

I rewrote 'main.jsx' file based on the code in PPTX, implemented the whole process and UI part, it can be started by the following steps.

**Demonstration is shown in the screenshot 30s video, to be more succinct.**

Step1: Geth start

```
WEB3 — geth --http --http.corsdomain=* --http.api web3,eth,debug,personal,net -...
Last login: Wed Apr 26 19:53:16 on ttys001
guohanze@Hanzes-MacBook-Pro ~ % cd web3
guohanze@Hanzes-MacBook-Pro web3 % ls
index.html      node_modules    package.json     src               vite.config.js   yarn.lock
guohanze@Hanzes-MacBook-Pro web3 % geth --http --http.corsdomain=* --http.api web3,eth,debug,personal,net --vmdebug --datadir /tmp/test --dev console --ws --ws.api web3,eth,debug,net --ws.origins "*" --rpc.enableddeprecatedpersonal --allow-insecure-unlock
INFO [04-27|00:49:55.111] Starting Geth in ephemeral dev mode...
WARN [04-27|00:49:55.111] You are running Geth in --dev mode. Please note the following:

1. This mode is only intended for fast, iterative development without assumptions on security or persistence.
2. The database is created in memory unless specified otherwise. Therefore, shutting down your computer or losing power will wipe your entire block data and chain state for your dev environment.
3. A random, pre-allocated developer account will be available and unlocked as eth.coinbase, which can be used for testing. The random dev account is temporary, stored on a ramdisk, and will be lost if your machine is restarted.
4. Mining is enabled by default. However, the client will only seal blocks if transactions are pending in the mempool. The miner's minimum accepted gas price is 1.
5. Networking is disabled; there is no listen-address, the maximum number of peers is set
```

Step1.1: Create accounts

Step1.2 Import and unlock accounts (the accounts in JSX file wallet)

Step1.3 Transfer ETH to the unlocked accounts to get initial funds

```
WEB3 — geth --http --http.corsdomain=* --http.api web3,eth,debug,personal,net -...
INFO [04-27|18:52:47.465] Commit new sealing work          number=290 sealhash=66
63d8..5ceb9e uncles=0 txs=1 gas=41256 fees=0.0001089684747 elapsed="529µs"
INFO [04-27|18:52:47.465] Successfully sealed new block    number=290 sealhash=66
63d8..5ceb9e hash=54ce4e..43ba29 elapsed="756.75µs"
INFO [04-27|18:52:47.466] " block reached canonical chain" number=283 hash=3fcb1
d..b072ed
INFO [04-27|18:52:47.466] " mined potential block"         number=290 hash=54ce4
e..43ba29
INFO [04-27|18:52:47.466] Commit new sealing work          number=291 sealhash=07
054e..42f17d uncles=0 txs=0 gas=0 fees=0 elapsed="442.084µs"
WARN [04-27|18:52:47.466] Block sealing failed            err="sealing paused wh
ile waiting for transactions"
INFO [04-27|18:52:47.466] Commit new sealing work          number=291 sealhash=07
054e..42f17d uncles=0 txs=0 gas=0 fees=0 elapsed="537.292µs"
> INFO [04-27|19:28:30.162] Writing clean trie cache to disk path=/tmp/test/geth/
triecache threads=1
INFO [04-27|19:28:30.166] Persisted the clean trie cache   path=/tmp/test/geth/tr
iecache elapsed=3.472ms
INFO [04-27|19:28:30.166] Regenerated local transaction journal transactions=0 account
s=0
> eth.accounts
["0x2e8951c0f001bd28c40d86fb39f550b956fb4de4", "0x06e27975fa876fac4bc8e986d5f96ac1e3d3b7d
4", "0x808b22697fcf08e3e74e01b580bb0db1f7ca824f"]
>
```

```
WEB3 — geth --http --http.corsdomain=* --http.api web3,eth,debug,personal,net -...
INFO [04-27|18:52:47.466] "⏏ mined potential block" number=290 hash=54ce4e..43ba29
INFO [04-27|18:52:47.466] Commit new sealing work number=291 sealhash=07054e..42f17d uncles=0 txs=0 gas=0 fees=0 elapsed="442.084µs"
WARN [04-27|18:52:47.466] Block sealing failed err="sealing paused while waiting for transactions"
INFO [04-27|18:52:47.466] Commit new sealing work number=291 sealhash=07054e..42f17d uncles=0 txs=0 gas=0 fees=0 elapsed="537.292µs"
> INFO [04-27|19:28:30.162] Writing clean trie cache to disk path=/tmp/test/geth/triecache threads=1
INFO [04-27|19:28:30.166] Persisted the clean trie cache path=/tmp/test/geth/triecache elapsed=3.472ms
INFO [04-27|19:28:30.166] Regenerated local transaction journal transactions=0 accounts=0
> eth.accounts
["0x2e8951c0f001bd28c40d86fb39f550b956fb4de4", "0x06e27975fa876fac4bc8e986d5f96ac1e3d3b7d4", "0x808b22697fcf08e3e74e01b580bb0db1f7ca824f"]
> personal.unlockAccounts("0x2e8951c0f001bd28c40d86fb39f550b956fb4de4","",0)
TypeError: Object has no member 'unlockAccounts'
    at <eval>:1:24(5)

> personal.unlockAccount("0x2e8951c0f001bd28c40d86fb39f550b956fb4de4","",0)
true
> █

WEB3 — geth --http --http.corsdomain=* --http.api web3,eth,debug,personal,net -...
> personal.unlockAccount("0x2e8951c0f001bd28c40d86fb39f550b956fb4de4","",0)
true
> eth.sendTransaction({from: "0x2e8951c0f001bd28c40d86fb39f550b956fb4de4", to: "0x808b22697fcf08e3e74e01b580bb0db1f7ca824f", value: web3.toWei(10000, "ether")})
INFO [04-27|19:52:20.392] Submitted transaction hash=0x4d26dff1d765f5823207e27ccc34a473ec51d69f17332810b102077e3c141fb7 from=0x2E8951C0F001Bd28C40d86fB39f550B956fB4dE4 nonce=12 recipient=0x808B22697fCf08E3e74e01B580bb0Db1f7Ca824F value=10,000,000,000,000,000,000,000,000
"0x4d26dff1d765f5823207e27ccc34a473ec51d69f17332810b102077e3c141fb7"
> INFO [04-27|19:52:20.392] Commit new sealing work number=291 sealhash=736e4a..e239f2 uncles=0 txs=1 gas=21000 fees=5.546679193e-05 elapsed="328.25µs"
INFO [04-27|19:52:20.394] Successfully sealed new block number=291 sealhash=736e4a..e239f2 hash=e61fdd..8614b0 elapsed=1.395ms
INFO [04-27|19:52:20.394] "🔗 block reached canonical chain" number=284 hash=3c7a9a..8d3f64
INFO [04-27|19:52:20.394] "⏏ mined potential block" number=291 hash=e61fd..8614b0
INFO [04-27|19:52:20.394] Commit new sealing work number=292 sealhash=f392d9..c19950 uncles=0 txs=0 gas=0 fees=0 elapsed="161.459µs"
WARN [04-27|19:52:20.394] Block sealing failed err="sealing paused while waiting for transactions"
INFO [04-27|19:52:20.394] Commit new sealing work number=292 sealhash=f392d9..c19950 uncles=0 txs=0 gas=0 fees=0 elapsed="228.084µs"
█
```

Total 3 accounts, the first one is the banker, others are the players.

Step2: Run Vite

```

WEB3 — esbuild • node /opt/homebrew/bin/yarn run vite --port 3000 — 8...

Last login: Wed Apr 26 19:53:11 on ttys000
[guohanze@Hanzes-MacBook-Pro ~ % cd web3
[guohanze@Hanzes-MacBook-Pro web3 % yarn run vite --port 3000
yarn run v1.22.19
warning package.json: No license field
$ /Users/guohanze/WEB3/node_modules/.bin/vite --port 3000

VITE v4.2.1 ready in 334 ms

  → Local:   http://localhost:3000/
  → Network: use --host to expose
  → press h to show help
7:56:38 PM [vite] hmr update /src/main.jsx
7:56:38 PM [vite] hmr invalidate /src/main.jsx Could not Fast Refresh. Learn more
e at https://github.com/vitejs/vite-plugin-react/tree/main/packages/plugin-react
#consistent-components-exports
8:34:24 PM [vite] hmr update /src/main.jsx
8:34:25 PM [vite] hmr invalidate /src/main.jsx Could not Fast Refresh. Learn more
e at https://github.com/vitejs/vite-plugin-react/tree/main/packages/plugin-react
#consistent-components-exports
12:36:09 AM [vite] hmr update /src/main.jsx
12:36:10 AM [vite] hmr invalidate /src/main.jsx Could not Fast Refresh. Learn more
e at https://github.com/vitejs/vite-plugin-react/tree/main/packages/plugin-react
#consistent-components-exports

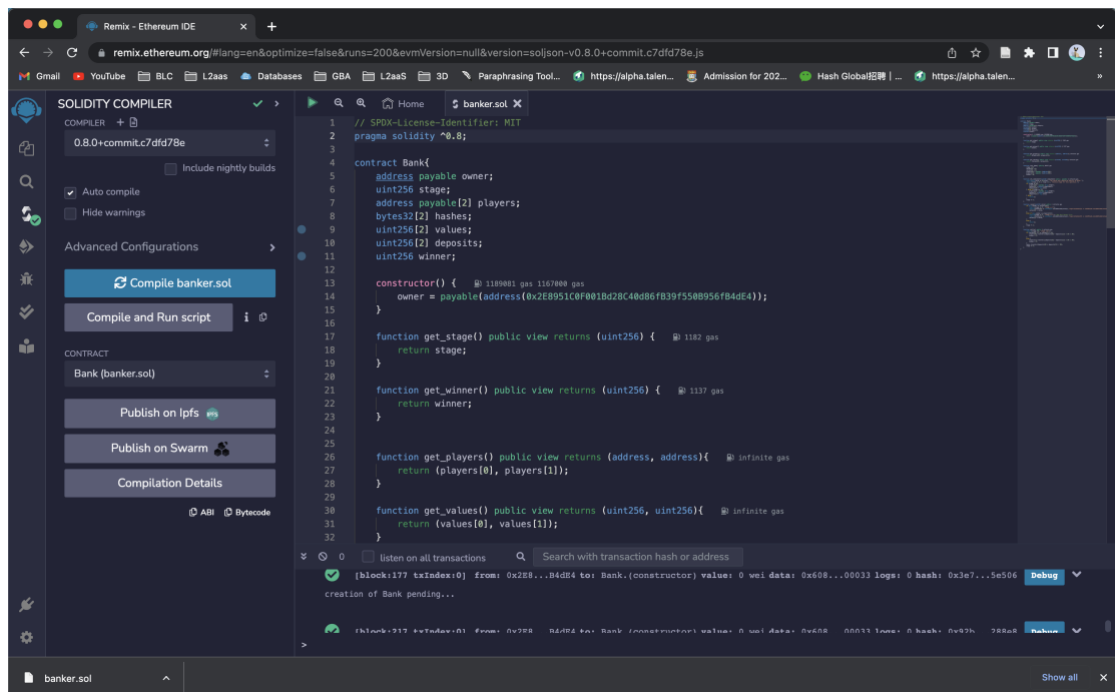
```

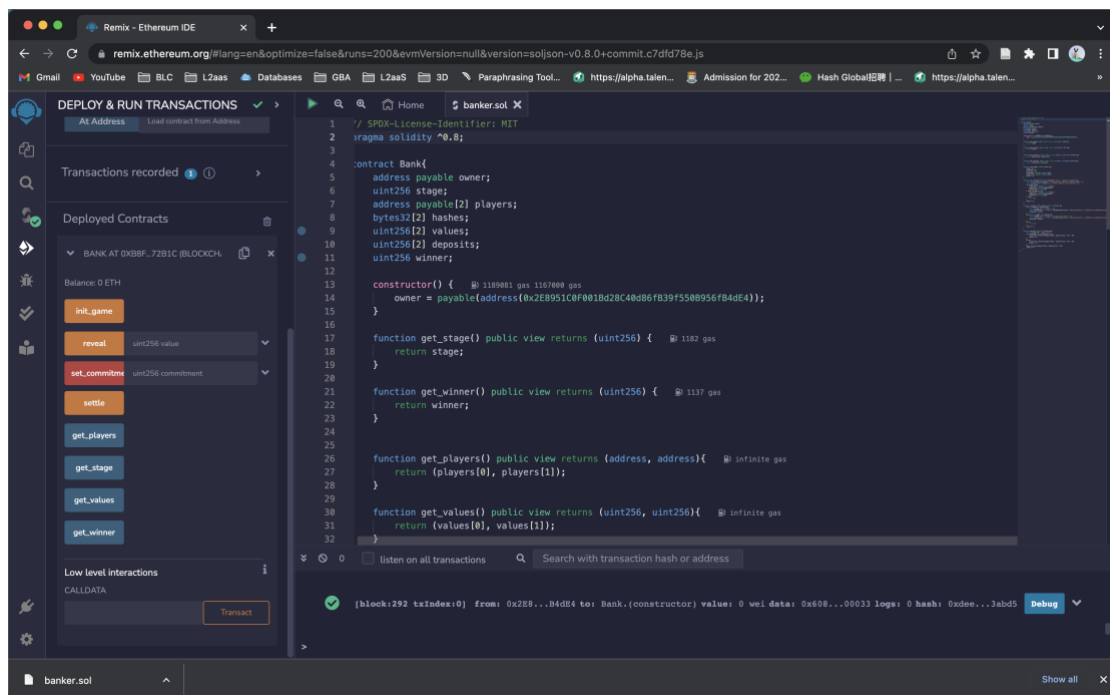
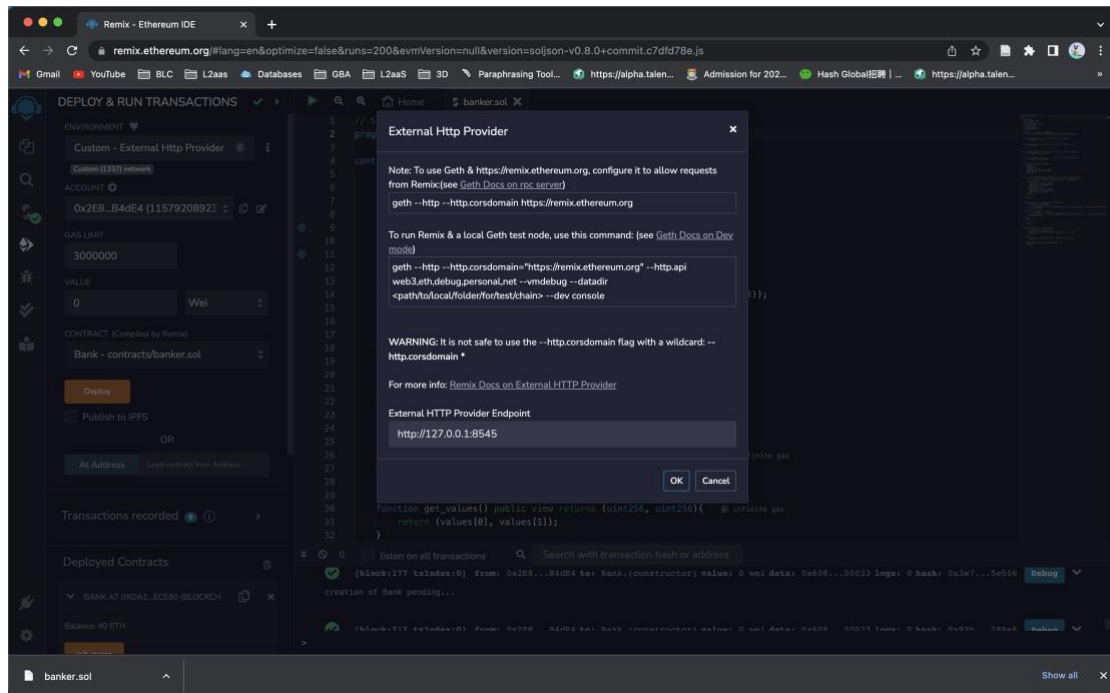
Step3: Run Remix

Step3.1: Compile the smart contract 'banker.sol'

Step3.2: Deploy the contract on the provider

Step3.3: Set the JSX file parameters





Step4: Play Demo on 'http://localhost:3000/'

**Demonstration is shown in the screenshot 30s video, please have a check.**

The question in Guan's PDF, Phase 2, Requirement 1, the following is the answer:

***Question:***

Please explain/prove whether this protocol actually works or not, in term of fairness, and what are the possible attacks (cheating) among the three parties. In your implementation, how have you solved them? You need to answer this question in your design document.

***Answer:***

The protocol is fair because it guarantees that neither party can cheat or manipulate the outcome of the game, assuming a reliable hash function is used. This is because the hash function is a one-way function, meaning that it is infeasible to determine the input value that corresponds to a given hash value. As a result, the parties cannot change their chosen number after sending the hash value to the banker without it being detected in Round 2 when the hash value is revealed along with the chosen number. Therefore, the winner is determined by the outcome of the coin flip based on the values chosen by the participants and the hash values sent to the banker.

However, there are still possible attacks that can occur in the protocol if it is not implemented correctly or if there are malicious actors involved. One possible attack is collusion between one of the parties and the banker, where they work together to cheat the other party. For example, the banker could provide one of the parties with the hash value of the other party's chosen number, allowing them to choose a number that will result in a win. This can be prevented by ensuring that the banker is trustworthy and does not collude with either party.

**[In my implementation, the banker is the smart contract and it is impossible to reveal the number to anyone of the two players to collude, and the number is generated totally randomly.]**

Another possible attack is the use of a weak hash function that can be easily reversed, allowing a party to determine the other party's chosen number from the hash value. This can be prevented by using a secure hash function that is resistant to preimage attacks, such as SHA-256.

**[In my implementation, hash function is SHA-256.]**

Finally, there is a risk that one of the parties may not reveal their chosen number in Round 2, which would prevent the outcome of the game from being determined. This can be addressed by having a time limit for revealing the chosen numbers and by having the banker notify all parties when the game is over.

**[In my implementation, the smart contract will inform all two players when the game is over, and the state information will always be on the board.]**