

Blockchain & Solidity Whitepaper

Project: Flying Coins

Team: Group 2

Member:

Sofia Steinleitner,

Nimra Ayub,

Ton Thanh Huyen Ha



PROJECT NAME: FLYING COINS

The project Flying Coins is about a Flight Delay Insurance that should help customers to automatically get compensation for a delayed flight without the need of paperwork. The Insurance Company does not need to examine if a claim is valid. Therefore, the customers are able to receive the payment faster. This project should benefit both parties in saving time and efforts.

What is the problem?

Customers have to put a lot of work and effort to claim a compensation if their flight was delayed. This procedure is complicated and takes a lot of time. On the other hand the insurance company must check the validation of the claim as well. Customers might lose their right to claim for the payment if they do not apply for it on time.

What is the solution?

With an insurance as a smart contract the claim is secure. After taking out the flight delay policy both parties do not need to look after the fulfillment of the contract. The smart contract will automatically prove the conditions and trigger the payment.

How will it work?

The customers can usually buy the insurance with the booking of their flight or separately after the booking. With buying the insurance the smart contract gets activated automatically based on the data of the customer and the booked flight. After the activation, an involvement from both sides is not needed anymore. The smart contract is connected to an air traffic database so that it can get the flight information to check a delay. If a flight delay occurs the payment will be triggered by the smart contract. Otherwise the contract will be terminated. In both cases the customer gets notified about the status of the claim.



ELEMENTS OF THE APP

BackEnd:

Smart Contract: FlightDelay

This contract reads the information of the customers from the front-end and creates a new claim based on the given information. It enables the automatic transfer of the money.

Variables:

```
/**
 * Claim detail information
 */
struct Claim {
    string flightNr;
    uint256 amount;
    string start;
    string destination;
    string date;
    string time;
    address payable client;
}

// list of claims
Claim[] public claims;

// address of the insurance company
address public insurance_company;

// airport of the departure
string public start;

// airport of the destination
string public destination;

// date of the flight
string public date;

// time of the flight
string public time;

// ID of the flight
string public flightId;

// delay must be higher than 60 minutes
uint256 constant delay = 60;
```

Function:

```
/**
 * @dev init claim
 * @param owner who manages insurance
 * @param _flightId ID of the flight
 */
constructor(string memory _flightId, address owner) public {
    insurance_company = owner;
    flightId = _flightId;
}

// checks if msg.sender is the address of the insurance company
modifier companyOnly() {
    require(insurance_company == msg.sender);
    _;
}

/**
 * @dev Creates a claim, only the company is allowed to create claims
 * @param _flightId ID of the flight
 * @param _amount amount of the payout
 * @param _start airport of the departure
 * @param _destination airport of the destination
 * @param _date date of the flight
 * @param _time time of the flight
 * @param _client address of the client who receives the payout
 */
function createClaim(string memory _flightId, uint256 _amount, string memory _start, string memory _destination,
    string memory _date, string memory _time, address payable _client) public companyOnly() {
    Claim memory newClaim =
        Claim({
            flightNr: _flightId,
            amount: _amount,
            start: _start,
            destination: _destination,
            date: _date,
            time: _time,
            client: _client
        });
    claims.push(newClaim);
}
```

```
/**
 * @dev checks if a delay for a specific claim occurred and pays a fixed amount, only the company can pay the client
 * @param actualDelay actual delay of flight in minutes
 * @param claim_id id of the claim in the list
 */
function payout(uint256 actualDelay, uint256 claim_id) public payable companyOnly() {
    require(claim_id < claims.length, "FlightDelay: Index out-of-bound");
    Claim memory claim = claims[claim_id];

    if (actualDelay > delay) {
        claim.client.transfer(claim.amount);
    }
}
```

Smart Contract: InsuranceClient

This contract creates a new smart contract with the information from the contract FlightDelay.

Variables:

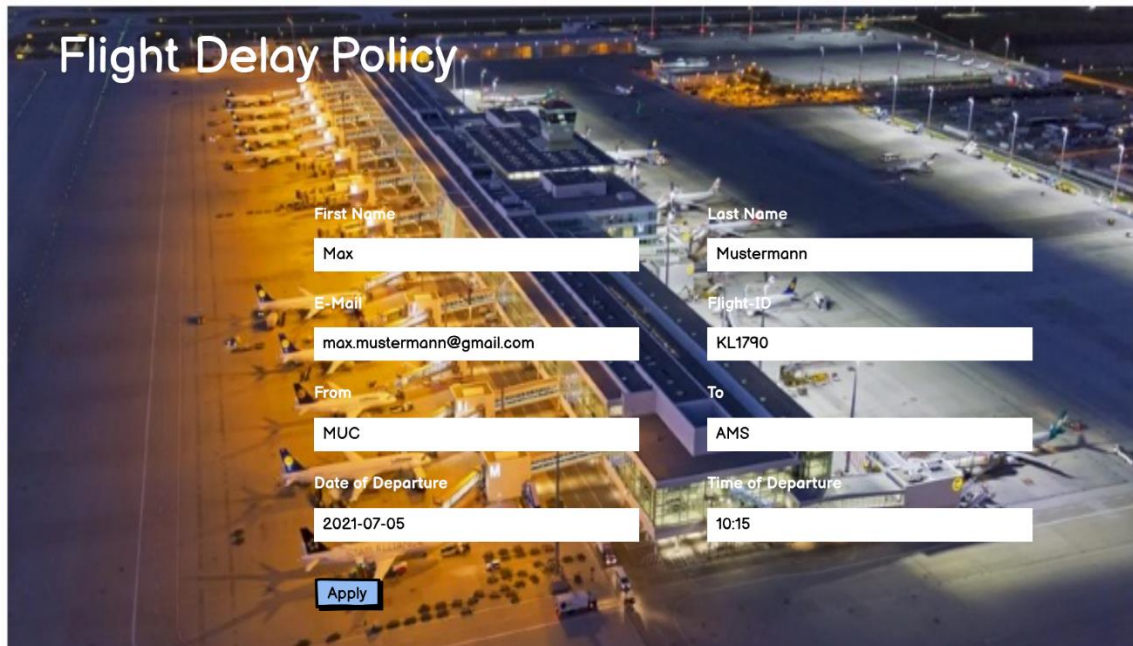
```
// list of clients  
address[] public clients;
```

Function:

```
/**  
 * @dev Creates new client with the given flight ID  
 * @param flightId ID of the flight  
 */  
function createClient(string memory flightId) public {  
    address newClient = address (new FlightDelay(flightId, msg.sender));  
    clients.push(newClient);  
}  
  
/**  
 * @dev get all deployed clients  
 */  
function getDeployedClients() public view returns (address[] memory) {  
    return clients;  
}
```


FrontEnd:

A mockup has been created instead of the implementation of the user interface. These mockups simulate how the insurance would have looked and worked.

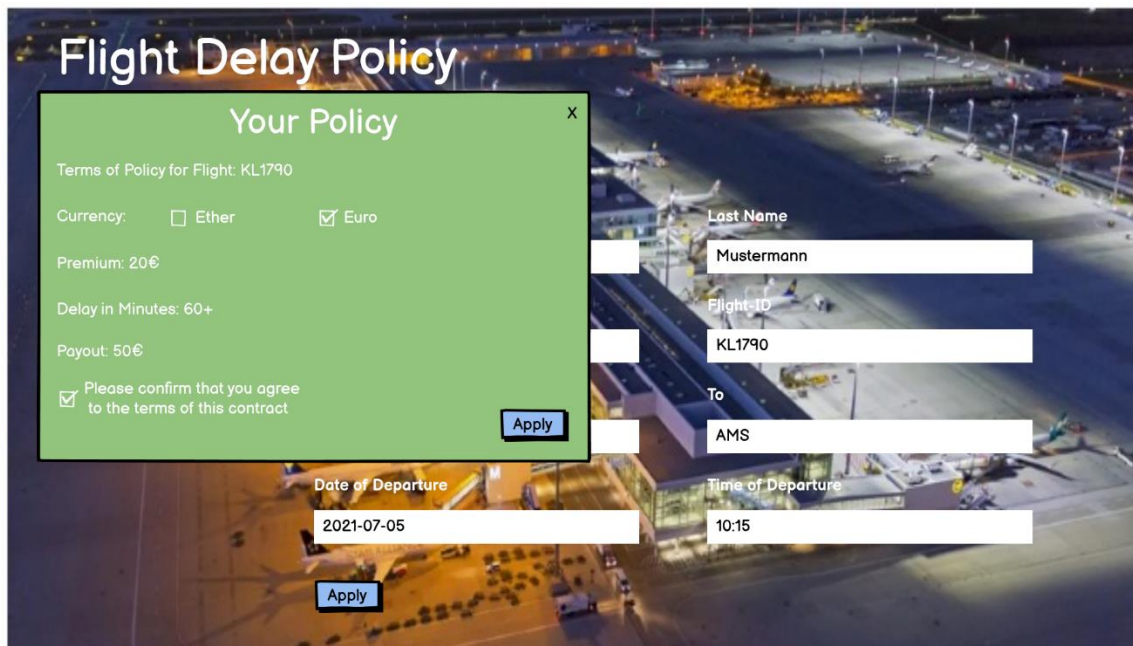


Flight Delay Policy

First Name	Last Name
Max	Mustermann
E-Mail	Flight-ID
max.mustermann@gmail.com	KL1790
From	To
MUC	AMS
Date of Departure	Time of Departure
2021-07-05	10:15

Apply

The insurance starts with entering the data of the customer as well as of the flight.



Flight Delay Policy

Your Policy

Terms of Policy for Flight: KL1790

Currency: ☐ Ether ☒ Euro

Premium: 20€

Delay in Minutes: 60+

Payout: 50€

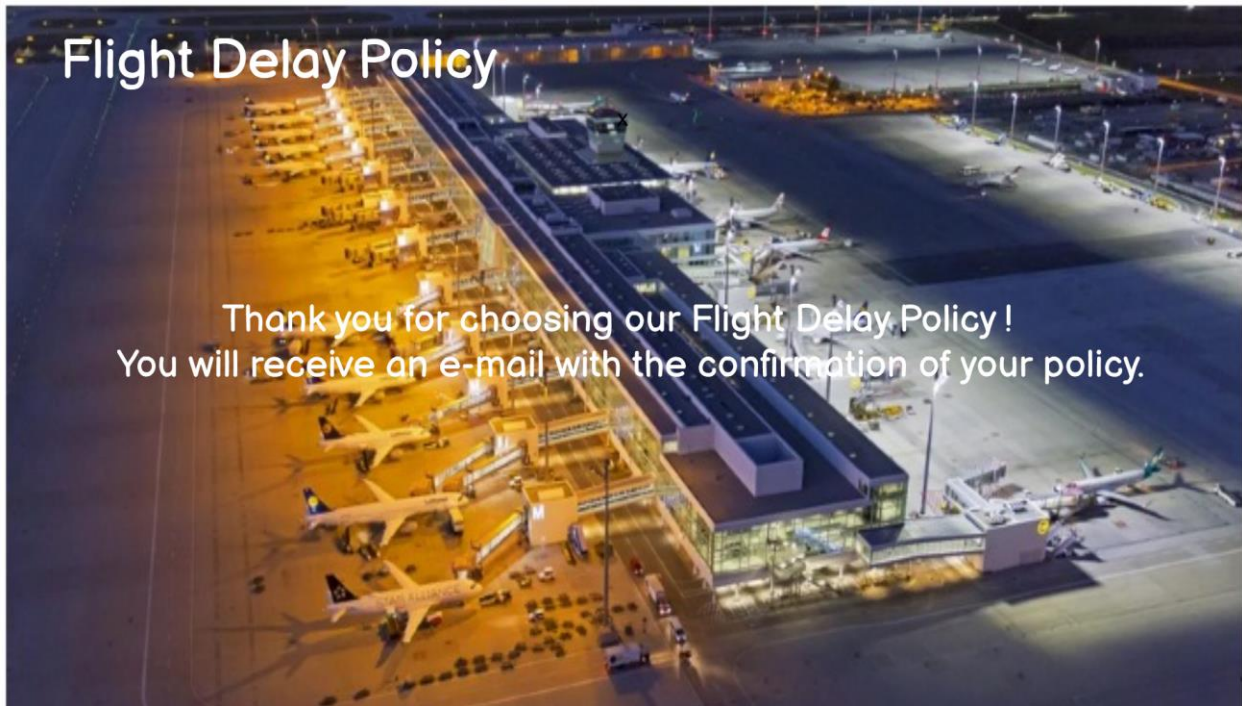
☒ Please confirm that you agree to the terms of this contract

Apply

Last Name
Mustermann
Flight-ID
KL1790
To
AMS
Date of Departure
2021-07-05
Time of Departure
10:15

Apply








After applying the information the customer has to confirm the terms of the policy. He is able to chose the currency of the payout. With confirming the policy the customer gives his agreement and activates the smart contract.



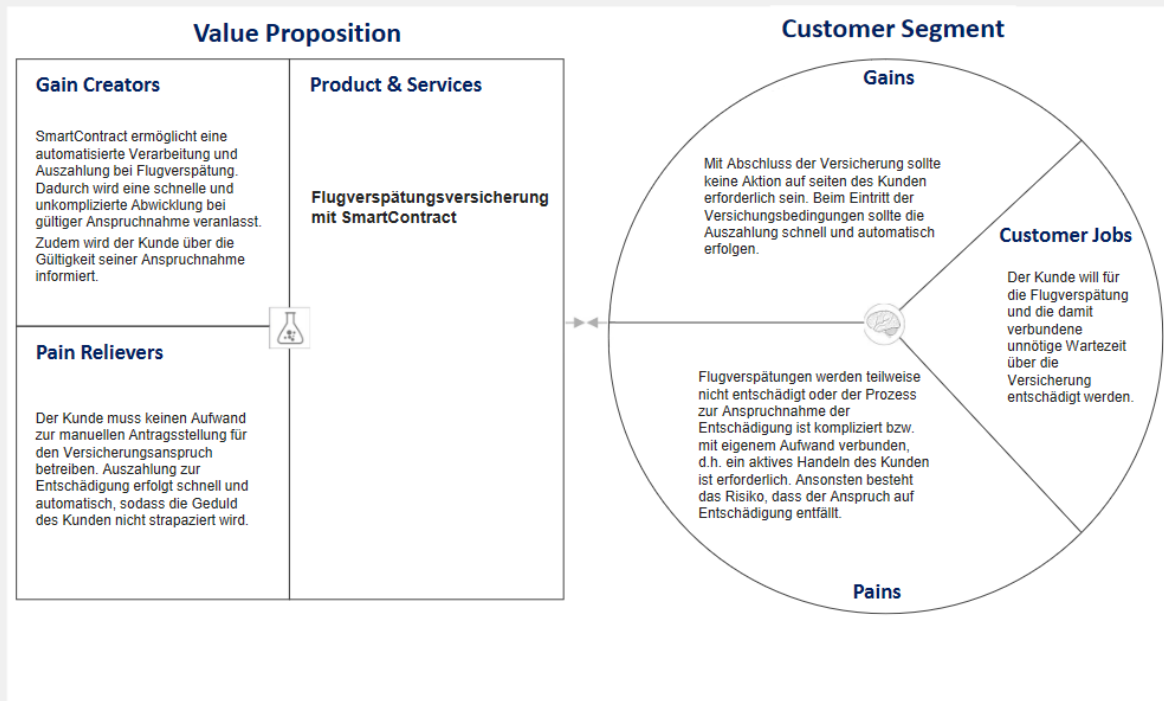
The customer will receive a summary of his chosen policy.

OPPORTUNITY & VALUE PROPOSITION

Projektname: Flugverspätungsversicherung als SmartContract

 Schlüsselpartner <ul style="list-style-type: none"> • Flughäfen • Reiseveranstalter • Fluggesellschaften 	 Schlüsselaktivitäten <ul style="list-style-type: none"> • Promoten durch Campaignen, Werbevideos, Social-Media-Posts, Anzeigen • Vermarktung der Versicherung auf Booking-Seiten und persönlicher Webseite 	 Wertangebot & Problemlösung <ul style="list-style-type: none"> • Automatisierte Verarbeitung und Auszahlung bei Flugverspätung • Minimierung von Verarbeitungs- und Verwaltungsaufwände • Reduzierung des Aufwands zur manuellen Einreichung des Versicherungsanspruches seitens des Kunden • Schnelle Abwicklung bei gültiger Anspruchsnahme 	 Kundenbeziehungen <ul style="list-style-type: none"> • Automatisierte und schnelle Schadensauszahlung • Vereinfachte Anspruchsnahme der Versicherung • Transparenter Informationsaustausch • Kundenservice 	 Kundensegmente in den Problemfeldern <ul style="list-style-type: none"> • Flugreisende
	 Schlüsselressourcen <ul style="list-style-type: none"> • Schnittstelle zur Flugverkehrsdatenbank • Kundendaten • Schnittstelle zu den Reiseveranstaltern • Schnittstelle zum Buchungssystem zur Auszahlung 		 Kanäle der Ideenverbreitung <ul style="list-style-type: none"> • Fluggesellschaften • Social Media • Reiseveranstalter 	
Kostenstrukturen: Aufteilung in Personal, Invest, Reisen			Verwertungspotenzial bzw. angestrebte Einnahmequellen	

Value Proposition Canvas



THE TEAM

- Stakeholder: comes up with the idea and the inputs of the project => description of the project
- Product Owner: plans and organizes the execution of the project => creation of the project model
- Programmer: executes the project => implementation of the idea

ANNEX

- Check that your source code all checked in your GitHub repository (BC-Group2)
- Provide the link here: <https://github.com/HM2021-BC/BC-GROUP2>
- Take care that a readme is uploaded, where all the needed information are provided which will be needed to run the application
- Connect the Jury Group to your team and Workspace on ML SEED

- Upload as well a small video with your show cases
- Create a pitch deck (template will be provided)
- Do run the multiple choice questionnaire in Moodle.

