



## **PROJECT SCHOOL CERTIFICATE**

**Project Title :** CryptoSwap

**Project Description :** Building a blockchain mobile application for getting real time crypto prices and for exchanging cryptocurrencies.

**Section :** CSE - D

**Project Guide:**

Dr.Raja Sekaran

**Submitted by:**

D. Sahitya (57D)

(Signature)

## **Project Specification :**

### **a) Project Objective :**

The main objective of the project is to let users swap different type of tokens according to their current values. It also helps the users find the current prices of bitcoin and ethereum cryptocurrencies.

### **b) Developed solution with Tech stack :**

A mobile application developed using flutter, which allows users to connect with metamask, get current prices and swap tokens between two accounts. The backend, which is entirely about blockchain is integrated with flutter frontend using Infura API.

### **TECH STACK :**

- Frontend
  - Flutter
- Backend
  - Solidity, Blockchain
- Flutter dependencies :
  - url\_launcher , flutter\_web3 , walletconnect\_dart ,  
external\_app\_launcher , web3 , http

Metamask wallet is used to store user tokens.

c) System Workflow :

- Login Page : This is the first page of the app. It directs user to connect with metamask, swap tokens and get current prices of cryptocurrencies.
- Connect with Metamask : This launches an external application (metamask) using “walletconnect.org”, which helps us connect with our metamask account for future purposes.
- Swap Tokens : This page accepts user’s accounts and tokens addresses, the amount of gold and silver tokens to be exchanged and swaps the tokens accordingly. Users can also check their balance after swapping.
- Get current price : In this page, users can check the real time crypto prices of bitcoin and ethereum in USD. This is done with the help of Chainlink Data Feeds, which helps us connect our smart contracts to asset pricing data like ETH/USD feed.

d) Screenshots of the entire app :

GetCurrentPrice.sol :

```
contracts > gcp.sol
35 contract PriceConsumerV3 {
36     AggregatorV3Interface internal btcpricefeed;
37     AggregatorV3Interface internal ethpricefeed;
38
39     /**
40      * Network: Sepolia
41      * Aggregator: BTC/USD
42      * Address: 0x1b44F3514812d835EB18D80acB33d3fA3351Ee43
43      */
44     constructor() {
45         btcpricefeed=AggregatorV3Interface(0xA39434A63A52E749F02807ae27335515BA4b07F7);
46         ethpricefeed=AggregatorV3Interface(0xD4a33860578De61D8AbDc88FDb98FD742fA7028e);
47     }
48
49     /**
50      * Returns the latest price.
51      */
52     function getbtclatestPrice() public view returns (int) {
53         // prettier-ignore
54         (
55             /* uint80 roundID */,
56             int price,
57             /*uint startedAt*/,
58             /*uint timeStamp*/,
59         );
60     }
61 }
```

SwapTokens.sol :

```
contracts > swapTokens.sol
21 function swap(uint amount1,uint amount2) public {
22     require(msg.sender==owner1 || msg.sender==owner2,"Not authorized");
23     require(
24         token1.allowance(owner1,address(this)) >= amount1,
25         "Token 1 allowance too low"
26     );
27     require(
28         token2.allowance(owner2,address(this)) >= amount2,
29         "Token 2 allowance too low"
30     );
31
32     //transfer tokens
33     //token1 , owner1,amount1 ->owner2
34     safeTransferFrom(token1,owner1,owner2,amount1);
35     //token2 , owner2,amount2 ->owner1
36     safeTransferFrom(token2,owner2,owner1,amount2);
37 }
38
39 function safeTransferFrom(
40     IERC20 token,
41     address sender,
42     address recipient,
43     uint amount
44 ) private{
45     bool sent = token.transferFrom(sender,recipient,amount);
46     require(sent,"Token Transfer Failed");
47 }
```

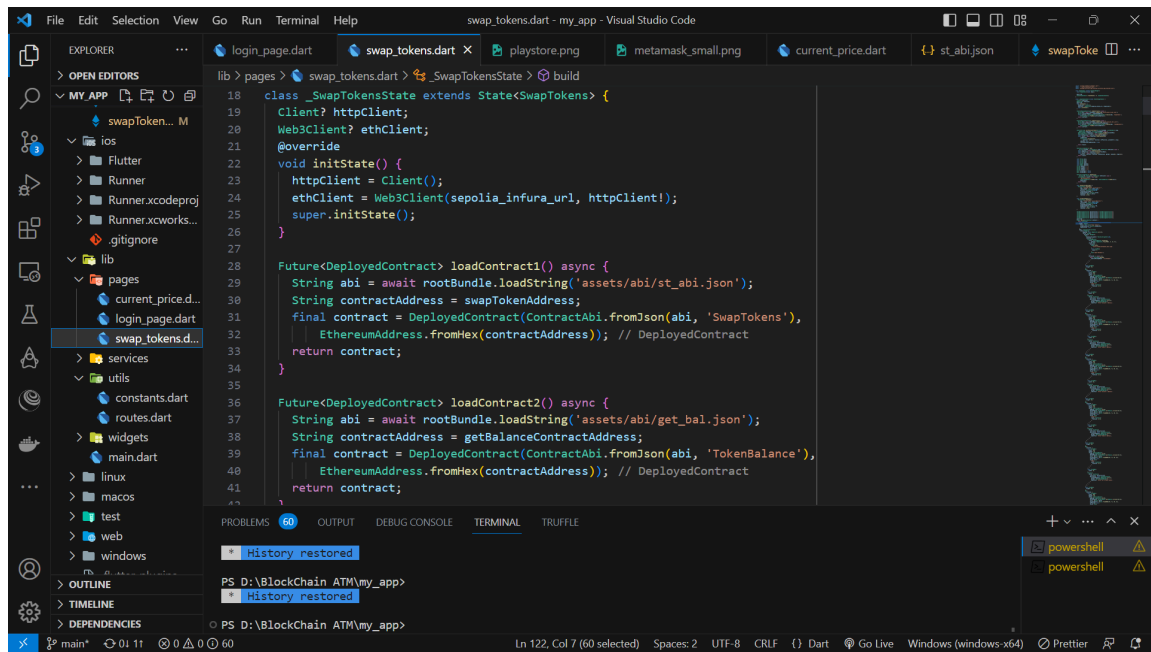
## Login\_page.dart :

```
lib > pages > login_page.dart > _LoginPageState > build
1 import 'package:flutter/material.dart';
2 import 'package:my_app/pages/swap_tokens.dart';
3 import 'package:walletconnect_dart/walletconnect_dart.dart';
4 import 'package:url_launcher/url_launcher_string.dart';
5 import 'package:my_app/pages/current_price.dart';
6
7 class LoginPage extends StatefulWidget {
8   const LoginPage({Key? key}) : super(key: key);
9
10  @override
11  State<LoginPage> createState() => _LoginPageState();
12
13  loginUsingMetamask(BuildContext context) {}
14 }
15
16 class _LoginPageState extends State<LoginPage> {
17   var connector = WalletConnect(
18     bridge: 'https://bridge.walletconnect.org',
19     clientMeta: const PeerMeta(
20       name: 'My App',
21       description: 'An app for converting pictures to NFT',
22       url: 'https://walletconnect.org',
23       icons: [
24         'https://files.gitbook.com/v0/b/gitbook-legacy-files/o/spaces%2F-L3JJeCjLrr530cT1M17X2Favator.png?alt=med
25       ],
26     ),
27   );
28 }
```

## Current\_price.dart :

```
lib > pages > current_price.dart > _GetCurrentPriceState > build
16 class _GetCurrentPriceState extends State<GetCurrentPrice> {
17   Client? httpClient;
18   Web3Client? ethClient;
19   TextEditingController controller = TextEditingController();
20
21   @override
22   void initState() {
23     httpClient = Client();
24     ethClient = Web3Client(infura_url, httpClient!);
25     super.initState();
26   }
27
28   Future<DeployedContract> loadContract() async {
29     String abi = await rootBundle.loadString('assets/abi/cp_abi.json');
30     String contractAddress = currentPriceAddress;
31     final contract = DeployedContract(
32       ContractAbi.fromJson(abi, 'GetCurrentPrice'),
33       EthereumAddress.fromHex(contractAddress)); // DeployedContract
34     return contract;
35   }
36
37   Future<List<dynamic>> ask(
38     String functionName, List<dynamic> args, Web3Client ethClient) async {
39     final contract = await loadContract();
40     final ethFunction = contract.function(functionName);
41     final result = await ethClient
42   }
```

## Swap\_tokens.dart :



```
lib > pages > swap_tokens.dart > _SwapTokensState > build
18 class _SwapTokensState extends State<SwapTokens> {
19   Client? httpClient;
20   Web3Client? ethClient;
21   @override
22   void initState() {
23     httpClient = Client();
24     ethClient = Web3Client(sepolia_infura_url, httpClient!);
25     super.initState();
26   }
27
28   Future<DeployedContract> loadContract1() async {
29     String abi = await rootBundle.loadString('assets/abi/st_abi.json');
30     String contractAddress = swapTokenAddress;
31     final contract = DeployedContract(ContractAbi.fromJson(abi, 'SwapTokens'),
32       EthereumAddress.fromHex(contractAddress)); // DeployedContract
33     return contract;
34   }
35
36   Future<DeployedContract> loadContract2() async {
37     String abi = await rootBundle.loadString('assets/abi/get_bal.json');
38     String contractAddress = getBalanceContractAddress;
39     final contract = DeployedContract(ContractAbi.fromJson(abi, 'TokenBalance'),
40       EthereumAddress.fromHex(contractAddress)); // DeployedContract
41     return contract;
42   }
43 }
```

PROBLEMS 60 OUTPUT DEBUG CONSOLE TERMINAL TRUFFLE

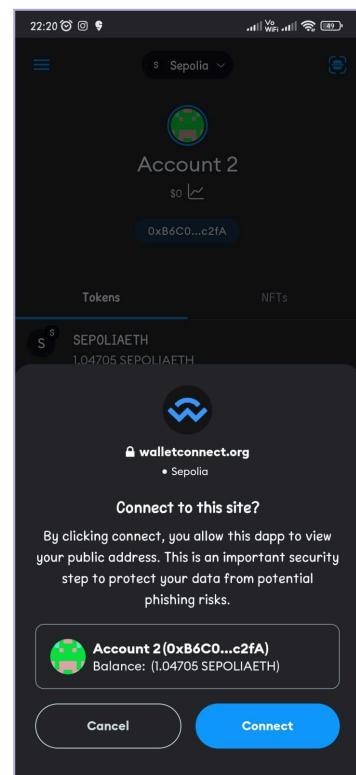
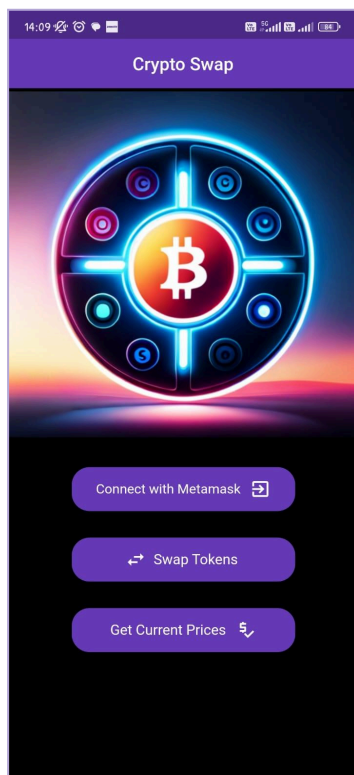
History restored

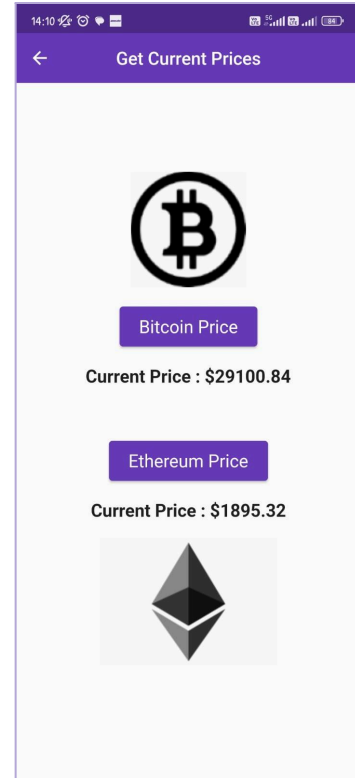
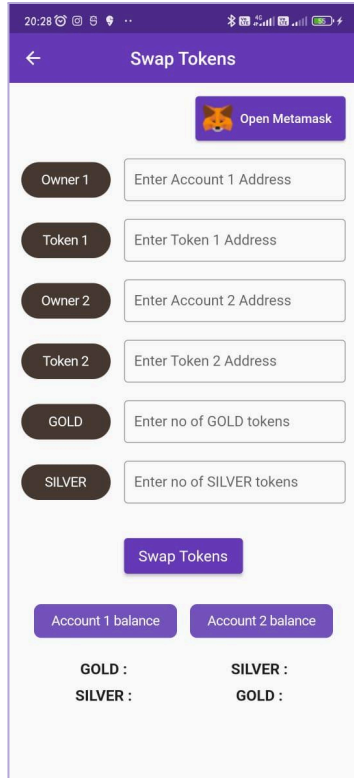
PS D:\Blockchain ATM\my\_app

History restored

PS D:\Blockchain ATM\my\_app

## Running app in mobile :





#### e) Future Enhancement Plan :

- In get current price page, we can add an option to convert the value of a cryptocurrency to another, for example, bitcoin to ethereum and vice versa.
- We can include a feature which automatically detects all the tokens (token addresses) present in the selected account, which allows users to select the required token address from the dropdown menu option.
- We can add a feature to validate the account and token addresses entered by user, manually.

**Individual Github link :**

Project link - <https://github.com/Sahitya19-code/BlockChainATM.git>

**Demo Video :**

[https://drive.google.com/file/d/1xSa\\_RYEC1gXOvajVU9BhOG4tCliCWoCa/view?usp=share\\_link](https://drive.google.com/file/d/1xSa_RYEC1gXOvajVU9BhOG4tCliCWoCa/view?usp=share_link)