CCT College Dublin

Assessment Cover Page

Module Title:	Distributed Digital Transactions
Assessment Title:	CA - Group Project: CabChain
Lecturer Name:	Dr. Muhammad Iqbal
Student Full Name:	Robert Szlufik, Ingrid Menezes Castro
Student Number:	2020358, 2020341
Assessment Due Date:	04/12/2022
Date of Submission:	04/12/2022
GITHUB link:	https://github.com/rpsbobby/Distributed_Digital_Transactions_CA2
Video Link:	https://drive.google.com/file/d/1qeIb0wBxZCEqQi3_4UeSrr2uygeKDZbc/view?usp=sharing

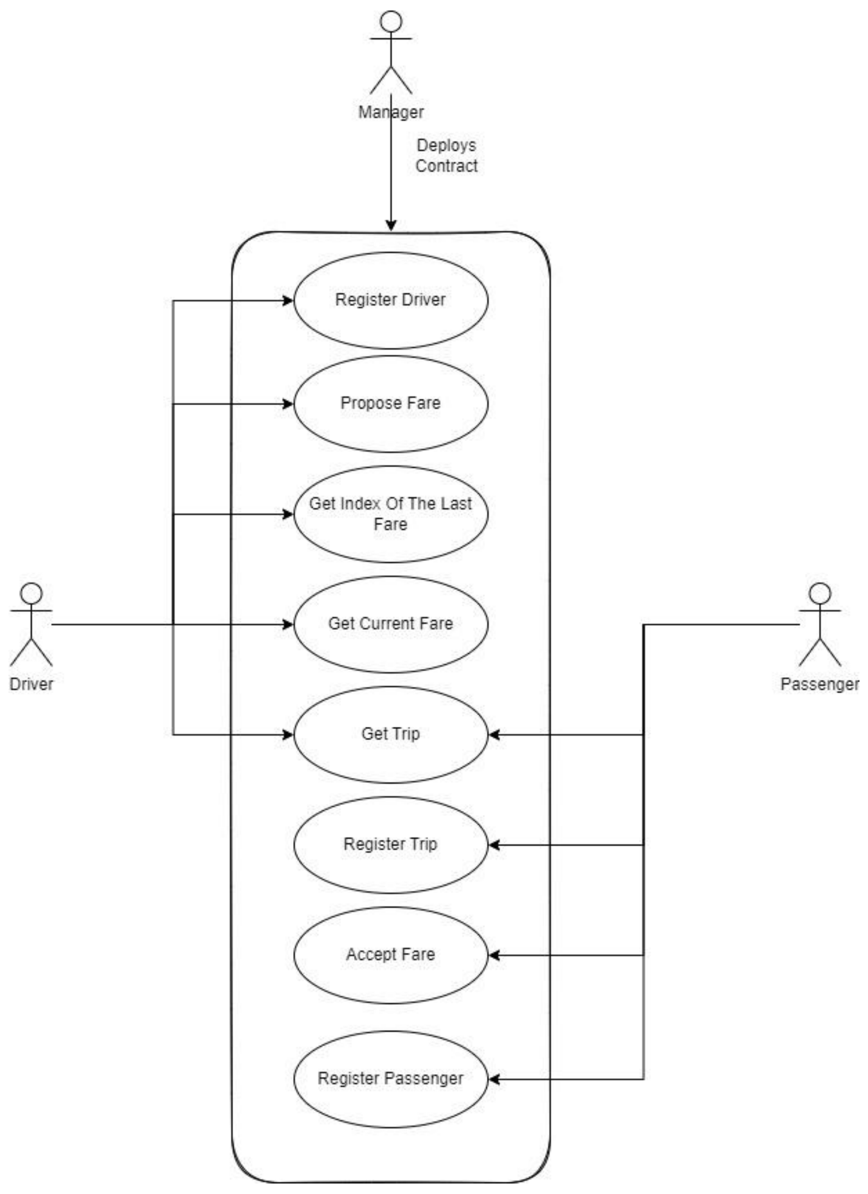
Declaration

Motivation, plan and design of the Contract

In this project we were asked to imagine a scenario where car rentals/ pooling companies would no longer operate, leaving customers and drivers with no means to contract each other. To resolve this problem we imagined and conceived CabChain, an Ethereum contract, through Solidity programming language. The motivation behind it was to enable passengers to contract drivers without the need of a third party, like Uber, Free Now etc. to intermediate the contract.

It was thought out in a way that would enable a driver to propose fares and get trips, and, on the other side, allow passengers to find a driver to take them to their destination.

As a group we divided the workload in two, and each member took care of one side: either the driver or the passenger. As explained in more depth on the Screenshots section, the passenger proposes a trip, checks the fare asked by the driver and chooses to accept the trip or not.



Use of Blockchain: advantages and disadvantages and Cryptographic methods

The use of Blockchain and Smart Contracts in some ways disrupted the old ways of establishing a contract between two parts. Etherium is a blockchain-based software that introduced the use of Smart Contracts and can send and receive value without the need of a third party to intermediate the transaction. There are many advantages and disadvantages to this new technology, which will be explored below.

Advantages of blockchain technology (Budhi, 2022)

- **Immutability** – once a block is added to the blockchain, it cannot be changed. Data recorded on blockchain will be distributed to all nodes on blockchain and it will be not possible to change it.

- **Transparency** – blockchains support transparency to the great degree. It is very easy to trace transactions recorded on blockchain all the way to the very block.

- **Non-existent censorship** - blockchain is a distributed technology that is managed by peers and all interactions happen peer-to-peer. There is no centralised control over the blockchain. It supports free-market concepts, and reciprocal validation by peers on the blockchain network.

The advantages of blockchain, in the context of CabChain, guarantee that peers' requirements are clearly stated, and cannot be influenced by other factors. Passengers and Drivers will not be constrained by 3rd parties which promotes free-market leverage, and ultimately, benefits users of application.

Disadvantages

- **Speed** – Due to the computational processing overhead (hashing), on different blockchain networks, it can take from 15 – 30 seconds to consent. It is significantly slower than traditional web 2.0 server-side services such as card payment, which usually take under a second to process.

- **Lack of data modification ability** – If we consider a case in which some company wishes to develop applications of high complexity. We could assume that there will be a certain number of mistakes, or changes in requirements, that were not accounted for at the time of developing application. Blockchains make it incredibly difficult to update requirements after technology is deployed.

In the context of CabChain, speed and lack of ability to change login begins application contract means that peers will experience longer time to process their interaction with blockchain than expected by most of the users. As users, we are used to applications that have instant responses. Unfortunately, in blockchain, this is not the case. It would take a certain amount of time to process every transaction. The fact that some requirements might change, or new ones could arise, would force the developer to start a new blockchain at every iteration of the application. This process will force data to be erased or stored in external data storage, which poses a big question over transparency and security.

Cryptographic methods used in blockchain technology

Cryptography is a core part of the blockchain technology, in fact, blockchain could not exist without it. Asymmetric cryptography (public-private key pair) and hashing are the two methods that blockchain utilises the most. In the blockchain realm, public key is an address of the user and private key is a secret that allows to access addresses data and authorise transactions. (Sahu, 2022)

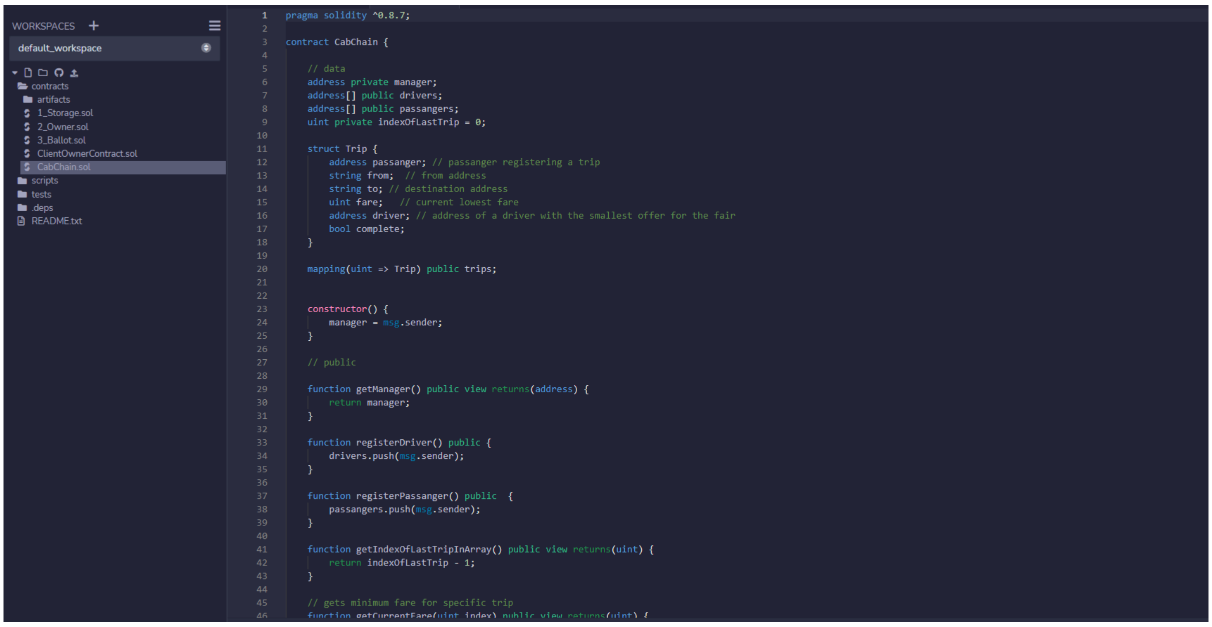
Most common hashing algorithm used in blockchain technology is the SHA-256 algorithm. SHA stands for Secure Hash Algorithm, which was developed by NAS (National Security Agency of USA) and published in 2001. SHA-256 produces 256 bit output of a given file.

In blockchain SHA-256 is used to hash transactions, among other details, such as proceeding block in blockchain to the value that suffices value of a nonce below a certain threshold. Then, it is signed by a private key of a sender, we call this a signature. After all the processing the block is added to the blockchain. This computational process guarantees integrity and immutability of each block and overall blockchain.

When it comes to Ethereum, according to its documentation the private key is encrypted by ECC (Elliptic Curve Cryptography) before the keccak-256 hash (Sinkevicius, 2022).

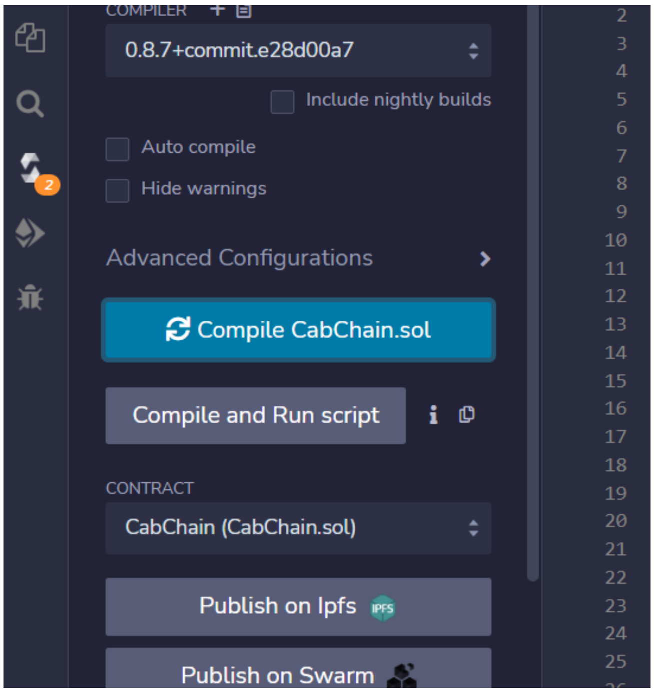
Screenshots

(1)
After finishing a contract in Remix Online IDE, Remix allows us to deploy to different test blockchains. Firstly, we need to compile the contract, we can achieve it by clicking on the compiler icon on the left hand side.



```
pragma solidity ^0.8.7;

contract CabChain {

    // data
    address private manager;
    address[] public drivers;
    address[] public passangers;
    uint private indexOfLastTrip = 0;

    struct Trip {
        address passanger; // passanger registering a trip
        string from;  // from address
        string to; // destination address
        uint fare;   // current lowest fare
        address driver; // address of a driver with the smallest offer for the fair
        bool complete;
    }

    mapping(uint => Trip) public trips;


    constructor() {
        manager = msg.sender;
    }

    // public

    function getManager() public view returns(address) {
        return manager;
    }

    function registerDriver() public {
        drivers.push(msg.sender);
    }

    function registerPassanger() public  {
        passangers.push(msg.sender);
    }

    function getIndexOfLastTripInArray() public view returns(uint) {
        return indexOfLastTrip - 1;
    }

    // gets minimum fare for specific trip
    function getCurrentFare(uint index) public view returns(uint) {
```

(2)
If the compiler doesn't throw an error, we can deploy to the blockchain of our choice. In our case, we will use ganache software which runs on host's local machine.

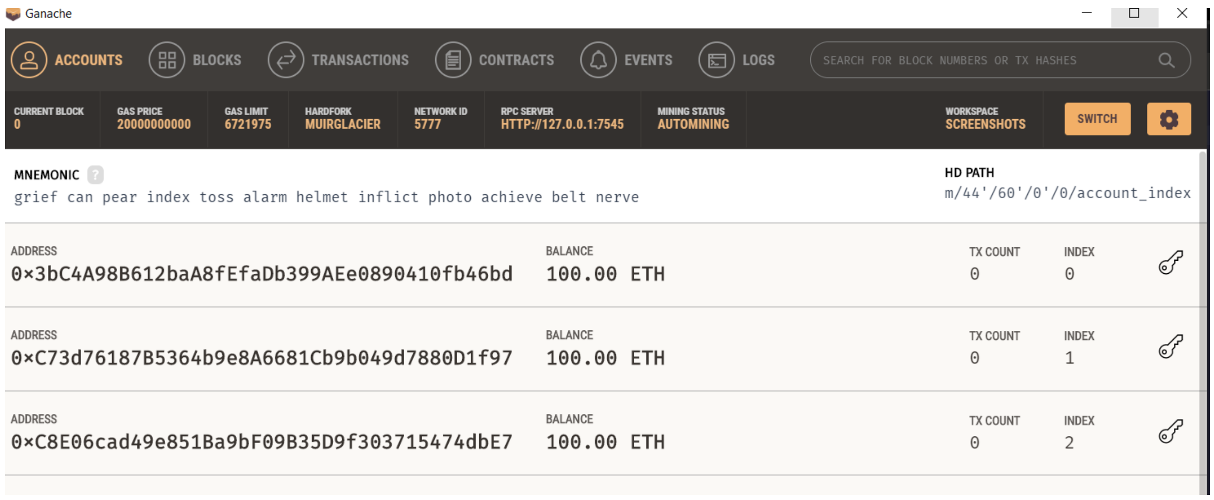


COMPILER: 0.8.7+commit.e28d00a7

CONTRACT: CabChain (CabChain.sol)

(3)

We can create a new workspace in Ganache and start a server. Ganache will provide us with 10 accounts that we can use to interact with and test our contract on the local machine.



CURRENT BLOCK 0 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MUIRGLACIER | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE SCREENSHOTS

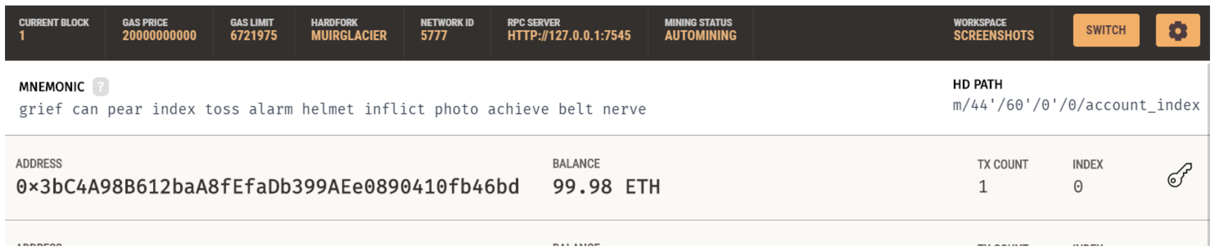
MNEMONIC
grief can pear index toss alarm helmet inflict photo achieve belt nerve

HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×3bC4A98B612baA8fEfaDb399AEe0890410fb46bd	100.00 ETH	0	0
0×C73d76187B5364b9e8A6681Cb9b049d7880D1f97	100.00 ETH	0	1
0×C8E06cad49e851Ba9bF09B35D9f303715474dbE7	100.00 ETH	0	2

(4)

Now we are ready to deploy our compiled contract to the test environment.



CURRENT BLOCK 1 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MUIRGLACIER | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE SCREENSHOTS

MNEMONIC
grief can pear index toss alarm helmet inflict photo achieve belt nerve

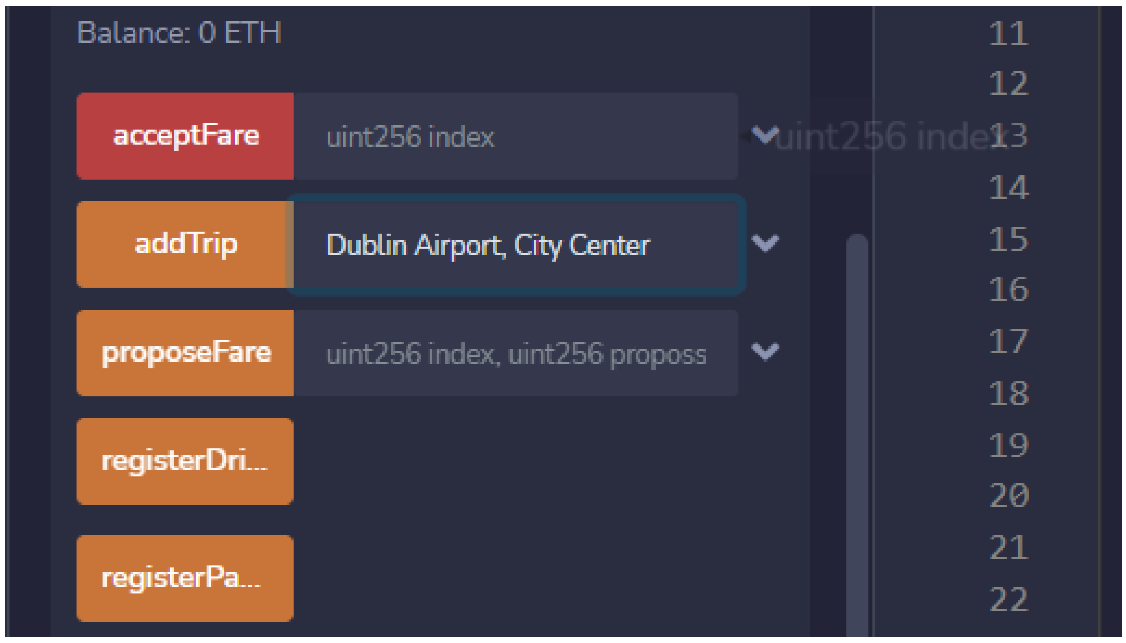
HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×3bC4A98B612baA8fEfaDb399AEe0890410fb46bd	99.98 ETH	1	0

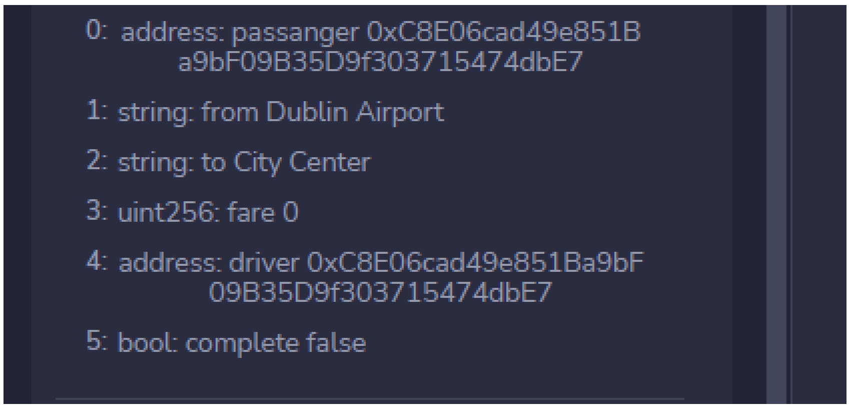
After clicking the deploy button, we can observe that the first account has paid for deploying the contract to the blockchain.

(5)

We choose the 2nd and the 3rd account to participate in the contract as driver and passenger in this order. We need to register and then, passengers can propose a trip.



We can observe that a trip was posted.

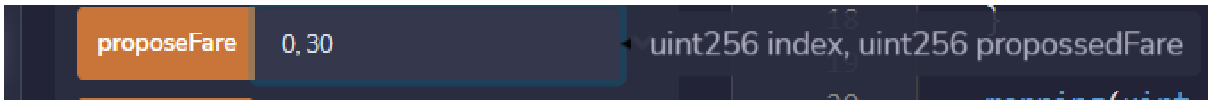


```
0: address: passanger 0xC8E06cad49e851B
         a9bF09B35D9f303715474dbE7
1: string: from Dublin Airport
2: string: to City Center
3: uint256: fare 0
4: address: driver 0xC8E06cad49e851Ba9bF
            09B35D9f303715474dbE7
5: bool: complete false
```

For now, since there are no offers on the drivers side, passenger and driver address are the same. Passenger address is used here as a placeholder.

(6)

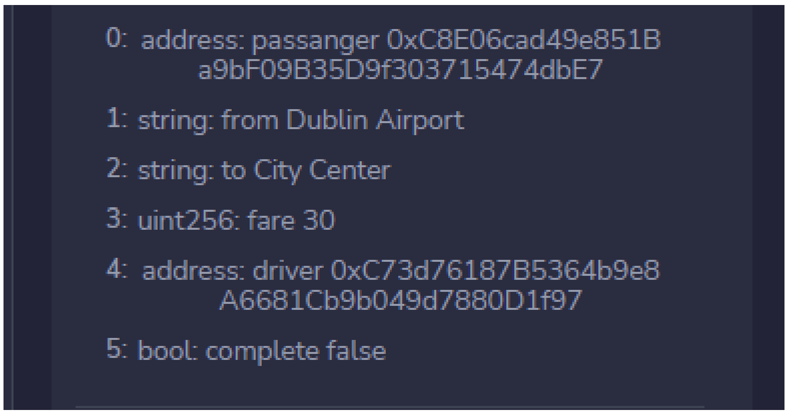
To advance, we will need to change the account on the top of the page, to interact with the contract as a registered driver.



```
proposeFare   0, 30   uint256 index, uint256 propossedFare
```

We need to specify the index of the tip we propose a fare to, and a fare.

We can verify whether our proposed fair was accepted by looking and trip details.

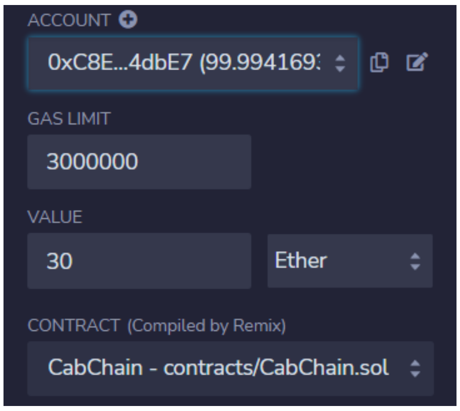


```
0: address: passanger 0xC8E06cad49e851B
         a9bF09B35D9f303715474dbE7
1: string: from Dublin Airport
2: string: to City Center
3: uint256: fare 30
4: address: driver 0xC73d76187B5364b9e8
            A6681Cb9b049d7880D1f97
5: bool: complete false
```

We can see that the driver address matches the second address from ganache server.

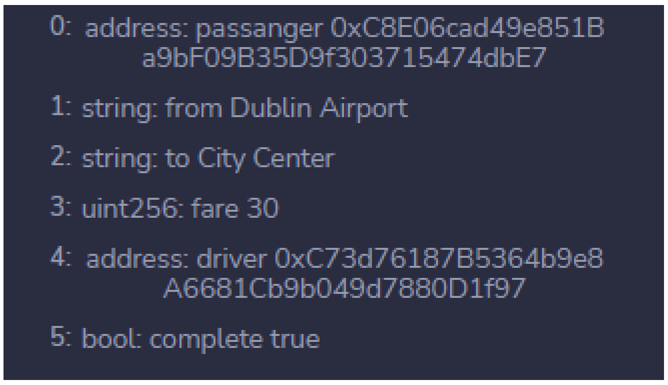
(7)

Last step is to accept the fare. In order to do that, we need to switch over to the passenger account. We also need to send required amount of ether with method call.



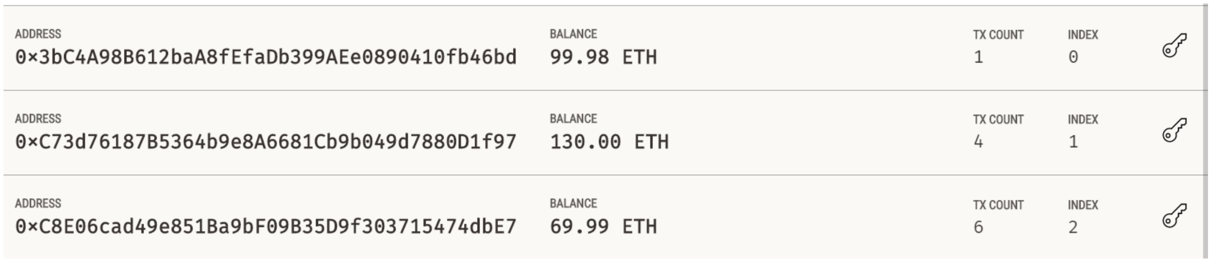
```
ACCOUNT
0xC8E...4dbE7 (99.994169:
GAS LIMIT
3000000
VALUE
30   Ether
CONTRACT (Compiled by Remix)
CabChain - contracts/CabChain.sol
```

Now we can call the method and verify the trip once more.



```
0: address: passanger 0xC8E06cad49e851B
           a9bF09B35D9f303715474dbE7
1: string: from Dublin Airport
2: string: to City Center
3: uint256: fare 30
4: address: driver 0xC73d76187B5364b9e8
           A6681Cb9b049d7880D1f97
5: bool: complete true
```

We can see that the value of the complete variable has changed to true.
More importantly, we can see that ether (30) has been transferred from the passenger to the driver account.



ADDRESS	BALANCE	TX COUNT	INDEX
0×3bC4A98B612baA8fEfaDb399AEe0890410fb46bd	99.98 ETH	1	0
0×C73d76187B5364b9e8A6681Cb9b049d7880D1f97	130.00 ETH	4	1
0×C8E06cad49e851Ba9bF09B35D9f303715474dbE7	69.99 ETH	6	2

Conclusions

We can conclude that the contract works and enables peers to register as drivers and passengers respectively. It allows passengers to post trips and drivers to propose fares. It supports transferring ether from the passenger to the driver account involved in the trip.

Individual Contributions

Robert Szlufik, #2020358

I was responsible of structuring the overall contract, adding the following driver methods:

- getManager()
- registerDriver()
- getCurrentFare()
- proposeFare()
- filterArray()

Besides that, I added the modifiers and was responsible for the video presentation, did the UML diagram and contributed with this report. I organised the workflow, researched the technology and did some troubleshooting when there were any problems with the code.

Ingrid Castro, #2020341

I was responsible for adding the following customer related methods:

- registerPassenger()
- getIndexOfLastTripInArray()
- addTrip()
- acceptFare()

Additionally, I was responsible for this report and to revise all the content. Did some research on the blockchain technology, Ethereum, organised the bibliography, edited the video and set up the overall project to be sent, observing the requirements asked.

To make it easier to demonstrate each member's contribution for the activity, there is a copy of the Contract CabChain, with highlights in blue (for Robert) and pink (for Ingrid) in the appendix.

Appendix

Copy of the Contract:

```
pragma solidity ^0.8.7;

contract CabChain {

    // data
    address private manager;
    address[] public drivers;
    address[] public passengers;
    uint private indexOfLastTrip = 0;

    struct Trip {
        address passenger; // passenger registering a trip
        string from;  // from address
        string to; // destination address
        uint fare;   // current lowest fare
        address driver; // address of a driver with the smallest offer for the fair
        bool complete;
    }

    mapping(uint => Trip) public trips;


    constructor() {
        manager = msg.sender;
    }

    // public

    function getManager() public view returns(address) {
        return manager;
    }

    function registerDriver() public {
        drivers.push(msg.sender);
    }

    function registerPassenger() public  {
        passengers.push(msg.sender);
    }
```

```
function getIndexOfLastTripInArray() public view returns(uint) {
    return indexOfLastTrip - 1;
}

// gets minimum fare for specific trip
function getCurrentFare(uint index) public view returns(uint) {
    return trips[index].fare;
}

// passengers only

function addTrip(string memory from, string memory to) public passengerOnly(msg.sender) {
    trips[indexOfLastTrip] = Trip(msg.sender, from, to, 0, msg.sender, false);
    indexOfLastTrip++;
}

// accept fare

// user needs to send fare cost with contract
function acceptFare(uint index) public payable passengerOnly(msg.sender) {
    Trip storage trip = trips[index];
    require(trip.complete == false);
    // require passenger address and person accepting to have the same address
    require(trip.passenger == msg.sender);
    // require fare amount to be send by the user
    require(trip.fare - 1 < uint248(msg.value));
    // wrap driver's address to be payable and accept ether
    address payable driver = payable(trip.driver);
    driver.transfer(msg.value);
    trip.complete = true;

}


// drivers only

function proposeFare(uint index, uint propossedFare) public driverOnly(msg.sender) {
    require(trips[index].complete == false);
    if(trips[index].fare == 0 ||trips[index].fare > propossedFare){
        trips[index].fare = propossedFare;
        trips[index].driver = msg.sender;
    } else {
        revert();
    }
    }



    // modifiers

    modifier passengerOnly(address sender) {
        require(filterArray(passengers, sender));
        _;
    }

    modifier driverOnly(address sender) {
        require(filterArray(drivers, sender));
        _;
    }



    // utils

    function filterArray(address[] memory array, address  sender) private returns(bool) {
        bool includes = false;
        for(uint i = 0; i < array.length; i++) {
            if(array[i] == sender){
                includes = true;
            }
        }
        return includes;
    }
}
```

References

Budhi, V. (2022) *Council post: Advantages and disadvantages of Blockchain technology*, *Forbes*. Forbes Magazine. Available at: https://www.forbes.com/sites/forbestechcouncil/2022/10/20/advantages-and-disadvantages-of-blockchain-technology/ (Accessed: December 4, 2022).

ChainTrade, . (2017) *10 advantages of using smart contracts*, *Medium*. Medium. Available at: https://medium.com/@ChainTrade/10-advantages-of-using-smart-contracts-bc29c508691a (Accessed: December 3, 2022).

Frankenfield, J. (2022) *What is ethereum and how does it work?*, *Investopedia*. Edited by S. Anderson. Investopedia. Available at: https://www.investopedia.com/terms/e/ethereum.asp (Accessed: December 3, 2022).

Sahu, M. (2022) *Cryptography in Blockchain: Types & applications [2023]*, *upGrad blog*. upGrad blog. Available at: https://www.upgrad.com/blog/cryptography-in-blockchain/ (Accessed: December 4, 2022).

Sinkevicius, A. (2022) *Which algorithm is used to Encrypt Ethereum private keys?*, *Medium*. Coinmonks. Available at: https://medium.com/coinmonks/which-algorithm-is-used-to-encrypt-ethereum-private-keys-e4077496aff8 (Accessed: December 4, 2022).

Team, C.F.I. (2022) *Smart contracts*, *Corporate Finance Institute*. Corporate Finance Institute. Available at: https://corporatefinanceinstitute.com/resources/valuation/smart-contracts/ (Accessed: December 3, 2022).

Warren, M. (2022) *Why use solidity for blockchain development*, *BairesDev*. BairesDev. Available at: https://www.bairesdev.com/blog/why-use-solidity-for-blockchain-development/ (Accessed: December 3, 2022).