



Centurion
UNIVERSITY
Mysore City
Karnataka, India

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

ALGORITHM:

1. Start
2. Open your remix IDE and create a new file and name it as per as your choice.
3. Write the solidity code to for creating a token
4. Now compile that file
5. After compiling go to deploy and transaction, select injected provider metamask, then write your token name and symbol in deploy and deploy the contract.
6. Confirm the transaction.
7. Now go to your wallet to import token.
8. End

* Software used

1. Remix IDE
2. Metamask
3. OpenZeppelin Contracts
4. Etherscan

Page No.....

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

* Testing Phase: Compilation of Code (error detection)

First open your remix IDE and create a new file in contracts named as Token.sol. Then write the contract for deploying the token

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
contract BlockToken is ERC20 {
    constructor(string memory name, string memory symbol) ERC20(name, symbol) {
        _mint(msg.sender, 1000000 * 10 ** decimals());
    }
}
```

This code creates an ERC-20 token with customizable name and symbol. It also mints 1 million tokens to the deployer's wallet.

Now go to solidity compiler select compiler version 0.8.30 and then click on “Compile Token.sol”.

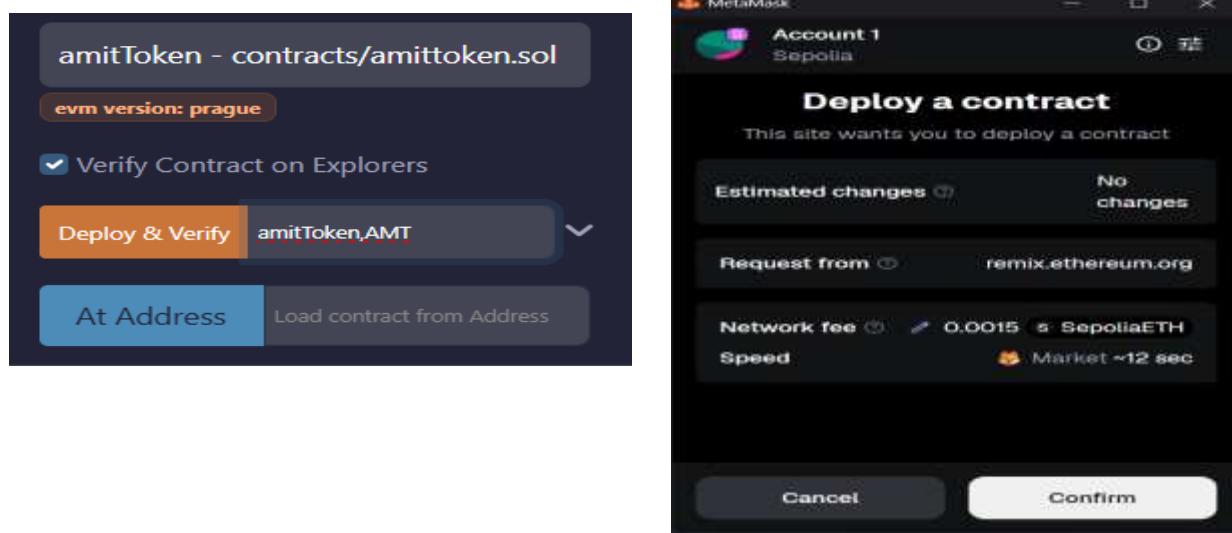


Now go to deploy and transaction select “injected provider metamask” as your environment. Approve the connection



* Testing Phase: Compilation of Code (error detection)

Write your token name and symbol in the constructor parameter (e.g. BlockToken, BLK) and then deploy the contract. It will open a pop-up to deploy the contract click on confirm.



After deploying copy the address and paste it in etherscan to check transaction details of your token.

Transaction Hash	Method	Block	Age	From	To	Amount	Gas Fee
0xa2a5695794...	Approve	9507801	2 days ago	0x0AFE4DFD...3cAECCb81	0x8D63920C...	0 ETH	0.0001402
0xb06e8983f67c...	Transfer	9507592	2 days ago	0x0AFE4DFD...3cAECCb81	0x8D63920C...	0 ETH	0.00003728

* Testing Phase: Compilation of Code (error detection)

Click on the token symbol written on transaction details page it will open to your created token where you will get your token contract address, copy that address.

Sent AMT	
Status	Confirmed
From	 0xDAFe4...C...
To	 Account 1
Transaction	
Nonce	7
Amount	-100 AMT
Gas Limit (Units)	45427
Gas Used (Units)	27489
Base fee (GWEI)	0.000600261
Priority fee (GWEI)	1.5
Total gas fee	0.000041 SepoliaETH
Max fee per gas	0.000000002 SepoliaETH
Total	0.00004128 SepoliaETH

Now import your token in your metamask wallet. Open your wallet go to tokens click on the three dots and then click on import tokens. Select the network sepolia and then paste your token contract address and click next.

Tokens	DeFi	NFTs	Activity
Sepolia ▾			≡ :
 SepoliaETH	No conversion rate available 0.09507 SepoliaETH		
 AMT	No conversion rate available 1,000,000 AMT		

* Implementation Phase: Final Output (no error)

Applied and Action Learning

Your token is successfully launched in your wallet.

The screenshot shows the MetaMask wallet interface. At the top, it displays "Account 1" with the address "0xDADe4...CCb81". Below this, the balance is shown as "0.0951 SepoliaETH" with a note "+\$0 (+0.00%) Portfolio". There are five buttons: Buy/Sell, Swap, Bridge, Send, and Receive. A "MetaMask Missions" sidebar is visible, encouraging users to complete missions for rewards. The main area shows a table of transaction details:

Transaction	From	To
Nonce		
Amount		-100 AMT
Gas Limit (Units)		45427
Gas Used (Units)		27489
Base fee (GWEI)		0.000600251
Priority fee (GWEI)		1.5
Total gas fee		0.000041 SepoliaETH
Max fee per gas		0.000000002 SepoliaETH
Total		0.00004125 SepoliaETH

* Observations

- Successfully deployed an ERC-20 compatible token smart contract on a local blockchain environment to simulate token creation.
- Verified token functionalities like total supply, balance transfer, and allowance management in a controlled local setup before mainnet/testnet deployment.

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.