# Project Writeup

# Background

FlightSurety is a sample application project for Udacity's Blockchain course.
It allows user to purchase flight insurance from a registered airline and get paid at a rate of 1.5 if the flight is delay due to airline fault.

FlightSurety is a collaboration between multiple airline, in order to be qualified as one, an airline must get the invitation from registered airlines, and fund at least 10 ether.

# Component

It consists of 4 components

- Smart application contract for app logic and oracles code
- Smart data contract for data persistence
- Client application for interacting with the user and airline
- Server application for answering why the flight is delay

# Technical specification

## Version

Truffle v5.1.21 (core: 5.1.21)
Solidity - ^0.4.24 (solc-js)
Node v9.4.0
Web3.js v1.2.1

# Setup development environment

Code: https://github.com/ymlai87416/blockchain-nanodegree-project4

## Install

This repository contains Smart Contract code in Solidity (using Truffle), tests (also using Truffle), dApp scaffolding (using HTML, CSS and JS) and server app scaffolding.

To install, download or clone the repo, then:

```
npm install
truffle compile
```

## Develop Client

Ganache is required to click start the project.

To run Ganache

```
ganache-cli -m "candy maple cake sugar pudding cream honey rich smooth
crumble sweet treat" -a 50 -l 9999999 -q
```

To run truffle tests:

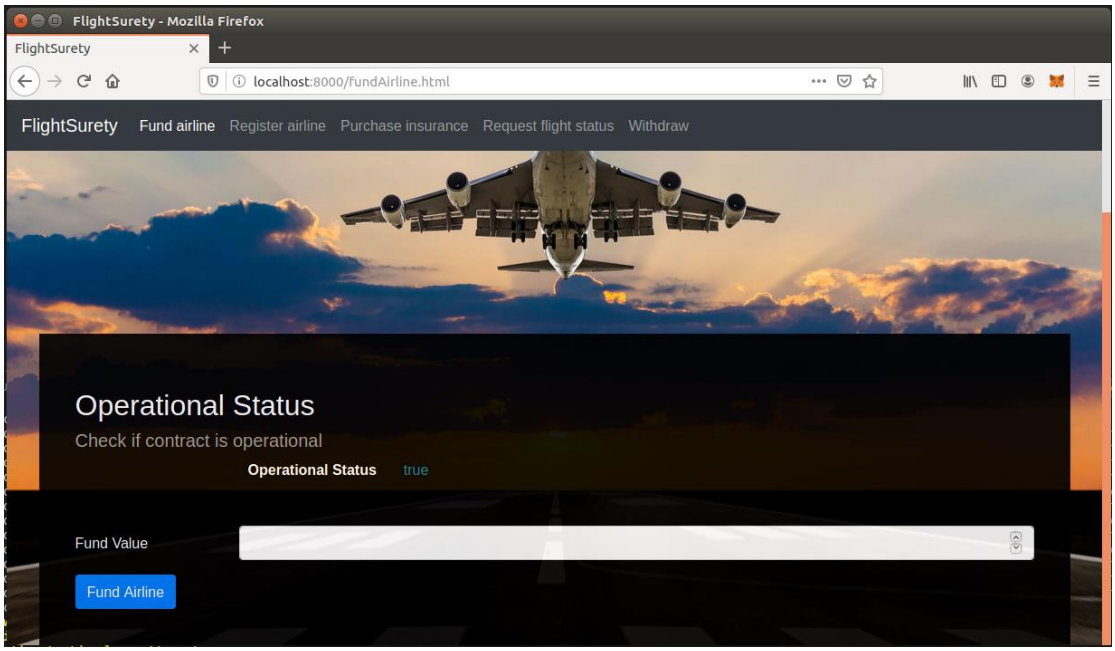
```
truffle test ./test/flightSurety.js
truffle test ./test/oracles.js
```

To use the dapp:

```
truffle migrate --reset --network development
npm run dapp
```

To view dapp:

```
http://localhost:8000
```




There are 5 pages, and each provides a function.

| Page | Function |
|---|---|
| Fund airline | Allow registered airline to provide funding |
| Register airline | Allow registered airline to invite another airline to join. |
| Purchase insurance | Allow passenger to buy insurance |
| Request flight status | Allow anyone to request flight status |
| Withdraw | Allow passenger or airline to get their ETH stored in the contract. |

## Server

```
npm run server
truffle test ./test/oracles.js
```

## Deploy

To build dapp for prod: `npm run dapp:prod`
Deploy the contents of the ./dapp folder

# Requirement

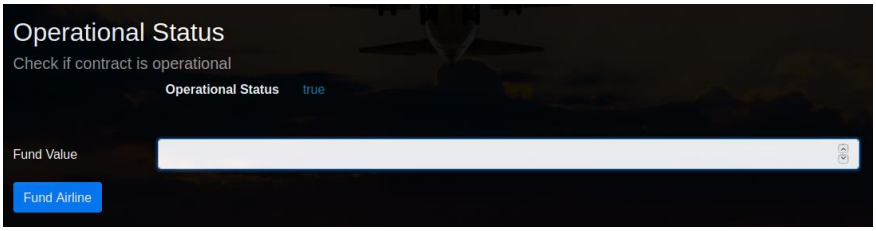
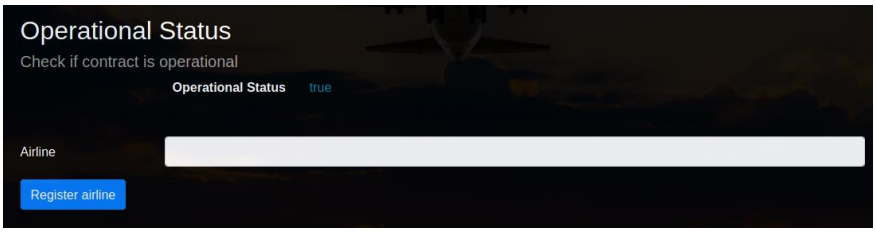
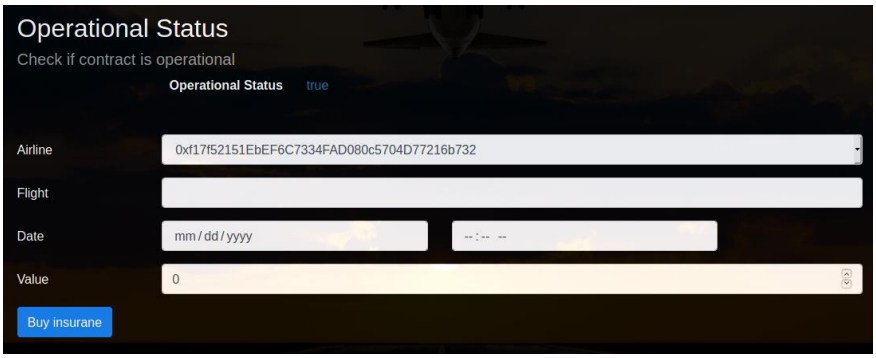
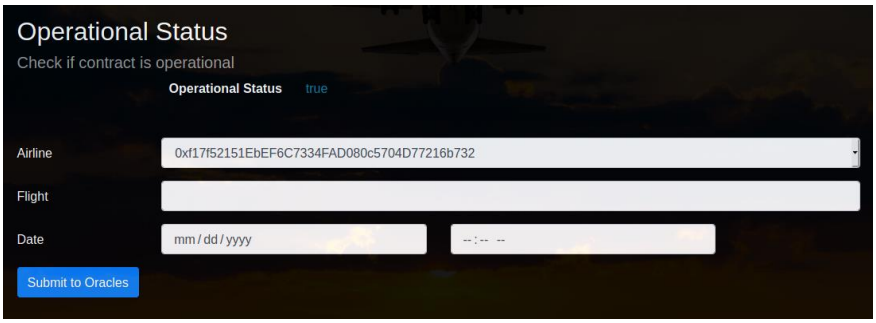
### 1 Separation of Concerns, Operational Control and "Fail Fast"

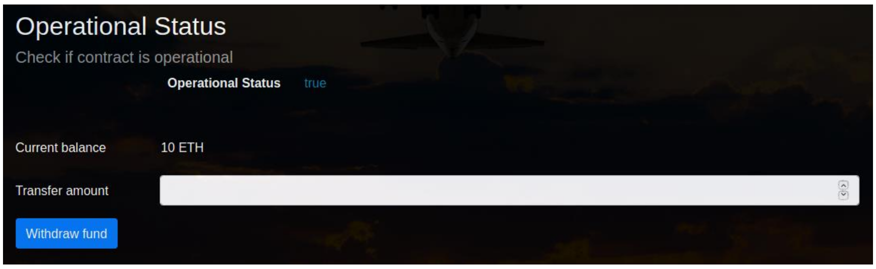
### 1.1 Smart contract separation:

1) FlightSuretyData.sol for data persistence
2) FlightSuretyApp.sol for app logic and oracles code

### 1.2 Dapp Created and Used for Contract Calls

The Dapp is capable to do the following:

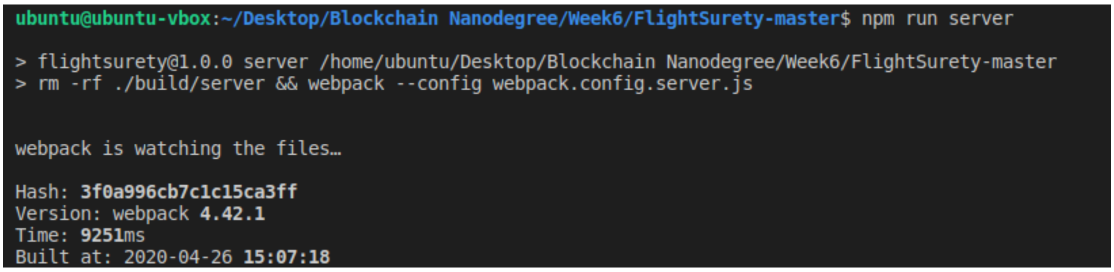
| | |
|---|---|
| Allow airline to provide funding | http://localhost:8000/fundAirline.html   |
| Allow airline to register another airline | http://localhost:8000/regAirline.html   |
| Allow passenger to purchase insurance | http://localhost:8000/buyInsurance.html   |
| Allow anyone to request flight status | http://localhost:8000/flightStatus.html   |
| Allow | http://localhost:8000/withdraw.html |

| | |
|---|---|
| passenger or airline to withdraw credit. |  <br>Operational Status<br>Check if contract is operational<br>Operational Status true<br>Current balance 10 ETH<br>Transfer amount<br>Withdraw fund |

## 1.3 Oracle Server Application

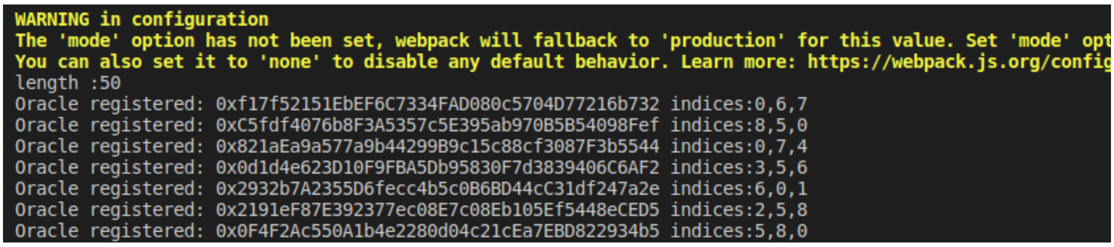
The oracle server is written to simulate a group of 30 oracles.

To start the server



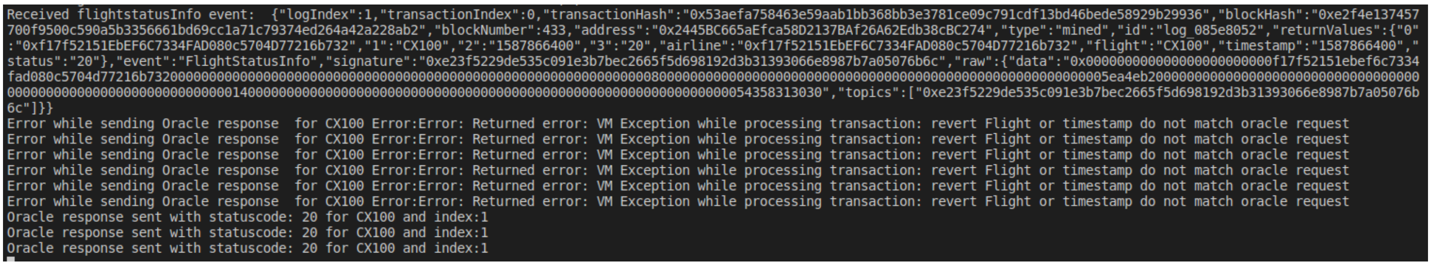
```
ubuntu@ubuntu-vbox:~/Desktop/Blockchain Nanodegree/Week6/FlightSurety-master$ npm run server

> flightsurety@1.0.0 server /home/ubuntu/Desktop/Blockchain Nanodegree/Week6/FlightSurety-master
> rm -rf ./build/server && webpack --config webpack.config.server.js


webpack is watching the files…

Hash: 3f0a996cb7c1c15ca3ff
Version: webpack 4.42.1
Time: 9251ms
Built at: 2020-04-26 15:07:18
```

Register oracle



```
WARNING in configuration
The 'mode' option has not been set, webpack will fallback to 'production' for this value. Set 'mode' opt
You can also set it to 'none' to disable any default behavior. Learn more: https://webpack.js.org/config
length :50
Oracle registered: 0xf17f52151EbEF6C7334FAD080c5704D77216b732 indices:0,6,7
Oracle registered: 0xC5fdf4076b8F3A5357c5E395ab970B5B54098Fef indices:8,5,0
Oracle registered: 0x821aEa9a577a9b44299B9c15c88cf3087F3b5544 indices:0,7,4
Oracle registered: 0x0d1d4e623D10F9FBA5Db95830F7d3839406C6AF2 indices:3,5,6
Oracle registered: 0x2932b7A2355D6fecc4b5c0B6BD44cC31df247a2e indices:6,0,1
Oracle registered: 0x2191eF87E392377ec08E7c08Eb105Ef5448eCED5 indices:2,5,8
Oracle registered: 0x0F4F2Ac550A1b4e2280d04c21cEa7EBD822934b5 indices:5,8,0
```

Response to fetch flight status request



```
Received flightstatusInfo event:  {"logIndex":1,"transactionIndex":0,"transactionHash":"0x53aefa758463e59aab1bb368bb3e3781ce09c791cdf13bd46bede58929b29936","blockHash":"0xe2f4e137457
700f9500c590a5b3356661bd69cc1a71c79374ed264a42a228ab2","blockNumber":433,"address":"0x2445BC665aEfca58D2137BAf26A62Edb38cBC274","type":"mined","id":"log_085e8052","returnValues":{"0"
:"0xf17f52151EbEF6C7334FAD080c5704D77216b732","1":"CX100","2":"1587866400","3":"20","airline":"0xf17f52151EbEF6C7334FAD080c5704D77216b732","flight":"CX100","timestamp":"1587866400","
status":"20"},"event":"FlightStatusInfo","signature":"0xe23f5229de535c091e3b7bec2665f5d698192d3b31393066e8987b7a05076b6c","raw":{"data":"0x000000000000000000000000f17f52151ebef6c7334
fad080c5704d77216b73200000000000000000000000000000000000000000000000000000000000000008000000000000000000000000000000000000000000000000000000005ea4eb20000000000000000000000000000000000
0000000000000000000000000000014000000000000000000000000000000000000000000000000000000000000000054358313030","topics":["0xe23f5229de535c091e3b7bec2665f5d698192d3b31393066e8987b7a05076b
6c"]}}
Error while sending Oracle response  for CX100 Error:Error: Returned error: VM Exception while processing transaction: revert Flight or timestamp do not match oracle request
Error while sending Oracle response  for CX100 Error:Error: Returned error: VM Exception while processing transaction: revert Flight or timestamp do not match oracle request
Error while sending Oracle response  for CX100 Error:Error: Returned error: VM Exception while processing transaction: revert Flight or timestamp do not match oracle request
Error while sending Oracle response  for CX100 Error:Error: Returned error: VM Exception while processing transaction: revert Flight or timestamp do not match oracle request
Error while sending Oracle response  for CX100 Error:Error: Returned error: VM Exception while processing transaction: revert Flight or timestamp do not match oracle request
Error while sending Oracle response  for CX100 Error:Error: Returned error: VM Exception while processing transaction: revert Flight or timestamp do not match oracle request
Oracle response sent with statuscode: 20 for CX100 and index:1
Oracle response sent with statuscode: 20 for CX100 and index:1
Oracle response sent with statuscode: 20 for CX100 and index:1
```

## 1.4 Operational status control is implemented in contracts

The operation status control is implemented to allow contract owner to stop people from buying insurance, new airline registration and withdraw credit.

## 1.5 Fail Fast Contract

Contract functions “fail fast” by having a majority of “require()” calls at the beginning of function body.

## 2 Airline

### 2.1 Airline Contract Initialization

The first airline is registered by contract owner when contract is deployed.

In file 2_deploy_contract.js line 26-28, first airline of address 0xf17f52151EbEF6C7334FAD080c5704D77216b732 is registered.

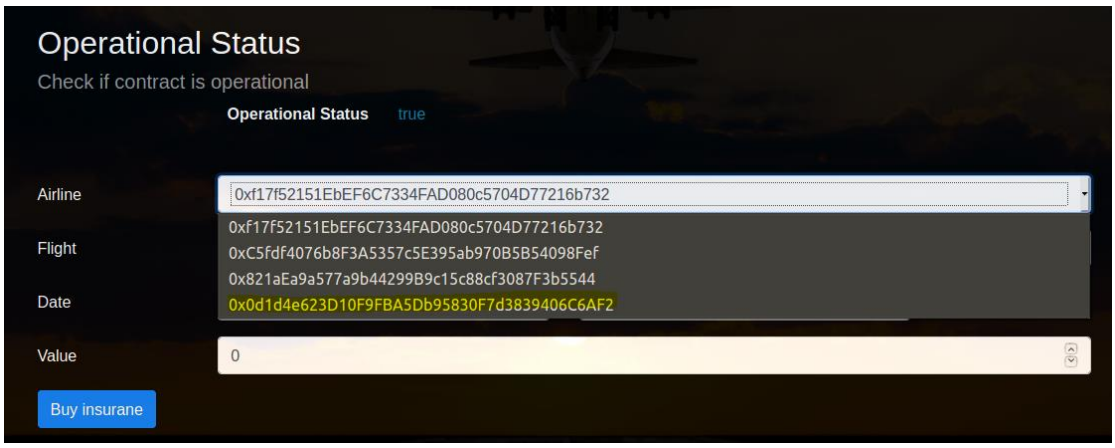
```
21
22                  //now call function to authorize app contract
23                  FlightSuretyData.deployed().then((dataContract) =>{
24                      dataContract.authorizeCaller(FlightSuretyApp.address).then(() => {
25                          //register the first airline
26                          FlightSuretyApp.deployed().then((appContract) =>{
27                              appContract.registerAirline(firstAirline);
28                          })
29                      })
30                  });
```

### 2.2 – 2.3 Multiparty Consensus

Only existing airline may register a new airline until there are at least 4 airlines registered

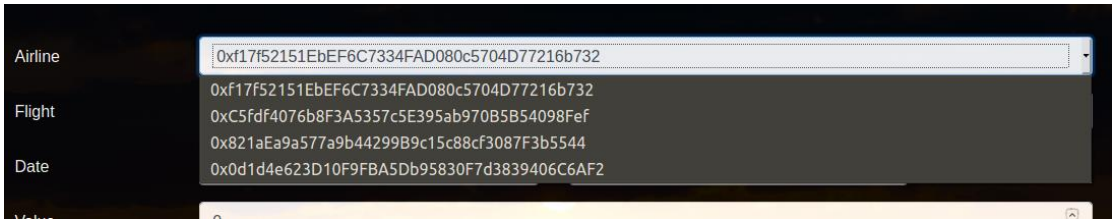
Registering the 4th airline 0x0d1d4e623D10F9FBA5Db95830F7d3839406C6AF2 by first airline.


To confirm the airline is registered, go to “Purchase insurance”

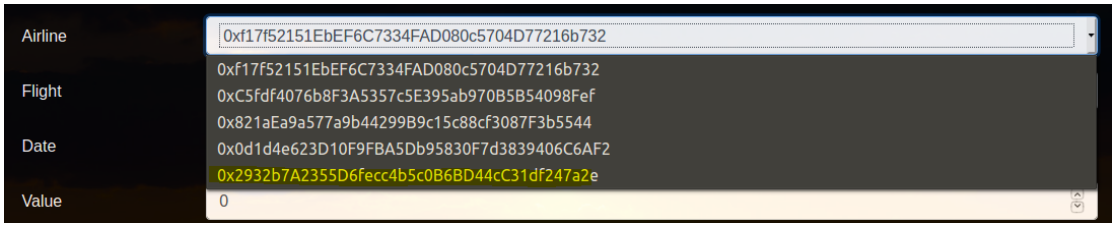



Registration of 5th and subsequent airlines requires multi-party consensus of 50% of registered airline


Even the 5th airline is registered by the 1st airline, it is not shown in the airline selection list.



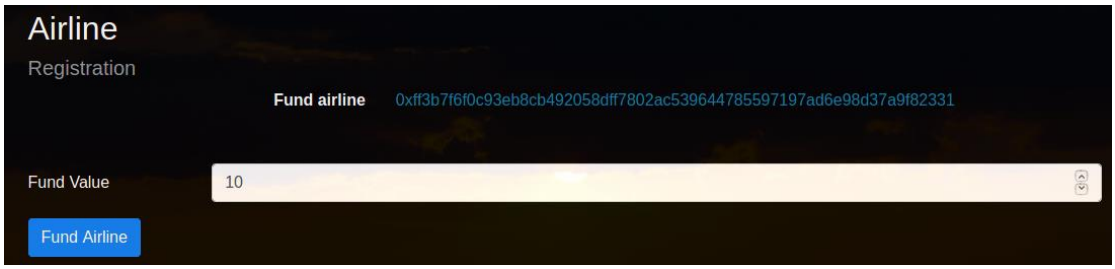

Later, 2nd airline registered airline 5th airline. Now it is available for selection in the "Purchase insurance" page.



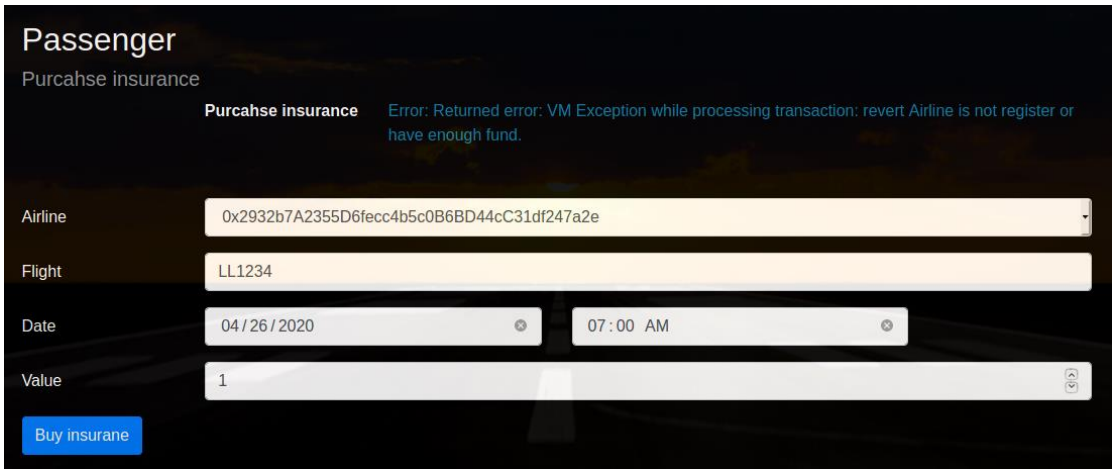

## 2.4 Airline Ante

Airline can be registered, but does not participate in the contract until it submits funding of 10 ether

Using “Fund airline” function, an airline can submit their funding.
Below show airline 0xf17f52151EbEF6C7334FAD080c5704D77216b732 submit 10 ether to the contract.



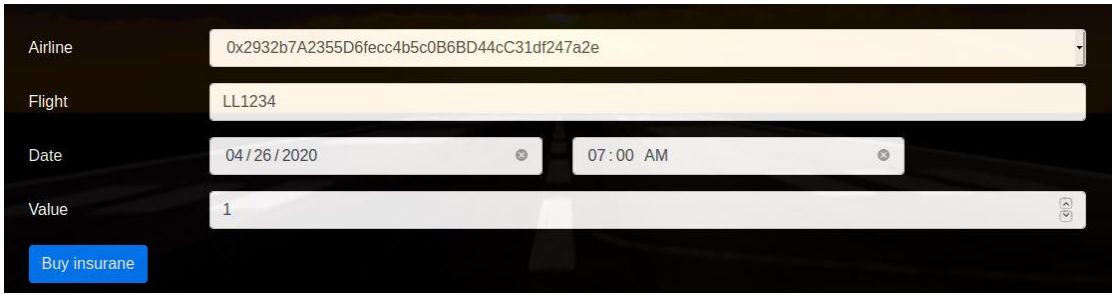

5th airline is registered, but has not yet submit any funding. When a passenger try to buy insurance for flight operating by 5th airline, system returns the following error.



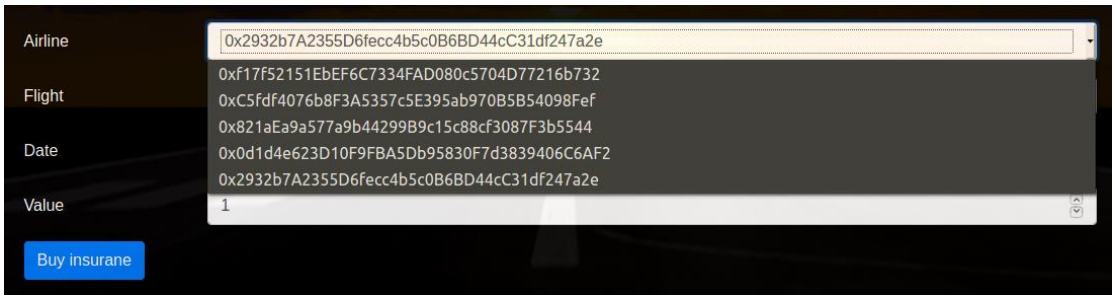

## 3 Passenger

### 3.1 Passenger Airline Choice

Passenger can go to the page “Purchase insurance”



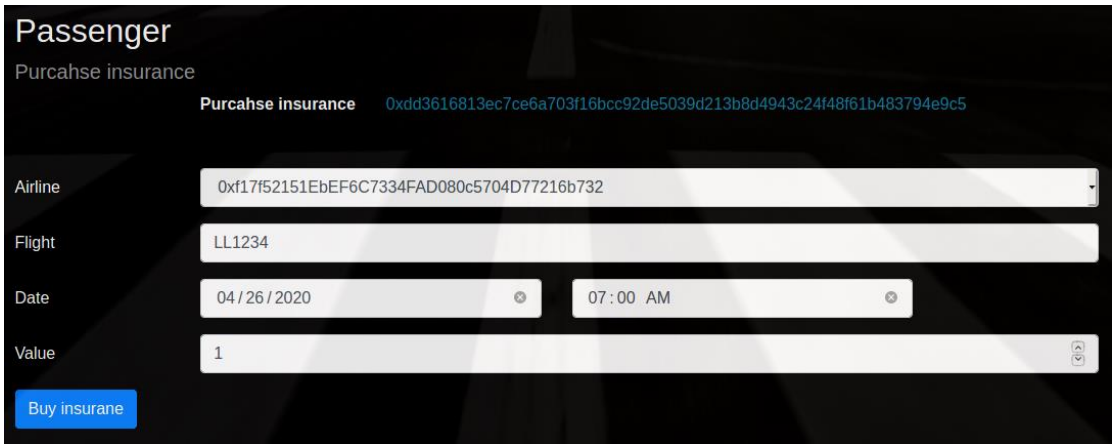
Passenger can choose from a list of airlines.




Passenger enters the flight information such as flight number and time of departure. He also decides how much money to put into this insurance project.

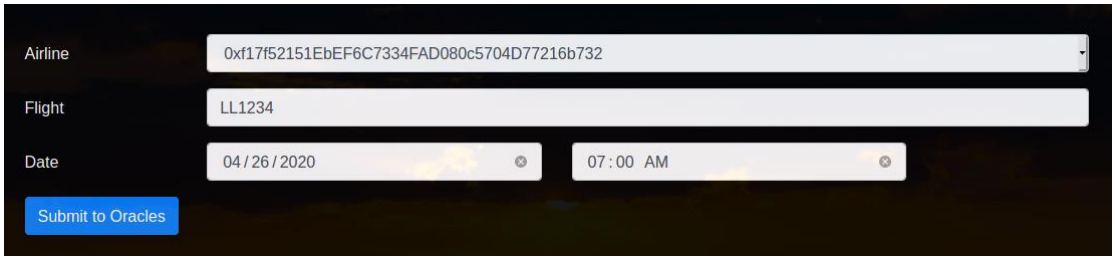
### 3.2 Passenger Payment

Passenger 0x2191eF87E392377ec08E7c08Eb105Ef5448eCED5 purchase an insurance worth 1 eth for the flight LL1234 departed at time 26th April 2020 7a.m. The transaction was successful.




### 3.3 Passenger Repayment

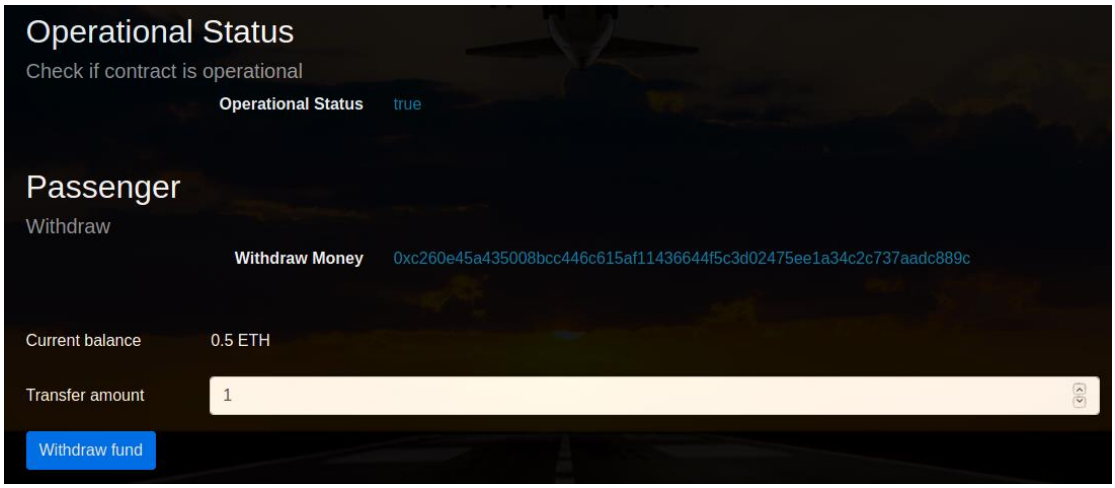
After the flight is landed, anyone can trigger "**Request flight status**" to allow the contract to get the flight status. If the flight is delayed and it is due to the airline fault, passenger receives credit of 1.5X the amount they paid




### 3.4 Passenger Withdraw

The oracles replied that the flight LL1234 at departed at time 26th April 2020 7a.m. is delayed due to airline fault. Now passenger 0x2191eF87E392377ec08E7c08Eb105Ef5448eCED5 can go to "**Withdraw**" page to check his credit balance. 1.5 ETH is now available for him to retrieve.


The passenger can enter the amount of ETH he wants to retrieve and click "Withdraw fund" and the remaining balance is updated to 0.5 ETH




### 3.5 Insurance Payouts

Insurance payouts are not sent directly to passenger's wallet

## 4 Oracle (Server App)

### 4.1 Functioning Oracle

Oracle functionality is implemented in the server app. The logic is at ROOT/src/server/server.js

### 4.2 Oracle Initialization

Upon startup, 20+ oracles are registered and their assigned indexes are persisted in memory

Upon start the server register 30 oracles. Below is the code fragment showing the server registers 30 oracles.

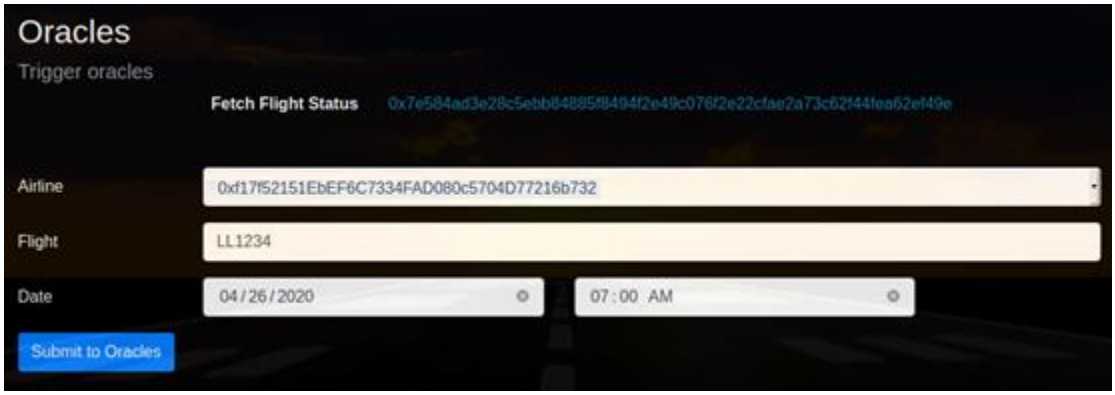
```
//initialize
web3.eth.getAccounts().then((accounts) => {
  console.log("length :"+ accounts.length);

  flightSuretyApp.methods.REGISTRATION_FEE().call().then(fee => {
    for(let a=1; a<ORACLES_COUNT; a++) {
      flightSuretyApp.methods.registerOracle()
      .send({ from: accounts[a], value: fee,gas:4000000 })
      .then(result=>{
        flightSuretyApp.methods.getMyIndexes().call({from: accounts[a]})
        .then(indices =>{
          oracles[accounts[a]] = indices;
          console.log("Oracle registered: " + accounts[a] + " indices:" + indices);
        })
      })
      .catch(error => {
        console.log("Error while registering oracles: " + accounts[a] +  " Error: " + error);
      });
    }
  })


});
```

When server starts, server log showing oracles are registered.

```
Oracle registered: 0xf17f52151EbEF6C7334FAD080c5704D77216b732 indices:2,3,8
Oracle registered: 0xC5fdf4076b8F3A5357c5E395ab970B5B54098Fef indices:5,1,2
Oracle registered: 0x821aEa9a577a9b44299B9c15c88cf3087F3b5544 indices:9,0,4
Oracle registered: 0x0d1d4e623D10F9FBA5Db95830F7d3839406C6AF2 indices:2,7,4
Oracle registered: 0x2932b7A2355D6fecc4b5c0B6BD44cC31df247a2e indices:5,4,7
Oracle registered: 0x2191eF87E392377ec08E7c08Eb105Ef5448eCED5 indices:8,4,5
Oracle registered: 0x0F4F2Ac550A1b4e2280d04c21cEa7EBD822934b5 indices:9,3,2
Oracle registered: 0x6330A553Fc93768F612722BB8c2eC78aC90B3bbc indices:7,0,1
Oracle registered: 0x5AEDA56215b167893e80B4fE645BA6d5Bab767DE indices:7,5,0
```

### 4.3 Oracle Update

Client Dapp can trigger a flight status request using page “**Request flight status**”.




Server receives the event, and submit oracle responses to the contract. The server log below shows the server is handling the **OracleRequest** event

```
Error while sending Oracle response  for LL1234 Error:Error: Returned error: VM Exception
while processing transaction: revert Index does not match oracle request
Error while sending Oracle response  for LL1234 Error:Error: Returned error: VM Exception
while processing transaction: revert Flight or timestamp do not match oracle request
Error while sending Oracle response  for LL1234 Error:Error: Returned error: VM Exception
while processing transaction: revert Flight or timestamp do not match oracle request
Error while sending Oracle response  for LL1234 Error:Error: Returned error: VM Exception
while processing transaction: revert Flight or timestamp do not match oracle request
Oracle response sent with statuscode: 20 for LL1234 and index:7
Oracle response sent with statuscode: 20 for LL1234 and index:7
```

The server log below shows that it gets the **FlightStatusInfo** event, and display it to the console. This event shows that Flight is a delayed due to airline fault.

Flight information:
Airline: `0xf17f52151EbEF6C7334FAD080c5704D77216b732`
Flight number: LL1234
Departure time: 1587855600 => Sunday, April 26, 2020 7:00:00 AM


```
Received flightstatusInfo event:
{"logIndex":1,"transactionIndex":0,"transactionHash":"0xde180da7e3661
08339682d4b04b5b597dfd7e2f374f2d6238e53c1e767861bcf","blockHash":"0xc
6907788d754bced57cec20853a8e0fdf60d7c60bd64c85d468c0382707ce4ff","blo
ckNumber":646,"address":"0xABa7902442c5739c6f0c182691d48D63d06A212E",
"type":"mined","id":"log_ac15cd9b","returnValues":{"0":"0xf17f52151Eb
EF6C7334FAD080c5704D77216b732","1":"LL1234","2":"1587855600","3":"20"
```

```
,"airline":"0xf17f52151EbEF6C7334FAD080c5704D77216b732","flight":"LL1
234","timestamp":"1587855600","status":"20"},"event":"FlightStatusInf
o","signature":"0xe23f5229de535c091e3b7bec2665f5d698192d3b31393066e89
87b7a05076b6c","raw":{"data":"0x000000000000000000000000f17f52151ebef
6c7334fad080c5704d77216b732000000000000000000000000000000000000000000
00000000000000000000800000000000000000000000000000000000000000000000
0000000005ea4c0f000000000000000000000000000000000000000000000000000
000000000140000000000000000000000000000000000000000000000000000000
00000064c4c31323334","topics":["0xe23f5229de535c091e3b7bec2665f5d6981
92d3b31393066e8987b7a05076b6c"]}}
```


### 4.4 Oracle Functionality

Server will loop through all registered oracles, identify those oracles for which the OracleRequest event applies, and respond by calling into FlightSuretyApp contract with random status code of Unknown (0), On Time (10) or Late Airline (20), Late Weather (30), Late Technical (40), or Late Other (50)

The code fragment in ROOT/src/server/server.js shows that

```
const index = event.returnValues.index;
const airline = event.returnValues.airline;
const flight = event.returnValues.flight;
const timestamp = event.returnValues.timestamp;
//let randomstatusCode = STATUSCODES[Math.floor(Math.random()*STATUSCODES.length)];
//console.log("statuscode is:" + randomstatusCode);
for(var key in oracles)
{
  var indexes = oracles[key];
  if(indexes.includes(index))
  {
    //let randomstatusCode = STATUS_CODE_LATE_AIRLINE;
    let randomstatusCode = STATUSCODES[Math.floor(Math.random()*STATUSCODES.length)];
    flightSuretyApp.methods.submitOracleResponse(index, airline, flight, timestamp, randomstatusCode)
    .send({ from: key,gas:1000000})
    .then(result =>{
      console.log("Oracle response sent with statuscode: "  + randomstatusCode + " for "+ flight + " and index:"+ index);
    })
    .catch(error =>{
      console.log("Error while sending Oracle response  for "+ flight + " Error:" + error)
    });

  }
}
```

# Resources

- [How does Ethereum work anyway?]
- [BIP39 Mnemonic Generator]
- [Truffle Framework]
- [Ganache Local Blockchain]

- Remix Solidity IDE
- Solidity Language Reference
- Ethereum Blockchain Explorer
- Web3Js Reference