# Team Information

## Batch

* August 2022 – November 2022

## Members

* Anuradha
* Mohan
* Reema
* Sachin

# Project Goals

## Blockchain Goals

* Develop a Private Ethereum Blockchain implementation, using **geth** nodes running directly on a single **AWS EC2** (Ubuntu) server.
* 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 Goals

### 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 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.

# Private Blockchain Network Environment

## Accounts

<TODO: include details steps used for Node creation>

* 1 **Signer** Account
  + Signer (Sealer/etherbase) account
  + Declare & Pre-fund account in genesis “alloc” block
  + Use to provide initial funds to participating Airline & Customer (Bug token simulation) accounts.
* 1 **Airline Domestic** account
  + Eagle Airline – Domestic Account
  + Declare in genesis file with “0” funds
  + After creation transfer initial funds (‘Buy’ Ether in real-world) from **Signer** Account to this account.
* 1 **Airline International** account
  + Eagle Airline – International Account
  + Declare in genesis file with “0” funds
  + After creation transfer initial funds (‘Buy’ Ether in real-world) from **Signer** Account to this account.
* 4 **Customer** accounts:
  + Use MetaMask to create accounts.
  + Accounts will have “0” funds initially.
  + In MetaMask, connect to sealer node.

Network name: **eaglepoa**

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

Chain ID: **80801**

Currency symbol: **ARMS**

* + After creation transfer initial funds (‘Buy’ Ether in real-world) from **Signer** Account to this account.
* Total **Seven** accounts

## Nodes

<TODO: include layout diagram>

<TODO: include details steps used for Node creation>

* Network ID: 80801
* No Bootnode
* Use geth “--nodiscover” option when starting ALL nodes:
  + To disable peer discovery mechanism and allow us to manually add peers.
* **Eagle1** (Node-1):
  + “Full” node
  + Mining Enabled – Use **Signer** Account
  + Customer Accounts (MetaMask) will connect to this node.
  + geth command

geth --identity Eagle1 --networkid 80801 --datadir ./data --port 30301 --ipcdisable --syncmode "full" --http --allow-insecure-unlock --http.corsdomain "\*" --http.addr 0.0.0.0 --http.port 9001 --unlock "*[****Signer*** *Account]*" --password eagle1.txt --mine console --miner.etherbase  "*[****Signer*** *Account]*" --authrpc.port 9551 --nodiscover

* **Eagle2** (Node-2): Domestic
  + “Full” node
  + Mining disabled
  + Add Peer: **Eagle1**
  + geth command

geth --identity Eagle2 --networkid 80801 --datadir ./data --port 30302 --ipcdisable --syncmode "full" --http --allow-insecure-unlock --http.corsdomain "\*" --http.addr 0.0.0.0 --http.port 9002 --unlock "*[****Airline Domestic*** *Account]*" --password eagle2.txt console --authrpc.port 9552 --nodiscover

* **Eagle3** (Node-3): Domestic
  + “Full” node
  + Mining disabled
  + Add Peer: **Eagle1**
  + geth command

geth --identity Eagle3 --networkid 80801 --datadir ./data --port 30303 --ipcdisable --syncmode "full" --http --allow-insecure-unlock --http.corsdomain "\*" --http.addr 0.0.0.0 --http.port 9003 --unlock "*[****Airline International*** *Account]*" --password eagle3.txt console --authrpc.port 9553 --nodiscover

* Total **Three** nodes

## AWS – EC2 Instances

### Instance-1: EC2\_Eagle1

* Install Node: **Eagle1**
* Instance Type: t2.micro
* Platform: Ubuntu (Linux)
* Security – Inbound Rules (Required for MetaMask & Remix):

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

### Instance-2: EC2\_Eagle2

* Install Node: **Eagle2**
* Instance Type: t2.micro
* Platform: Ubuntu (Linux)

### Instance-3: EC2\_Eagle3

* Install Node: **Eagle3**
* Instance Type: t2.micro
* Platform: Ubuntu (Linux)

## Fungible Token Implementation (stretch goal)

* Smart Contract - <ARMSToken.sol>
* Implement ARMS Token using Create New Token ERC20 Interface specification.
* Consider 100 ARMS = 1 ETH
* Mint 1M ARMS tokens and deposit in Faucet Account
* Transfer 10000 ARMS tokens (Buy Tokens using Ether in real-world) to Customer and Airline Accounts

## Airline Ticket Smart Contract Components

### Actors (Account Types)

* Eagle Airlines
* Customers
* Escrow Agent (TBD: Smart Contract address / Signer Account)

### Data Structures

|  |  |
| --- | --- |
| **AirlineInfo** | **CustomerInfo** |
| * Airline Address * Airline Type (Domestic/International) * Airline Code (2-character) | * Customer Address * Customer Name |

|  |  |
| --- | --- |
| **Ticket** | **FlightInfo** |
| * Ticket Number (13-digit autogenerated) * Customer Address (buyer) * Flight Number * Seat Category * Seat Number * Ticket Price (Buying Price) * Refund Amount (Amount refunded to customer, id any) * Paid Amount (Amount paid to airline if any) * Ticket Status   + enum – Reserved, Cancellation-In-Progress, Cancelled * Payment Status   + enum – Escrowed, Paid-In-Full, Paid-In-Part (split between airline and customer due to late cancellation/delay penalty rules) | * Flight Number (Unique numeric ID) * Airline Address * Scheduled Departure Datetime (original planned EPOCH timestamp) * Adjusted Departure Datetime (latest delayed/rescheduled EPOCH timestamp; initialize to Scheduled Departure Datetime) * Actual Departure Datetime (EPOCH timestamp – updated with status = In-Air) * Flight Origin (Airport Code) * Flight Destination (Airport Code) * Flight Status   + enum – Scheduled, On-Time, Delayed, In-Air, Cancelled, Landed * Flight Seat Map (Seat Number => Ticket Number) * Ticket Seat Map (Ticket Number => Seat Number) |

### Functions

|  |  |
| --- | --- |
| **Customer Functions** | **Airline Functions** |
| * Buy Ticket * Cancel Ticket * Seat Selection * Claim Refund | * Set Flight * Update/Cancel Flight |

|  |  |
| --- | --- |
| **Other Functions** | **Internal/Private (Auto) Functions** |
| * Check Flight Status (anyone) * Refund Processing   + Only Airline OR Customer (Buyer) * View Ticket Details   + Only Airline OR Customer (Buyer) | * Auto Cancellation:   + When Update Flight to ‘Cancelled’   + When Cancel Ticket completes. * Seat Unblocking:   + When Cancel Ticket completes. |

### Cancellation / Delay Penalties

#### Cancellation by Customer

|  |  |
| --- | --- |
| **Rule: Based on DIFFERENCE of (Scheduled Departure Datetime – Cancellation Datetime)** | **Refund Amount**  **(% of Ticket Price)** |
| If >= 24 hours | 100% |
| If >= 4 hours and < 24 hours | 80% |
| If >= 2 hour and < 4 hours | 40% |
| If < 2 hour | 0% |

#### Cancellation by Airline - Refund Rules

* Anytime: 100% refund

#### Delayed by Airline – Penalty Rules

* Customer eligible to claim Refund 24 hours after the flight departure time.

|  |  |
| --- | --- |
| **Rule: Based on DIFFERENCE of (Scheduled Departure Datetime – Actual Departure Datetime)** | **Refund Amount**  **(% of Ticket Price)** |
| If > 2 hours and <= 10 hours | 10% |
| If > 10 hours and <= 24 hours | 40% |
| If > 24 hours | 100% |
| If Actual Departure Datetime is not updated within 24 hours from Scheduled Departure Datetime | 100% |

# Project Deliverables

### AWS EC2 Instance

<TODO: include AWS setup details/screenshots>

### geth – PoA network setup

<TODO: include geth PoA setup details/screenshots>

{

      "config": {

      "chainId": 80801,

      "homesteadBlock": 0,

      "eip150Block": 0,

      "eip155Block": 0,

      "eip158Block": 0,

      "byzantiumBlock": 0,

      "constantinopleBlock": 0,

      "petersburgBlock": 0,

      "istanbulBlock": 0,

      "berlinBlock": 0,

      "clique": {

            "period": 30,

            "epoch": 30000

      }

      },

      "difficulty": "1",

      "gasLimit": "8000000",

      "extradata": "0x0000000000000000000000000000000000000000000000000000000000000000954147177c89b1cC9f1cd5751352b798832CDAA50000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",

      "alloc": {

            "954147177c89b1cC9f1cd5751352b798832CDAA5": { "balance": "9000000000000000000000" },

            "51FC9578B9688a861d494ad309B825037C616Eb3": { "balance": "0" },

            "4352b4d7317810Bd0e1Cc71776c597f3978b0837": { "balance": "0" }

      }

}

### Smart Contract - Solidity

<TODO: include smart contract description/details/screenshots>