**WINNGOO Token Project - Complete Development & Deployment Guide**

**Prepared by Madhubala/16-01-2025**

***Table of Contents***

1. Project Setup

2. Development Environment Setup

3. Smart Contract Development

4. Testing

5. Deployment (Truffle)

6. Deployment (Hardhat)

7. Contract Verification

8. Frontend Development

9. Troubleshooting Guide

***1. Project Setup***

1.1 Create Project Directory

mkdir winngoo-token-project

cd winngoo-token-project

1.2 Initialize Node Project

npm init -y

1.3 Install Development Dependencies

# Install Truffle

npm install -g truffle

# Install project dependencies

npm install @openzeppelin/contracts@4.9.0

npm install @truffle/hdwallet-provider

npm install dotenv

npm install web3

# For testing

npm install chai

npm install @openzeppelin/test-helpers

1.4 Initialize Truffle Project

truffle init

This creates:

- `contracts/` - Directory for Solidity contracts

- `migrations/` - Directory for deployment scripts

- `test/` - Directory for test files

- `truffle-config.js` - Truffle configuration file

***2. Development Environment Setup***

2.1 Configure Environment Variables

Create `.env` file:

MNEMONIC="your twelve word mnemonic phrase here"

INFURA\_PROJECT\_ID="your-infura-project-id"

ETHERSCAN\_API\_KEY="your-etherscan-api-key"

POLYGONSCAN\_API\_KEY="your-polygonscan-api-key"

BSCSCAN\_API\_KEY="your-bscscan-api-key"

2.2 Configure Truffle

Update `truffle-config.js`:

```javascript

require('dotenv').config();

const HDWalletProvider = require('@truffle/hdwallet-provider');

module.exports = {

networks: {

development: {

host: "127.0.0.1",

port: 8545,

network\_id: "\*"

},

goerli: {

provider: () => new HDWalletProvider(

process.env.MNEMONIC,

`https://goerli.infura.io/v3/${process.env.INFURA\_PROJECT\_ID}`

),

network\_id: 5,

gas: 5500000,

confirmations: 2,

timeoutBlocks: 200,

skipDryRun: true

},

bsc\_testnet: {

provider: () => new HDWalletProvider(

process.env.MNEMONIC,

`https://data-seed-prebsc-1-s1.binance.org:8545`

),

network\_id: 97,

confirmations: 10,

timeoutBlocks: 200,

skipDryRun: true

},

mainnet: {

provider: () => new HDWalletProvider(

process.env.MNEMONIC,

`https://mainnet.infura.io/v3/${process.env.INFURA\_PROJECT\_ID}`

),

network\_id: 1,

gas: 5500000,

gasPrice: 20000000000, // 20 gwei

confirmations: 2,

timeoutBlocks: 200

}

},

compilers: {

solc: {

version: "0.8.19",

settings: {

optimizer: {

enabled: true,

runs: 200

}

}

}

},

plugins: ['truffle-plugin-verify']

};

***3. Smart Contract Development***

3.1 Create Contract File

Create `contracts/WINNGOOToken.sol`:

```solidity

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.19;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

import "@openzeppelin/contracts/access/Ownable.sol";

contract WINNGOOToken is ERC20, Ownable {

// Contract code as provided in previous documentation

}

```

3.2 Create Migration Script

Create `migrations/2\_deploy\_token.js`:

```javascript

const WINNGOOToken = artifacts.require("WINNGOOToken");

module.exports = function (deployer) {

const silverThreshold = 100; // First 100 users

const bronzeThreshold = 300; // Next 200 users

deployer.deploy(WINNGOOToken, silverThreshold, bronzeThreshold);

};

```

***4. Testing***

4.1 Create Test File

Create `test/WINNGOOToken.test.js`:

```javascript

const WINNGOOToken = artifacts.require("WINNGOOToken");

const { expectRevert, expectEvent } = require('@openzeppelin/test-helpers');

const { expect } = require('chai');

contract("WINNGOOToken", accounts => {

let token;

const [owner, user1, user2] = accounts;

beforeEach(async () => {

token = await WINNGOOToken.new(100, 300);

});

it("should register new user with correct tier", async () => {

const result = await token.register(constants.ZERO\_ADDRESS, { from: user1 });

expectEvent(result, 'UserRegistered', {

user: user1,

referrer: constants.ZERO\_ADDRESS

});

const tier = await token.getUserTier(user1);

expect(tier.toString()).to.equal('1'); // SILVER tier

});

});

4.2 Run Tests

# Run all tests

truffle test

# Run specific test file

truffle test ./test/WINNGOOToken.test.js

***5. Deployment (Truffle)***

5.1 Compile Contracts

truffle compile

5.2 Deploy to Local Network

# Start local blockchain

truffle develop

# Deploy in truffle console

migrate

5.3 Deploy to Testnet

# Deploy to Goerli testnet

truffle migrate --network goerli

# Deploy to BSC testnet

truffle migrate --network bsc\_testnet

5.4 Deploy to Mainnet

# Deploy to Ethereum mainnet

truffle migrate --network mainnet

***6. Deployment (Hardhat)***

6.1 Setup Hardhat Project

# Install Hardhat

npm install --save-dev hardhat

# Initialize Hardhat project

npx hardhat

# Install plugins

npm install --save-dev @nomiclabs/hardhat-ethers @nomiclabs/hardhat-waffle ethereum-waffle chai

6.2 Configure Hardhat

Create `hardhat.config.js`:

```javascript

require("@nomiclabs/hardhat-waffle");

require("@nomiclabs/hardhat-etherscan");

require('dotenv').config();

module.exports = {

solidity: "0.8.19",

networks: {

goerli: {

url: `https://goerli.infura.io/v3/${process.env.INFURA\_PROJECT\_ID}`,

accounts: {

mnemonic: process.env.MNEMONIC

}

},

mainnet: {

url: `https://mainnet.infura.io/v3/${process.env.INFURA\_PROJECT\_ID}`,

accounts: {

mnemonic: process.env.MNEMONIC

}

}

},

etherscan: {

apiKey: process.env.ETHERSCAN\_API\_KEY

}

};

6.3 Deploy using Hardhat

# Compile contracts

npx hardhat compile

# Deploy to network

npx hardhat run scripts/deploy.js --network goerli

***7. Contract Verification***

7.1 Verify using Truffle

# Install truffle-plugin-verify

npm install -D truffle-plugin-verify

# Verify on Etherscan

truffle run verify WINNGOOToken --network goerli

# Verify on BSCScan

truffle run verify WINNGOOToken --network bsc\_testnet

7.2 Verify using Hardhat

# Verify on Etherscan

npx hardhat verify --network goerli DEPLOYED\_CONTRACT\_ADDRESS "constructor\_arg\_1" "constructor\_arg\_2"

7.3 Manual Verification

1. Go to network explorer (Etherscan/BSCScan)

2. Find your contract address

3. Click "Verify and Publish"

4. Select compiler version and license

5. Upload source code

6. Provide constructor arguments

7. Submit for verification

***8. Frontend Development***

8.1 Create React Project

npx create-react-app winngoo-frontend

cd winngoo-frontend

8.2 Install Dependencies

npm install ethers@5.7.2

npm install @shadcn/ui

8.3 Contract Integration

# Copy contract ABI

cp ../build/contracts/WINNGOOToken.json src/contracts/

***9. Troubleshooting Guide***

9.1 Common Deployment Issues

1. \*\*Insufficient Funds\*\*

# Check account balance

truffle exec scripts/check-balance.js --network goerli

2. \*\*Nonce Issues\*\*

# Reset account nonce

await web3.eth.getTransactionCount(accounts[0])

3. \*\*Gas Price Issues\*\*

- Increase gas price in configuration

- Use gas price oracle

9.2 Verification Issues

1. \*\*Compiler Mismatch\*\*

- Ensure exact compiler version matches

- Check optimization settings

2. \*\*Constructor Arguments\*\*

- Encode constructor arguments correctly

- Use contract verification helper tools

9.3 Network-Specific Issues

1. \*\*Ethereum Networks\*\*

- Check gas prices

- Ensure sufficient ETH for gas

- Verify network congestion

2. \*\*BSC Networks\*\*

- Check BNB balance

- Verify RPC endpoint status

***## Important Notes***

1. \*\*Security\*\*

- Never commit `.env` file

- Use hardware wallet for mainnet

- Audit contract before mainnet deployment

2. \*\*Testing\*\*

- Test on local network first

- Then test on testnet

- Monitor initial transactions

3. \*\*Maintenance\*\*

- Keep dependencies updated

- Monitor contract events

- Implement upgrade strategy

## Pre-deployment Checklist

1. \*\*Contract\*\*

- [ ] All tests passing

- [ ] Gas optimization completed

- [ ] Security audit done

- [ ] Constructor arguments prepared

2. \*\*Environment\*\*

- [ ] Network selected

- [ ] Gas price checked

- [ ] Sufficient funds available

3. \*\*Documentation\*\*

- [ ] Contract addresses documented

- [ ] Deployment steps recorded

- [ ] ABI saved

- [ ] Function documentation ready