**Cypherock Technical Screening task - Full**

**Stack- Bitcoin Management System**

1. **Candidate information**

Name-Abhinaw Ratan

Email- [Ratanabhinaw@gmail.com](mailto:Ratanabhinaw@gmail.com)

Phone-no: +917523817665

Resume-

1. **Introduction**
2. **Installation**
3. **Usage**
4. **Dependencies**
5. **Test Outputs**

**1.Introduction-**

The Bitcoin Wallet Management code is a Node.js application that allows users to create and manage Bitcoin wallets. It provides a set of functions to perform various wallet operations, including creating wallets, importing wallets from mnemonic phrases, listing existing wallets, checking balance, fetching transactions, and generating unused addresses.

**2.Installation-**

To run the Bitcoin Wallet Management code, please follow these steps:

* Ensure you have Node.js installed on your machine. If not, download and install Node.js from the official website: [https://nodejs.org](https://nodejs.org/)
* Clone or download the code from the repository.
* Open a terminal or command prompt and navigate to the project directory.
* Install the required dependencies by running the following command:

Copy code

npm install

* You're now ready to use the Bitcoin Wallet Management code!

**3.Usage-**

The Bitcoin Wallet Management code can be used as a command-line application to perform various wallet operations. The available commands are:

* **create <walletName>**: Creates a new Bitcoin wallet with the specified name.
* **import <walletName> <mnemonic>**: Imports a BIP39 wallet using the provided mnemonic phrase and assigns it the specified name.
* **list**: Lists all existing wallets.
* **balance <walletName>**: Retrieves the balance of the specified wallet.
* **transactions <walletName>**: Fetches the transactions associated with the specified wallet.
* **generate-address <walletName>**: Generates an unused Bitcoin address for the specified wallet.

To execute a command, run the following command in the terminal:

node main.js <command> [arguments]

**4.Code Structure**

The code follows a modular structure to group related functionalities together. Here's an overview of the code structure:

* **wallet.js**: Contains functions related to wallet operations, including creating wallets, importing wallets, listing wallets, checking balance, fetching transactions, and generating addresses.
* **api.js**: Contains functions for making API calls to interact with the Bitcoin blockchain.
* **util.js**: Contains utility functions used by other modules.
* **main.js**: An example file showcasing how to use the functions from the **wallet.js** module.

You can organize the code files further based on your project structure and requirements. Ensure to keep related functions together and maintain a clear separation of concerns.

**5.Dependencies**

The Bitcoin Wallet Management code relies on the following dependencies:

* **axios**: Used for making HTTP requests to interact with the Bitcoin blockchain.
* **dotenv**: Used for loading environment variables from a **.env** file.
* **bitcoinjs-lib**: Used for Bitcoin wallet creation and address generation.
* **bip39**: Used for generating and converting mnemonic phrases.

These dependencies are defined in the **package.json** file and will be automatically installed when you run **npm install**.

**6.Test Outputs**

Here are some sample test outputs for the Bitcoin Wallet Management code:

* Creating a new wallet:

Write node cypherrock.js create test

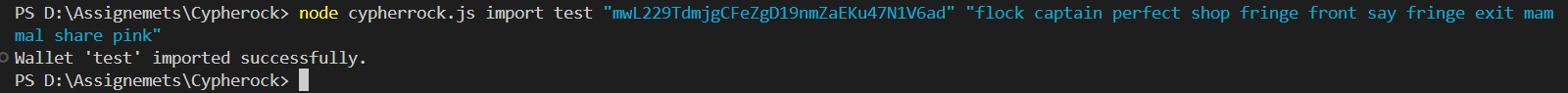
A black screen with white text

Description automatically generated with low confidence

This will generate and store a new BIP39 wallet with the name given by the user and store it locally.

* Importing a wallet:

Write node cypherrock.js import “<address >” “<mnemonics>”



This will import the wallet as test.json and will include the address and mnemonics in the json file and store it locally.

* Listing all wallets:

Write node cypherrock.js list

A picture containing text, screenshot, font

Description automatically generated

This function will list all the existing wallets in the local system with their address and mnemonics.

* Getting Bitcoin Balance:

Write node cypherrock.js balance <wallet\_name>

A picture containing text, screenshot, font, line

Description automatically generated

I have sent some testnet btc to this particular address and the transaction function converts the Satoshi into BTC and console in a human readable form

* Getting Bitcoin Transaction:

Write node cypherrock.js transaction <wallet\_name>

A screen shot of a computer

Description automatically generated with low confidence

* Generating an unused bitcoin:

Write node cypherrock.js generate-address

Output- Unused address generated for Wallet 'MyWallet': 1ABCxyz