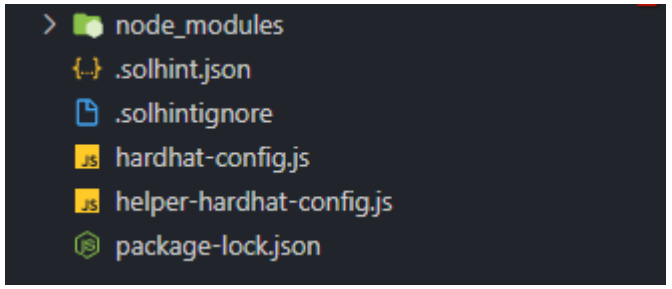
LESSON 15

NEXTJS NFT MARKETPLACE

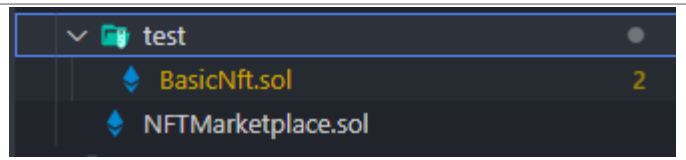
Hikmah Nisya - 1103184094
Radzis Araaf Jaya Jamaludin - 1103184234
Raudhatul Rafiqah Assyahiddini - 1103180225

Part I: NFT Marketplace Contracts (Setup)




```
+ chai@4.3.6
+ ethereum-waffle@3.4.4
+ ethers@5.6.9
+ hardhat-gas-reporter@1.0.8
+ hardhat-deploy@0.11.11
+ prettier@2.7.1
+ dotenv@16.0.1
+ solidity-coverage@0.7.21
+ prettier-plugin-solidity@1.0.0-beta.19
+ solhint@3.3.7
+ hardhat-deploy-ethers@0.3.0-beta.13 (as @nomiclabs/hardhat-ethers)
+ hardhat@2.9.9
+ @nomiclabs/hardhat-waffle@2.0.3
+ @nomiclabs/hardhat-etherscan@3.1.0
+ hardhat-contract-sizer@2.6.1
added 1979 packages from 878 contributors in 455.988s

163 packages are looking for funding
```

Part I: NFT Marketplace Contracts



NFTMarketplace.sol

contracts > NFTMarketplace.sol

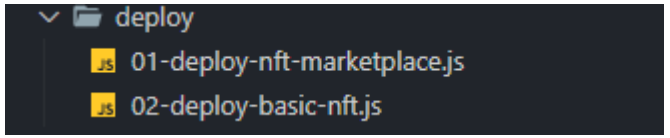
```sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

import "@openzeppelin/contracts/token/ERC721/IERC721.sol";
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";

error NFTMarketplace__PriceMustBeAboveZero();
error NFTMarketplace__NotApprovedForMarketplace();
error NFTMarketplace__AlreadyListed(address nftAddress, uint256 tokenId);
error NFTMarketplace__NotOwner();
error NFTMarketplace__NotListed(address nftAddress, uint256 tokenId);
error NFTMarketplace__PriceNotMet(address nftAddress, uint256 tokenId, uint256 price);
error NFTMarketplace__NoProceeds();

contract NFTMarketplace is ReentrancyGuard {
  struct Listing {
    uint256 price;
    address seller;
  }

  mapping(address => mapping(uint256 => Listing)) private s_listings;
  mapping(address => uint256) private s_proceeds;

  modifier notListed(
    address nftAddress,
    uint256 tokenId,
    address owner
  ) {
    Listing memory listing = s_listings[nftAddress][tokenId];
```

BasicNft.sol 2

contracts > test > BasicNft.sol

```sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

contract BasicNft is ERC721 {
  string public constant TOKEN_URI = "ipfs://bafybeig37ioir76s7mg5oobetncojcm3c3hxasyd4rvid4jqhy4gkaheg4/?
  uint256 private s_tokenCounter;

  event DogMinted(uint256 indexed tokenId);

  constructor() ERC721("Dogie", "DOG") {
    s_tokenCounter = 0;
  }

  function mintNft() public {
    _safeMint(msg.sender, s_tokenCounter);
    emit DogMinted(s_tokenCounter);
    s_tokenCounter = s_tokenCounter + 1;
  }

  function tokenURI(uint256 tokenId) public view override returns (string memory) {
    require(_exists(tokenId), "ERC721Metadata: URI query for nonexistent token");
    return TOKEN_URI;
  }

  function getTokenCounter() public view returns (uint256) {
    return s_tokenCounter;
  }
```

Part I: NFT Marketplace Contracts (deploy)



01-deploy-nft-marketplace.js

deploy > 01-deploy-nft-marketplace.js > ...

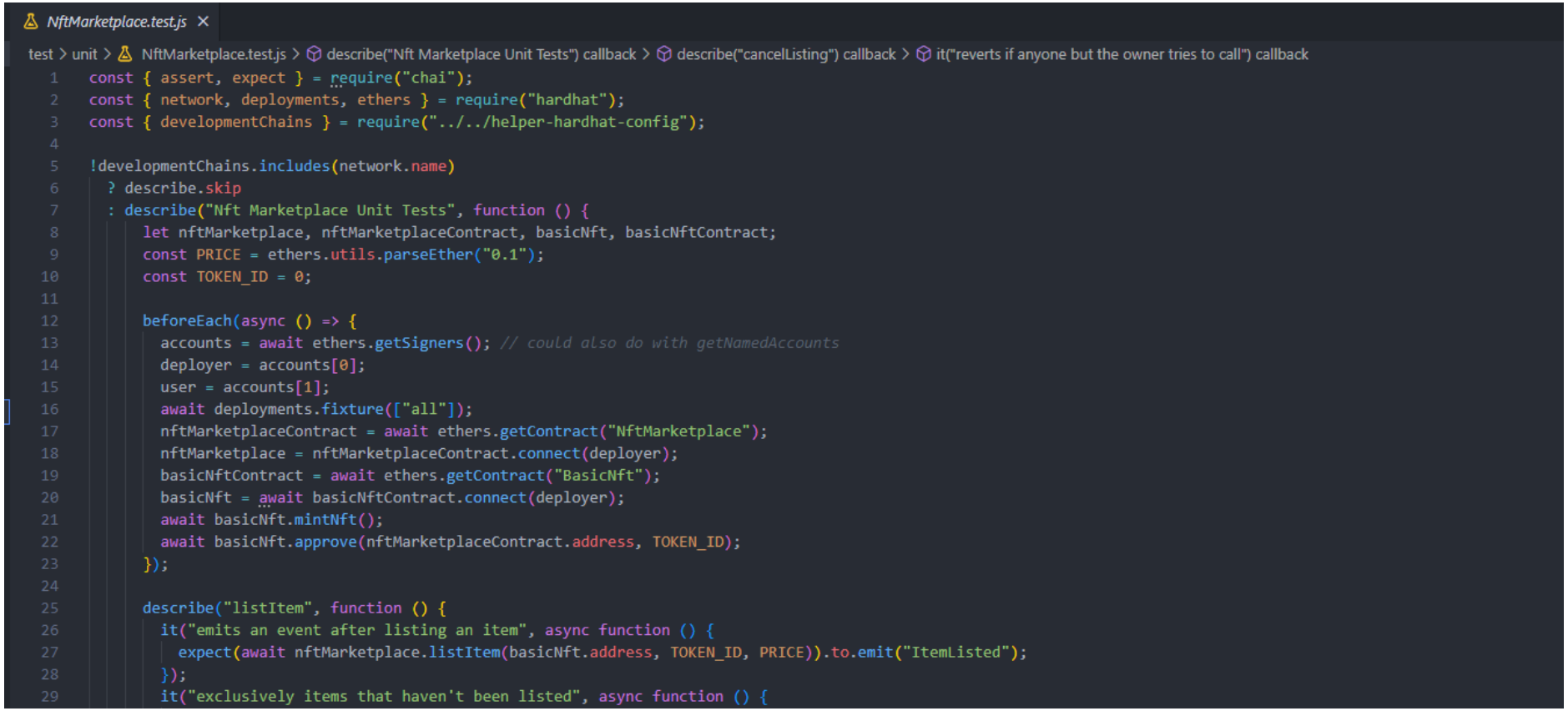
```
const { network } = require("hardhat");
const { developmentChains } = require("../helper-hardhat-config");
const { verify } = require("../utils/verify");

module.exports = async ({ getNamedAccounts, deployments }) => {
  const { deploy, log } = deployments;
  const { deployer } = await getNamedAccounts();

  args = [];

  const nftMarketplace = await deploy("NFTMarketplace", {
    from: deployer,
    args: args,
    log: true,
    waitConfirmations: network.config.blockConfirmation || 1,
  });

  if (!developmentChains.includes(network.name) && process.env.ETHERSCAN_API_K
    console.log("verifying");
    await verify(nftMarketplace.address, args);
  }
};

module.exports.tags = ["all", "nftmarketplace"];
```

02-deploy-basic-nft.js

deploy > 02-deploy-basic-nft.js > <unknown> > exports

```
const { network } = require("hardhat");
const { developmentChains } = require("../helper-hardhat-config");
const { verify } = require("../utils/verify");

module.exports = async ({ getNamedAccounts, deployments }) => {
  const { deploy, log } = deployments;
  const { deployer } = await getNamedAccounts();

  const args = [];
  const basicNft = await deploy("BasicNft", {
    from: deployer,
    args: args,
    log: true,
    waitConfirmations: network.config.blockConfirmation || 1,
  });

  if (!developmentChains.includes(network.name) && process.env.ETHERSCAN_API_K
    console.log("verifying");
    await verify(nftMarketplace.address, args);
  }
};

module.exports.tags = ["all", "basicNft"];
```

Part I: NFT Marketplace Contracts (unit test)



NftMarketplace.test.js

test > unit > NftMarketplace.test.js > describe("Nft Marketplace Unit Tests") callback > describe("cancelListing") callback > it("reverts if anyone but the owner tries to call") callback

```
const { assert, expect } = require("chai");
const { network, deployments, ethers } = require("hardhat");
const { developmentChains } = require("../../helper-hardhat-config");

!developmentChains.includes(network.name)
  ? describe.skip
  : describe("Nft Marketplace Unit Tests", function () {
      let nftMarketplace, nftMarketplaceContract, basicNft, basicNftContract;
      const PRICE = ethers.utils.parseEther("0.1");
      const TOKEN_ID = 0;

      beforeEach(async () => {
        accounts = await ethers.getSigners(); // could also do with getNamedAccounts
        deployer = accounts[0];
        user = accounts[1];
        await deployments.fixture(["all"]);
        nftMarketplaceContract = await ethers.getContract("NftMarketplace");
        nftMarketplace = nftMarketplaceContract.connect(deployer);
        basicNftContract = await ethers.getContract("BasicNft");
        basicNft = await basicNftContract.connect(deployer);
        await basicNft.mintNft();
        await basicNft.approve(nftMarketplaceContract.address, TOKEN_ID);
      });

      describe("listItem", function () {
        it("emits an event after listing an item", async function () {
          expect(await nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)).to.emit("ItemListed");
        });
        it("exclusively items that haven't been listed", async function () {
```