TRƯỜNG ĐẠI HỌC QUỐC GIA TP – HỒ CHÍ MINH
ĐẠI HỌC KHOA HỌC TỰ NHIÊN

BÁO CÁO NHÓM

ĐỒ ÁN: XÂY DỰNG GAMEFI TRÊN MẠNG BLOCKCHAIN

GIẢNG VIÊN:	Nguyễn Đình Thúc	
	Nguyễn Văn Quang Huy	
	Ngô Đình Hy	
LỚP:	20CNTThuc	
SINH VIÊN:	Hoàng Hữu Minh An	20127102
	Trần Hoàng Minh Quang	20127299

Ho Chi Minh, 24-04-2024

Contents

I. Smart Contract:

1. Floppy contract:

Hợp đồng thông minh trên có chức năng tạo token có tên là "Floppy" – "FLP". Dưới đây là một số chức năng chính của hợp đồng:

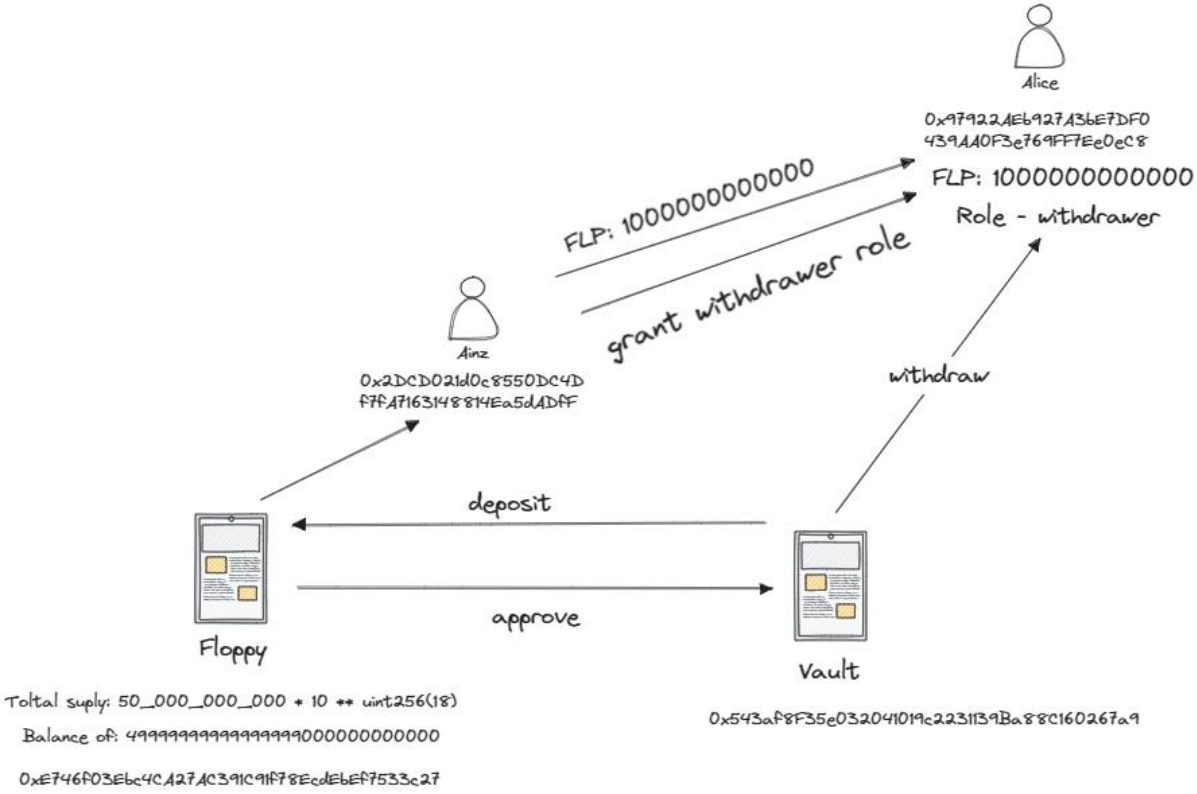
- Cung cấp token ban đầu: Khi hợp đồng được triển khai, nó sẽ tự động tạo ra một lượng token ban đầu và gán chúng cho tài khoản triển khai hợp đồng. Số lượng token được cung cấp ban đầu là một giá trị cố định và được xác định bởi biến cap.
- Giới hạn cung lượng token: Hợp đồng sẽ kiểm tra và đảm bảo rằng tổng lượng cung lượng token trong hợp đồng không vượt quá giới hạn được quy định bởi biến cap. Nếu cung lượng token được tạo ra mới vượt quá giới hạn này, hợp đồng sẽ từ chối yêu cầu tạo thêm token.
- Tạo thêm token: Chủ sở hữu hợp đồng có khả năng tạo thêm token bằng cách gọi hàm mint. Tuy nhiên, trước khi tạo thêm, hợp đồng sẽ kiểm tra xem việc tạo thêm có làm vượt quá giới hạn cung lượng token được phép hay không. Nếu việc tạo thêm không vượt quá giới hạn, hợp đồng sẽ tạo ra số lượng token được chỉ định và gán chúng cho địa chỉ được chỉ định.
- Chức năng Burnable: Hợp đồng sử dụng thư viện OpenZeppelin để kích hoạt chức năng Burnable, cho phép chủ sở hữu token đốt (burn) một số lượng token đã được tạo ra, giảm lượng cung lượng token tổng thể.

2. Vault contract:

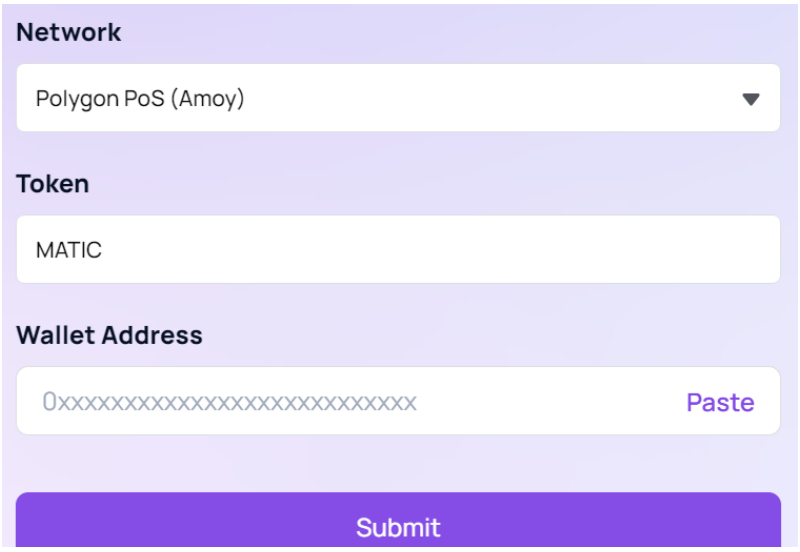
Hợp đồng thông minh này là một kho lưu trữ (vault) token có chức năng cho phép người dùng tương tác từ kho lưu trữ đó. Dưới đây là một số chức năng chính của hợp đồng:

- Thiết lập token: Chủ sở hữu của hợp đồng có thể thiết lập loại token mà kho lưu trữ sẽ sử dụng bằng cách gọi hàm setToken. Điều này cho phép hợp đồng biết được loại token mà nó sẽ lưu trữ và xử lý.
- Cho phép rút tiền: Chủ sở hữu cũng có khả năng bật hoặc tắt chức năng rút tiền từ kho lưu trữ thông qua hàm setWithdrawEnable. Khi chức năng này được tắt, người dùng sẽ không thể rút tiền từ kho lưu trữ.
- Giới hạn lượng rút tối đa: Chủ sở hữu có thể thiết lập một giới hạn về lượng tiền tối đa mà mỗi lần rút được phép thông qua hàm setMaxWithdrawAmount. Điều này giúp ngăn người dùng rút ra một lượng quá lớn và gây ảnh hưởng đến hoạt động của hợp đồng.
- Rút tiền: Người dùng có quyền rút tiền từ kho lưu trữ thông qua hàm withdraw. Tuy nhiên, họ chỉ có thể thực hiện điều này nếu họ được cấp quyền làm điều đó. Hàm này kiểm tra xem người gọi có phải là chủ sở hữu của hợp đồng hoặc có quyền được cấp làm điều này không.
- Nạp tiền: Người dùng cũng có thể nạp tiền vào kho lưu trữ thông qua hàm deposit. Họ chỉ có thể nạp tiền từ địa chỉ không phải là hợp đồng (không phải là một hợp đồng thông minh) và số dư của họ phải đủ để thực hiện giao dịch này.

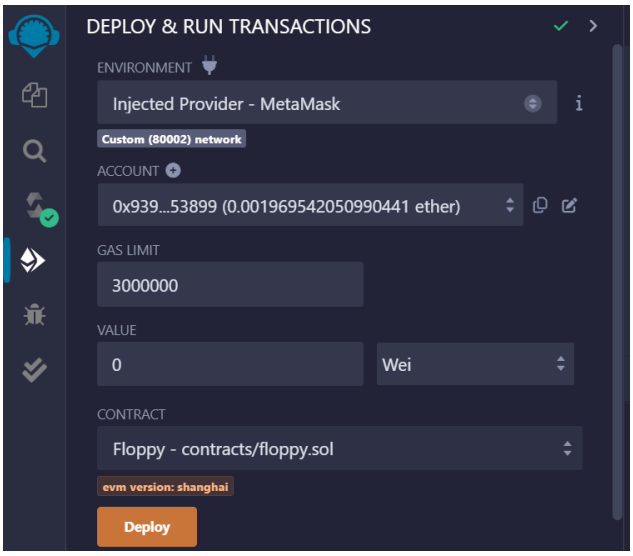
- Rút tiền khẩn cấp: Chủ sở hữu có khả năng rút tất cả số token có trong kho lưu trữ về địa chỉ của họ thông qua hàm emergencyWithdraw. Điều này chỉ nên được thực hiện trong trường hợp khẩn cấp.



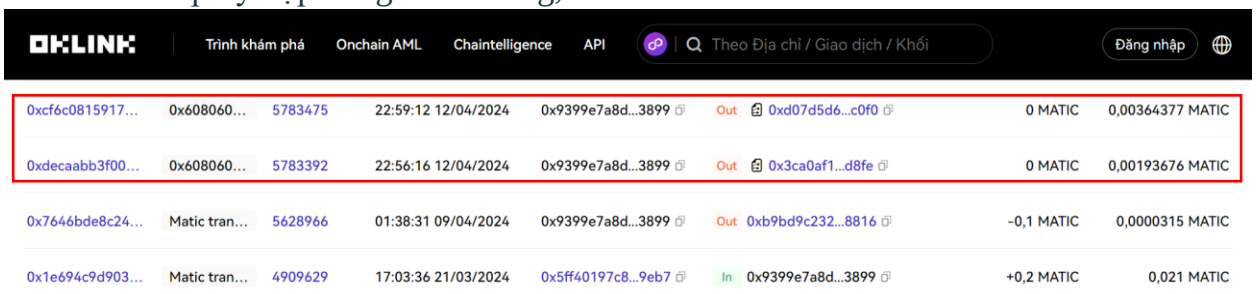
3. Deploy và Verify hợp đồng thông minh:
   Sử dụng **Polygon Faucet** chuyển test token MATIC đến ví để làm phí giao dịch, trên mạng Polygon Amoy:



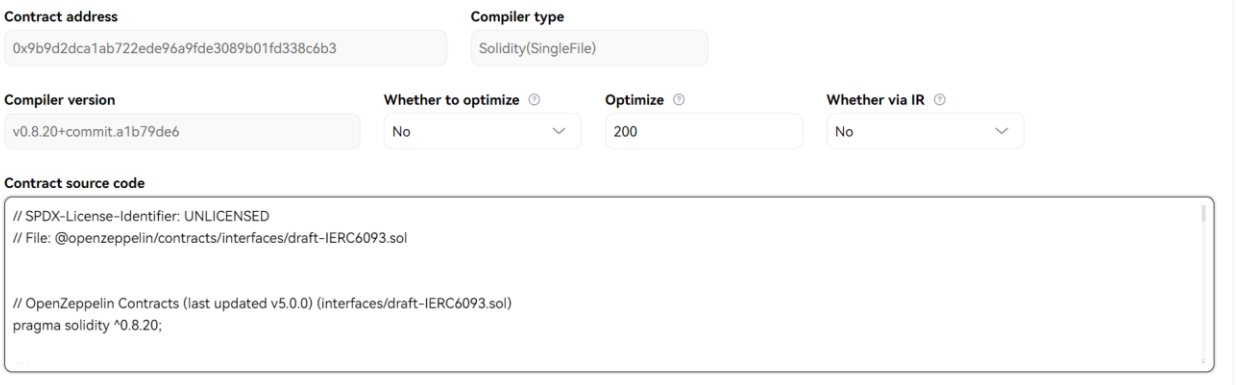
Sử dụng **Remix IDE** để deploy, kèm với đó wallet address để quản lý hợp đồng:



Sau khi đã deploy hợp đồng thành công, có thể theo dõi trên **OKLINK**:



Để Verify, ta tiến hành flatten file.sol, sau đó chọn mục Verify Smart Contract, và paste file_flattened.sol vào:



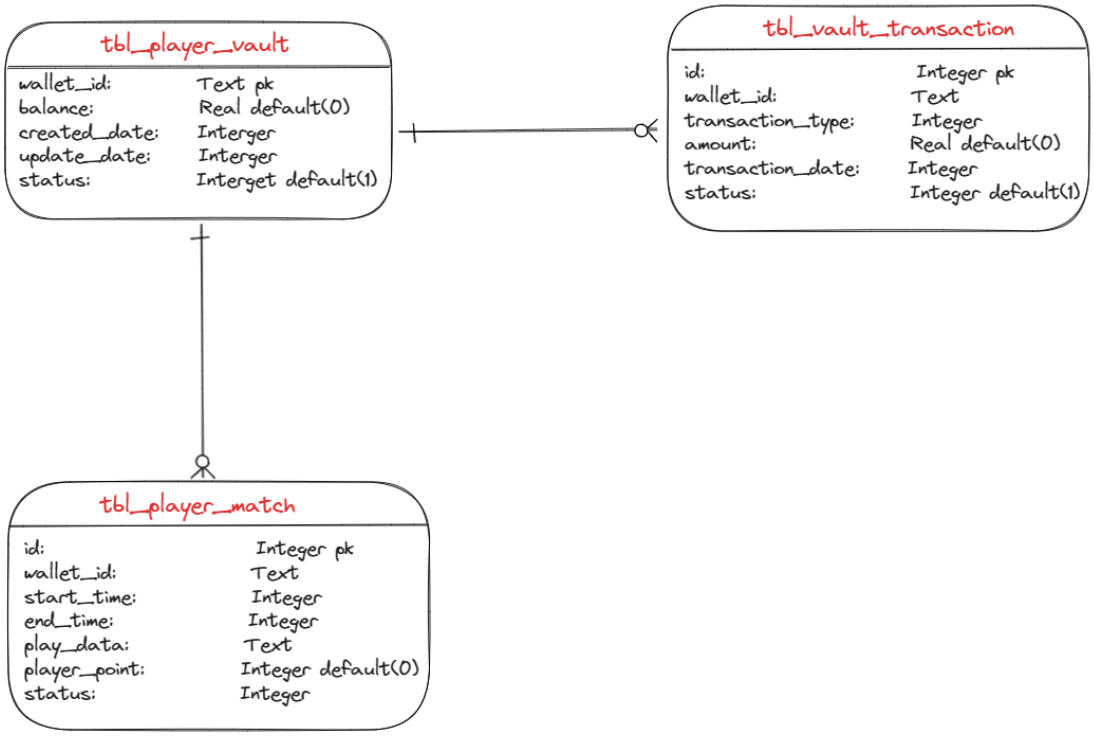
II. API:

1. Môi trường xây dựng và thư viện:
   a) Môi trường: Nodejs v18.18.0
   b) Thư viện:

- Express : v4.19.1 – Framework xây dựng ứng dụng web và API
- Web3: v1.8.0 – Là một thư viện cho phép tương tác với ethereum cục bộ hoặc từ xa thông qua HTTP, IPC hoặc WebSocket.
- Sqlite3: v5.1.7 – Thư viện để tương tác database sqlite3

2. Tạo model:
   a) Database:

Sử dụng SQLite hệ quả trị cơ sở dữ liệu (DBMS) quan hệ tương tự như Mysql,… Đặc điểm nổi bật của SQLite so với các DBMS khác là gọn, nhẹ, đơn giản, đặt biệt không cần mô hình server-client.



- tbl_player_vault: lưu thông tin người chơi
- tbl_player_match: lưu thông tin một game của người chơi
- tbl_vault_transaction: lưu thông tin giao dịch của người chơi

Người chơi (tbl_vault_transaction) có thể có nhiều giao dịch (tbl_vault_transaction) hoặc không. Người chơi có thể chơi nhiều lượt chơi hoặc không.

b) Nội dung migration:

Tạo một file theo đường dẫn sau `server/src/model/MigrationContext.js`. Trong file migration này, chùng ta tiến hành tạo 3 bảng dữ liệu.

- tbl_player_vault_ddl: lưu thông tin ví của của chơi
  - wallet_id (TEXT) là khóa chính
  - balance (REAL) mặc định là 0
  - created_date (INTERGER)
  - updated_date (INTERGER)
  - status (INTERGER) mặc định là 1

- tbl_vault_transaction : Lưu thông tin thực hiện hành động deposit hoặc withdraw từ ví
  - id (INTERGER) khóa chính tự tăng
  - wallet_id (TEXT)
  - transaction_type (INTERGER)
  - amount (REAL) mặc định là 0
  - transaction_date (INTERGER)
  - status (INTERGER) mặc định là 1
  - transaction_id (TEXT)

- tbl_player_match: lưu thông tin lượt chơi của người chơi
  - id (INTERGER) khóa chính tự tăng
  - wallet_id (TEXT)
  - start_time (INTERGER)
  - end_time (INTERGER)
  - play_data (TEXT)
  - player_point (INTERGER)
  - status (INTERGER)

Sau khi chạy server, nên chạy file MigrationContext.js bằng lệnh node `src/model/MigrationContext.js`. Sau khi chạy thành công ta thấy một file database được tạo thành `./ FloppyBird.db`

c) Tạo các DAO(Data access objects):

- FloppyBirdDAO:
  - Khởi tạo: tiến hành kết nối tới database.
  - AddPlayerVault: thêm thông tin ví người dùng vào database với balance là 10.
  - GetPlayerBalance: lấy balance của người chơi theo địa chỉ ví.
  - GetTopPlayer: lấy top danh sách thông tin lượt chơi của người chơi.
  - AddPlayerBalanceTransaction: thêm thông tin transaction của người chơi vào tbl_vault_transaction.
  - AddPlayerBalance: thêm ticket vào balance ví của người chơi.

- UpdateTransaction: cập nhật thông tin vault transation_id của người chơi.
- WithdrawPlayerBalance: rút ticket từ balnace ví của người chơi
- StartPlayerMatch: khởi tạo thông tin lượt chơi khi bắt đầu chơi
- EndPlayerMatch: update thông tin người chơi sau khi kết thúc lượt chơi

- SmartContractDAO:
  - Khởi tạo: thiết lập kết với với các smart contract bằng web3 với địa chỉ người chơi.
  - getBalance: sử dụng Floppy smart contract lấy số dư FLP của một địa chỉ ví
  - withdraw : sử dụng Vault smart contract rút tiền từ một địa chỉ

3. Các endpoint:
   Server sử dụng địa chỉ local với base url: http://localhost:8000/v1
   Sử dụng **AIChemy** để kết nối với web3.

a) GET – ` getTicketBalance`:

- Query:
  - Address: địa chỉ ví
- Lấy số dư ticket của một địa chỉ. Nếu người chơi chưa tồn tại thì tạo.
- response:

```
{ “code”: 0,
  “data”: {
    “balances”: 10
  },
  “message”: “Success”}
```

b) GET – ` /getBalance`

- Query:
  - Address: địa chỉ ví
- Lấy số dư của một địa chỉ.
- response:

```
{ “code”: 0,
  “data”: {
    “balances”: 0
  },
  “message”: “Success” }
```

c) GET – ` /startMatch`

- Query:

- Address: địa chỉ ví
- Tạo lượt chơi, mỗi lần chơi tốn 1 ticket.
- response:

{ “code”: 0,

“data”: {

“Id”: 1

},

“message”: “Success” }

d) POST – `/endMatch`
- Query:
  - Address: địa chỉ ví
- Body

{
“address”: địa chỉ ví,
“id”: id lượt chơi khi start game,
“point”: số diểm ghi được,
“matchData”: cấu hình game.
}

- Kết thúc lượt chơi, cập nhật lại ticket người chơi nhận được
- response:

{ “code”: 0,

“data”: {

“result”: 105

},

“message”: “Success” }

e) GET – `/getTop`
- Lấy top người chơi cao điểm nhất.
- response:

{ “code”: 0,

“data”: {

“result”: [

{
“wallet_id”: “0x97922aeb927a3be7df0439aa0f3e769ff7ee0ec8”,
“point”: 100,
“start_time”: 1712159203,
“end_time”: 1712159500
}]
},
“message”: “Success” }

f) POST – `/deposit`
- Body
{
“address”: địa chỉ ví,
“ amount”:số lượng gửi,
“ transaction_id”: transaction hash,
}
- Nạp thêm ticket vào số dư của người chơi

g) POST – `/withdraw`
- Body
{
“address”: địa chỉ ví,
“ amount”:số lượng rút,
}
- Rút ticket về lại ví người chơi.

4. Thông tin lưu trữ trên database (SQLite):
   - Thông tin user:

	wallet id	balance	created date	update date	status
1	0x1aa450dfaea7016806dd4682f3a1279dc262632f	10	1712489993	NULL	1
2	0x1aa380dfaea7016806dd4682f3a1279dc262632f	209	1712734867	NULL	1

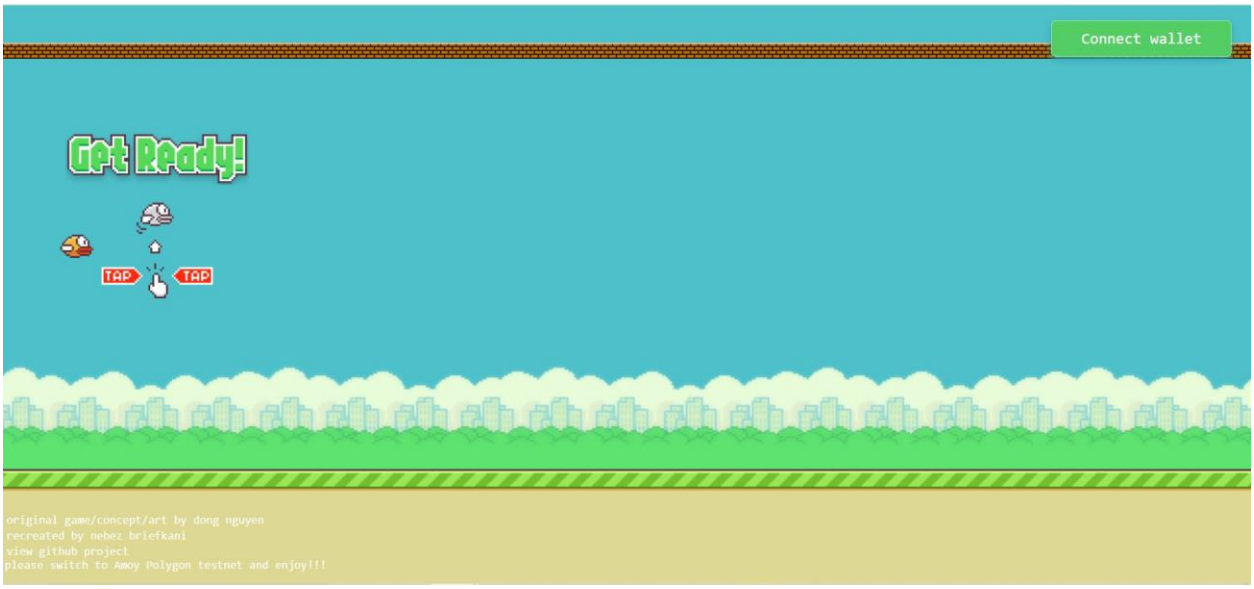
   - Thông tin lịch sử chơi:

	id	wallet id	start time	end time	play data	player po	status
1	1	0x1aa380dfaea7016806dd4682f3a1279dc262632f	1712435985	1712436175	[]	100	2

- Thông tin nạp/rút:

	id	wallet id	transaction type	amount	transaction date	status	transaction id
1	1	0x1aa380dfaea7016806dd4682f3a1279dc262632f	2	5	1712435985	1	NULL
2	2	0x1aa380dfaea7016806dd4682f3a1279dc262632f	1	100	1712436175	1	NULL
3	3	0x1aa380dfaea7016806dd4682f3a1279dc262632f	1	200	1712436454	1	100
4	4	0x1aa380dfaea7016806dd4682f3a1279dc262632f	2	50	1712436514	1	NULL
5	5	0x1aa380dfaea7016806dd4682f3a1279dc262632f	2	50	1712436611	1	NULL

III. Floopy Bird:



Màn hình khi bắt đầu khởi động client bằng live server

1. Kịch bản game:

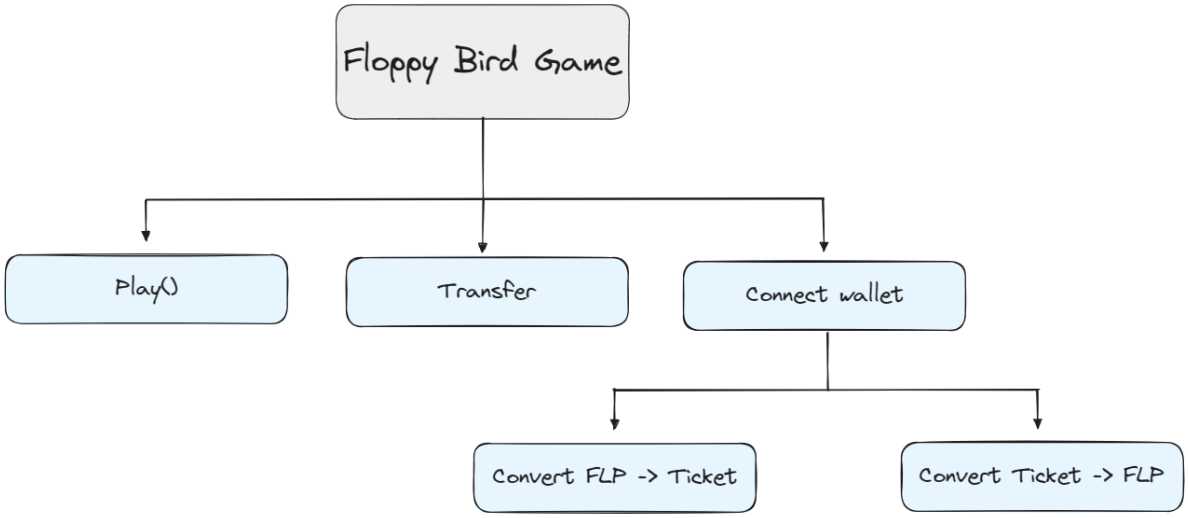
Trước khi bắt đầu chơi, người chơi phải kết nối vào ví của mình bằng meta mask hoặc phương thức khác.

Sau khi kết nối thành công, hiện lên màn hình nhưng thông tin của ví như là: address, MATIC, FLP, Ticket.

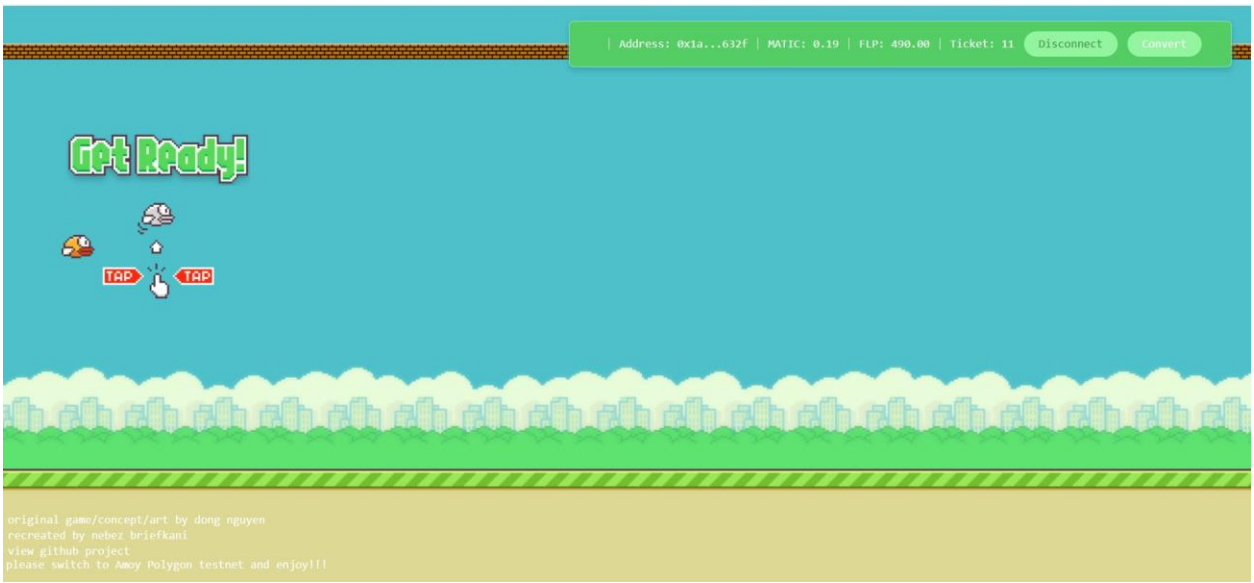
Người dùng nhấn vào màn hình để bắt đầu chơi game. Điều khiển Floppy Bird bằng cách click chuột vào màn hình để có thể bay lên và giúp Floppy Bird qua khỏi Pipe (chướng ngại vật). Mỗi lần vượt qua một chướng ngại vật người chơi sẽ được tính là một điểm, mỗi điểm là một ticket. Số ticket người chơi sẽ được cập nhật vào database.

2. Chức năng:

2.1. Cây chức năng

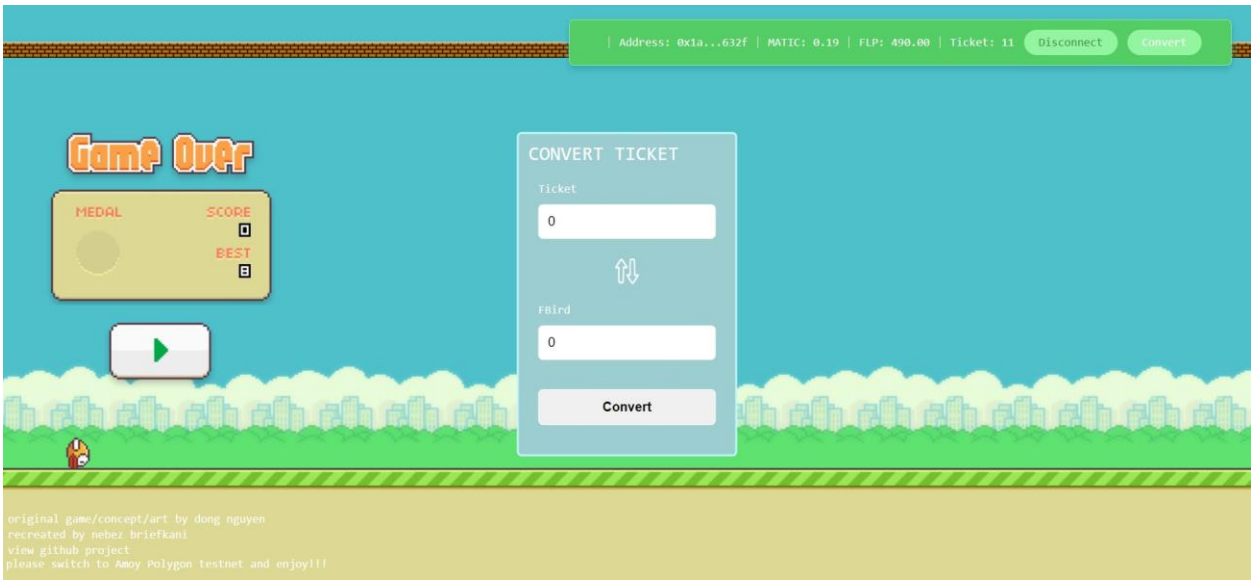


2.2. Kết nối với ví



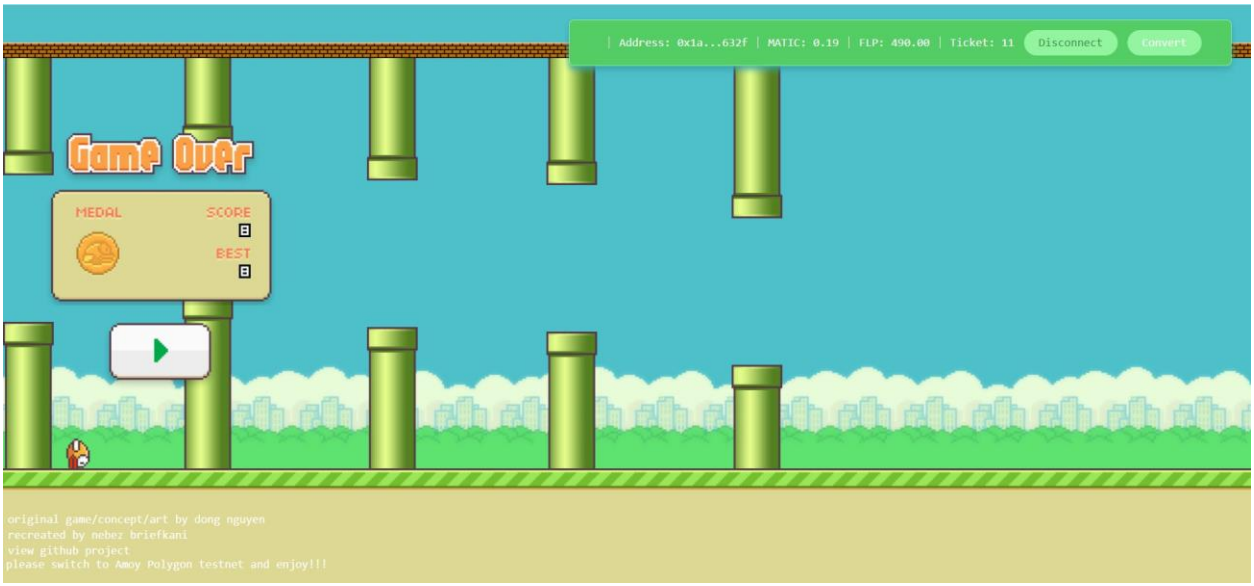
Sử dụng thư viện web3 để kết nối tới ví và gọi API getTicketBalance(address) để lấy thông tin ticket hiện tại.

2.3. Convert FLP



Người dùng có thể nhập số ticket của mình muốn đổi thành FLP token. Sau đó nhấn convert, để tiến hành convert sang FLP token. Người chơi muốn thay đổi từ Ticket sang FLP token chỉ cần nhấn vào button mũi tên, sau đó thực hiện nhập số lượng mình muốn quy đổi và nhấn Convert.

2.4. Play()



Khi Game Over, người chơi sẽ nhận được số ticket mà mình chơi được. Đề khuyến khích người chơi, với các số điểm tương ứng người chơi sẽ nhận được một medal. Số điểm được gọi API và lưu ở server.

IV. Link github:

Link github: floppy-bird-smartcontract: Build a Game Fi – Solidity, Smart Contracts #P2 #HCMUS #doanmahoa (github.com)

V. Tham khảo:

[1]: ERC20 – OpenZeppelin Docs

[2]: Hardhat's tutorial for beginners | Ethereum development environment for professionals by Nomic Foundation

[3]: jayden-dang/01-blockchain-basic: The Curriculums (github.com)

[4]: nebez/floppybird: in case you missed the hype (github.com)

[5]: web3.js – Ethereum JavaScript API — web3.js 1.0.0 documentation (web3js.readthedocs.io)

Cảm ơn thầy/cô và các bạn đã xem