



Centurion
UNIVERSITY
*Shaping Lives...
Empowering Communities...*

School: Campus:

Academic Year: Subject Name: Subject Code:

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:

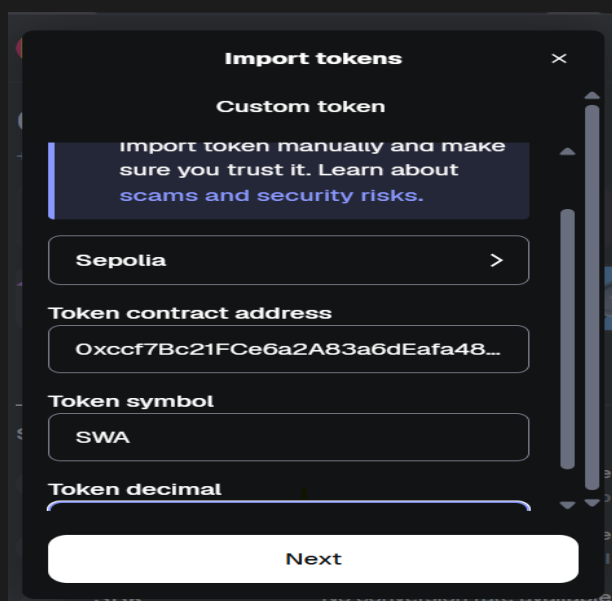
```
1  // SPDX-License-Identifier: GPL-3.0
2
3  pragma solidity ^0.8.20;
4  import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5
6  contract SwadToken is ERC20{
7      constructor(string memory name, string memory symbol) ERC20(name, symbol){
8          _mint(msg.sender,1000000*10 **decimals());
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 page is divided into several sections: Overview, Market, and Other Info. The Overview section displays the token's contract address, total supply, and other basic information. The Market section shows the token's market capitalization and supply. The Other Info section provides additional details about the token contract. Below these sections, there is a table of transactions, showing the transaction hash, method, block, age, from, to, and amount.

Transaction Hash	Method	Block	Age	From	To	Amount
0xa4a6c6f7240...	Provide Liquid...	8937729	2 days ago	0x754eFC74...DE73dE931	0x01f374BF...D0087e1fc	0.000000000000000001
0x6d18fb5b380...	Transfer	8937725	2 days ago	0x754eFC74...DE73dE931	0x01f374BF...D0087e1fc	500

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 mobile application interface for sending tokens. The screen is titled 'Send' and has a back arrow in the top left corner. It features a 'From' section with a dropdown menu showing 'Account 1' and the address '0x754eF...dE931'. Below this is a section for the token, showing 'SWA' with a balance of '0'. The 'To' section has a dropdown menu showing the recipient address '0x0A3DC...EAC2E'. At the bottom, there are two buttons: 'Cancel' and 'Continue'.

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. :

Signature of the Faculty:

Page No.....

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

