NFTSuits

Cavit Çakır, Kaya Kapağan, Görkem Köse, Gökberk Yar

Instructor: Kamer Kaya

Outline

- ➔ What is NFT Suits?
- ➔ Main Features of Website & Gas Analysis
- ➔ Security Analysis
- ➔ Other Cool Stuff

What is NFT Suits?

- NFT Suits is a game centered around collectible unique items that are used to create one-of-a-kind avatars where the uniqueness is guaranteed by ERC721 protocol.

- In order to see the items and interact with the marketplace, user has to install MetaMask extension to her/his browser.

Main functionalities

Within her/his profile, a user can

- Set a username
- Withdraw her/his balance
- Create an avatar with the items that s/he owns by wearing
- Save the item and display it in the avatars page
- List his/her items with certain filtering options
- See his/her statistics

Within the item page, a user can

- Sell her/his item for a fixed price
- Cancel sale for his/her item
- Buy an item
- Create an auction for his/her item
- Cancel an auction for his/her item
- Accept the highest bid
- Bid on an item
- Withdraw his/her bid
- Wear his/her item under some condition
- Unwear his/her item under some condition
- List all of the items with certain filtering options

Mint NFT

```python
def mint(clothType, name, cid, rarity):
  print(name)
  transaction = contract.functions.mint(clothType, name, cid, rarity).buildTransaction({
      'gas': 800000,
      'gasPrice': w3.toWei('10', 'gwei'),
      'from': '0xFFc867A24F9cf53ed5514D94Cd0992329987132A',
      'nonce': w3.eth.getTransactionCount('0xFFc867A24F9cf53ed5514D94Cd0992329987132A'),
      #'value': w3.toWei(415992086870360064,'wei')
      })
  private_key = "c47c2537e651e436836c48ce252fe4ceabae53364e72f78f09c4592a33872bf6"
  signed_txn = w3.eth.account.signTransaction(transaction, private_key=private_key)
  w3.eth.sendRawTransaction(signed_txn.rawTransaction)

```

```python
import time
for item in items_to_mint:
  mint(**item)
  time.sleep(60)
```

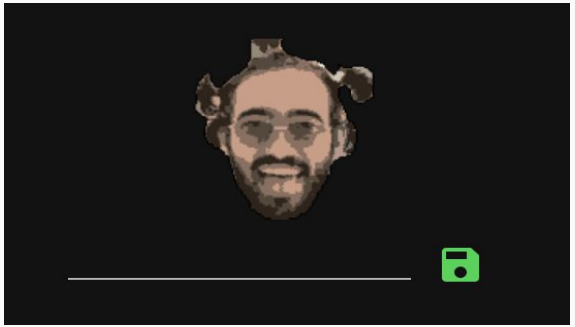
```
Elon Musk
Simba
Black Pants
Green Pants
Blue Pants
Dark Pink Pants
Light Pink Pants
Short Skirt
Striped Pants
Short
Long Skirt
Jeans
Striped Shirt
```

```
function mint(
    uint256 _clothType,
    string memory _name,
    string memory _cid,
    string memory _rarity
) public {
    require(isExist[_cid] == false, "Item link should be unique, for you to mint it");
    require(this.totalSupply() < maxSupply, "You cannot mint any more item since you already reached the maximum supply.");
    require(msg.sender ==owner, "Only owner can of this contract can mint, you are trying to some fraud.");
    require(_clothType == 1 || _clothType == 2 || _clothType == 3, "Invalid cloth type.");
    uint256 _id =
        nfts.push(
            nftData(
                _clothType,
                _name,
                _cid,
                _rarity,
                false,
                0,
                false,
                0,
                address(0x0),
                false
            )
        );
    _mint(msg.sender, _id);
    isExist[_cid] = true;


    emit nftTransaction(_id, "claimed", address(0x0), msg.sender, 0);

    putOnSale(_id, 10000000000000000);
}
```

Set username

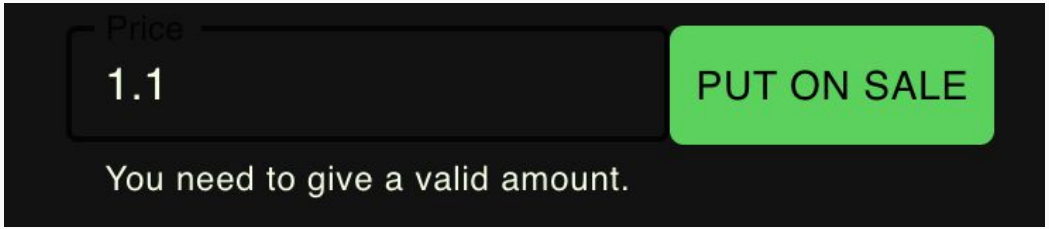
```
function setUsername(string memory _username) public {
    users[msg.sender].username = _username;
}
```

Gas used by
transaction ≈ 29,874



Put on sale

```
function putOnSale(uint256 _tokenId, uint256 _sellPrice) public {

    require(msg.sender == this.ownerOf(_tokenId), "You cannot put this item on sale, because you are not the owner of it.");
    require(nfts[_tokenId - 1].isOnSale == false, "Item is already on sale!");
    require(nfts[_tokenId - 1].isWearing == false, "You must unwear it first, then you can sell it." );

    nfts[_tokenId - 1].isOnSale = true;
    nfts[_tokenId - 1].sellPrice = _sellPrice;
    approve(address(this), _tokenId);

    emit nftTransaction(
        _tokenId,
        "On Sale",
        msg.sender,
        address(0x0),
        _sellPrice
    );
}
```



Gas used by transaction ≈ 81,517

Cancel sale

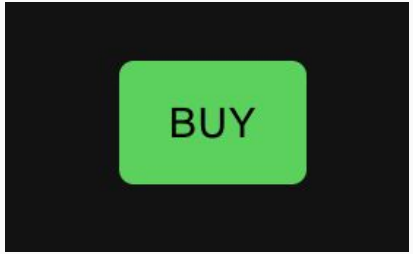
```
function cancelSale(uint256 _tokenId) public {
    require(nfts[_tokenId - 1].isOnSale == true, "Item should be on sale first, to be cancelled.");
    require(msg.sender == this.ownerOf(_tokenId), "You cannot cancel the sale of this item, because you are not the owner.");

    nfts[_tokenId - 1].isOnSale = false;
    nfts[_tokenId - 1].sellPrice = 0;

    emit nftTransaction(
        _tokenId,
        "Sale Cancelled",
        msg.sender,
        address(0x0),
        0
    );
}
```

CANCEL SALE

Gas used by transaction ≈ 20,088

Buy item

```
function buyFromSale(uint256 _tokenId) public payable {

    require(msg.sender != this.ownerOf(_tokenId), "You cannot buy your own item!");

    require(nfts[_tokenId - 1].isOnSale == true, "Item should be on sale for you to buy it.");

    require (nfts[_tokenId - 1].sellPrice <= msg.value, "The amount you tried to buy, is less than price.");

    require(this.getApproved(_tokenId) == address(this), "Seller did not give the allowance for us to sell this item, contact with seller.");
    address sellerAddress = this.ownerOf(_tokenId);
    this.safeTransferFrom(sellerAddress, msg.sender, _tokenId);
    nfts[_tokenId - 1].isOnSale = false;
    nfts[_tokenId - 1].sellPrice = 0;
    users[sellerAddress].userBalance = add256(users[sellerAddress].userBalance, msg.value);


    if (nfts[_tokenId - 1].maxBid > 0) {
        users[nfts[_tokenId - 1].maxBidder].userBalance = add256(users[nfts[_tokenId - 1].maxBidder].userBalance ,  nfts[_tokenId - 1].maxBid);
    }
    nfts[_tokenId - 1].maxBid = 0;
    nfts[_tokenId - 1].maxBidder = address(0x0);
    nfts[_tokenId - 1].isBiddable = false;

    emit nftTransaction(
        _tokenId,
        "sold",
        sellerAddress,
        msg.sender,
        msg.value
    );
}
```



Gas used by transaction ≈ 91,989

Put on auction

```
function putOnAuction(uint256 _tokenId) public {


    require(msg.sender == this.ownerOf(_tokenId), "Only owner of this item can put on sale" );
    require(nfts[_tokenId - 1].isWearing == false,  "You must unwear it first, then you can put it on auction.");
    require(nfts[_tokenId - 1].isBiddable == false, "This item is already on auction!");

    nfts[_tokenId - 1].isBiddable = true;
    nfts[_tokenId - 1].maxBid = 0;
    approve(address(this), _tokenId);

    emit nftTransaction(
        _tokenId,
        "Auction Starts",
        msg.sender,
        address(0x0),
        0
    );
}
```

START AN AUCTION

Gas used by transaction ≈ 61,455

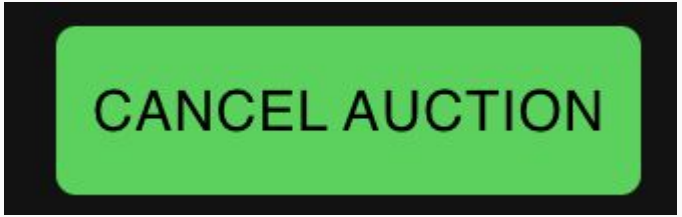
Accept highest bid

```
function acceptHighestBid(uint256 _tokenId) public {

    require(msg.sender == this.ownerOf(_tokenId), "You need to be owner of this item, to accept its highest bid.");
    require(nfts[_tokenId - 1].isBiddable == true , "Item should be biddable for you to accept its highest bid.");
    require(nfts[_tokenId - 1].maxBid > 0 ,"Max bid must be more than 0 to accept it, currently it is not!");
    require(nfts[_tokenId - 1].maxBidder != msg.sender, "Max bidder cannot be the same person as seller!");

    address buyer = nfts[_tokenId - 1].maxBidder;
    uint256 soldValue = nfts[_tokenId - 1].maxBid;
    this.safeTransferFrom(
        msg.sender,
        nfts[_tokenId - 1].maxBidder,
        _tokenId
    );
    users[msg.sender].userBalance = add256(users[msg.sender].userBalance, nfts[_tokenId - 1].maxBid);
    nfts[_tokenId - 1].maxBid = 0;
    nfts[_tokenId - 1].maxBidder = address(0x0);
    nfts[_tokenId - 1].isBiddable = false;
    nfts[_tokenId - 1].isOnSale = false;
    nfts[_tokenId - 1].sellPrice = 0;

    emit nftTransaction(
        _tokenId,
        "Sold From Auction",
        msg.sender,
        buyer,
        soldValue
    );
}
```

Gas used by transaction ≈ 68,646

Cancel auction

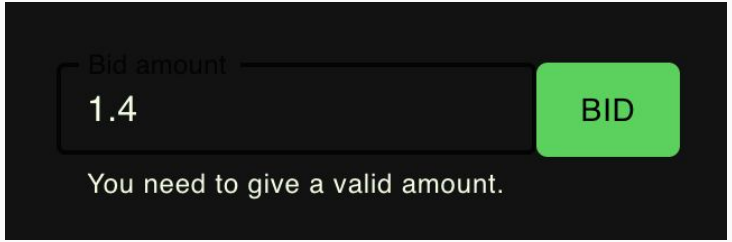
```
function cancelAuction(uint256 _tokenId) public {
    require(msg.sender == this.ownerOf(_tokenId), "You cannot cancel the auction of this item, because you are not the owner.");
    require(nfts[_tokenId - 1].isBiddable == true, "Item must be on auction before it can be canceled, currently it is not!");


    if (nfts[_tokenId - 1].maxBid > 0) {
        users[nfts[_tokenId - 1].maxBidder].userBalance = add256(users[nfts[_tokenId - 1].maxBidder].userBalance, nfts[_tokenId - 1].maxBid);
    }
    nfts[_tokenId - 1].isBiddable = false;
    nfts[_tokenId - 1].maxBid = 0;
    nfts[_tokenId - 1].maxBidder = address(0x0);

    emit nftTransaction(
        _tokenId,
        "Auction Cancelled",
        msg.sender,
        address(0x0),
        0
    );
}
```



Gas used by transaction ≈ 25,088

Bid on an item

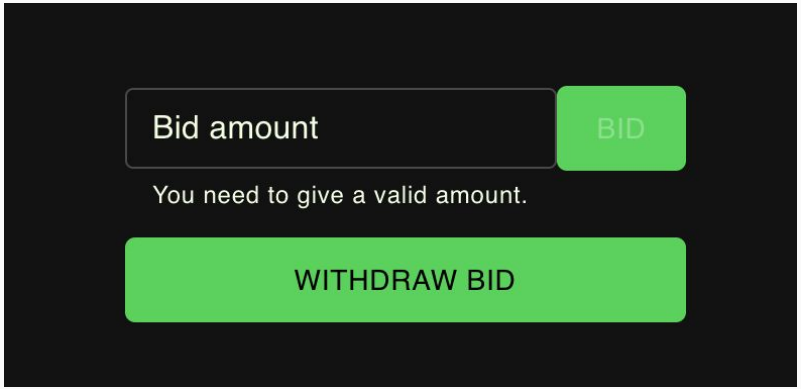
```
function bid(uint256 _tokenId) public payable {


    require(msg.value > 0, "You did not send any money");
    require(nfts[_tokenId - 1].isBiddable == true,"Item you tried to bid, is not biddable!");
    require(msg.value >= nfts[_tokenId - 1].maxBid, "The amount you tried to bid, is less than current max bid.");
    require(msg.sender != this.ownerOf(_tokenId), "You cannot bid your own item.");

    if (nfts[_tokenId - 1].maxBid > 0) {
        users[nfts[_tokenId - 1].maxBidder].userBalance = add256(users[nfts[_tokenId - 1].maxBidder].userBalance, nfts[_tokenId - 1].maxBid);
    }
    nfts[_tokenId - 1].maxBid = msg.value;
    nfts[_tokenId - 1].maxBidder = msg.sender;

    emit nftTransaction(
        _tokenId,
        "Bidded",
        msg.sender,
        ownerOf(_tokenId),
        msg.value
    );
}
```



Gas used by transaction ≈ 66,343

Withdraw bid

```
function withdrawBid(uint256 _tokenId) public {

    require(msg.sender == nfts[_tokenId - 1].maxBidder, "You must be the max bidder to withdraw your bid!");
    uint256 withdrawnValue = nfts[_tokenId - 1].maxBid;
    users[nfts[_tokenId - 1].maxBidder].userBalance = add256(users[nfts[_tokenId - 1].maxBidder].userBalance, nfts[_tokenId - 1].maxBid)
    nfts[_tokenId - 1].maxBid = 0;
    nfts[_tokenId - 1].maxBidder = address(0x0);

    emit nftTransaction(
        _tokenId,
        "Bid Withdrawn",
        msg.sender,
        address(0x0),
        withdrawnValue
    );
}
```



Gas used by transaction ≈ 30,751

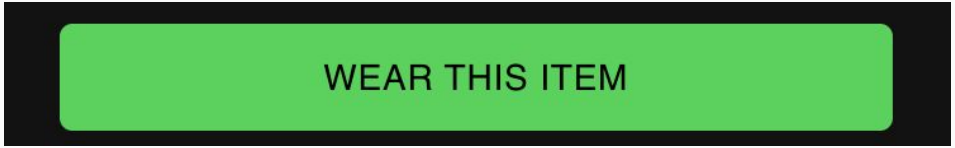
Withdraw money

```
function withdrawMoney(uint256 _amount) public {

    require(users[msg.sender].userBalance >= _amount, "You do not have enough balance to withdraw this amount");

    uint initialBalance = users[msg.sender].userBalance;
    users[msg.sender].userBalance = sub256(initialBalance, _amount);
    msg.sender.transfer(_amount);

}
```

Gas used by transaction ≈ 18,912

Wear item - single

```
function wearItem(uint256 _tokenId) public {
    require(this.ownerOf(_tokenId)        == msg.sender, "You are not the owner of this item, so you cannot wear it.");
    require(nfts[_tokenId - 1].isOnSale   == false,      "You cannot wear an item while it is on sale.");
    require(nfts[_tokenId - 1].isBiddable == false,      "You cannot wear an item while it is on auction.");


    if (nfts[_tokenId - 1].clothType == 1) {
        if (users[msg.sender].head != 0)
        {
            nfts[users[msg.sender].head - 1].isWearing = false;
        }
        users[msg.sender].head = _tokenId;
    } else if (nfts[_tokenId - 1].clothType == 2) {
        if (users[msg.sender].middle != 0)
        {
            nfts[users[msg.sender].middle - 1].isWearing = false;
        }
        users[msg.sender].middle = _tokenId;
    } else if (nfts[_tokenId - 1].clothType == 3) {
        if (users[msg.sender].bottom != 0)
        {
            nfts[users[msg.sender].bottom - 1].isWearing = false;
        }
        users[msg.sender].bottom = _tokenId;
    }
     nfts[_tokenId - 1].isWearing = true;
}
```



Gas used by transaction ≈ 78,555

Unwear item - single

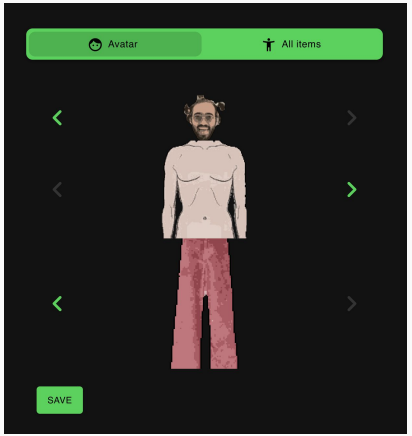
```
function unWearItem(uint256 _clothType) public {
    require(_clothType == 1 || _clothType == 2 || _clothType == 3, "Invalid cloth type.");

    if (_clothType == 1) {
        require(users[msg.sender].head !=0, "You must wear a head item first to unwear.");
        nfts[users[msg.sender].head - 1].isWearing = false;
        users[msg.sender].head = 0;
    } else if (_clothType == 2) {
        require(users[msg.sender].middle !=0, "You must wear a middle item first to unwear.");

        nfts[users[msg.sender].middle - 1].isWearing = false;
        users[msg.sender].middle = 0;
    } else if (_clothType == 3) {
        require(users[msg.sender].bottom !=0, "You must wear a bottom item first to unwear.");
        nfts[users[msg.sender].bottom - 1].isWearing = false;
        users[msg.sender].bottom = 0;
    }
}
```

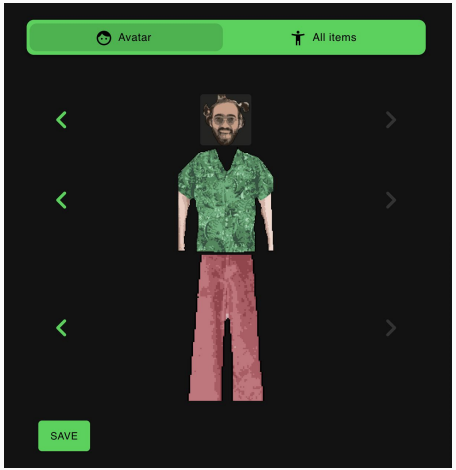
UNWEAR THIS ITEM

Gas used by transaction ≈ 17,105

Wear items

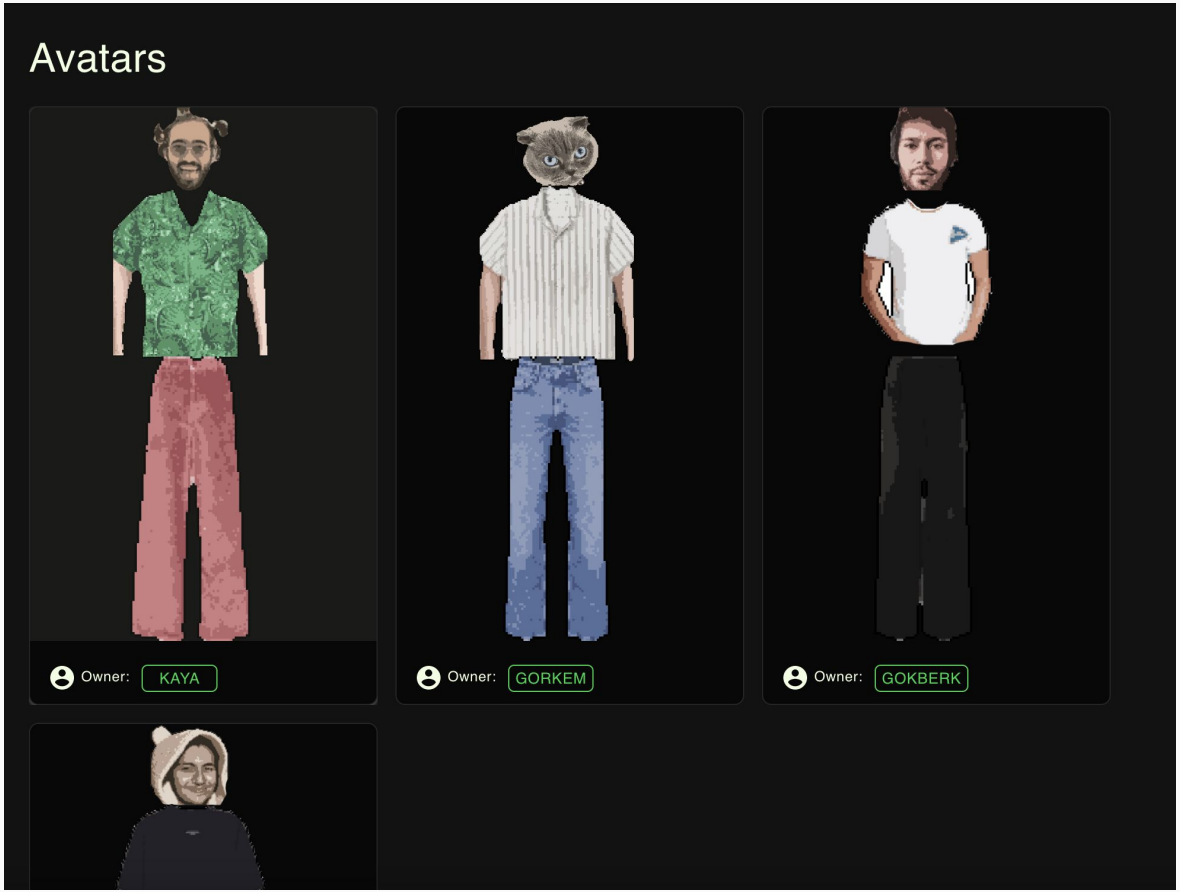
```
function wearItems(
    uint256 _headTokenId,
    uint256 _middleTokenId,
    uint256 _bottomTokenId
) public {
    require(
        _headTokenId == 0 ||
            (this.ownerOf(_headTokenId) == msg.sender &&
                nfts[_headTokenId - 1].clothType == 1)
    ,"you must be the owner or you tried not head item on head");
    require(
        _middleTokenId == 0 ||
            (this.ownerOf(_middleTokenId) == msg.sender &&
                nfts[_middleTokenId - 1].clothType == 2)
    ,"you must be the owner or you tried not middle item on middle");
    require(
        _bottomTokenId == 0 ||
            (this.ownerOf(_bottomTokenId) == msg.sender &&
                nfts[_bottomTokenId - 1].clothType == 3)
    ,"you must be the owner or you tried not bottom item on bottom");

    require(_headTokenId   == 0 || nfts[_headTokenId   - 1].isOnSale   == false, "head on sale, you cannot wear it!");
    require(_headTokenId   == 0 || nfts[_headTokenId   - 1].isBiddable == false, "head on bid, you cannot wear it!");
    require(_middleTokenId == 0 || nfts[_middleTokenId - 1].isOnSale   == false, "middle on sale, you cannot wear it!");
    require(_middleTokenId == 0 || nfts[_middleTokenId - 1].isBiddable == false, "middle on bid, you cannot wear it!");
    require(_bottomTokenId == 0 || nfts[_bottomTokenId - 1].isOnSale   == false, "bottom on sale, you cannot wear it!");
    require(_bottomTokenId == 0 || nfts[_bottomTokenId - 1].isBiddable == false, "middle on bid, you cannot wear it!");

    if (users[msg.sender].head != 0) {
        nfts[users[msg.sender].head - 1].isWearing = false;
    }
    if (users[msg.sender].middle != 0) {
        nfts[users[msg.sender].middle - 1].isWearing = false;
    }
    if (users[msg.sender].bottom != 0) {
        nfts[users[msg.sender].bottom - 1].isWearing = false;
    }


    if (_headTokenId != 0) {
        nfts[_headTokenId - 1].isWearing = true;
    }
    if (_middleTokenId != 0) {
        nfts[_middleTokenId - 1].isWearing = true;
    }
    if (_bottomTokenId != 0) {
        nfts[_bottomTokenId - 1].isWearing = true;
    }

    users[msg.sender].head   = _headTokenId;
    users[msg.sender].middle = _middleTokenId;
    users[msg.sender].bottom = _bottomTokenId;

}
```



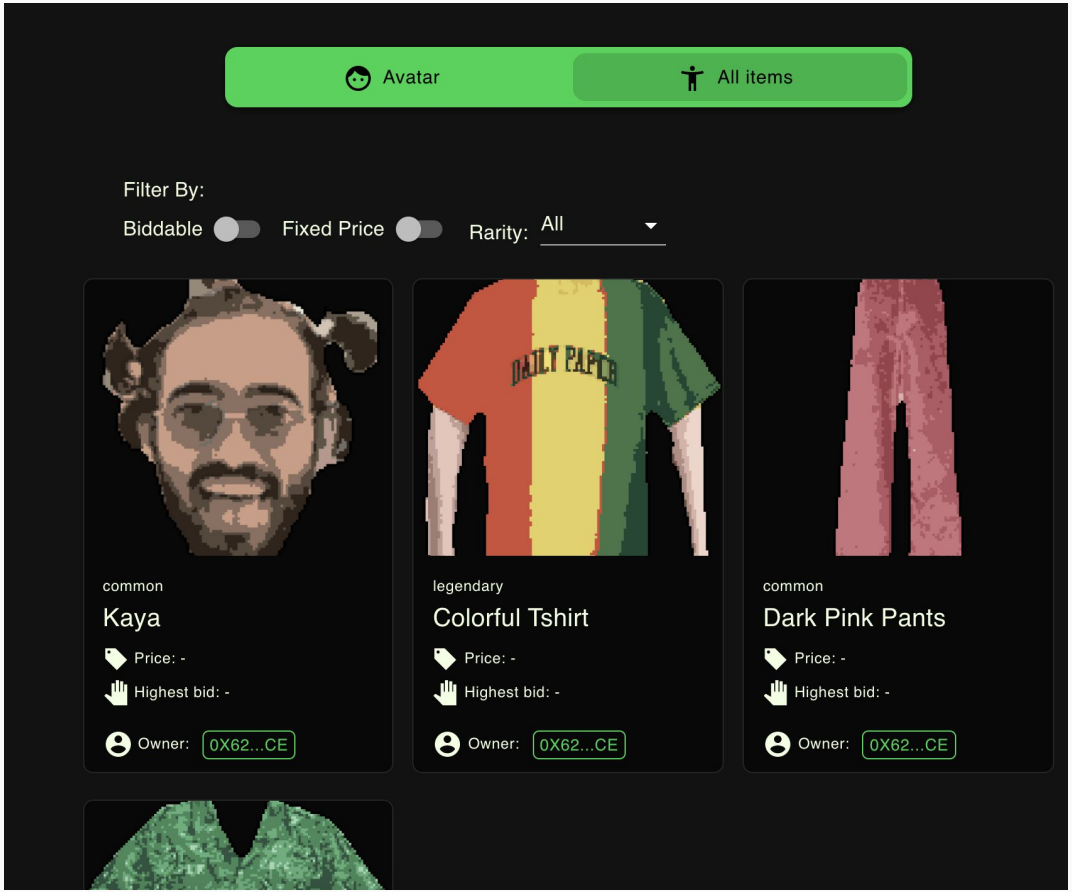


Gas used by transaction ≈ 108,235

Display avatars



List her/his items with certain filtering options

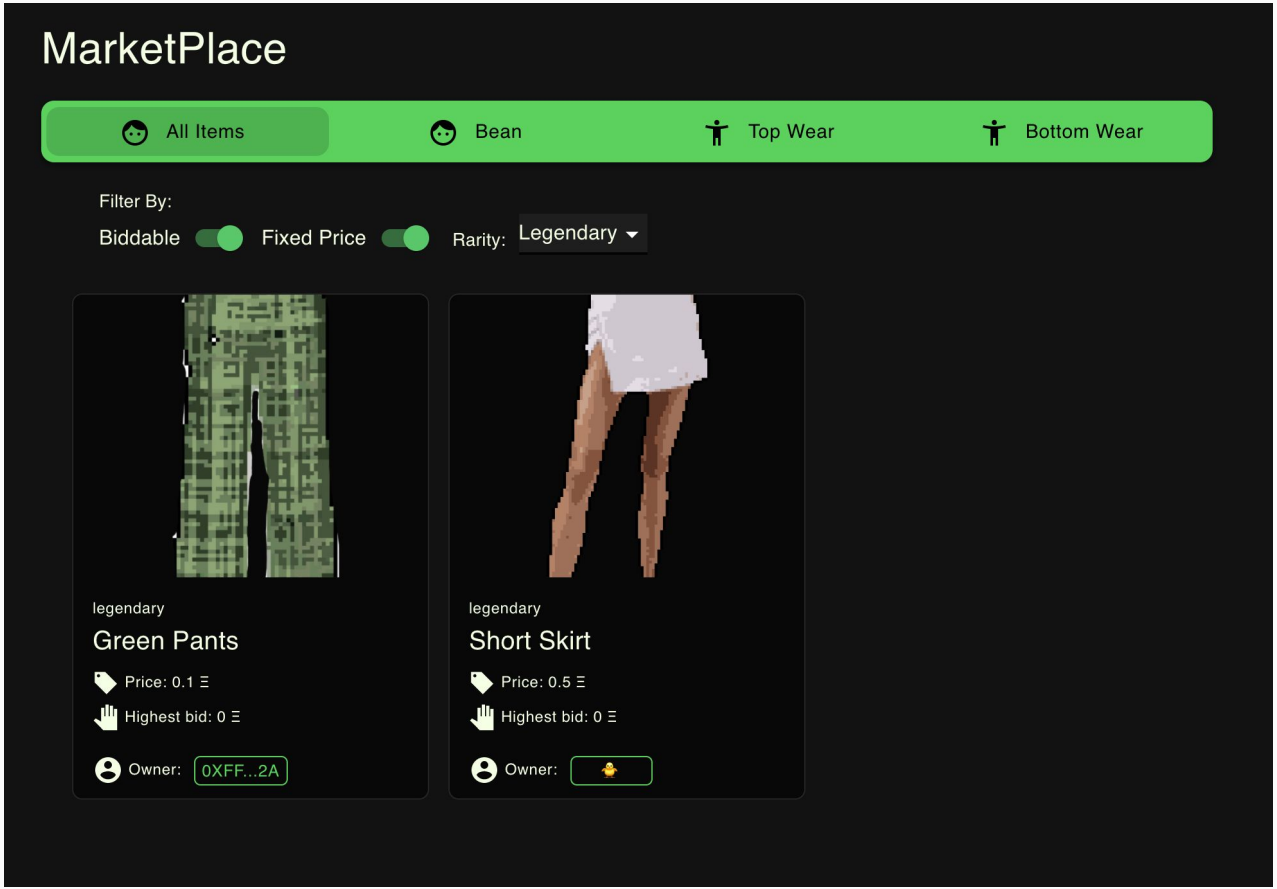


See the statistics

@Kaya

#Items	#combinations
4	12
Spent	#Bought
1.53 Ξ	4
Earned	#Sold
0 Ξ	0

List all items with certain filtering options



Sample Item Page

NFT Suits

AVATARS ALL ITEMS GORKEM

Striped Shirt

RARITY: EPIC

Owner: GORKEM

Price: -

Highest Bid: -

UNWEAR THIS ITEM

START AN AUCTION

Price PUT ON SALE

You need to give a valid amount.

Type	From	To	Amount	Txn
Bidded	0X20...96	0XFF...2A	0.00001 Ξ	0X66...38
Auction Starts	0XFF...2A	-	-	0X95...5E
Sale Cancelled	0XFF...2A	-	-	0X24...A3
On Sale	0XFF...2A	-	0.01 Ξ	0XC0...C7

Security Analysis

- Testing & Unauthorized access (i.e. selling something you don’t own)
- Overflow Attacks
- Re-entrance attacks
- Static Analysis

Testing & Unauthorized Access

```
require(msg.sender == this.ownerOf(_tokenId), "You cannot put this item on sale, because you are not the owner of it.");
```

```
  Contract: NFT
before called parent
    DEPLOYMENT
      ✓ deploys succesfully
      ✓ deployer is valid (57ms)
    MINT
      ✓ deployer try mint (1079ms)
      ✓ not deployer cannot mint (784ms)
      ✓ deployer cannot mint the same link (92ms)
    OWNER FEATURES
      ✓ owner cannot putOnSale if it is sale (95ms)
      ✓ not owner cannot cancelSale (94ms)
      ✓ owner can cancelSale (259ms)
      ✓ owner can wear item 1 (259ms)
      ✓ owner cannot wear another item 2 if it is on sale (83ms)
      ✓ owner can wear another item 2 if it is not on sale (570ms)
      ✓ not owner cannot wear (103ms)
      ✓ owner can unwear (164ms)
      ✓ owner cannot unwear not wear item  (51ms)
      ✓ someone cannot wear not existing token (163ms)
      ✓ owner cannot cancelSale if not on sale (91ms)
      ✓ not owner cannot putOnSale (92ms)
      ✓ someone cannot sell if item does not exist (163ms)
      ✓ owner cannot putOnSale if wearing the item (263ms)
      ✓ owner can putOnSale while item is not wearing (241ms)
      ✓ owner cannot buy his own item (116ms)
      ✓ someone cannot buy if item does not exist (143ms)
      ✓ buyer cannot buy if item not on sale (75ms)
      ✓ buyer cannot buy if sending value is less then sellPrice (114ms)
      ✓ buyer can buy owner's item (285ms)
      ✓ owner cannot withdrawMoney if withdraw amount is higher then owner has (90ms)
      ✓ owner cannot withdrawMoney if owner trys to trick us with negative values
      ✓ owner can withdrawMoney (101ms)
      ✓ owner cannot cancelAuction if not biddable (139ms)
      ✓ someone cannot cancelAuction if item not exists (94ms)
      ✓ owner cannot putOnAuction if wearing (289ms)
      ✓ someone cannot putOnAuction random one (95ms)
      ✓ not owner cannot putOnAuction (172ms)
      ✓ buyer cannot bid if item is not biddable (66ms)
      ✓ owner cannot accept bid if item is not biddable (118ms)
      ✓ owner can putOnAuction (222ms)
      ✓ buyer cannot bid on a biddable item with cash equals 0 at beggining or any time (53ms)
      ✓ owner cannot putOnAuction if already on auction (105ms)
      ✓ not owner cannot cancelAuction (105ms)
      ✓ owner can cancelAuction if biddable (248ms)
      ✓ owner can putOnAuction again (136ms)
      ✓ someone cannot bid item not exist  (155ms)
      ✓ owner cannot bid (105ms)
      ✓ owner cannot accept bid if maxBid == 0 (150ms)
      ✓ buyer can bid on a biddable item with cash biggrer than 0 at beggining (191ms)
      ✓ another buyer cannot bid if bid amount is smaller then max bid (57ms)
      ✓ another buyer (or same buyer) can bid on a biddable item with enough cash (max bid) (559ms)
      ✓ not maxBidder cannot withdrawBid (127ms)
      ✓ maxBidder can withdrawBid (340ms)
      ✓ buyer can bid on a biddable item with cash biggrer than 0 at beggining (158ms)
      ✓ not owner cannot accept bid (147ms)
      ✓ someone cannot accept bid of not existing item (207ms)
      ✓ owner can accept bid maxbid > 0 item is biddable (599ms)
      ✓ owner can withdrawMoney (238ms)
```

```
    WEAR/UNWEAR MULTIPLE ITEMS
      ✓ owner cannot wear head on sale (85ms)
      ✓ owner cannot wear bottom on sale  (131ms)
      ✓ owner cannot wear all if on sale (105ms)
      ✓ cancel sales to wear items (320ms)
      ✓ put on auction to show cannot wear items (410ms)
      ✓ owner cannot wear head on auction (91ms)
      ✓ owner cannot wear middle on auction (84ms)
      ✓ owner cannot wear bottom on auction (102ms)
      ✓ owner cannot wear all if on auction (121ms)
      ✓ cancel auction to test other cases without affected by auction (285ms)
      ✓ not owner cannot wear head (71ms)
      ✓ not owner cannot wear middle (110ms)
      ✓ not owner cannot wear bottom (78ms)
      ✓ not owner cannot wear all (68ms)
      ✓ not head cannot be weared as head (154ms)
      ✓ not middle cannot be weared as middle (114ms)
      ✓ not bottom cannot be weared  bottom (95ms)
      ✓ not matching cannot wear (153ms)
      ✓ wear all (366ms)
      ✓ cancel sales to wear other items (351ms)
      ✓ wear another all (664ms)
      ✓ owner can unwear all (353ms)


  76 passing (19s)
```

```
it("not owner cannot wear", async () => {
    await nftContract.wearItem(tokenId_1, {from: buyer}).should.be.rejected;
});
```

Testing & Unauthorized Access

```
it("owner can wear item 1", async () => {
    await nftContract.wearItem(tokenId_1, {from: deployer,});
    const userData = await nftContract.users.call(deployer);
    assert.equal(userData.head, tokenId_1);
    const newClothData = await nftContract.nfts.call(nftId_1);
    assert.equal(newClothData.isWearing, true);

});
```

```
it("owner cannot unwear not wear item ", async () => {
    await nftContract.unWearItem(1, {from: deployer}).should.be.rejected;
});
```

```
it("not owner cannot wear", async () => {
    await nftContract.wearItem(tokenId_1, {from: buyer}).should.be.rejected;
});
```

```
it("owner can wear another item 2 if it is not on sale", async () => {
    await nftContract.cancelSale(tokenId_2, {from: deployer});
    await nftContract.wearItem(tokenId_2, {from: deployer,});
    const userData = await nftContract.users.call(deployer);
    assert.equal(userData.head, tokenId_2);
    const oldClothData = await nftContract.nfts.call(nftId_1);
    assert.equal(oldClothData.isWearing, false);
    const newClothData = await nftContract.nfts.call(nftId_2);
    assert.equal(newClothData.isWearing, true);
});
```

```
it("owner can unwear", async () => {
    await nftContract.unWearItem(1, {from: deployer});
    const userData = await nftContract.users.call(deployer);
    assert.equal(userData.head, 0);
    const oldClothData = await nftContract.nfts.call(nftId_2);
    assert.equal(oldClothData.isWearing, false);
});
```

Overflow Attacks

```
function add256(uint256 a, uint256 b) internal pure returns (uint) {
    uint c = a + b;
    require(c >= a, "addition overflow");
    return c;
}

function sub256(uint256 a, uint256 b) internal pure returns (uint) {
    require(b <= a, "subtraction underflow");
    return a - b;
}
```

```
users[nfts[_tokenId - 1].maxBidder].userBalance = add256(users[nfts[_tokenId - 1].maxBidder].userBalance, nfts[_tokenId - 1].maxBid);
```

```
users[sellerAddress].userBalance = add256(users[sellerAddress].userBalance, msg.value);
```

Re-entrancy attacks

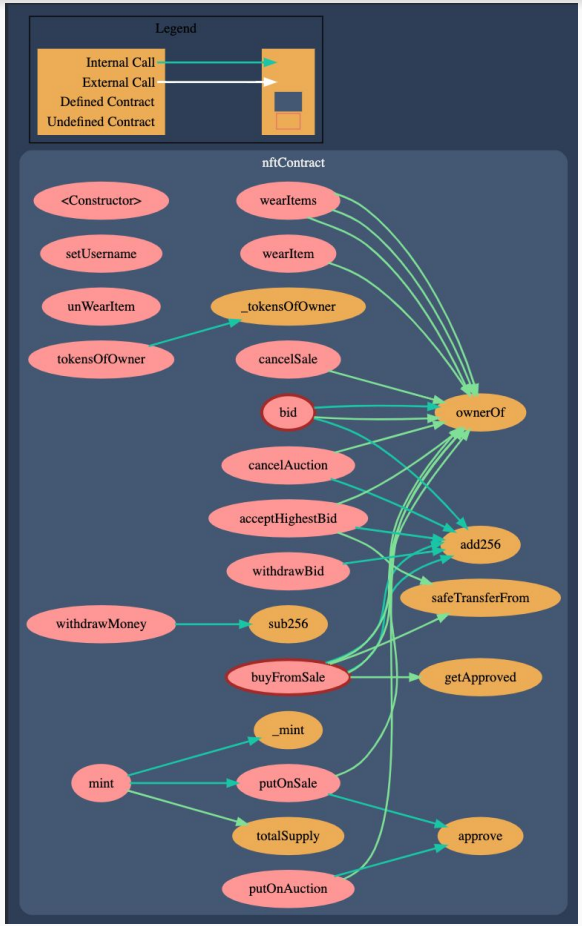
Inherited class (ERC721) has safeTransferFrom for NFTs.

```
this.safeTransferFrom(sellerAddress, msg.sender, _tokenId);
```

State related updates are done before external call(transfer), also used transfer for all money transfers.

```
function withdrawMoney(uint256 _amount) public {

    require(users[msg.sender].userBalance >= _amount, "You do not have enough balance to withdraw this amount");

    uint initialBalance = users[msg.sender].userBalance;
    users[msg.sender].userBalance = sub256(initialBalance, _amount);
    msg.sender.transfer(_amount);

}
```

Visualization of the Contract as a Graph (Sūrya)



Static Analysis

```
 [0;31mSeverity:    LOW
Pattern:     External Calls of Functions
Description: A public function that is never called within the
             contract should be marked as external
Type:        Violation
Contract:    nftContract
Line:        1330
Source:
>
>     function acceptHighestBid(uint256 _tokenId) public {
>                                   ^^^^^^^^^^^^^^^^^^^^
>         //msg.sender tokenId owner1 olmal1

 [0m
 [0;31mSeverity:    LOW
Pattern:     External Calls of Functions
Description: A public function that is never called within the
             contract should be marked as external
Type:        Violation
Contract:    nftContract
Line:        1368
Source:
>
>     function withdrawBid(uint256 _tokenId) public {
>                                     ^^^^^^^^^^^^^
>         //msg.sender maxBidder olmal1
```

Securify

```
ethsec@499d804af9cf:/home/trufflecon$ slither FullContract_v2.sol
INFO:Detectors:
Reentrancy in nftContract.acceptHighestBid(uint256) (FullContract_v2.sol#1330-1361):
        External calls:
        - this.safeTransferFrom(msg.sender,nfts[_tokenId - 1].maxBidder,_tokenId) (FullContract_v2.sol#1342-1346)
        State variables written after the call(s):
        - nfts[_tokenId - 1].maxBid = 0 (FullContract_v2.sol#1348)
        - nfts[_tokenId - 1].maxBidder = address(0x0) (FullContract_v2.sol#1349)
        - nfts[_tokenId - 1].isBiddable = false (FullContract_v2.sol#1350)
        - nfts[_tokenId - 1].isOnSale = false (FullContract_v2.sol#1351)
        - nfts[_tokenId - 1].sellPrice = 0 (FullContract_v2.sol#1352)
Reentrancy in nftContract.buyFromSale(uint256) (FullContract_v2.sol#1227-1257):
        External calls:
        - this.safeTransferFrom(sellerAddress,msg.sender,_tokenId) (FullContract_v2.sol#1237)
        State variables written after the call(s):
        - nfts[_tokenId - 1].isOnSale = false (FullContract_v2.sol#1238)
        - nfts[_tokenId - 1].sellPrice = 0 (FullContract_v2.sol#1239)
        - nfts[_tokenId - 1].maxBid = 0 (FullContract_v2.sol#1246)
        - nfts[_tokenId - 1].maxBidder = address(0x0) (FullContract_v2.sol#1247)
        - nfts[_tokenId - 1].isBiddable = false (FullContract_v2.sol#1248)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

Slither

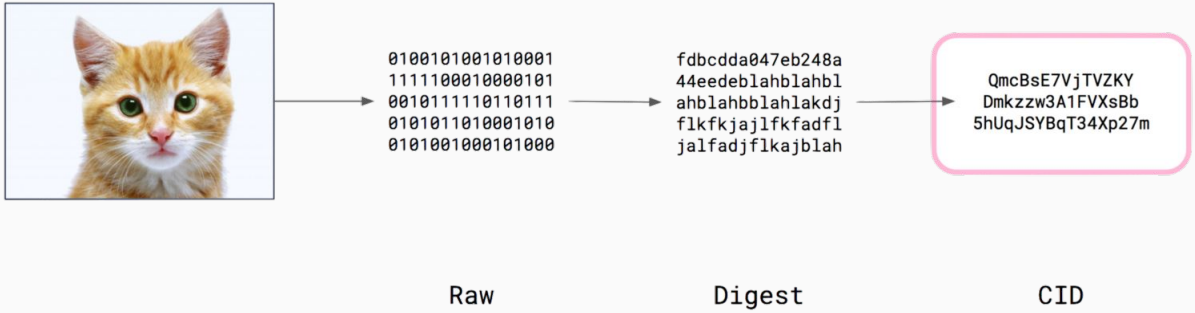
Static Analysis (Cont.)

```
[0;33mSeverity:    HIGH
Pattern:     Unhandled Exception
Description: The return value of statements that may return error
             values must be explicitly checked.
Type:        Warning
Contract:    nftContract
Line:        1402
Source:
>         require(isExist[_cid] == false, "Item link should be unique, for you to mint it");
>         require(this.totalSupply() < maxSupply, "You cannot mint any more item since you already reached the maximum supply.");
>                                                 ^^^^^^^^^^^^^^^^^^
>         require(msg.sender ==owner, "Only owner can of this contract can mint, you are trying to some fraud.");
```

Securify

```
[0m
[0;33mSeverity:    HIGH
Pattern:     Unhandled Exception
Description: The return value of statements that may return error
             values must be explicitly checked.
Type:        Warning
Contract:    nftContract
Line:        1063
Source:
>             _middleTokenId == 0 ||
>                 (this.ownerOf(_middleTokenId) == msg.sender &&
>                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
>                     nfts[_middleTokenId - 1].clothType == 2)

[0m
[0;33mSeverity:    HIGH
Pattern:     Unhandled Exception
Description: The return value of statements that may return error
             values must be explicitly checked.
Type:        Warning
Contract:    nftContract
Line:        1068
Source:
>             _bottomTokenId == 0 ||
>                 (this.ownerOf(_bottomTokenId) == msg.sender &&
>                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
>                     nfts[_bottomTokenId - 1].clothType == 3)
```

Extra Cool Stuff : IPFS



Pinata

Thank you!

https://nftsuits.com