



School: Campus:
Academic Year: Subject Name: Subject Code:
Centurion
UNIVERSITY
Shaping Lives... Empowering Communities...

Semester: Program: Branch: Specialization:

Date:

Applied and Action Learning (Learning by Doing and Discovery)

Name of the Experiment : Token Launch – Deploying a Token Locally

Coding Phase : Pseudo Code/Flow Chart/Algorithm

- Open Remix IDE.
- Import the ERC20 contract from OpenZeppelin.
- Create a constructor that sets the token name, symbol, and mints initial supply.
- Compile the contract with Solidity ^0.8.20.
- Deploy to local blockchain using account 1.
- Use functions `name()`, `symbol()`, and `totalSupply()` to verify token.

Apparatus/Software Used:

- OS: Windows or others.
- Remix IDE.
- Wallet: MetaMask.
- Library: OpenZeppelin ERC20

Testing Phase:

- Call `name()` → returns "**SwadToken**"
- Call `symbol()` → returns "**SWA**"
- Call `totalSupply()` → returns **100000000**
- Call `balanceOf(owner)` → shows total supply in deployer's account.

Implementation Phase: Final Output (no error)

Step 1: Open Remix IDE.

- Open Browser/Brave.
- Search Remix IDE.

Step 2: Write Smart Contract.

- Create a new file in inside of contract using .sol.
- Write code:

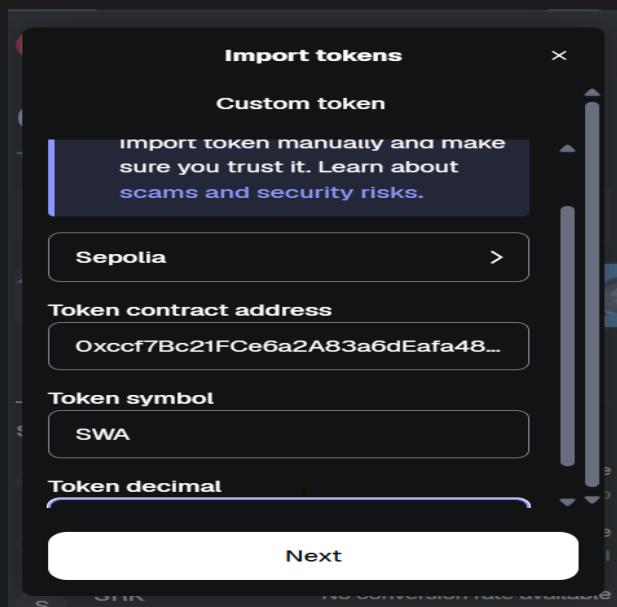
```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity ^0.8.20;
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5
6 contract SwadToken is ERC20{
7     constructor(string memory name, string memory symbol) ERC20(name, symbol){
8         _mint(msg.sender, 1000000*10 **decimals());
9     }
10 }
11 }
```

Step 3: Code Compile and Deploy .

- Click the Solidity compiler and compile this file.
- Click Deploy & run transaction Option .
- Deploy the write of some string name and symbol.

Step 4: Add Token to MetaMask.

- In MetaMask, click "Import Tokens".
- Enter your deployed contract address.
- Your token should appear in the wallet.



Step 5: Check Token Details.

- name() → should return "SwadToken"
- symbol() → should return "SWA"
- totalSupply() → should return 10000000.

The screenshot shows the Etherscan interface for the token 'swadToken (SWA)'. The 'Overview' section displays the maximum total supply of 1,000,000 SWA, with 2 holders and 3 total transfers. The 'Market' section shows the on-chain market cap and circulating supply market cap. The 'Transfers' tab is selected, showing a list of 3 transactions. The first transaction is a 'Provide Liquidity' event from 0xa4a6c6f724... to 0x01f374BF... with an amount of 0.0000000000000001. The second and third transactions are 'Transfer' events from 0xd18fb5b380... to 0x01f374BF... with amounts of 500 and 500 respectively. There are tabs for 'Holders' and 'Contract' at the bottom.

Step 6: Transfer Token:

- Use transfer(receiver_address, amount) to send tokens to another account.
- Then check balanceOf(receiver_address) to confirm transfer.

The screenshot shows a 'Send' screen for a token. The 'From' section shows 'Account 1' with address 0x754eF...dE931. Below it is a dropdown menu showing a 'SWA' entry with a balance of 0 SWA. The 'To' section shows an address 0x0A3DC..EAC2E with a balance of 0 SWA. At the bottom are 'Cancel' and 'Continue' buttons.

Observations

- The OpenZeppelin library makes ERC20 token creation quick and secure.
- Local deployment requires no real ETH.
- Token functions behave exactly the same as on testnet/mainnet.

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	50		

Signature of the Student:

Name :

Regn. No. :

Page No.....

Signature of the Faculty:

* As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

