Scope and Design Documentation for CardanoVSC

Udai Solanki
Gaurav Raj
Aditya Solanki

December 2024

Contents

1 Introduction

CardanoVSC is a Visual Studio(VS) Code Integrated Development Environment(IDE) extension designed to enhance the development experience for Plutus and Haskell programmers by offering:

- Syntax Highlighting.
- Code Completion.
- Cardano Blockchain Explorer Integration.
- Smart Contract Deployment
- Wallet Management
- Cardano node connection
- Debugger tool

2 Objective

- Syntax Highlighting and Code Completion: Improve code readability and provide coding assistance.
- Cardano Blockchain Explorer API Integration: Enable interaction with the Cardano blockchain directly from VS Code.
- Smart Contract Deployment/Connecting Node: Facilitate the deployment of smart contracts and connection to a Cardano node.
- Wallet Management: Provide tools for creating and managing wallets.
- Debugging Tools: Enhance the debugging capabilities for Plutus smart contracts.

3 Scope

3.1 Syntax highlighting and code completion

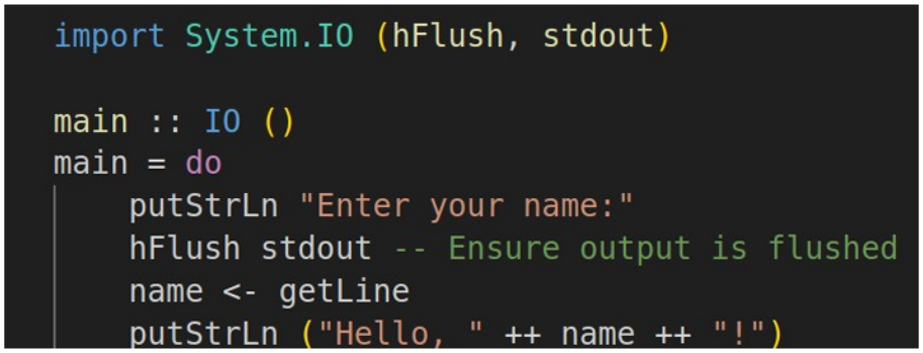
The extension will be designed to provide comprehensive syntax highlighting for Plutus programming language. By delivering precise and visually appealing syntax highlighting, the extension aims to enhance productivity and make coding more enjoyable. This includes all function, operators and variables being highlighted with different colors, as is the norm for syntax highlighting in Visual Studio Code for other languages. We will also be providing code completion suggestions. If a user wants they can use the suggestion to auto complete their code. Auto completion will also add closing brackets to any opening brackets in Plutus.Our objective here is to provide clear and distinct visual cues for different elements of Plutus code and to assist developers by suggesting possible completions for partially typed code in Visual Studio Code.

Syntax Highlighting for:

- Keywords
- Data types
- Functions
- Variables
- Comments

Code Completion for:

- Function names and parameters.
- Suggest Variable names.
- Suggest Data types.
- Suggest Snippets for common patterns.
- Auto complete brackets.



```haskell
import System.IO (hFlush, stdout)

main :: IO ()
main = do
    putStrLn "Enter your name:"
    hFlush stdout -- Ensure output is flushed
    name <- getLine
    putStrLn ("Hello, " ++ name ++ "!")
```

Figure 1: Syntax Highlighting(e.g., keywords, operators, comments).

This makes coding on Visual Studio Code a more desirable experience for Plutus developers.

3.2 Cardano Blockchain Explorer API Integration

- Query blockchain data from one of the blockchain explorers. We will use CardanoScan for this due to it's popularity.
- Connect with the Cardano Blockchain Explorer API to display blockchain data.

3.3 Smart Contract Deployment/Connecting Node

- Allow users to deploy Plutus smart contracts directly from VS Code.
- Enable connecting to a Cardano node from inside of Visual Studio Code. For this step it is required to have a Linux Operating System.
- Deployment of smart contract is dependent on the node connection and Cardano CLI.

3.4 Wallet Management

- Provide users with the ability to create and manage Cardano wallets within the extension.
- Include features for Wallet creation,Restore via mnemonic phrases,Viewing balances and transaction history.

3.5 Debugging Tools

- Create debugging tools that allow developers to step through smart contract execution and transaction processing.

4 Technical Specifications for CardanoVSC Extension.

4.1 Setup Environment:

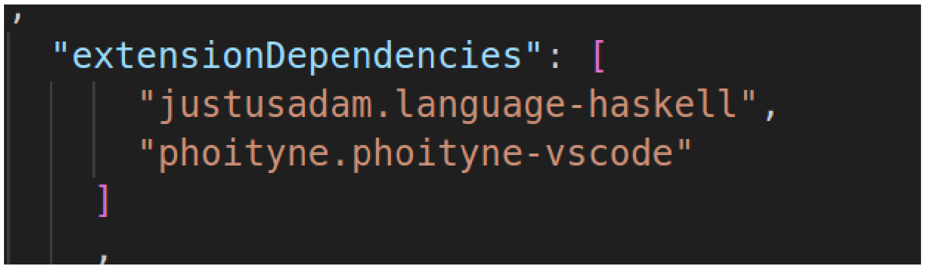
- **Step 1** First, use VS Code Extension Generator to scaffold a JavaScript project ready for development.
- **Step 2** Run the following

  ```
  npx --package yo --package generator-code -- yo
  code
  ```

  to initialize building the extension. This will create the necessary project structure.

4.2 Develop Syntax Highlighting and Code Completion

- For Syntax Highlighting we add dependencies to the existing package.json file to include the Haskell language dependency provided to us by Justusadam.
- Test the modified grammar to ensure it can be used for Cardano specific functionalities.
- Themes: Ensure compatibility with popular VS Code themes.



```
,
  "extensionDependencies": [
      "justusadam.language-haskell",
      "phoityne.phoityne-vscode"
    ]
    ,
```

Figure 2: Syntax Highlighting

Code Completion

- Language Server Protocol (LSP): Utilize HLS to provide code completion features.
- Custom Snippets: To create snippets for common Plutus constructs and patterns we use Completion_API() Function to for suggesting code snippets.
- Create boilerplate templates for Plutus Smart contract development that can be accessed using preset command for getting started.

4.3 Implement Cardano Blockchain Explorer API Integration

- We use the following list of API's to connect to a Cardano Blockchain Explorer and get the required information. It requires a free API key from CardanoScan to connect to it. Each user should provide their own Free API key in the URL.

  Get Block Details:
  Query Parameter: Block hash(string), blockHeight(integer), absoluteSlot(integer), epoch(integer), slot(integer).

```
curl \
 -X GET https://api.cardanoscan.io/{User_api_key}/
   v1/block
```

Listing 1: Bash Example: Get Block Details

Get latest block details:

```
curl \
 -X GET https://api.cardanoscan.io/{User_api_key}/
   v1/block/latest
```

Listing 2: Bash Example:Get latest block details

Get address balance:
Query parameters: Address(integer).

```
    curl \
    -X GET https://api.cardanoscan.io/{
   User_api_key}/v1/address/balance?address=string
```

Listing 3: Bash example: Get address balance.

Get pool details:
Query parameters: PoolID(string).

```
    curl \
 -X GET https://api.cardanoscan.io/{User_api_key}/
   v1/pool?poolId=string
```

Listing 4: Bash Example: Get pool id.

Get pool stats:
Query parameters: poolId

```
    curl \
 -X GET https://api.cardanoscan.io/{User_api_key}/
   v1/pool/stats?poolId=string
```

Listing 5: Bash example: Get Pool Stats

Get pools list:
Query parameters: pageNo(integer), search(string), retiredPools(boolean), sortBy(string), order(string), limit (integer).

```
    curl \
 -X GET https://api.cardanoscan.io/{User_api_key}/
   v1/pool/list?pageNo=1
```

Listing 6: Bash example: Get Pool List.

Get pools which are set to expire:
Query parameters: pageNo(integer), limit(integer).

```
        curl \
-X GET https://api.cardanoscan.io/{User_api_key}/
  v1/pool/list/expiring?pageNo=42

```

Listing 7: Bash example: Get pools which are set to expire.

Get expired pools:
Query parameters: pageNo(integer), limit(integer).

```
    curl \
-X GET https://api.cardanoscan.io/{User_api_key}/
  v1/pool/list/expired?pageNo=1
```

Listing 8: Bash example: Get expired pools.

Get asset details
Query parameters: assetId(string), fingerprint(string).

```
    curl \
-X GET https://api.cardanoscan.io/{User_api_key}/
  v1/asset
```

Listing 9: Bash example: Get asset details.

Get assets by policyId
Query parameters: policyId(string), pageNo(integer), limit integer.

```
    curl \
-X GET https://api.cardanoscan.io/{User_api_key}/
  v1/asset/list/byPolicyId?policyId=string&pageNo
  =42
```

Listing 10: Bash example: Get assets by policyId.

Get assets by address:
Query parameters: address(string), pageNo(integer), limit (integer).

```
    curl \
-X GET https://api.cardanoscan.io/{User_api_key}/
  v1/asset/list/byAddress?address=string&pageNo=2
```

Listing 11: Bash example: Get assets by address.

Get transaction details:
Query parameters: hash(string).

```
    curl \
-X GET https://api.cardanoscan.io/{User_api_key}/
  v1/transaction?hash=string
```

Listing 12: Bash example: Get transaction details.

Get transaction list by address:
Query parameters: address(string), pageNo(integer), limit (integer), order(string).

```
    curl \
-X GET https://api.cardanoscan.io/{User_api_key}/
   v1/transaction/list?address=string&pageNo=42
```

Listing 13: Bash example: Get transaction details.

Get stake key details:
Query parameters: rewardAddress(string).

```
    curl \
 -X GET https://api.cardanoscan.io/{User_api_key}/
   v1/rewardAccount?rewardAddress=string
```

Listing 14: Bash example: Get stake key details.

Get addresses associated with a stake key:
Query parameters: rewardAddress(string), pageNo(integer), limit(integer).

```
    curl \
 -X GET https://api.cardanoscan.io/{User_api_key}/
   v1/rewardAccount/addresses?rewardAddress=string
   &pageNo=42
```

Listing 15: Bash example: Get addresses associated with a stake key.

Get network details:

```
    curl \
 -X GET https://api.cardanoscan.io/{User_api_key}/
   v1/network/state
```

Listing 16: Bash example: Get network Details.

Get network protocol details:

```
    curl \
 -X GET https://api.cardanoscan.io/{User_api_key}/
   v1/network/protocolParams
```

Listing 17: Bash example: Get network protocol details.

Get CCHot details:

```
    curl \
 -X GET https://api.cardanoscan.io/{User_api_key}/
   v1/governance/ccHot?hotHex=string
```

Listing 18: Bash example: Get CCHOT details.

Get CCMember details:

```bash
    curl \
-X GET https://api.cardanoscan.io/{User_api_key}/
  v1/governance/ccMember?coldHex=string
```

Listing 19: Bash example: Get CCMember details.

Get Committee Information:

```bash
    curl \
-X GET https://api.cardanoscan.io/{User_api_key}/
  v1/governance/committee
```

Get dRep Information:

```bash
    curl \
-X GET https://api.cardanoscan.io/{User_api_key}/
  v1/governance/dRep?dRepId=string
```

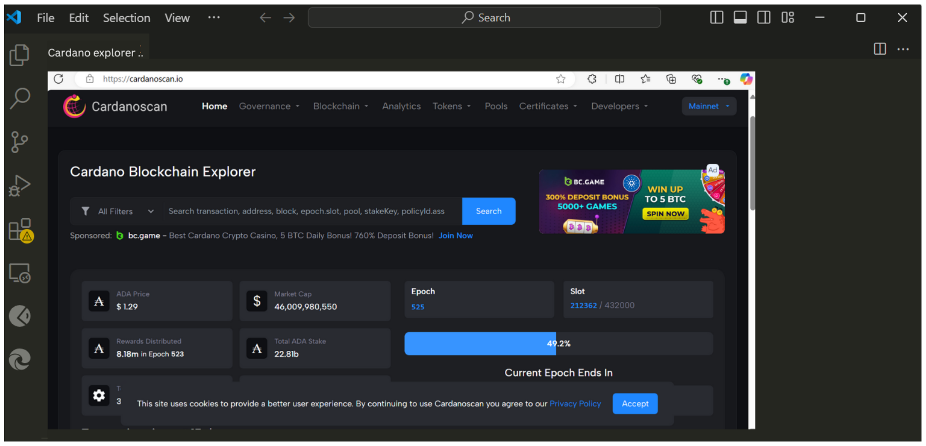
Listing 20: Bash example: Get dRep Information.

Get Governance Action:
Query parameters: actionId(string).

```bash
    curl \
-X GET https://api.cardanoscan.io/{User_api_key}/
  v1/governance/action?actionId=string
```

Listing 21: Bash example: Get Governance Action.

- Enable web view of Cardano Explorer Website(CardanoScan) Within VS Code.



Cardano Blockchain Explorer

Search transaction, address, block, epoch.slot, pool, stakeKey, policyId.ass

Sponsored: bc.game – Best Cardano Crypto Casino, 5 BTC Daily Bonus! 760% Deposit Bonus! Join Now

ADA Price $ 1.29

Market Cap 46,009,980,550

Epoch 525

Slot 212362 / 432000

Rewards Distributed 8.18m in Epoch 523

Total ADA Stake 22.81b

49.2%

Current Epoch Ends In

This site uses cookies to provide a better user experience. By continuing to use Cardanoscan you agree to our Privacy Policy

Figure 3: This is the mock up for Cardano explorer UI integrated in VS code.

4.4 Smart Contract Deployment from VS Code IDE

Connecting Node

- Allow users to execute Cardano CLI directly from within VSCode. Commands such as *Transaction list by address, stake key details, et, etc.*
- Prompt user input or use a for the username, password, IP address, and port number.
- Finally, display the response as a notification.

Smart contract deployment

- User creates a Haskell file and is provided with the boiler plate Plutus code.
- User can Click on a deploy button or execute cabal commands to build a Plutus file.
- Display a Success message and create the Plutus file for the user.
- If user node is connected to a node, then user can deploy the smart contract using Cardano CLI directly to blockchain using configured address and keys stored in VS code local storage state.
- Error handling will be done if the build fails due to cabal or configuration issue from the user side.
- Create a boiler plate code for Plutus in a Haskell file.

4.5 Wallet Management

- Click sidebar button or execute command for wallet management. This UI component will be used in the vs code sidebar.
  We will enable the on Chain connection using Lucid Cardano, BlockfrostAPI or Cardano-cli and a node connection. We will also add it in documentation on both, how a developer can make his wallet using Command or how to do it via the UI.

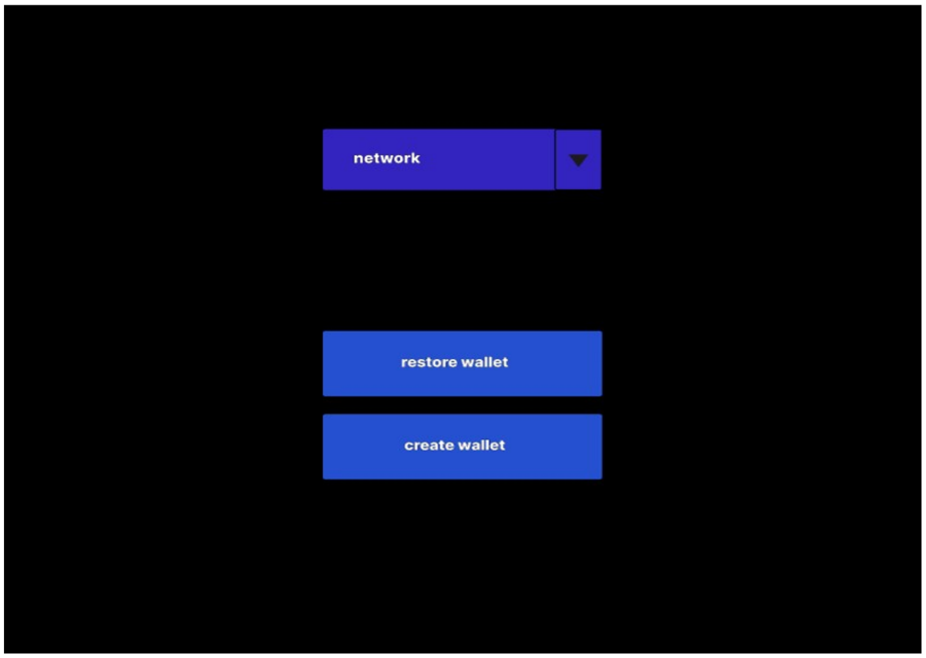


Figure 4: Wallet UI Mockup.

- Allow user to access different networks for wallet Preview, Preprod, Mainnet. This will be possible using Lucid cardano for the UI part. If a person wants to make changes using CLI they need to change their node network to the network the want(i.e. Preview, Preprod, Mainnet). We will add details in documentation so the user is not confused.

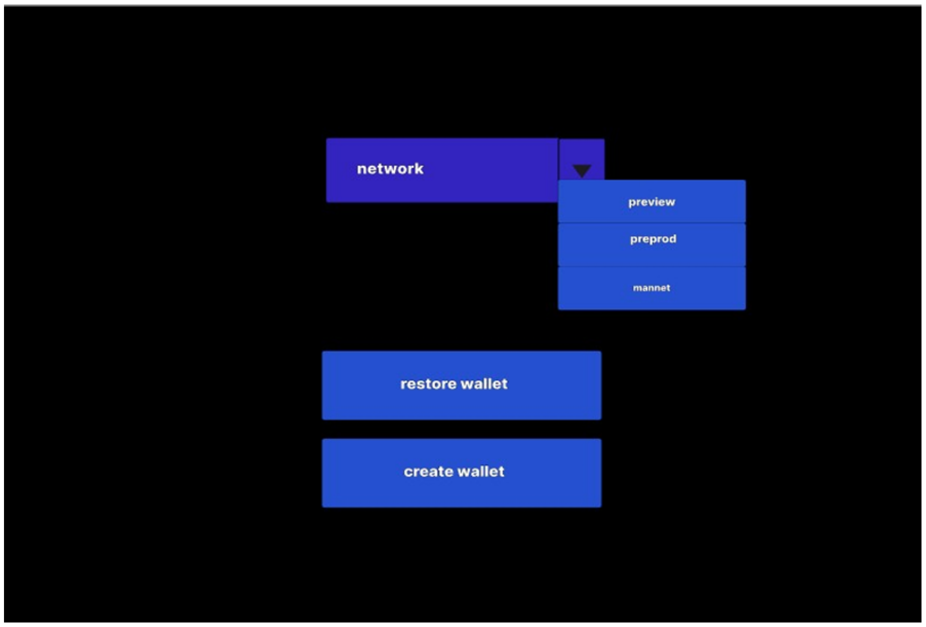


Figure 5: UI Mockup for network selection.

- Create and show the seed phrase to the user the user can choose how to store securely on their own.

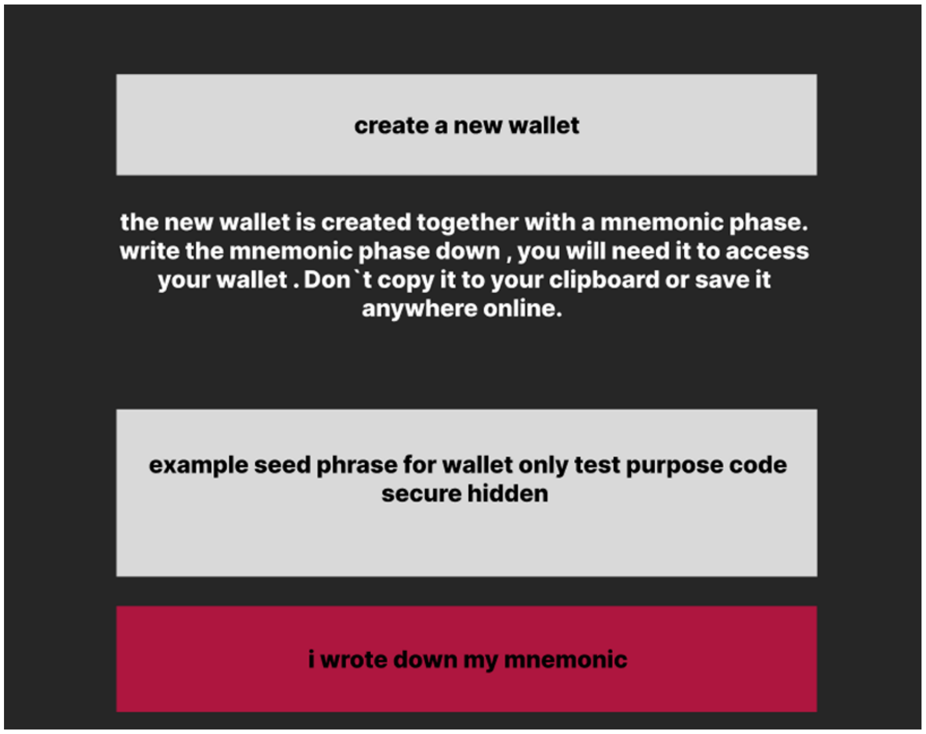

create a new wallet
the new wallet is created together with a mnemonic phase. write the mnemonic phase down , you will need it to access your wallet . Don`t copy it to your clipboard or save it anywhere online.
example seed phrase for wallet only test purpose code secure hidden
i wrote down my mnemonic

Figure 6: UI Mockup for Wallet seed phrase generation for GUI wallet.

- Alternatively after selecting the Cardano network the user can select to restore wallet by entering their seed phrase.

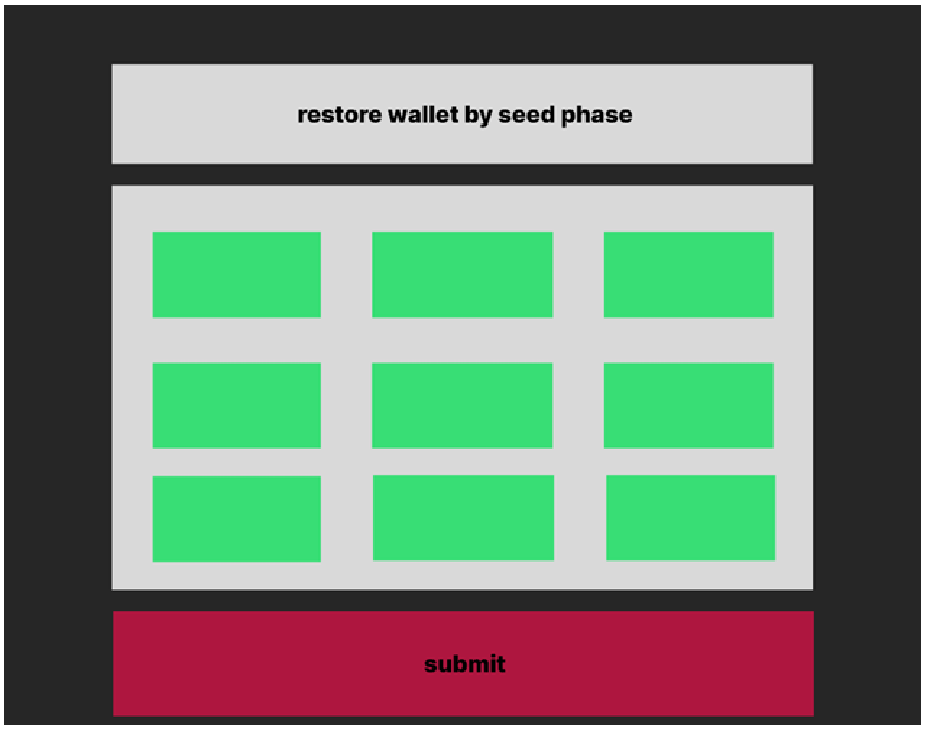
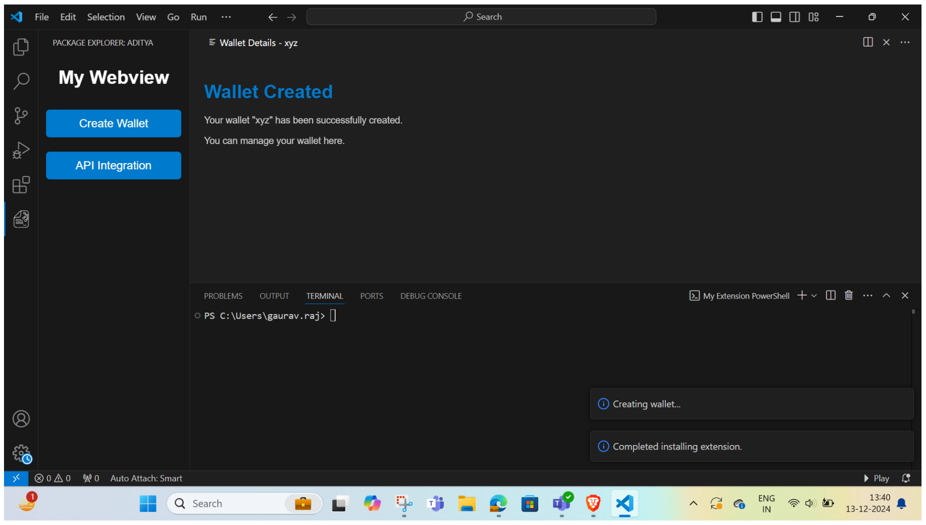


Figure 7: UI Mockup for restoration for GUI wallet using wallet seed.

- Enable wallet functionality such as sending, receiving and checking balance.
- We will position this Wallet in the sidebar of Visual Studio Code. The entire wallet will be bounded inside it, so it is easily accessible to users.



Wallet Created

Your wallet "xyz" has been successfully created.

You can manage your wallet here.

```
PS C:\Users\gaurav.raj>
```

Creating wallet...

Completed installing extension.

Figure 8: This is an UI mockup for how our extension will look in the sidebar view.

4.6 Debugging Tools for Smart Contracts and Wallet Transactions

-

- **Step-through Debugging:** The CardanoVSC extension will integrate the GHCi interpreter and Haskell debugger, enabling developers to step through Plutus smart contract code. Users will be able to inspect variables and track the flow of execution directly within Visual Studio Code. This will help identify logic errors and optimize contract behavior in a more granular and precise manner.

- **Devcontainer Execution:** The debugging environment will include a fully configured devcontainer equipped with essential Plutus development tools and the Glasgow Haskell Compiler (GHC). This environment will support both the execution and debugging of Plutus contracts, ensuring that users can seamlessly build, test, and debug their smart contracts without requiring extensive local setup.

- We create a launch configurations in launch.json to specify how the debugger should start. We also add scripts in tasks.json which allows us to debug and get rid of repetitive commands required in compiling and deployment of Plutus Smart Contracts.

- The extension will enable users to interactively run Plutus contracts within GHCi, the interactive Haskell shell, inside the configured devcontainer. Visual Studio Code will provide real-time feedback by displaying evaluation results, errors, and warnings directly in the IDE, improving the overall debugging experience and productivity.

- **Potential Risks:**
  - Complex Edge Cases: There is a risk that the system might encounter numerous edge cases, which could complicate the debugging process.
  - Debugging Challenges: The system may struggle to effectively debug very complex code, potentially leading to delays and increased development time.

4.7 Code Examples and Documentation

Include Haskell and Plutus code snippets for:

- Writing smart contracts.
- Creating and submitting transactions.

The examples should be organized and accessible from within the extension.

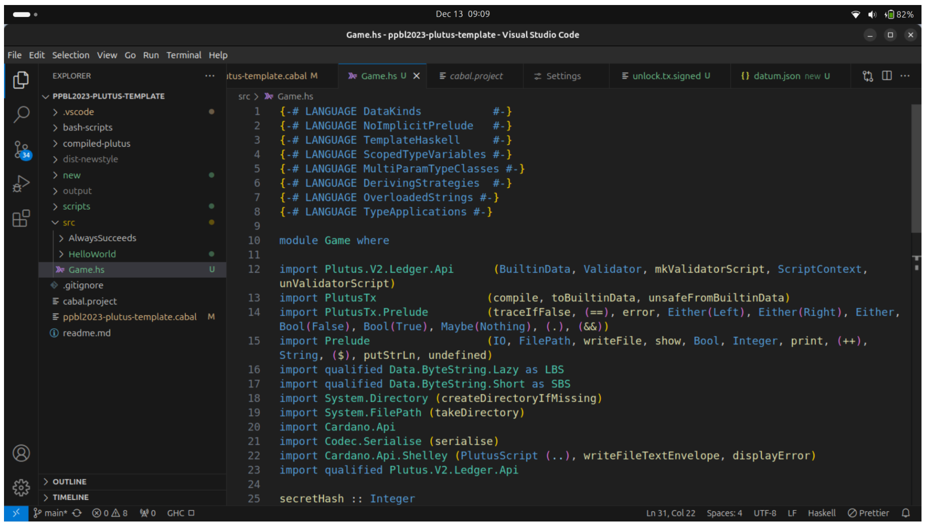
In the documentation, we will provide a comprehensive guide for:

- Setting up the extension.
- Writing and deploying smart contracts.
- Using wallet management tools.

We will also use links to external resources in the documentation(e.g., Cardano and Plutus documentation). We make the documentation for all the components we are adding to the extension, so if any developer wants to get into details they have full access to the documentation in Github Docs.We will also create a Readme.md file for Step by Step Installation and use, and we will maintain and update the file as the extension gets updated.

5 UI/UX Mockups

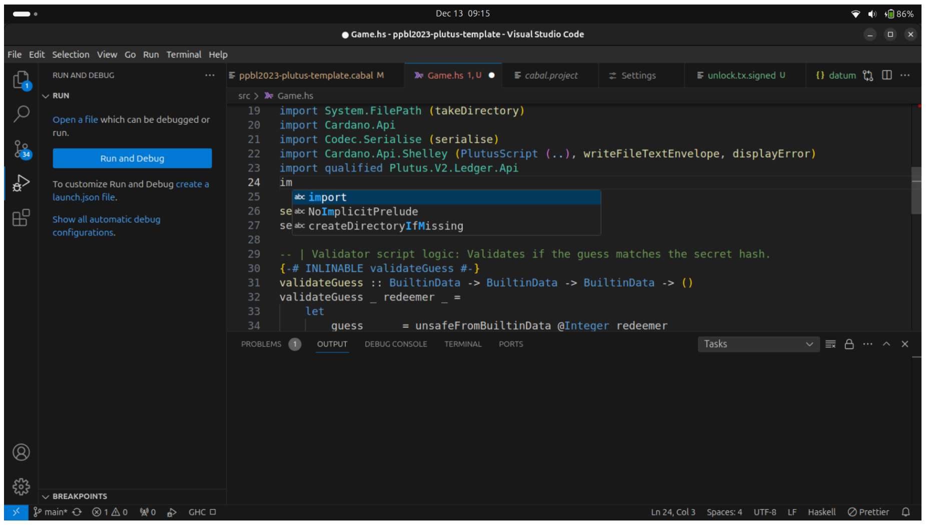
- **Syntax Highlighting:**



```haskell
{-# LANGUAGE DataKinds           #-}
{-# LANGUAGE NoImplicitPrelude   #-}
{-# LANGUAGE TemplateHaskell     #-}
{-# LANGUAGE ScopedTypeVariables #-}
{-# LANGUAGE MultiParamTypeClasses #-}
{-# LANGUAGE DerivingStrategies  #-}
{-# LANGUAGE OverloadedStrings #-}
{-# LANGUAGE TypeApplications #-}

module Game where

import Plutus.V2.Ledger.Api      (BuiltinData, Validator, mkValidatorScript, ScriptContext,
unValidatorScript)
import PlutusTx                 (compile, toBuiltinData, unsafeFromBuiltinData)
import PlutusTx.Prelude         (traceIfFalse, (==), error, Either(Left), Either(Right), Either,
Bool(False), Bool(True), Maybe(Nothing), (.), (&&))
import Prelude                  (IO, FilePath, writeFile, show, Bool, Integer, print, (++),
String, ($), putStrLn, undefined)
import qualified Data.ByteString.Lazy as LBS
import qualified Data.ByteString.Short as SBS
import System.Directory (createDirectoryIfMissing)
import System.FilePath (takeDirectory)
import Cardano.Api
import Codec.Serialise (serialise)
import Cardano.Api.Shelley (PlutusScript (..), writeFileTextEnvelope, displayError)
import qualified Plutus.V2.Ledger.Api

secretHash :: Integer
```

Figure 9: UI for syntax highlighting.

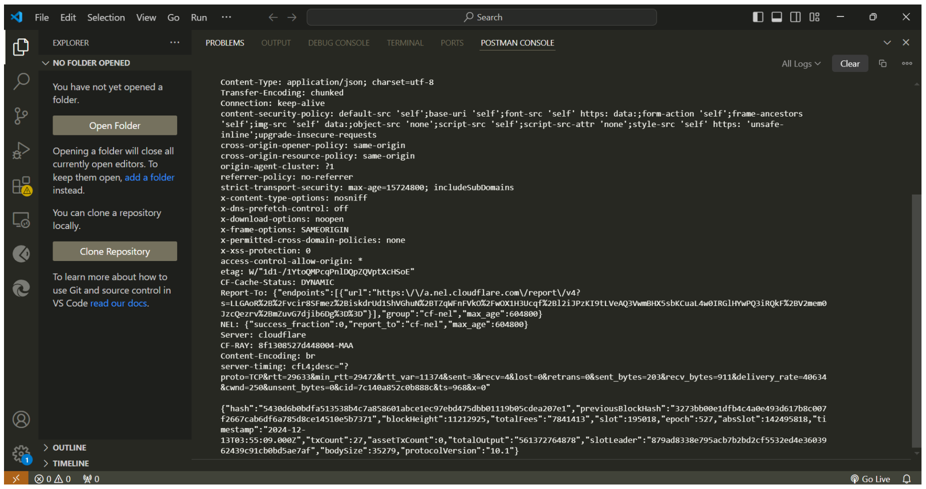
- **Code Completion:**



```haskell
import System.FilePath (takeDirectory)
import Cardano.Api
import Codec.Serialise (serialise)
import Cardano.Api.Shelley (PlutusScript (..), writeFileTextEnvelope, displayError)
import qualified Plutus.V2.Ledger.Api
im
    import
se  NoImplicitPrelude
se  createDirectoryIfMissing

-- | Validator script logic: Validates if the guess matches the secret hash.
{-# INLINABLE validateGuess #-}
validateGuess :: BuiltinData -> BuiltinData -> BuiltinData -> ()
validateGuess _ redeemer _ =
    let
        guess      = unsafeFromBuiltinData @Integer redeemer
```

Figure 10: UI for Code Completion.

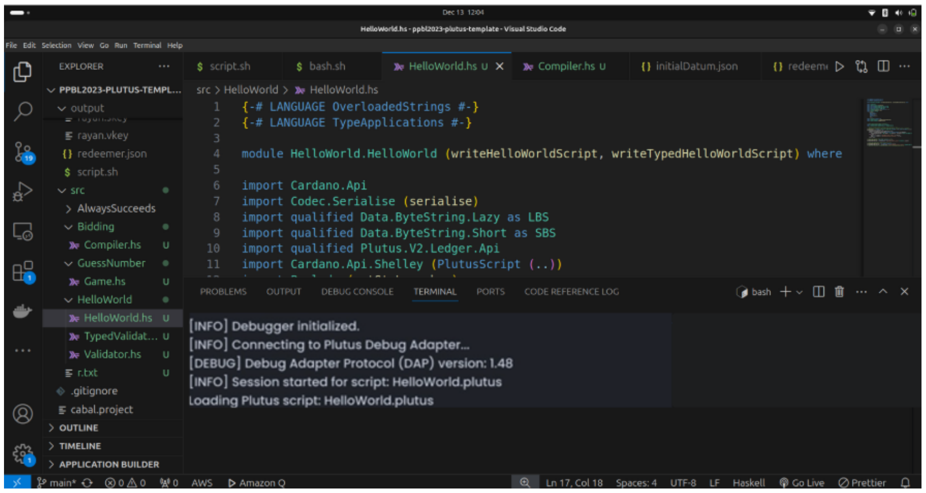
- **Cardano Blockchain explorer API Data:**


File Edit Selection View Go Run ···
Search
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
You can clone a repository locally.
Clone Repository
To learn more about how to use Git and source control in VS Code read our docs.
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
All Logs
Clear
Go Live

```
Content-Type: application/json; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
content-security-policy: default-src 'self';base-uri 'self';font-src 'self' https: data:;form-action 'self';frame-ancestors
'self';img-src 'self' data:;object-src 'none';script-src 'self';script-src-attr 'none';style-src 'self' https: 'unsafe-
inline';upgrade-insecure-requests
cross-origin-opener-policy: same-origin
cross-origin-resource-policy: same-origin
origin-agent-cluster: ?1
referrer-policy: no-referrer
strict-transport-security: max-age=15724800; includeSubDomains
x-content-type-options: nosniff
x-dns-prefetch-control: off
x-download-options: noopen
x-frame-options: SAMEORIGIN
x-permitted-cross-domain-policies: none
x-xss-protection: 0
access-control-allow-origin: *
etag: W/"1d1-/1YtoQMPcqPnlDQpZQVptXcHSoE"
CF-Cache-Status: DYNAMIC
Report-To: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?
s=LLGAoR%2B%2Fvcir8SFmez%2BiskdrUd1ShVGhuN%2BTZqWFnFVkO%2FwOX1H3Ucqf%2Bl2iJPzKI9tLVeAQ3VwmBHX5sbKCuaL4w0IRGlHYwPQ3iRQkF%2BV2mem0
JzcQezrv%2BmZuvG7djib6Dg%3D%3D"}],"group":"cf-nel","max_age":604800}
NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 8f1308527d448004-MAA
Content-Encoding: br
server-timing: cfL4;desc="?
proto=TCP&rtt=29633&min_rtt=29472&rtt_var=11374&sent=3&recv=4&lost=0&retrans=0&sent_bytes=203&recv_bytes=911&delivery_rate=40634
&cwnd=250&unsent_bytes=0&cid=7c140a852c0b888c&ts=968&x=0"

{"hash":"5430d6b0bdfa513538b4c7a858601abce1ec97ebd475dbb01119b05cdea207e1","previousBlockHash":"3273bb00e1dfb4c4a0e493d617b8c007
f2667cab6df6a785d8ce14510e5b7371","blockHeight":11212925,"totalFees":"7841413","slot":195018,"epoch":527,"absSlot":142495818,"ti
mestamp":"2024-12-
13T03:55:09.000Z","txCount":27,"assetTxCount":0,"totalOutput":"561372764878","slotLeader":"879ad8338e795acb7b2bd2cf5532ed4e36039
62439c91cb0bd5ae7af","bodySize":35279,"protocolVersion":"10.1"}
```

Figure 11: UI for Visual Studio Code Integration with CardanoScan.

- **Deployment and Debugging:**


HelloWorld.hs - ppbl2023-plutus-template - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
PPBL2023-PLUTUS-TEMPL...
output
rayan.vkey
redeemer.json
script.sh
src
AlwaysSucceeds
Bidding
Compiler.hs
GuessNumber
Game.hs
HelloWorld
HelloWorld.hs
TypedValidat...
Validator.hs
r.txt
.gitignore
cabal.project
OUTLINE
TIMELINE
APPLICATION BUILDER
script.sh bash.sh HelloWorld.hs Compiler.hs initialDatum.json redeem
src > HelloWorld > HelloWorld.hs
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS CODE REFERENCE LOG
bash
main* AWS Amazon Q Ln 17, Col 18 Spaces: 4 UTF-8 LF Haskell Go Live Prettier

```haskell
{-# LANGUAGE OverloadedStrings #-}
{-# LANGUAGE TypeApplications #-}

module HelloWorld.HelloWorld (writeHelloWorldScript, writeTypedHelloWorldScript) where

import Cardano.Api
import Codec.Serialise (serialise)
import qualified Data.ByteString.Lazy as LBS
import qualified Data.ByteString.Short as SBS
import qualified Plutus.V2.Ledger.Api
import Cardano.Api.Shelley (PlutusScript (..))
```

```
[INFO] Debugger initialized.
[INFO] Connecting to Plutus Debug Adapter...
[DEBUG] Debug Adapter Protocol (DAP) version: 1.48
[INFO] Session started for script: HelloWorld.plutus
Loading Plutus script: HelloWorld.plutus
```

Figure 12: A Visual representation of Plutus debugging and deployment UI in Visual Studio Code.

6 Architecture diagram

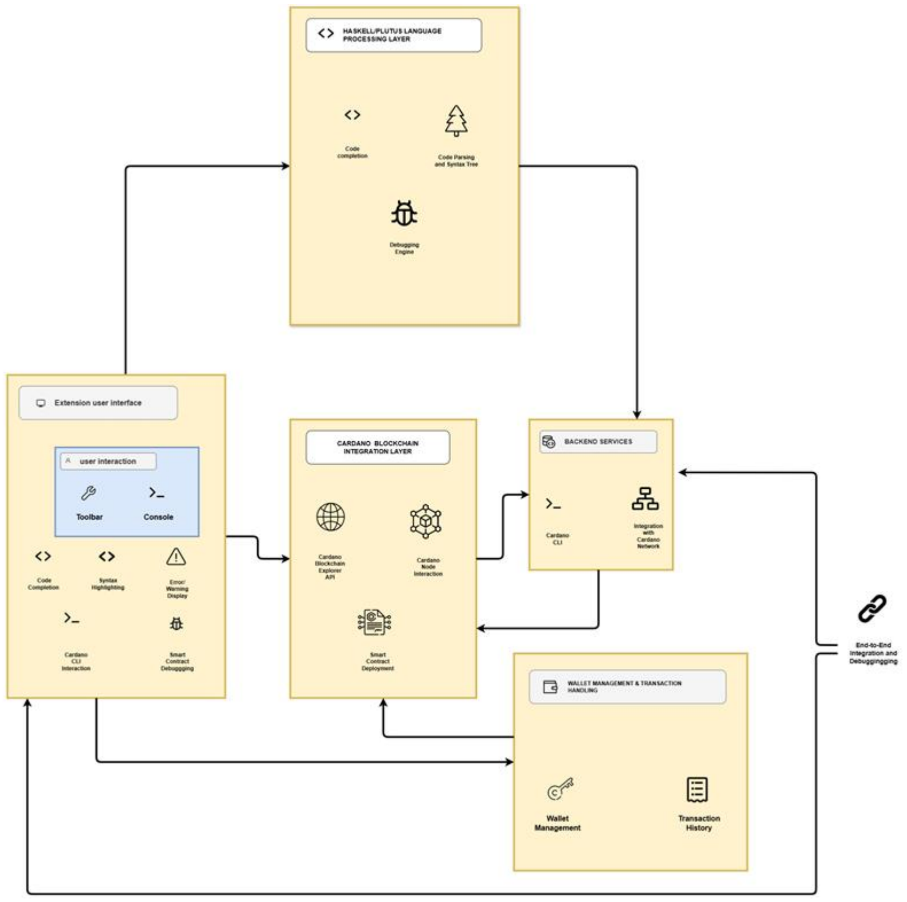

HASKELL/PLUTUS LANGUAGE PROCESSING LAYER
Code completion
Code Parsing and Syntax Tree
Debugging Engine
Extension user interface
user interaction
Toolbar
Console
Code Completion
Syntax Highlighting
Error/ Warning Display
Cardano CLI Interaction
Smart Contract Debuggging
CARDANO BLOCKCHAIN INTEGRATION LAYER
Cardano Blockchain Explorer API
Cardano Node Interaction
Smart Contract Deployment
BACKEND SERVICES
Cardano CLI
Integration with Cardano Network
End-to-End Integration and Debuggingging
WALLET MANAGEMENT & TRANSACTION HANDLING
Wallet Management
Transaction History

Figure 13: Visual Studio code Architecture.

7 Conclusion

This Plutus and Haskell extension for VS Code(CardanoVSC) aims to enhance the developer experience by providing syntax highlighting and code completion. Throughout this document, we have outlined the design principles, architecture, and implementation details that drive the extension. By adhering to these guidelines, we ensure that the extension remains maintainable, scalable, and user-friendly. Future enhancements should continue to align with these core principles to maintain the integrity and usability of the extension. This scope and design document is fundamental to the development work, which will be carried out in next milestones.