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 Experiement : Token Launch – Deploying a Token Locally

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

ALGORITHM:

•**Start** by setting up the blockchain development environment using **Truffle** or **Hardhat** and connect it to a local blockchain like **Ganache**.
•**Create a new Solidity file** (e.g., MyToken.sol) and define the ERC-20 token contract with parameters such as token name, symbol, and total supply.
•**Compile** the smart contract using the Truffle or Hardhat compiler to check for syntax or logical errors.
•**Deploy** the compiled token contract on the local blockchain network using deployment scripts or the Truffle migration command.
•**Verify deployment** by checking the generated token address and confirming token details in Ganache or Remix.
•**Test transactions** by transferring tokens between different accounts to ensure proper token functionality.
•**End** the process after successful deployment and validation of token behavior on the local blockchain.

* Software used

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

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

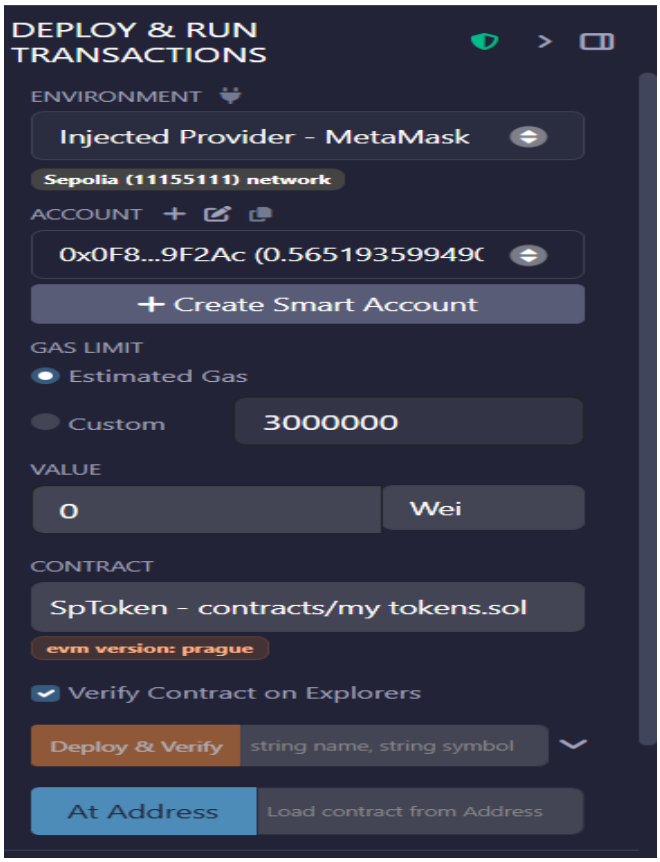
* Testing Phase: Compilation of Code (error detection)

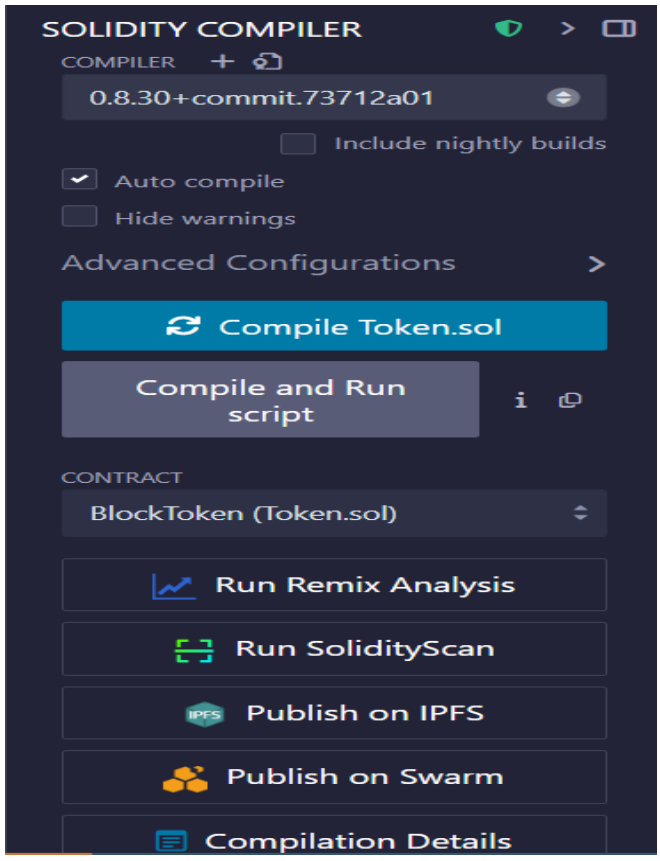
First open your remix IDE and create a new file in contracts named as Token.sol. Then write the contract for deploying the token

```
//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());
    }
}
```

1st we compile the token and then we deploy and transaction select "injected provider metamask" as your environment. Approve the connection

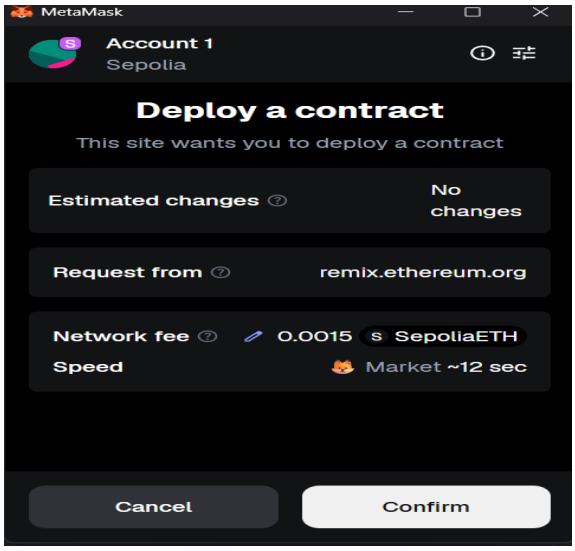




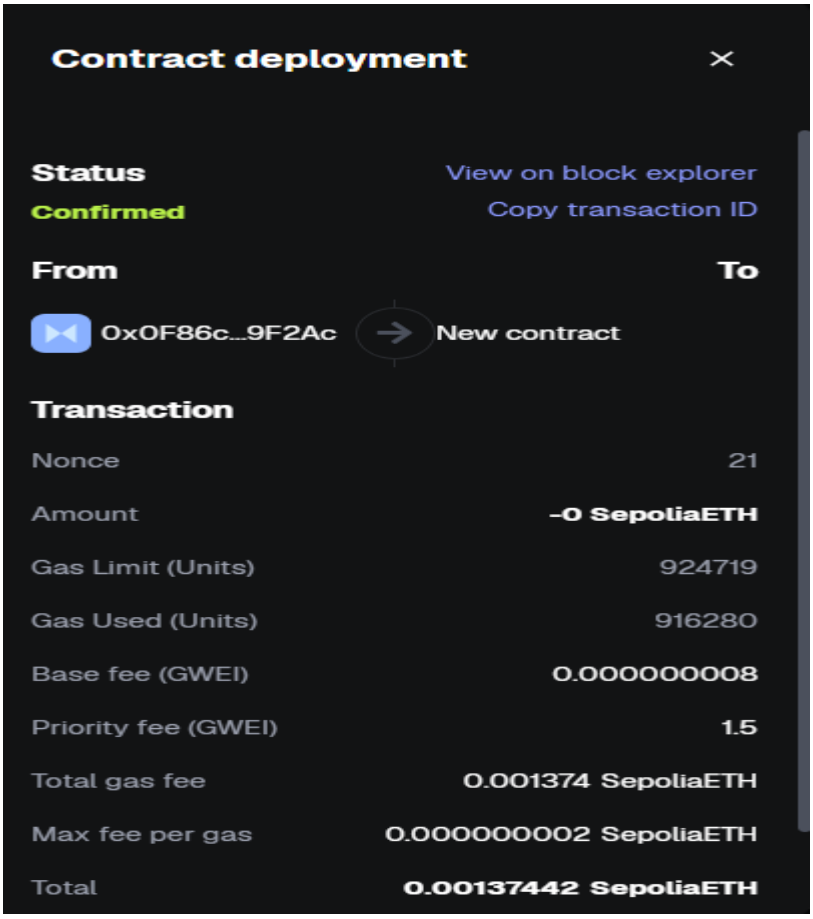
*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* 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.



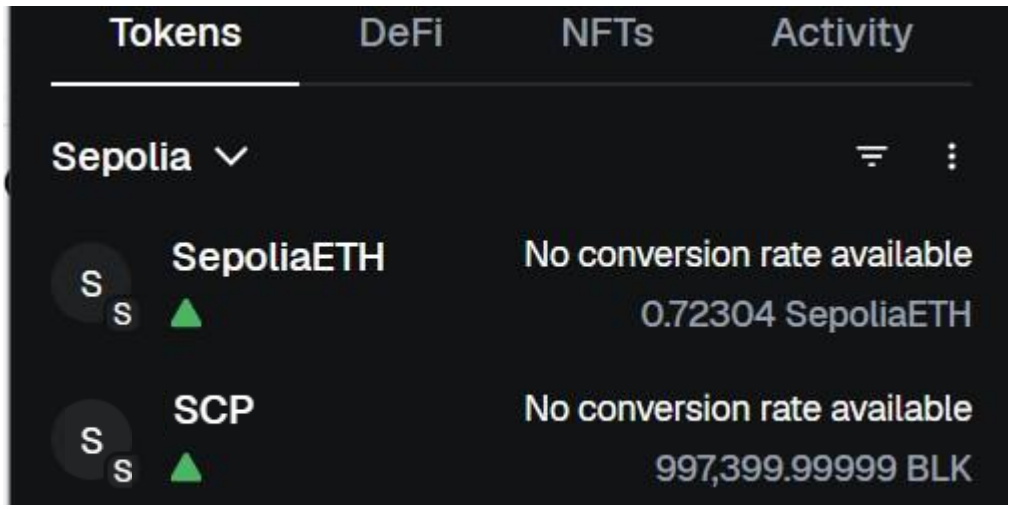
After deploying copy the address and paste it in etherscan to check transaction details of your token.



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

* Implementation Phase: Final Output (no error)

Your token is successfully launched in your wallet.



* Observations

•The ERC-20 token smart contract was successfully compiled without any errors.
•The token was deployed on the local blockchain (Ganache) and generated a unique contract address.
•Token parameters such as name, symbol, and total supply were correctly initialized.
•Test token transfers between different accounts were executed successfully.

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:

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