Crypto Investment Platform Project

1. Project Overview

The Crypto Investment Platform project is powered by using Python, Tkinter, and SQL. The platform enables users to create accounts, view account, manage their funds, and interact with different crypto assets through graphical interface.

Users can perform actions such as buying and selling assets, viewing their portfolios, and depositing or withdrawing funds. Each action communicates with a server to keep the data updated in real-time.

In my implementation, I focused on making the system easy to use and secure, ensuring smooth interactions for the users. I paid special attention to handling errors, such as missing input fields, and making sure all user actions were validated. The platform's functions, like managing assets and viewing portfolios, were made simple and responsive. Additionally, I used SQL to store all transactions and manage user data, which helps in keeping everything consistent and accessible while ensuring that transactions are processed smoothly.

- **List of files**
    - crypto_investment_SQL.py
    - assets.py
    - server.py
    - client_GUI.py
    - crypto_investment_db.sql

Description of File Functionalities:

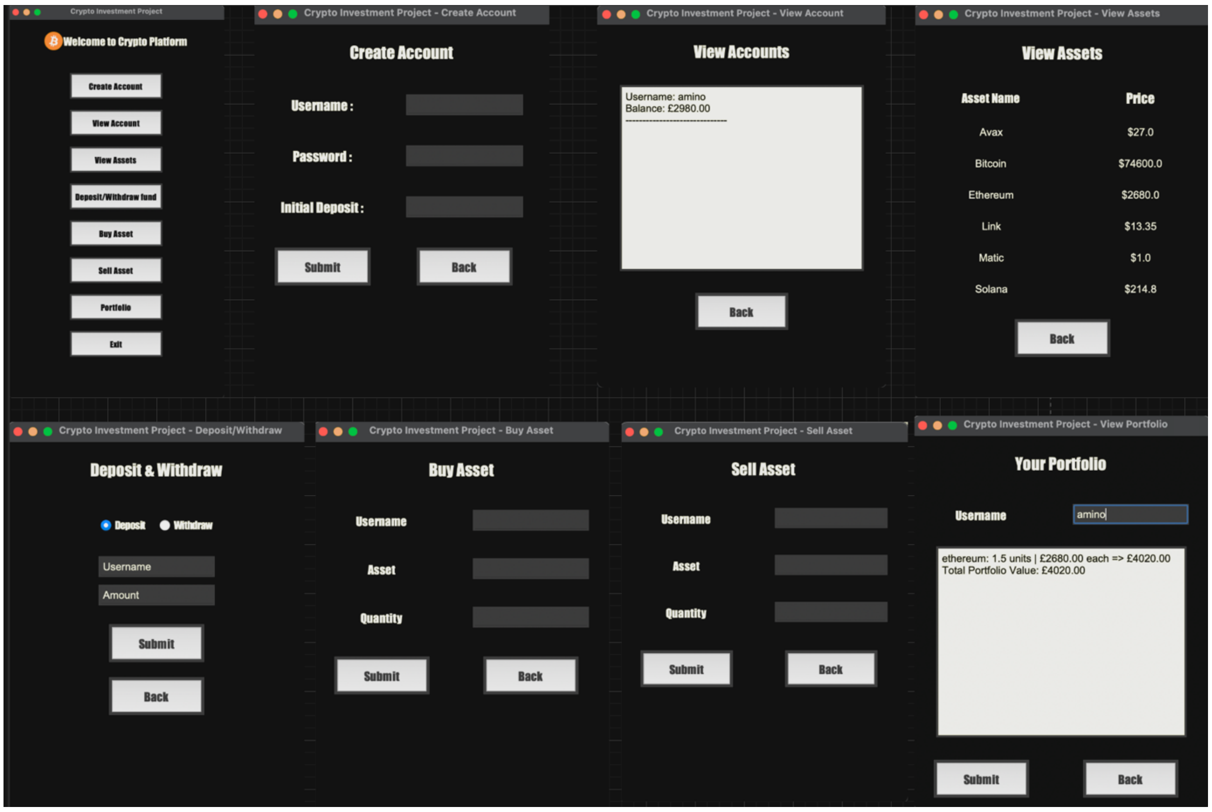
- **crypto_investment_SQL.py**: The script uses **MySQL** database for storing and fetching data. It handles accounts, assets, deposits, withdrawals, and crypto trades while keeping balances, portfolios, and transaction records updated. Users can check their portfolio details, including asset quantities, prices, and total value, with accurate and error handling.
- **server.py**: The script runs a server for handling client requests like managing accounts, assets, transactions, and portfolios. It uses sockets for communication, integrates 'crypto_investment_SQL.py' functions, and sends responses using pickle.
- **client_GUI.py**: This script implements the graphical user interface, for a Crypto Investment Platform, built with Python's **Tkinter** module. It connects to server via socket enables users to manage cryptocurrency assets, including creating accounts, viewing portfolios, and buying or selling assets. The interface includes, labels, buttons, radio buttons, text box, entry box and many more.
- **assets.py:** This file uses Object-Oriented Programming (**OOP**) to create a blue print of assets structure and related methods within a classes.
- **crypto_investment_db.sql:** The script creates a database called crypto. It defines four tables: **accounts** for user details and deposits, **assets** for cryptocurrency names and prices, **transactions** to record buy and sell actions, and **portfolio** to track user holdings. The script also adds initial data for various cryptocurrencies to the assets table.

3. Implementation Plan

3.1 The project was implemented in the following stages:

1. Initial Text-Based Setup:
    - Start by creating an account with a custom username, password and initial balance and storing into **account.txt**.
    - Investment Options: Browse available investment types and view current prices.

   - Trading: Buy or sell based on current funds or asset holdings.
   - Portfolio Management: View current holdings and monitor their performance over time for each asset.
   - Saved transaction logs in **transactions.txt** for record-keeping each time user buys or sell assets.
2. Object-Oriented Refactoring:
   - Defined an **asset class** with name, price, and quantity. Added methods like ‘get _name’ and ‘get_current_price’ allowing easy access to asset information within the platform.
3. Client-Server Model:
   - Developed a server program to manage account and portfolio information. Load accounts assets and portfolio data from **accounts.txt**, **assets.txt** and **portfolio dictionary** on server startup.
   - Created a client application to request create account, view accounts, view assets, deposit & withdraw funds, buy and sell assets, view portfolio and exit the server and client side.
4. Database Integration:
   - Transitioned from text files to an MySQL database for better performance and scalability.
   - Used SQL queries for: create account, view accounts, view assets, deposit & withdraw funds, buy and sell assets, view portfolio and save transactions.
5. GUI Implementation:
   - Built a Tkinter-based graphical interface for above functions and creating eight buttons for each function.



```
Username: amino
Balance: £2980.00
----------------------------
```

Asset Name	Price
Avax	$27.0
Bitcoin	$74600.0
Ethereum	$2680.0
Link	$13.35
Matic	$1.0
Solana	$214.8

Username: amino

```
ethereum: 1.5 units | £2680.00 each => £4020.00
Total Portfolio Value: £4020.00
```

4. Running Instructions

4.1 Prerequisites:

- Python 3.x installed on your system.
- Required libraries: socket, pickle, MySQL, tkinter, tkinter(messagebox), mysql-connector-python.

4.2 Steps to Run:

1. Connect to database and run the **crypto** database
2. Start the server: server.py
3. Run the client application: client_GUI.py

5. Future Improvements

- User authentication

- Real-time API integration

- Better UI

- Transaction filters

- Docker support

License

Author

Created by Amin Kazemi

Cybersecurity and digital forensic Student

Feel free to connect or contribute!