Coding Challenge - Backend Engineer - Blockchain event stream processor

The Task

Your task is to write a console app in C# that receives some subset of transactions, and processes them in such a way that enables the program to answer questions about NFT ownership.

Your program must execute only a single command each time it is run, and must persist state between runs.

Messages

The messages your program must handle are:

Mint:

```
{
  "Type": "Mint",
  "TokenId": string,
  "Address": string
}
```

A mint transaction creates a new token in the wallet with the provided address

Burn:

```
{
  "Type": "Burn",
  "TokenId": string
}
```

A burn transaction destroys the token with the given id.

Transfer:

```
{
  "Type": "Transfer",
  "TokenId": string,
  "From": string,
  "To": string
}
```

A transfer transaction changes ownership of a token by removing the “from” wallet address, and adds it to the “to” wallet address.

Commands

Your program must handle the following commands:

Read Inline (`--read-inline <json>`)

Reads either a single json element, or an array of json elements representing transactions as an argument.

```
program --read-inline '{"Type": "Burn", "TokenId": “0x..."}'

program --read-inline '[{"Type": "Mint", "TokenId": "0x...",
"Address": "0x..."}, {"Type": "Burn", "TokenId": "0x..."}]'
```

Read File (`--read-file <file>`)

Reads either a single json element, or an array of json elements representing transactions from the file in the specified location.

```
program --read-file transactions.json
```

NFT Ownership (`--nft <id>`)

Returns ownership information for the nft with the given id

```
program --nft 0x...
```

Wallet Ownership (`--wallet <address>`)

Lists all NFTs currently owned by the wallet of the given address

```
program --wallet 0x...
```

Reset (`--reset`)

Deletes all data previously processed by the program

Sample Input / Output

Given the file `transactions.json` with the following contents:

```
[
    {
        "Type": "Mint",
        "TokenId": "0xA000000000000000000000000000000000000000",
        "Address": "0x1000000000000000000000000000000000000000"
    },
    {
        "Type": "Mint",
        "TokenId": "0xB000000000000000000000000000000000000000",
        "Address": "0x2000000000000000000000000000000000000000"
    },
    {
        "Type": "Mint",
        "TokenId": "0xC000000000000000000000000000000000000000",
```

```json
        "Address": "0x3000000000000000000000000000000000000000"
    },
    {
        "Type": "Burn",
        "TokenId": "0xA000000000000000000000000000000000000000"
    },
    {
        "Type": "Transfer",
        "TokenId": "0xB000000000000000000000000000000000000000",
        "From": "0x2000000000000000000000000000000000000000",
        "To": "0x3000000000000000000000000000000000000000"
    }
]
```

Here is a sample of several sequential executions of the program:

```
>program --read-file transactions.json
Read 5 transaction(s)

>program --nft 0xA000000000000000000000000000000000000000
Token 0xA000000000000000000000000000000000000000 is not owned by any wallet

>program --nft 0xB000000000000000000000000000000000000000
Token 0xA000000000000000000000000000000000000000 is owned by 0x3000000000000000000000000000000000000000

>program -nft 0xC000000000000000000000000000000000000000
Token 0xC000000000000000000000000000000000000000 is owned by 0x3000000000000000000000000000000000000000

>program --nft 0xD000000000000000000000000000000000000000
Token 0xA000000000000000000000000000000000000000 is not owned by any wallet

>program --read-inline '{ "Type": "Mint", "TokenId": "0xD000000000000000000000000000000000000000", "Address": "0x1000000000000000000000000000000000000000" }’
Read 1 transaction(s)

>program --nft 0xD000000000000000000000000000000000000000
Token 0xA000000000000000000000000000000000000000 is owned by 0x1000000000000000000000000000000000000000

>program --wallet 0x3000000000000000000000000000000000000000
Wallet 0x3000000000000000000000000000000000000000 holds 2 Tokens:
0xB000000000000000000000000000000000000000
0xC000000000000000000000000000000000000000

>program –reset
Program was reset

>program --wallet 0x3000000000000000000000000000000000000000
Wallet 0x3000000000000000000000000000000000000000 holds no Tokens
```

Rules

Read carefully the following rules:

- You may use any third party resources such as Google, StackOverflow, etc. Make sure you indicate the source of any code snippets you leverage.
- Your submission should include all source code.
- You should provide complete instructions for running your function and automation scripts in a README file.
- The scripts should be runnable from Mac OSX terminal and/or windows PowerShell. Your documentation should identify any prerequisites.

Guidelines and FAQ

How long do I have?

You will generally be given 24 hours to complete the challenge. We recommend you limit yourself to spending four hours on the challenge.

How is my work assessed?

There are no formal criteria for assessing your work however we are looking for qualities such as the following: clarity of code, extent of automation, ability to meet requirements, understanding of the ramifications of design choices.

How do I submit my entry?

Your entry should be submitted as a zip file containing your code and README files. You will be given an email address to which your zip file should be submitted.