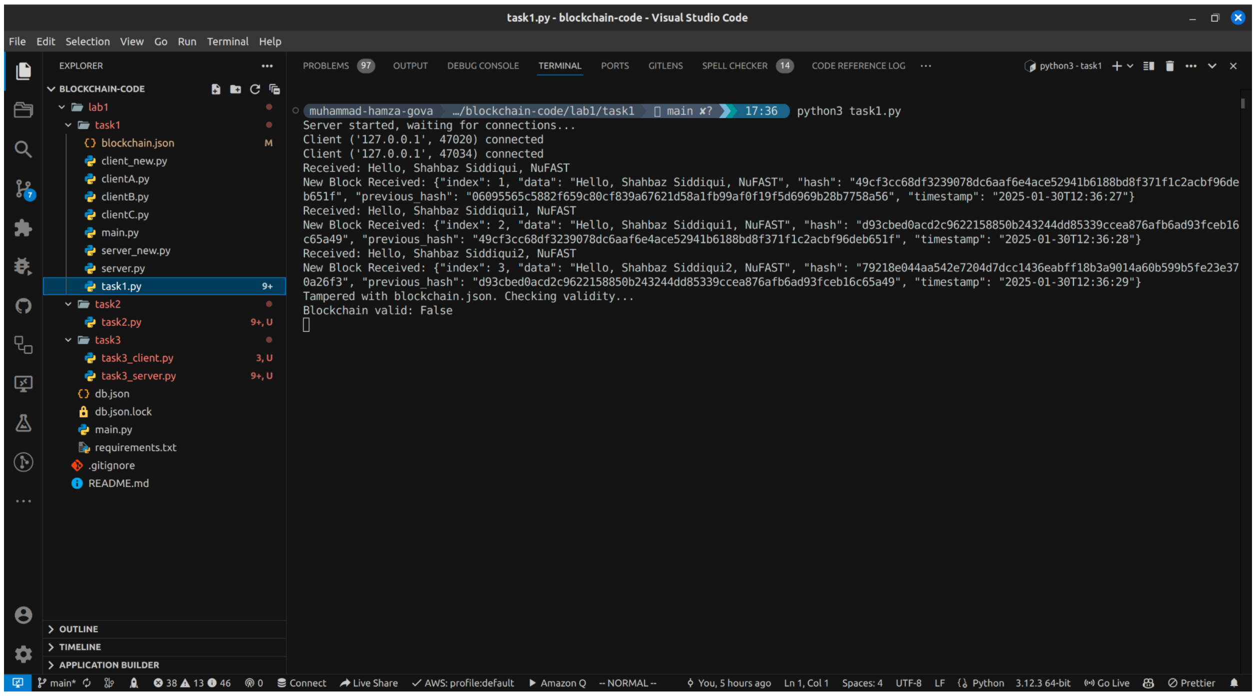
Blockchain & Cryptocurrency Lab 01
K21-4579 K21-4871

Task 1: Blockchain Implementation and Client-Server Communication

Objective

The objective of this task is to implement a simple blockchain and demonstrate its usage in a client-server communication setup. The blockchain will be used to ensure the integrity and immutability of messages exchanged between clients and the server.



```
muhammad-hamza-gova  …/blockchain-code/lab1/task1   main ✘?  17:36  python3 task1.py
Server started, waiting for connections...
Client ('127.0.0.1', 47020) connected
Client ('127.0.0.1', 47034) connected
Received: Hello, Shahbaz Siddiqui, NuFAST
New Block Received: {"index": 1, "data": "Hello, Shahbaz Siddiqui, NuFAST", "hash": "49cf3cc68df3239078dc6aaf6e4ace52941b6188bd8f371f1c2acbf96de
b651f", "previous_hash": "06095565c5882f659c80cf839a67621d58a1fb99af0f19f5d6969b28b7758a56", "timestamp": "2025-01-30T12:36:27"}
Received: Hello, Shahbaz Siddiqui1, NuFAST
New Block Received: {"index": 2, "data": "Hello, Shahbaz Siddiqui1, NuFAST", "hash": "d93cbed0acd2c9622158850b243244dd85339ccea876afb6ad93fceb16
c65a49", "previous_hash": "49cf3cc68df3239078dc6aaf6e4ace52941b6188bd8f371f1c2acbf96deb651f", "timestamp": "2025-01-30T12:36:28"}
Received: Hello, Shahbaz Siddiqui2, NuFAST
New Block Received: {"index": 3, "data": "Hello, Shahbaz Siddiqui2, NuFAST", "hash": "79218e044aa542e7204d7dcc1436eabff18b3a9014a60b599b5fe23e37
0a26f3", "previous_hash": "d93cbed0acd2c9622158850b243244dd85339ccea876afb6ad93fceb16c65a49", "timestamp": "2025-01-30T12:36:29"}
Tampered with blockchain.json. Checking validity...
Blockchain valid: False
```

Results

1. The server successfully starts and waits for client connections.
2. Clients connect to the server and send messages.
3. The server creates a new block for each message and broadcasts it to all connected clients.
4. The blockchain's integrity is verified, and any tampering is detected.

Conclusion

This task demonstrates the implementation of a simple blockchain and its usage in a client-server communication setup. The blockchain ensures the integrity and immutability of messages exchanged between clients and the server. The tamper test verifies the effectiveness of the blockchain in detecting any unauthorized modifications.

Task 2: Secure Message Transmission with HMAC and Encryption

Objective

The objective of this task is to implement secure message transmission between two clients using HMAC for message integrity and Fernet encryption for confidentiality.

Implementation Details

Key Components

- Fernet Key: A symmetric key used for encrypting and decrypting messages.
- HMAC Key: A shared secret key used for creating and verifying HMAC signatures.
- Processed Transactions: A set to store processed transactions and prevent double spending.

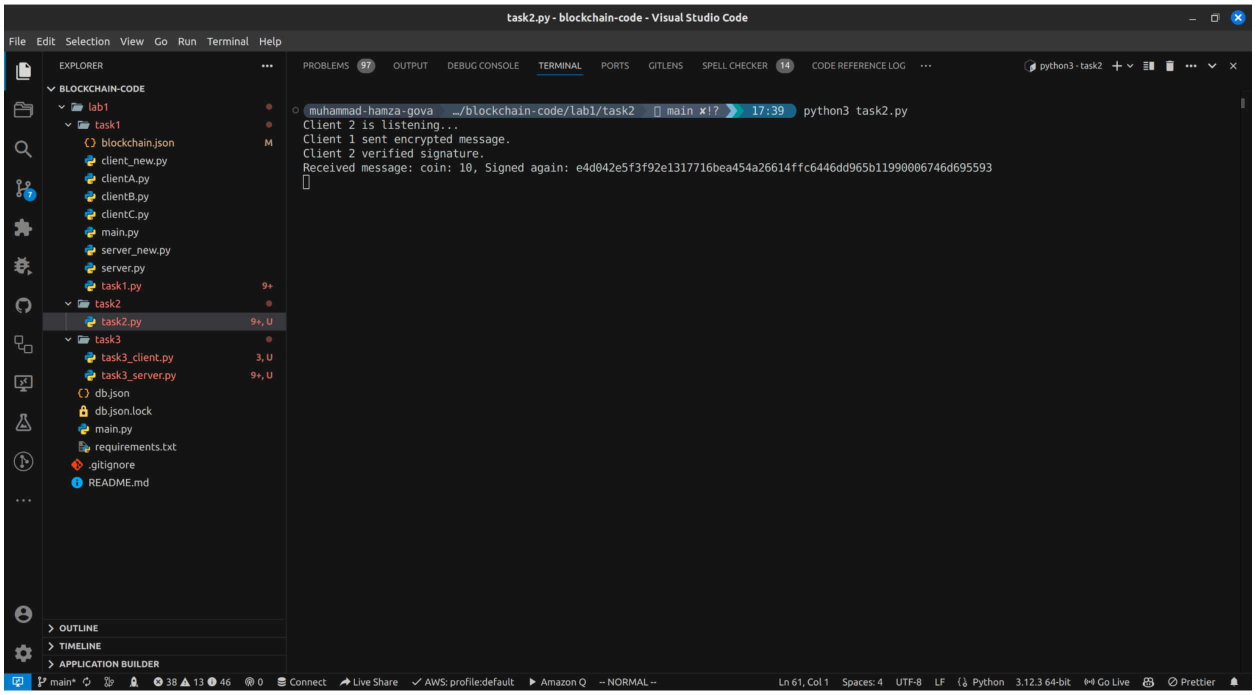
Client 1

- Creates a message.
- Signs the message using HMAC.
- Encrypts the signed message using Fernet.
- Sends the encrypted message to Client 2.

Client 2

- Listens for incoming connections.
- Receives the encrypted message from Client 1.
- Decrypts the message using Fernet.
- Verifies the HMAC signature.
- Checks for double spending.
- Signs the message again if it is valid.

Blockchain & Cryptocurrency Lab 01
K21-4579   K21-4871



```
muhammad-hamza-gova  …/blockchain-code/lab1/task2   main ✘!?  17:39  python3 task2.py
Client 2 is listening...
Client 1 sent encrypted message.
Client 2 verified signature.
Received message: coin: 10, Signed again: e4d042e5f3f92e1317716bea454a26614ffc6446dd965b11990006746d695593
```

Results

1. Client 1 successfully creates, signs, encrypts, and sends a message to Client 2.
2. Client 2 receives the encrypted message, decrypts it, verifies the signature, and checks for double spending.
3. Client 2 signs the message again if it is valid and displays the message along with the new signature.
4. Double spending is detected and prevented by Client 2.

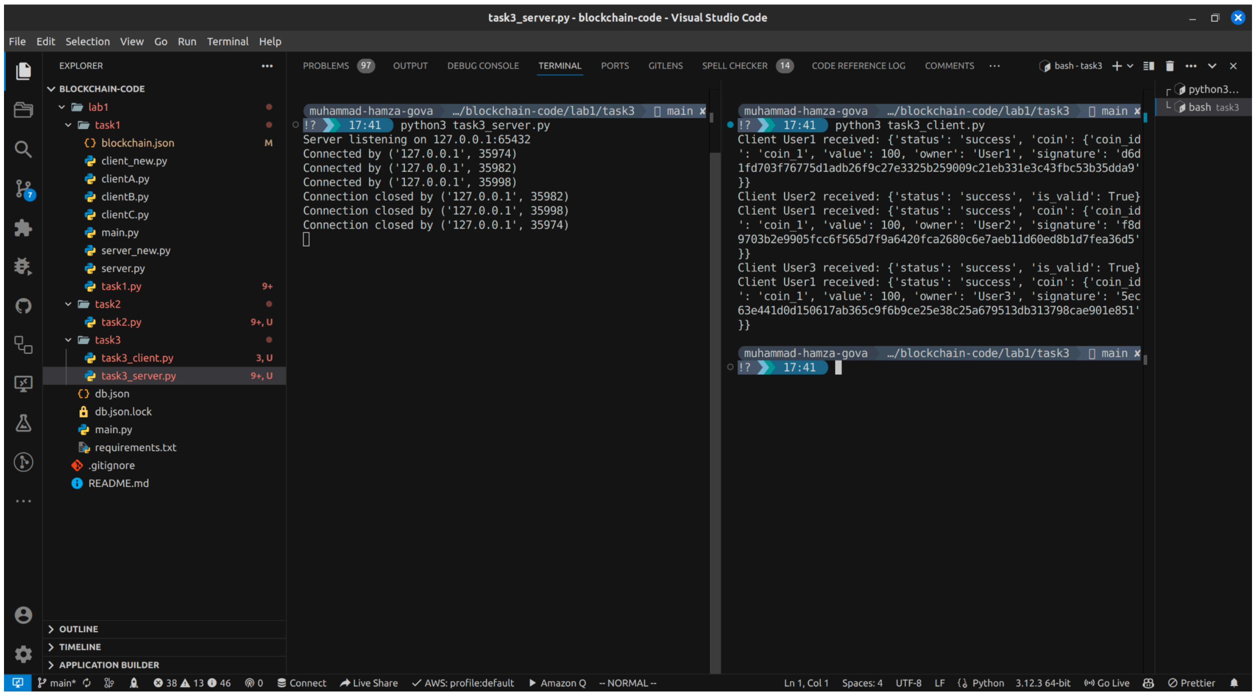
Conclusion

This task demonstrates the implementation of secure message transmission using HMAC for integrity and Fernet encryption for confidentiality. The system effectively detects and prevents double spending, ensuring the integrity and security of the transmitted messages.

Task 3: GoofyCoin Implementation and Client-Server Communication

Objective

The objective of this task is to implement a simple cryptocurrency called GoofyCoin and demonstrate its usage in a client-server communication setup. The server will handle the creation, transfer, and verification of GoofyCoins, while clients will perform actions such as creating coins, sending coins, and verifying coins.



```
muhammad-hamza-gova  …/blockchain-code/lab1/task3   main ✘
!?  17:41  python3 task3_server.py
Server listening on 127.0.0.1:65432
Connected by ('127.0.0.1', 35974)
Connected by ('127.0.0.1', 35982)
Connected by ('127.0.0.1', 35998)
Connection closed by ('127.0.0.1', 35982)
Connection closed by ('127.0.0.1', 35998)
Connection closed by ('127.0.0.1', 35974)
```

```
muhammad-hamza-gova  …/blockchain-code/lab1/task3   main ✘
!?  17:41  python3 task3_client.py
Client User1 received: {'status': 'success', 'coin': {'coin_id
': 'coin_1', 'value': 100, 'owner': 'User1', 'signature': 'd6d
1fd703f76775d1adb26f9c27e3325b259009c21eb331e3c43fbc53b35dda9'
}}
Client User2 received: {'status': 'success', 'is_valid': True}
Client User1 received: {'status': 'success', 'coin': {'coin_id
': 'coin_1', 'value': 100, 'owner': 'User2', 'signature': 'f8d
9703b2e9905fcc6f565d7f9a6420fca2680c6e7aeb11d60ed8b1d7fea36d5'
}}
Client User3 received: {'status': 'success', 'is_valid': True}
Client User1 received: {'status': 'success', 'coin': {'coin_id
': 'coin_1', 'value': 100, 'owner': 'User3', 'signature': '5ec
63e441d0d150617ab365c9f6b9ce25e38c25a679513db313798cae901e851'
}}

muhammad-hamza-gova  …/blockchain-code/lab1/task3   main ✘
!?  17:41
```

Results

1. The server successfully starts and listens for client connections.
2. Client 1 creates a GoofyCoin, sends it to User2, and then sends it to User3.
3. Client 2 verifies the coin's validity after it is sent to User2.
4. Client 3 verifies the coin's validity after it is sent to User3.
5. The server handles the creation, transfer, and verification of GoofyCoins, ensuring the integrity of the coin's signature.

Conclusion

This task demonstrates the implementation of a simple cryptocurrency called GoofyCoin and its usage in a client-server communication setup. The server effectively handles the creation, transfer, and verification of GoofyCoins, ensuring the integrity and security of the transactions. The clients perform actions such as creating coins, sending coins, and verifying coins, demonstrating the functionality of the GoofyCoin system.