ATicket

by A

Member

6231322621
Nitchakran Chaipojjana

6231363321
Siwagarn Jitwarodom

6230428321
Phumpakorn Saranun

Problems

Overpricing

Reselling

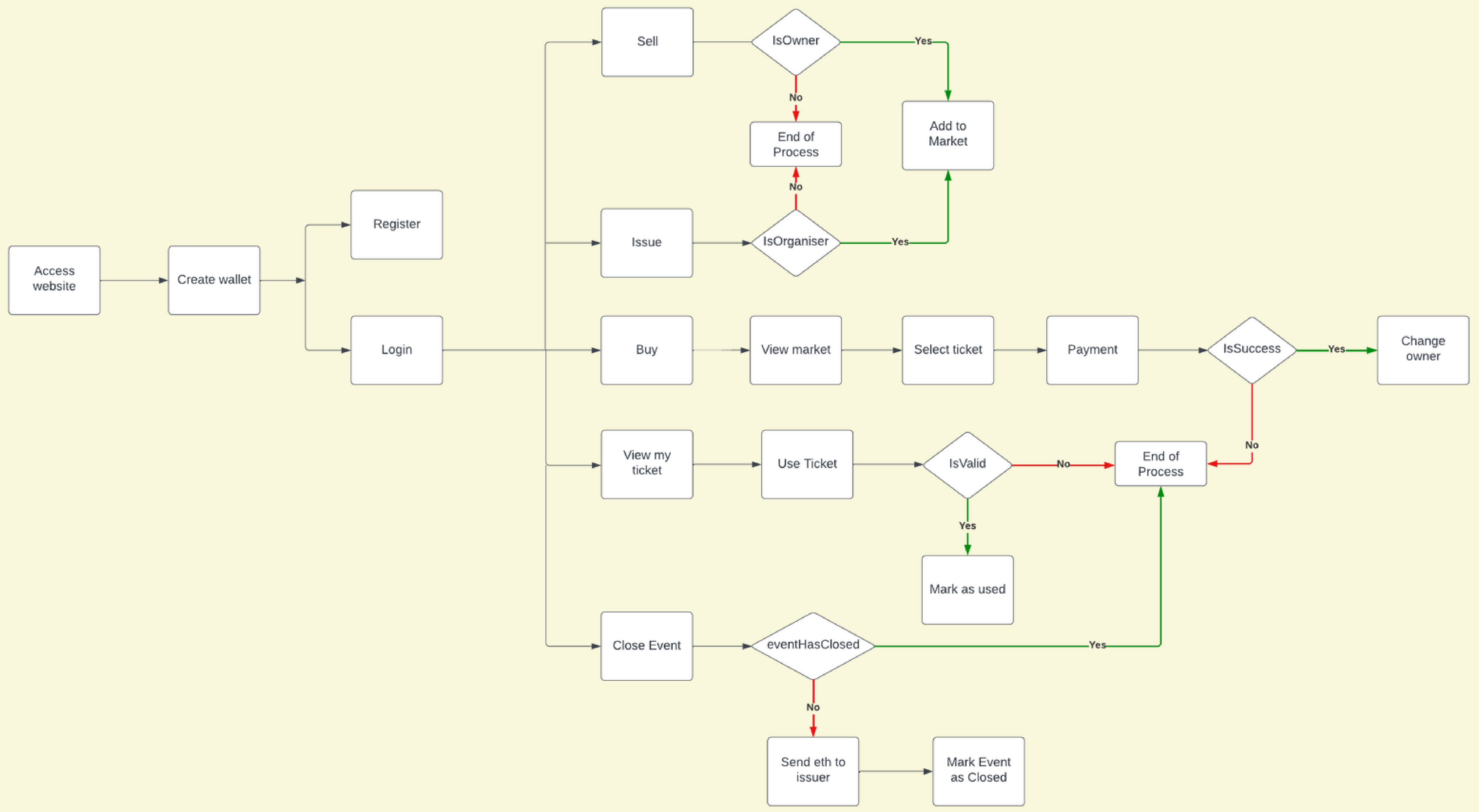
Cheating

Why need DLT

Fraud Prevention

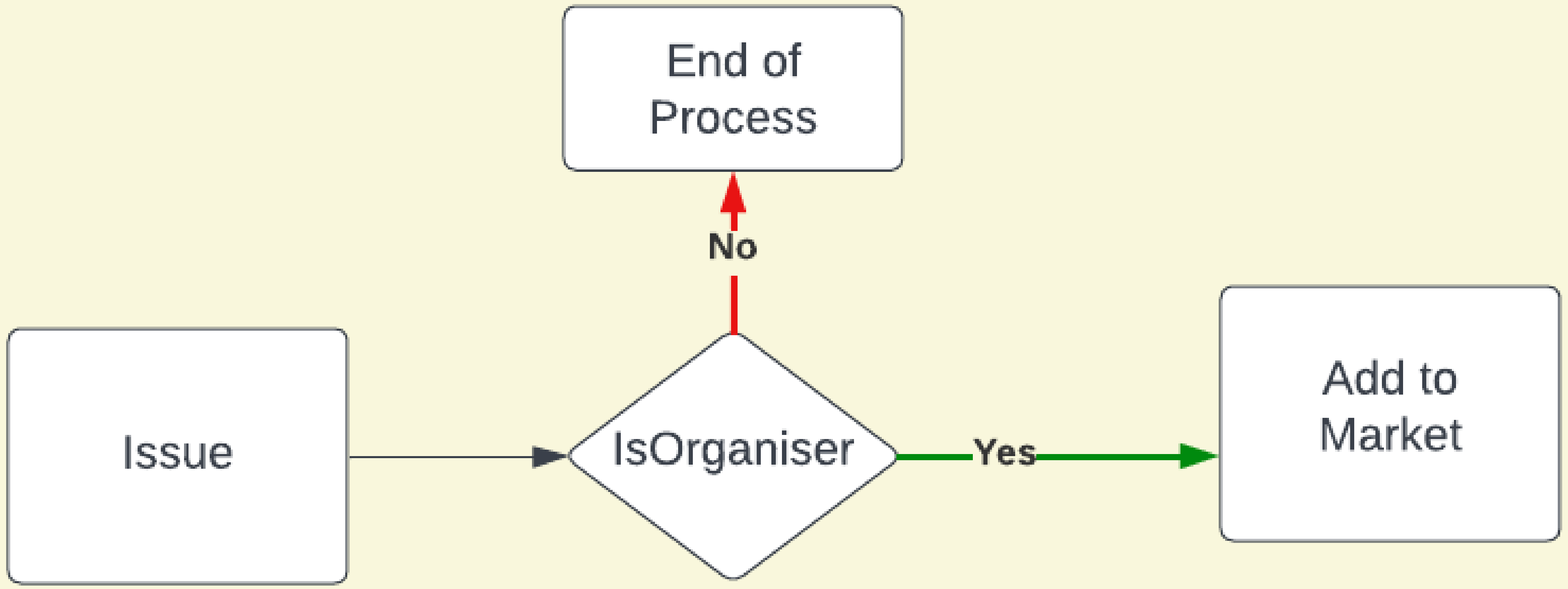
Transparent pricing

SOLUTION

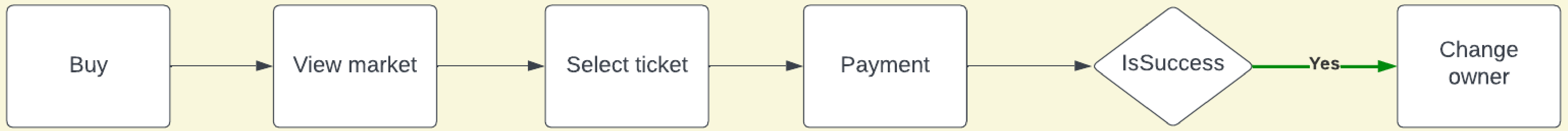


Access website
Create wallet
Register
Login
Sell
IsOwner
Yes
No
End of Process
Add to Market
No
Issue
IsOrganiser
Yes
Buy
View market
Select ticket
Payment
IsSuccess
Yes
Change owner
No
View my ticket
Use Ticket
IsValid
No
End of Process
Yes
Mark as used
Close Event
eventHasClosed
Yes
No
Send eth to issuer
Mark Event as Closed

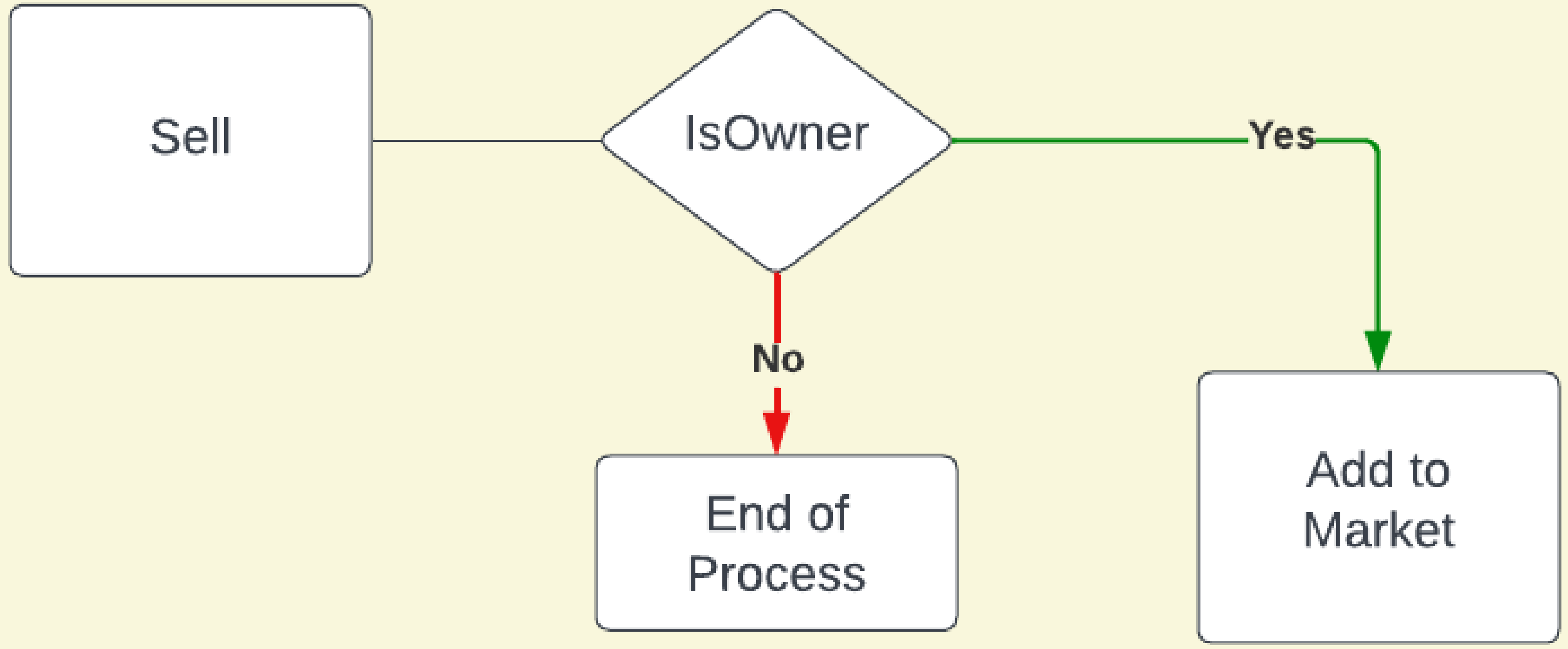
NEW EVENT& ISSUE TICKET



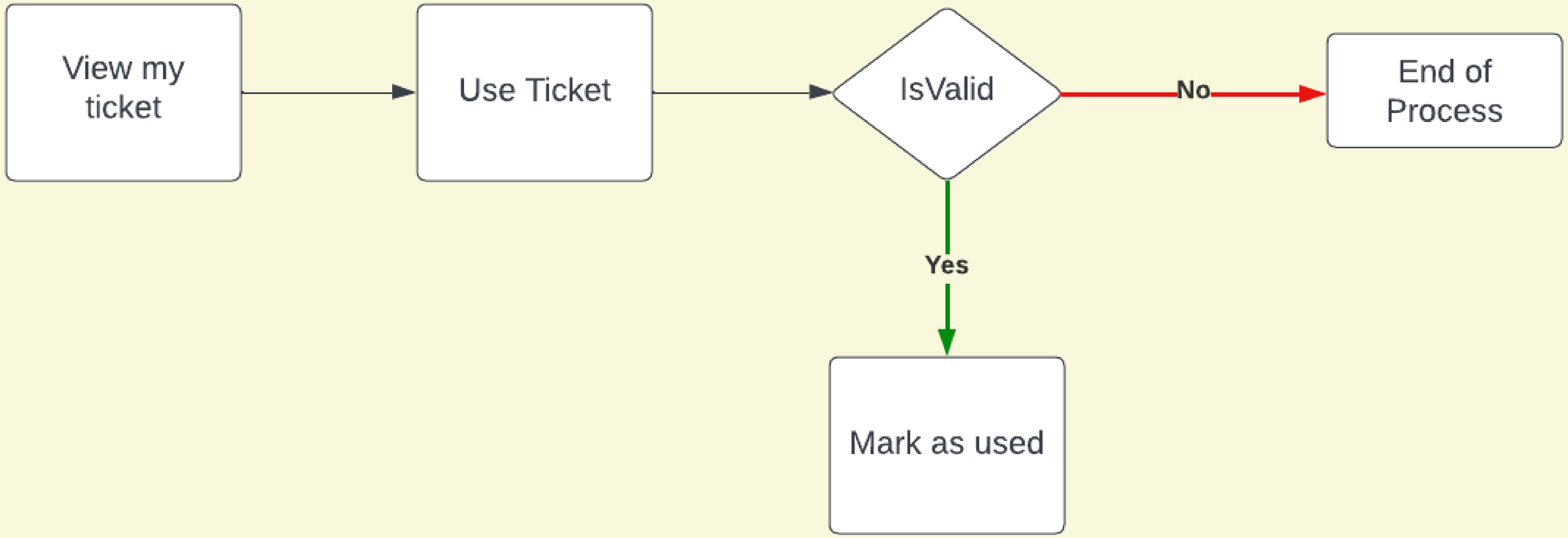
BUY TICKET


Buy
View market
Select ticket
Payment
IsSuccess
Yes
Change owner

SELL TICKET



USE TICKET


View my ticket
Use Ticket
IsValid
No
End of Process
Yes
Mark as used

SYSTEM OVERVIEW

$$\begin{tikzcd}
 & \text{Front-end server} \arrow[dl, shift right] \arrow[dr, shift left] & \\
\text{smart contract} \arrow[ur, shift right] & & \text{Back-end server} \arrow[ul, shift left]
\end{tikzcd}$$

ISSUER

create event and
issue ticket

FUNCTION

claim money after
event has closed

CUSTOMER

$$\begin{tikzcd}
\text{Buy ticket} & & \text{Sell ticket} \\
& \text{FUNCTION} \arrow[ul] \arrow[ur] \arrow[dl] \arrow[dr] & \\
\text{Use ticket} & & \text{View my ticket}
\end{tikzcd}$$

TOOLS

NEXT.JS

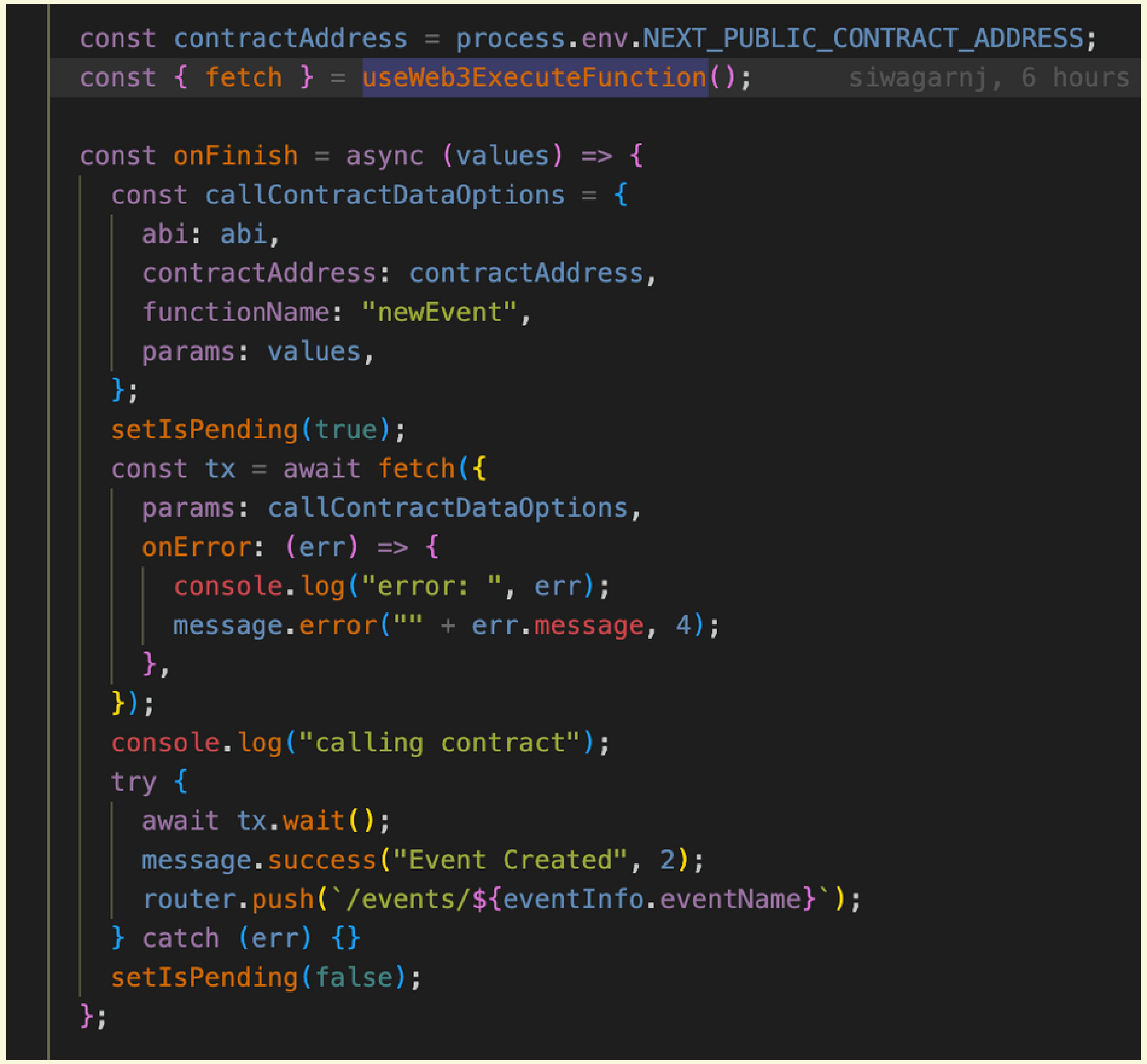
nest

react-moralis

alchemy

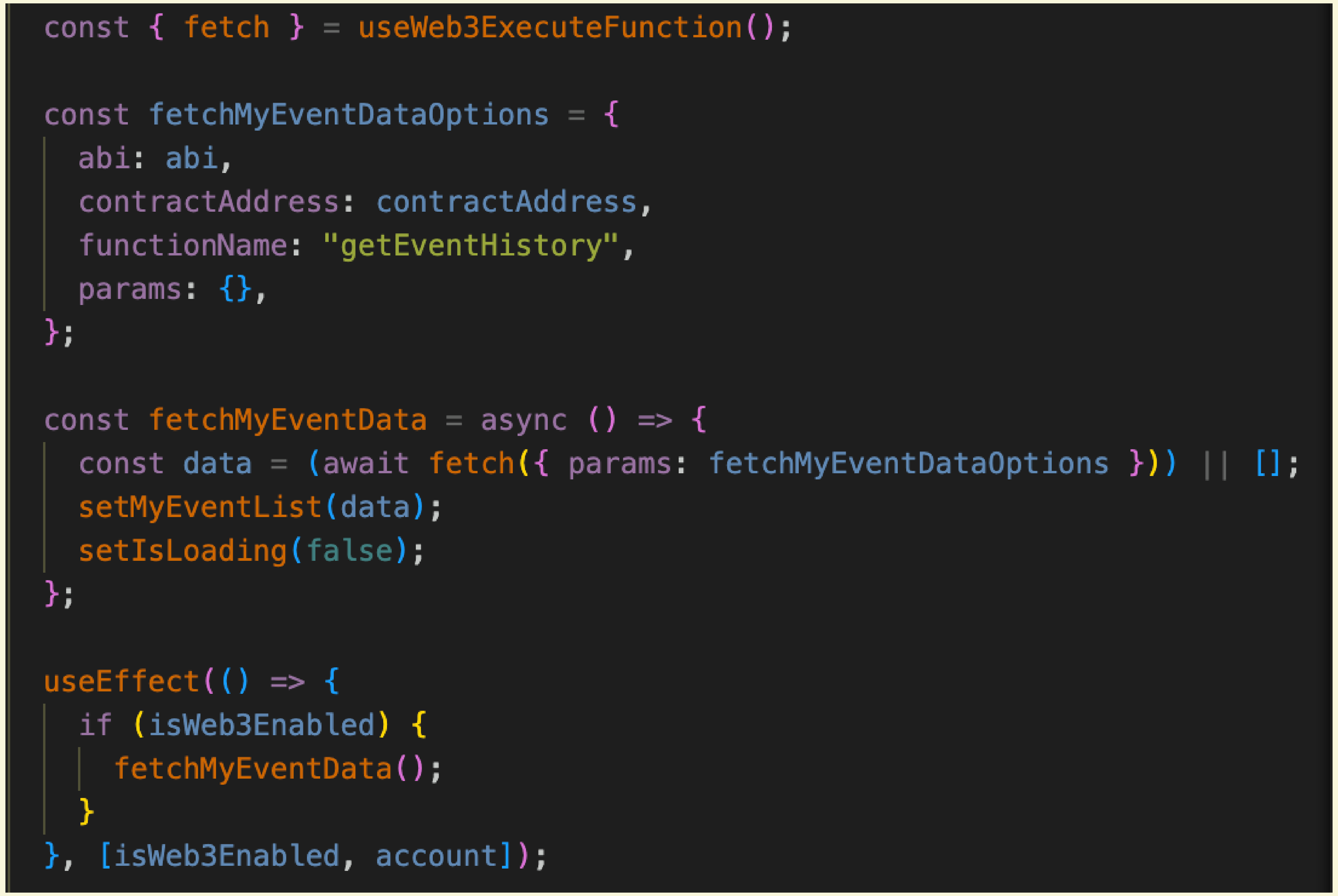
goerli testnet

CALLING SMART CONTRACT



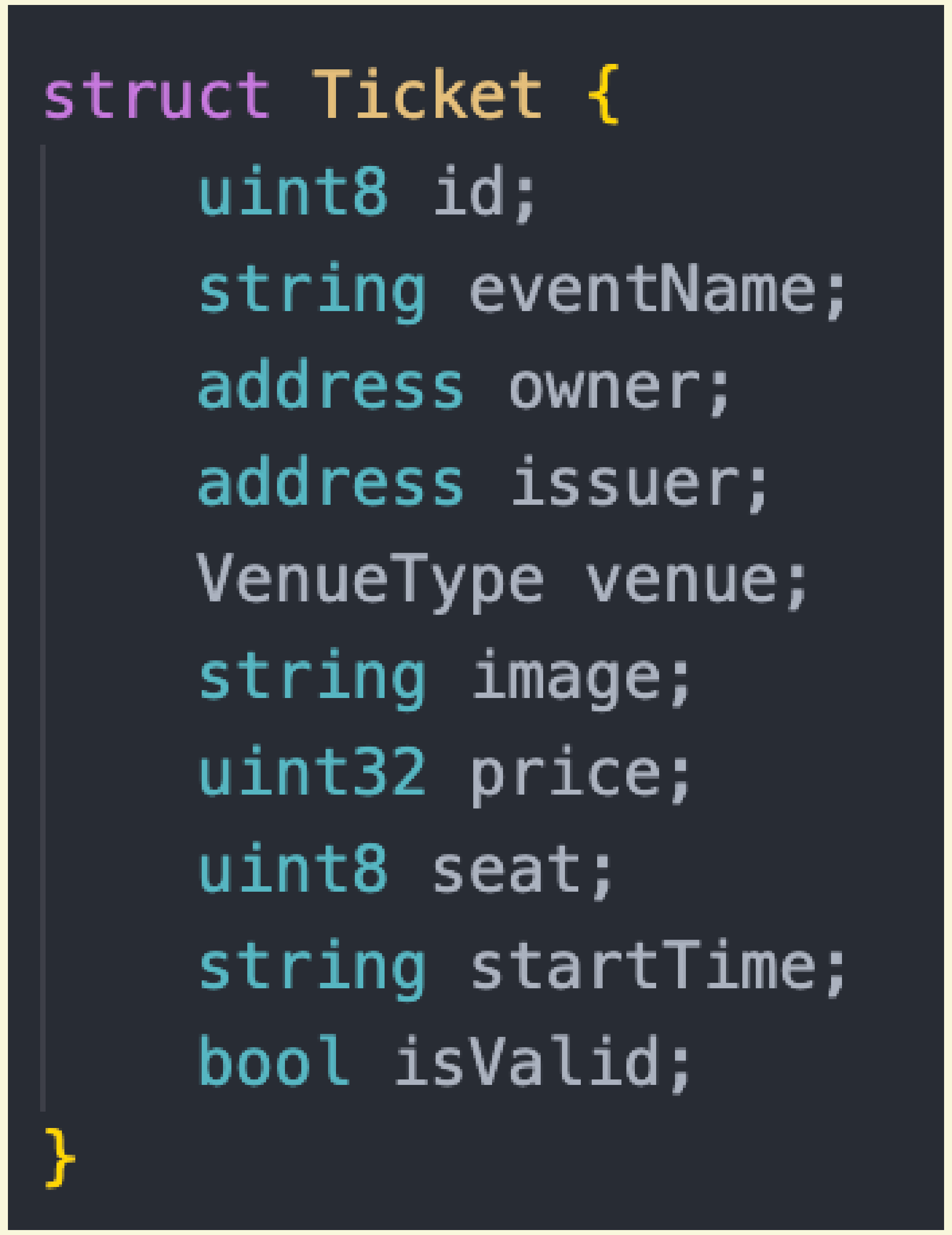
```
const contractAddress = process.env.NEXT_PUBLIC_CONTRACT_ADDRESS;
const { fetch } = useWeb3ExecuteFunction();      siwagarnj, 6 hours

const onFinish = async (values) => {
  const callContractDataOptions = {
    abi: abi,
    contractAddress: contractAddress,
    functionName: "newEvent",
    params: values,
  };
  setIsPending(true);
  const tx = await fetch({
    params: callContractDataOptions,
    onError: (err) => {
      console.log("error: ", err);
      message.error("" + err.message, 4);
    },
  });
  console.log("calling contract");
  try {
    await tx.wait();
    message.success("Event Created", 2);
    router.push(`/events/${eventInfo.eventName}`);
  } catch (err) {}
  setIsPending(false);
};
```

CALLING SMART CONTRACT

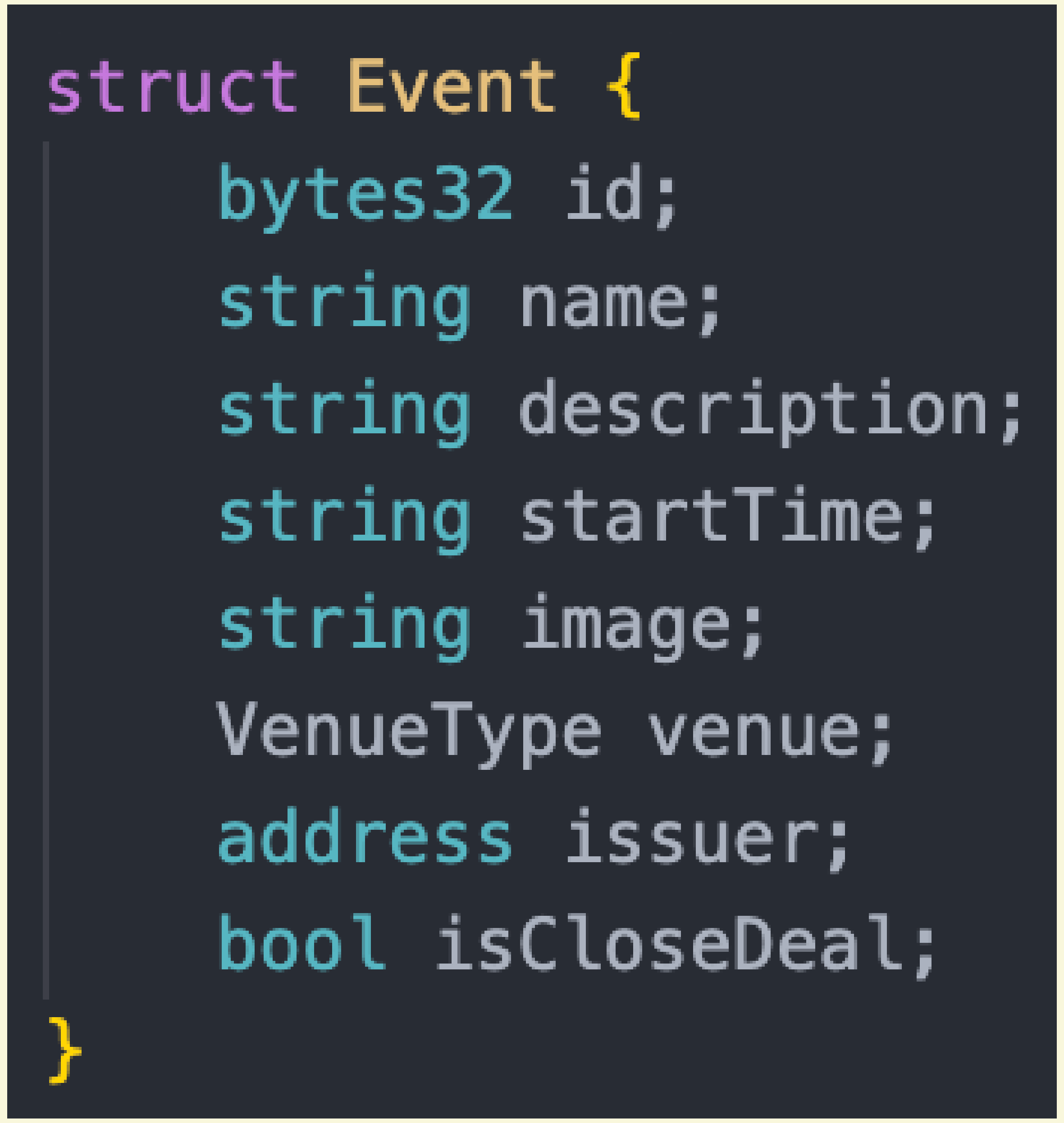


```
const { fetch } = useWeb3ExecuteFunction();

const fetchMyEventDataOptions = {
  abi: abi,
  contractAddress: contractAddress,
  functionName: "getEventHistory",
  params: {},
};

const fetchMyEventData = async () => {
  const data = (await fetch({ params: fetchMyEventDataOptions })) || [];
  setMyEventList(data);
  setIsLoading(false);
};

useEffect(() => {
  if (isWeb3Enabled) {
    fetchMyEventData();
  }
}, [isWeb3Enabled, account]);
```

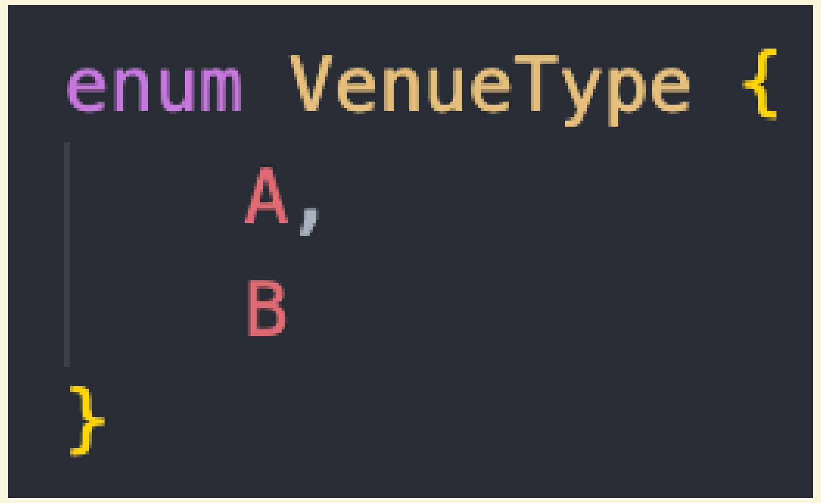
SMART CONTRACT



```
struct Ticket {
    uint8 id;
    string eventName;
    address owner;
    address issuer;
    VenueType venue;
    string image;
    uint32 price;
    uint8 seat;
    string startTime;
    bool isValid;
}
```



```
struct Event {
    bytes32 id;
    string name;
    string description;
    string startTime;
    string image;
    VenueType venue;
    address issuer;
    bool isCloseDeal;
}
```



```
enum VenueType {
    A,
    B
}
```

DEMO

● ● ● ● ●

Q&A

Thank You!