Contents

[Chapter 1 Team Information 3](#_Toc147641884)

[Batch 3](#_Toc147641885)

[Members 3](#_Toc147641886)

[Github Repository 3](#_Toc147641887)

[Chapter 2 Project Goals 4](#_Toc147641888)

[Blockchain 4](#_Toc147641889)

[Smart Contract 4](#_Toc147641890)

[Basic 4](#_Toc147641891)

[Advanced 4](#_Toc147641892)

[Chapter 3 Assumptions 5](#_Toc147641893)

[Chapter 4 Deliverables 6](#_Toc147641894)

[Chapter 5 Eaglepoa network Layout 7](#_Toc147641895)

[Chapter 6 AWS Infrastructure and Configuration 8](#_Toc147641896)

[Security Group Configuration 9](#_Toc147641897)

[Chapter 7 Blockchain node setup 10](#_Toc147641898)

[List of Commands used to set up the nodes and accounts. 10](#_Toc147641899)

[Gensis File (eaglepoa.json) 11](#_Toc147641900)

[Nodes & Accounts 12](#_Toc147641901)

[Add as Peers 14](#_Toc147641902)

[Connectivity Check 15](#_Toc147641903)

[MetaMask Configuration 16](#_Toc147641904)

[Chapter 8 Solidity contract Setup 17](#_Toc147641905)

[Solidity code of the contract(s) 17](#_Toc147641906)

[Contract Architecture: Contract Structure 18](#_Toc147641907)

[Overall Contract structure: 18](#_Toc147641908)

[Contract – EagleFactory 19](#_Toc147641909)

[Contract – EagleAirline 20](#_Toc147641910)

[Contract - EagleTicket 21](#_Toc147641911)

[Contract Architecture: High-level Sequence diagrams 22](#_Toc147641912)

[Airline Functions 22](#_Toc147641913)

[Flight Operator Functions 23](#_Toc147641914)

[Customer Functions 24](#_Toc147641915)

[EC2 Execution Screenshots 25](#_Toc147641916)

[EC2 Node startup – Eagel1 25](#_Toc147641917)

[EC2 Node startup – Eagel2 26](#_Toc147641918)

[EC2 Node startup – Eagel3 27](#_Toc147641919)

[EC2 Node Peers 28](#_Toc147641920)

[MetaMask Integration Screenshots 29](#_Toc147641921)

[MetaMask: EaglePoA Network connection setup 29](#_Toc147641922)

[MetaMask: EaglePoA Account setup 30](#_Toc147641923)

[Remix – MetaMask Integration Screenshots 31](#_Toc147641924)

[Injected Provider – MetaMask setup 31](#_Toc147641925)

[Remix – Deploy & Run Transactions 32](#_Toc147641926)

[Deploy EagleFactory 32](#_Toc147641927)

[Deploy EagleAirline 35](#_Toc147641928)

[Contract Execution 39](#_Toc147641929)

[Contract Execution - Cancellation 52](#_Toc147641930)

# 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

# Solidity contract Setup

## Solidity Source Code – Github links

* **<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>

## Contract Architecture: Contract Structure

### Overall Contract structure:

A diagram of a computer program

Description automatically generated with medium confidence

### Contract – EagleFactory

A screen shot of a computer

Description automatically generated

### Contract – EagleAirline

A screenshot of a computer

Description automatically generated

### Contract - EagleTicket

A screenshot of a computer

Description automatically generated

## Contract Architecture: High-level Sequence diagrams

### Airline Functions

A diagram of a company

Description automatically generated

### Flight Operator Functions

A diagram of flight operations

Description automatically generated

### Customer Functions

A diagram of a company

Description automatically generated with medium confidence

## EC2 Execution Screenshots

### EC2 Node startup – Eagel1

A screenshot of a computer program

Description automatically generated

### EC2 Node Peers

A screenshot of a computer

Description automatically generated

## MetaMask Integration Screenshots

### MetaMask: EaglePoA Network connection setup

A screenshot of a computer

Description automatically generated

### MetaMask: EaglePoA Account setup

Import Airline & Customer Accounts into Metamask

A screenshot of a computer

Description automatically generated A screenshot of a black phone

Description automatically generated A screenshot of a phone

Description automatically generated

## Remix – MetaMask Integration Screenshots

### Injected Provider – MetaMask setup

A screenshot of a computer

Description automatically generated

## Remix – Deploy & Run Transactions

### Deploy EagleFactory

#### Confirm Contract Deployment

A screenshot of a computer program

Description automatically generated

#### geth Contract Creation

A screen shot of a computer

Description automatically generated

#### Deployed Contract

A computer screen shot of a program

Description automatically generated

### Deploy EagleAirline

#### Confirm Contract Deployment

A screenshot of a computer program

Description automatically generated

#### Geth Contract Creation

A screen shot of a computer

Description automatically generated

#### Deployed Contract

A screenshot of a computer

Description automatically generated

#### MetaMask Contract Deployed confirmation.

A screenshot of a computer

Description automatically generated

### Contract Execution

#### Operator: EagleAirline.registerDomesticOperator

A computer screen with text

Description automatically generated

#### Operator: EagleAirline.setupFlight

A screenshot of a computer

Description automatically generated

#### Operator: EagleAirline.setupFlight

A computer screen with white text

Description automatically generated

#### Customer: EagleAirline.registerCustomer

A computer screen shot of text

Description automatically generated

#### Customer: EagleAirline.checkflightStatus

A screenshot of a computer

Description automatically generated

#### Customer: EagleAirline.reserveTicket

A screenshot of a computer

Description automatically generated

#### Remix: Load EagleTicket “At Address”

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

#### Customer: EagleTicket.viewTicketInfo (Reserved Ticket)

A screenshot of a computer

Description automatically generated

#### Customer: EagleTicket.confirmTicket

A screenshot of a computer

Description automatically generated

#### Customer: EagleTicket.selectSeat

A screenshot of a computer

Description automatically generated

#### Customer: EagleTicket.viewTicketInfo (Confirmed Ticket)

A screenshot of a computer

Description automatically generated

#### Operator: EagleAirline.flightBoarding

A screenshot of a computer

Description automatically generated

#### Operator: EagleAirline.flightInAir

A screenshot of a computer

Description automatically generated

#### Operator: EagleAirline.flightLanded

A screenshot of a computer

Description automatically generated

### Contract Execution - Cancellation

#### Customer: EagleTicket.cancelTicket (80% Refund)

A screenshot of a computer

Description automatically generated

#### Operator: EagleAirline.flightCancelled

**[vm]**

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

**to:** EagleAirline.flightCancelled(uint256) 0xf8e...9fBe8

**value:** 0 wei

**data:** 0xe8f...00065

**logs:** 3

**hash:** 0xc6b...363f1

**Debug**

|  |  |
| --- | --- |
| **status** | true Transaction mined and execution succeed |
| **transaction hash** | 0xc6b9d81a9fcd7e25ceb6ed9a2d926b88ad14c2869e7d5974df1b293a9ca363f1 |
| **block hash** | 0xe84761ffe6964d3a3956886dcb6bd0b645ed24fb02676f56c777687cab5f3838 |
| **block number** | 18 |
| **from** | 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 |
| **to** | EagleAirline.flightCancelled(uint256) 0xf8e81D47203A594245E36C48e151709F0C19fBe8 |
| **gas** | 773808 gas |
| **transaction cost** | 538476 gas |
| **execution cost** | 584472 gas |
| **input** | 0xe8f...00065 |
| **decoded input** | { "uint256 flightNum": "101" } |
| **decoded output** | { "0": "bool: true" } |
| **logs** | [ { "from": "0x5C9eb5D6a6C2c1B3EFc52255C0b356f116f6f66D", "topic": "0x9d768ac2a59d50fdc8755ef01b3016e47e3b8fabaa9114e9c37d09286373f789", "event": "TicketClosed", "args": { "0": "1000000000001", "1": "10000000000000000000", "2": "0", "3": "10000000000000000000", "4": "INFO: Ticket Closed", "ticketNumber": "1000000000001", "refundAmount": "10000000000000000000", "paidAmount": "0", "collectedAmount": "10000000000000000000", "message": "INFO: Ticket Closed" } }, { "from": "0xb8f43EC36718ecCb339B75B727736ba14F174d77", "topic": "0x9d768ac2a59d50fdc8755ef01b3016e47e3b8fabaa9114e9c37d09286373f789", "event": "TicketClosed", "args": { "0": "1000000000002", "1": "10000000000000000000", "2": "0", "3": "10000000000000000000", "4": "INFO: Ticket Closed", "ticketNumber": "1000000000002", "refundAmount": "10000000000000000000", "paidAmount": "0", "collectedAmount": "10000000000000000000", "message": "INFO: Ticket Closed" } }, { "from": "0xf8e81D47203A594245E36C48e151709F0C19fBe8", "topic": "0xe15e622369f6cd215131b21a6219d8216334c9c314c5293389f99fa06927e608", "event": "FlightUpdate", "args": { "0": "101", "1": "CANCELLED", "flightNum": "101", "uMsg": "CANCELLED" } } ] |
| **val** | 0 wei |