb": { "balance": "9000000000000000000000" },

* + 4 **Customer** accounts on Eagle1 node:

"d9cAa7ecb5500c5FafcA1F4A954Db75EA34c3361": { "balance": "9000000000000000000000" },

"A4A19c7b7cAfE09063498F697E647971dC236dca": { "balance": "9000000000000000000000" },

"E69F4c213c19a6E421869e3a62162f92dDE3050E": { "balance": "9000000000000000000000" },

"0F97E2dFA64A4ceDcAd9F2Ad5f00578308d73b2e": { "balance": "9000000000000000000000" }

* **Node: Eagle1** (Node-1): Airline Miner

A screenshot of a computer screen

Description automatically generated

* **Node: Eagle2** (Node-2): Domestic

A screenshot of a computer program

Description automatically generated

* **Node: Eagle3** (Node-3): International

A screenshot of a computer program

Description automatically generated

### Add as Peers

Eagle1 as peer of Eagle2

A screenshot of a computer

Description automatically generated

Eagle1 as peer of Eagle3

A screenshot of a computer

Description automatically generated

### Connectivity Check

Getting Balance of one node from other nodes

A screen shot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generated

### MetaMask Configuration

Network name: eaglepoa (connect to Node - Eagle1).

New RPC URL: http://[EC2 Public IP]:9001

Chain ID: 80801

Currency symbol: ETH

# Solidity contract Setup

## Solidity code of the contract(s)

* **<README.md>**: <https://github.com/sachin-gg/ACSECapstone/blob/main/eaglepoa/readme.md>
* **<EagleTicket.sol>**: <https://github.com/sachin-gg/ACSECapstone/blob/main/solidity/EagleTicket.sol>
* **<EagleAirline.sol>**: <https://github.com/sachin-gg/ACSECapstone/blob/main/solidity/EagleAirline.sol>
* **<EagleLib.sol>:** <https://github.com/sachin-gg/ACSECapstone/blob/main/solidity/EagleLib.sol>

# Overall Design

## Data Structures & Functions

A diagram of a computer program

Description automatically generated with medium confidence

A diagram of a company

Description automatically generated

A diagram of flight operations

Description automatically generated

A diagram of a flight

Description automatically generated with medium confidence

## Remix – Execution Screenshot

Eagle Factory

Eagle Airline

Eagle Ticket contract

Detailed screenshots – Ticket, Register etc, Each function to be executed.

Set of fligts

A screenshot of a computer program

Description automatically generated

## Remix – Execution Log

**Welcome to Remix 0.35.1**

Your files are stored in indexedDB, 63.27 MB / 284.83 GB used

You can use this terminal to:

* Check transactions details and start debugging.
* Execute JavaScript scripts:  
  *- Input a script directly in the command line interface*   
  *- Select a Javascript file in the file explorer and then run `remix.execute()` or `remix.exeCurrent()` in the command line interface*   
  *- Right click on a JavaScript file in the file explorer and then click `Run`*

The following libraries are accessible:

* [web3 version 1.5.2](https://web3js.readthedocs.io/en/1.0/)
* [ethers.js](https://docs.ethers.io/)
* remix

Type the library name to see available commands.

creation of EagleTicket pending...

**[vm]**

**from:** 0x5B3...eddC4

**to:** EagleTicket.(constructor)

**value:** 0 wei

**data:** 0x608...10033

**logs:** 1

**hash:** 0xef1...a8bf4

**Debug**

|  |  |
| --- | --- |
| **status** | true Transaction mined and execution succeed |
| **transaction hash** | 0xef1268e9fb776fbe60061819207abeaeaa582b0000776420b477957efeda8bf4 |
| **block hash** | 0xcb42c55e6885d472b42221627ae8a46b9118aa7d1476df5ed9d26acf727fec81 |
| **block number** | 1 |
| **contract address** | 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **from** | 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 |
| **to** | EagleTicket.(constructor) |
| **gas** | 5044033 gas |
| **transaction cost** | 4387371 gas |
| **execution cost** | 4033695 gas |
| **input** | 0x608...10033 |
| **decoded input** | {} |
| **decoded output** | - |
| **logs** | [ { "from": "0xd9145CCE52D386f254917e481eB44e9943F39138", "topic": "0x8be0079c531659141344cd1fd0a4f28419497f9722a3daafe3b4186f6b6457e0", "event": "OwnershipTransferred", "args": { "0": "0x0000000000000000000000000000000000000000", "1": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4", "previousOwner": "0x0000000000000000000000000000000000000000", "newOwner": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4" } } ] |
| **val** | 0 wei |

call to EagleTicket.owner

***CALL*[call]**

**from:** 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

**to:** EagleTicket.owner()

**data:** 0x8da...5cb5b

**Debug**

|  |  |
| --- | --- |
| **from** | 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 |
| **to** | EagleTicket.owner() 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **execution cost** | 2611 gas (Cost only applies when called by a contract) |
| **input** | 0x8da...5cb5b |
| **decoded input** | {} |
| **decoded output** | { "0": "address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4" } |
| **logs** | [] |

transact to EagleTicket.registerAirline errored: Error encoding arguments: Error: invalid BigNumber string (argument="value", value="", code=INVALID\_ARGUMENT, version=bignumber/5.7.0)

transact to EagleTicket.registerAirline pending ...

**[vm]**

**from:** 0xAb8...35cb2

**to:** EagleTicket.registerAirline(uint8,string) 0xd91...39138

**value:** 0 wei

**data:** 0x2a6...00000

**logs:** 0

**hash:** 0x2c1...160d3

**Debug**

|  |  |
| --- | --- |
| **status** | true Transaction mined and execution succeed |
| **transaction hash** | 0x2c136b98fac522e56e074900617fc4a9f3f4bbca89a7b83d6ddd979bd34160d3 |
| **block hash** | 0x2d53b52541771c1e8a787019d8c4d3a1085121b5dbae3759bb56a748297589d0 |
| **block number** | 2 |
| **from** | 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 |
| **to** | EagleTicket.registerAirline(uint8,string) 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **gas** | 82288 gas |
| **transaction cost** | 71554 gas |
| **execution cost** | 49858 gas |
| **input** | 0x2a6...00000 |
| **decoded input** | { "uint8 airlineType": 0, "string airlineName": "Domestic" } |
| **decoded output** | { "0": "bool: true", "1": "string: !INFO! Airline has been setup." } |
| **logs** | [] |
| **val** | 0 wei |

transact to EagleTicket.registerAirline pending ...

**[vm]**

**from:** 0x4B2...C02db

**to:** EagleTicket.registerAirline(uint8,string) 0xd91...39138

**value:** 0 wei

**data:** 0x2a6...00000

**logs:** 0

**hash:** 0xaf0...e00f8

**Debug**

|  |  |
| --- | --- |
| **status** | true Transaction mined and execution succeed |
| **transaction hash** | 0xaf0cc8f398b4e1ab74a8ac3600fc8819df8e4825466dc02222ace912f1fe00f8 |
| **block hash** | 0x3894acb2ca7c3cdc67733a18eae8a412927f2c615639324d9bae308b8577d137 |
| **block number** | 3 |
| **from** | 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db |
| **to** | EagleTicket.registerAirline(uint8,string) 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **gas** | 82370 gas |
| **transaction cost** | 71626 gas |
| **execution cost** | 49858 gas |
| **input** | 0x2a6...00000 |
| **decoded input** | { "uint8 airlineType": 1, "string airlineName": "International" } |
| **decoded output** | { "0": "bool: true", "1": "string: !INFO! Airline has been setup." } |
| **logs** | [] |
| **val** | 0 wei |

transact to EagleTicket.registerCustomer pending ...

**console.log:**

Customer Registered: 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C

**[vm]**

**from:** 0x147...C160C

**to:** EagleTicket.registerCustomer(string) 0xd91...39138

**value:** 0 wei

**data:** 0x137...00000

**logs:** 0

**hash:** 0xe64...94dea

**Debug**

|  |  |
| --- | --- |
| **status** | true Transaction mined and execution succeed |
| **transaction hash** | 0xe649c2a0fc42270f77cc61f6c9b9261ccdb20695c5db624eb652a576d3d94dea |
| **block hash** | 0x1ba9a9b124be11beaf721ea3d3b038a35290039f03d018d0bea048a89eaccfd3 |
| **block number** | 4 |
| **from** | 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C |
| **to** | EagleTicket.registerCustomer(string) 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **gas** | 85383 gas |
| **transaction cost** | 74246 gas |
| **execution cost** | 52678 gas |
| **input** | 0x137...00000 |
| **decoded input** | { "string customerName": "MS Dhoni" } |
| **decoded output** | { "0": "bool: true" } |
| **logs** | [] |
| **val** | 0 wei |

transact to EagleTicket.registerCustomer pending ...

**console.log:**

Customer Registered: 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB

**[vm]**

**from:** 0x4B0...4D2dB

**to:** EagleTicket.registerCustomer(string) 0xd91...39138

**value:** 0 wei

**data:** 0x137...00000

**logs:** 0

**hash:** 0x09f...91f21

**Debug**

|  |  |
| --- | --- |
| **status** | true Transaction mined and execution succeed |
| **transaction hash** | 0x09f2b3daeedfb5e60b71b99459899197536b0fc2b2d43f0692eb10b32e391f21 |
| **block hash** | 0x116e9bccbba9213b514b1190a329b7aa825635c719c064900a7ca0482da048f9 |
| **block number** | 5 |
| **from** | 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB |
| **to** | EagleTicket.registerCustomer(string) 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **gas** | 85342 gas |
| **transaction cost** | 74210 gas |
| **execution cost** | 52678 gas |
| **input** | 0x137...00000 |
| **decoded input** | { "string customerName": "Kapil" } |
| **decoded output** | { "0": "bool: true" } |
| **logs** | [] |
| **val** | 0 wei |

transact to EagleTicket.registerCustomer pending ...

**console.log:**

Customer Registered: 0x583031D1113aD414F02576BD6afaBfb302140225

**[vm]**

**from:** 0x583...40225

**to:** EagleTicket.registerCustomer(string) 0xd91...39138

**value:** 0 wei

**data:** 0x137...00000

**logs:** 0

**hash:** 0x9e7...3b9b7

**Debug**

|  |  |
| --- | --- |
| **status** | true Transaction mined and execution succeed |
| **transaction hash** | 0x9e79227c752dbd04c3ff0aa3be22fe90feb0bbb6b531ac5a218d0767ab43b9b7 |
| **block hash** | 0xc980e9faf2128759c6d463d4799d099767abe29ed223c440e380e38e6500c9d0 |
| **block number** | 6 |
| **from** | 0x583031D1113aD414F02576BD6afaBfb302140225 |
| **to** | EagleTicket.registerCustomer(string) 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **gas** | 85356 gas |
| **transaction cost** | 74222 gas |
| **execution cost** | 52678 gas |
| **input** | 0x137...00000 |
| **decoded input** | { "string customerName": "Sachin" } |
| **decoded output** | { "0": "bool: true" } |
| **logs** | [] |
| **val** | 0 wei |

transact to EagleTicket.registerCustomer pending ...

**console.log:**

Customer Registered: 0xdD870fA1b7C4700F2BD7f44238821C26f7392148

**[vm]**

**from:** 0xdD8...92148

**to:** EagleTicket.registerCustomer(string) 0xd91...39138

**value:** 0 wei

**data:** 0x137...00000

**logs:** 0

**hash:** 0x1ba...c70ac

**Debug**

|  |  |
| --- | --- |
| **status** | true Transaction mined and execution succeed |
| **transaction hash** | 0x1ba458527349e21797919a6226dd45e336c7a227d90ecdc43b751ffd565c70ac |
| **block hash** | 0x6b7a5d5612c0f05a72e1befce3a23b6ec3f1734637a4fae4b8221853f38f4f8f |
| **block number** | 7 |
| **from** | 0xdD870fA1b7C4700F2BD7f44238821C26f7392148 |
| **to** | EagleTicket.registerCustomer(string) 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **gas** | 85342 gas |
| **transaction cost** | 74210 gas |
| **execution cost** | 52678 gas |
| **input** | 0x137...00000 |
| **decoded input** | { "string customerName": "Virat" } |
| **decoded output** | { "0": "bool: true" } |
| **logs** | [] |
| **val** | 0 wei |

transact to EagleTicket.setupFlight pending ...

**[vm]**

**from:** 0xAb8...35cb2

**to:** EagleTicket.setupFlight(uint256,string,uint256,string,string,uint256,uint256) 0xd91...39138

**value:** 0 wei

**data:** 0x3dc...00000

**logs:** 0

**hash:** 0xdeb...f60c0

**Debug**

|  |  |
| --- | --- |
| **status** | true Transaction mined and execution succeed |
| **transaction hash** | 0xdeb96286dd656e1d7858ddb6fc8be96b491a18579f8878ba7b054dfb4f0f60c0 |
| **block hash** | 0x91c7c736cd60952368e49d5ff8f90d7899ada86c40e9e32494fc9d507930ede5 |
| **block number** | 8 |
| **from** | 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 |
| **to** | EagleTicket.setupFlight(uint256,string,uint256,string,string,uint256,uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **gas** | 344684 gas |
| **transaction cost** | 299725 gas |
| **execution cost** | 276661 gas |
| **input** | 0x3dc...00000 |
| **decoded input** | { "uint256 flightNumber": "100001", "string flightName": "AI101", "uint256 scheduledDatetime": "1695734824", "string flightOrigin": "BOM", "string flightDestination": "DEL", "uint256 seatingCapacity": "100", "uint256 fixedPrice": "19" } |
| **decoded output** | { "0": "bool: success true", "1": "string: message !INFO! Flight has been setup." } |
| **logs** | [] |
| **val** | 0 wei |

transact to EagleTicket.setupFlight pending ...

**[vm]**

**from:** 0x4B2...C02db

**to:** EagleTicket.setupFlight(uint256,string,uint256,string,string,uint256,uint256) 0xd91...39138

**value:** 0 wei

**data:** 0x3dc...00000

**logs:** 0

**hash:** 0x2d4...283b0

**Debug**

|  |  |
| --- | --- |
| **status** | true Transaction mined and execution succeed |
| **transaction hash** | 0x2d412632c7eee9d28a4730ff7c6114566a6a158c6504d2af7c975aa7ccc283b0 |
| **block hash** | 0xc380154d110c1119162cb1c4b565613c6ea54892bc26a0b736d2785f35d65d8a |
| **block number** | 9 |
| **from** | 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db |
| **to** | EagleTicket.setupFlight(uint256,string,uint256,string,string,uint256,uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **gas** | 344684 gas |
| **transaction cost** | 299725 gas |
| **execution cost** | 276661 gas |
| **input** | 0x3dc...00000 |
| **decoded input** | { "uint256 flightNumber": "200001", "string flightName": "DL240", "uint256 scheduledDatetime": "1695781624", "string flightOrigin": "NYC", "string flightDestination": "AMS", "uint256 seatingCapacity": "100", "uint256 fixedPrice": "129" } |
| **decoded output** | { "0": "bool: success true", "1": "string: message !INFO! Flight has been setup." } |
| **logs** | [] |
| **val** | 0 wei |

call to EagleTicket.checkFlightStatus

***CALL*[call]**

**from:** 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db

**to:** EagleTicket.checkFlightStatus(uint256)

**data:** 0xe9e...186a1

**Debug**

|  |  |
| --- | --- |
| **from** | 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db |
| **to** | EagleTicket.checkFlightStatus(uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **execution cost** | 32331 gas (Cost only applies when called by a contract) |
| **input** | 0xe9e...186a1 |
| **decoded input** | { "uint256 flightNumber": "100001" } |
| **decoded output** | { "0": "bool: found true", "1": "string: message !INFO! Flight is scheduled." } |
| **logs** | [] |

transact to EagleTicket.buyticket pending ...

**[vm]**

**from:** 0x147...C160C

**to:** EagleTicket.buyticket(uint256) 0xd91...39138

**value:** 20000000000000000000 wei

**data:** 0x143...186a1

**logs:** 1

**hash:** 0x391...1591a

**Debug**

|  |  |
| --- | --- |
| **status** | true Transaction mined and execution succeed |
| **transaction hash** | 0x391a1dc78da1f9f09bb3504b853b1df2553a0eebef215a8c9c3897f17d21591a |
| **block hash** | 0xc491b4ecdd30d459cadec4e69b5b5d6413c9dd82306a0fb055f6baa1b3ac7928 |
| **block number** | 10 |
| **from** | 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C |
| **to** | EagleTicket.buyticket(uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **gas** | 270262 gas |
| **transaction cost** | 235010 gas |
| **execution cost** | 213782 gas |
| **input** | 0x143...186a1 |
| **decoded input** | { "uint256 flightNumber": "100001" } |
| **decoded output** | { "0": "bool: success true", "1": "string: !INFO! Ticket Purchased." } |
| **logs** | [ { "from": "0xd9145CCE52D386f254917e481eB44e9943F39138", "topic": "0x8259ee41bbd011f94af5636d12626cbb8f7a43757f629a201ba17a11fa8aad5e", "event": "TicketReserved", "args": { "0": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2", "1": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C", "2": "100001", "3": "1000000000001", "4": "1000000000000000000", "5": "Ticket Purchased!", "airline": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2", "customer": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C", "flightNumber": "100001", "ticketNumber": "1000000000001", "transferredAmount": "1000000000000000000", "message": "Ticket Purchased!" } } ] |
| **val** | 20000000000000000000 wei |

call to EagleTicket.checkTicketstatus

***CALL*[call]**

**from:** 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C

**to:** EagleTicket.checkTicketstatus(uint256)

**data:** 0x8a2...51001

**Debug**

|  |  |
| --- | --- |
| **from** | 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C |
| **to** | EagleTicket.checkTicketstatus(uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138 |
| **execution cost** | 25088 gas (Cost only applies when called by a contract) |
| **input** | 0x8a2...51001 |
| **decoded input** | { "uint256 ticketNumber": "1000000000001" } |
| **decoded output** | { "0": "bool: success true", "1": "string: message !INFO! Ticket is Reserved." } |
| **logs** | [] |