

Subject: - CCMP-606-T01 Integrated  
Services Using Smart Contracts

Title: - Smart Contract for Freelance Service  
Management

Professor: - Prakhyat Khati

Project Details:

[https://github.com/yash91066/block\\_chain\\_freelancer.git](https://github.com/yash91066/block_chain_freelancer.git)

Members: - Aksh Patel, Prem Solanki, Yash  
Patel

## 1. Problem Definition

The freelance economy has grown rapidly, but it still faces several challenges:

1. **Trust Issues:** Clients may hesitate to pay upfront, while freelancers worry about not receiving payments after completing work.
2. **Manual Processes:** Relying on intermediaries or platforms increases overhead costs and delays.
3. **Lack of Accountability:** No transparent system exists for enforcing agreed-upon terms or collecting reliable feedback.

## Proposed Solution

This project implements a **Freelance Service Management Smart Contract** on the Ethereum blockchain to:

- Provide a transparent, automated system to manage freelance contracts.
- Secure payments through deposits in the blockchain until task completion.
- Enable feedback mechanisms for future client-freelancer relationships.

The smart contract eliminates the need for intermediaries, reduces transaction costs, and ensures fair dealings between parties.

---

## 2. Design and Architecture

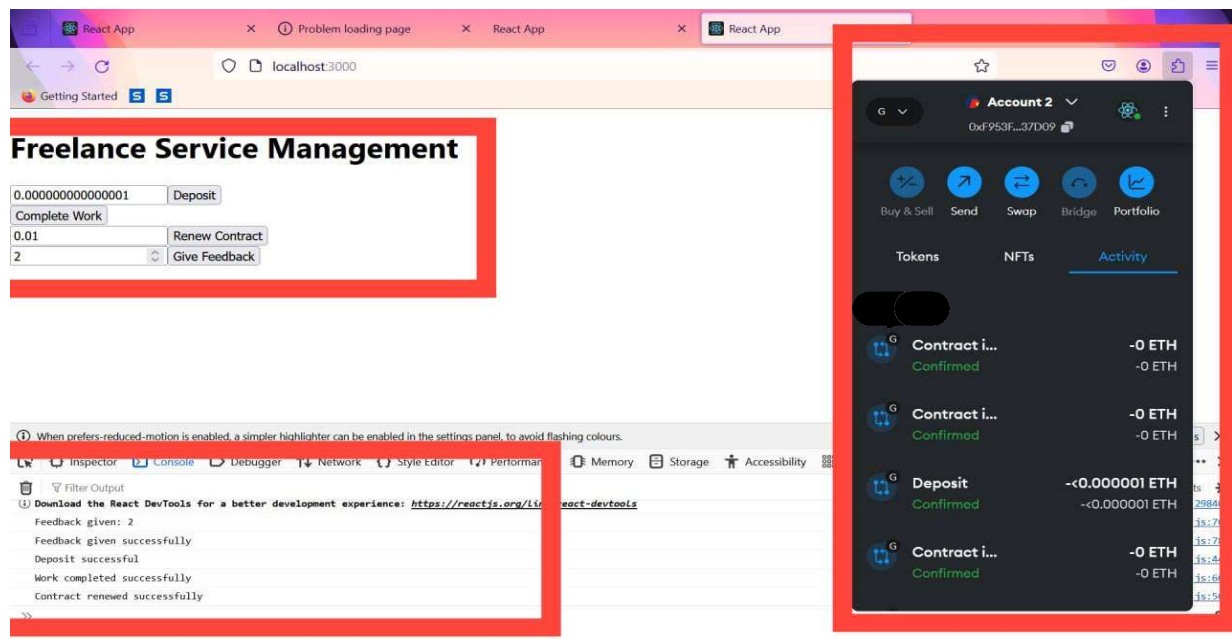
### How It Works

1. **Client deposits money:** The client deposits funds into the smart contract. This money is locked in until the freelancer completes their work.
2. **Freelancer finishes the task:** The freelancer notifies the contract when the work is done. The system checks and releases the payment.
3. **Feedback is provided:** The client gives a rating to the freelancer, which is stored on the blockchain for future reference.

## Tools Used

- **Ethereum Blockchain:** The backend where all transactions and functions are executed securely.
- **Ganache:** A tool to simulate a blockchain locally for testing purposes.
- **Truffle:** Used to write, compile, and deploy the smart contract.
- **MetaMask:** A browser extension for handling blockchain accounts and transactions.
- **Frontend Web Interface:** A user-friendly interface to interact with the contract.

## System Diagram



## 3. Implementation

### Setup Process

#### 1. Install Required Tools

- Install **Node.js** to manage dependencies.
- Install **Truffle** (a framework for Ethereum development).
- Download and install **Ganache** to simulate a blockchain locally.

## 2. Project Setup

- Extract the project files into a folder.
- Open the folder in VS Code or another editor.

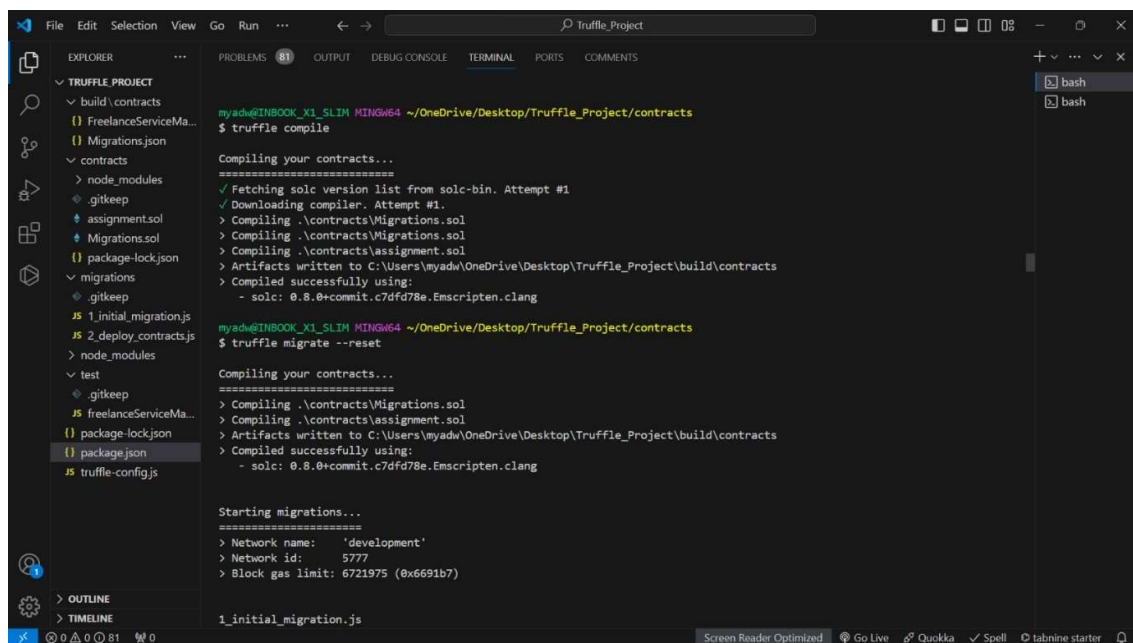
## 3. Run the Blockchain

- Start Ganache, which creates a local blockchain.
- Note the network address (e.g., <http://127.0.0.1:7545>).

## 4. Deploy the Smart Contract

- Open a terminal in the project folder and run the following commands:

**truffle compile** # Compile our the smart contract



The screenshot shows the Visual Studio Code interface with a project named 'Truffle\_Project'. The Explorer panel on the left shows the project structure, including 'build', 'contracts', 'migrations', and 'test' folders. The Terminal panel on the right shows the execution of 'truffle compile' and 'truffle migrate --reset' commands. The output of these commands is displayed in the terminal, showing the compilation of smart contracts and the successful migration of the blockchain network.

```
myadw@INBOOK_X1_SLIM MINGW64 ~/OneDrive/Desktop/Truffle_Project/contracts
$ truffle compile

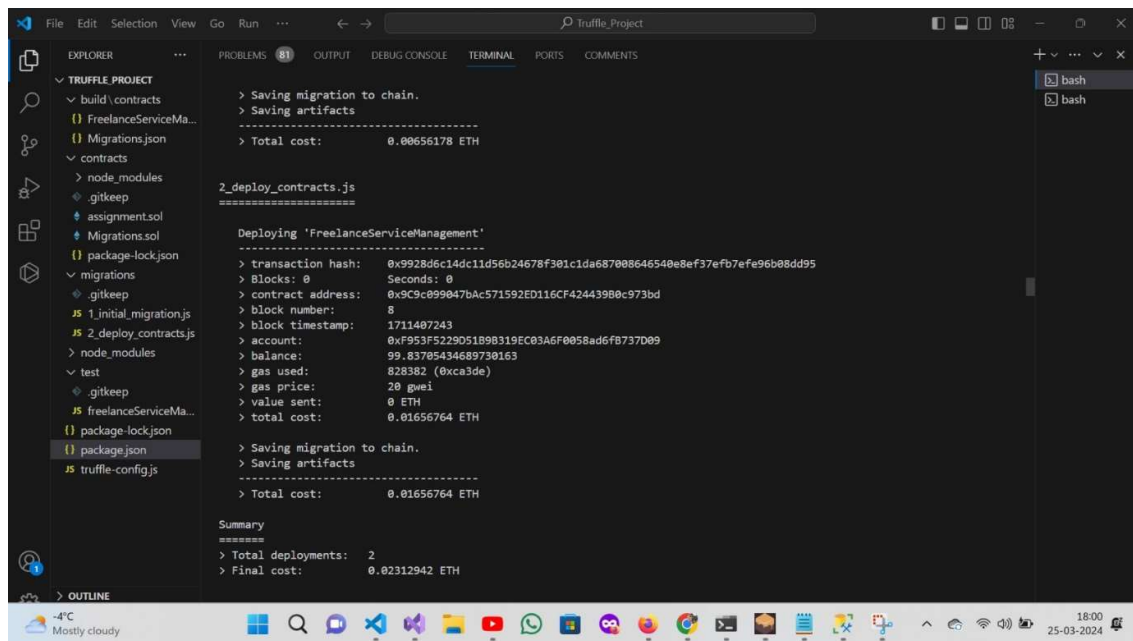
Compiling your contracts...
=====
✓ Fetching solc version list from solc-bin. Attempt #1
✓ Downloading compiler. Attempt #1.
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\assignment.sol
> Artifacts written to C:\Users\myadw\OneDrive\Desktop\Truffle_Project\build\contracts
> Compiled successfully using:
   - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang

myadw@INBOOK_X1_SLIM MINGW64 ~/OneDrive/Desktop/Truffle_Project/contracts
$ truffle migrate --reset

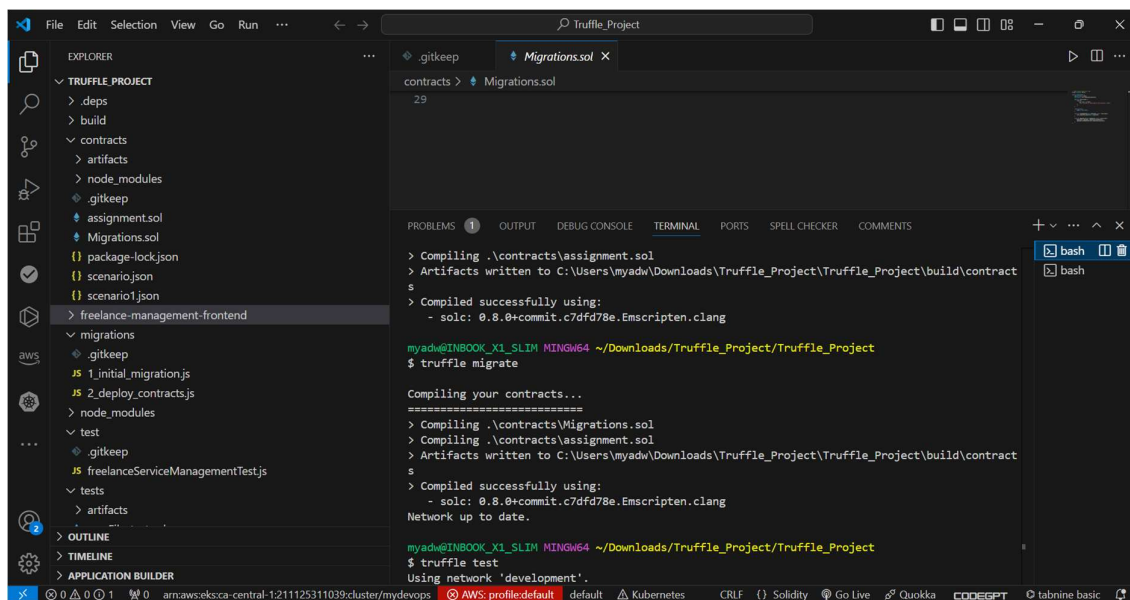
Compiling your contracts...
=====
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\assignment.sol
> Artifacts written to C:\Users\myadw\OneDrive\Desktop\Truffle_Project\build\contracts
> Compiled successfully using:
   - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang

Starting migrations...
=====
> Network name: 'development'
> Network id: 5777
> Block gas limit: 6721975 (0x6691b7)

1_initial_migration.js
```



**truffle migrate # Deploy the smart contract**

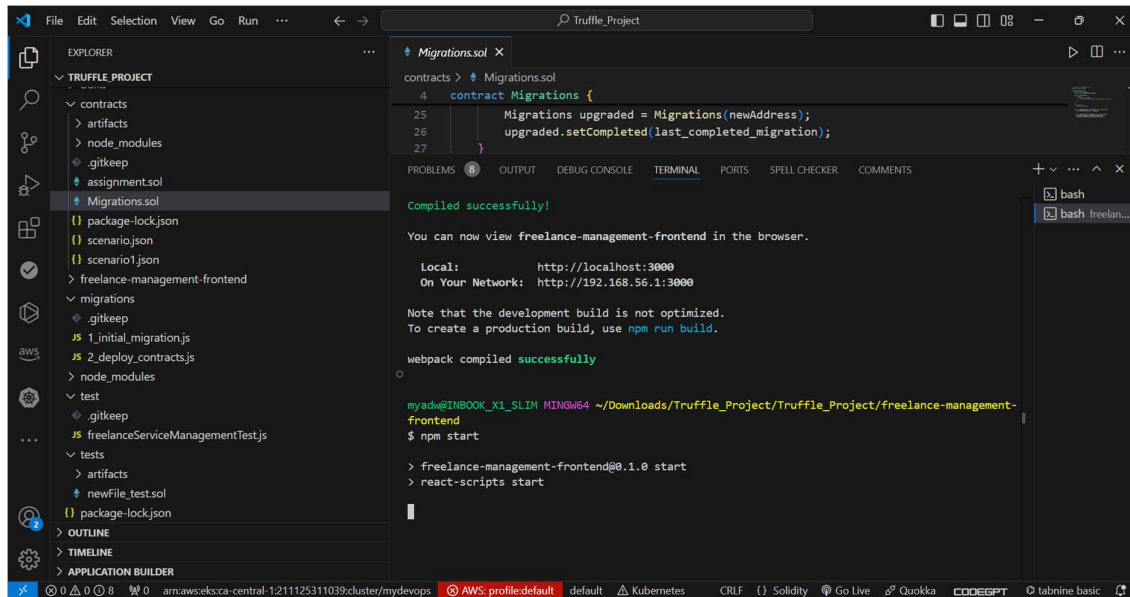


## 5. Launch the Frontend

- Navigate to the frontend folder:

**cd freelance-management-frontend**

**npm start**



- Open your browser and access the interface.

## 6. Connect MetaMask

- Set MetaMask to connect to the Ganache network.
- Use the accounts from Ganache for testing the system.

---

## 4. Key Functionalities

### 1. Deposit Funds

- **Functionality:**

The client deposits an agreed amount (e.g., 0.0000000000000001 ETH).

- The smart contract locks the funds and confirms the transaction, ensuring payment security and initiating the contract workflow.

## 1.1 Details of Deposit Functionality:

- **Input Parameter Format:**

The deposit function expects an input value in Ether, specifically 0.0000000000000001 ETH for testing purposes. This amount must be provided in the **transaction value field** during the transaction, not passed as an argument to the function.

- **Expected Result:**

- If the exact value of 0.0000000000000001 ETH is sent:
  - The smart contract records the deposit successfully.
  - A success message is displayed in the console.
- If a different amount is entered:
  - The transaction fails and reverts.
  - An error message explains that the deposit amount must match the exact required value.

This ensures the system adheres to agreed terms and protects both the client and freelancer from unintended discrepancies.

## 2. Complete Work

- **Functionality:**

The freelancer, upon completing their assigned task, invokes the `completeWork()` function.

- The system automatically transfers the locked funds to the freelancer's address.
- The contract verifies task completion before executing the transfer, ensuring the freelancer is paid securely and trustlessly.

## 3. Renew Contract

- **Functionality:**

If both the client and freelancer agree, they can extend the contract for additional work by invoking the `renewContract()` function.

- This allows renegotiation of terms without creating a new contract, simplifying ongoing collaborations.

- Ensures efficient contract management for long-term partnerships.

#### 4. Feedback System

- **Functionality:**

The client provides feedback on the freelancer's performance using the `giveFeedback()` function.

- The feedback is recorded on the blockchain, contributing to the freelancer's reputation.
- This stored rating ensures transparency and accountability for future engagements.

#### Test Results

The smart contract was tested successfully in the following scenarios:

Function	Action Taken	Result
<b>Deposit</b>	Client deposited funds	Funds locked in the contract.
<b>Complete Work</b>	Freelancer marked task complete	Funds released successfully.
<b>Renew Contract</b>	Updated contract terms	Contract renewed successfully.
<b>Feedback</b>	Client provided feedback	Rating stored on blockchain.

---



0.000000000000000000  
Complete V  
0.01  
2

Free

0.000000000000000000  
Complete V  
0.01  
2

When pr  
In  
Downloa  
Feedback  
Feedback  
Deposit  
Work co  
Contract  
>>

Ganache

ACCOUNTS BLOCKS TRANSACTIONS CONTRACTS EVENTS LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK 235 GAS PRICE 20000000000 GAS LIMIT 6721975 HARDFORK MERGE NETWORK ID 5777 RPC SERVER HTTP://127.0.0.1:7545 MINING STATUS AUTOMINING WORKSPACE YASHBRO-123 SWITCH

TX HASH  
0x2ed8f6e9aaa007fae56c85762394ea7bb9e937c408b3076bd9ad69b316e7c065

CONTRACT CALL

FROM ADDRESS 0xF953F5229D51B9B319EC03A6F0058ad6fB737D09 TO CONTRACT ADDRESS 0x9C9c099047bAc571592ED116CF424439B0c973bd GAS USED 27670 VALUE 1000

TX HASH  
0x37ea08bcf1ac83845c13e22fe9f6e4dd020649b396217aa1c6b7ac3bc3ac01c6

CONTRACT CALL

FROM ADDRESS 0xF953F5229D51B9B319EC03A6F0058ad6fB737D09 TO CONTRACT ADDRESS 0x9C9c099047bAc571592ED116CF424439B0c973bd GAS USED 31062 VALUE 0

TX HASH  
0x5f64fe5f478e588950585901a8656edf81c2e0ccb57e060f694670dc73045e77

CONTRACT CALL

FROM ADDRESS 0xF953F5229D51B9B319EC03A6F0058ad6fB737D09 TO CONTRACT ADDRESS Migrations GAS USED 28835 VALUE 0

TX HASH  
0xdb93902219eb47c129c8c66b44a527352b2d3b3c6efd3d898dcec9943766e1dd

CONTRACT CREATION

FROM ADDRESS 0xF953F5229D51B9B319EC03A6F0058ad6fB737D09 CREATED CONTRACT ADDRESS 0xdb22725251D62cEe575c9298a9d13E36E5af2568 GAS USED 877314 VALUE 0

settings

requests  
t.js:29800  
App.js:76  
App.js:78  
App.js:44  
App.js:66  
App.js:56

0.000000000000000000  
Complete V  
0.01  
2

Free

0.000000000000000000  
Complete V  
0.01  
2

When pr  
In  
Downloa  
Feedback  
Feedback  
Deposit  
Work co  
Contract  
>>

Ganache

ACCOUNTS BLOCKS TRANSACTIONS CONTRACTS EVENTS LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK 235 GAS PRICE 20000000000 GAS LIMIT 6721975 HARDFORK MERGE NETWORK ID 5777 RPC SERVER HTTP://127.0.0.1:7545 MINING STATUS AUTOMINING WORKSPACE YASHBRO-123 SWITCH

TX HASH  
0x5408ebcbefb0f9a3c4e8b2bce8ac71806ec9919fff37d015c668049e6cbaff38

CONTRACT CALL

FROM ADDRESS 0xF953F5229D51B9B319EC03A6F0058ad6fB737D09 TO CONTRACT ADDRESS 0x9C9c099047bAc571592ED116CF424439B0c973bd GAS USED 51268 VALUE 0

TX HASH  
0x12651598711eb527746131f5ee01a0dc2a236982ffdc3920776ffae0e2891d24

CONTRACT CALL

FROM ADDRESS 0xF953F5229D51B9B319EC03A6F0058ad6fB737D09 TO CONTRACT ADDRESS 0x9C9c099047bAc571592ED116CF424439B0c973bd GAS USED 33005 VALUE 0

TX HASH  
0x2ed8f6e9aaa007fae56c85762394ea7bb9e937c408b3076bd9ad69b316e7c065

CONTRACT CALL

FROM ADDRESS 0xF953F5229D51B9B319EC03A6F0058ad6fB737D09 TO CONTRACT ADDRESS 0x9C9c099047bAc571592ED116CF424439B0c973bd GAS USED 27670 VALUE 1000

TX HASH  
0x37ea08bcf1ac83845c13e22fe9f6e4dd020649b396217aa1c6b7ac3bc3ac01c6

CONTRACT CALL

FROM ADDRESS 0xF953F5229D51B9B319EC03A6F0058ad6fB737D09 TO CONTRACT ADDRESS 0x9C9c099047bAc571592ED116CF424439B0c973bd GAS USED 31062 VALUE 0

settings

requests  
t.js:29800  
App.js:76  
App.js:78  
App.js:44  
App.js:66  
App.js:56

```
contracts > assignment.sol
4 contract FreelanceServiceManagement {

Node v20.11.1

myadw@INBOOK_X1_SLIM MINGW64 ~/OneDrive/Desktop/Truffle_Project
$ truffle test
Using network 'development'.

Compiling your contracts...
=====
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\assignment.sol
> Artifacts written to C:\Users\myadw\AppData\Local\Temp\test--13316-605FhlyT6DOP
> Compiled successfully using:
   - solc: 0.8.0+commit.c7d678e.Emscripten.clang
FreelanceServiceManagement contract deployed to: 0xc8C37243f0f1132ef625DA1a7D9c94BF12E5C775

Contract: FreelanceServiceManagement
  ✓ should set initial contract states correctly (114ms)
  ✓ allows the client to deposit the correct amount (58ms)
  ✓ does not allow depositing an incorrect amount (303ms)
  ✓ allows the freelancer to complete work and receive payment (118ms)
  ✓ allows contract renewal with a new amount by the client (227ms)
  ✓ allows the client to give feedback (75ms)

6 passing (2s)

myadw@INBOOK_X1_SLIM MINGW64 ~/OneDrive/Desktop/Truffle_Project
$
```

## 5. Conclusion

The **Smart Contract for Freelance Service Management** successfully addresses key issues in the freelance industry by:

- Automating payments.
- Enforcing agreed terms.
- Building trust through transparent feedback.

Testing showed that the contract could handle common scenarios like deposits, task completion, and contract renewal efficiently. This project demonstrates how blockchain can improve the reliability and fairness of freelance services, making the process seamless for both clients and freelancers.