School register project

Author: Andrei Marchenko ftkvyn@gmail.com, @ftkvyn, LinkedIn

Profile on forum.freeton.org

* FreeTON network is supported as well.

Short description

The goal of the project is to implement the register for giving grades to the students in a school or university. The grades are stored in TON blockchain. It is not possible to remove or to edit the grade after it was given. It's also not possible to fake the grades.

Components

On-chain

Following instruments are working on TON blockchain:

- Smartcontract (FreeTON version, Solidity)
    - **register.sol** - source code of the contract. Creates and stores the students and the grades.
    - **register.code**, **register.abi.json** - compiled from **register.sol** using `solc` Solidity to TON compiler.
- Smartcontract (Classic TON version)
    - **register.fc** - creates and stores the students and the grades.
- Fift scripts
    - **new-register.fif** - creates new register smart contract
    - **new-student.fif** - adds a student to the existing register smart contract
    - **new-mark.fif** - gives a new mark to the student on the existing register

Each register contract stores two keys - teacher's one and the principal's. The teacher's key is used to add students and to give grades. In case when the teacher should transfer her register to another teacher - the teacher's key may be changed. Both teacher and principal may do this. In case when the teacher is not avaliable - the principal may also transfer the register to another teacher.

The grades are only created - after the creation it's not possible to remove or change it. It was made on purpose to ensure honesty of the register usage and to prevent any possible frauds and machinations.

The project was made with the assumption that the student's grade is not a secret or sencitive information. That's why the grades are avaliable as-is, without any encription.

Off-chain (Only for Classic TON, FreeTON is not yet supported)

- Teacher's application
    - Creates and manage several registers (for different classes and subjects)
    - Displays list of students for each register
    - Adds new students
    - Displays student's marks
    - Gives marks
- Student's application
    - Handles several existing register
    - Displays the single student's marks in given register

Requirements (FreeTON)

Applications are developed and tested on Ubuntu Linux, correct work on other systems is not guaraneed. To be able to build and run scripts and to be able to use teacher's and student's application following requirements have to be fulfilled:

- Download or build from sources FreeTON tools - Solidity compiler and `tvm_linker` as described in the instruction:
    - Solidity tools for FreeTON: https://docs.ton.dev/86757ecb2/p/950f8a-write-smart-contract-in-solidity
    - General FreeTON tools: https://docs.ton.dev/86757ecb2/p/552389-general
    - Solidity to TON compiler: https://github.com/tonlabs/TON-Solidity-Compiler
    - `tvm_linker` : https://github.com/tonlabs/TVM-linker
    - Useful tookit for communitcation with TON - tonos-cli https://github.com/tonlabs/tonos-cli

Build instructions (FreeTON)

Crypto

To build smart contract from Solidity sources, go to `freeton-crypto` folder and run

```
solc register.sol
```

Then to prepare the contract for uploading to the blockchain, run:

```
tvm_linker compile register.code --lib /usr/lib/stdlib_sol.tvm --abi-json
register.abi.json
```

You'll get a file with a long name and `.tvc` extension. The file name is the address of your contract. Then run the following command to generate your keys:

```
tonos-cli genaddr <contract>.tvc register.abi.json --genkey register.keys.json
```

Address of your contract in the blockchain is located after `Raw address:`. It's time to send some coins to that address. After you've done that, you may deploy your contract using the following instruction:

```
tonos-cli deploy --abi register.abi.json --sign register.keys.json <contract>.tvc
{}
```

Congratulations! Now you may run following commands on the contract:

Creating a student:

```
tonos-cli call '<YourAddress>' addStudent '{"studentId":"123"}' --abi
register.abi.json --sign register.keys.json
```

Creating a mark:

```
tonos-cli call '<YourAddress>' addMark '{"studentId":"123", "markValue":"5",
"message":"Good job!"}' --abi register.abi.json --sign register.keys.json
```

Getting students:

```
tonos-cli call '<YourAddress>' getStudents '{}' --abi register.abi.json
```

Getting a students mark:

```
tonos-cli call '<YourAddress>' getStudentMarks '{"studentId":"123"}' --abi
register.abi.json
```

Requirements (Classic TON)

Applications are developed and tested on Ubuntu Linux, correct work on other systems is not guaraneed. To be able to build and run scripts and to be able to use teacher's and student's application following requirements have to be fulfilled:

- Build TON libraries (`lite-client` , `fift` and `func`) as described in the instruction on https://test.ton.org/;
- Set up correct `FIFTPATH` ;
- Set up links to `fift` and `func` so that they can be used from the command line.
- For now the applications save their generated files (private keys, contract addresses and .boc files to upload to the blockchain) to the directory from which they are runned, so they need to have write permissions for their directories.

Build instructions (Classic TON)

Crypto

To build fift scripts from sources, run

```
./build-crypto.sh
```

Applications

Enter the application folder, `/src/teacher-app` or `/src/student-app` , run

```
npm install
npm start
```

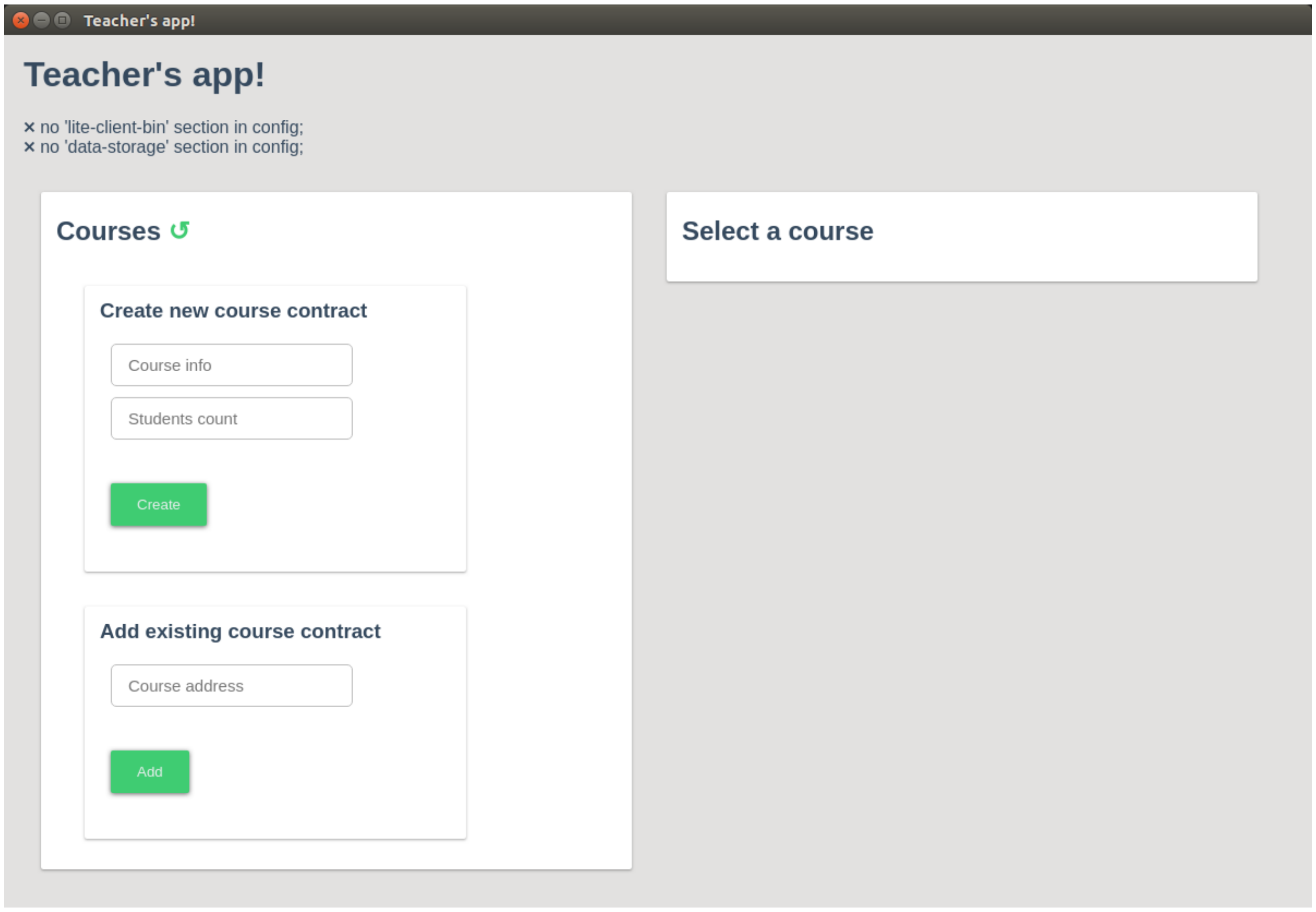
Instructions (For the Classic TON version)

Teacher's app

First, config file should be set up. It is located in the file `/src/teacher-app/config.json` . It should have following sections:

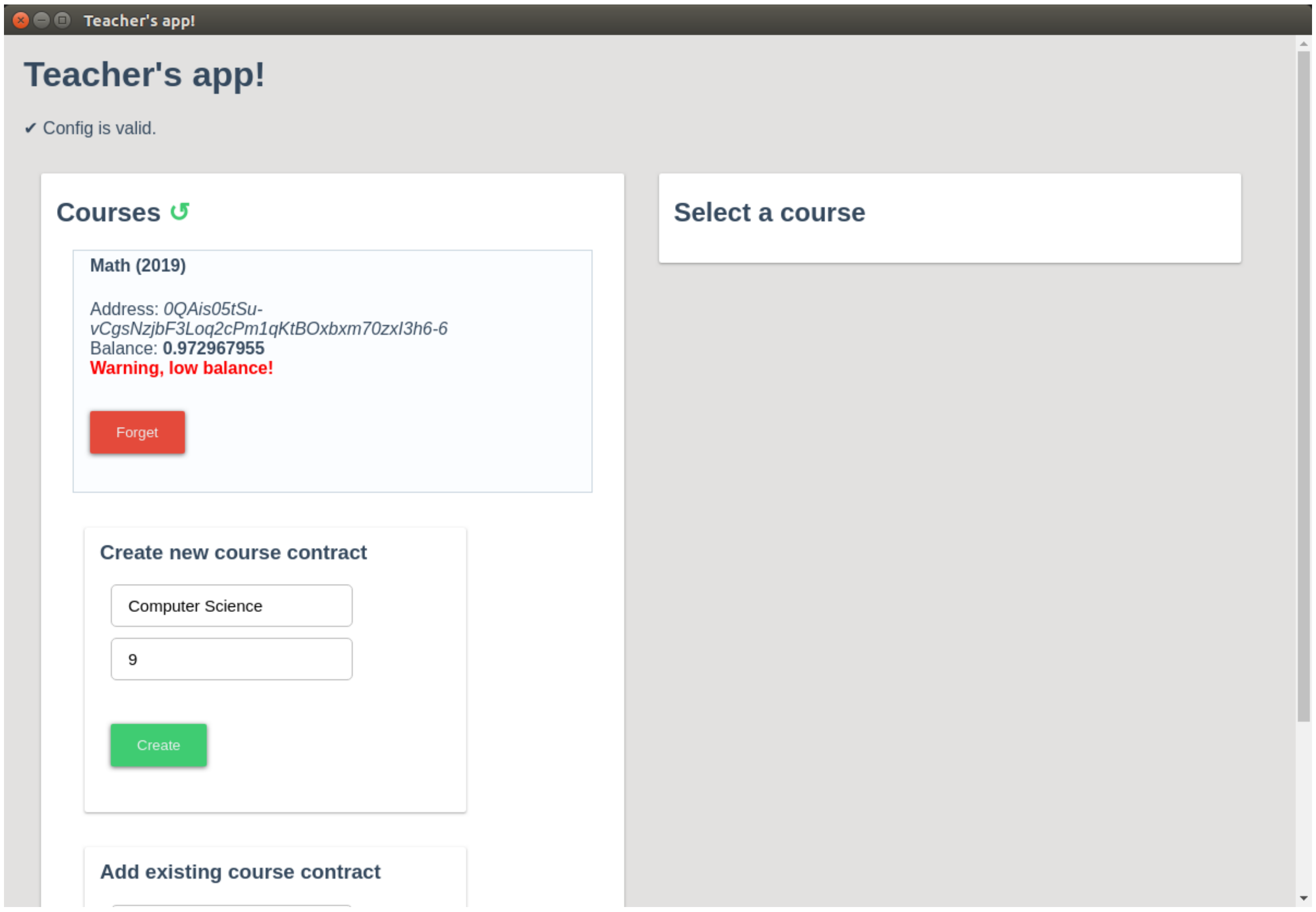
- `"lite-client-config"` - path to the configuration file of lite-client.
- `"lite-client-bin"` - path to the binary file of the lite-client.
- `"data-storage"` - file that stores application's data, the application needs to have read and write permissions for this file.
- `"fift-folder"` - folder with all the fift scripts that are used by the application. May be set to `/dist` folder of current repo folder.

If something is not correct in the configuration, it will be displayed on the application's screen:


Teacher's app!

× no 'lite-client-bin' section in config;
× no 'data-storage' section in config;

First, we are creating a new register contract:


Teacher's app!

✔ Config is valid.

Courses ↺

Math (2019)

Address: *0QAis05tSu-vCgsNzjbF3Loq2cPm1qKtBOxbxm70zxI3h6-6*
Balance: **0.972967955**
Warning, low balance!

Forget

Create new course contract

Computer Science

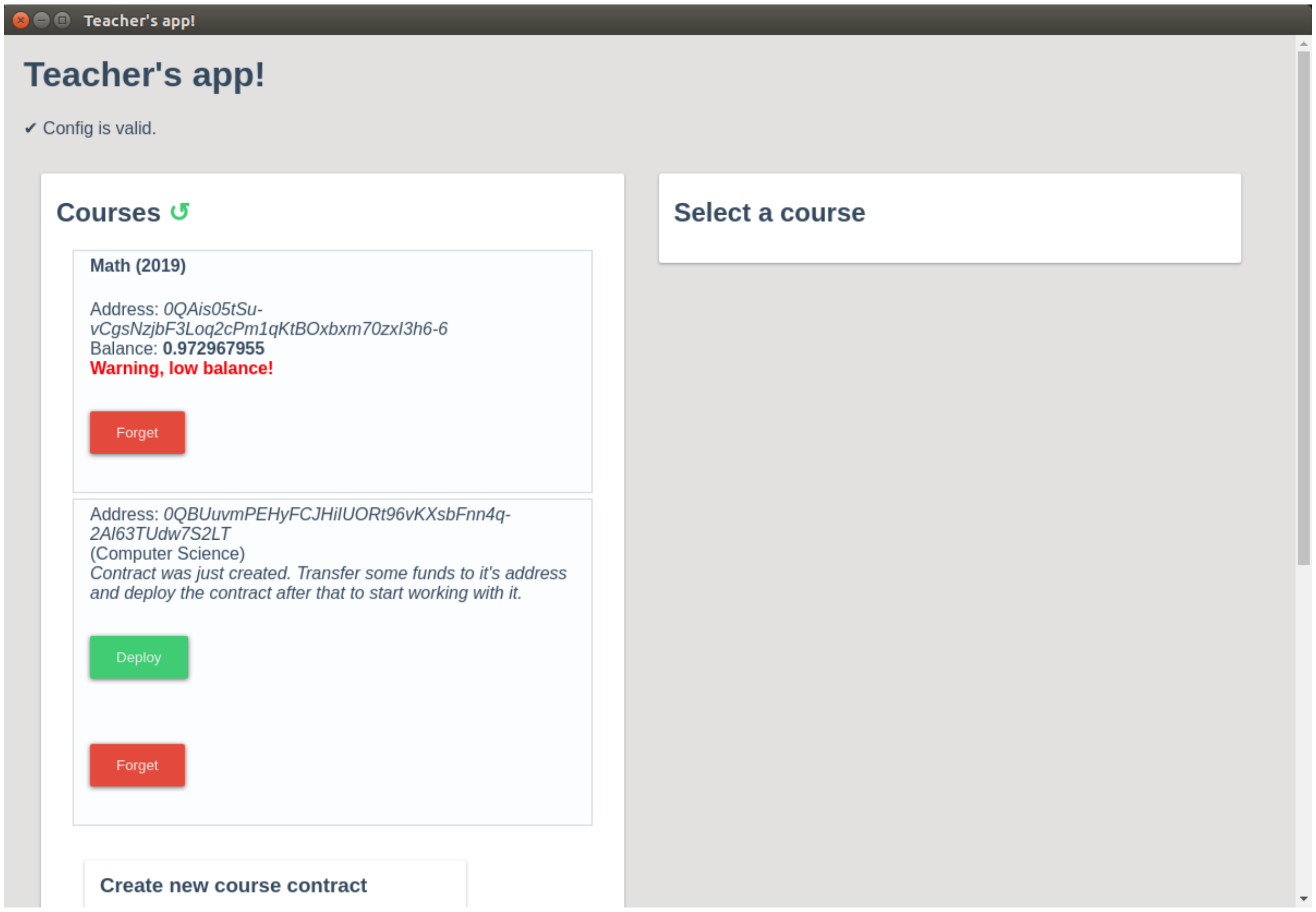
9

Create

Add existing course contract

Select a course

The contract is created:


Teacher's app!

✔ Config is valid.

Courses ↺

Math (2019)

Address: *0QAis05tSu-vCgsNzjbF3Loq2cPm1qKtBOxbxm70zxI3h6-6*
Balance: **0.972967955**
Warning, low balance!

Forget

Address: *0QBUuvmPEHyFCJHiIUORt96vKXsbFnn4q-2AI63TUdw7S2LT*
(Computer Science)
Contract was just created. Transfer some funds to it's address and deploy the contract after that to start working with it.

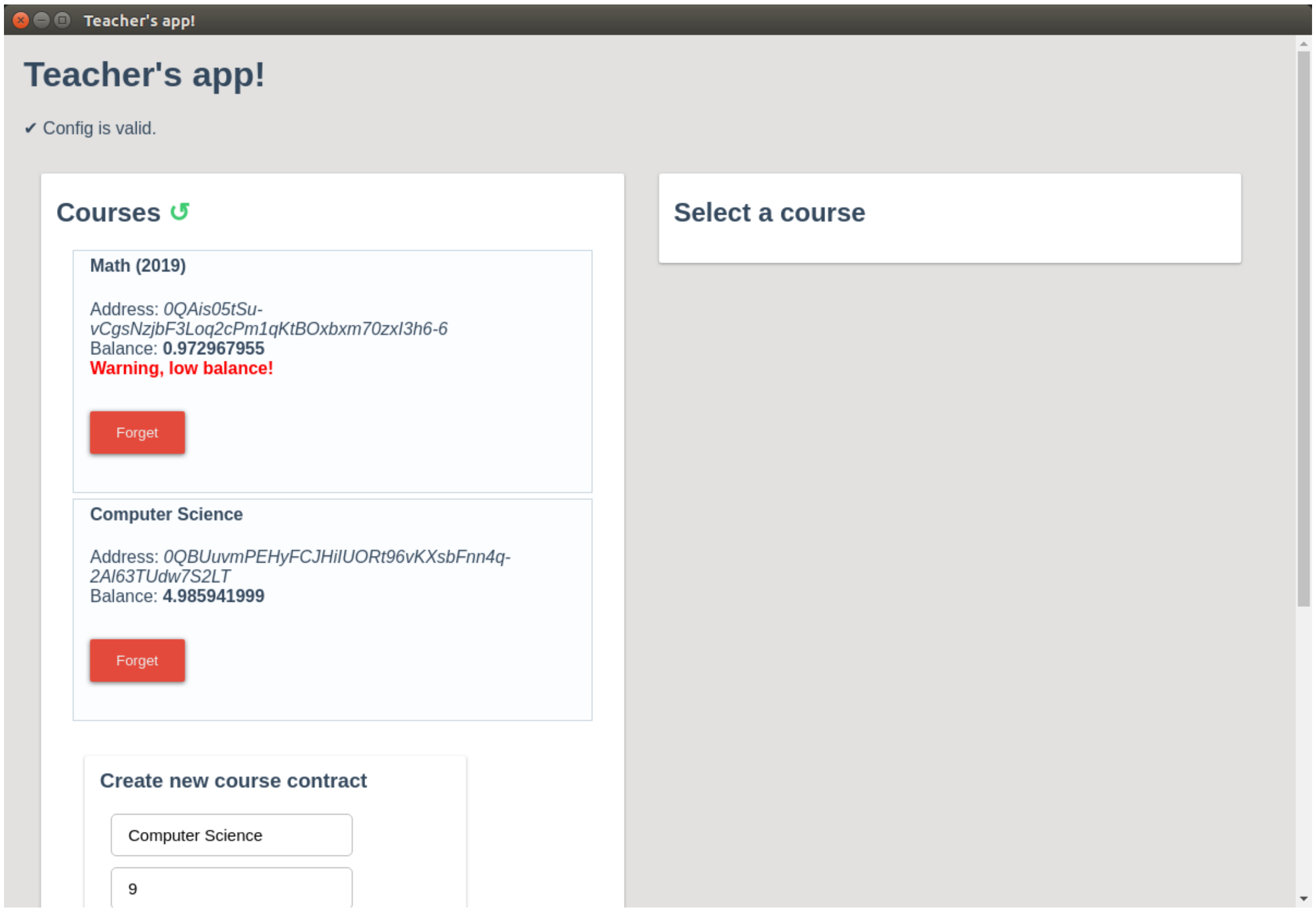
Deploy

Forget

Create new course contract

Select a course

To Use it we need to transfer some funds to it's address (0QBUuvmPEHyFCJHiIUORt96vKXsbFnn4q-2Al63TUdw7S2LT) and then click "Deploy". Then we wait some time while the payment and init message are processed and voilà, the contract is created and ready to be used:


Teacher's app!

Teacher's app!

✔ Config is valid.

Courses ↺

Math (2019)

Address: *0QAis05tSu-vCgsNzjbF3Loq2cPm1qKtBOxbxm70zxI3h6-6*
Balance: **0.972967955**
Warning, low balance!

Computer Science

Address: *0QBUuvmPEHyFCJHiIUORt96vKXsbFnn4q-2Al63TUdw7S2LT*
Balance: **4.985941999**

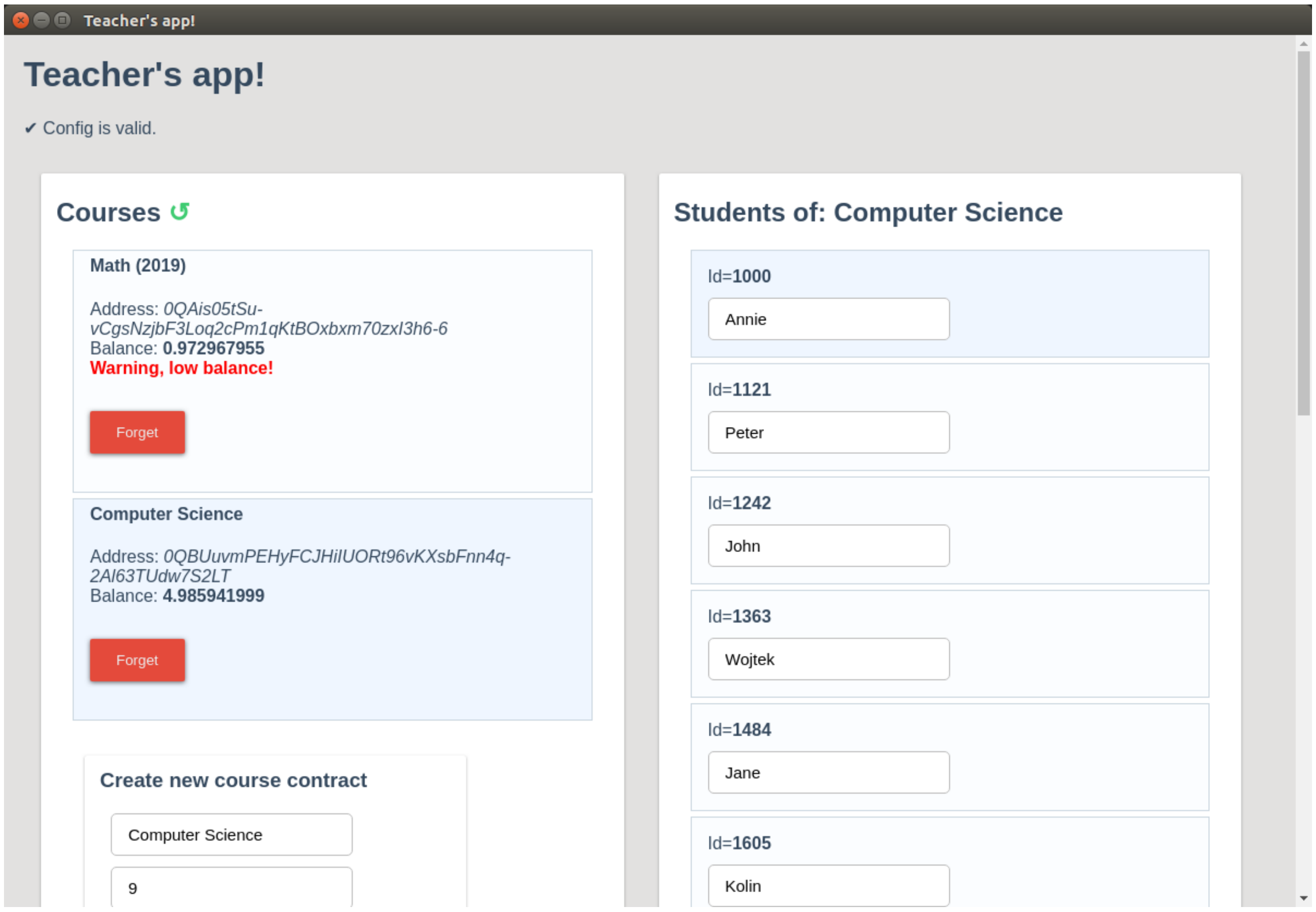
Create new course contract

Computer Science

9

Select a course

We may click on the contract to select it. Then a list of students will appear on the right side. Student names aren't stored in the blockchain, only in the application's local data storage.



Teacher's app!

Teacher's app!

✔ Config is valid.

Courses ↺

Math (2019)

Address: *0QAis05tSu-vCgsNzjbF3Loq2cPm1qKtBOxbxm70zxI3h6-6*
Balance: **0.972967955**
Warning, low balance!

Forget

Computer Science

Address: *0QBUuvmPEHyFCJHiIUORt96vKXsbFnn4q-2AI63TUdw7S2LT*
Balance: **4.985941999**

Forget

Create new course contract

Computer Science

9

Students of: Computer Science

Id=**1000** Annie

Id=**1121** Peter

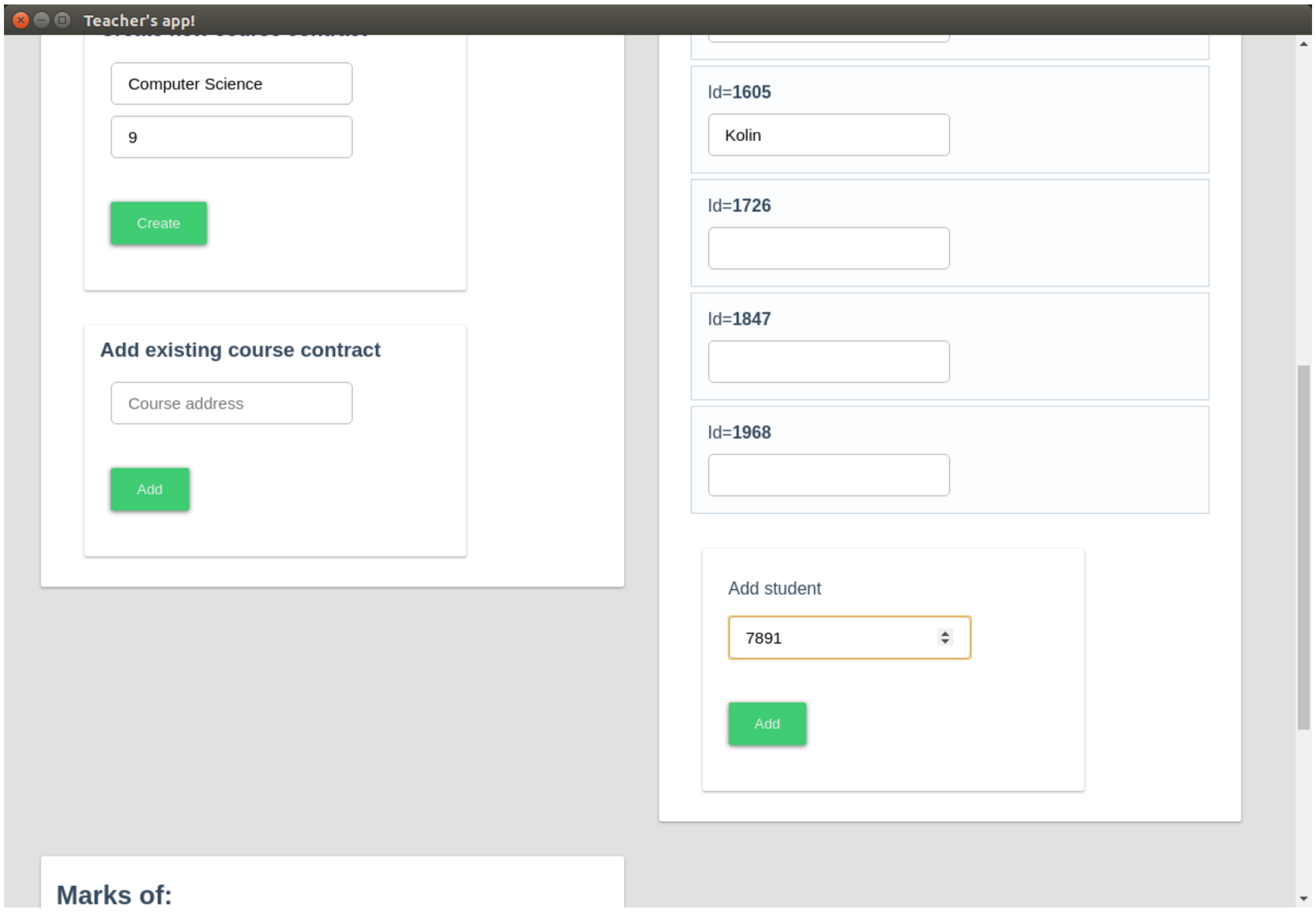
Id=**1242** John

Id=**1363** Wojtek

Id=**1484** Jane

Id=**1605** Kolin

We may add new students as well:



Teacher's app!

Computer Science

9

Create

Add existing course contract

Course address

Add

Id=**1605** Kolin

Id=**1726**

Id=**1847**

Id=**1968**

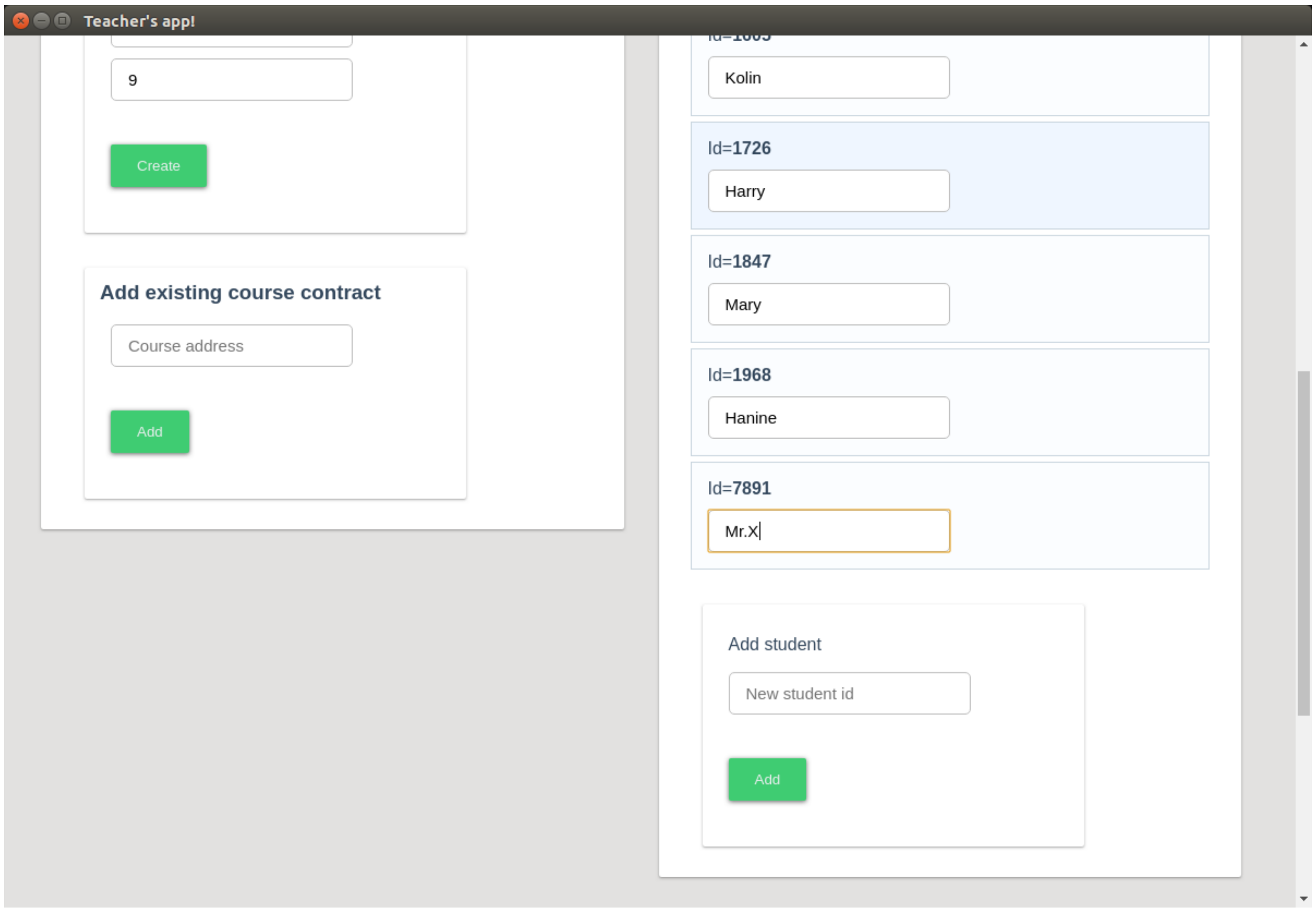
Add student

7891

Add

Marks of:

To refresh the students list click on the contract again. The new student will appear when it will be saved in the blockchain.


Teacher's app!

9

Create

Add existing course contract

Course address

Add

Kolin

Id=1726

Harry

Id=1847

Mary

Id=1968

Hanine

Id=7891

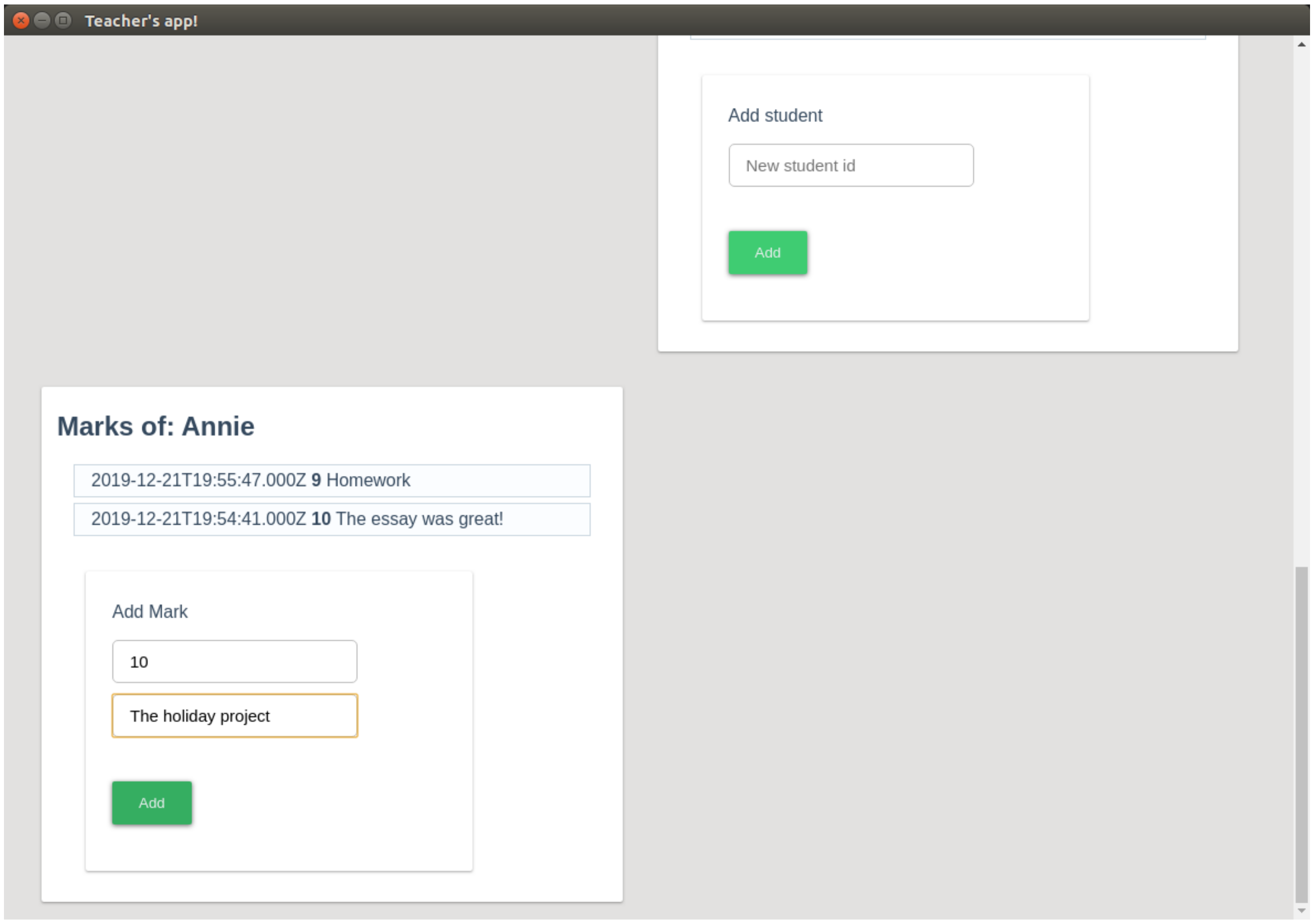
Mr.X

Add student

New student id

Add

Now we may select a student and we'll see her marks and we'll be able to give her new ones. It takes some time for new grades to be saved to the blockchain, so be patient ;)



Marks of: Annie

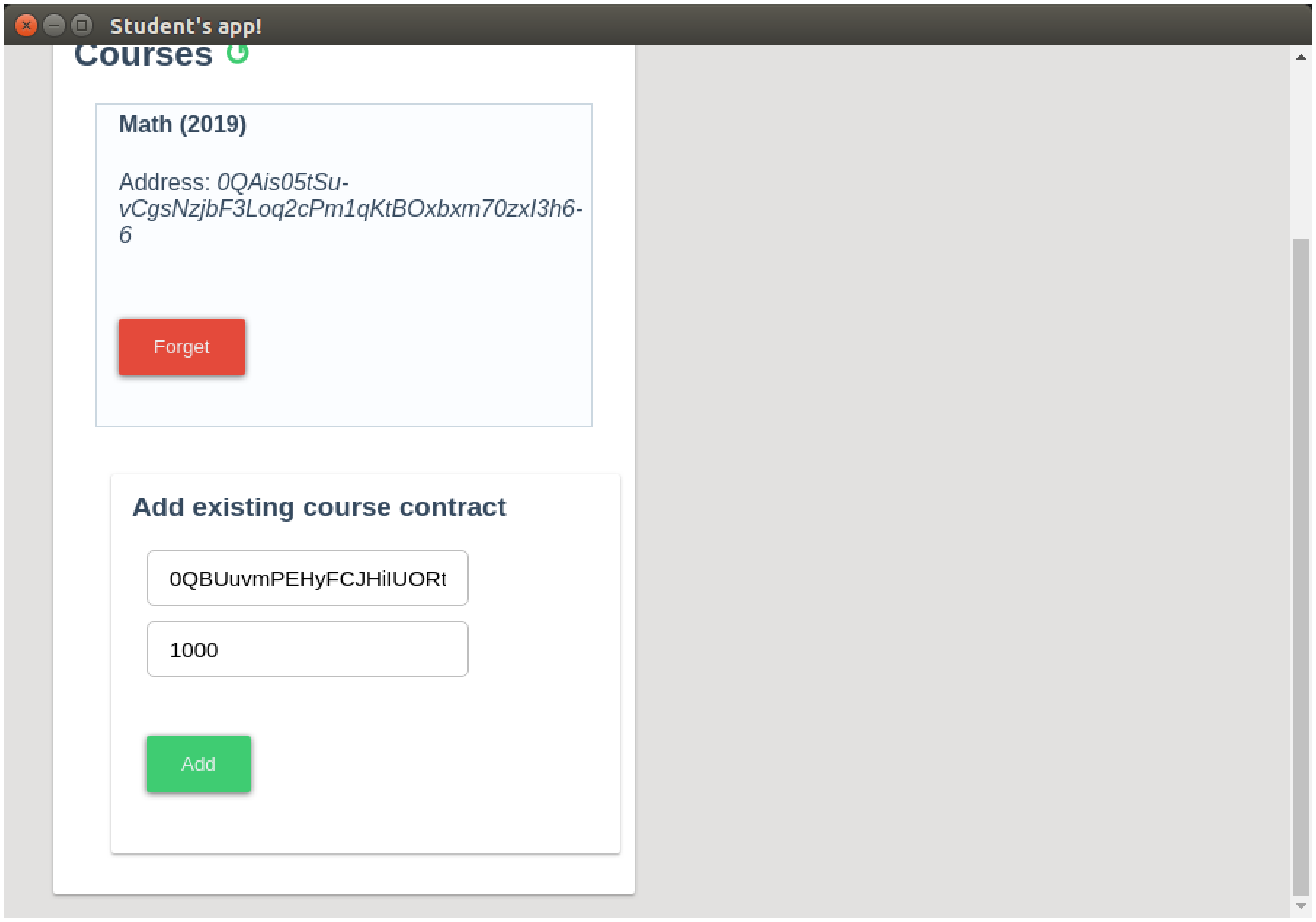
2019-12-21T19:55:47.000Z **9** Homework

2019-12-21T19:54:41.000Z **10** The essay was great!

Student's app

Student's application is a somewhat reduced version of the teacher's one. It can't deploy new contracts and can't give grades. The configuration is similar to the teacher's app, the only difference is that student's one doesn't need "fift-folder" configuration entry. And for sure it has to have it's own separate file for data storage.

In the student's app we need to provide the information about the register that is to be used. The student should get from the teacher the contract address and her own id:


Student's app!

Courses

Math (2019)

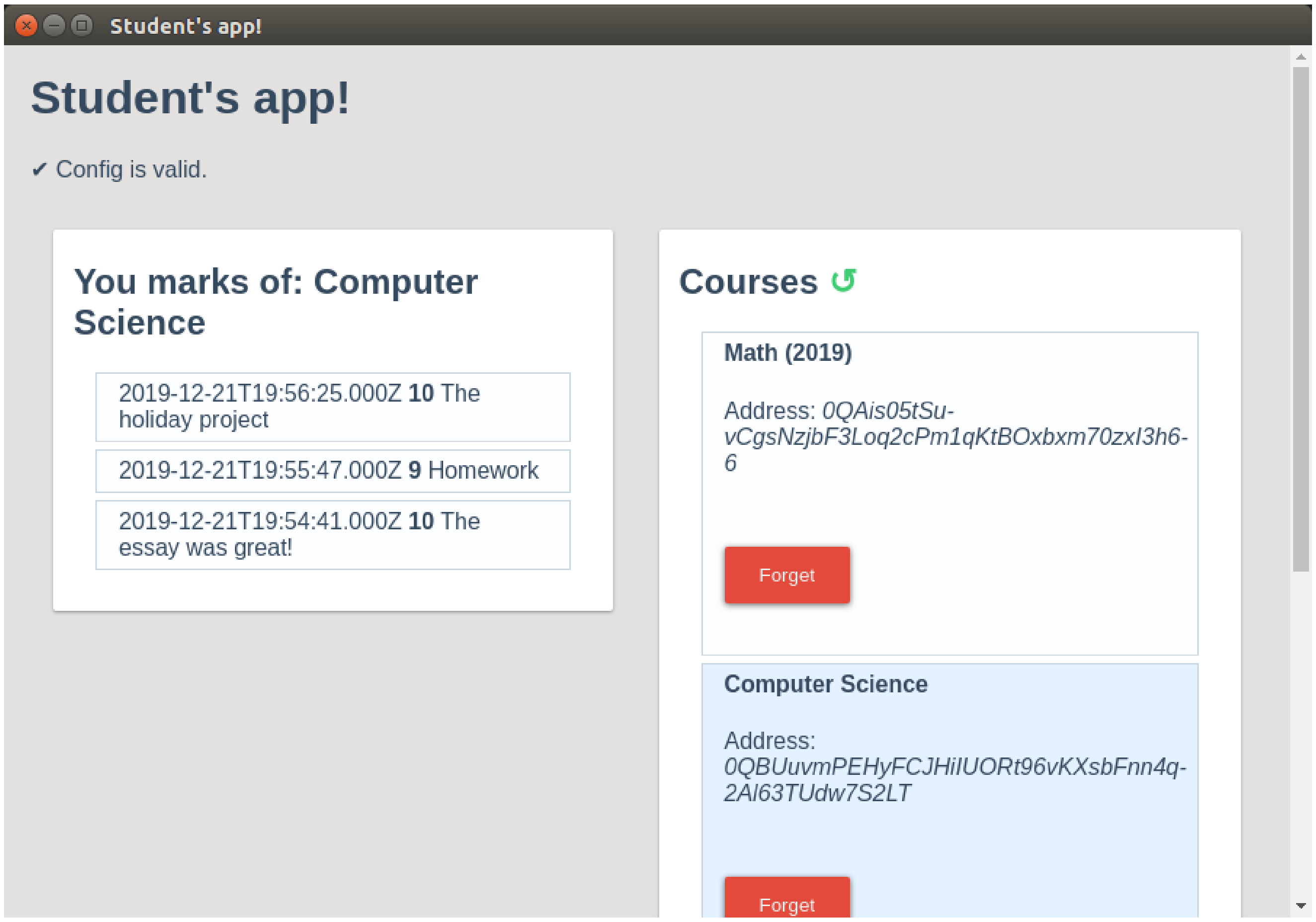
Address: *0QAis05tSu-vCgsNzjbF3Loq2cPm1qKtBOxbxm70zxl3h6-6*

Add existing course contract

0QBUuvmPEHyFCJHiIUORt

1000

She then will be able to select the contract and see her grades:


Student's app!

Student's app!

✔ Config is valid.

You marks of: Computer Science

2019-12-21T19:56:25.000Z **10** The holiday project

2019-12-21T19:55:47.000Z **9** Homework

2019-12-21T19:54:41.000Z **10** The essay was great!

Courses ↺

Math (2019)

Address: *0QAis05tSu-vCgsNzjbF3Loq2cPm1qKtBOxbxm70zxI3h6-6*

Computer Science

Address: *0QBUuvmPEHyFCJHiIUORt96vKXsbFnn4q-2AI63TUdw7S2LT*