DateCoin

Team 1: Michael De Paula, Jeff Berger, Julian Martinez, Manny Russell, Emmanuel Henao & Mitchel Voloshin

Motivation & Summary

Goal:

We wanted to create a Decentralized Dating Application that provides users with a better and more secure experience than traditional dating platforms. Many dating apps today are filled with bots, subpar filtering and limited profile information. By offering a more secure online dating experience we are hoping users will be more comfortable in sharing additional information which will in turn lead to better filtering for matching.

Method:
Create a smart contract on the Ethereum network.

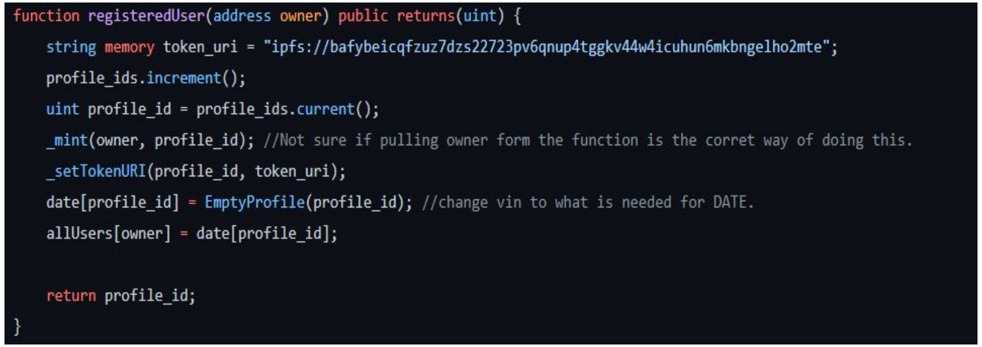
Smart Contract Functionality

Register User Function:

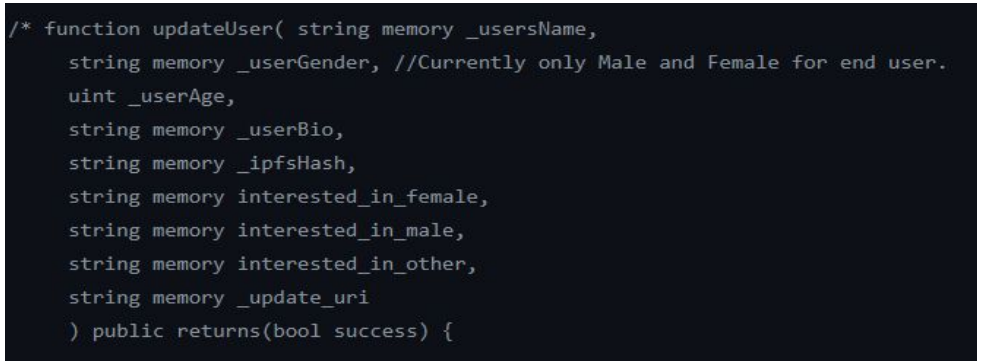
- ❑ Mints Token
- ❑ Combines token with User Profile

Update User Function:

- ❑ Name
- ❑ Bio
- ❑ Age
- ❑ Location
- ❑ Interests
- ❑ Update Picture



```
function registeredUser(address owner) public returns(uint) {
    string memory token_uri = "ipfs://bafybeicqfzuz7dzs22723pv6qnup4tggkv44w4icuhun6mkbngelho2mte";
    profile_ids.increment();
    uint profile_id = profile_ids.current();
    _mint(owner, profile_id); //Not sure if pulling owner form the function is the corret way of doing this.
    _setTokenURI(profile_id, token_uri);
    date[profile_id] = EmptyProfile(profile_id); //change vin to what is needed for DATE.
    allUsers[owner] = date[profile_id];

    return profile_id;
}
```



```
/* function updateUser( string memory _usersName,
     string memory _userGender, //Currently only Male and Female for end user.
     uint _userAge,
     string memory _userBio,
     string memory _ipfsHash,
     string memory interested_in_female,
     string memory interested_in_male,
     string memory interested_in_other,
     string memory _update_uri
     ) public returns(bool success) {
```

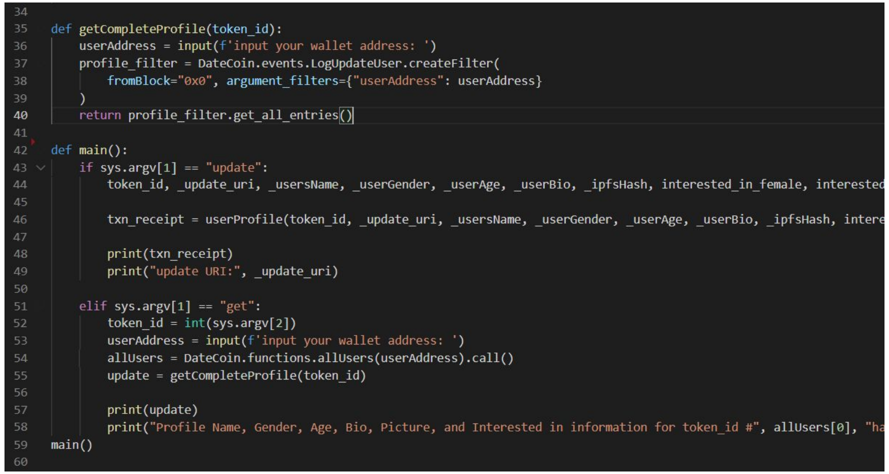
Smart Contract Functionality (Continued)

Profile Filter Function:

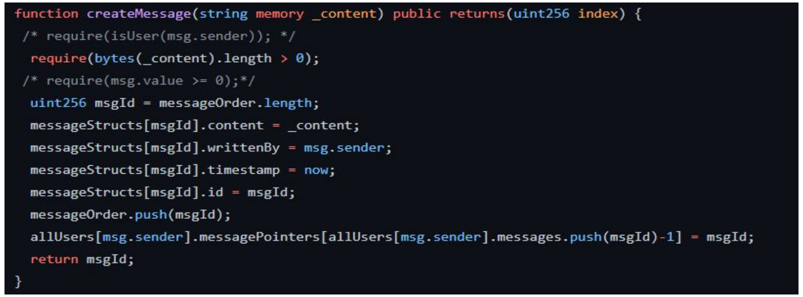
- ❑ LogUpdateUser event from solidity
- ❑ Creates Filter of Token Ids.

Messaging Function:

- ❑ Message connections
- ❑ Like comments
- ❑ Unlike comments



```python

def getCompleteProfile(token_id):
    userAddress = input(f'input your wallet address: ')
    profile_filter = DateCoin.events.LogUpdateUser.createFilter(
        fromBlock="0x0", argument_filters={"userAddress": userAddress}
    )
    return profile_filter.get_all_entries()

def main():
    if sys.argv[1] == "update":
        token_id, _update_uri, _usersName, _userGender, _userAge, _userBio, _ipfsHash, interested_in_female, interested

        txn_receipt = userProfile(token_id, _update_uri, _usersName, _userGender, _userAge, _userBio, _ipfsHash, intere

        print(txn_receipt)
        print("update URI:", _update_uri)

    elif sys.argv[1] == "get":
        token_id = int(sys.argv[2])
        userAddress = input(f'input your wallet address: ')
        allUsers = DateCoin.functions.allUsers(userAddress).call()
        update = getCompleteProfile(token_id)

        print(update)
        print("Profile Name, Gender, Age, Bio, Picture, and Interested in information for token_id #", allUsers[0], "ha
main()

```

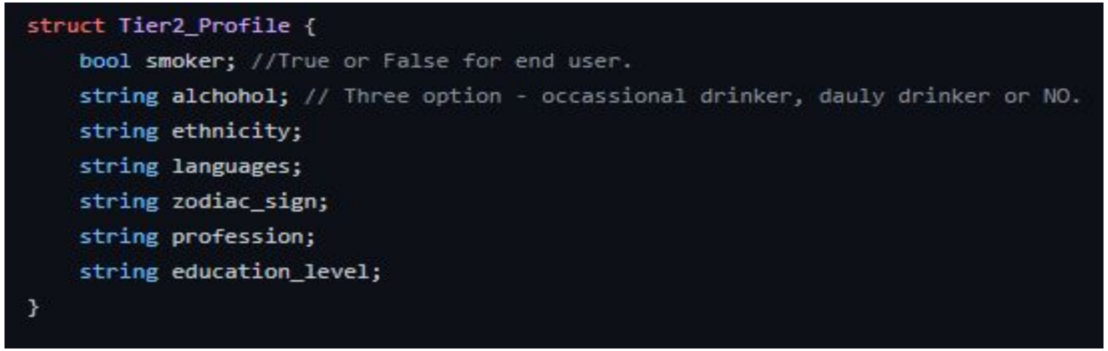


```solidity
function createMessage(string memory _content) public returns(uint256 index) {
 /* require(isUser(msg.sender)); */
  require(bytes(_content).length > 0);
 /* require(msg.value >= 0);*/
  uint256 msgId = messageOrder.length;
  messageStructs[msgId].content = _content;
  messageStructs[msgId].writtenBy = msg.sender;
  messageStructs[msgId].timestamp = now;
  messageStructs[msgId].id = msgId;
  messageOrder.push(msgId);
  allUsers[msg.sender].messagePointers[allUsers[msg.sender].messages.push(msgId)-1] = msgId;
  return msgId;
}
```

Profile Tiers for Filtering

Profile Tier 2

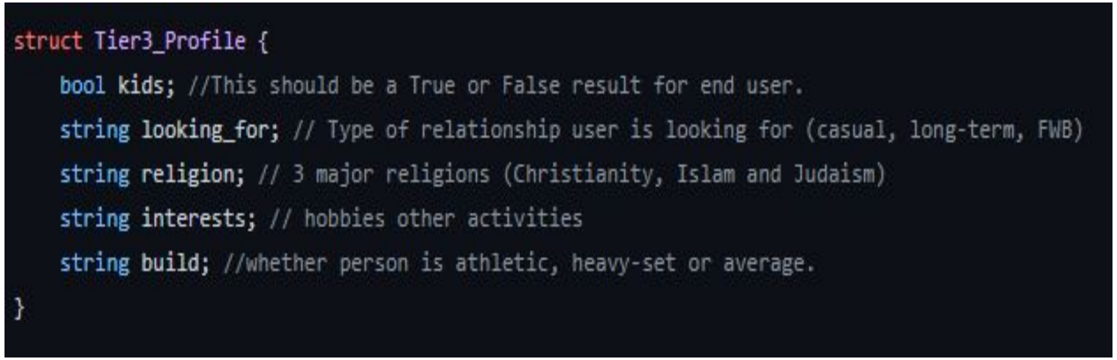
- ❏ Smoker
- ❏ Alcohol
- ❏ Ethnicity
- ❏ Languages
- ❏ Zodiac Sign
- ❏ Profession
- ❏ Education level



```
struct Tier2_Profile {
    bool smoker; //True or False for end user.
    string alchohol; // Three option - occassional drinker, dauly drinker or NO.
    string ethnicity;
    string languages;
    string zodiac_sign;
    string profession;
    string education_level;
}
```

Profile Tier 3

- ❏ Kids
- ❏ Looking for
- ❏ Religion
- ❏ Interests
- ❏ Build



```
struct Tier3_Profile {
    bool kids; //This should be a True or False result for end user.
    string looking_for; // Type of relationship user is looking for (casual, long-term, FWB)
    string religion; // 3 major religions (Christianity, Islam and Judaism)
    string interests; // hobbies other activities
    string build; //whether person is athletic, heavy-set or average.
}
```

Tokens & Dependencies

- ❑ ERC 721
- ❑ VS Code
- ❑ SYS
- ❑ Pin Json to IPFS
- ❑ Init Contract
- ❑ Web3
- ❑ Python
- ❑ Pinata

Obstacles Faced

- ❏ Sending private messages.
- ❏ Providing filtering capabilities.
- ❏ Limited time developing a Front End Software.
- ❏ Keeping contract under 24 KB.
- ❏ Gas Fees.
- ❏ Length of code.

Conclusion

In Conclusion, we were able to compile and deploy the solidity contracts. We were able to register a user, update a user, and activate a user using the deployment feature within solidity.

In the future, we would like to build a full front-end GUI for the end user. We tried building an interactive prompt from python for use within terminal, this was partially successful as additional coding was required to connect other functions from solidity to Python.

We also would like to have a full deployment of tokens for every tier of profiles that are available to users within a public blockchain network.