# Appendix

6CCS3PRJ Final Project

BSc Computer Science

Author: Henry Li
Student ID: 1923074
Supervisor: Dr Agi Kurucz

April 2023

# Contents

# Appendix A: Background Information

## A1. Graphical User Interface Wireframes

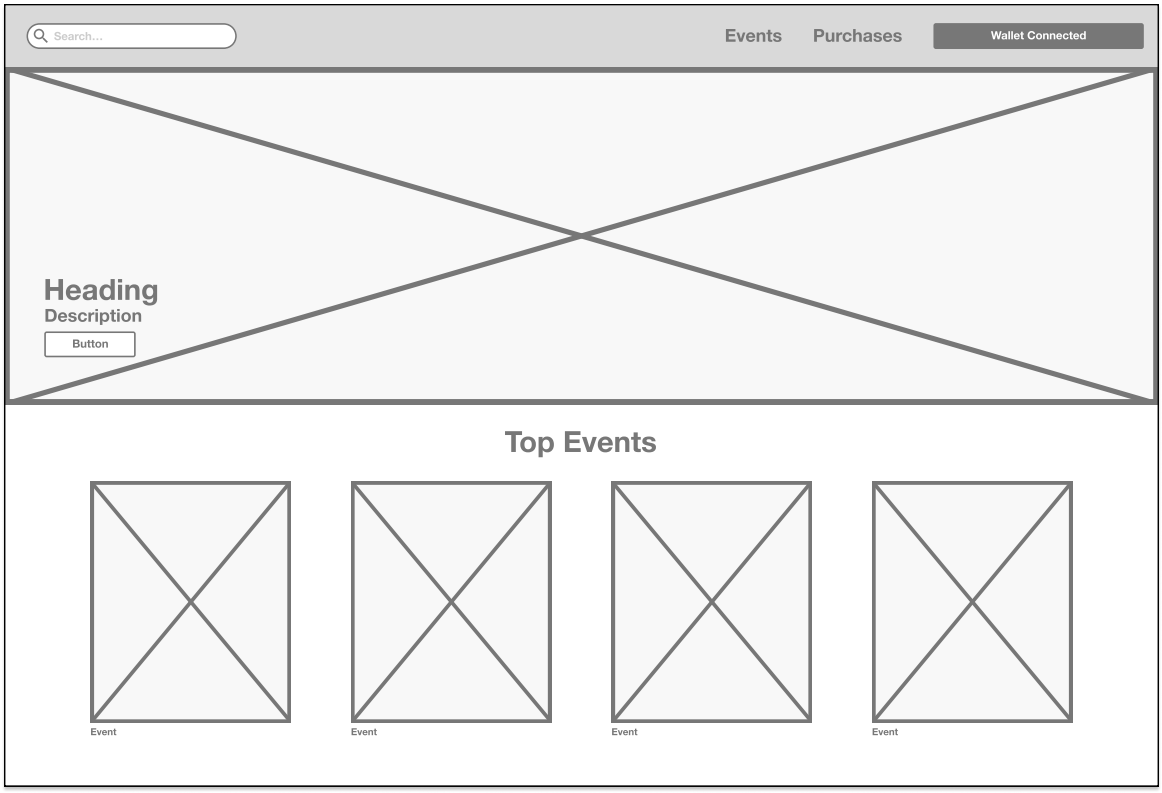


**Figure A1.1: Web3 Application Home Page Wireframe**

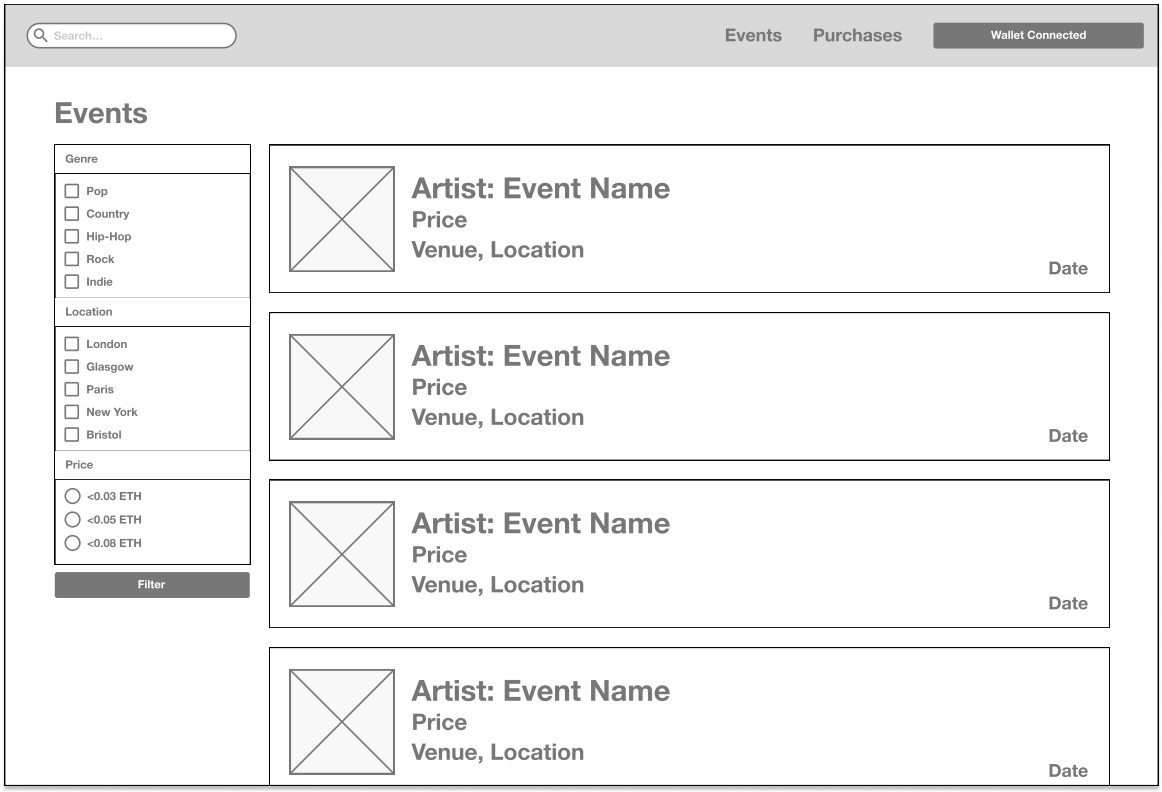


**Figure A1.2: Web3 Application Browse Events Page Wireframe**

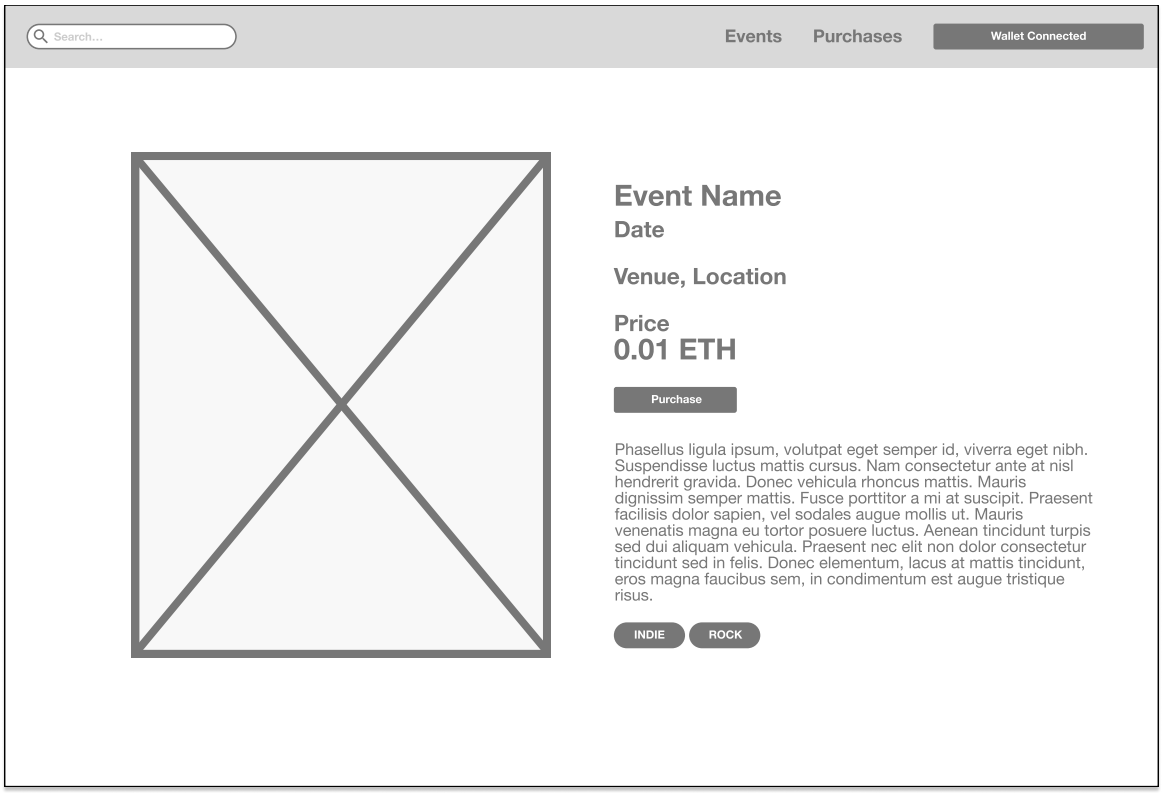


**Figure A1.3: Web3 Application Event Page Wireframe**

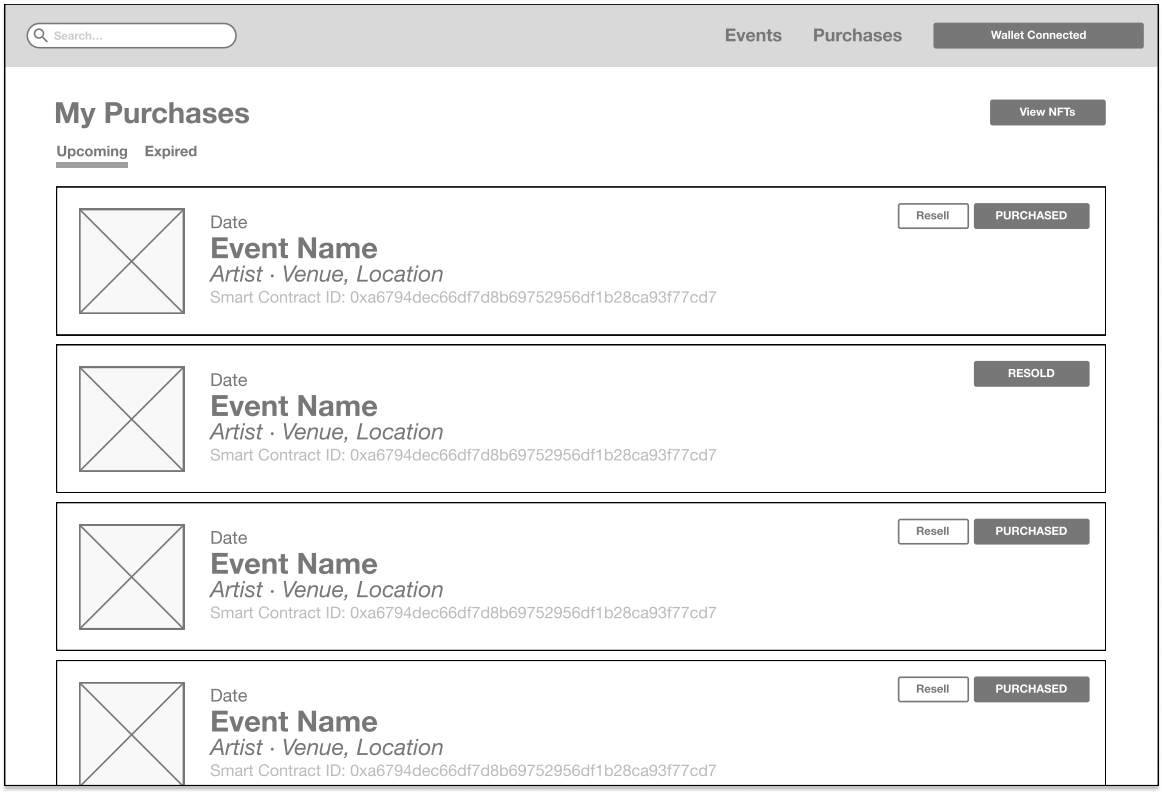


**Figure A1.4: Web3 Application Purchases Page Wireframe**

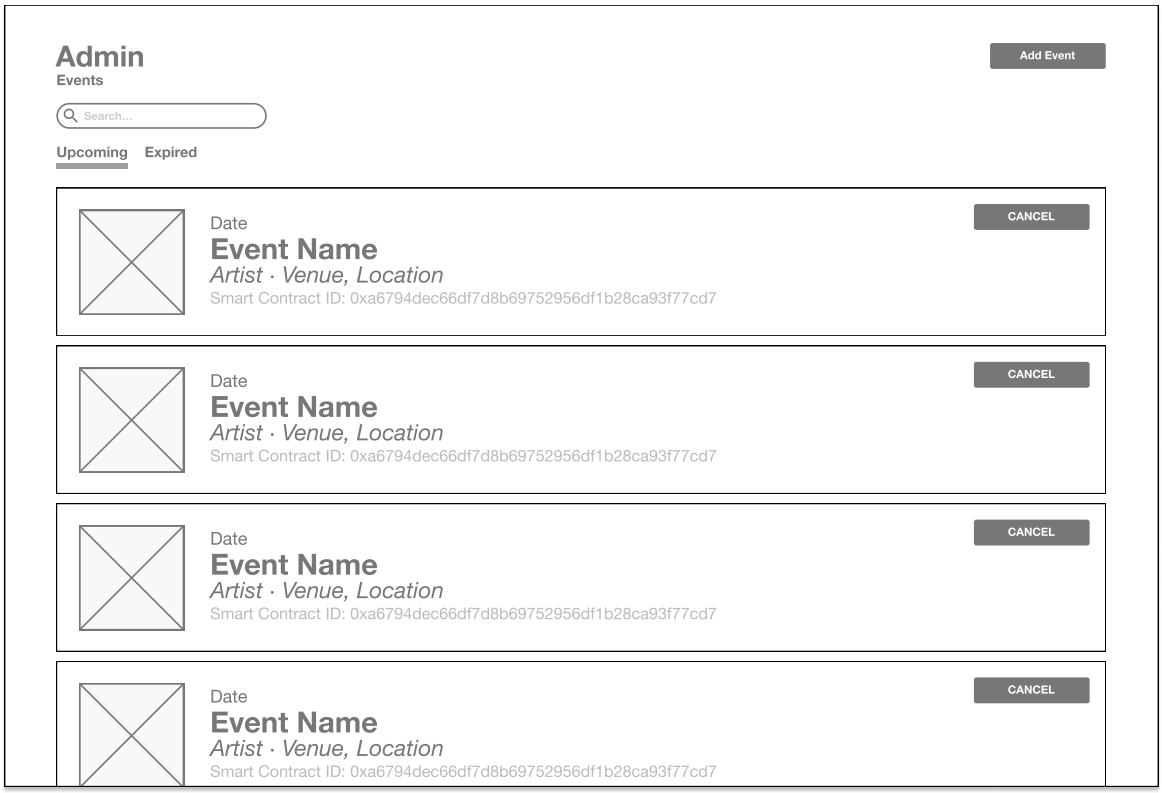


**Figure A1.5: Web3 Application Admin Page Wireframe**

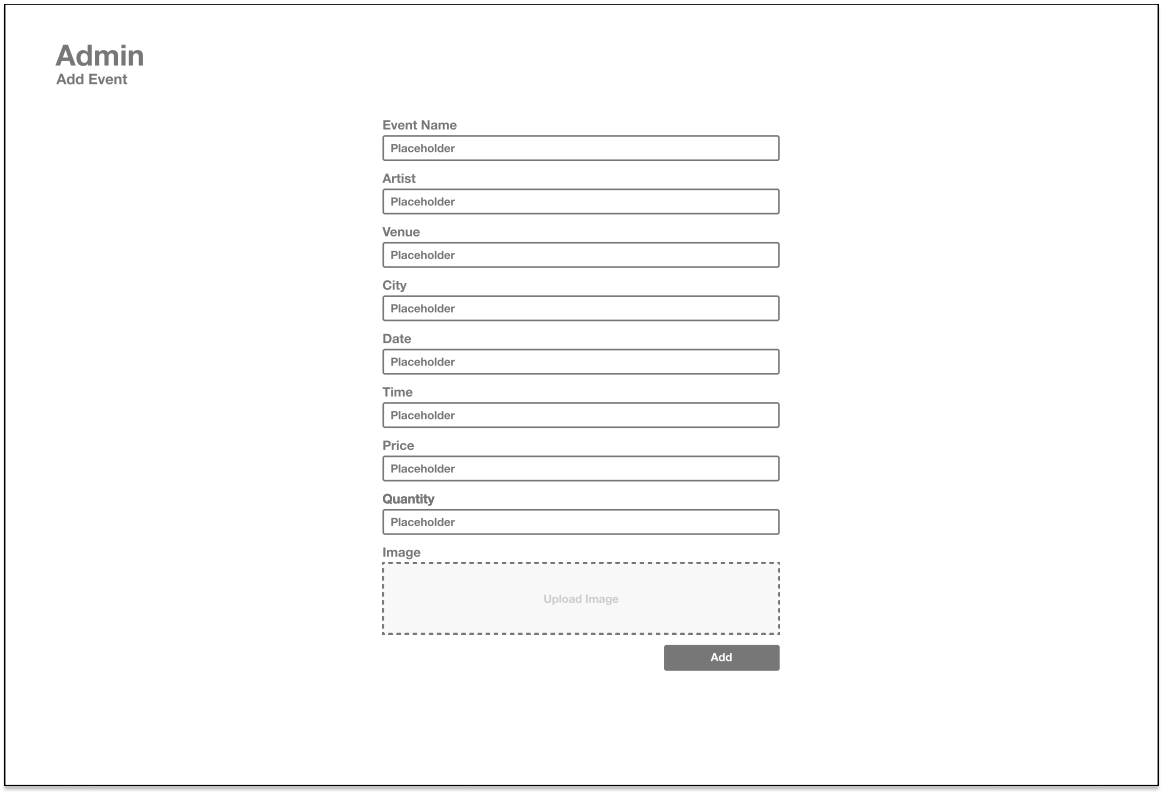


**Figure A1.6: Web3 Application Admin Add/Edit Event Page Wireframe**

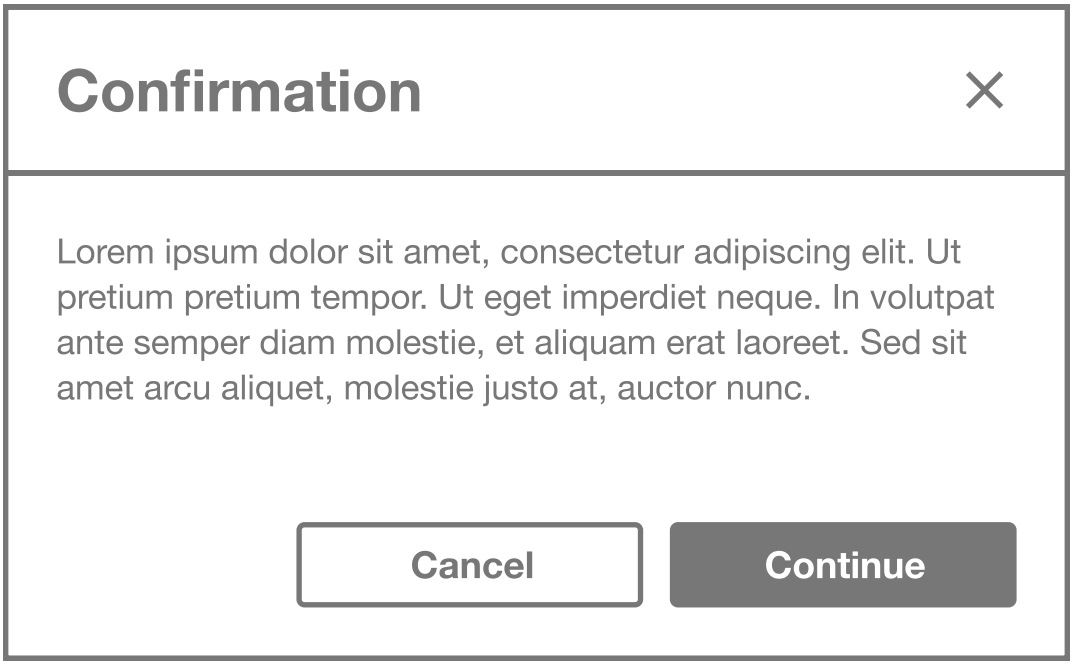


**Figure A1.7: Web3 Application Confirmation Modal Wireframe**

## A2. MetaMask Transaction Interface

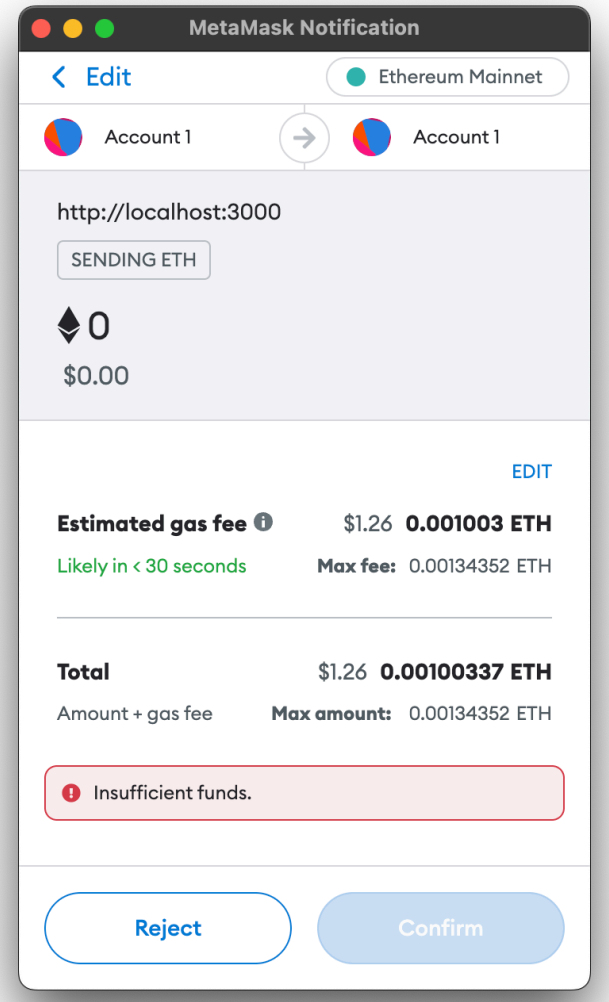


**Figure A2: MetaMask Transaction Pop-up**

# Appendix B: Project Information

## B1. Unit Test Output

```
Contract: Events
  ✔ deploys a contract
  createEvent
    ✔ creates an event (126ms)
    ✔ reverts if event already created (273ms)
    ✔ reverts if time is in the past (98ms)
    ✔ reverts when not owner (101ms)
    ✔ reverts when quantity is 0 (49ms)
    ✔ reverts when event with same ID exists (68ms)
  updateEvent
    ✔ updates an event
    ✔ reverts when not owner
    ✔ reverts if event not created
    ✔ reverts if event is cancelled (48ms)
    ✔ reverts if quantity is less than what has already been supplied
    (49ms)
    ✔ reverts if quantity is 0
    ✔ reverts if time is in the past
  buyToken
    ✔ buys a token
    ✔ reverts if event does not exist
    ✔ reverts if event has been cancelled (47ms)
    ✔ reverts with insufficient payment
    ✔ reverts with too much quantity (52ms)
    ✔ reverts when event is in the past (59ms)
  getBalance
    ✔ returns correct balance (49ms)
    ✔ reverts when not owner
  transferBalance
    ✔ transfers contract balance (84ms)
    ✔ reverts when not owner
  cancelEvent
    ✔ cancels an event with no token owners
    ✔ cancels an event, refunding token owners (59ms)
    ✔ reverts when there is insufficient ETH to refund token owners (77ms)
    ✔ reverts if the event does not exist
    ✔ reverts if the event has already been cancelled (45ms)
    ✔ reverts if `owners` and `quantity` parameters are of different
    length (70ms)
    ✔ reverts when not owner
  markTokenAsUsed
    ✔ marks tokens as used (40ms)
    ✔ reverts if not owner
    ✔ reverts if event does not exist
    ✔ reverts if event has been cancelled (47ms)
    ✔ reverts if there are not enough tokens to mark as used (54ms)
```

```
  getUsedCount
    ✔ gets the correct used count for an address (61ms)
    ✔ gets the correct used count for an address, for a specific event
    (146ms)
  getUsedTokens
    ✔ gets the used tokens for an address (89ms)
  listTokenForResale
    ✔ lists tokens for resale (71ms)
    ✔ succeeds with quantity 0 (42ms)
    ✔ reverts if the event does not exist (47ms)
    ✔ reverts if the event has been cancelled (65ms)
    ✔ reverts if trying to list more than available (125ms)
  unlistTokenForResale
    ✔ unlists tokens for resale (89ms)
    ✔ unlists multiple tokens for resale (73ms)
    ✔ succeeds with quantity 0 (47ms)
    ✔ reverts if event does not exist
    ✔ reverts if event has been cancelled (88ms)
    ✔ reverts if quantity more than what has been listed (71ms)
  buyResaleToken
    ✔ buys a token listed for resale (113ms)
    ✔ reverts if event does not exist (61ms)
    ✔ reverts if event is cancelled (106ms)
    ✔ reverts if insufficient payment (60ms)
    ✔ reverts if there are no tokens for resale (129ms)
  getListedCount
    ✔ gets number of tokens listed for address (211ms)
  getResaleTokenEntries
    ✔ gets resale token entries for an address (173ms)
  getResaleTokens
    ✔ gets resale token entries for an event (212ms)


58 passing (13s)

------------|----------|----------|----------|----------|
File        |  % Stmts | % Branch |  % Funcs |  % Lines |
------------|----------|----------|----------|----------|
 contracts/ |      100 |      100 |      100 |      100 |
  Events.sol |      100 |      100 |      100 |      100 |
------------|----------|----------|----------|----------|
All files   |      100 |      100 |      100 |      100 |
------------|----------|----------|----------|----------|
```

## B2. Implemented System

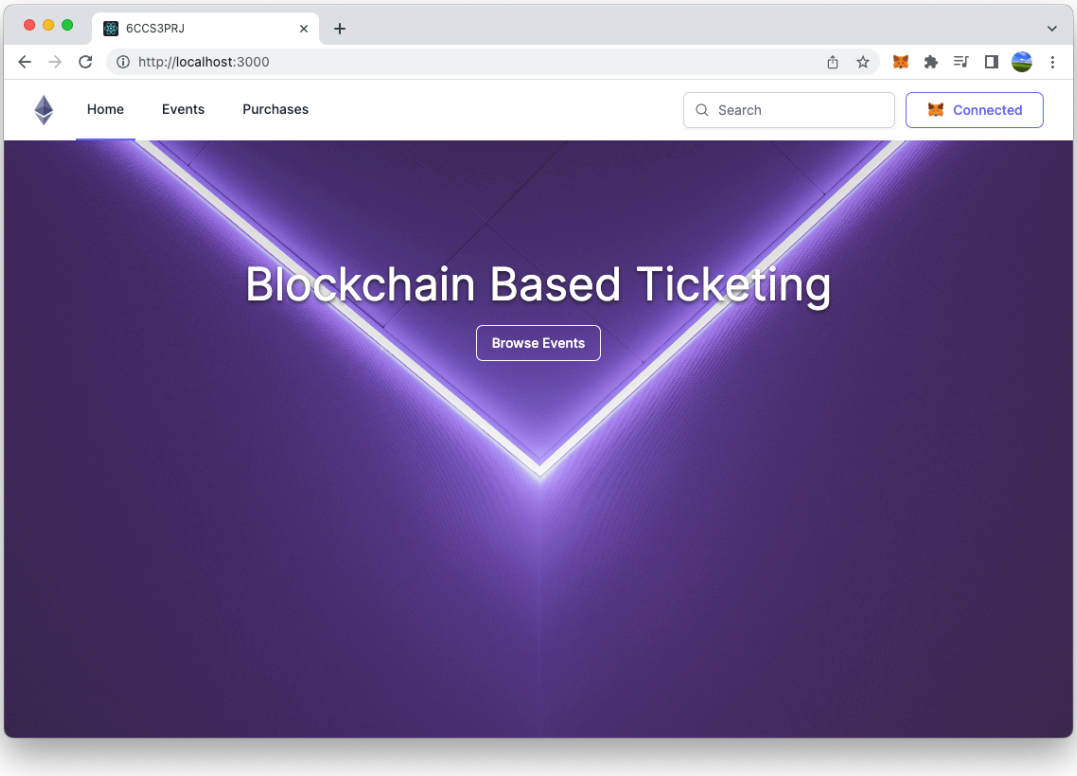


**Figure B2.1: Application Home Screen**

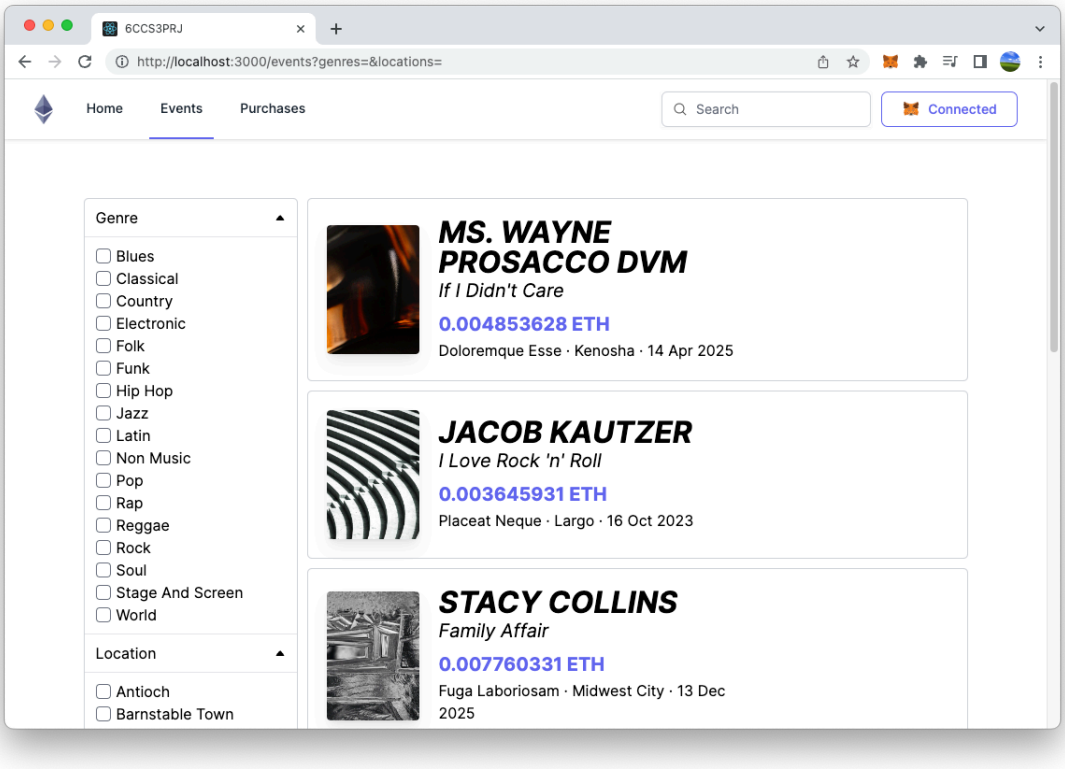


**Figure B2.2: Application Events Page**

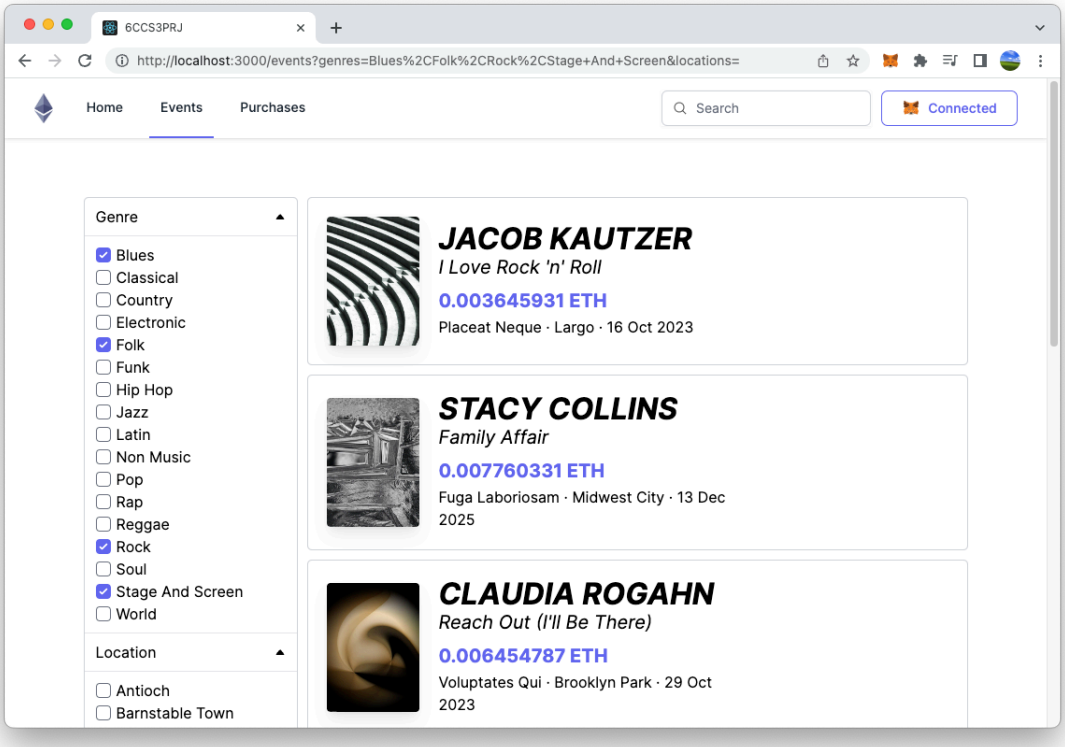


**Figure B2.3: Application Events Page Filtering**

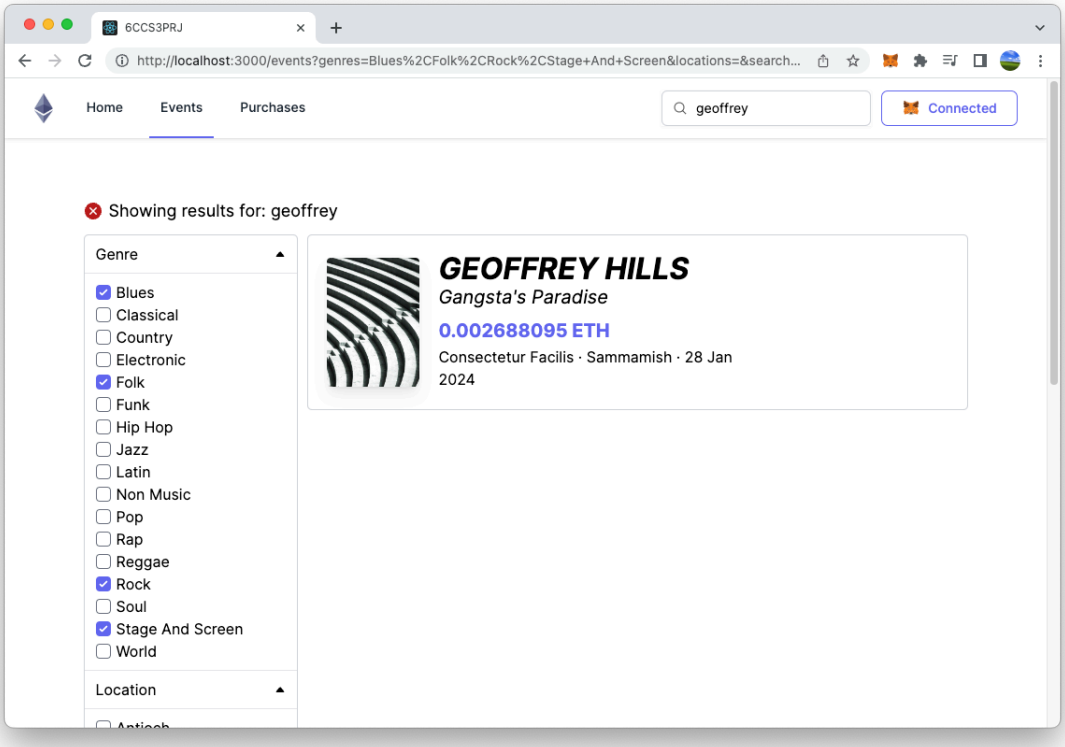


**Figure B2.4: Application Events Page Searching**

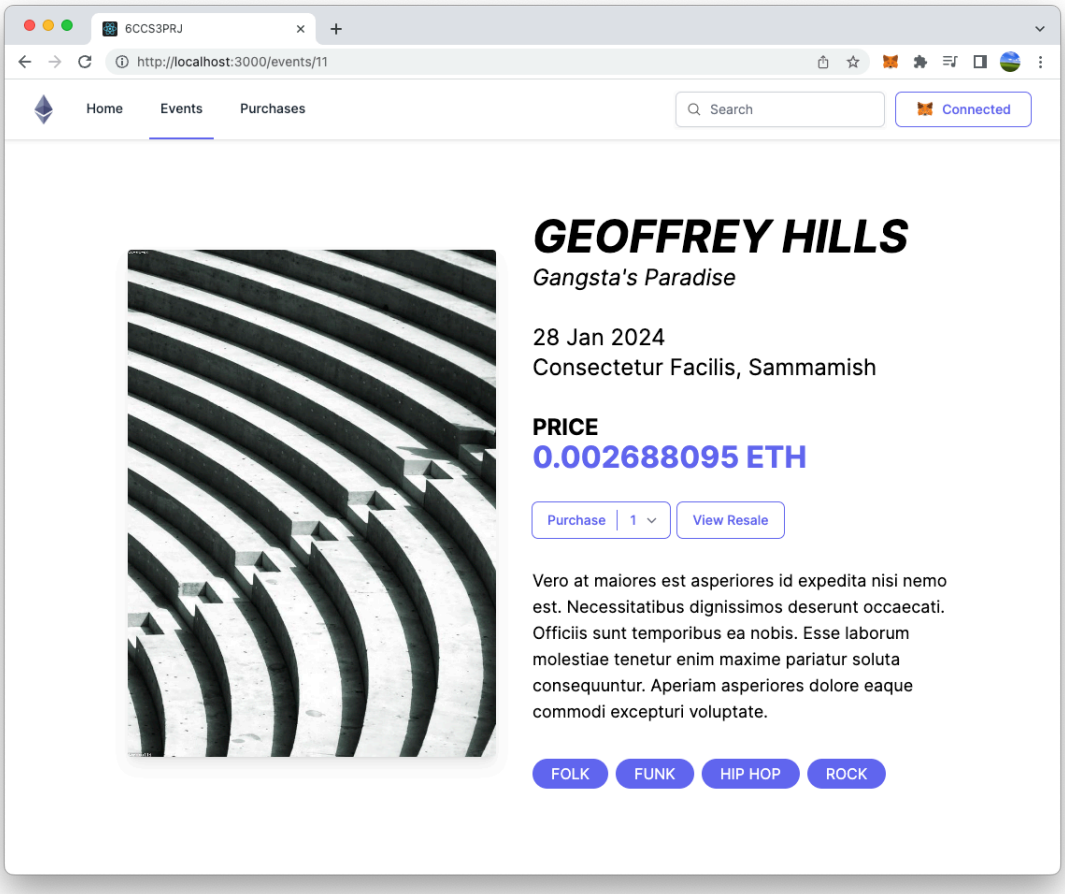


**Figure B2.5: Application Event Page**

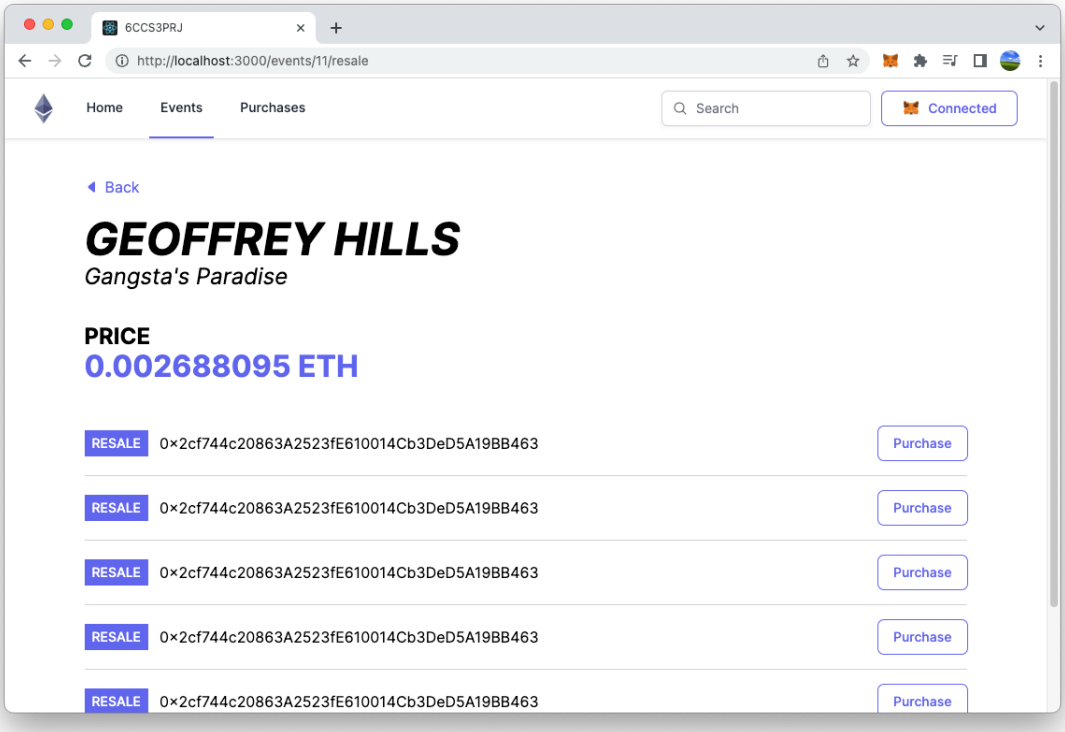


**Figure B2.6: Application Event Resale Page**

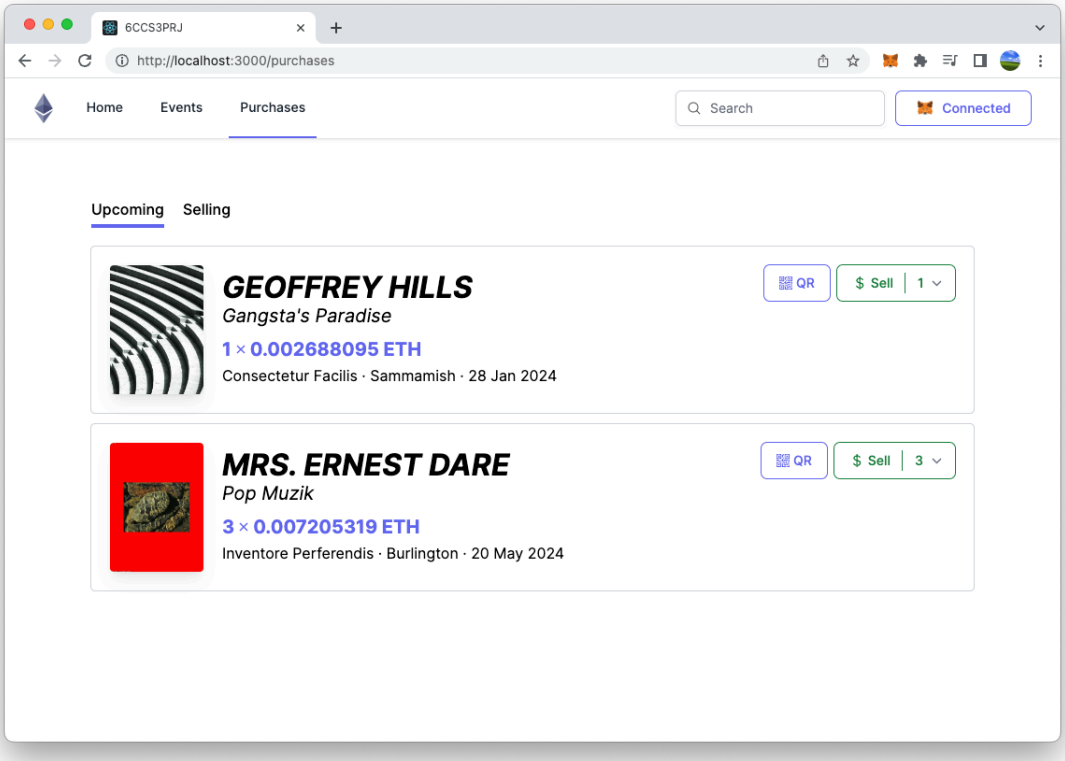


**Figure B2.7: Application Purchases Page**

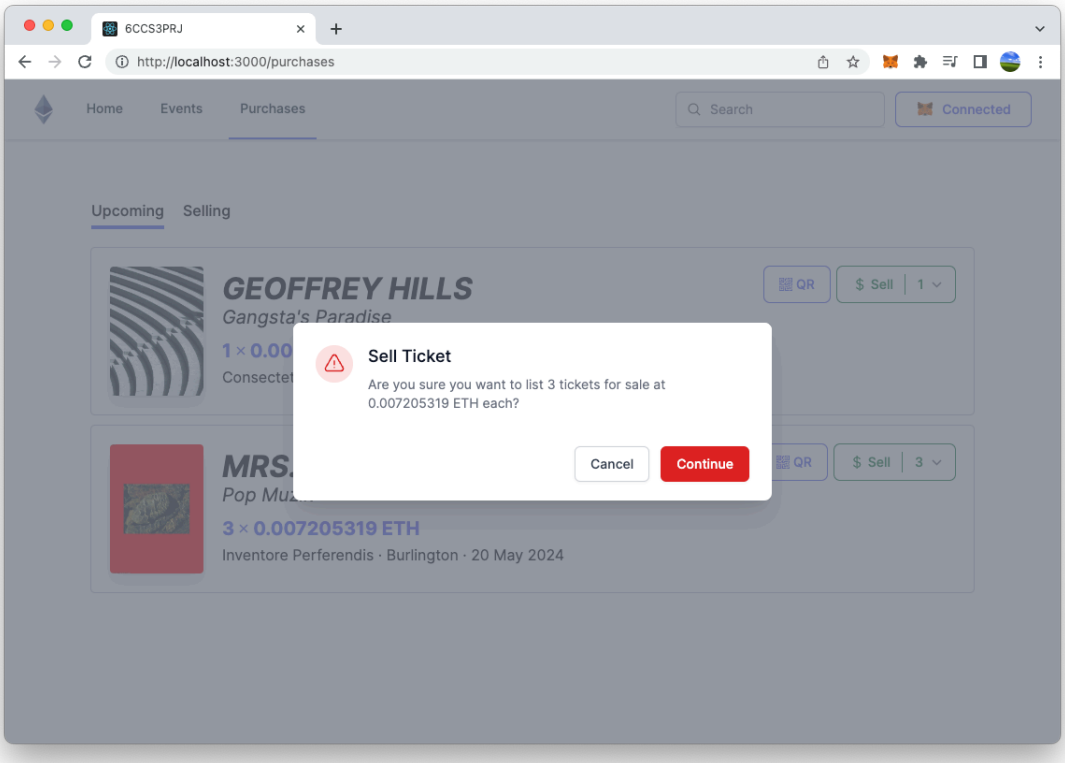


**Figure B2.8: Application Ticket Resale Confirmation**

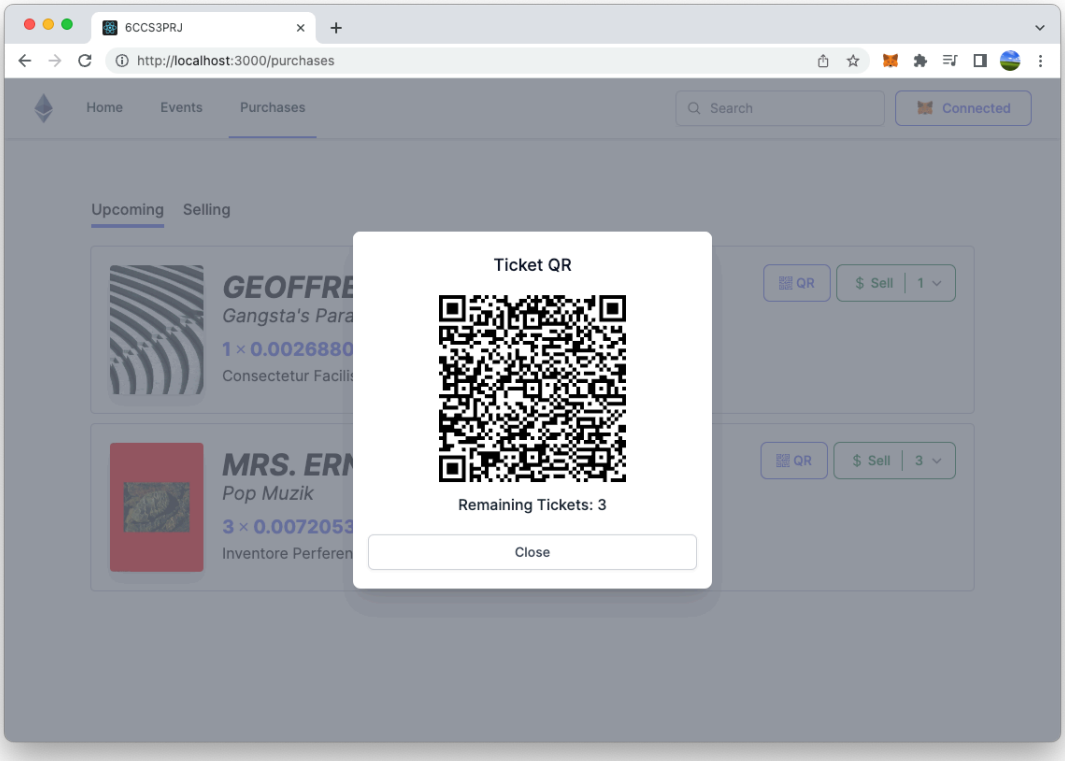


**Figure B2.9: Application Ticket QR Code Modal**

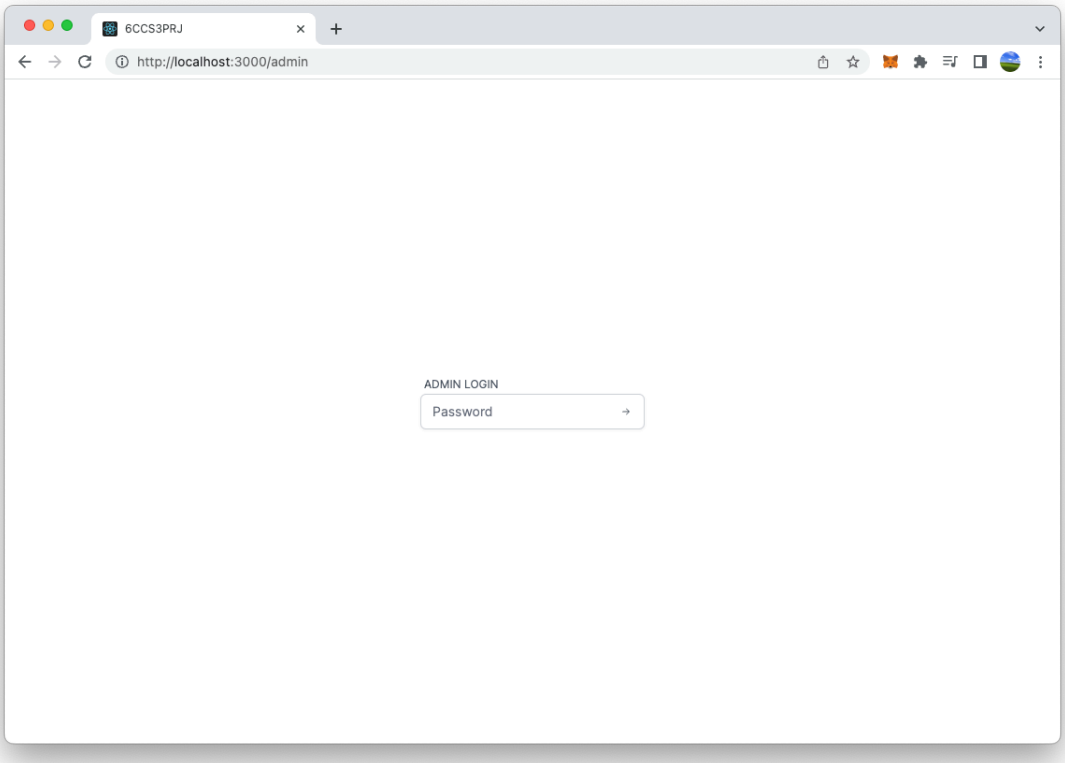


**Figure B2.10: Application Admin Interface Login Page**

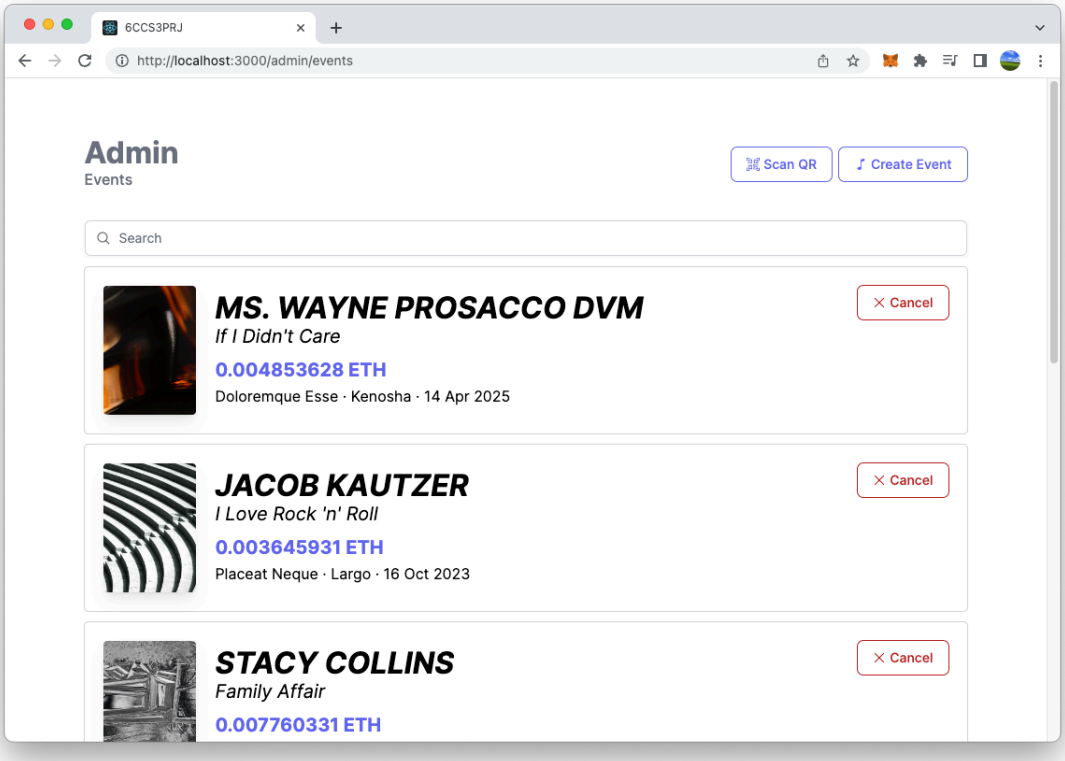


**Figure B2.11: Application Admin Interface Events Page**

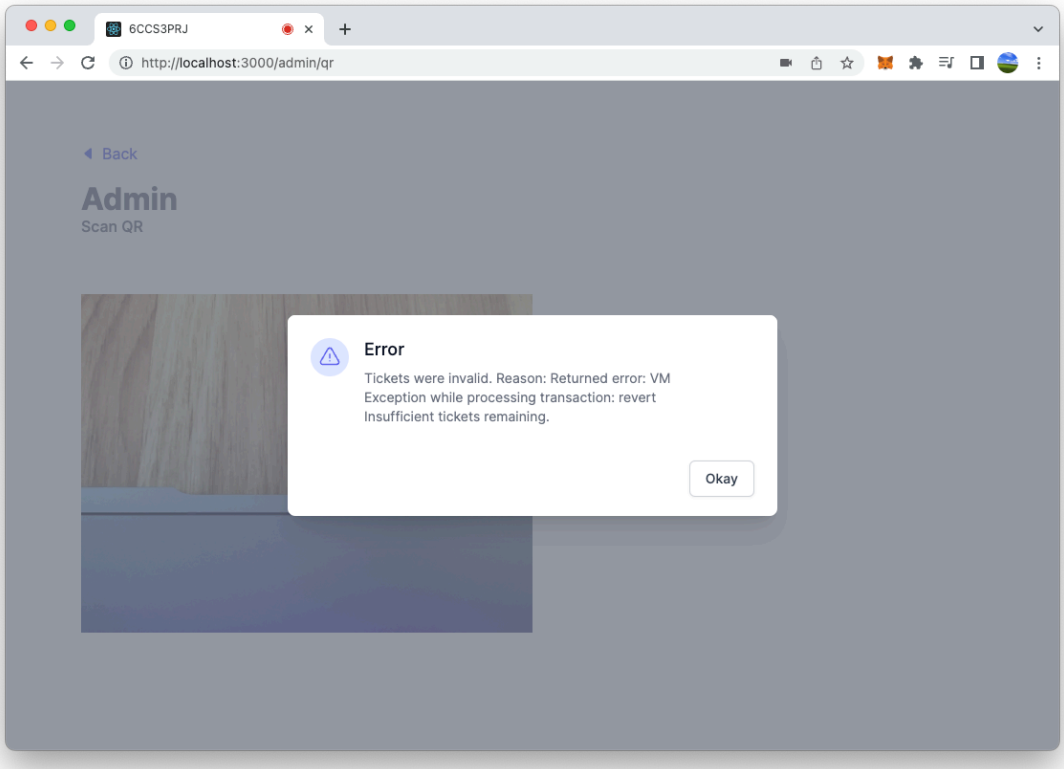


**Figure B2.12: Application Error Prompt Example**

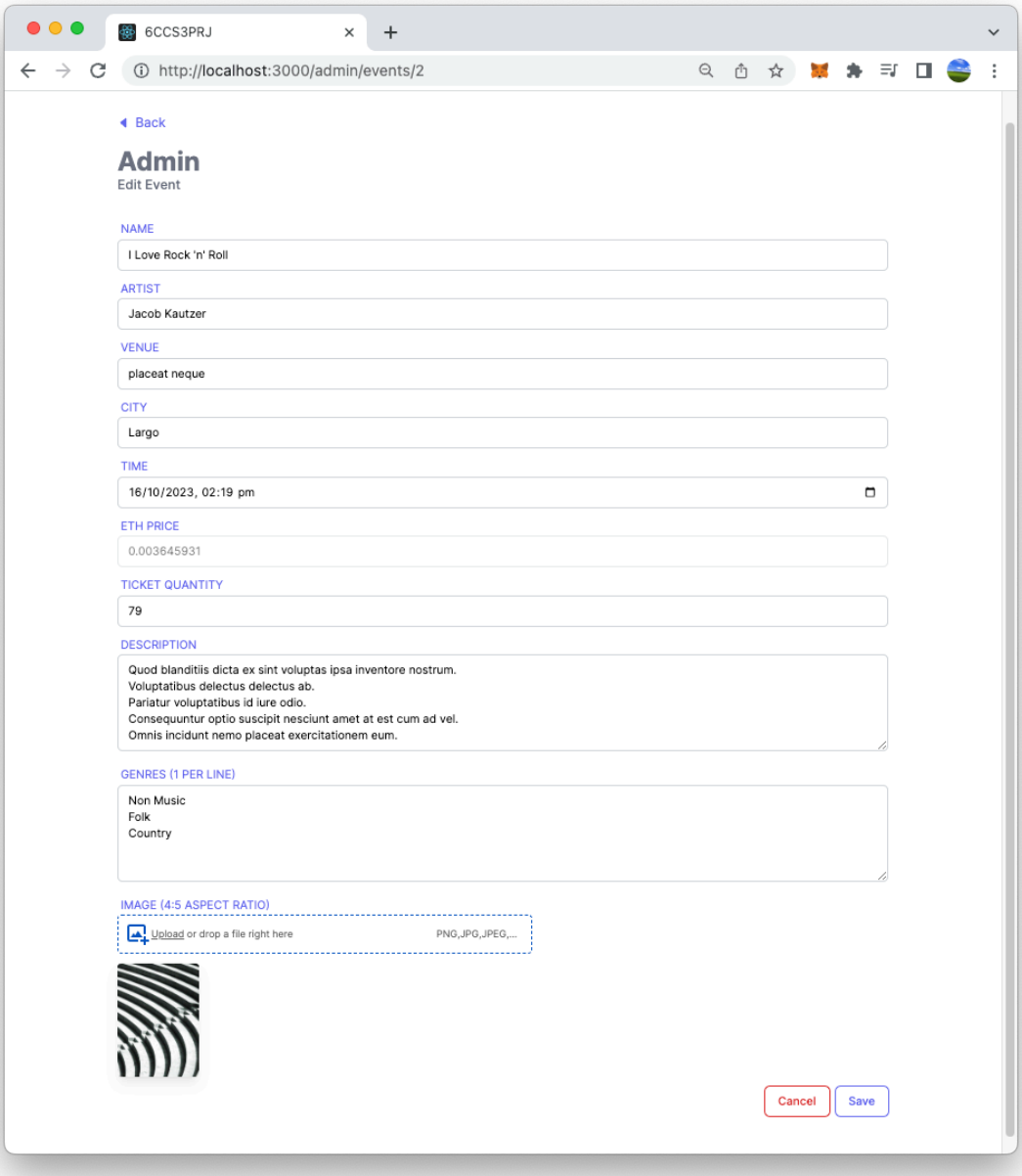


**Figure B2.13: Application Admin Interface Create/Edit Event Page**

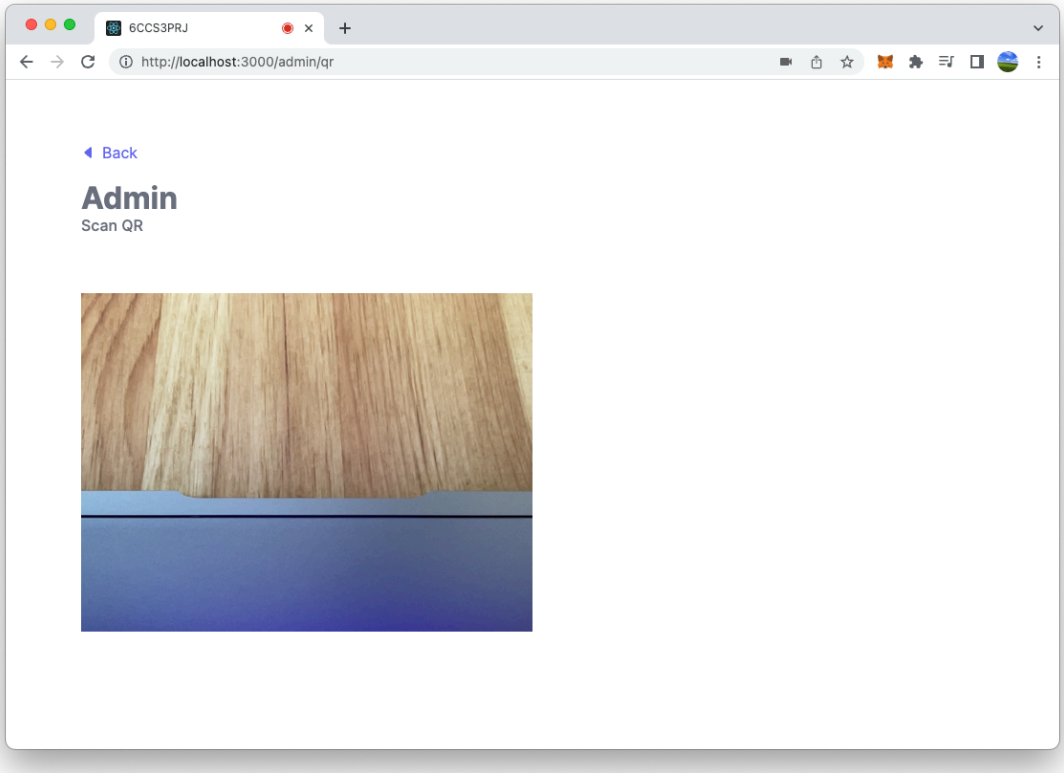


**Figure B2.14: Application Admin Interface Scan QR Page**

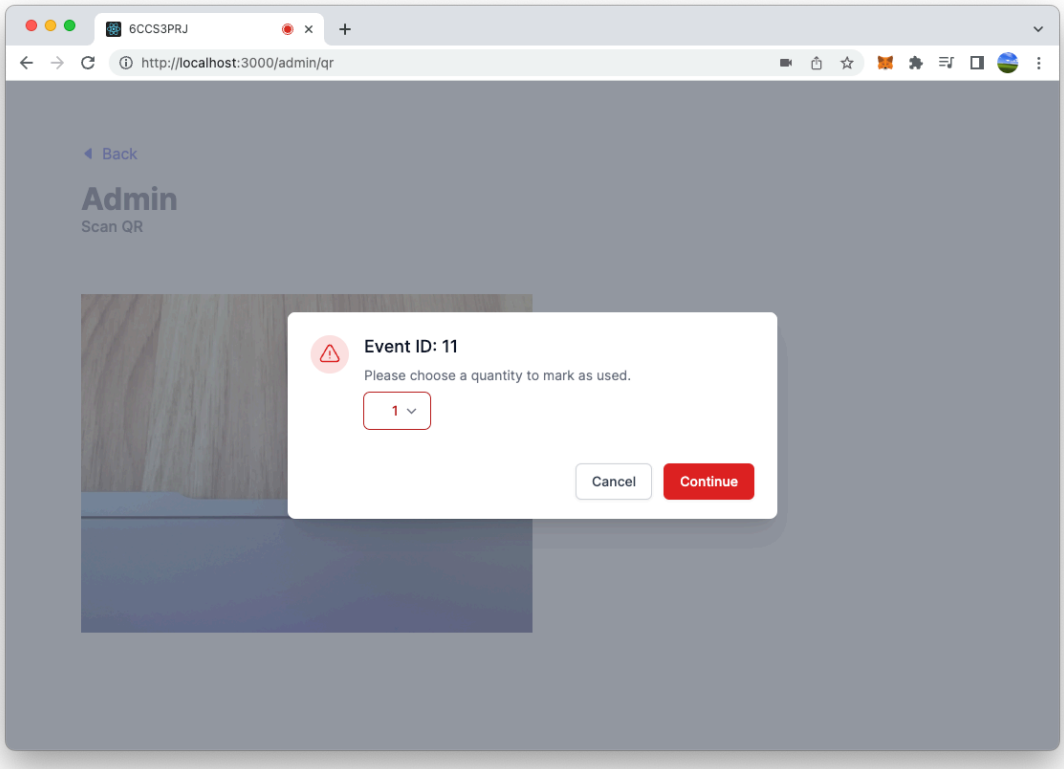


**Figure B2.15: Application Admin Interface Use Ticket Confirmation Modal**

# Appendix C: User Guide

Instructions to run the application locally is detailed in the project code's `README` file. This application has also been deployed to a live server and can be accessed at https://muddy-sunset-2817.fly.dev/.

## C1. Setting up your MetaMask Wallet

In order to perform transactions and interact with the smart contract, the MetaMask browser extension is used. To set this up:

1. Download MetaMask 10.27.0 from https://github.com/MetaMask/metamask-extension/releases/download/v10.27.0/metamask-chrome-10.27.0.zip.

2. On Google Chrome, visit `chrome://extensions` and drag the downloaded zip file into this window. This should install the MetaMask browser extension.

3. Once installed, you should be prompted to create a wallet by MetaMask. If not, you can manually trigger this by clicking the MetaMask icon from your browser toolbar; you may need to customise your browser toolbar to show MetaMask. Follow the instructions to create your wallet.

4. If you are using the live deployment version of the application, you will need to obtain test currency. Visit the Goerli faucet at https://goerlifaucet.com/ to receive test ETH. You will need to enter the wallet address shown in MetaMask (Figure C1.1). Proceed to skip to step 5.

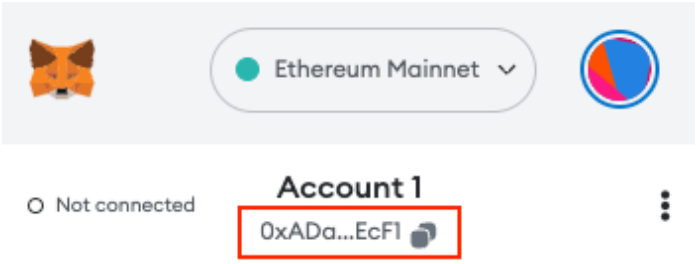


**Figure C1.1: MetaMask Wallet Address**

4. If you are using a locally run version of the application, you will need to import the test accounts with preloaded ETH. This can be done by importing accounts into MetaMask (Figure C1.2). The preloaded ETH will not show in the wallet until you switch to the Ethereum test network, which will be handled later when connecting your wallet to the application. There are ten available test accounts you can import, with their private keys listed below:

   - `0×4397dbd437030df7c8ed3cb213f34aeea9786debd4bbd62767021eb19ae7d345`
   - `0×59025abe73ea743b663276546cfdefe7e6ceab2618d3d851c0053228d21b3e89`
   - `0×98545d87cbdb20fb4198b407d45c19a6ae476e7b327ec2511b11df34d9e0e36d`
   - `0×97b543ccd0b333450872f010287211f54b7442c261a956181cc5d7599825bc0d`
   - `0×ae8e040c53e069b017b7582db2e4ea368a07a90806488b1c7d0f0d3bf23b79df`
   - `0×39bc36a89c1b732127bca39030874f30222dfad8415f102af6feaf5b60d6a6af`
   - `0×f927e60ffbb8a1965362ee4a97ccdea87cf9bb3a703620e37cf61f13cfbfdcd7`
   - `0×206a85417e715f3b8e99b69c990ecc97db264cc36cb901e8beddb4aeb08d1b9e`
   - `0×efeb100ca1c3763ccf962d09f4fe863f4c6c80238987f87d57fe9dbd0c012bb4`
   - `0×dba880a5c0554e06487b38b0105702fe2b2d387dded8066a68b8885869e70ab2`

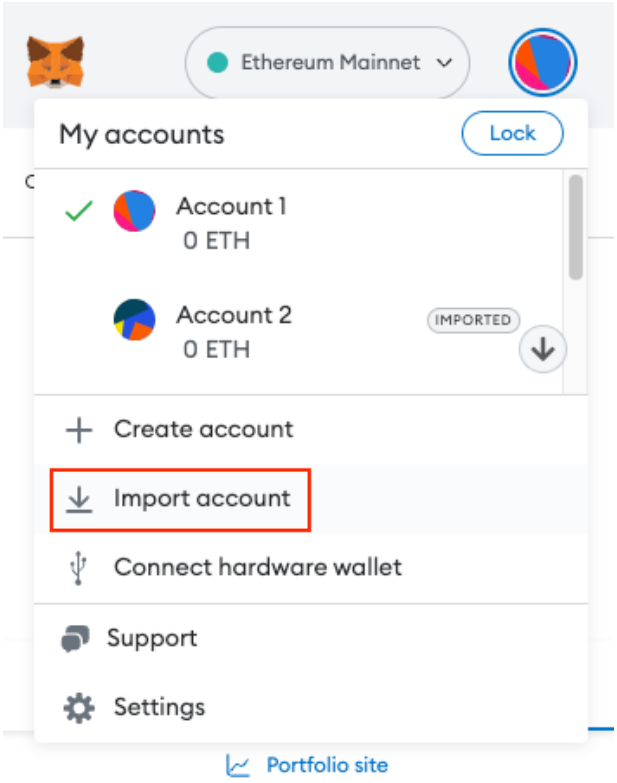


**Figure C1.2: MetaMask Import Account**

5. Visit or refresh the web application. You should now be able to connect your MetaMask wallet to the application by pressing the 'Connect Wallet' button located in the top right of the page. MetaMask may prompt you to switch Ethereum networks and authorise the connection.

## C2. Making Purchases

1. Browse for events by clicking the 'Events' tab in the navigation bar. Here you will be able to filter events based on genre, location and maximum price. To search for events, use the search bar located in the navigation bar.

2. Once you have selected an event, you should be able to press the 'Purchase' button to purchase a ticket for the event. A 'View Resale' button will appear if there are resale tickets available for the event. This

will take you to a page that lists all available resale tickets that can be purchased.

### C3. Viewing Purchases

- Purchases can be viewed by clicking the 'Purchases' tab in the navigation bar.
- Sort purchases by upcoming, selling, cancelled or expired by switching between the tabs. Only the relevant tabs will be shown.
- List or unlist unused, upcoming tickets for resale.
- View the QR code for unused, upcoming tickets.

### C4. Admin Interface

The admin interface can be accessed at the "/admin" path of the application. If running locally, http://localhost:3000/admin/ or https://muddy-sunset-2817.fly.dev/admin/ for the deployed version. The admin password has been set to "kcladmin".

- To cancel an event, press the "Cancel" button on the event listing. This will refund ticket holders and prevent users from purchasing tickets for the event.
- To edit an event, click on the event and update the details using the form. On the deployed version, image upload has been silently disabled due to persistent storage limitations.
- To scan a QR code for a ticket, press the 'Scan QR' button in the top right of the admin events page. Hold the QR code so that it is visible in the viewfinder of the camera. Once a valid ticket is scanned, a prompt will appear to confirm how many tickets you want to mark as used.

## C5. Troubleshooting

- If you have deployed locally more than once, the nonce MetaMask uses for the local blockchain can become out of sync. This will cause transactions to hang indefinitely. This can be fixed by reinstalling the MetaMask extension.

- Due to persistent storage limitations with the live deployment, image upload for creating or editing events has been disabled. To mitigate this, newly created events are given a random generic image and editing images of existing events will have no effect.

- On the live deployment, the smart contract has been deployed using my personal Ethereum wallet, so please be conservative when performing administrative actions (creating/cancelling/editing events) as these operations will use ETH from my wallet. In the case my wallet does run out of ETH, you can use the Goerli faucet to send ETH to `0×ADa95D02B0DAb0d52CCDDa4b9fDFa1a6d068EcF1`.

# Appendix D: Program Listing

I verify that I am the sole author of the programs contained in this folder, except where explicitly stated to the contrary.

Henry Li

April 2023

## Open Source Work Used

- React. [Online] https://reactjs.org/.
- Tailwind CSS. [Online] https://tailwindcss.com/.
- Bootstrap Icons. [Online] https://icons.getbootstrap.com/.
- React Router. [Online] https://reactrouter.com/en/main/.
- Web3.js. [Online] https://web3js.org/.
- useHooks. [Online] https://usehooks.com/.
- Axios. [Online] https://axios-http.com/.
- rosskhanas. react-qr-code. GitHub. [Online] https://github.com/rosskhanas/react-qr-code/.
- Formik. [Online] https://formik.org/.
- jquense. Yup. GitHub. [Online] https://github.com/jquense/yup/.
- KarimMokhtar. React Drag and Drop Files. GitHub. [Online] https://github.com/KarimMokhtar/react-drag-drop-files/.

- JodusNodus. React QR Reader. GitHub. [Online] https://github.com/JodusNodus/react-qr-reader/.
- Node.js. [Online] https://nodejs.org/en/.
- Restify. [Online] http://restify.com/.
- Knex.js. [Online] https://knexjs.org/.
- tdegrunt. jsonschema. GitHub. [Online] https://github.com/tdegrunt/jsonschema/.
- nathschmidt. restify-cookies. GitHub. [Online] https://github.com/nathschmidt/restify-cookies/.
- uuidjs. uuid. GitHub. [Online] https://github.com/uuidjs/uuid/.
- Faker. [Online] https://fakerjs.dev/.
- Truffle Suite. [Online] https://trufflesuite.com/.
- OpenZeppelin. openzeppelin-contracts. GitHub. [Online] https://github.com/OpenZeppelin/openzeppelin-contracts.

## Smart Contract

File: eth/contracts/Events.sol

```
001: // SPDX-License-Identifier: MIT
002:
003: pragma solidity ^0.8.16;
004: pragma abicoder v2;
005:
006: import "@openzeppelin/contracts/token/ERC1155/ERC1155.sol";
007: import "@openzeppelin/contracts/access/Ownable.sol";
008:
009: contract Events is ERC1155, Ownable {
010:    struct Event {
011:        uint time;
012:        uint price;
013:        uint quantity;
014:        uint supplied;
015:        bool created;
016:        bool cancelled;
017:    }
018:
019:    struct ResaleToken {
020:        address owner;
021:        bool sold;
022:    }
023:
024:    struct ResaleTokenEntry {
025:        uint eventId;
026:        bool sold;
027:        uint idx; // Index referencing the relevant `ResaleToken` in the
    `resaleTokens` mapping
028:    }
029:
030:    // Resale Token Owner Address => ResaleTokenEntry[]
031:    mapping(address => ResaleTokenEntry[]) public resaleTokenEntries;
032:
033:    // Event ID => ResaleToken[]
034:    mapping(uint => ResaleToken[]) public resaleTokens;
035:
036:    // Event ID => Event
037:    mapping(uint => Event) public events;
038:
039:    // Token Owner Address => Event IDs
040:    mapping(address => uint[]) public usedTokens;
041:
042:    constructor() ERC1155("https://muddy-sunset-2817.fly.dev/api/events/
    {id}/metadata") {
043:
044:    }
045:
046:    function getResaleTokenEntries(address _owner) public view
    returns(ResaleTokenEntry[] memory) {
047:        return resaleTokenEntries[_owner];
048:    }
049:
050:    function getResaleTokens(uint _eventId) public view
    returns(ResaleToken[] memory) {
051:        return resaleTokens[_eventId];
052:    }
053:
054:    function getUsedTokens(address _owner) public view returns(uint[]
    memory) {
055:        return usedTokens[_owner];
056:    }
057:
058:    function buyToken(uint _eventId, uint _quantity) external payable {
059:        Event storage e = events[_eventId];
060:
```

```
061:        require(e.created, "An event with this ID does not exist.");
062:        require(!e.cancelled, "This event has been cancelled.");
063:        require(msg.value >= e.price * _quantity, "Insufficient amount of
    ETH provided.");
064:        require(e.supplied + _quantity <= e.quantity, "Maximum number of
    tickets have been issued.");
065:        require(block.timestamp < e.time, "This event has already passed.");
066:
067:        e.supplied += _quantity;
068:
069:        _mint(msg.sender, _eventId, _quantity, "");
070:    }
071:
072:    function markTokenAsUsed(address _owner, uint _eventId, uint _quantity)
    external onlyOwner {
073:        Event storage e = events[_eventId];
074:
075:        require(e.created, "An event with this ID does not exist.");
076:        require(!e.cancelled, "This event has been cancelled.");
077:
078:        uint listedCount = getListedCount(_owner, _eventId);
079:        uint usedCount = getUsedCount(_owner, _eventId);
080:
081:        require(listedCount + usedCount + _quantity <= balanceOf(_owner,
    _eventId), "Insufficient tickets remaining.");
082:
083:        for (uint i = 0; i < _quantity; i++) {
084:            usedTokens[_owner].push(_eventId);
085:        }
086:    }
087:
088:    function getListedCount(address _addr, uint _eventId) public view
    returns(uint) {
089:        ResaleTokenEntry[] storage addrResaleTokenEntries =
    resaleTokenEntries[_addr];
090:        uint listedCount = 0;
091:
092:        for (uint i = 0; i < addrResaleTokenEntries.length; i++) {
093:            if (addrResaleTokenEntries[i].eventId == _eventId) {
094:                if (!addrResaleTokenEntries[i].sold) {
095:                    listedCount++;
096:                }
097:            }
098:        }
099:
100:        return listedCount;
101:    }
102:
103:    function getUsedCount(address _addr, uint _eventId) public view
    returns(uint) {
104:        uint[] storage addrUsedTokens = usedTokens[_addr];
105:        uint usedCount = 0;
106:
107:        for (uint i = 0; i < addrUsedTokens.length; i++) {
108:            if (addrUsedTokens[i] == _eventId) {
109:                usedCount++;
110:            }
111:        }
112:
113:        return usedCount;
114:    }
115:
116:    function listTokenForResale(uint _eventId, uint _quantity) external {
117:        Event storage e = events[_eventId];
118:
119:        require(e.created, "An event with this ID does not exist.");
120:        require(!e.cancelled, "This event has been cancelled.");
121:
122:        uint listedCount = getListedCount(msg.sender, _eventId);
```

```
123:        uint usedCount = getUsedCount(msg.sender, _eventId);
124:
125:        require(listedCount + usedCount + _quantity <= balanceOf(msg.sender,
    _eventId), "Insufficient number of tickets remaining.");
126:
127:        ResaleTokenEntry[] storage senderResaleTokenEntries =
    resaleTokenEntries[msg.sender];
128:
129:        for (uint i = 0; i < _quantity; i++) {
130:            resaleTokens[_eventId].push(
131:                ResaleToken({
132:                    owner: msg.sender,
133:                    sold: false
134:                })
135:            );
136:
137:            senderResaleTokenEntries.push(
138:                ResaleTokenEntry({
139:                    eventId: _eventId,
140:                    idx: resaleTokens[_eventId].length - 1,
141:                    sold: false
142:                })
143:            );
144:        }
145:    }
146:
147:    function unlistTokenForResale(uint _eventId, uint _quantity) external {
148:        Event storage e = events[_eventId];
149:
150:        require(e.created, "An event with this ID does not exist.");
151:        require(!e.cancelled, "This event has been cancelled.");
152:
153:        ResaleTokenEntry[] storage senderResaleTokenEntries =
    resaleTokenEntries[msg.sender];
154:        uint[] memory idxs = new uint[](_quantity);
155:        uint count = 0;
156:
157:        for (uint i = 0; i < senderResaleTokenEntries.length; i++) {
158:            if (senderResaleTokenEntries[i].eventId == _eventId) {
159:                if (count == _quantity) {
160:                    break;
161:                } else if (!senderResaleTokenEntries[i].sold) {
162:                    idxs[count++] = i;
163:                }
164:            }
165:        }
166:
167:        require(count == _quantity, "Quantity is too large.");
168:
169:        for (uint i = 0; i < idxs.length; i++) {
170:            senderResaleTokenEntries[idxs[i]].sold = true;
171:            resaleTokens[_eventId]
    [senderResaleTokenEntries[idxs[i]].idx].sold = true;
172:        }
173:    }
174:
175:    function buyResaleToken(address _owner, uint _eventId) external payable
    {
176:        Event storage e = events[_eventId];
177:
178:        require(e.created, "An event with this ID does not exist.");
179:        require(!e.cancelled, "This event has been cancelled.");
180:        require(msg.value >= e.price, "Insufficient amount of ETH
    provided.");
181:
182:        ResaleTokenEntry[] storage ownerResaleTokenEntries =
    resaleTokenEntries[_owner];
183:
184:        for (uint i = 0; i < ownerResaleTokenEntries.length; i++) {
```

```
185:             if (ownerResaleTokenEntries[i].eventId == _eventId) {
186:                 if (!ownerResaleTokenEntries[i].sold) {
187:                     ResaleTokenEntry storage ownerResaleTokenEntry =
    ownerResaleTokenEntries[i];
188:
189:                     _safeTransferFrom(_owner, msg.sender, _eventId, 1, "");
190:                     payable(_owner).transfer(msg.value);
191:
192:                     ownerResaleTokenEntry.sold = true;
193:                     resaleTokens[_eventId][ownerResaleTokenEntry.idx].sold =
    true;
194:
195:                     return;
196:                 }
197:             }
198:         }
199:
200:         revert("Invalid parameters.");
201:     }
202:
203:     function createEvent(uint _id, uint _time, uint _price, uint _quantity)
    external onlyOwner {
204:         Event storage e = events[_id];
205:
206:         require(!e.created, "An event with this ID has already been
    created.");
207:         require(_quantity > 0, "Quantity must be greater than 0.");
208:         require(_time > block.timestamp, "Time needs to be in the future.");
209:
210:         events[_id] = Event({
211:             time: _time,
212:             price: _price * (1 gwei),
213:             quantity: _quantity,
214:             supplied: 0,
215:             created: true,
216:             cancelled: false
217:         });
218:     }
219:
220:     function updateEvent(uint _id, uint _time, uint _quantity) external
    onlyOwner {
221:         Event storage e = events[_id];
222:
223:         require(e.created, "An event with this ID does not exist.");
224:         require(!e.cancelled, "This event has been cancelled.");
225:         require(_quantity >= e.supplied, "Quantity must be greater or equal
    than what has been supplied already.");
226:         require(_quantity > 0, "Quantity must be greater than 0.");
227:         require(_time > block.timestamp, "Time needs to be in the future.");
228:
229:         e.time = _time;
230:         e.quantity = _quantity;
231:     }
232:
233:     function cancelEvent(uint _id, address[] calldata owners, uint[]
    calldata quantity) external onlyOwner {
234:         Event storage e = events[_id];
235:         uint totalTransferAmount = 0;
236:
237:         require(e.created, "An event with this ID does not exist.");
238:         require(!e.cancelled, "This event has already been cancelled.");
239:         require(owners.length == quantity.length, "`owners` and `quantity`
    parameters must be of equal length.");
240:
241:         for (uint i = 0; i < quantity.length; i++) {
242:             totalTransferAmount +=  e.price * quantity[i];
243:         }
244:
```

```
245:         require(getBalance() >= totalTransferAmount, "Insufficient ETH in
     contract to refund owners.");
246:
247:         e.cancelled = true;
248:
249:         // Refund token owners
250:         for (uint i = 0; i < owners.length; i++) {
251:             payable(owners[i]).transfer(e.price * quantity[i]);
252:         }
253:     }
254:
255:     function getBalance() public view onlyOwner returns(uint) {
256:         return address(this).balance;
257:     }
258:
259:     function transferBalance(address payable addr) external onlyOwner {
260:         addr.transfer(getBalance());
261:     }
262: }
```

## Smart Contract Deployment Script

File: eth/migrations/1_initial_migration.js

```
1: const Events = artifacts.require("Events");
2:
3: module.exports = function(deployer, network, accounts) {
4:  deployer.deploy(Events, { from: accounts[0] });
5: };
```

## Smart Contract Unit Tests

File: eth/test/Events.test.js

```
001: const utils = require("./helpers/utils");
002: const Events = artifacts.require("Events");
003: const timeMachine = require('ganache-time-traveler');
004:
005: contract("Events", (accounts) => {
006:    const [alice, bob, charlie] = accounts;
007:    const defaultEvent = utils.defaultEvent();
008:    let contract;
009:    let snapshotId;
010:
011:    beforeEach(async () => {
012:        const snapshot = await timeMachine.takeSnapshot();
013:        snapshotId = snapshot['result'];
014:        contract = await Events.new();
015:    });
016:
017:    afterEach(async () => {
018:        await timeMachine.revertToSnapshot(snapshotId);
019:    });
020:
021:    it("deploys a contract", () => {
022:        assert.ok(contract.address);
023:    });
024:
025:    describe("createEvent", () => {
026:        it("creates an event", async () => {
027:            await utils.createEvent(contract, alice, defaultEvent);
028:
029:            const event = await
    contract.events.call(defaultEvent.id);
030:
031:            assert.equal(event.time.toNumber(), defaultEvent.time);
032:            assert.equal(BigInt(event.price),
    defaultEvent.priceInWei());
033:            assert.equal(event.quantity.toNumber(),
    defaultEvent.quantity);
034:            assert.equal(event.supplied.toNumber(), 0);
035:            assert.equal(!!event.created, true);
036:            assert.equal(!!event.cancelled, false);
037:        });
038:
039:        it("reverts if event already created", async () => {
040:            await utils.createEvent(contract, alice, defaultEvent);
041:            await utils.shouldThrow(
042:                utils.createEvent(contract, alice, defaultEvent)
043:            );
044:        });
045:
046:        it("reverts if time is in the past", async () => {
047:            await utils.shouldThrow(
048:                utils.createEvent(contract, alice, { time:
    Math.floor(Date.now() / 1000) - 1 })
049:            );
050:        });
051:
052:        it("reverts when not owner", async () => {
053:            await utils.shouldThrow(
054:                utils.createEvent(contract, bob)
055:            );
```

```
056:         });
057:
058:         it("reverts when quantity is 0", async () => {
059:             await utils.shouldThrow(
060:                 utils.createEvent(contract, alice, { quantity: 0 })
061:             );
062:         });
063:
064:         it("reverts when event with same ID exists", async () => {
065:             await utils.createEvent(contract, alice);
066:             await utils.shouldThrow(
067:                 utils.createEvent(contract, alice)
068:             );
069:         });
070:     });
071:
072:     describe("updateEvent", () => {
073:         const defaultEvent = utils.defaultEvent();
074:         const updatedEvent = {
075:             id: defaultEvent.id,
076:             time: defaultEvent.time + 1,
077:             quantity: defaultEvent.quantity + 1
078:         };
079:
080:         beforeEach(async () => {
081:             await utils.createEvent(contract, alice);
082:         });
083:
084:         it("updates an event", async () => {
085:             await contract.updateEvent.sendTransaction(
086:                 updatedEvent.id,
087:                 updatedEvent.time,
088:                 updatedEvent.quantity
089:             );
090:
091:             const event = await
    contract.events.call(defaultEvent.id);
092:
093:             assert.equal(event.time.toNumber(), updatedEvent.time);
094:             assert.equal(event.quantity.toNumber(),
    updatedEvent.quantity);
095:         });
096:
097:         it("reverts when not owner", async () => {
098:             await utils.shouldThrow(
099:                 contract.updateEvent.sendTransaction(
100:                     updatedEvent.id,
101:                     updatedEvent.time,
102:                     updatedEvent.quantity,
103:                     { from: bob }
104:                 )
105:             );
106:         });
107:
108:         it("reverts if event not created", async () => {
109:             await utils.shouldThrow(
110:                 contract.updateEvent.sendTransaction(
111:                     updatedEvent.id + 1,
112:                     updatedEvent.time,
113:                     updatedEvent.quantity
114:                 )
115:             );
116:         });
```

```
117:
118:         it("reverts if event is cancelled", async () => {
119:             await
    contract.cancelEvent.sendTransaction(updatedEvent.id, [], []);
120:             await utils.shouldThrow(
121:                 contract.updateEvent.sendTransaction(
122:                     updatedEvent.id,
123:                     updatedEvent.time,
124:                     updatedEvent.quantity
125:                 )
126:             );
127:         });
128:
129:         it("reverts if quantity is less than what has already been
    supplied", async () => {
130:             const quantity = defaultEvent.quantity;
131:
132:             await contract.buyToken.sendTransaction(
133:                 updatedEvent.id,
134:                 quantity,
135:                 { from: charlie, value: defaultEvent.priceInWei() *
    quantity }
136:             );
137:             await utils.shouldThrow(
138:                 contract.updateEvent.sendTransaction(
139:                     updatedEvent.id,
140:                     updatedEvent.time,
141:                     quantity - 1
142:                 )
143:             );
144:         });
145:
146:         it("reverts if quantity is 0", async () => {
147:             await utils.shouldThrow(
148:                 contract.updateEvent.sendTransaction(
149:                     updatedEvent.id,
150:                     updatedEvent.time,
151:                     0
152:                 )
153:             );
154:         });
155:
156:         it("reverts if time is in the past", async () => {
157:             await utils.shouldThrow(
158:                 contract.updateEvent.sendTransaction(
159:                     updatedEvent.id,
160:                     Math.floor(Date.now() / 1000) - 1,
161:                     updatedEvent.quantity
162:                 )
163:             );
164:         });
165:     });
166:
167:     describe("buyToken", () => {
168:         beforeEach(async () => {
169:             await utils.createEvent(contract, alice);
170:         });
171:
172:         it("buys a token", async () => {
173:             await contract.buyToken.sendTransaction(
174:                 defaultEvent.id,
175:                 1,
176:                 { from: charlie, value: defaultEvent.priceInWei() }
```

```
177:             );
178:
179:             utils.assertTokenCount(contract, charlie,
    defaultEvent.id, 1);
180:         });
181:
182:         it("reverts if event does not exist", async () => {
183:             await utils.shouldThrow(
184:                 contract.buyToken.sendTransaction(
185:                     defaultEvent.id + 1,
186:                     defaultEvent.quantity,
187:                     { from: charlie, value:
    defaultEvent.priceInWei() }
188:                 )
189:             );
190:
191:             utils.assertTokenCount(contract, charlie,
    defaultEvent.id, 0);
192:         });
193:
194:         it("reverts if event has been cancelled", async () => {
195:             await
    contract.cancelEvent.sendTransaction(defaultEvent.id, [], []);
196:             await utils.shouldThrow(
197:                 contract.buyToken.sendTransaction(
198:                     defaultEvent.id,
199:                     defaultEvent.quantity,
200:                     { from: charlie, value:
    defaultEvent.priceInWei() }
201:                 )
202:             );
203:
204:             utils.assertTokenCount(contract, charlie,
    defaultEvent.id, 0);
205:         });
206:
207:         it("reverts with insufficient payment", async () => {
208:             await utils.shouldThrow(
209:                 contract.buyToken.sendTransaction(
210:                     defaultEvent.id,
211:                     2,
212:                     { from: charlie, value:
    defaultEvent.priceInWei() }
213:                 )
214:             );
215:
216:             utils.assertTokenCount(contract, charlie,
    defaultEvent.id, 0);
217:         });
218:
219:         it("reverts with too much quantity", async () => {
220:             await contract.buyToken.sendTransaction(
221:                 defaultEvent.id,
222:                 defaultEvent.quantity,
223:                 { from: alice, value: defaultEvent.priceInWei() *
    defaultEvent.quantity }
224:             )
225:             await utils.shouldThrow(
226:                 contract.buyToken.sendTransaction(
227:                     defaultEvent.id,
228:                     1,
229:                     { from: charlie, value:
    defaultEvent.priceInWei() }
```

```
230:                 )
231:             );
232:
233:             utils.assertTokenCount(contract, charlie,
    defaultEvent.id, 0);
234:         });
235:
236:         it("reverts when event is in the past", async () => {
237:             const seconds = 100;
238:             const event = {
239:                 id: defaultEvent.id + 1,
240:                 time: Math.floor(Date.now() / 1000) + seconds
241:             };
242:
243:             await utils.createEvent(contract, alice, event);
244:             await timeMachine.advanceTimeAndBlock(seconds + 1);
245:             await utils.shouldThrow(
246:                 contract.buyToken.sendTransaction(
247:                     event.id,
248:                     1,
249:                     { from: charlie, value:
    defaultEvent.priceInWei() }
250:                 )
251:             );
252:
253:             utils.assertTokenCount(contract, charlie, event.id, 0);
254:         });
255:     });
256:
257:     describe("getBalance", () => {
258:         it("returns correct balance", async () => {
259:             const gwei = defaultEvent.priceInWei();
260:
261:             await utils.createEvent(contract, alice, defaultEvent);
262:             await contract.buyToken.sendTransaction(
263:                 defaultEvent.id,
264:                 1,
265:                 { from: charlie, value: gwei }
266:             );
267:
268:             const balance = await contract.getBalance.call();
269:
270:             assert.equal(BigInt(balance), gwei);
271:         });
272:
273:         it("reverts when not owner", async () => {
274:             await utils.shouldThrow(
275:                 contract.getBalance.call({ from: bob })
276:             );
277:         });
278:     });
279:
280:     describe("transferBalance", () => {
281:         it("transfers contract balance", async () => {
282:             const gwei = defaultEvent.priceInWei();
283:
284:             await utils.createEvent(contract, alice, defaultEvent);
285:             await contract.buyToken.sendTransaction(
286:                 defaultEvent.id,
287:                 1,
288:                 { from: charlie, value: gwei }
289:             );
290:
```

```
291:             const balanceBefore = await contract.getBalance.call();
292:
293:             assert.equal(BigInt(balanceBefore), gwei);
294:
295:             await contract.transferBalance.sendTransaction(charlie);
296:
297:             const balanceAfter = await contract.getBalance.call();
298:
299:             assert.equal(BigInt(balanceAfter), 0);
300:         });
301:
302:         it("reverts when not owner", async () => {
303:             await utils.shouldThrow(
304:                 contract.transferBalance.sendTransaction(charlie,
    { from: bob })
305:             );
306:         });
307:     });
308:
309:     describe("cancelEvent", () => {
310:         beforeEach(async () => {
311:             await utils.createEvent(contract, alice);
312:         });
313:
314:         const assertEventCancelled = async (id, cancelled) => {
315:             const event = await contract.events(id);
316:
317:             assert.equal(!!event.cancelled, cancelled);
318:         };
319:
320:
321:         it("cancels an event with no token owners", async () => {
322:             await
    contract.cancelEvent.sendTransaction(defaultEvent.id, [], []);
323:
324:             await assertEventCancelled(defaultEvent.id, true);
325:         });
326:
327:         it("cancels an event, refunding token owners", async () => {
328:             const quantity = defaultEvent.quantity;
329:             const price = defaultEvent.priceInWei() * quantity;
330:
331:             await contract.buyToken.sendTransaction(
332:                 defaultEvent.id,
333:                 quantity,
334:                 { from: charlie, value: price }
335:             )
336:             const charlieBalanceBefore = await
    web3.eth.getBalance(charlie);
337:
338:             await
    contract.cancelEvent.sendTransaction(defaultEvent.id, [charlie],
    [quantity]);
339:
340:             await assertEventCancelled(defaultEvent.id, true);
341:
342:             const charlieBalanceAfter = await
    web3.eth.getBalance(charlie);
343:
344:             assert.equal(BigInt(charlieBalanceBefore) +
    BigInt(price), BigInt(charlieBalanceAfter));
345:         });
346:
```

```
347:         it("reverts when there is insufficient ETH to refund token
     owners", async () => {
348:             const quantity = defaultEvent.quantity;
349:
350:             await contract.buyToken.sendTransaction(
351:                 defaultEvent.id,
352:                 quantity,
353:                 { from: charlie, value: defaultEvent.priceInWei() *
     quantity }
354:             )
355:             await contract.transferBalance(charlie);
356:             await utils.shouldThrow(
357:
     contract.cancelEvent.sendTransaction(defaultEvent.id, [charlie],
     [quantity])
358:             );
359:
360:             await assertEventCancelled(defaultEvent.id, false);
361:         });
362:
363:         it("reverts if the event does not exist", async () => {
364:             await utils.shouldThrow(
365:                 contract.cancelEvent.sendTransaction(defaultEvent.id
     + 1, [], [])
366:             );
367:
368:             await assertEventCancelled(defaultEvent.id, false);
369:         });
370:
371:         it("reverts if the event has already been cancelled", async
     () => {
372:             await
     contract.cancelEvent.sendTransaction(defaultEvent.id, [], []);
373:             await utils.shouldThrow(
374:
     contract.cancelEvent.sendTransaction(defaultEvent.id, [], [])
375:             );
376:         });
377:
378:         it("reverts if `owners` and `quantity` parameters are of
     different length", async () => {
379:             await utils.shouldThrow(
380:
     contract.cancelEvent.sendTransaction(defaultEvent.id, [charlie,
     bob], [1])
381:             );
382:             await utils.shouldThrow(
383:
     contract.cancelEvent.sendTransaction(defaultEvent.id, [charlie,
     bob], [1, 2, 3])
384:             );
385:
386:             await assertEventCancelled(defaultEvent.id, false);
387:         });
388:
389:         it("reverts when not owner", async () => {
390:             await utils.shouldThrow(
391:
     contract.cancelEvent.sendTransaction(defaultEvent.id, [], [],
     { from: bob })
392:             );
393:         });
394:     });
```

```
395:
396:     describe("markTokenAsUsed", () => {
397:         const quantity = defaultEvent.quantity;
398:         const buyer = charlie;
399:
400:         beforeEach(async () => {
401:             await utils.createEvent(contract, alice);
402:             await contract.buyToken.sendTransaction(
403:                 defaultEvent.id,
404:                 quantity,
405:                 { from: buyer, value: defaultEvent.priceInWei() *
    quantity }
406:             )
407:         });
408:
409:         it("marks tokens as used", async () => {
410:             await contract.markTokenAsUsed.sendTransaction(
411:                 charlie,
412:                 defaultEvent.id,
413:                 quantity
414:             );
415:
416:             const usedTokens = await
    contract.getUsedTokens.call(charlie);
417:
418:             for (const usedToken of usedTokens) {
419:                 assert.equal(usedToken.words[0], defaultEvent.id);
420:             }
421:         });
422:
423:         it("reverts if not owner", async () => {
424:             await utils.shouldThrow(
425:                 contract.markTokenAsUsed.sendTransaction(
426:                     charlie,
427:                     defaultEvent.id,
428:                     quantity,
429:                     { from: bob }
430:                 )
431:             );
432:         });
433:
434:         it("reverts if event does not exist", async () => {
435:             await utils.shouldThrow(
436:                 contract.markTokenAsUsed.sendTransaction(
437:                     charlie,
438:                     defaultEvent.id + 1,
439:                     0
440:                 )
441:             );
442:         });
443:
444:         it("reverts if event has been cancelled", async () => {
445:             await
    contract.cancelEvent.sendTransaction(defaultEvent.id, [charlie],
    [quantity]);
446:             await utils.shouldThrow(
447:                 contract.markTokenAsUsed.sendTransaction(
448:                     charlie,
449:                     defaultEvent.id,
450:                     quantity
451:                 )
452:             );
453:         });
```

```
454:
455:         it("reverts if there are not enough tokens to mark as used",
     async () => {
456:             await contract.markTokenAsUsed.sendTransaction(
457:                 charlie,
458:                 defaultEvent.id,
459:                 quantity
460:             );
461:             await utils.shouldThrow(
462:                 contract.markTokenAsUsed.sendTransaction(
463:                     charlie,
464:                     defaultEvent.id,
465:                     1
466:                 )
467:             );
468:         });
469:     });
470:
471:     describe("getUsedCount", () => {
472:         beforeEach(async () => {
473:             await utils.createEvent(contract, alice);
474:         });
475:
476:         it("gets the correct used count for an address", async () =>
     {
477:             const buyer = charlie;
478:             const quantity = defaultEvent.quantity;
479:             const usedCountBefore = await
     contract.getUsedCount.call(buyer, defaultEvent.id);
480:
481:             assert.equal(usedCountBefore.toNumber(), 0);
482:
483:             await contract.buyToken.sendTransaction(
484:                 defaultEvent.id,
485:                 quantity,
486:                 { from: charlie, value: defaultEvent.priceInWei() *
     quantity }
487:             );
488:             await contract.markTokenAsUsed.sendTransaction(charlie,
     defaultEvent.id, quantity);
489:
490:             const usedCountAfter = await
     contract.getUsedCount.call(buyer, defaultEvent.id);
491:
492:             assert.equal(usedCountAfter.toNumber(), quantity);
493:         });
494:
495:         it("gets the correct used count for an address, for a
     specific event", async () => {
496:             const buyer = charlie;
497:             const quantity = defaultEvent.quantity;
498:             const secondEventId = defaultEvent.id + 1;
499:
500:             await utils.createEvent(contract, alice, { id:
     secondEventId });
501:             await contract.buyToken.sendTransaction(
502:                 defaultEvent.id,
503:                 quantity,
504:                 { from: charlie, value: defaultEvent.priceInWei() *
     quantity }
505:             );
506:             await contract.buyToken.sendTransaction(
507:                 secondEventId,
```

```
508:                 quantity,
509:                 { from: charlie, value: defaultEvent.priceInWei() *
    quantity }
510:             );
511:             await contract.markTokenAsUsed.sendTransaction(charlie,
    defaultEvent.id, quantity);
512:             await contract.markTokenAsUsed.sendTransaction(charlie,
    secondEventId, quantity);
513:
514:             const usedCount = await
    contract.getUsedCount.call(buyer, defaultEvent.id);
515:
516:             assert.equal(usedCount.toNumber(), quantity);
517:         });
518:     });
519:
520:     describe("getUsedTokens", async () => {
521:         it("gets the used tokens for an address", async () => {
522:             const buyer = charlie;
523:             const quantity = defaultEvent.quantity;
524:             const usedTokensBefore = await
    contract.getUsedTokens.call(buyer);
525:
526:             assert.equal(usedTokensBefore.length, 0);
527:
528:             await utils.createEvent(contract, alice);
529:             await contract.buyToken.sendTransaction(
530:                 defaultEvent.id,
531:                 quantity,
532:                 { from: charlie, value: defaultEvent.priceInWei() *
    quantity }
533:             );
534:             await contract.markTokenAsUsed.sendTransaction(charlie,
    defaultEvent.id, quantity);
535:
536:             const usedTokensAfter = await
    contract.getUsedTokens.call(buyer);
537:
538:             assert.equal(usedTokensAfter.length, quantity);
539:
540:             for (const usedTokenAfter of usedTokensAfter) {
541:                 assert.equal(usedTokenAfter.words[0],
    defaultEvent.id);
542:             }
543:         });
544:     });
545:
546:     describe("listTokenForResale", async () => {
547:         const buyer = charlie;
548:         const quantity = defaultEvent.quantity;
549:
550:         beforeEach(async () => {
551:             await utils.createEvent(contract, alice);
552:             await contract.buyToken.sendTransaction(
553:                 defaultEvent.id,
554:                 quantity,
555:                 { from: buyer, value: defaultEvent.priceInWei() *
    quantity }
556:             );
557:         });
558:
559:         it("lists tokens for resale", async () => {
560:             await contract.listTokenForResale.sendTransaction(
```

```
561:                 defaultEvent.id,
562:                 quantity,
563:                 { from: buyer }
564:             );
565:             await utils.assertTokenListedForResale(contract,
    defaultEvent.id, buyer, quantity);
566:         });
567:
568:         it("succeeds with quantity 0", async () => {
569:             const quantity = 0;
570:
571:             await contract.listTokenForResale.sendTransaction(
572:                 defaultEvent.id,
573:                 quantity,
574:                 { from: buyer }
575:             );
576:             await utils.assertTokenListedForResale(contract,
    defaultEvent.id, buyer, quantity);
577:         });
578:
579:         it("reverts if the event does not exist", async () => {
580:             const falseEventId = defaultEvent.id + 1;
581:             const quantity = 0;
582:
583:             await utils.shouldThrow(
584:                 contract.listTokenForResale.sendTransaction(
585:                     falseEventId,
586:                     quantity,
587:                     { from: buyer }
588:                 )
589:             );
590:             await utils.assertTokenListedForResale(contract,
    falseEventId, buyer, quantity);
591:         });
592:
593:         it("reverts if the event has been cancelled", async () => {
594:             await
    contract.cancelEvent.sendTransaction(defaultEvent.id, [], []);
595:             await utils.shouldThrow(
596:                 contract.listTokenForResale.sendTransaction(
597:                     defaultEvent.id,
598:                     quantity,
599:                     { from: buyer }
600:                 )
601:             );
602:             await utils.assertTokenListedForResale(contract,
    defaultEvent.id, buyer, 0);
603:         });
604:
605:         it("reverts if trying to list more than available", async ()
    => {
606:             await utils.shouldThrow(
607:                 contract.listTokenForResale.sendTransaction(
608:                     defaultEvent.id,
609:                     quantity + 1,
610:                     { from: buyer }
611:                 )
612:             );
613:             await contract.markTokenAsUsed.sendTransaction(buyer,
    defaultEvent.id, quantity);
614:             await utils.shouldThrow(
615:                 contract.listTokenForResale.sendTransaction(
616:                     defaultEvent.id,
```

```
617:                     quantity,
618:                     { from: buyer }
619:                 )
620:             );
621:
622:             await utils.assertTokenListedForResale(contract,
    defaultEvent.id, buyer, 0);
623:         });
624:     });
625:
626:     describe("unlistTokenForResale", async () => {
627:         const secondEventId = 999;
628:         const buyer = charlie;
629:         const listedQuantity = 2;
630:
631:         beforeEach(async () => {
632:             await utils.createEvent(contract, alice);
633:             await utils.createEvent(contract, alice, { id:
    secondEventId });
634:
635:             for (const eventId of [defaultEvent.id, secondEventId])
    {
636:                 await contract.buyToken.sendTransaction(
637:                     eventId,
638:                     listedQuantity,
639:                     { from: buyer, value: defaultEvent.priceInWei()
    * listedQuantity }
640:                 );
641:
642:                 await contract.listTokenForResale.sendTransaction(
643:                     eventId,
644:                     listedQuantity,
645:                     { from: buyer }
646:                 );
647:             }
648:         });
649:
650:         it("unlists tokens for resale", async () => {
651:             for (let _ = 0; _ < listedQuantity; _++) {
652:                 await contract.unlistTokenForResale.sendTransaction(
653:                     defaultEvent.id,
654:                     1,
655:                     { from: buyer }
656:                 );
657:             }
658:
659:             await utils.assertTokenListedForResale(contract,
    defaultEvent.id, buyer, 0);
660:         });
661:
662:         it("unlists multiple tokens for resale", async () => {
663:             await contract.unlistTokenForResale.sendTransaction(
664:                 defaultEvent.id,
665:                 listedQuantity,
666:                 { from: buyer }
667:             );
668:
669:             await utils.assertTokenListedForResale(contract,
    defaultEvent.id, buyer, 0);
670:         });
671:
672:         it("succeeds with quantity 0", async () => {
673:             await contract.unlistTokenForResale.sendTransaction(
```

```
674:                 defaultEvent.id,
675:                 0,
676:                 { from: buyer }
677:             );
678:
679:             await utils.assertTokenListedForResale(contract,
    defaultEvent.id, buyer, listedQuantity);
680:         });
681:
682:         it("reverts if event does not exist", async () => {
683:             await utils.shouldThrow(
684:                 contract.unlistTokenForResale.sendTransaction(
685:                     defaultEvent.id + 1,
686:                     0,
687:                     { from: buyer }
688:                 )
689:             );
690:         });
691:
692:         it("reverts if event has been cancelled", async () => {
693:             await
    contract.cancelEvent.sendTransaction(defaultEvent.id, [], []);
694:             await utils.shouldThrow(
695:                 contract.unlistTokenForResale.sendTransaction(
696:                     defaultEvent.id,
697:                     listedQuantity,
698:                     { from: buyer }
699:                 )
700:             );
701:
702:             await utils.assertTokenListedForResale(contract,
    defaultEvent.id, buyer, listedQuantity);
703:         });
704:
705:         it("reverts if quantity more than what has been listed",
    async () => {
706:             await utils.shouldThrow(
707:                 contract.unlistTokenForResale.sendTransaction(
708:                     defaultEvent.id,
709:                     listedQuantity + 1,
710:                     { from: buyer }
711:                 )
712:             );
713:
714:             await utils.assertTokenListedForResale(contract,
    defaultEvent.id, buyer, listedQuantity);
715:         });
716:     });
717:
718:     describe("buyResaleToken", async () => {
719:         const buyer = bob;
720:         const reseller = charlie;
721:         const secondEventId = 999;
722:         const listedQuantity = defaultEvent.quantity;
723:
724:         beforeEach(async () => {
725:             await utils.createEvent(contract, alice);
726:             await utils.createEvent(contract, alice, { id:
    secondEventId });
727:
728:             for (const eventId of [defaultEvent.id, secondEventId])
    {
729:                 await contract.buyToken.sendTransaction(
```

```
730:                     eventId,
731:                     listedQuantity,
732:                     { from: reseller, value:
    defaultEvent.priceInWei() * listedQuantity }
733:                 );
734:
735:                 await contract.listTokenForResale.sendTransaction(
736:                     eventId,
737:                     listedQuantity,
738:                     { from: reseller }
739:                 );
740:             }
741:         });
742:
743:         it("buys a token listed for resale", async () => {
744:             await contract.unlistTokenForResale.sendTransaction(
745:                 defaultEvent.id,
746:                 1,
747:                 { from: reseller }
748:             );
749:             await contract.buyResaleToken.sendTransaction(
750:                 reseller,
751:                 defaultEvent.id,
752:                 { from: buyer, value: defaultEvent.priceInWei() }
753:             );
754:
755:             await utils.assertTokenCount(contract, bob,
    defaultEvent.id, 1);
756:         });
757:
758:         it("reverts if event does not exist", async () => {
759:             const falseEventId = defaultEvent.id + 1;
760:
761:             await utils.shouldThrow(
762:                 contract.buyResaleToken.sendTransaction(
763:                     reseller,
764:                     falseEventId,
765:                     { from: buyer, value:
    defaultEvent.priceInWei() }
766:                 )
767:             );
768:
769:             await utils.assertTokenCount(contract, buyer,
    falseEventId, 0);
770:         });
771:
772:         it("reverts if event is cancelled", async () => {
773:             await
    contract.cancelEvent.sendTransaction(defaultEvent.id, [], []);
774:             await utils.shouldThrow(
775:                 contract.buyResaleToken.sendTransaction(
776:                     reseller,
777:                     defaultEvent.id,
778:                     { from: buyer, value:
    defaultEvent.priceInWei() }
779:                 )
780:             );
781:
782:             await utils.assertTokenCount(contract, buyer,
    defaultEvent.id, 0);
783:         });
784:
785:         it("reverts if insufficient payment", async () => {
```

```
786:             await utils.shouldThrow(
787:                 contract.buyResaleToken.sendTransaction(
788:                     reseller,
789:                     defaultEvent.id,
790:                     { from: buyer, value: 1 }
791:                 )
792:             );
793: 
794:             await utils.assertTokenCount(contract, buyer,
    defaultEvent.id, 0);
795:         });
796: 
797:         it("reverts if there are no tokens for resale", async () =>
    {
798:             const eventId = defaultEvent.id + 1;
799: 
800:             await utils.createEvent(contract, alice, { id:
    eventId });
801:             await utils.shouldThrow(
802:                 contract.buyResaleToken.sendTransaction(
803:                     reseller,
804:                     eventId,
805:                     { from: buyer, value:
    defaultEvent.priceInWei() }
806:                 )
807:             );
808: 
809:             await utils.assertTokenCount(contract, buyer, eventId,
    0);
810:         });
811:     });
812: 
813:     describe("getListedCount", async () => {
814:         const buyer = charlie;
815:         const secondEventId = defaultEvent.id + 1;
816: 
817:         beforeEach(async () => {
818:             await utils.createEvent(contract, alice);
819:             await utils.createEvent(contract, alice, { id:
    secondEventId });
820: 
821:             for (const eventId of [defaultEvent.id, secondEventId])
    {
822:                 await contract.buyToken.sendTransaction(
823:                     eventId,
824:                     defaultEvent.quantity,
825:                     { from: buyer, value: defaultEvent.priceInWei()
    * defaultEvent.quantity }
826:                 );
827:             }
828:         });
829: 
830:         it("gets number of tokens listed for address", async () => {
831:             const listedCountBefore = await
    contract.getListedCount.call(buyer, defaultEvent.id);
832: 
833:             assert.equal(listedCountBefore.toNumber(), 0);
834: 
835:             await contract.listTokenForResale.sendTransaction(
836:                 secondEventId,
837:                 defaultEvent.quantity,
838:                 { from: buyer }
839:             );
```

```
840:             await contract.listTokenForResale.sendTransaction(
841:                 defaultEvent.id,
842:                 defaultEvent.quantity,
843:                 { from: buyer }
844:             );
845:             await contract.unlistTokenForResale.sendTransaction(
846:                 defaultEvent.id,
847:                 defaultEvent.quantity - 1,
848:                 { from: buyer }
849:             );
850: 
851:             const listedCountAfter = await
    contract.getListedCount.call(buyer, defaultEvent.id);
852: 
853:             assert.equal(listedCountAfter.toNumber(), 1);
854:         });
855:     });
856: 
857:     describe("getResaleTokenEntries", async () => {
858:         it("gets resale token entries for an address", async () => {
859:             const reseller = charlie;
860: 
861:             await utils.createEvent(contract, alice);
862:             await contract.buyToken.sendTransaction(
863:                 defaultEvent.id,
864:                 defaultEvent.quantity,
865:                 { from: charlie, value: defaultEvent.priceInWei() *
    defaultEvent.quantity }
866:             );
867:             await contract.listTokenForResale.sendTransaction(
868:                 defaultEvent.id,
869:                 defaultEvent.quantity,
870:                 { from: charlie }
871:             );
872: 
873:             const resaleTokenEntries = await
    contract.getResaleTokenEntries.call(reseller);
874: 
875:             assert.equal(resaleTokenEntries.length,
    defaultEvent.quantity);
876: 
877:             for (const resaleTokenEntry of resaleTokenEntries) {
878:                 assert.equal(parseInt(resaleTokenEntry.eventId),
    defaultEvent.id);
879:             }
880:         });
881:     });
882: 
883:     describe("getResaleTokens", async () => {
884:         it("gets resale token entries for an event", async () => {
885:             const reseller = charlie;
886: 
887:             await utils.createEvent(contract, alice);
888:             await contract.buyToken.sendTransaction(
889:                 defaultEvent.id,
890:                 defaultEvent.quantity,
891:                 { from: charlie, value: defaultEvent.priceInWei() *
    defaultEvent.quantity }
892:             );
893:             await contract.listTokenForResale.sendTransaction(
894:                 defaultEvent.id,
895:                 defaultEvent.quantity,
896:                 { from: charlie }
```

```
897:             );
898:
899:             const resaleTokens = await
    contract.getResaleTokens.call(defaultEvent.id);
900:
901:             assert.equal(resaleTokens.length,
    defaultEvent.quantity);
902:
903:             for (const resaleToken of resaleTokens) {
904:                 assert.equal(resaleToken.owner, reseller);
905:             }
906:         });
907:     });
908: });
```

## Smart Contract Test Utilities Helper

File: eth/test/helpers/utils.js

```
01: const shouldThrow = async (promise) => {
02:     try {
03:         await promise;
04:     } catch (err) {
05:         assert(true);
06:         return;
07:     }
08:
09:     assert(false, "The contract was expected to throw an exception
    here.");
10: };
11:
12: const defaultEvent = () => {
13:     return {
14:         id: 1,
15:         name: "TestEvent1",
16:         time: Math.floor(Date.now() / 1000) + 86400, // Tomorrow
17:         price: 10000000, // 10000000 gwei / 0.01 ETH
18:         quantity: 2,
19:         priceInWei: function() {
20:             return web3.utils.toWei(this.price.toString(), "gwei");
21:         }
22:     };
23: };
24:
25: const createEvent = async (contract, caller, event) => {
26:     event = { ...defaultEvent(), ...event };
27:
28:     await contract.createEvent.sendTransaction(
29:         event.id,
30:         event.time,
31:         event.price,
32:         event.quantity,
33:         { from: caller }
34:     );
35: };
36:
37: const assertTokenListedForResale = async (contract, eventId, owner,
    quantity) => {
38:     const resaleTokens = await
    contract.getResaleTokens.call(eventId);
39:     const resaleTokenEntries = await
    contract.getResaleTokenEntries.call(owner);
40:     const activeResaleTokenEntries = resaleTokenEntries.filter(rte
    => !rte.sold && parseInt(rte.eventId) === eventId);
41:     const activeResaleTokensForOwner = resaleTokens.filter(rt => !
    rt.sold && rt.owner === owner);
42:
43:     assert.equal(activeResaleTokenEntries.length, quantity);
44:     assert.equal(activeResaleTokensForOwner.length, quantity);
45: };
46:
47: const assertTokenCount = async (contract, account, id, count) => {
48:     const tokenCount = await contract.balanceOf.call(
49:         account,
50:         id
51:     );
52:
53:     assert.equal(tokenCount.toNumber(), count);
```

```
54: };
55:
56: module.exports = {
57:     createEvent,
58:     defaultEvent,
59:     shouldThrow,
60:     assertTokenListedForResale,
61:     assertTokenCount
62: };
```

## Backend Database Schema

File: backend/migrations/schema.sql

```
01: CREATE TABLE
02:   `event-genre` (
03:     `id` integer NOT NULL PRIMARY KEY AUTOINCREMENT,
04:     `eventId` integer NOT NULL,
05:     `genreId` integer NOT NULL,
06:     UNIQUE (`eventId`, `genreId`),
07:     CONSTRAINT `event-genre_ibfk_1` FOREIGN KEY (`eventId`)
    REFERENCES `events` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
08:     CONSTRAINT `event-genre_ibfk_2` FOREIGN KEY (`genreId`)
    REFERENCES `genres` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
09:   );
10:
11: CREATE TABLE
12:   `events` (
13:     `id` integer NOT NULL PRIMARY KEY AUTOINCREMENT,
14:     `name` varchar(255) NOT NULL,
15:     `artist` varchar(255) NOT NULL,
16:     `venue` varchar(255) NOT NULL,
17:     `city` varchar(255) NOT NULL,
18:     `time` datetime NOT NULL,
19:     `price` integer NOT NULL,
20:     `imageUrl` varchar(500) NOT NULL,
21:     `description` varchar(500) NOT NULL,
22:     `deployed` integer NOT NULL,
23:     `cancelled` integer NOT NULL
24:   );
25:
26: CREATE TABLE
27:   `genres` (
28:     `id` integer NOT NULL PRIMARY KEY AUTOINCREMENT,
29:     `name` varchar(255) NOT NULL,
30:     UNIQUE (`name`)
31:   );
32:
33: CREATE INDEX "idx_event-genre_eventgenre_ibfk_1" ON "event-
    genre" (`eventId`);
34:
35: CREATE INDEX "idx_event-genre_eventgenre_ibfk_2" ON "event-
    genre" (`genreId`);
```

## Backend Contract Helper

File: backend/src/helpers/contract.js

```
001: require('dotenv').config();
002:
003: const Web3 = require("web3");
004: const HDWalletProvider = require('@truffle/hdwallet-provider');
005:
006: const web3 = new Web3(
007:    (process.env.NODE_ENV === "production")
008:    ? new HDWalletProvider(
009:        process.env.ETH_MNEUMONIC,
010:        "https://goerli.infura.io/v3/" +
     process.env.INFURA_PROJECT_ID
011:    )
012:    : new Web3.providers.HttpProvider("http://127.0.0.1:8545")
013: );
014:
015: const ABI = require("./contractABI.json");
016: const SIGNATURE_MESSAGE = Web3.utils.toHex("Please sign this
     transaction to authenticate via your Ethereum wallet.");
017: const OWNER = process.env.ETH_CONTRACT_OWNER ||
     "0x3b26935917de7f5fac60f6d15ff02b1cf468dfb0";
018: const ADDRESS = process.env.ETH_CONTRACT_ADDRESS;
019: const instance = new web3.eth.Contract(ABI, ADDRESS,
     { handleRevert: true });
020:
021: const caseInsensitiveMatch = (a, b) => a.toLowerCase() ===
     b.toLowerCase();
022:
023: const getTokens = async (address) => {
024:    const tokens = new Map();
025:    const events = await instance.getPastEvents(
026:        'TransferSingle',
027:        { fromBlock: 0, toBlock: 'latest' }
028:    );
029:
030:    for (const event of events) {
031:        const eventId = Number(event.returnValues.id);
032:        const quantity = Number(event.returnValues.value);
033:        let diff;
034:
035:        if (caseInsensitiveMatch(event.returnValues.to, address)) {
036:            diff = quantity;
037:        } else if (caseInsensitiveMatch(event.returnValues.from,
     address)) {
038:            diff = -quantity;
039:        } else {
040:            continue;
041:        }
042:
043:        tokens.set(
044:            eventId,
045:            (tokens.has(eventId)) ? (tokens.get(eventId) + diff) :
     diff
046:        );
047:    }
048:
049:    return tokens;
050: };
051:
052: const getOwners = async (eventId) => {
```

```
053:     const owners = new Map();
054:     const events = await instance.getPastEvents(
055:         'TransferSingle',
056:         { fromBlock: 0, toBlock: 'latest' }
057:     );
058:
059:     for (const event of events) {
060:         if (Number(event.returnValues.id) !== eventId) {
061:             continue;
062:         }
063:
064:         const quantity = Number(event.returnValues.value);
065:         const to = event.returnValues.to.toLowerCase();
066:         const from = event.returnValues.from.toLowerCase();
067:
068:         owners.set(
069:             to,
070:             (owners.has(to)) ? (owners.get(to) + quantity) :
    quantity
071:         );
072:
073:         owners.set(
074:             from,
075:             (owners.has(from)) ? (owners.get(from) - quantity) :
    -quantity
076:         );
077:     }
078:
079:     for (const [address, quantity] of owners) {
080:         if (quantity <= 0) {
081:             owners.delete(address);
082:         }
083:     }
084:
085:     return owners;
086: };
087:
088: const signatureToAddress = signature =>
    web3.eth.accounts.recover(SIGNATURE_MESSAGE, signature);
089:
090: const sendContractTx = async (method) => {
091:     await method.send({
092:         from: OWNER,
093:         gas: await method.estimateGas({ from: OWNER })
094:     });
095: };
096:
097: module.exports = {
098:     ABI,
099:     ADDRESS,
100:     OWNER,
101:     instance,
102:     getTokens,
103:     getOwners,
104:     signatureToAddress,
105:     sendContractTx,
106:     SIGNATURE_MESSAGE
107: };
```

## Backend Database Helper

File: backend/src/helpers/db.js

```
001: const DB_FILE = (process.env.NODE_ENV === "production") ? "/litefs/
     db.sqlite3" : "db.sqlite3";
002: const knex = require('knex')({
003:     client: "sqlite3",
004:     connection: {
005:         filename: DB_FILE
006:     },
007:     useNullAsDefault: true
008: });
009:
010: const getEvents = async (options, deployedOnly = true,
     showCancelled = false) => {
011:     const events = {};
012:     const rows = await knex
013:         .select("events.*")
014:         .select("genres.name AS genre")
015:         .table("events")
016:         .leftJoin("event-genre", "events.id", "event-genre.eventId")
017:         .leftJoin("genres", "event-genre.genreId", "genres.id")
018:         .whereRaw("events.time > unixepoch('now')")
019:         .modify(query => {
020:             if (options?.ids && options.ids.length) {
021:                 query.whereIn("events.id", options.ids);
022:             }
023:
024:             if (deployedOnly) {
025:                 query.where("events.deployed", 1);
026:             }
027:
028:             if (!showCancelled) {
029:                 query.where("events.cancelled", 0);
030:             }
031:
032:             if (options?.search) {
033:                 const search = "%" + options.search + "%";
034:
035:                 query
036:                     .where(builder => {
037:                         builder
038:                             .whereLike("genres.name", search)
039:                             .orWhereLike("events.city", search)
040:                             .orWhereLike("events.artist", search)
041:                         ;
042:                     })
043:                 ;
044:             }
045:
046:             if (options?.genres && options.genres.length) {
047:                 query.whereIn("genres.name", options.genres);
048:             }
049:
050:             if (options?.locations && options.locations.length) {
051:                 query.whereIn("events.city", options.locations);
052:             }
053:
054:             if (options?.maxPrice) {
055:                 query.where("events.price", "<=", options.maxPrice);
056:             }
057:         })
```

```
058:     ;
059:
060:     // Group genres into array for each event
061:     for (const row of rows) {
062:         if (!events[row.id]) {
063:             events[row.id] = {
064:                 ...row,
065:                 genres: row.genre ? [row.genre] : []
066:             };
067:
068:             delete events[row.id].genre;
069:         } else {
070:             events[row.id].genres.push(row.genre);
071:         }
072:     }
073:
074:     return Object.values(events);
075: };
076:
077: const getEvent = async (id, showCancelled = true, showExpired =
     true) => {
078:     let event;
079:     const rows = await knex
080:         .select("events.*")
081:         .select("genres.name AS genre")
082:         .table("events")
083:         .leftJoin("event-genre", "events.id", "event-genre.eventId")
084:         .leftJoin("genres", "event-genre.genreId", "genres.id")
085:         .where("events.id", id)
086:         .modify(query => {
087:             if (!showCancelled) {
088:                 query.where("events.cancelled", 0);
089:             }
090:
091:             if (!showExpired) {
092:                 query.whereRaw("events.time > unixepoch('now')");
093:             }
094:         })
095:     ;
096:
097:     if (!rows.length) {
098:         return false;
099:     }
100:
101:     // Group genres into array for each event
102:     for (const row of rows) {
103:         if (!event) {
104:             event = {
105:                 ...row,
106:                 genres: row.genre ? [row.genre] : []
107:             };
108:
109:             delete event.genre;
110:         } else {
111:             event.genres.push(row.genre);
112:         }
113:     }
114:
115:     return event;
116: };
117:
118: const getGenres = () => {
119:     return knex
```

```
120:         .select("name")
121:         .table("genres")
122:         .orderBy("name")
123:         .pluck("name")
124:     ;
125: };
126:
127: const getLocations = () => {
128:     return knex
129:         .distinct("city")
130:         .table("events")
131:         .orderBy("city")
132:         .pluck("city")
133:     ;
134: };
135:
136: const createEvent = async (event) => {
137:     const fields = await knex('events').insert(event);
138:     return fields.shift();
139: };
140:
141: const updateEvent = (id, event) => {
142:     return knex('events').where({ id: id }).update(event);
143: };
144:
145: const setGenresForEvent = async (eventId, eventGenres) => {
146:     for (const name of eventGenres) {
147:         await knex("genres")
148:             .insert({ name: name })
149:             .onConflict("name")
150:             .ignore()
151:         ;
152:     }
153:
154:     const genresByName = new Map();
155:     const genres = await knex
156:         .select("id")
157:         .select("name")
158:         .table("genres")
159:     ;
160:
161:     for (const genre of genres) {
162:         genresByName.set(genre.name, genre);
163:     }
164:
165:     await knex("event-genre")
166:         .where({ eventId: eventId })
167:         .del()
168:     ;
169:
170:     for (const name of eventGenres) {
171:         await knex("event-genre")
172:             .insert({
173:                 eventId: eventId,
174:                 genreId: genresByName.get(name).id
175:             })
176:             .onConflict(["eventId", "genreId"])
177:             .ignore()
178:         ;
179:     }
180: };
181:
182: const reset = async () => {
```

```
183:     await knex("events").del();
184:     await knex("genres").del();
185: };
186:
187: module.exports = {
188:     DB_FILE,
189:     getEvents,
190:     getEvent,
191:     getGenres,
192:     getLocations,
193:     createEvent,
194:     updateEvent,
195:     setGenresForEvent,
196:     reset
197: };
```

## Backend Utilities Helper

File: backend/src/helpers/utils.js

```
001: const contract = require("./contract");
002: const db = require("./db");
003: const fs = require("fs");
004: const path = require("path");
005: const Web3 = require("web3");
006:
007: const getPurchases = async (address) => {
008:     const eventsById = new Map();
009:     const resaleTokenEntriesById = new Map();
010:     const usedTokensById = new Map();
011:     const purchases = [];
012:     const tokens = await contract.getTokens(address);
013:     const resaleTokenEntries = await
     contract.instance.methods.getResaleTokenEntries(address).call();
014:     const usedTokens = await
     contract.instance.methods.getUsedTokens(address).call();
015:     const events = await db.getEvents({
016:         ids: [
017:             ...Array.from(tokens.keys()),
018:             ...resaleTokenEntries.map(r => r.eventId)
019:         ]
020:     }, false, true);
021:
022:     // Map events by ID
023:     for (const event of events) {
024:         eventsById.set(event.id, event);
025:     }
026:
027:     // Map unsold resaleTokenEntries by their event ID
028:     for (const resaleTokenEntry of resaleTokenEntries) {
029:         const key = Number(resaleTokenEntry.eventId);
030:
031:         if (!resaleTokenEntry.sold) {
032:             resaleTokenEntriesById.set(
033:                 key,
034:                 (resaleTokenEntriesById.get(key) || 0) + 1
035:             );
036:         }
037:     }
038:
039:     // Map usedTokens by their event ID
040:     for (const usedToken of usedTokens) {
041:         const key = Number(usedToken);
042:
043:         usedTokensById.set(
044:             key,
045:             (usedTokensById.get(key) || 0) + 1
046:         );
047:     }
048:
049:     // Get all purchases not listed for sale
050:     for (let [eventId, quantity] of tokens.entries()) {
051:         const event = eventsById.get(eventId);
052:         const expired = new Date(event.time) < new Date();
053:
054:         // Adjust quantity by those current listed for sale unless
     it is expired/cancelled
055:         if (!expired && !event.cancelled) {
```

```
056:             const resaleTokenEntry =
    resaleTokenEntriesById.get(eventId);
057:
058:             if (resaleTokenEntry) {
059:                 quantity -= resaleTokenEntry;
060:             }
061:         }
062:
063:         if (quantity <= 0) {
064:             continue;
065:         }
066:
067:         purchases.push({
068:             event: event,
069:             quantity: quantity,
070:             forSale: false,
071:             expired: expired,
072:             used: usedTokensById.get(eventId) || 0
073:         });
074:     }
075:
076:     // Get all purchases currently listed for sale
077:     for (const [eventId, quantity] of
    resaleTokenEntriesById.entries()) {
078:         const event = eventsById.get(eventId);
079:         const expired = new Date(event.time) < new Date();
080:
081:         if (expired || event.cancelled) {
082:             continue;
083:         }
084:
085:         if (quantity) {
086:             purchases.push({
087:                 event: eventsById.get(eventId),
088:                 quantity: quantity,
089:                 forSale: true,
090:                 expired: expired
091:             });
092:         }
093:     }
094:
095:     return purchases
096:         .filter(p => p.event.deployed) // We only care about
    deployed events
097:         .sort((a, b) => a.event.time - b.event.time) // Sort by
    event time ascending
098:     ;
099: };
100:
101: const getEvent = async (id, showCancelled = true, showExpired =
    true) => {
102:     const event = await db.getEvent(id, showCancelled, showExpired);
103:
104:     if (!event) {
105:         return false;
106:     }
107:
108:     const contractEvent = await
    contract.instance.methods.events(event.id).call();
109:
110:     return {
111:         ...event,
112:         quantity: parseInt(contractEvent.quantity),
```

```
113:         supplied: parseInt(contractEvent.supplied),
114:         remaining: parseInt(contractEvent.quantity -
     contractEvent.supplied)
115:     };
116: };
117:
118: const omit = (object, keys) => {
119:     let final = object;
120:
121:     const omitOne = (object, key) => {
122:         const { [key]: _, ...rest } = object;
123:         return rest;
124:     };
125:
126:     for (const key of keys) {
127:         final = omitOne(final, key);
128:     }
129:
130:     return final;
131: };
132:
133: const weiToEth = (wei) => Web3.utils.fromWei(wei.toString());
134: const ethToGwei = (eth) => eth * Math.pow(10, 9);
135: const random = (min, max) => Math.random() * (max - min) + min;
136:
137: const moveFile = (from, to) => {
138:     const dirname = path.dirname(to);
139:
140:     if (!fs.existsSync(dirname)) {
141:         fs.mkdirSync(dirname);
142:     }
143:
144:      const read = fs.createReadStream(from);
145:      const write = fs.createWriteStream(to);
146:
147:      return new Promise((resolve, reject) => {
148:          read.on('end', () => resolve(to));
149:          read.on('error', reject);
150:          read.pipe(write);
151:      });
152: };
153:
154: module.exports = {
155:     random,
156:     getPurchases,
157:     getEvent,
158:     omit,
159:     weiToEth,
160:     ethToGwei,
161:     moveFile
162: };
```

## Backend JSON Schema Validators Helper

File: backend/src/helpers/validators.js

```
01: const Validator = require("jsonschema").Validator;
02: const structuredClone = require("@ungap/structured-clone");
03:
04: const genresSchema = {
05:     "id": "/genres",
06:     "type": "array",
07:     "items": { "type": "string" }
08: };
09:
10: const eventSchema = {
11:     "id": "/Event",
12:     "type": "object",
13:     "properties": {
14:         "id": { "type": "number" },
15:         "name": { "type": "string" },
16:         "artist": { "type": "string" },
17:         "venue": { "type": "string" },
18:         "city": { "type": "string" },
19:         "time": { "type": "integer", "minimum": new Date().getTime()
    / 1000 },
20:         "price": { "type": "integer", "minimum": 0 },
21:         "quantity": { "type": "integer", "minimum": 1 },
22:         "description": { "type": "string" },
23:         "genres": { "$ref": "/genres" }
24:     },
25:     "required": [
26:         "id",
27:         "name",
28:         "artist",
29:         "venue",
30:         "city",
31:         "time",
32:         "price",
33:         "quantity",
34:         "description",
35:         "genres"
36:     ],
37:     "additionalProperties": false
38: };
39:
40: const createEvent = (event) => {
41:     const v = new Validator();
42:     const schema = structuredClone.default(eventSchema);
43:
44:     delete schema.properties.id;
45:     schema.required = schema.required.filter(r => r !== "id")
46:
47:     v.addSchema(genresSchema, '/genres');
48:
49:     return v.validate(event, schema);
50: };
51:
52: const updateEvent = (event) => {
53:     const v = new Validator();
54:     const schema = structuredClone.default(eventSchema);
55:
56:     delete schema.properties.price;
57:     schema.required = schema.required.filter(r => r !== "price")
58:
```

```
59:     v.addSchema(genresSchema, '/genres');
60:
61:     return v.validate(event, schema);
62: };
63:
64: const eventToken = (data) => {
65:     const v = new Validator();
66:     const schema = {
67:         "id": "/Data",
68:         "type": "object",
69:         "properties": {
70:             "signature": { "type": "string", "minLength": 132,
     "maxLength": 132 },
71:             "quantity": { "type": "integer", "minimum": 1 }
72:         },
73:         "required": [
74:             "signature",
75:             "quantity"
76:         ],
77:         "additionalProperties": false
78:     };
79:
80:     return v.validate(data, schema);
81: }
82:
83: module.exports = {
84:     createEvent,
85:     updateEvent,
86:     eventToken
87: };
```

## Backend Event Deployment Script

File: backend/src/deploy.js

```
01: require('dotenv').config()
02:
03: const db = require("./helpers/db");
04: const utils = require("./helpers/utils");
05: const contract = require("./helpers/contract");
06:
07: const main = async () => {
08:     const events = await db.getEvents({}, false);
09:
10:     for (const event of events) {
11:         await contract.sendContractTx(
12:             contract.instance.methods.createEvent(
13:                 parseInt(event.id),
14:                 parseInt(event.time / 1000),
15:                 parseInt(event.price),
16:                 parseInt(utils.random(1, 100))
17:             )
18:         );
19:
20:         await db.updateEvent(event.id, { deployed: 1 });
21:         console.log(`Deployed event ${event.id} to contract $
    {contract.ADDRESS}`);
22:     }
23: };
24:
25: main()
26:     .then(process.exit)
27:     .catch(e => { throw e })
28: ;
```

## Backend Index Entry Script

File: backend/src/index.js

```
001: require('dotenv').config();
002: console.clear();
003:
004: const ganache = require("ganache");
005: const restify = require("restify");
006: const cookies = require('restify-cookies');
007: const crypto = require('crypto');
008: const errs = require('restify-errors');
009: const db = require("./helpers/db");
010: const fs = require("fs");
011: const contract = require("./helpers/contract");
012: const utils = require("./helpers/utils");
013: const validators = require("./helpers/validators");
014: const path = require('path');
015: const { faker } = require('@faker-js/faker');
016: const { v4: uuidv4 } = require('uuid');
017: const server = restify.createServer();
018: const sessions = new Map();
019: const PRODUCTION = (process.env.NODE_ENV === "production");
020: const API_BASE = "/api";
021: const API_ADMIN_PASSWORD = "kcladmin";
022: const PORT = (PRODUCTION) ? 80 : 3001;
023: const HOST = `${(PRODUCTION) ? "https://muddy-
     sunset-2817.fly.dev" : "http://localhost"}:${PORT}`;
024: const IMG_PATH = "images";
025:
026: if (!PRODUCTION) {
027:    ganache
028:        .server({
029:            database: { dbPath: path.resolve('./ganache') },
030:            wallet: { seed: "seed" }
031:        })
032:        .listen(8545, err => {
033:            if (err) {
034:                throw err;
035:            } else {
036:                console.log("Ganache Started");
037:            }
038:        })
039:    ;
040: }
041:
042: server.listen(PORT);
043: server.use(restify.plugins.queryParser());
044: server.use(restify.plugins.bodyParser());
045: server.use(cookies.parse);
046: server.use((req, res, next) => {
047:    const sid = req.cookies.sid ||
     crypto.randomBytes(32).toString('base64');
048:
049:    res.setCookie("sid", sid);
050:
051:    if (!sessions.has(sid)) {
052:        sessions.set(sid, {});
053:    }
054:
055:    res.header("Access-Control-Allow-Origin", (PRODUCTION) ? HOST :
     "http://localhost:3000");
056:    res.header("Access-Control-Allow-Headers", "X-Requested-With");
```

```
057:     res.header("Access-Control-Allow-Credentials", "true");
058:     req.session = sessions.get(sid);
059:
060:     return next();
061: });
062:
063: server.opts("*", (req, res, next) => {
064:     res.header("Access-Control-Allow-Methods", req.header("Access-
     Control-Request-Method"));
065:     res.header("Access-Control-Allow-Headers", req.header("Access-
     Control-Request-Headers"));
066:     res.send(200);
067:     return next();
068: });
069:
070: const response = async (req, res, fn) => {
071:     try {
072:         const data = await fn(req);
073:         res.send(
074:             200,
075:             data ? JSON.parse(JSON.stringify(data)) : undefined
076:         );
077:     } catch (e) {
078:         console.error(e);
079:         res.send(
080:             (e instanceof errs.HttpError) ? e : new
     errs.InternalServerError(e.reason || e.message)
081:         );
082:     }
083: };
084:
085: server.get("*", (req, res, next) => {
086:     const options = {
087:         directory: __dirname + "/frontend"
088:     };
089:
090:     // Serve static file if exists otherwise serve frontend home
     page
091:     try {
092:         fs.readFileSync(options.directory + req.path());
093:     } catch (e) {
094:         options.file = "index.html";
095:     }
096:
097:     const handler = restify.plugins.serveStatic(options);
098:
099:     handler(req, res, next);
100: });
101:
102: server.get("/images/*", restify.plugins.serveStatic({ directory:
     __dirname + "/.." }));
103:
104: server.get(API_BASE + "/events", async (req, res) => {
105:     await response(req, res, async (req) => {
106:         const events = await db.getEvents({
107:             genres: (req.query?.genres !== "") ?
     req.query?.genres?.split(",") : null,
108:             locations: (req.query?.locations !== "") ?
     req.query?.locations?.split(",") : null,
109:             maxPrice: req.query?.maxPrice,
110:             search: req.query?.search
111:         });
112:
```

```
113:         const limit = 7;
114:         const offset = Math.max(parseInt(req.query?.offset) || 0,
     0);
115:         const slice = events.slice(offset, offset + limit);
116:
117:         return {
118:             events: slice,
119:             prevOffset: slice.length ? Math.max(offset - limit, 0) :
     0,
120:             nextOffset: (offset + limit >= events.length) ? false :
     offset + limit
121:         };
122:     });
123: });
124:
125: server.get(API_BASE + "/events/:id", async (req, res) => {
126:     await response(req, res, async (req) => {
127:         const res = await utils.getEvent(req.params.id, false,
     false);
128:
129:         if (!res) {
130:             throw new errs.NotFoundError();
131:         }
132:
133:         return res;
134:     });
135: });
136:
137: server.get(API_BASE + "/events/:id/metadata", async (req, res) => {
138:     await response(req, res, async (req) => {
139:         const event = await utils.getEvent(req.params.id);
140:
141:         return {
142:             title: event.name,
143:             description: event.description,
144:             image: event.imageUrl,
145:             properties: {
146:                 ...utils.omit(event, ["description", "name",
     "imageUrl"])
147:             }
148:         }
149:     });
150: });
151:
152: server.get(API_BASE + "/signature", async (req, res) => {
153:     await response(req, res, async (req) => {
154:         return { message: contract.SIGNATURE_MESSAGE };
155:     });
156: });
157:
158: server.post(API_BASE + "/events/:id/token", async (req, res) => {
159:     await response(req, res, async (req) => {
160:         if (!req.session.admin) {
161:             throw new errs.UnauthorizedError();
162:         }
163:
164:         const validator = validators.eventToken(req.body);
165:
166:         if (validator.errors.length) {
167:             throw new
     errs.BadRequestError(validator.errors.shift().stack);
168:         }
169:
```

```
170:         // Check the signature is a valid hex string of 132
     characters (0x prefixed)
171:         if (!(/^0x[A-Fa-f0-9]{130}$/).test(req.body.signature)) {
172:             throw new errs.BadRequestError();
173:         }
174:
175:         const address = await
     contract.signatureToAddress(req.body.signature);
176:
177:         await contract.sendContractTx(
178:             contract.instance.methods.markTokenAsUsed(
179:                 address,
180:                 parseInt(req.params.id),
181:                 parseInt(req.body.quantity)
182:             )
183:         );
184:     });
185: });
186:
187: server.get(API_BASE + "/genres", async (req, res) => {
188:     await response(req, res, async (req) => {
189:         return db.getGenres();
190:     });
191: });
192:
193: server.get(API_BASE + "/locations", async (req, res) => {
194:     await response(req, res, async (req) => {
195:         return db.getLocations();
196:     });
197: });
198:
199: server.get(API_BASE + "/purchases/:address", async (req, res) => {
200:     await response(req, res, async (req) => {
201:         return utils.getPurchases(req.params.address);
202:     });
203: });
204:
205: server.get(API_BASE + "/contract", async (req, res) => {
206:     await response(req, res, async (req) => {
207:         return {
208:             ABI: contract.ABI,
209:             address: contract.ADDRESS
210:         };
211:     });
212: });
213:
214: server.post(API_BASE + "/login", async (req, res) => {
215:     await response(req, res, async (req) => {
216:         if (req.session.admin || req.body.password ===
     API_ADMIN_PASSWORD) {
217:             req.session.admin = true;
218:         } else {
219:             throw new errs.UnauthorizedError()
220:         }
221:     });
222: });
223:
224: server.post(API_BASE + "/events", async (req, res) => {
225:     await response(req, res, async (req) => {
226:         if (!req.session.admin) {
227:             throw new errs.UnauthorizedError();
228:         }
229:
```

```
230:         // Convert FormData strings to numbers
231:         for (const key of ["price", "quantity", "time"]) {
232:             req.body[key] = Number(req.body[key]);
233:         }
234:
235:         // Convert FormData string to array, filtering empty genres
236:         req.body.genres = req.body?.genres.split(/\r?\n/).filter(g
    => !!g);
237:
238:         const validator = validators.createEvent(req.body);
239:
240:         if (validator.errors.length) {
241:             throw new
    errs.BadRequestError(validator.errors.shift().stack);
242:         }
243:
244:         if (!req.files.image) {
245:             throw new errs.BadRequestError("Image required.");
246:         }
247:
248:         const event = {
249:             id: await db.createEvent(
250:                 {
251:                     ...utils.omit(req.body, ["genres", "quantity"]),
252:                     time: new Date(req.body.time * 1000),
253:                     deployed: 0,
254:                     cancelled: 0,
255:                     imageUrl: (!PRODUCTION) ? HOST + "/" + await
    utils.moveFile(
256:                         req.files.image.path,
257:                         `${IMG_PATH}/${uuidv4()}_$
    {req.files.image.name}`
258:                     ) : faker.image.abstract(707, 976, true)
259:                 }
260:             ),
261:             ...req.body
262:         };
263:
264:         await contract.sendContractTx(
265:             contract.instance.methods.createEvent(
266:                 event.id,
267:                 event.time,
268:                 event.price,
269:                 event.quantity
270:             )
271:         );
272:
273:         await db.setGenresForEvent(event.id, event.genres);
274:         await db.updateEvent(event.id, { deployed: 1 });
275:     });
276: });
277:
278: server.put(API_BASE + "/events", async (req, res) => {
279:     await response(req, res, async (req) => {
280:         if (!req.session.admin) {
281:             throw new errs.UnauthorizedError();
282:         }
283:
284:         // Convert FormData strings to numbers
285:         for (const key of ["id", "quantity", "time"]) {
286:             req.body[key] = Number(req.body[key]);
287:         }
288:
```

```
289:         // Convert FormData string to array, filtering empty genres
290:         req.body.genres = req.body?.genres.split(/\r?\n/).filter(g
    => !!g);
291:
292:         const validator = validators.updateEvent(req.body);
293:
294:         if (validator.errors.length) {
295:             throw new
    errs.BadRequestError(validator.errors.shift().stack);
296:         }
297:
298:         const event = { ...req.body };
299:         const contractEvent = await contract
300:             .instance
301:             .methods
302:             .events(event.id)
303:             .call()
304:         ;
305:
306:         // Only update contract event if time difference is more
    than 60 seconds or quantity is different
307:         if ((event.time - contractEvent.time >= 60) ||
    event.quantity !== Number(contractEvent.quantity)) {
308:             // Attempt to update contract event first
309:             await contract.sendContractTx(
310:                 contract.instance.methods.updateEvent(
311:                     event.id,
312:                     event.time,
313:                     event.quantity
314:                 )
315:             );
316:         }
317:
318:         // Contract event update did not throw exception, update DB
    event
319:         await db.setGenresForEvent(event.id, event.genres);
320:
321:         let imageUrl;
322:
323:         if (!PRODUCTION && req.files.image) {
324:             imageUrl = HOST + "/" + await utils.moveFile(
325:                 req.files.image.path,
326:                 `${IMG_PATH}/${uuidv4()}_${req.files.image.name}`
327:             );
328:         }
329:
330:         const updatedEvent = await db.updateEvent(
331:             event.id,
332:             {
333:                 ...utils.omit(event, ["genres", "quantity"]),
334:                 time: new Date(event.time * 1000),
335:                 imageUrl: imageUrl
336:             }
337:         );
338:
339:         return updatedEvent;
340:     });
341: });
342:
343: server.del(API_BASE + "/events/:id", async (req, res) => {
344:     await response(req, res, async (req) => {
345:         if (!req.session.admin) {
346:             throw new errs.UnauthorizedError();
```

```
347:         }
348:
349:         const id = Number(req.params.id);
350:         const contractEvent = await contract
351:             .instance
352:             .methods
353:             .events(id)
354:             .call()
355:         ;
356:
357:         if (!contractEvent.created) {
358:             throw new errs.BadRequestError();
359:         }
360:
361:         if (!contractEvent.cancelled) {
362:             const owners = await contract.getOwners(id);
363:             const totalQuantity =
    Array.from(owners.values()).reduce((acc, quantity) => acc +
    quantity, 0);
364:             const contractBalance = await
    contract.instance.methods.getBalance().call({ from:
    contract.OWNER });
365:             const refundAmount = totalQuantity *
    contractEvent.price;
366:
367:             if (contractBalance < refundAmount) {
368:                 throw new errs.InternalServerError(
369:                     `Contract requires $
    {utils.weiToEth(refundAmount)} ETH to refund` +
370:                     `owners but currently only has $
    {utils.weiToEth(contractBalance)}.`
371:                 );
372:             }
373:
374:             // Attempt to update contract event first
375:             await contract.sendContractTx(
376:                 contract.instance.methods.cancelEvent(
377:                     id,
378:                     Array.from(owners.keys()),
379:                     Array.from(owners.values())
380:                 )
381:             );
382:         }
383:
384:         // Contract event update did not throw exception, update DB
    event
385:         await db.updateEvent(id, { cancelled: 1 });
386:     });
387: });
```

## Backend Event Seeding Script

File: backend/src/seed.js

```
01: require('dotenv').config()
02:
03: const db = require("./helpers/db");
04: const utils = require("./helpers/utils");
05: const { faker } = require('@faker-js/faker');
06: const SEED_COUNT = 25;
07:
08: const generateEvents = (n) => {
09:     const events = [];
10:
11:     for (let i = 0; i < n; i++) {
12:         const genres = [];
13:         const count = utils.random(1, 4);
14:         const time = new Date();
15:
16:         for (let j = 0; j < count; j++) {
17:             genres.push(faker.music.genre());
18:         }
19:
20:         time.setDate(time.getDate() + utils.random(1, 1000));
21:
22:         events.push({
23:             name: faker.music.songName(),
24:             artist: faker.name.fullName(),
25:             venue: faker.lorem.words(2),
26:             city: faker.address.cityName(),
27:             time: time,
28:             price: parseInt(utils.ethToGwei(utils.random(0.001,
    0.01))),
29:             imageUrl: faker.image.abstract(707, 976, true),
30:             description: faker.lorem.lines(5),
31:             genres: genres,
32:             cancelled: 0,
33:             deployed: 0
34:         });
35:     }
36:
37:     return events;
38: };
39:
40: const main = async () => {
41:     for (const event of generateEvents(SEED_COUNT)) {
42:         await db.setGenresForEvent(
43:             await db.createEvent(utils.omit(event, ["genres"])),
44:             event.genres
45:         );
46:     }
47:
48:     console.log(`Successfully seeded ${SEED_COUNT} events to the
    database!`);
49: };
50:
51: main()
52:     .then(process.exit)
53:     .catch(e => { throw e })
54: ;
```

## Backend Event Unseeding Script

File: backend/src/unseed.js

```
01: require('dotenv').config()
02:
03: const db = require("./helpers/db");
04:
05: const main = async () => {
06:     await db.reset();
07:     console.log(`Successfully unseeded database!`);
08: };
09:
10: main()
11:     .then(process.exit)
12:     .catch(e => { throw e })
13: ;
```

## Frontend AdminEventCard Component

File: frontend/src/components/AdminEventCard.tsx

```tsx
import { useState } from "react";
import { XLg } from "react-bootstrap-icons";
import { deleteEvent, Event } from "../helpers/api";
import routes from "../routes";
import AlertModal from "./AlertModal";
import ConfirmationModal from "./ConfirmationModal";
import EventCard from "./EventCard";

const AdminEventCard = (props: { event: Event, onCancel: () => any }) => {
    const [showConfirmationModal, setShowConfirmationModal] = useState(false);
    const [showErrorModal, setShowErrorModal] = useState(false);
    const [errorMessage, setErrorMessage] = useState("");

    const cancelEvent = async () => {
        try {
            await deleteEvent(props.event.id);
            props.onCancel();
        } catch (e: any) {
            setErrorMessage(e.response.data?.message || "");
            setShowErrorModal(true);
        }
    };

    return (
        <>
            <EventCard event={props.event} to={routes.admin.event(props.event.id)}>
                <div className="col-span-12 lg:col-span-4 xl:col-span-3 pt-5 lg:pt-0 flex">
                    <div className="lg:ml-auto">
                        <button
                            className="btn btn-base text-red-700 hover:text-red-900"
                            onClick={e => {
                                e.preventDefault();
                                setShowConfirmationModal(true);
                            }}
                        >
                            <XLg className="mr-1" />
                            Cancel
                        </button>
                    </div>
                </div>
            </EventCard>
            {
                showConfirmationModal &&
                <ConfirmationModal
                    title={"Cancel Event"}
                    message={`Are you sure you want to cancel ${props.event.name} by ${props.event.artist}? This cannot be undone!`}
                    close={() => setShowConfirmationModal(false)}
                    action={() => cancelEvent()}
                />
            }
            {
                showErrorModal &&
```

```
53:                 <AlertModal
54:                     close={() => setShowErrorModal(false)}
55:                     title="Error"
56:                     message={errorMessage || "An error occurred when
    trying to cancel the event."}
57:                     className="bg-red-50 text-red-500"
58:                 />
59:             }
60:         </>
61:     );
62: };
63:
64: export default AdminEventCard;
```

## Frontend AdminHeader Component

File: frontend/src/components/AdminHeader.tsx

```tsx
const AdminHeader = (props: { subtitle: string, children?: React.ReactNode, className?: string }) => {
    return (
        <div className={"text-gray-500 " + props.className}>
            <div>
                <h2 className="font-bold">Admin</h2>
                <div className="font-medium">{props.subtitle}</div>
            </div>
            {props.children}
        </div>
    );
};

export default AdminHeader;
```

## Frontend Alert Component

File: frontend/src/components/Alert.tsx

```
01: const Alert = (props: { title?: string, message: string, className?:
    string }) => {
02:     return (
03:         <div className={"flex p-4 mb-4 text-sm rounded-lg " +
    props.className}>
04:             <svg className="flex-shrink-0 inline w-5 h-5 mr-3"
    fill="currentColor" viewBox="0 0 20 20" xmlns="http://www.w3.org/
    2000/svg">
05:                 <path fillRule="evenodd" d="M18 10a8 8 0 11-16 0 8 8
    0 0116 0zm-7-4a1 1 0 11-2 0 1 1 0 012 0zM9 9a1 1 0 000 2v3a1 1 0
    001 1h1a1 1 0 100-2v-3a1 1 0 00-1-1H9z" clipRule="evenodd" />
06:             </svg>
07:             <div className="break-words w-[calc(100%-2rem)]">
08:                 <span className="font-medium mr-1">{props.title}</
    span>
09:                 {props.message}
10:             </div>
11:         </div>
12:     );
13: };
14:
15: export default Alert;
```

## Frontend AlertModal Component

File: frontend/src/components/AlertModal.tsx

```tsx
const AlertModal = (props: {
    close: () => any,
    title: string,
    message: string,
    className?: string
}) => {
    return (
        <div className="relative z-10" role="dialog">
            <div className="fixed inset-0 bg-gray-500 bg-opacity-75" />
            <div className="fixed inset-0 z-10 overflow-y-auto">
                <div className="flex min-h-full items-end justify-center p-4 text-center sm:items-center sm:p-0">
                    <div className="relative transform overflow-hidden rounded-lg bg-white text-left shadow-xl transition-all sm:my-8 sm:w-full sm:max-w-lg">
                        <div className="bg-white px-4 pt-5 pb-4 sm:p-6 sm:pb-4">
                            <div className="sm:flex sm:items-start">
                                <div className={"mx-auto flex h-12 w-12 flex-shrink-0 items-center justify-center rounded-full sm:mx-0 sm:h-10 sm:w-10 " + (props.className || "bg-indigo-100")}>
                                    <svg className={"h-6 w-6 " + (props.className || "text-indigo-500")} fill="none" viewBox="0 0 24 24" strokeWidth="1.5" stroke="currentColor" aria-hidden="true">
                                        <path strokeLinecap="round" strokeLinejoin="round" d="M12 9v3.75m-9.303 3.376c-.866 1.5.217 3.374 1.948 3.374h14.71c1.73 0 2.813-1.874 1.948-3.374L13.949 3.378c-.866-1.5-3.032-1.5-3.898 0L2.697 16.126zM12 15.75h.007v.008H12v-.008z" />
                                    </svg>
                                </div>
                                <div className="mt-3 text-center sm:mt-0 sm:ml-4 sm:text-left w-full">
                                    <h3 className="text-lg font-medium leading-6 text-gray-900" id="modal-title">
                                        {props.title}
                                    </h3>
                                    <div className="mt-2">
                                        <p className="text-sm text-gray-500 break-words sm:w-[calc(100%-3.5rem)]">
                                            {props.message}
                                        </p>
                                    </div>
                                </div>
                            </div>
                        </div>
                        <div className="px-4 py-5 sm:flex sm:flex-row-reverse sm:px-6">
                            <button
                                type="button"
                                className="mt-3 inline-flex w-full justify-center rounded-md border border-gray-300 bg-white px-4 py-2 text-base font-medium text-gray-700 shadow-sm hover:bg-gray-50 focus:outline-none sm:mt-0 sm:ml-3 sm:w-auto sm:text-sm disabled:opacity-25"
                                onClick={() => props.close()}
                            >
                                Okay
```

```
39:                                 </button>
40:                             </div>
41:                         </div>
42:                     </div>
43:                 </div>
44:             </div>
45:         );
46: };
47:
48: export default AlertModal;
```

## Frontend BackCaret Component

File: frontend/src/components/BackCaret.tsx

```tsx
import { CaretLeftFill } from "react-bootstrap-icons";
import { Link } from "react-router-dom";

const BackCaret = (props: { to: string }) => {
    return (
        <Link to={props.to} className="flex items-center text-indigo-500 mb-5">
            <CaretLeftFill className="mr-1.5" />
            Back
        </Link>
    );
};

export default BackCaret;
```

## Frontend CheckboxGroup Component

File: frontend/src/components/CheckboxGroup.tsx

```tsx
const CheckboxGroup = (props: {
    items: CheckboxItem[],
    dispatch: React.Dispatch<React.SetStateAction<CheckboxItem[]>>,
    onUpdate: (items: CheckboxItem[]) => any
}) => {
    const toggle = (name: string) => {
        props.dispatch(items => {
            const updated = items.map(item => {
                return {
                    ...item,
                    checked: (item.name === name) ? !item.checked : item.checked
                };
            });

            props.onUpdate(updated);

            return updated;
        });
    };

    return (
        <>
            {
                props.items.map((item, i) => {
                    return (
                        <div key={i} className="flex items-center">
                            <input
                                type="checkbox"
                                className="rounded text-indigo-500 mr-1.5 appearance-none focus:ring-0 focus:ring-offset-0"
                                id={item.name}
                                value={item.name}
                                onChange={(e) => toggle(item.name)}
                                checked={item.checked}
                            />
                            <label htmlFor={item.name}>
                                {item.name}
                            </label>
                        </div>
                    );
                })
            }
        </>
    );
};

export interface CheckboxItem {
    name: string,
    checked: boolean
};

export default CheckboxGroup;
```

## Frontend ConfirmationModal Component

File: frontend/src/components/ConfirmationModal.tsx

```tsx
import { useState } from "react";
import Spinner from "./Spinner";

const ConfirmationModal = (props: {
    close: () => any,
    action: () => any,
    title: string,
    message: string,
    children?: React.ReactNode
}) => {
    const [disabled, setDisabled] = useState(false);

    const action = async () => {
        setDisabled(true);
        await props.action();
        props.close();
    };

    return (
        <div className="relative z-10" role="dialog">
            <div className="fixed inset-0 bg-gray-500 bg-opacity-75" />
            <div className="fixed inset-0 z-10 overflow-y-auto">
                <div className="flex min-h-full items-end justify-center p-4 text-center sm:items-center sm:p-0">
                    <div className="relative transform overflow-hidden rounded-lg bg-white text-left shadow-xl transition-all sm:my-8 sm:w-full sm:max-w-lg">
                        <div className="bg-white px-4 pt-5 pb-4 sm:p-6 sm:pb-4">
                            <div className="sm:flex sm:items-start">
                                <div className="mx-auto flex h-12 w-12 flex-shrink-0 items-center justify-center rounded-full bg-red-100 sm:mx-0 sm:h-10 sm:w-10">
                                    <svg className="h-6 w-6 text-red-600" fill="none" viewBox="0 0 24 24" strokeWidth="1.5" stroke="currentColor" aria-hidden="true">
                                        <path strokeLinecap="round" strokeLinejoin="round" d="M12 9v3.75m-9.303 3.376c-.866 1.5.217 3.374 1.948 3.374h14.71c1.73 0 2.813-1.874 1.948-3.374L13.949 3.378c-.866-1.5-3.032-1.5-3.898 0L2.697 16.126zM12 15.75h.007v.008H12v-.008z" />
                                    </svg>
                                </div>
                                <div className="mt-3 text-center sm:mt-0 sm:ml-4 sm:text-left w-full">
                                    <h3 className="text-lg font-medium leading-6 text-gray-900" id="modal-title">
                                        {props.title}
                                    </h3>
                                    <div className="mt-2 break-words sm:w-[calc(100%-3.5rem)]">
                                        <p className="text-sm text-gray-500">
                                            {props.message}
                                        </p>
                                        {props.children}
                                    </div>
                                </div>
```

```
43:                             </div>
44:                         </div>
45:                         <div className="px-4 py-5 sm:flex sm:flex-
    row-reverse sm:px-6">
46:                             <button
47:                                 type="button"
48:                                 disabled={disabled}
49:                                 className="inline-flex w-full
    justify-center rounded-md border border-transparent bg-red-600 px-4
    py-2 text-base font-medium text-white shadow-sm hover:bg-red-700
    focus:outline-none sm:ml-3 sm:w-auto sm:text-sm
    disabled:opacity-25"
50:                                 onClick={() => action()}
51:                             >
52:                                 {disabled && <Spinner />}
53:                                 Continue
54:                             </button>
55:                             <button
56:                                 type="button"
57:                                 disabled={disabled}
58:                                 className="mt-3 inline-flex w-full
    justify-center rounded-md border border-gray-300 bg-white px-4 py-2
    text-base font-medium text-gray-700 shadow-sm hover:bg-gray-50
    focus:outline-none sm:mt-0 sm:ml-3 sm:w-auto sm:text-sm
    disabled:opacity-25"
59:                                 onClick={() => props.close()}
60:                             >
61:                                 Cancel
62:                             </button>
63:                         </div>
64:                     </div>
65:                 </div>
66:             </div>
67:         </div>
68:     );
69: };
70:
71: export default ConfirmationModal;
```

## Frontend ConnectWallet Component

File: frontend/src/components/ConnectWallet.tsx

```tsx
import { useState } from 'react';
import { useLocalStorage } from 'usehooks-ts';
import Web3 from 'web3';
import { isCorrectNetwork, switchNetwork } from '../helpers/contract';
import { useAddress } from '../middleware/Wallet';

const ConnectWallet = (props: { className?: string, onLocked?: () => any }) => {
    const [loggedOut, setLoggedOut] = useLocalStorage("loggedOut", false);
    const [address, setAddress] = useAddress();
    const [hovering, setHovering] = useState(false);

    const connectWallet = async (e: React.MouseEvent<HTMLElement>) => {
        if (!Web3.givenProvider) {
            return;
        }

        e.preventDefault();

        // Logout
        if (!loggedOut) {
            setAddress("");
            setLoggedOut(true);
            return;
        }

        const web3 = new Web3(Web3.givenProvider);

        try {
            const chainId = await web3.eth.getChainId();

            if (!isCorrectNetwork(chainId)) {
                await switchNetwork();
            }

            const accounts = await web3.eth.requestAccounts();
            setAddress(accounts[0]);
            setLoggedOut(false);
        } catch (e: any) {
            // Wallet is locked
            if (e.code && e.code === -32002 && props.onLocked) {
                props.onLocked();
            }
        }
    };

    return (
        <a
            href="https://metamask.io/"
            target="_blank"
            rel="noreferrer"
            className={`btn btn-basic ${address ? "w-36 hover:text-red-700 hover:outline-red-700" : "w-44"} ${props.className} `}
            onClick={(e) => connectWallet(e)}
            onMouseEnter={() => setHovering(true)}
            onMouseLeave={() => setHovering(false)}
```

```
55:         >
56:             <img src={process.env.PUBLIC_URL + "/static/
    metamask.svg"} className="mr-2" alt="MetaMask icon" height={20}
    width={20}/>
57:             {address ? (hovering ? "Disconnect" : "Connected") :
    "Connect Wallet"}
58:         </a>
59:     );
60: };
61:
62: export default ConnectWallet;
```

## Frontend EventCard Component

File: frontend/src/components/EventCard.tsx

```tsx
import { X } from "react-bootstrap-icons";
import { Link } from "react-router-dom";
import { Event } from "../helpers/api";
import { gweiToEth, prettyDate } from "../helpers/utils";
import routes from "../routes";

const EventCard = (props: {
    event: Event,
    className?: string,
    to?: string,
    quantity?: number,
    children?: React.ReactNode,
    used?: number
}) => {
    return (
        <Link to={props.to || routes.event(props.event.id)} 
    className={"card flex w-full py-5 px-5 " + props.className}>
            <div className="my-auto grid grid-cols-12 w-full">
                <div className="col-span-12 lg:col-span-8 xl:col-
    span-9 flex items-center">
                    <img className="rounded shadow-lg hidden 
    sm:block thumbnail mr-5" src={props.event.imageUrl} 
    alt={props.event.artist} />
                    <div className="text-left w-full flex">
                        <div className="my-auto w-full">
                            <div className="italic mb-2">
                                <h2 className="font-bold uppercase 
    mr-2">
                                    {props.event.artist}
                                </h2>
                                <div className="text-xl">
                                    {props.event.name}
                                </div>
                            </div>
                            <div className="text-xl font-bold text-
    indigo-500 flex flex-wrap items-center mb-0.5">
                                <div className="flex items-center 
    mr-2">
                                    {
                                        props.quantity &&
                                        <>
                                            {props.quantity} <X />
                                        </>
                                    }
                                    {gweiToEth(props.event.price)} 
    ETH
                                </div>
                                {
                                    !!props.used && props.used > 0 
    &&
                                    <div className="text-indigo-500 
    font-medium text-xs uppercase">
                                        [{props.used} Used]
                                    </div>
                                }
                            </div>
                            <div className="text-md block 
    capitalize">
```

```
48:                                     {props.event.venue} ¬∑
    {props.event.city} ¬∑ {prettyDate(props.event.time)}
49:                                 </div>
50:                             </div>
51:                         </div>
52:                     </div>
53:                     {props.children}
54:                 </div>
55:             </Link>
56:         );
57: };
58:
59: export default EventCard;
```

## Frontend GenrePill Component

File: frontend/src/components/GenrePill.tsx

```
01: import { Link } from "react-router-dom";
02: import routes from "../routes";
03:
04: const GenrePill = (props: { name: string, className?: string }) => {
05:     return (
06:         <Link to={`${routes.events()}?genres=$
     {encodeURIComponent(props.name)}`} className={props.className}>
07:             <div className="rounded-full bg-indigo-500 uppercase
     text-white px-5 py-1 h-8 whitespace-nowrap">
08:                 {props.name}
09:             </div>
10:         </Link>
11:     );
12: };
13:
14: export default GenrePill;
```

## Frontend LoadingCard Component

File: frontend/src/components/LoadingCard.tsx

```
01: const LoadingCard = (props: { className?: string }) => {
02:     return (
03:         <div className={"card shadow rounded-md p-5 w-full mx-auto "
    + props.className}>
04:             <div className="animate-pulse flex space-x-4">
05:                 <div className="rounded bg-slate-200 h-32 w-24" />
06:                 <div className="flex-1 space-y-6 py-1">
07:                     <div className="h-2 bg-slate-200 rounded" />
08:                     <div className="space-y-3">
09:                         <div className="grid grid-cols-3 gap-4">
10:                             <div className="h-2 bg-slate-200 rounded
    col-span-2" />
11:                             <div className="h-2 bg-slate-200 rounded
    col-span-1" />
12:                         </div>
13:                         <div className="grid grid-cols-3 gap-4">
14:                             <div className="h-2 bg-slate-200 rounded
    col-span-2" />
15:                             <div className="h-2 bg-slate-200 rounded
    col-span-1" />
16:                         </div>
17:                         <div className="h-2 bg-slate-200 rounded" />
18:                         <div className="h-2 bg-slate-200 rounded" />
19:                         <div className="h-2 bg-slate-200 rounded" />
20:                     </div>
21:                 </div>
22:             </div>
23:         </div>
24:     );
25: };
26:
27: export default LoadingCard;
```

## Frontend NavBar Component

File: frontend/src/components/NavBar.tsx

```
001: import { useEffect, useState } from "react";
002: import { Link, useNavigate, useSearchParams } from "react-router-
     dom";
003: import { useAddress } from "../middleware/Wallet";
004: import routes from "../routes";
005: import AlertModal from "./AlertModal";
006: import ConnectWallet from "./ConnectWallet";
007: import SearchBar from "./SearchBar";
008:
009: const NavBar = () => {
010:     const navigate = useNavigate();
011:     const [address] = useAddress();
012:     const [mobileMenuHidden, setMobileMenuHidden] = useState(true);
013:     const [navEntries, setNavEntries] =
     useState<NavEntry[]>(allNavEntries);
014:     const [showAlert, setShowAlert] = useState(false);
015:     const [searchParams, setSearchParams] = useSearchParams();
016:
017:     const search = (search: string) => {
018:         if (window.location.pathname !== routes.events()) {
019:             navigate(
020:                 `${routes.events()}?search=$
     {encodeURIComponent(search)}`
021:             );
022:         } else {
023:             searchParams.set("search", search);
024:             searchParams.delete("offset");
025:             setSearchParams(searchParams);
026:         }
027:     };
028:
029:     useEffect(() => {
030:         const isActive = (navEntry: NavEntry) => {
031:             if (navEntry.location === "/") {
032:                 return window.location.pathname === "/";
033:             } else {
034:                 return
     window.location.pathname.startsWith(navEntry.location);
035:             }
036:         };
037:
038:         const navEntries = allNavEntries
039:             .filter(navEntry => address || !navEntry.requiresWallet)
040:             .map(navEntry => {
041:                 return { ...navEntry, active: isActive(navEntry)};
042:             })
043:         ;
044:
045:         setNavEntries(navEntries);
046:     }, [navigate, address]);
047:
048:     return (
049:         <>
050:             <nav className="bg-white drop-shadow">
051:                 <div className="mx-auto px-2 sm:px-6 lg:px-8">
052:                     <div className="relative flex h-16 items-center
     justify-between">
053:                         <div className="absolute inset-y-0 left-0
     flex items-center sm:hidden">
```

```
054:                             <button
055:                                 type="button"
056:                                 onClick={() => setMobileMenuHidden(!
    mobileMenuHidden)}
057:                                 className="inline-flex items-center
    justify-center rounded-md p-2 text-gray-400 focus:outline-none
    focus:ring-2 focus:ring-inset focus:ring-white"
058:                             >
059:                                 <svg className="block h-6 w-6"
    xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
    strokeWidth="1.5" stroke="currentColor">
060:                                     <path strokeLinecap="round"
    strokeLinejoin="round" d="M3.75 6.75h16.5M3.75 12h16.5m-16.5
    5.25h16.5" />
061:                                 </svg>
062:                             </button>
063:                         </div>
064:                         <div className="flex flex-1 justify-center
    sm:justify-start h-full">
065:                             <Link to={routes.home()} className="flex
    flex-shrink-0 items-center">
066:                                 <img className="block h-8 w-auto"
    src="https://ethereum.org/static/a183661dd70e0e5c70689a0ec95ef0ba/
    13c43/eth-diamond-purple.png" alt="Ethereum Logo" />
067:                             </Link>
068:                             <div className="hidden sm:ml-6 sm:block
    justify-center">
069:                                 <div className="flex space-x-4 h-
    full">
070:                                     {
071:                                         navEntries.map((navEntry, i)
    => {
072:                                             if (navEntry.location
    === routes.home()) {
073:                                                 return <Tab key={i}
    navEntry={navEntry} className="hidden md:flex" />
074:                                             } else {
075:                                                 return <Tab key={i}
    navEntry={navEntry} />;
076:                                             }
077:                                         })
078:                                     }
079:                                 </div>
080:                             </div>
081:                         </div>
082:                         <div className="pr-2 sm:pr-0 hidden sm:flex
    items-center">
083:                             <SearchBar className="w-40 md:w-56"
    onSubmit={search} />
084:                             <ConnectWallet className="ml-3"
    onLocked={() => setShowAlert(true)} />
085:                         </div>
086:                     </div>
087:                 </div>
088:                 <div className={"sm:hidden " + (mobileMenuHidden ?
    "hidden" : "")}>
089:                     <div className="pt-2 pb-3 w-full">
090:                         <SearchBar
091:                             className="mx-2 mb-2"
092:                             onSubmit={s => {
093:                                 setMobileMenuHidden(true);
094:                                 search(s);
095:                             }}
```

```
096:                         />
097:                         <div className="flex mx-2 mb-3">
098:                             <ConnectWallet className="w-full"
    onLocked={() => setShowAlert(true)} />
099:                         </div>
100:                         {
101:                             navEntries.map((navEntry, i) => {
102:                                 return (
103:                                     <MobileTab
104:                                         key={i}
105:                                         navEntry={navEntry}
106:                                         onClick={() =>
    setMobileMenuHidden(true)}
107:                                     />
108:                                 );
109:                             })
110:                         }
111:                     </div>
112:                 </div>
113:             </nav>
114:             {
115:                 showAlert &&
116:                 <AlertModal
117:                     close={() => setShowAlert(false)}
118:                     title="Unlock Wallet"
119:                     message="Please manually unlock your MetaMask
    wallet to allow us to connect to it."
120:                 />
121:             }
122:         </>
123:     );
124: };
125:
126: interface NavEntry {
127:     location: string,
128:     text: string,
129:     active: boolean,
130:     requiresWallet: boolean
131: };
132:
133: const allNavEntries = [
134:     { location: routes.home(), text: "Home", active: false,
    requiresWallet: false },
135:     { location: routes.events(), text: "Events", active: false,
    requiresWallet: false },
136:     { location: routes.purchases(), text: "Purchases", active:
    false, requiresWallet: true }
137: ];
138:
139: const Tab = (props: { navEntry: NavEntry, className?: string }) =>
    {
140:     const borderColour = (props.navEntry.active) ? "border-
    indigo-500" : "border-transparent";
141:
142:     return (
143:         <Link to={props.navEntry.location} className={`border-b-2 h-
    full text-gray-800 px-3 py-2 text-sm font-medium h-full items-
    center flex ${borderColour} ${props.className}`}>
144:             {props.navEntry.text}
145:         </Link>
146:     );
147: };
148:
```

```
149: const MobileTab = (props: { navEntry: NavEntry, onClick: () =>
     any }) => {
150:     const className = (props.navEntry.active) ? "border-l-4 border-
     indigo-500 bg-indigo-100 text-indigo-800" : "text-gray-500";
151:
152:     return (
153:         <Link
154:             to={props.navEntry.location}
155:             className={"py-2 w-full text-left flex " + className}
156:             onClick={() => props.onClick()}
157:         >
158:             <div className={(props.navEntry.active) ? "ml-3 text-
     indigo-800" : "ml-4 text-gray-600"}>
159:                 {props.navEntry.text}
160:             </div>
161:         </Link>
162:     );
163: };
164:
165: export default NavBar;
```

## Frontend PageError Component

File: frontend/src/components/PageError.tsx

```
01: import { useNavigate } from "react-router-dom";
02: import { ETH_NETWORK } from "../helpers/contract";
03:
04: const PageError = () => {
05:     const navigate = useNavigate();
06:
07:     return (
08:         <div className="text-center my-24 mx-5">
09:             <h1 className="font-bold text-red-600 mb-3">Oops!</h1>
10:             <div>
11:                 Something went wrong whilst trying to process your
    request.
12:             </div>
13:             <div className="mb-3">
14:                 Please ensure MetaMask is connected to the
    {ETH_NETWORK} network.
15:             </div>
16:             <div className="flex justify-center">
17:                 <button type="button" className="btn w-fit text-
    red-600 hover:text-red-800" onClick={() => navigate(0)}>
18:                     Try Again
19:                 </button>
20:             </div>
21:         </div>
22:     );
23: };
24:
25: export default PageError;
```

## Frontend PaginationButtons Component

File: frontend/src/components/PaginationButtons.tsx

```tsx
import { ArrowLeft, ArrowRight } from "react-bootstrap-icons";

const PaginationButtons = (props: {
    prev: () => any,
    next: () => any,
    prevDisabled: boolean,
    nextDisabled: boolean,
    hideWhenDisabled: boolean,
    className?: string,
    onChange?: () => any
}) => {
    return (
        <>
            {
                (!props.hideWhenDisabled || !props.prevDisabled) &&
                <button
                    type="button"
                    className={props.className}
                    disabled={props.prevDisabled}
                    onClick={() => {
                        props.prev();

                        if (props.onChange) {
                            props.onChange();
                        }
                    }}
                >
                    <ArrowLeft className="mr-2" />
                    Prev
                </button>
            }
            {
                (!props.hideWhenDisabled || !props.nextDisabled) &&
                <button
                    type="button"
                    className={props.className}
                    disabled={props.nextDisabled}
                    onClick={() => {
                        props.next();

                        if (props.onChange) {
                            props.onChange();
                        }
                    }}
                >
                    Next
                    <ArrowRight className="ml-2" />
                </button>
            }
        </>
    );
};

export default PaginationButtons;
```

## Frontend PurchaseCard Component

File: frontend/src/components/PurchaseCard.tsx

```
001: import { useEffect, useState } from "react";
002: import { CurrencyDollar, QrCode, XLg } from "react-bootstrap-
     icons";
003: import QRCode from "react-qr-code";
004: import { getSignature, Purchase } from "../helpers/api";
005: import { getInstance } from "../helpers/contract";
006: import { gweiToEth } from "../helpers/utils";
007: import { useAddress } from "../middleware/Wallet";
008: import ConfirmationModal from "./ConfirmationModal";
009: import EventCard from "./EventCard";
010: import QRModal from "./QRModal";
011: import QuantityButton from "./QuantityButton";
012:
013: export type QrData = {
014:    signature: string,
015:    eventId: number,
016:    quantity: number
017: };
018:
019: const PurchaseCard = (props: { purchase: Purchase, className?:
     string, onChange: () => any }) => {
020:    const [selectedQuantity, setSelectedQuantity] = useState(1);
021:    const [showConfirmationModal, setShowConfirmationModal] =
     useState(false);
022:    const [remainingQuantity, setRemainingQuantity] = useState(0);
023:    const [qrData, setQrData] = useState<QrData>();
024:    const [address] = useAddress();
025:    const price = gweiToEth(props.purchase.event.price);
026:
027:    const quantityButtonClick = (e:
     React.MouseEvent<HTMLButtonElement, MouseEvent>) => {
028:        e.preventDefault();
029:        e.stopPropagation();
030:        setShowConfirmationModal(true);
031:    };
032:
033:    useEffect(() => {
034:        if (props.purchase.used) {
035:            setRemainingQuantity(props.purchase.quantity -
     props.purchase.used);
036:            setSelectedQuantity(props.purchase.quantity -
     props.purchase.used);
037:        } else {
038:            setRemainingQuantity(props.purchase.quantity);
039:            setSelectedQuantity(props.purchase.quantity);
040:        }
041:    }, [props.purchase, address]);
042:
043:    const toggleTokenListing = async () => {
044:        try {
045:            const contract = await getInstance();
046:
047:            if (props.purchase.forSale) {
048:                await contract
049:                    .methods
050:                    .unlistTokenForResale(props.purchase.event.id,
     selectedQuantity)
051:                    .send({ from: address })
052:                ;
```

```
053:             } else {
054:                 await contract
055:                     .methods
056:                     .listTokenForResale(props.purchase.event.id,
    selectedQuantity)
057:                     .send({ from: address })
058:                 ;
059:             }
060:         } catch (e: any) {
061:             console.error(e);
062:         }
063:
064:         props.onChange();
065:     };
066:
067:     const generateQr = async (e: React.MouseEvent<HTMLButtonElement,
    MouseEvent>) => {
068:         e.preventDefault();
069:
070:         try {
071:             const w = (window as any);
072:
073:             if (!w.ethereum) {
074:                 return;
075:             }
076:
077:             const message = (await getSignature()).message;
078:             const signature = await w.ethereum.request({
079:                 method: "personal_sign",
080:                 params: [message, address]
081:             });
082:
083:             setQrData(
084:                 {
085:                     signature: signature,
086:                     eventId: props.purchase.event.id,
087:                     quantity: props.purchase.quantity
088:                 }
089:             );
090:         } catch (e) {
091:             console.error(e);
092:         }
093:     };
094:
095:     return (
096:         <>
097:             <EventCard
098:                 event={props.purchase.event}
099:                 quantity={props.purchase.quantity}
100:                 to={(props.purchase.event.cancelled) ? "#" :
    undefined}
101:                 used={props.purchase.used}
102:             >
103:                 {
104:                     !props.purchase.expired &&
105:                     !props.purchase.event.cancelled &&
106:                     remainingQuantity > 0 &&
107:                     <div className="col-span-12 lg:col-span-4
    xl:col-span-3 lg:pt-0 pt-5 flex">
108:                         <div className="lg:ml-auto flex h-fit space-
    x-2">
109:                             {
110:                                 !props.purchase.forSale &&
```

```
111:                                 <button className="btn btn-basic"
    onClick={e => generateQr(e)}>
112:                                     <QrCode className="mr-1" />
113:                                     QR
114:                                 </button>
115:                             }
116:                             {
117:                                 <QuantityButton
118:                                     quantity={remainingQuantity}
119:
    className={props.purchase.forSale ? "outline-red-700 text-red-700
    hover:outline-red-900 hover:text-red-900" : "outline-green-700
    text-green-700 hover:outline-green-900 hover:text-green-900"}
120:                                     value={selectedQuantity}
121:                                     onClick={e =>
    quantityButtonClick(e)}
122:                                     onChange={e =>
    setSelectedQuantity(parseInt(e.target.value))}
123:                                 >
124:                                     {
125:                                         !props.purchase.forSale &&
126:                                         <>
127:                                             <CurrencyDollar
    size={16} className="mr-1" />
128:                                             Sell
129:                                         </>
130:                                     }
131:                                     {
132:                                         props.purchase.forSale &&
133:                                         <>
134:                                             <XLg size={16}
    className="mr-1" />
135:                                             Unlist
136:                                         </>
137:                                     }
138:                                 </QuantityButton>
139:                             }
140:                         </div>
141:                     </div>
142:                 }
143:             </EventCard>
144:             {
145:                 showConfirmationModal &&
146:                 <ConfirmationModal
147:                     title={props.purchase.forSale ? "Unlist
    Ticket" : "Sell Ticket"}
148:                     message={
149:                         props.purchase.forSale
150:                             ? `Are you sure you want to unlist $
    {selectedQuantity} tickets?`
151:                             : `Are you sure you want to list $
    {selectedQuantity} tickets for sale at ${price} ETH each?`
152:                     }
153:                     close={() => setShowConfirmationModal(false)}
154:                     action={() => toggleTokenListing()}
155:                 />
156:             }
157:             {
158:                 qrData &&
159:                 <QRModal
160:                     close={() => setQrData(undefined)}
161:                     title="Ticket QR"
162:                 >
```

```
163:                     <div className="mt-5 mb-1 space-y-3">
164:                         <QRCode value={JSON.stringify(qrData)}
    size={200} />
165:                         <div className="font-medium text-gray-800">
166:                             Remaining Tickets: {remainingQuantity}
167:                         </div>
168:                     </div>
169:                 </QRModal>
170:             }
171:         </>
172:     );
173: };
174:
175: export default PurchaseCard;
```

## Frontend QRModal Component

File: frontend/src/components/QRModal.tsx

```tsx
const QRModal = (props: {
    close: () => any,
    title: string,
    children: React.ReactNode
}) => {
    return (
        <div className="relative z-10" role="dialog">
            <div className="fixed inset-0 bg-gray-500 bg-opacity-75" />
            <div className="fixed inset-0 z-10 overflow-y-auto">
                <div className="flex min-h-full items-end justify-center p-4 text-center sm:items-center sm:p-0">
                    <div className="relative transform overflow-hidden rounded-lg bg-white text-left shadow-xl transition-all sm:my-8 sm:w-full sm:max-w-sm">
                        <div className="bg-white px-4 pt-5 pb-4 sm:p-6 sm:pb-4 justify-center flex">
                            <div className="text-center">
                                <h3 className="text-lg font-medium leading-6 text-gray-900" id="modal-title">
                                    {props.title}
                                </h3>
                                <div className="mt-2">
                                    {props.children}
                                </div>
                            </div>
                        </div>
                        <div className="px-4 pb-5 w-full">
                            <button
                                type="button"
                                className="rounded-md border border-gray-300 bg-white px-4 py-2 text-base font-medium text-gray-700 shadow-sm hover:bg-gray-50 focus:outline-none text-sm disabled:opacity-25 w-full"
                                onClick={() => props.close()}
                            >
                                Close
                            </button>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    );
};

export default QRModal;
```

## Frontend QuantityButton Component

File: frontend/src/components/QuantityButton.tsx

```tsx
import { range } from "../helpers/utils";

const QuantityButton = (props: {
    children?: React.ReactNode,
    quantity: number,
    value: number,
    className?: string,
    onClick?: React.MouseEventHandler<HTMLButtonElement>,
    onChange?: React.ChangeEventHandler<HTMLSelectElement>,
    disabled?: boolean
}) => {
    const quantityOnClick = (e: React.MouseEvent<HTMLSelectElement, MouseEvent>) => {
        e.preventDefault();
        e.stopPropagation();
    };

    return (
        <button
            type="button"
            className={"btn-base py-2 pl-4 pr-0 outline outline-1 " + props.className}
            onClick={e => props.onClick && props.onClick(e)}
            disabled={props.disabled}
        >
            {
                props.children &&
                <div className="flex pr-3 items-center border-r border-current h-full">
                    {props.children}
                </div>
            }
            <select
                className="border-transparent h-full bg-transparent py-0 pl-3 pr-8 text-sm focus:ring-0 focus:border-transparent"
                onClick={e => quantityOnClick(e)}
                value={props.value}
                onChange={e => props.onChange && props.onChange(e)}
                disabled={props.disabled}
            >
                {range(1, props.quantity).map(i => <option key={i} >{i}</option>)}
            </select>
        </button>
    );
};

export default QuantityButton;
```

## Frontend SearchBar Component

File: frontend/src/components/SearchBar.tsx

```tsx
import { useState } from "react";

const SearchBar = (props: {
    className?: string,
    onSubmit?: (search: string) => any,
    onChange?: React.ChangeEventHandler<HTMLInputElement>
}) => {
    const [search, setSearch] = useState("");

    const submit = (e: React.FormEvent<HTMLFormElement>) => {
        e.preventDefault();

        if (!props.onSubmit) {
            return;
        }

        props.onSubmit(search);
    };

    const change = (e: React.ChangeEvent<HTMLInputElement>) => {
        setSearch(e.target.value);

        if (props.onChange) {
            props.onChange(e);
        }
    };

    return (
        <form className={"relative shadow-sm " + props.className} onSubmit={e => submit(e)} >
            <div className="pointer-events-none absolute inset-y-0 left-0 flex items-center pl-3">
                <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1} stroke="currentColor" className="w-4 h-4">
                    <path strokeLinecap="round" strokeLinejoin="round" d="M21 21l-5.197-5.197m0 0A7.5 7.5 0 105.196 5.196a7.5 7.5 0 0010.607 10.607z" />
                </svg>
            </div>
            <input
                type="text"
                className="input pl-9 pr-5"
                placeholder="Search"
                value={search} onChange={e => change(e)}
            />
        </form>
    );
};

export default SearchBar;
```

## Frontend Spinner Component

File: frontend/src/components/Spinner.tsx

```
01: const Spinner = () => {
02:     return (
03:         <svg className="animate-spin mr-2 h-5 w-5" xmlns="http://
    www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">
04:             <circle className="opacity-25" cx="12" cy="12" r="10"
    stroke="currentColor" strokeWidth="4"></circle>
05:             <path className="opacity-75" fill="currentColor" d="M4
    12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014
    12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"></path>
06:         </svg>
07:     );
08: };
09:
10: export default Spinner;
```

## Frontend API Helper

File: frontend/src/helpers/api.ts

```ts
import axios from 'axios';
import { AbiItem } from 'web3-utils';

axios.defaults.withCredentials = true;

enum HttpMethod {
    GET = "get",
    POST = "post",
    PUT = "put",
    DELETE = "delete"
};

export type GetEventsResponse = {
    events: Event[],
    prevOffset: number,
    nextOffset: number | boolean,
    limit: number
};

export type Event = {
    id: number
    name: string
    artist: string
    venue: string
    city: string
    time: string | number
    price: number
    imageUrl: string
    description: string
    genres: string[]
    cancelled: number
    quantity?: number
    supplied?: number
    remaining?: number
};

export type Purchase = {
    event: Event,
    quantity: number,
    forSale: boolean,
    expired: boolean,
    used?: number
};

export type Contract = {
    ABI: AbiItem,
    address: string
};

export type DeleteEventResponse = {
    code: string
    message: string
};

export type GetSignatureResponse = {
    message: string
};

```

```
059: const request = async <T>(method: HttpMethod, endpoint: string,
     options?: { data?: object, params?: object }) => {
060:     const API_BASE = (process.env.NODE_ENV === 'production')
061:         ? "https://muddy-sunset-2817.fly.dev/api" // Production
062:         : "http://localhost:3001/api" // Development
063:     ;
064:
065:     const res = await axios({
066:         method: method,
067:         url: `${API_BASE}/${endpoint}`,
068:         data: options?.data,
069:         params: options?.params
070:     });
071:
072:     return res.data as T;
073: };
074:
075: export const getEvents = (params?: {
076:     offset?: number,
077:     genres?: string[],
078:     locations?: string[],
079:     maxPrice?: number,
080:     search?: string
081: }) => {
082:     return request<GetEventsResponse>(
083:         HttpMethod.GET,
084:         "events",
085:         {
086:             params: {
087:                 offset: params?.offset,
088:                 genres: params?.genres?.join(","),
089:                 locations: params?.locations?.join(","),
090:                 maxPrice: params?.maxPrice,
091:                 search: params?.search
092:             }
093:         }
094:     );
095: };
096:
097: export const getEvent = (id: number) => {
098:     return request<Event>(
099:         HttpMethod.GET,
100:         "events/" + id,
101:     );
102: };
103:
104: export const getGenres = () => {
105:     return request<string[]>(
106:         HttpMethod.GET,
107:         "genres"
108:     );
109: };
110:
111: export const getLocations = () => {
112:     return request<string[]>(
113:         HttpMethod.GET,
114:         "locations"
115:     );
116: };
117:
118: export const getPurchases = (address: string) => {
119:     return request<Purchase[]>(
120:         HttpMethod.GET,
```

```
121:         "purchases/" + address
122:     );
123: };
124:
125: export const getContract = () => {
126:     return request<Contract>(
127:         HttpMethod.GET,
128:         "contract"
129:     );
130: };
131:
132: export const login = (password?: string) => {
133:     return request<undefined>(
134:         HttpMethod.POST,
135:         "login",
136:         { data: { password: password } }
137:     );
138: };
139:
140: export const createEvent = (event: FormData) => {
141:     return request<undefined>(
142:         HttpMethod.POST,
143:         "events",
144:         { data: event }
145:     );
146: };
147:
148: export const updateEvent = (event: FormData) => {
149:     return request<undefined>(
150:         HttpMethod.PUT,
151:         "events",
152:         { data: event }
153:     );
154: };
155:
156: export const deleteEvent = (id: number) => {
157:     return request<undefined>(
158:         HttpMethod.DELETE,
159:         "events/" + id
160:     );
161: };
162:
163: export const eventToken = (signature: string, eventId: number,
     quantity: number) => {
164:     return request<undefined>(
165:         HttpMethod.POST,
166:         `events/${eventId}/token`,
167:         { data: { signature: signature, quantity: quantity } }
168:     );
169: };
170:
171: export const getSignature = () => {
172:     return request<GetSignatureResponse>(
173:         HttpMethod.GET,
174:         "signature"
175:     );
176: };
```

## Frontend Contract Helper

File: frontend/src/helpers/contract.ts

```
01: import Web3 from "web3";
02: import { getContract } from './api';
03:
04: const web3 = new Web3(Web3.givenProvider);
05: const w = (window as any);
06:
07: enum EthNetwork {
08:     GOERLI = "Goerli",
09:     GANACHE = "Ganache"
10: };
11:
12: export const ETH_NETWORK = (process.env.NODE_ENV === 'production') ?
     EthNetwork.GOERLI : EthNetwork.GANACHE;
13:
14: export const CHAIN_ID = (ETH_NETWORK === EthNetwork.GANACHE) ?
     1337 : 5;
15:
16: export type ResaleToken = {
17:     owner: string,
18:     sold: boolean
19: };
20:
21: export const isCorrectNetwork = (chainId: string | number) =>
     (Number(chainId) === CHAIN_ID);
22:
23: export const switchNetwork = () => {
24:     return w.ethereum.request({
25:         method: 'wallet_switchEthereumChain',
26:         params: [{ chainId: web3.utils.toHex(CHAIN_ID) }]
27:     });
28: };
29:
30: export const getInstance = async () => {
31:     const contract = await getContract();
32:     return new web3.eth.Contract(contract.ABI, contract.address);
33: };
34:
35: export const getResaleTokens = async (eventId: number):
     Promise<ResaleToken[]> => {
36:     const instance = await getInstance();
37:     const resaleTokens = await
     instance.methods.getResaleTokens(eventId).call();
38:
39:     return resaleTokens.map((resaleToken: any) => {
40:         return {
41:             owner: resaleToken.owner,
42:             sold: resaleToken.sold
43:         };
44:     });
45: };
```

## Frontend Utilities Helper

File: frontend/src/helpers/utils.ts

```
01: import Web3 from "web3";
02:
03: export const prettyDate = (time: string | number) => {
04:     const date = new Date(time);
05:     const months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
     "Aug", "Sep", "Oct", "Nov", "Dec"];
06:
07:     return [
08:         date.getDate(),
09:         months[date.getMonth()],
10:         date.getFullYear()
11:     ].map(n => String(n).padStart(2, '0')).join(" ");
12: };
13:
14: export const range = (start: number, end: number) => {
15:     return Array(end - start + 1).fill(start).map((x, y) => x + y);
16: };
17:
18: export const omit = (object: any, key: string) => {
19:     const { [key]: _, ...rest } = object;
20:     return rest;
21: };
22:
23: export const gweiToEth = (gwei: number | string) =>
     Web3.utils.fromWei(gwei.toString(), "gwei");
24: export const gweiToWei = (gwei: number | string) =>
     Web3.utils.toWei(gwei.toString(), "gwei");
25: export const ethToGwei = (eth: number | string) => Number(eth) *
     Math.pow(10, 9);
```

## Frontend Admin Middleware Component

File: frontend/src/middleware/Admin.tsx

```tsx
import React, { useState } from "react";
import { Outlet, useOutletContext } from "react-router-dom";

const Admin = () => {
    return <Outlet context={useState(false)} />;
};

export const useAdmin = () => {
    return useOutletContext<[boolean,
 React.Dispatch<React.SetStateAction<boolean>>]>();
}

export default Admin;
```

## Frontend RequireAdmin Middleware Component

File: frontend/src/middleware/RequireAdmin.tsx

```tsx
import { Navigate } from "react-router-dom";
import routes from "../routes";
import { useAdmin } from "./Admin";

const RequireAdmin = ({ redirect: Redirect }: { redirect: React.FC }) => {
    const [admin] = useAdmin();

    if (!admin) {
        return (
            <Navigate
                to={routes.admin.login()}
                state={{ next: window.location.href.replace(window.location.origin, "") }}
                replace
            />
        );
    }

    return <Redirect />;
};

export default RequireAdmin;
```

## Frontend RequireWallet Middleware Component

File: frontend/src/middleware/RequireWallet.tsx

```tsx
import { Navigate } from "react-router-dom";
import routes from "../routes";
import { useAddress } from "./Wallet";

const RequireWallet = ({ redirect: Redirect }: { redirect: React.FC }) => {
    const [address] = useAddress();

    // Still attempting to connect to wallet
    if (address === null) {
        return <></>;
    }

    // Wallet present, but not connected
    if (!address.length) {
        return <Navigate to={routes.home()} replace />;
    }

    return <Redirect />;
};

export default RequireWallet;
```

## Frontend ShowNavBar Middleware Component

File: frontend/src/middleware/ShowNavBar.tsx

```tsx
import { Outlet } from "react-router-dom";
import NavBar from "../components/NavBar";
import { useAddress } from "./Wallet";

const ShowNavBar = () => {
    return (
        <>
            <NavBar />
            <Outlet context={useAddress()} />
        </>
    );
};

export default ShowNavBar;
```

## Frontend Wallet Middleware Component

File: frontend/src/middleware/Wallet.tsx

```
01: import React, { useEffect, useState } from "react";
02: import { Outlet, useOutletContext } from "react-router-dom";
03: import { useLocalStorage } from "usehooks-ts";
04: import Web3 from "web3";
05: import { isCorrectNetwork } from "../helpers/contract";
06:
07: const Wallet = () => {
08:     const [loggedOut, setLoggedOut] = useLocalStorage("loggedOut",
    false);
09:     const [address, setAddress] = useState<string | null>(null);
10:     const web3 = new Web3(Web3.givenProvider);
11:     const w = (window as any);
12:
13:     const checkMetaMaskPermission = async () => {
14:         if (!(w.ethereum && w.ethereum.isMetaMask)) {
15:             return false;
16:         }
17:
18:         if (!(web3 && web3.currentProvider)) {
19:             return false;
20:         }
21:
22:         const accounts = await web3.eth.getAccounts();
23:
24:         return (accounts.length > 0);
25:     }
26:
27:     const setupWallet = async () => {
28:         const disconnect = () => {
29:             setAddress("");
30:             setLoggedOut(true);
31:         };
32:
33:         w.ethereum.on('accountsChanged', (accounts: Array<string>)
    => {
34:             setAddress(accounts.length ? accounts[0] : "");
35:         });
36:
37:         w.ethereum.on('chainChanged', async (chainId: string) => {
38:             if (!isCorrectNetwork(web3.utils.toNumber(chainId))) {
39:                 disconnect();
40:             }
41:         });
42:
43:         if (Web3.givenProvider) {
44:             try {
45:                 const chainId = await web3.eth.getChainId();
46:
47:                 if (!isCorrectNetwork(chainId)) {
48:                     disconnect();
49:                     return;
50:                 }
51:
52:                 // Automatically login if correct network
53:                 const accounts = await web3.eth.requestAccounts();
54:                 setAddress(accounts[0]);
55:                 setLoggedOut(false);
56:             } catch (e) {
57:                 disconnect();
```

```
58:             }
59:         }
60:     };
61:
62:     useEffect(() => {
63:         checkMetaMaskPermission().then((hasPermission) => {
64:             if (hasPermission && !loggedOut) {
65:                 setupWallet();
66:             } else {
67:                 setAddress("");
68:             }
69:         });
70:     });
71:
72:     return <Outlet context={[address, setAddress]} />;
73: }
74:
75: export const useAddress = () => {
76:     return useOutletContext<[string | null,
     React.Dispatch<React.SetStateAction<string | null>>]>();
77: }
78:
79: export default Wallet;
```

## Frontend AdminCreateEditView Component

File: frontend/src/views/AdminCreateEditEventView.tsx

```
001: import { useFormik } from "formik";
002: import { useEffect, useState } from "react";
003: import { FileUploader } from "react-drag-drop-files";
004: import { Link, useNavigate, useParams } from "react-router-dom";
005: import * as Yup from "yup";
006: import AdminHeader from "../components/AdminHeader";
007: import Alert from "../components/Alert";
008: import BackCaret from "../components/BackCaret";
009: import Spinner from "../components/Spinner";
010: import { createEvent, Event, getEvent, updateEvent } from "../
     helpers/api";
011: import { ethToGwei, gweiToEth } from "../helpers/utils";
012: import { useAdmin } from "../middleware/Admin";
013: import routes from "../routes";
014:
015: enum Action {
016:    CREATE = "Create",
017:    EDIT = "Edit"
018: };
019:
020: const AdminCreateEditEventView = () => {
021:    const { id } = useParams();
022:    const [success, setSuccess] = useState<boolean>();
023:    const [disabled, setDisabled] = useState(false);
024:    const [refreshEvent, setRefreshEvent] = useState(false);
025:    const [event, setEvent] = useState<Event>();
026:    const [image, setImage] = useState<File>();
027:    const [errorMessage, setErrorMessage] = useState("");
028:    const [, setAdmin]= useAdmin();
029:    const navigate = useNavigate();
030:    const dateToString = (date: Date) => {
031:        return date.toISOString().substring(
032:            0,
033:            date.toISOString().length - 8)
034:        ;
035:    };
036:    const formik = useFormik({
037:        initialValues: {
038:            name: event?.name || "",
039:            artist: event?.artist || "",
040:            venue: event?.venue || "",
041:            city: event?.city || event?.name || "",
042:            time: (event?.city) ? dateToString(new Date(event.time))
     : "",
043:            price: (event?.price) ? gweiToEth(event.price) : "",
044:            quantity: event?.quantity || "",
045:            description: event?.description || "",
046:            genres: event?.genres.join("\n") || ""
047:        },
048:        validationSchema: Yup.object({
049:            name: Yup.string().required("Required"),
050:            artist: Yup.string().required("Required"),
051:            venue: Yup.string().required("Required"),
052:            city: Yup.string().required("Required"),
053:            time: Yup.string().required("Required"),
054:            price: Yup.number().required("Required").min(0),
055:            quantity:
     Yup.number().required("Required").min(event?.supplied || 1),
056:            description: Yup.string().required("Required"),
```

```
057:             genres: Yup.string().required("Required")
058:         }),
059:         onSubmit: async values => {
060:             setDisabled(true);
061:
062:             try {
063:                 const event: any = {
064:                     ...values,
065:                     id: Number(id),
066:                     time: new Date(values.time).getTime() / 1000,
067:                     price: ethToGwei(values.price),
068:                     quantity: Number(values.quantity)
069:                 };
070:
071:                 const formData = new FormData();
072:
073:                 if (image) {
074:                     formData.append("image", image);
075:                 }
076:
077:                 for (const [key, value] of Object.entries(event)) {
078:                     formData.append(key, String(value));
079:                 }
080:
081:                 if (action === Action.CREATE) {
082:                     formData.delete("id");
083:                     await createEvent(formData);
084:                     navigate(routes.admin.events());
085:                 } else {
086:                     formData.delete("price");
087:                     await updateEvent(formData);
088:                     setSuccess(true);
089:                     setRefreshEvent(!refreshEvent);
090:                 }
091:             } catch (e: any) {
092:                 console.error(e);
093:
094:                 // 401 Unauthorised
095:                 if (e?.response.status === 401) {
096:                     setAdmin(false);
097:                 } else {
098:                     setSuccess(false);
099:                 }
100:
101:                 if (e?.response?.data.message) {
102:                     setErrorMessage(e.response.data.message + ".");
103:                 }
104:             }
105:
106:             setDisabled(false);
107:         },
108:         enableReinitialize: true
109:     });
110:
111:     const action = (id === "create") ? Action.CREATE : Action.EDIT;
112:     const tomorrow = new Date();
113:
114:     tomorrow.setDate(tomorrow.getDate() + 1);
115:
116:     useEffect(() => {
117:         if (!id || action === Action.CREATE) {
118:             return;
119:         }
```

```
120:
121:         getEvent(Number(id))
122:             .then(setEvent)
123:             .catch(() => navigate(routes.admin.events()))
124:         ;
125:     }, [id, action, navigate, refreshEvent]);
126:
127:     const scrollToFirstFormikError = () => {
128:         const errors = Object.keys(formik.errors);
129:
130:         if (errors.length > 0) {
131:             const element = document.getElementsByName(errors[0])
    [0];
132:
133:             element.focus();
134:             element.scrollIntoView({
135:                 block: 'center',
136:                 inline: 'center'
137:             });
138:         }
139:     };
140:
141:     return (
142:         <div className="container mx-auto py-16 px-10">
143:             <BackCaret to={routes.admin.events()} />
144:             <AdminHeader className="mb-8" subtitle={action + "
    Event"} />
145:             <form
146:                 onSubmit={e => {
147:                     setSuccess(undefined);
148:                     scrollToFirstFormikError();
149:                     formik.handleSubmit(e);
150:                 }}
151:                 className="space-y-3"
152:             >
153:                 <Input
154:                     name="name"
155:                     label="Name"
156:                     type="text"
157:                     formik={formik}
158:                 />
159:                 <Input
160:                     name="artist"
161:                     label="Artist"
162:                     type="text"
163:                     formik={formik}
164:                 />
165:                 <Input
166:                     name="venue"
167:                     label="Venue"
168:                     type="text"
169:                     formik={formik}
170:                 />
171:                 <Input
172:                     name="city"
173:                     label="City"
174:                     type="text"
175:                     formik={formik}
176:                 />
177:                 <Input
178:                     name="time"
179:                     label="Time"
180:                     type="datetime-local"
```

```
181:                     min={dateToString(tomorrow)}
182:                     formik={formik}
183:                 />
184:                 <Input
185:                     name="price"
186:                     label="ETH Price"
187:                     type="number"
188:                     formik={formik}
189:                     step={0.001}
190:                     min={0}
191:                     disabled={action === Action.EDIT}
192:                 />
193:                 <Input
194:                     name="quantity"
195:                     label="Ticket Quantity"
196:                     type="number"
197:                     formik={formik}
198:                     min={event?.supplied || 1}
199:                 />
200:                 <Input
201:                     name="description"
202:                     label="Description"
203:                     type="textarea"
204:                     formik={formik}
205:                 />
206:                 <Input
207:                     name="genres"
208:                     label="Genres (1 per line)"
209:                     type="textarea"
210:                     formik={formik}
211:                 />
212:                 <div>
213:                     <div className="flex my-0.5">
214:                         <label className="uppercase text-sm text-
    indigo-500 mx-1">
215:                             Image (4:5 Aspect Ratio)
216:                         </label>
217:                     </div>
218:                     <FileUploader
219:                         handleChange={(file: File) =>
    setImage(file)}
220:                         types={["PNG", "JPG", "JPEG", "WEBP", "TIF",
    "GIF", "HEIC", "BMP"]}
221:                         required={action === Action.CREATE}
222:                     />
223:                     {
224:                         (event?.imageUrl || image) &&
225:                         <img
226:                             className="mt-3 rounded shadow-lg
    thumbnail"
227:                             src={image ? URL.createObjectURL((image
    as Blob)) : event?.imageUrl}
228:                             alt={event?.artist ||
    formik.values.artist}
229:                         />
230:                     }
231:                 </div>
232:                 {
233:                     success !== undefined && !success &&
234:                     <Alert
235:                         title="Error!"
236:                         message={`Failed to ${action.toLowerCase()}
    event. ${errorMessage ? "Reason: " + errorMessage : ""}`}
```

```
237:                         className="bg-red-50 text-red-700"
238:                     />
239:                 }
240:                 {
241:                     success &&
242:                     <Alert
243:                         message="Event successfully updated."
244:                         className="bg-green-50 text-green-700"
245:                     />
246:                 }
247:                 <div className="flex">
248:                     <div className="ml-auto flex space-x-2">
249:                         <Link to={routes.admin.events()}>
250:                             <button type="button" className="btn
    text-red-600 hover:text-red-800 disabled:text-gray-400"
    disabled={disabled}>
251:                                 Cancel
252:                             </button>
253:                         </Link>
254:                         <button type="submit" className="btn btn-
    basic" disabled={disabled}>
255:                             {disabled && <Spinner />}
256:                             Save
257:                         </button>
258:                     </div>
259:                 </div>
260:             </form>
261:         </div>
262:     );
263: };
264:
265: const Input = (props: {
266:     formik: any,
267:     name: string,
268:     type: string,
269:     label: string
270:     min?: string | number,
271:     max?: string | number,
272:     step?: string | number,
273:     disabled?: boolean
274: }) => {
275:     return (
276:         <div>
277:             <div className="flex my-0.5">
278:                 <label className="uppercase text-sm text-indigo-500
    mx-1">
279:                     {props.label}
280:                 </label>
281:                 {
282:                     props.formik.touched[props.name] &&
    props.formik.errors[props.name] &&
283:                     <div className="text-red-600 text-sm ml-auto">
284:                         {props.formik.errors[props.name]}
285:                     </div>
286:                 }
287:             </div>
288:             <div>
289:                 {
290:                     props.type === "textarea" &&
291:                     <textarea
292:                         placeholder={props.label}
293:                         className="input"
294:                         name={props.name}
```

```
295:                         value={props.formik.values[props.name]}
296:                         onChange={e => props.formik.handleChange(e)}
297:                         onBlur={e => props.formik.handleBlur(e)}
298:                         rows={5}
299:                         disabled={props.disabled}
300:                     />
301:                 }
302:                 {
303:                     props.type !== "textarea" &&
304:                     <input
305:                         type={props.type}
306:                         placeholder={props.label}
307:                         className="input"
308:                         name={props.name}
309:                         value={props.formik.values[props.name]}
310:                         onChange={e => props.formik.handleChange(e)}
311:                         onBlur={e => props.formik.handleBlur(e)}
312:                         min={props.min}
313:                         max={props.max}
314:                         step={props.step}
315:                         disabled={props.disabled}
316:                     />
317:                 }
318:
319:             </div>
320:         </div>
321:     );
322: };
323:
324: export default AdminCreateEditEventView;
```

## Frontend AdminEventsView Component

File: frontend/src/views/AdminEventsView.tsx

```
001: import { useEffect, useState } from "react";
002: import { MusicNote, QrCodeScan } from "react-bootstrap-icons";
003: import { Link, useSearchParams } from "react-router-dom";
004: import Web3 from "web3";
005: import AdminEventCard from "../components/AdminEventCard";
006: import AdminHeader from "../components/AdminHeader";
007: import Alert from "../components/Alert";
008: import PageError from "../components/PageError";
009: import PaginationButtons from "../components/PaginationButtons";
010: import SearchBar from "../components/SearchBar";
011: import { getEvents, GetEventsResponse } from "../helpers/api";
012: import routes from "../routes";
013:
014: const AdminEventsView = () => {
015:     const [eventsData, setEventsData] =
     useState<GetEventsResponse>();
016:     const [error, setError] = useState(false);
017:     const [updateEvents, setUpdateEvents] = useState(false);
018:     const [searchParams, setSearchParams] = useSearchParams();
019:     const offset = Number(searchParams.get("offset"));
020:     let timeout: NodeJS.Timeout;
021:
022:     const setSearch = (value: string) => {
023:         searchParams.set("search", value);
024:         setSearchParams(searchParams);
025:     };
026:
027:     const setOffset = (offset: number) => {
028:         searchParams.set("offset", offset.toString());
029:         setSearchParams(searchParams);
030:     };
031:
032:     const onEventCancel = () => {
033:         // Trigger events update either by resetting offset to 0 or
     updating `updateEvents`
034:         if (offset !== 0) {
035:             setOffset(0);
036:         } else {
037:             setUpdateEvents(!updateEvents);
038:         }
039:     };
040:
041:     useEffect(() => {
042:         const params = {
043:             offset: Number(searchParams.get("offset")) || 0,
044:             search: searchParams.get("search") || undefined
045:         };
046:
047:         getEvents(params)
048:             .then(setEventsData)
049:             .catch(() => setError(true))
050:         ;
051:     }, [searchParams, updateEvents]);
052:
053:     if (error) {
054:         return <PageError />;
055:     }
056:
057:     if (!eventsData) {
```

```
058:         return <></>;
059:     }
060:
061:     return (
062:         <div className="container mx-auto py-16 px-10">
063:             <AdminHeader subtitle="Events" className="mb-8 flex
    items-center">
064:                 <div className="ml-auto flex space-x-2">
065:                     <Link to={routes.admin.qr()} className="btn btn-
    basic">
066:                         <QrCodeScan className="mr-1" />
067:                         Scan QR
068:                     </Link>
069:                     <Link to={routes.admin.event("create")}
    className="btn btn-basic">
070:                         <MusicNote className="mr-1" />
071:                         Create Event
072:                     </Link>
073:                 </div>
074:             </AdminHeader>
075:             <div className="space-y-3 mb-4">
076:                 {
077:                     !Web3.givenProvider &&
078:                     <Alert
079:                         className="bg-red-50 text-red-900"
080:                         title="MetaMask client not installed!"
081:                         message="Functionality will be limited."
082:                     />
083:                 }
084:                 <SearchBar
085:                     onChange={e => {
086:                         // Only search if there has been no input
    for 300ms
087:                         clearTimeout(timeout);
088:
089:                         timeout = setTimeout(() => {
090:                             setSearch(e.target.value);
091:                         }, 300);
092:                     }}
093:                 />
094:                 {
095:                     !eventsData.events.length &&
096:                     <h2 className="font-medium py-5 italic
    uppercase">
097:                         No Events
098:                     </h2>
099:                 }
100:                 {eventsData.events.map(event => <AdminEventCard
    event={event} key={event.id} onCancel={() => onEventCancel()} />)}
101:             </div>
102:             <div className="flex justify-end space-x-2">
103:                 <PaginationButtons
104:                     prev={() => setOffset(eventsData.prevOffset)}
105:                     next={() => (typeof eventsData.nextOffset ===
    "number") && setOffset(eventsData.nextOffset)}
106:                     onChange={() => window.scrollTo(0, 0)}
107:                     prevDisabled={offset <= 0}
108:                     nextDisabled={!eventsData.nextOffset}
109:                     className="btn btn-basic"
110:                     hideWhenDisabled={true}
111:                 />
112:             </div>
113:         </div>
```

```
114:     );
115: };
116:
117: export default AdminEventsView;
```

## Frontend AdminLoginView Component

File: frontend/src/views/AdminLoginView.tsx

```tsx
import { useEffect, useState } from "react";
import { ArrowRightShort } from "react-bootstrap-icons";
import { useLocation, useNavigate } from "react-router-dom";
import { login } from "../helpers/api";
import { useAdmin } from "../middleware/Admin";
import routes from "../routes";

const AdminLoginView = () => {
    const { state } = useLocation();
    const [admin, setAdmin] = useAdmin();
    const [password, setPassword] = useState("");
    const [error, setError] = useState(false);
    const navigate = useNavigate();

    useEffect(() => {
        login()
            .then(() => setAdmin(true))
            .catch(() => setAdmin(false))
        ;
    }, [setAdmin])

    useEffect(() => {
        if (admin) {
            navigate(
                state?.next || routes.admin.events(),
                { replace: true }
            );
        }
    }, [admin, navigate, state]);

    const submit = async (e: React.FormEvent<HTMLFormElement>) => {
        e.preventDefault();

        login(password)
            .then(() => setAdmin(true))
            .catch(() => setError(true))
        ;
    };

    return (
        <div className="container mx-auto h-screen flex justify-
    center items-center">
            <form className="text-gray-600" onSubmit={e =>
    submit(e)}>
                <label className="text-xs uppercase  mx-1">Admin
    Login</label>
                <div className="flex items-center relative w-60">
                    <input
                        type="password"
                        className="input pr-9 shadow-sm"
                        placeholder="Password"
                        value={password}
                        onChange={e => setPassword(e.target.value)}
                        autoComplete="off"
                    />
                    <button type="submit" className="absolute inset-
    y-0 right-0 pr-3 flex items-center">
                        <ArrowRightShort />
                    </button>
```

```
56:                 </div>
57:                 <div className={"text-red-700 uppercase text-xs my-1
    mx-1 pointer-events-0 " + (error ? "opacity-1" : "opacity-0")}>
58:                     Incorrect Password
59:                 </div>
60:             </form>
61:         </div>
62:     );
63: };
64:
65: export default AdminLoginView;
```

## Frontend AdminQrScanView Component

File: frontend/src/views/AdminQrScanView.tsx

```
001: import { useEffect, useState } from 'react';
002: import { QrReader } from 'react-qr-reader';
003: import AdminHeader from '../components/AdminHeader';
004: import AlertModal from '../components/AlertModal';
005: import BackCaret from '../components/BackCaret';
006: import ConfirmationModal from '../components/ConfirmationModal';
007: import { QrData } from '../components/PurchaseCard';
008: import QuantityButton from '../components/QuantityButton';
009: import { eventToken } from '../helpers/api';
010: import { useAdmin } from '../middleware/Admin';
011: import routes from '../routes';
012:
013: const AdminQrScanView = () => {
014:     const [alertTitle, setAlertTitle] = useState("");
015:     const [alertMessage, setAlertMessage] = useState("");
016:     const [ticket, setTicket] = useState<QrData>();
017:     const [quantity, setQuantity] = useState(1);
018:     const [showConfirmationModal, setShowConfirmationModal] =
     useState(false);
019:     const [showAlertModal, setShowAlertModal] = useState(false);
020:     const [, setAdmin] = useAdmin();
021:     let scanned = showAlertModal;
022:
023:     useEffect(() => {
024:         // Reset quantity to 1 when ticket changes
025:         setQuantity(1);
026:     }, [ticket]);
027:
028:     const scan = (data: any) => {
029:         if (scanned || !data) {
030:             return;
031:         }
032:
033:         try {
034:             const ticket = JSON.parse(data.text);
035:             const keys = Object.keys(ticket);
036:
037:             // Check the QR code is for a ticket
038:             for (const key of ["signature", "eventId", "quantity"])
     {
039:                 if (!keys.includes(key)) {
040:                     return;
041:                 }
042:             }
043:
044:             setTicket(ticket);
045:             setShowConfirmationModal(true);
046:         } catch (e: any) {
047:             return;
048:         }
049:     };
050:
051:     const markTicketAsUsed = async () => {
052:         if (!ticket || quantity <= 0) {
053:             return;
054:         }
055:
056:         try {
```

```
057:             await eventToken(ticket.signature, ticket.eventId,
    quantity);
058:             setAlertTitle("Success");
059:             setAlertMessage("Tickets were validated and have now
    been marked as used.")
060:         } catch (e: any) {
061:             setAlertTitle("Error");
062:
063:             // 401 Unauthorised
064:             if (e?.response?.status === 401) {
065:                 setAdmin(false);
066:                 return;
067:             }
068:
069:             if (e?.response?.data?.message) {
070:                 setAlertMessage("Tickets were invalid. Reason: " +
    e.response.data.message);
071:             } else {
072:                 setAlertMessage("There was an issue whilst trying to
    mark tickets as used.");
073:             }
074:         }
075:
076:         setShowAlertModal(true);
077:     };
078:
079:     return (
080:         <div className="container mx-auto py-16 px-10">
081:             <BackCaret to={routes.admin.events()} />
082:             <AdminHeader subtitle="Scan QR" />
083:             <div className="w-1/2">
084:                 <QrReader
085:                     onResult={data => scan(data)}
086:                     constraints={{ facingMode: "environment" }}
087:                 />
088:             </div>
089:             {
090:                 showConfirmationModal &&
091:                 ticket &&
092:                 <ConfirmationModal
093:                     close={() => setShowConfirmationModal(false)}
094:                     action={() => markTicketAsUsed()}
095:                     title={`Event ID: ${ticket.eventId}`}
096:                     message="Please choose a quantity to mark as
    used."
097:                 >
098:                     <QuantityButton
099:                         quantity={ticket.quantity}
100:                         value={quantity}
101:                         className="mt-2 text-red-700"
102:                         onChange={e =>
    setQuantity(parseInt(e.target.value))}
103:                     />
104:                 </ConfirmationModal>
105:             }
106:             {
107:                 showAlertModal &&
108:                 <AlertModal
109:                     close={() => setShowAlertModal(false)}
110:                     title={alertTitle}
111:                     message={alertMessage}
112:                 />
113:             }
```

```
114:         </div>
115:     );
116: };
117:
118: export default AdminQrScanView;
```

## Frontend EventsView Component

File: frontend/src/views/EventsView.tsx

```
001: import { useEffect, useState } from "react";
002: import { CaretDownFill, CaretUpFill, XCircleFill } from 'react-
     bootstrap-icons';
003: import { useSearchParams } from "react-router-dom";
004: import CheckboxGroup, { CheckboxItem } from "../components/
     CheckboxGroup";
005: import EventCard from "../components/EventCard";
006: import PageError from "../components/PageError";
007: import PaginationButtons from "../components/PaginationButtons";
008: import { getEvents, GetEventsResponse, getGenres, getLocations }
     from "../helpers/api";
009: import { ethToGwei } from "../helpers/utils";
010:
011: const EventsView = () => {
012:     const [error, setError] = useState(false);
013:     const [eventsData, setEventsData] =
     useState<GetEventsResponse>();
014:     const [genres, setGenres] = useState<string[]>([]);
015:     const [locations, setLocations] = useState<string[]>([]);
016:     const [genreCheckboxItems, setGenreCheckboxItems] =
     useState<CheckboxItem[]>([]);
017:     const [locationCheckboxItems, setLocationCheckboxItems] =
     useState<CheckboxItem[]>([]);
018:     const [showFilters, setShowFilters] = useState(false);
019:     const [showGenres, setShowGenres] = useState(true);
020:     const [showLocations, setShowLocations] = useState(true);
021:     const [showPrice, setShowPrice] = useState(true);
022:     const [searchParams, setSearchParams] = useSearchParams();
023:     const search = searchParams.get("search") || undefined;
024:     const offset = Number(searchParams.get("offset"));
025:
026:     const updateFilterSearchParam = (key: string, value: any) => {
027:         searchParams.set(key, value.toString());
028:         searchParams.delete("offset");
029:         setSearchParams(searchParams);
030:     };
031:
032:     const setOffset = (offset: number) => {
033:         searchParams.set("offset", offset.toString());
034:         setSearchParams(searchParams);
035:     };
036:
037:     useEffect(() => {
038:         getGenres()
039:             .then(setGenres)
040:             .catch(() => setError(true))
041:         ;
042:
043:         getLocations()
044:             .then(setLocations)
045:             .catch(() => setError(true))
046:         ;
047:     }, []);
048:
049:     useEffect(() => {
050:         if (!genres.length || !locations.length) {
051:             return;
052:         }
053:
```

```
054:         const maxPrice = Number(searchParams.get("maxPrice"));
055:         const params = {
056:             offset: Number(searchParams.get("offset")) || 0,
057:             genres: searchParams.get("genres")?.split(",") || [],
058:             locations: searchParams.get("locations")?.split(",") ||
    [],
059:             maxPrice: (maxPrice > 0) ? ethToGwei(maxPrice) :
    undefined,
060:             search: searchParams.get("search") || undefined
061:         };
062:
063:         setGenreCheckboxItems(
064:             genres.map(genre => {
065:                 return { name: genre, checked:
    params.genres.includes(genre) };
066:             })
067:         );
068:
069:         setLocationCheckboxItems(
070:             locations.map(location => {
071:                 return { name: location, checked:
    params.locations.includes(location) };
072:             })
073:         );
074:
075:         getEvents(params)
076:             .then(setEventsData)
077:             .catch(() => setError(true))
078:         ;
079:     }, [searchParams, genres, locations]);
080:
081:     if (error) {
082:         return <PageError />;
083:     }
084:
085:     if (!eventsData) {
086:         return <></>;
087:     }
088:
089:     return (
090:         <div className="container py-16 px-10 mx-auto">
091:             {
092:                 search &&
093:                 <div className="mb-3 text-lg flex items-center">
094:                     <button type="button" onClick={() => {
095:                         searchParams.set("search", "");
096:                         setSearchParams(searchParams);
097:                     }}>
098:                         <XCircleFill className="mr-2 text-red-700
    hover:text-red-900" />
099:                     </button>
100:                     Showing results for: {search}
101:                 </div>
102:             }
103:             <button
104:                 type="button"
105:                 className="md:hidden text-gray-500 mb-2 text-xs
    uppercase"
106:                 onClick={() => setShowFilters(!showFilters)}
107:             >
108:                 {showFilters ? "Hide" : "Show"} Filters
109:             </button>
110:             <div className="grid grid-cols-12 gap-x-3">
```

```
111:                 <div className={"card-static col-span-12 mb-5
    md:mb-0 md:col-span-3 xl:col-span-2 h-fit md:block " + (!
    showFilters ? "hidden" : "")}>
112:                     <CheckboxItemsFilter
113:                         title="Genre"
114:                         items={genreCheckboxItems}
115:                         setItems={setGenreCheckboxItems}
116:                         show={showGenres}
117:                         setShow={setShowGenres}
118:                         onUpdate={(items) =>
    updateFilterSearchParam("genres", items.filter(i =>
    i.checked).map(i => i.name).join(","))}
119:                     />
120:                     <CheckboxItemsFilter
121:                         title="Location"
122:                         items={locationCheckboxItems}
123:                         setItems={setLocationCheckboxItems}
124:                         show={showLocations}
125:                         setShow={setShowLocations}
126:                         onUpdate={(items) =>
    updateFilterSearchParam("locations", items.filter(i =>
    i.checked).map(i => i.name).join(","))}
127:                     />
128:                     <button
129:                         type="button"
130:                         className={"filter-child flex items-center
    w-full " + (showPrice ? "" : "border-b-0")}
131:                         onClick={() => setShowPrice(!showPrice)}
132:                     >
133:                         Price
134:                         <div className="ml-auto">
135:                             {showPrice && <CaretUpFill size={12} />}
136:                             {!showPrice && <CaretDownFill
    size={12} />}
137:                         </div>
138:                     </button>
139:                     <div className={"filter-child " + (showPrice ?
    "" : "hidden")}>
140:                         <div className="relative shadow-sm my-1">
141:                             <div className="pointer-events-none
    absolute inset-y-0 left-0 flex items-center pl-3 text-xs text-
    gray-500">
142:                                 MAX ETH
143:                             </div>
144:                             <input
145:                                 type="number"
146:                                 className="input pl-20"
147:                                 placeholder="0"
148:                                 min={0}
149:                                 step="0.001"
150:                                 value={searchParams.get("maxPrice")
    || 0}
151:                                 onChange={e =>
    updateFilterSearchParam("maxPrice", e.target.value)}
152:                             />
153:                         </div>
154:                     </div>
155:                 </div>
156:                 <div className="space-y-3 col-span-12 md:col-span-9
    xl:col-span-10">
157:                     {
158:                         !eventsData.events.length &&
```

```
159:                         <h2 className="uppercase italic font-medium
    py-1 md:px-5">
160:                             No Events Found
161:                         </h2>
162:                     }
163:                     {eventsData.events.map(event => <EventCard
    key={event.id} event={event} />)}
164:                     <div className="flex justify-end space-x-2">
165:                         <PaginationButtons
166:                             prev={() =>
    setOffset(eventsData.prevOffset)}
167:                             next={() => (typeof
    eventsData.nextOffset === "number") ?
    setOffset(eventsData.nextOffset) : setOffset(0)}
168:                             onChange={() => window.scrollTo(0, 0)}
169:                             prevDisabled={offset <= 0}
170:                             nextDisabled={!eventsData.nextOffset}
171:                             className="btn btn-basic"
172:                             hideWhenDisabled={true}
173:                         />
174:                     </div>
175:                 </div>
176:             </div>
177:         </div>
178:     );
179: };
180:
181: const CheckboxItemsFilter = (props: {
182:     title: string,
183:     items: CheckboxItem[],
184:     setItems: React.Dispatch<React.SetStateAction<CheckboxItem[]>>,
185:     show: boolean,
186:     setShow: React.Dispatch<React.SetStateAction<boolean>>,
187:     onUpdate: (items: CheckboxItem[]) => any
188: }) => {
189:     return (
190:         <>
191:             <button type="button" className="filter-child flex
    items-center w-full" onClick={() => props.setShow(!props.show)}>
192:                 {props.title}
193:                 <div className="ml-auto">
194:                     {props.show && <CaretUpFill size={12} />}
195:                     {!props.show && <CaretDownFill size={12} />}
196:                 </div>
197:             </button>
198:             <div className={"filter-child " + (props.show ? "" :
    "hidden")}>
199:                 <CheckboxGroup items={props.items}
    dispatch={props.setItems} onUpdate={(items) =>
    props.onUpdate(items)} />
200:             </div>
201:         </>
202:     )
203: };
204:
205: export default EventsView;
```

## Frontend HomeView Component

File: frontend/src/views/HomeView.tsx

```tsx
import { Link } from "react-router-dom";
import routes from "../routes";

const HomeView = () => {
    return (
        <div className="home-page">
            <div className="py-32 px-6 flex justify-center text-
 center">
                <div>
                    <h1 className="mb-5 text-white text-shadow">
                        Blockchain Based Ticketing
                    </h1>
                    <div className="flex justify-center">
                        <Link to={routes.events()}>
                            <div className="btn text-white">
                                Browse Events
                            </div>
                        </Link>
                    </div>
                </div>
            </div>
        </div>
    );
}

export default HomeView;
```

## Frontend PurchasesView Component

File: frontend/src/views/PurchasesView.tsx

```tsx
import { useEffect, useState } from "react";
import LoadingCard from "../components/LoadingCard";
import PageError from "../components/PageError";
import PurchaseCard from "../components/PurchaseCard";
import { getPurchases, Purchase } from "../helpers/api";
import { useAddress } from "../middleware/Wallet";

enum PurchaseType {
    UPCOMING = "upcoming",
    SELLING = "selling",
    EXPIRED = "expired",
    CANCELLED = "cancelled"
};

const PurchasesView = () => {
    const [error, setError] = useState(false);
    const [address] = useAddress();
    const [purchaseData, setPurchaseData] = useState<PurchaseData>();
    const [updatePurchases, setUpdatePurchases] = useState(false);
    const [purchaseType, setPurchaseType] = useState<PurchaseType>(PurchaseType.UPCOMING);
    const [hasPurchases, setHasPurchases] = useState(false);

    useEffect(() => {
        if (!address) {
            return;
        }

        getPurchases(address)
            .then(purchases => {
                const purchaseData: PurchaseData = {
                    expired: [],
                    upcoming: [],
                    selling: [],
                    cancelled: []
                };

                for (const purchase of purchases) {
                    if (purchase.event.cancelled) {
                        purchaseData.cancelled.push(purchase);
                    } else if (purchase.expired) {
                        purchaseData.expired.push(purchase);
                    } else if (purchase.forSale) {
                        purchaseData.selling.push(purchase);
                    } else {
                        purchaseData.upcoming.push(purchase);
                    }
                }

                setPurchaseData(purchaseData);
            })
            .catch(() => setError(true))
        ;
    }, [address, updatePurchases]);

    useEffect(() => {
        if (!purchaseData) {
            return;
```

```
058:         }
059:
060:         if (purchaseData[purchaseType].length) {
061:             setHasPurchases(true);
062:             return;
063:         }
064:
065:         // If the selected purchase type has no data, try to select
     one that does
066:         for (const type of Object.values(PurchaseType)) {
067:             if (purchaseData[type].length) {
068:                 setPurchaseType(_ => type);
069:                 return;
070:             }
071:         }
072:
073:         // All purchase types had no data
074:         setHasPurchases(false);
075:     }, [purchaseData, purchaseType])
076:
077:     if (error) {
078:         return <PageError />;
079:     }
080:
081:     return (
082:         <div className="container mx-auto py-16 px-10 space-y-3">
083:             <div className="flex space-x-5 mb-5">
084:                 {
085:                     purchaseData &&
086:                     Object.values(PurchaseType)
087:                         .filter(purchaseType =>
     purchaseData[purchaseType].length) // Do not show tabs with no
     items
088:                         .map((type, i) => {
089:                             const border = (type === purchaseType) ?
     "border-indigo-500" : "border-transparent";
090:
091:                             return (
092:                                 <button
093:                                     type="button"
094:                                     key={i}
095:                                     onClick={() =>
     setPurchaseType(type)}
096:                                     className={"menu-tab capitalize
     " + border}
097:                                 >
098:                                     {type}
099:                                 </button>
100:                             );
101:                         })
102:                 }
103:             </div>
104:             {
105:                 purchaseData !== undefined && !hasPurchases &&
106:                 <h2 className="italic uppercase text-center font-
     medium">No Purchases Found</h2>
107:             }
108:             {
109:                 purchaseData === undefined &&
110:                 <LoadingCard />
111:             }
112:             {
113:                 purchaseData &&
```

```
114:                 purchaseData[purchaseType].map((purchase, i) => {
115:                     return (
116:                         <PurchaseCard
117:                             key={i}
118:                             purchase={purchase}
119:                             onChange={() => setUpdatePurchases(!
    updatePurchases)}
120:                         />
121:                     );
122:                 })
123:             }
124:         </div>
125:     );
126: };
127:
128: type PurchaseData = {
129:     expired: Purchase[],
130:     upcoming: Purchase[],
131:     selling: Purchase[],
132:     cancelled: Purchase[]
133: };
134:
135: export default PurchasesView;
```

## Frontend SingleEventResaleView Component

File: frontend/src/views/SingleEventResaleView.tsx

```
001: import { useEffect, useState } from "react";
002: import { useNavigate, useParams } from "react-router-dom";
003: import Alert from "../components/Alert";
004: import BackCaret from "../components/BackCaret";
005: import PageError from "../components/PageError";
006: import Spinner from "../components/Spinner";
007: import { Event, getEvent } from "../helpers/api";
008: import { getInstance, getResaleTokens, ResaleToken } from "../
     helpers/contract";
009: import { gweiToEth, gweiToWei } from "../helpers/utils";
010: import { useAddress } from "../middleware/Wallet";
011: import routes from "../routes";
012:
013: const SingleEventResaleView = () => {
014:     const { id } = useParams();
015:     const [event, setEvent] = useState<Event>();
016:     const [resaleTokens, setResaleTokens] =
     useState<ResaleToken[]>([]);
017:     const [address] = useAddress();
018:     const [error, setError] = useState(false);
019:     const [success, setSuccess] = useState<boolean>();
020:     const [updateResaleTokens, setUpdateResaleTokens] =
     useState(false);
021:     const navigate = useNavigate();
022:
023:     useEffect(() => {
024:         if (!address) {
025:             return;
026:         }
027:
028:         getResaleTokens(Number(id))
029:             .then(resaleTokens => {
030:                 const r = resaleTokens.filter(r => !r.sold &&
     r.owner !== address);
031:
032:                 if (r.length > 0) {
033:                     setResaleTokens(r);
034:                 } else {
035:                     navigate(routes.event(Number(id)), { replace:
     true })
036:                 }
037:             })
038:             .catch(() => setError(true))
039:         ;
040:     }, [id, address, updateResaleTokens, navigate]);
041:
042:     useEffect(() => {
043:         getEvent(Number(id))
044:             .then(setEvent)
045:             .catch(() => setError(true))
046:         ;
047:     }, [id]);
048:
049:     if (error) {
050:         return <PageError />;
051:     }
052:
053:     if (!event || !address) {
054:         return <></>;
```

```
055:     }
056:
057:     return (
058:         <div className="container mx-auto p-10">
059:             <BackCaret to={routes.event(Number(id))} />
060:             <div>
061:                 <h1 className="italic uppercase font-bold">
062:                     {event.artist}
063:                 </h1>
064:                 <div className="text-2xl mb-8 italic">
065:                     {event.name}
066:                 </div>
067:                 <div className="text-2xl uppercase font-bold">
068:                     Price
069:                 </div>
070:                 <h2 className="font-bold text-indigo-500 mb-8">
071:                     {gweiToEth(event.price)} ETH
072:                 </h2>
073:             </div>
074:             {
075:                 success &&
076:                 <Alert
077:                     title="Nice!"
078:                     message="Your tickets were successfully
    purchased!"
079:                     className="bg-green-50 text-green-800"
080:                 />
081:             }
082:             <div className="divide-y divide-gray-300">
083:                 {
084:                     resaleTokens.map((resaleToken, i) => {
085:                         return (
086:                             <ResaleListing
087:                                 key={i}
088:                                 event={event}
089:                                 resaleToken={resaleToken}
090:                                 address={address}
091:                                 onBeforePurchase={() =>
    setSuccess(undefined)}
092:                                 onPurchaseSuccess={() => {
093:                                     setSuccess(true);
094:                                     setUpdateResaleTokens(!
    updateResaleTokens);
095:                                 }}
096:                             />
097:                         );
098:                     })
099:                 }
100:             </div>
101:         </div>
102:     );
103: };
104:
105: const ResaleListing = (props: {
106:     event: Event,
107:     resaleToken: ResaleToken,
108:     address: string,
109:     onBeforePurchase: () => any,
110:     onPurchaseSuccess: () => any
111: }) => {
112:     const [disabled, setDisabled] = useState(false);
113:
114:     const purchase = async () => {
```

```
115:         props.onBeforePurchase();
116:         setDisabled(true);
117:
118:         try {
119:             const instance = await getInstance();
120:
121:             await instance
122:                 .methods
123:                 .buyResaleToken(props.resaleToken.owner,
    Number(props.event.id))
124:                 .send({ from: props.address, value:
    gweiToWei(props.event.price) })
125:             ;
126:
127:             props.onPurchaseSuccess();
128:         } catch (e) {
129:             console.error(e);
130:         }
131:
132:         setDisabled(false);
133:     };
134:
135:     return (
136:         <div className="grid grid-cols-12 py-4 items-center">
137:             <div className="flex space-x-3 items-center col-span-12
    md:col-span-10">
138:                 <div className="uppercase font-medium text-sm bg-
    indigo-500 text-white py-1 px-2">
139:                     Resale
140:                 </div>
141:                 <div className="break-words w-3/5">
142:                     {props.resaleToken.owner}
143:                 </div>
144:             </div>
145:             <div className="col-span-12 md:col-span-2 flex">
146:                 <button
147:                     className="ml-auto btn btn-basic md:w-fit w-full
    mt-3 md:mt-0"
148:                     type="button"
149:                     disabled={disabled}
150:                     onClick={() => purchase()}
151:                 >
152:                     {disabled && <Spinner />}
153:                     Purchase
154:                 </button>
155:             </div>
156:         </div>
157:     );
158: };
159:
160: export default SingleEventResaleView;
```

## Frontend SingleEventView Component

File: frontend/src/views/SingleEventView.tsx

```
001: import { useEffect, useState } from "react";
002: import { Link, useParams } from "react-router-dom";
003: import Alert from "../components/Alert";
004: import ConnectWallet from "../components/ConnectWallet";
005: import GenrePill from "../components/GenrePill";
006: import PageError from "../components/PageError";
007: import QuantityButton from "../components/QuantityButton";
008: import Spinner from "../components/Spinner";
009: import { Event, getEvent } from "../helpers/api";
010: import { getInstance, getResaleTokens, ResaleToken } from "../
     helpers/contract";
011: import { gweiToEth, gweiToWei, prettyDate } from "../helpers/
     utils";
012: import { useAddress } from "../middleware/Wallet";
013: import routes from "../routes";
014:
015: const SingleEventView = () => {
016:     const { id } = useParams();
017:     const [error, setError] = useState(false);
018:     const [event, setEvent] = useState<Event>();
019:     const [resaleTokens, setResaleTokens] =
     useState<ResaleToken[]>([]);
020:     const [refreshEvent, setRefreshEvent] = useState(false);
021:     const [success, setSuccess] = useState<boolean>();
022:     const [locked, setLocked] = useState(false);
023:     const [address] = useAddress();
024:
025:     useEffect(() => {
026:         getEvent(Number(id))
027:             .then(setEvent)
028:             .catch(() => setError(true))
029:         ;
030:     }, [id]);
031:
032:     useEffect(() => {
033:         if (!address) {
034:             return;
035:         }
036:
037:         setLocked(false);
038:         getResaleTokens(Number(id))
039:             .then(resaleTokens => {
040:                 setResaleTokens(
041:                     resaleTokens.filter(r => !r.sold && r.owner !==
     address)
042:                 )
043:             })
044:             .catch(() => setError(true))
045:         ;
046:     }, [id, address, refreshEvent]);
047:
048:     if (error) {
049:         return <PageError />;
050:     }
051:
052:     if (!event) {
053:         return <></>;
054:     }
055:
```

```
056:     return (
057:         <div className="container mx-auto p-10">
058:             {
059:                 success &&
060:                 <Alert
061:                     title="Nice!"
062:                     message="Your tickets were successfully
    purchased!"
063:                     className="bg-green-50 text-green-800"
064:                 />
065:             }
066:             {
067:                 locked &&
068:                 <Alert
069:                     message="Please manually unlock your MetaMask
    wallet to allow us to connect to it."
070:                     className="bg-indigo-50 text-indigo-800"
071:                 />
072:             }
073:             <div className="grid grid-cols-1 py-10 lg:grid-cols-2 ">
074:                 <div className="grid-span-1 flex mb-16 lg:mb-0">
075:                     <div className="my-auto">
076:                         <img className="mx-auto shadow-lg w-10/12
    md:w-8/12 lg:w-10/12 xl:w-8/12 rounded" src={event.imageUrl}
    alt={event.artist} />
077:                     </div>
078:                 </div>
079:                 <div className="grid-span-1 flex">
080:                     <div className="my-auto w-full">
081:                         <h1 className="italic uppercase font-bold
    mb-1">
082:                             {event.artist}
083:                         </h1>
084:                         <div className="text-2xl mb-8 italic">
085:                             {event.name}
086:                         </div>
087:                         <div className="text-2xl">
088:                             {prettyDate(event.time)}
089:                         </div>
090:                         <div className="text-2xl mb-8 capitalize">
091:                             {event.venue}, {event.city}
092:                         </div>
093:                         <div className="text-2xl uppercase font-
    bold">
094:                             Price
095:                         </div>
096:                         <h2 className="font-bold text-indigo-500
    mb-8">
097:                             {gweiToEth(event.price)} ETH
098:                         </h2>
099:                         <div className="flex mb-8 space-x-2">
100:                             <PurchaseButton
101:                                 address={address}
102:                                 event={event}
103:                                 className="btn-basic"
104:                                 onBeforePurchase={() =>
    setSuccess(undefined)}
105:                                 onSuccess={() => {
106:                                     setSuccess(true);
107:                                     setRefreshEvent(!refreshEvent);
108:                                 }}
109:                                 onLocked={() => setLocked(true)}
```

```
110:                                 quantityRemaining={event.remaining
    || 0}
111:                             />
112:                             {
113:                                 address && resaleTokens.length > 0
    &&
114:                                 <Link
    to={routes.eventResale(Number(id))} type="button" className="btn
    btn-basic">
115:                                     View Resale
116:                                 </Link>
117:                             }
118:                         </div>
119:                         <div className="mb-8 text-lg">
120:                             {event.description}
121:                         </div>
122:                         <div className="flex-wrap flex">
123:                             {event.genres.map(genre => <GenrePill
    name={genre} key={genre} className="my-1 mr-2" />)}
124:                         </div>
125:                     </div>
126:                 </div>
127:             </div>
128:         </div>
129:     );
130: };
131:
132: const PurchaseButton = (props: {
133:     className?: string,
134:     address: string | null,
135:     event: Event,
136:     onBeforePurchase: () => any,
137:     onSuccess: () => any,
138:     onLocked: () => any,
139:     quantityRemaining: number
140: }) => {
141:     const [quantity, setQuantity] = useState(1);
142:     const [disabled, setDisabled] = useState(false);
143:
144:     if (!props.address) {
145:         return <ConnectWallet className={props.className}
    onLocked={() => props.onLocked()} />;
146:     }
147:
148:     if (props.quantityRemaining <= 0) {
149:         return (
150:             <button
151:                 className={"btn btn-basic " + props.className}
152:                 disabled={true}
153:             >
154:                 Sold Out
155:             </button>
156:         );
157:     }
158:
159:     const purchase = async () => {
160:         props.onBeforePurchase();
161:         setDisabled(true);
162:
163:         try {
164:             const contract = await getInstance();
165:
166:             await contract
```

```
167:                 .methods
168:                 .buyToken(props.event.id, quantity)
169:                 .send({
170:                     from: props.address,
171:                     value: gweiToWei(props.event.price * quantity)
172:                 })
173:             ;
174:
175:             props.onSuccess();
176:         } catch (e: any) {
177:             console.error(e)
178:         }
179:
180:         setDisabled(false);
181:     };
182:
183:     return (
184:         <QuantityButton
185:             className={props.className}
186:             quantity={Math.min(6, props.quantityRemaining)}
187:             value={quantity}
188:             onClick={() => purchase()}
189:             onChange={e => setQuantity(Number(e.target.value))}
190:             disabled={disabled}
191:         >
192:             {disabled && <Spinner />}
193:             Purchase
194:         </QuantityButton>
195:     );
196: };
197:
198: export default SingleEventView;
```

## Frontend App Router Component

File: frontend/src/App.tsx

```
01: import { Navigate, Route, Routes } from "react-router-dom";
02: import './index.scss';
03: import Admin from './middleware/Admin';
04: import RequireAdmin from './middleware/RequireAdmin';
05: import RequireWallet from './middleware/RequireWallet';
06: import ShowNavBar from './middleware/ShowNavBar';
07: import Wallet from './middleware/Wallet';
08: import routes from "./routes";
09: import AdminCreateEditEventView from './views/
    AdminCreateEditEventView';
10: import AdminEventsView from './views/AdminEventsView';
11: import AdminLoginView from './views/AdminLoginView';
12: import AdminQrScanView from './views/AdminQrScanView';
13: import EventsView from './views/EventsView';
14: import HomeView from "./views/HomeView";
15: import PurchasesView from './views/PurchasesView';
16: import SingleEventResaleView from './views/SingleEventResaleView';
17: import SingleEventView from './views/SingleEventView';
18:
19: const App = () => {
20:     return (
21:         <Routes>
22:             <Route element={<Wallet />}>
23:                 <Route element={<ShowNavBar />}>
24:                     <Route path={routes.home()} element={<HomeView /
    >} />
25:                     <Route path={routes.events()}
    element={<EventsView />} />
26:                     <Route path={routes.event()}
    element={<SingleEventView />} />
27:                     <Route path={routes.eventResale()}
    element={<RequireWallet redirect={SingleEventResaleView} />} />
28:                     <Route path={routes.purchases()}
    element={<RequireWallet redirect={PurchasesView} />} />
29:                     <Route path="*" element={<Navigate
    to={routes.home()} replace />} />
30:                 </Route>
31:             </Route>
32:             <Route element={<Admin />}>
33:                 <Route path={routes.admin.login()}
    element={<AdminLoginView />} />
34:                 <Route path={routes.admin.events()}
    element={<RequireAdmin redirect={AdminEventsView} />} />
35:                 <Route path={routes.admin.event()}
    element={<RequireAdmin redirect={AdminCreateEditEventView} />} />
36:                 <Route path={routes.admin.qr()}
    element={<RequireAdmin redirect={AdminQrScanView} />} />
37:             </Route>
38:         </Routes>
39:     );
40: }
41:
42: export default App;
```

## Frontend Index Entry Script

File: frontend/src/index.tsx

```
01: import React from 'react';
02: import ReactDOM from 'react-dom/client';
03: import App from './App';
04: import reportWebVitals from './reportWebVitals';
05: import { BrowserRouter } from 'react-router-dom';
06:
07: const root = ReactDOM.createRoot(
08:     document.getElementById('root') as HTMLElement
09: );
10:
11: root.render(
12:     <React.StrictMode>
13:         <BrowserRouter>
14:             <App />
15:         </BrowserRouter>
16:     </React.StrictMode>
17: );
18:
19:
20: // If you want to start measuring performance in your app, pass a
     function
21: // to log results (for example: reportWebVitals(console.log))
22: // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-
     vitals
23: reportWebVitals();
```

## Frontend Routes Script

File: frontend/src/routes.ts

```
01: const param = (name: string, value: any) => {
02:     return (value || `:${name}`);
03: };
04:
05: const routes = {
06:     home: () => "/",
07:     events: () => "/events",
08:     event: (id?: number) => "/events/" + param("id", id),
09:     eventResale: (id?: number) => `/events/${param("id", id)}/
    resale`,
10:     purchases: () => "/purchases",
11:     admin: {
12:         login: () => "/admin",
13:         events: () => "/admin/events",
14:         event: (id?: number | string) => "/admin/events/" +
    param("id", id),
15:         qr: () => "/admin/qr"
16:     }
17: };
18:
19: export default routes;
```

## Frontend Style Sheet

File: frontend/src/index.scss

```
01: @tailwind base;
02: @tailwind components;
03: @tailwind utilities;
04:
05: body {
06:     background-color: white;
07: }
08:
09: * {
10:     // outline: 1px rgba(255, 0, 0, .5) solid;
11: }
12:
13: h1 {
14:     font-size: 3rem;
15:     line-height: 3rem;
16: }
17:
18: h2 {
19:     font-size: 2rem;
20:     line-height: 2rem;
21: }
22:
23: .btn {
24:     @apply flex justify-center text-sm font-medium py-2 px-4
     rounded-md outline outline-1 items-center transition ease-in-out
     duration-300;
25: }
26:
27: .btn-base {
28:     @apply flex justify-center text-sm font-medium rounded-md items-
     center transition ease-in-out duration-300;
29: }
30:
31: .btn-basic[disabled], .btn-basic[disabled]:hover {
32:     @apply outline-gray-400 text-gray-400
33: }
34:
35: .btn-basic {
36:     @apply outline-indigo-500 text-indigo-500;
37: }
38:
39: .btn-basic:hover {
40:     @apply outline-indigo-700 text-indigo-700;
41: }
42:
43: .filter-child {
44:     @apply py-2 px-3;
45: }
46:
47: .filter-child:not(:last-child):not(.border-b-0) {
48:     @apply border-b;
49: }
50:
51: .card {
52:     @apply outline outline-1 outline-gray-300 rounded transition-all
     ease-in-out duration-300 hover:outline-indigo-500;
53: }
54:
55: .card-static {
```

```
56:     @apply outline outline-1 outline-gray-300 rounded;
57: }
58:
59: .menu-tab {
60:     @apply border-b-4 pb-1 font-medium;
61: }
62:
63: .input {
64:     @apply w-full rounded-md border-gray-300 text-sm focus:ring-0;
65: }
66:
67: .input[disabled] {
68:     opacity: 0.5;
69: }
70:
71: .input:focus {
72:     @apply border-indigo-500 ring-indigo-500 border;
73: }
74:
75: .home-page {
76:     height: calc(100vh - 64px);
77:     background-image: url("~/public/static/home.jpg");
78:     background-size: cover;
79:     background-repeat: no-repeat;
80:     background-attachment: fixed;
81:     background-position-x: center;
82:     opacity: 0.9;
83: }
84:
85: .text-shadow {
86:     text-shadow: 0 2px 4px rgba(0, 0, 0, 0.8);
87: }
88:
89: .thumbnail {
90:     height: 138px;
91:     width: 100px;
92: }
93:
94: img:not([src]):not([srcset]) {
95:     visibility: hidden;
96: }
```