Cryptocurrency Trading Simulator

This is a web application that simulates a cryptocurrency trading platform. It allows users to view real-time cryptocurrency prices, manage a virtual account with balance and holdings, perform simulated transactions (buy/sell), and view their transaction history. The application also includes user registration and login functionalities, with access to core trading operations available only to authenticated users via JWT token.

1. Overview

The project is a cryptocurrency exchange simulator that offers the following core functionalities:

- **Real-time Prices:** Dynamic display of top 20 cryptocurrency prices obtained from the Kraken V2 WebSocket API.
- **Virtual Account:** Maintenance of a virtual balance ($10,000 initial value) and management of cryptocurrency holdings.
- **Trading Operations:** Ability to buy and sell cryptocurrencies with automatic balance and holdings updates. Validation for sufficient balance/holdings is included.
- **Transaction History:** A log of all completed transactions, including type, symbol, quantity, and price at the time of transaction.
- **Account Reset:** A button to restore the account to its initial state (balance $10,000, no holdings, cleared history).
- **User Authentication:** Registration and login functionalities for users, utilizing JWT tokens for secure access to trading operations.

2. Technologies

- **Frontend:** Angular (HTML, CSS, TypeScript)
- **Backend:** Java (Spring Boot)
- **API Integration:** Kraken V2 WebSocket API (for real-time prices)
- **Database:** MySQL
- **WebSockets:** Spring WebSockets (for Backend-Frontend communication) and Java-WebSocket (for Kraken communication)
- **JSON Processing:** Jackson, Gson
- **Authentication:** JWT (JSON Web Token)

3. Installation and Startup

Follow these steps to install and run the application locally.

3.1. Prerequisites

Ensure you have the following tools installed:

- Java Development Kit (JDK) 17 or newer
- Apache Maven (for Spring Boot Backend)

- Node.js and npm (or Yarn) (for Angular Frontend)
- Angular CLI (`npm install -g @angular/cli`)
- MySQL Server

3.2. Database Setup (MySQL)

1. **Create Database:** Open a MySQL client (e.g., MySQL Workbench, command line) and execute the following command to create the database:

```
CREATE DATABASE cryptosim;
```

3. **Execute DDL Script:** Execute the provided DDL script (`crypto-sim-ddl.sql`) to create the necessary tables and insert an initial account record.
4. **Backend Configuration:** Update the `backend/src/main/resources/application.properties` file with your MySQL database connection details:

```
spring.datasource.username=your_mysql_username
spring.datasource.password=your_mysql_password
# Add JWT configuration, for example:
jwt.secret=YOUR_SUPER_SECRET_KEY_FOR_JWT_SIGNING
```

3.3. Backend Startup (Spring Boot)

The backend will start on `http://localhost:8080`.

3.4. Frontend Startup (Angular)

```
cd frontend //Navigate to the Frontend directory
npm install
ng serve --open
```

The frontend will compile and start a development server on `http://localhost:4200`. The application will automatically open in your browser.

4. Application Usage

Once both the backend and frontend are running, open `http://localhost:4200` in your web browser.

- **Registration and Login:** Before accessing trading functionalities, you will need to register and log in to the system.
- **Real-time Price Overview:** On the home screen, you will see a table with the real-time prices of the top 20 cryptocurrencies.
- **Virtual Account and Trading:**
    - Your current virtual balance will be displayed at the top of the screen.
    - **Buying:** Select a cryptocurrency from the table, enter the quantity, and click "Buy". Your balance and holdings will be updated.
    - **Selling:** Select a cryptocurrency from the table, enter the quantity, and click "Sell". Your balance and holdings will be updated.
    - The application will display error messages for insufficient balance/holdings or invalid input.

- **Transaction History:** Click the "Transactions" button to view a list of all performed transactions.
- **Owned Currencies:** Click the "Wallet" button to see a list of all cryptocurrencies you own.
- **Account Reset:** Click the "Reset" button to revert your balance to the initial value ($10,000) and clear all holdings and transaction history.

5. Application Architecture

The application is divided into two main parts:

- **Frontend (Angular):** A Single-Page Application (SPA) built with Angular. It communicates with the backend via REST API for account and transaction data, and via STOMP WebSocket for real-time price updates. All authenticated requests send a JWT token in the headers.
- **Backend (Spring Boot):** A Java application based on Spring Boot. It provides REST API endpoints for managing accounts, transactions, and **user authentication**. The backend also maintains a WebSocket connection with the Kraken V2 WebSocket API, listening for cryptocurrency price updates, and pushes them to the frontend via STOMP WebSocket.
- **Database (MySQL):** A relational database used to store account information (balance, holdings), transaction history, and **user data (username, hashed password, email)**. Direct JDBC SQL queries are used, without an ORM, as requested.
- **API Integration:** The `KrakenWebSocketClient` in the backend maintains a persistent connection with the Kraken V2 WebSocket API, listening for cryptocurrency price updates.

6. API Endpoints (Backend)

The backend provides the following REST API endpoints:

6.1. Authentication

- **`POST /auth/register`**
  - **Description:** Registers a new user.
  - **Request Body:** `{"username": "your_username", "password": "your_password", "email": "your_email"}`
  - **Response:** Success message or error.
- **`POST /auth/login`**
  - **Description:** Logs in a user and generates a JWT token. Returns the authenticated user's current account balance and holdings.
  - **Request Body:** `{"username": "your_username", "password": "your_password"}`
  - **Response:** A JSON user object containing a JWT token (e.g., `{"token": "..."}`). This token should be stored by the frontend and sent with every authenticated request.

6.2. Account and Trading (Requires JWT Token)

- **`GET /transactions`**
  - **Description:** Returns a collection of transactions for the authenticated user.
  - **Requires:** `Authorization: Bearer <JWT_TOKEN>` in the header.
  - **Response:** A list of `TransactionDto` objects (JSON).
- **`POST /transactions/buy`**
  - **Description:** Performs a cryptocurrency purchase for the authenticated user.
  - **Requires:** `Authorization: Bearer <JWT_TOKEN>` in the header.
  - **Request Body:** `{"currency": "BTC/USD", "quantity": "0.01", "price: "0.01", "type": "BUYING", "userID": "1"}`
  - **Response:** The authenticated user's updated balance as a JSON object.
- **`POST /transactions/sell`**
  - **Description:** Performs a cryptocurrency sale for the authenticated user.
  - **Requires:** `Authorization: Bearer <JWT_TOKEN>` in the header.
  - **Request Body:** `{"currency": "BTC/USD", "quantity": "0.01", "price: "0.01", "type": "SELLING", "userID": "1"}`
  - **Response:** The authenticated user's updated balance as a JSON object.
- **`POST /transactions/reset`**
  - **Description:** Resets the authenticated user's account balance to the initial value and clears holdings and transactions.
  - **Requires:** `Authorization: Bearer <JWT_TOKEN>` in the header.
  - **Request Body:** `AppUserDto` as a JSON object.
  - **Response:** `AppUserDto` with a balance of 10000 as a JSON object.

6.3. Wallet

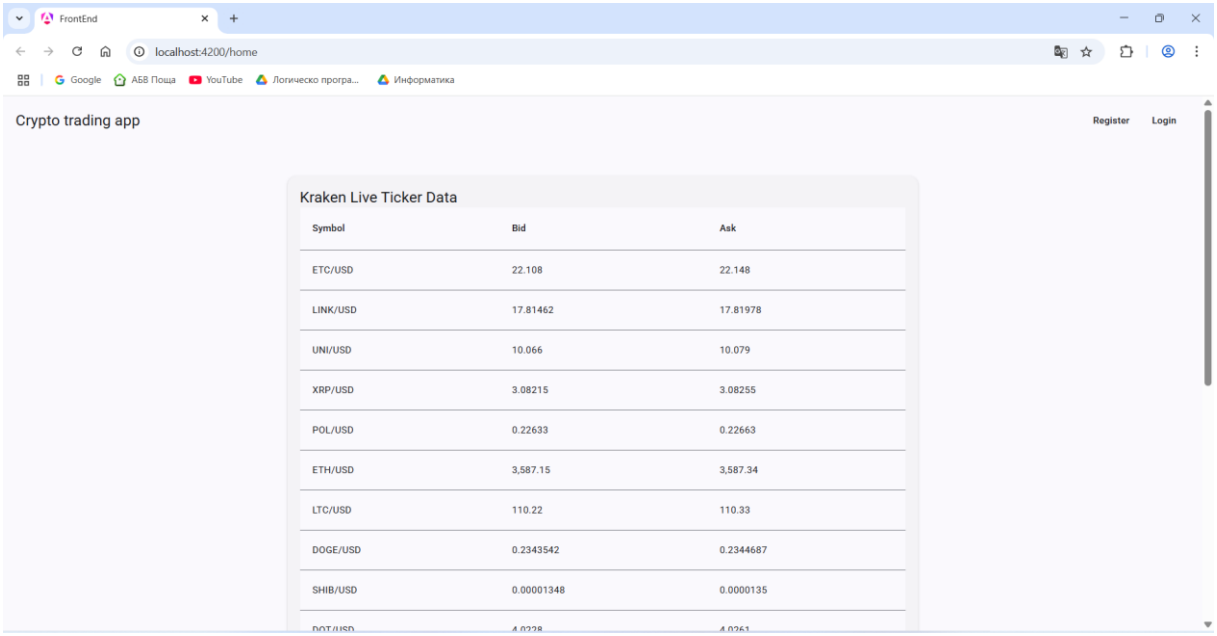
- **`GET /wallets`**
  - **Description:** Returns the quantity of a specific cryptocurrency owned by the authenticated user.
  - **Requires:** `Authorization: Bearer <JWT_TOKEN>` in the header.
  - **Request Body:** `{"userID": "1", "baseCurrency": "BTC", "quoteCurrency": "USD"}`
  - **Response:** `BigDecimal` as a JSON object.
- **`GET /wallets`**
  - **Description:** Returns a list of cryptocurrencies owned by the authenticated user.
  - **Requires:** `Authorization: Bearer <JWT_TOKEN>` in the header.
  - **Request Body:** `{"userID": "1"}`
  - **Response:** A list of `CurrencyDto` objects (JSON).

6.4. Prices (Accessible without Authentication)

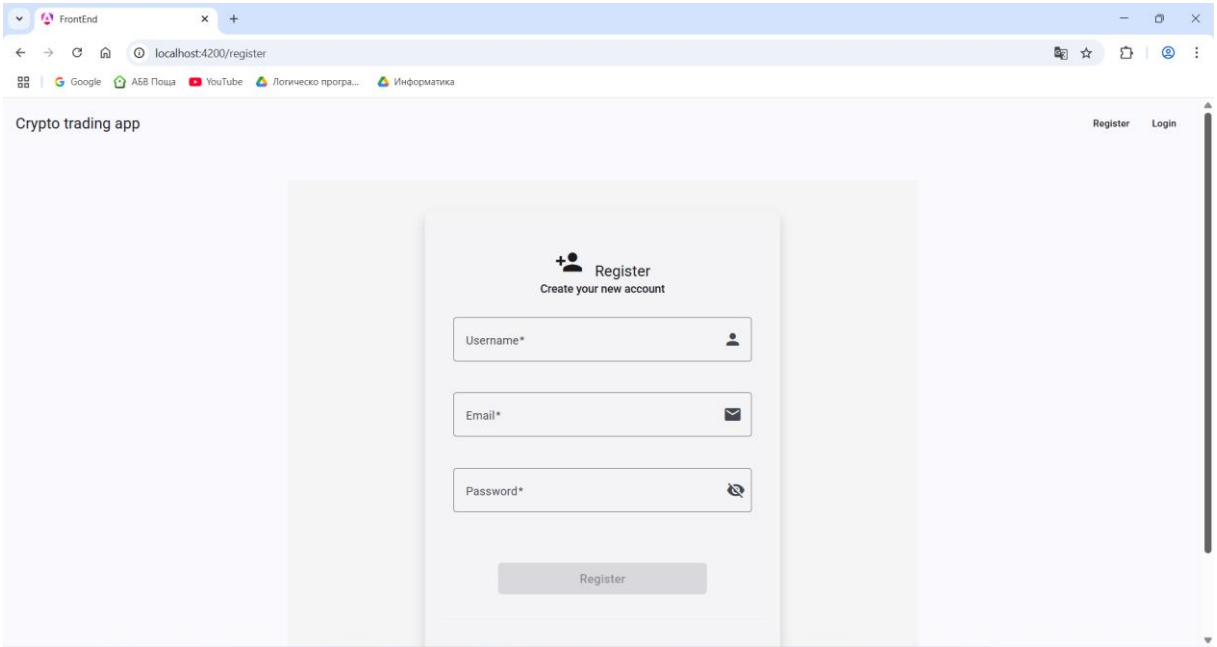
- **`GET /tickers`**
  - **Description:** Returns the latest known prices for all subscribed cryptocurrencies. This endpoint is primarily for initial loading; real-time updates are sent via WebSocket.
  - **Response:** A list of `TickerDto` objects (JSON).

7. Screenshots

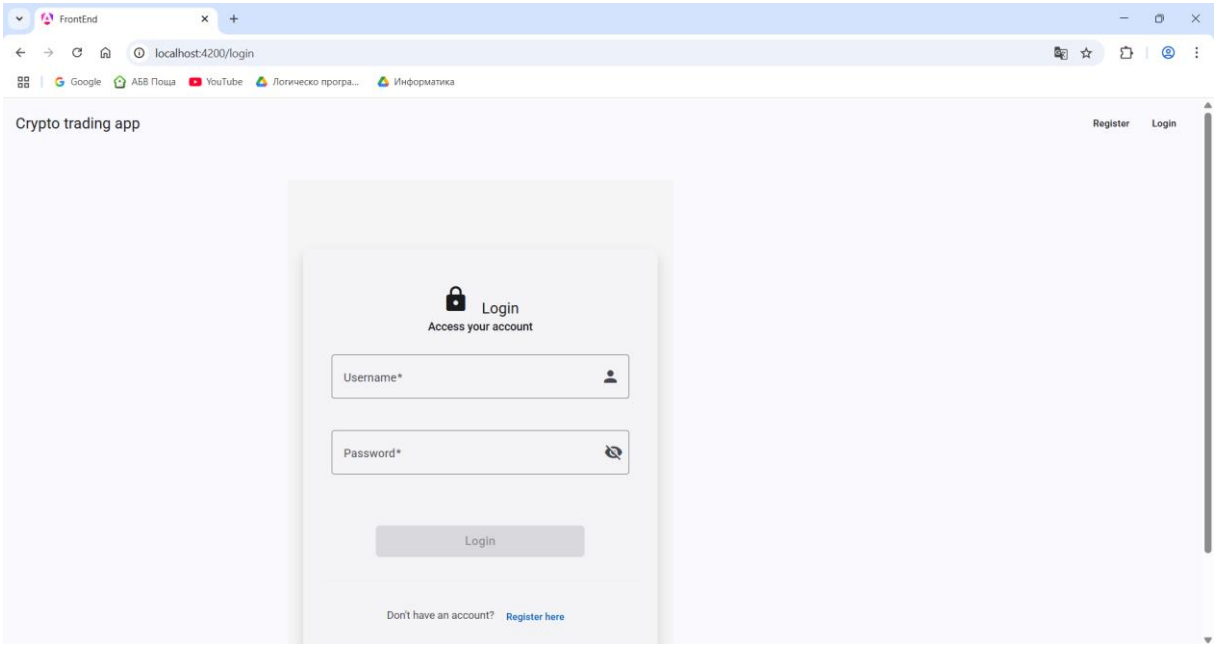
Initial Screen with Prices: Displays the main application interface with a real-time cryptocurrency price table and the user's current balance when authenticated.



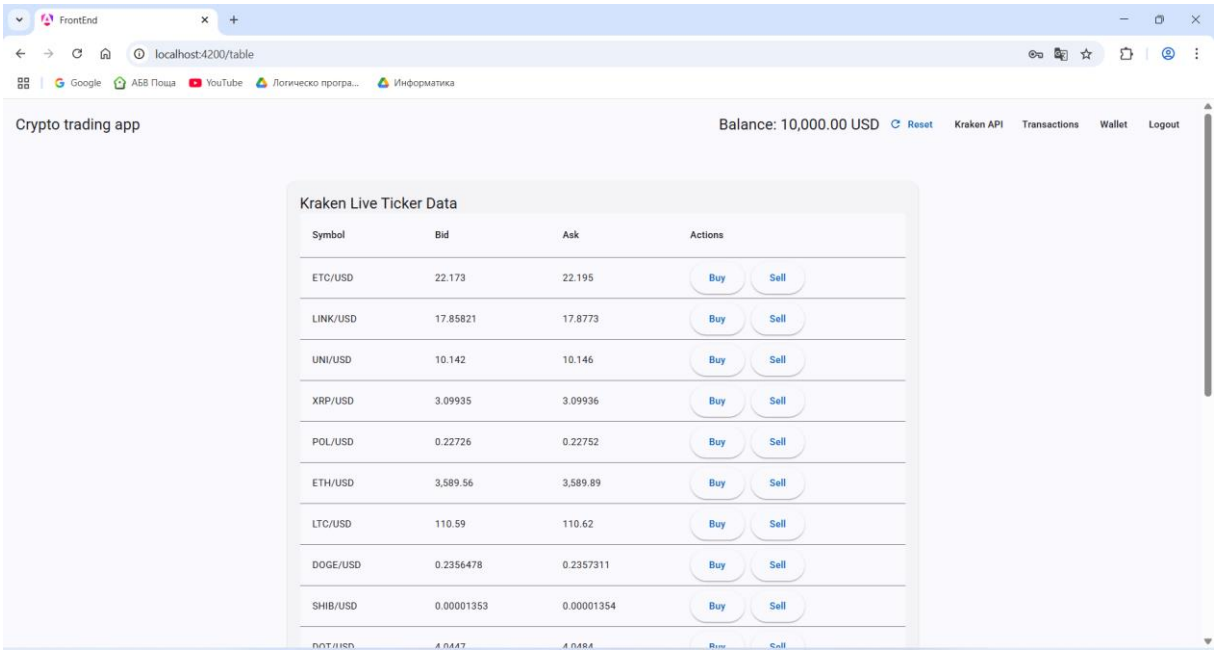
User Registration Screen: Demonstrates the form through which new users can create an account.

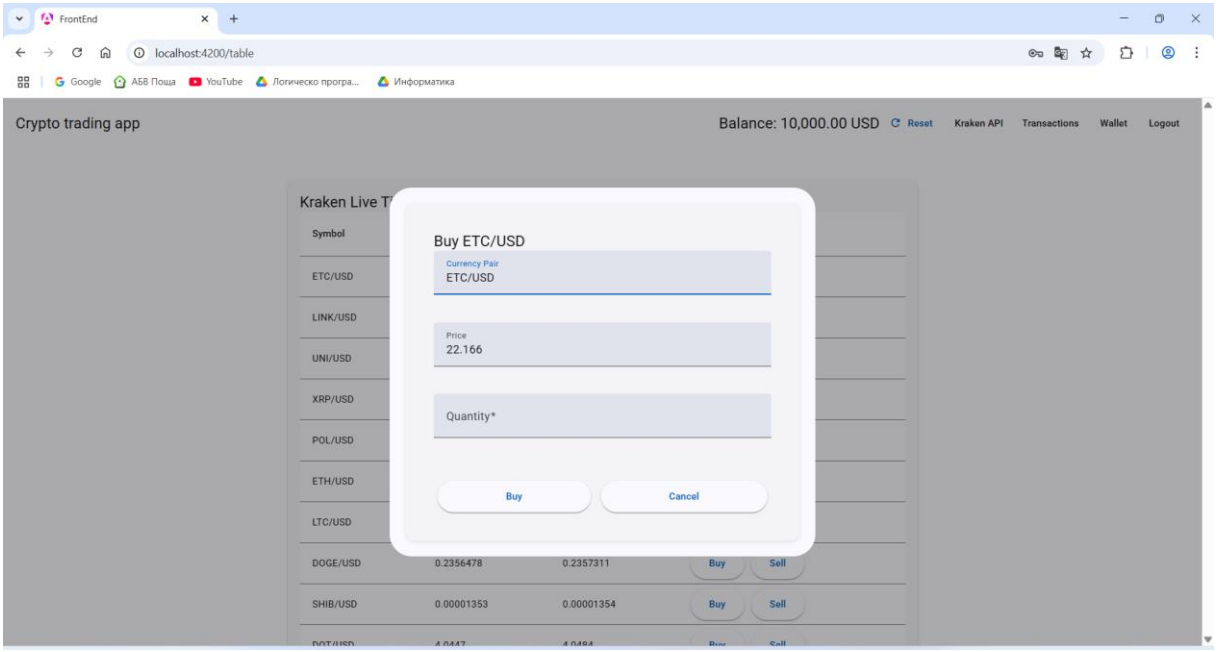


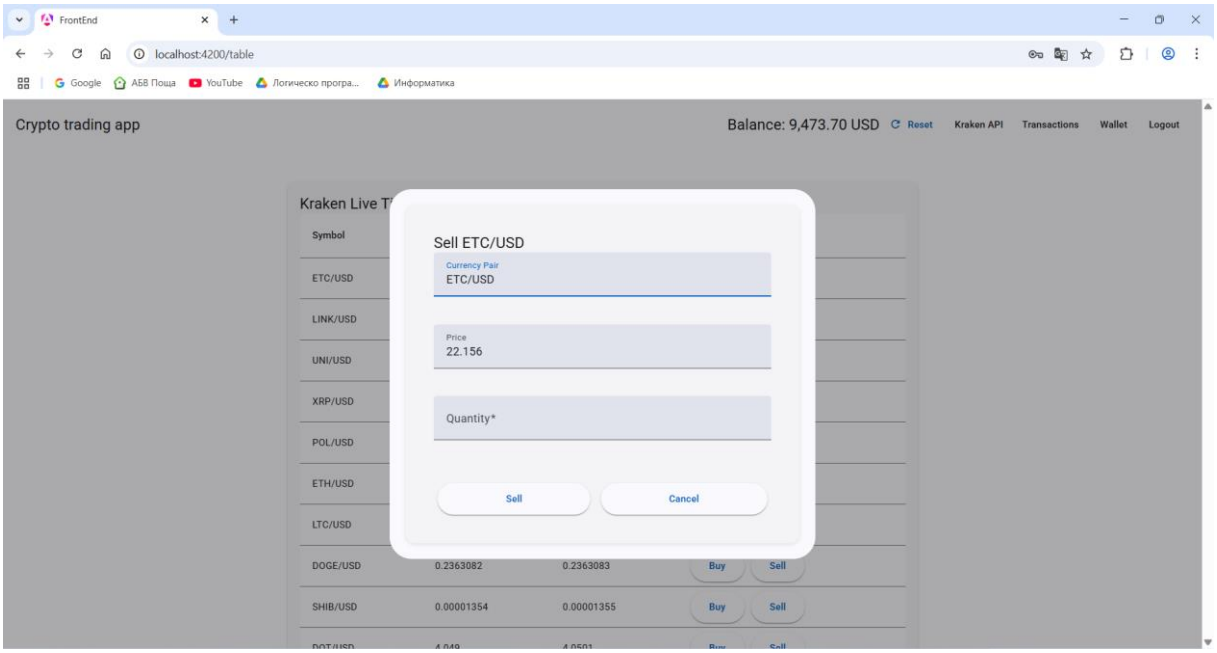
User Login Screen: Shows the login form where existing users can authenticate.



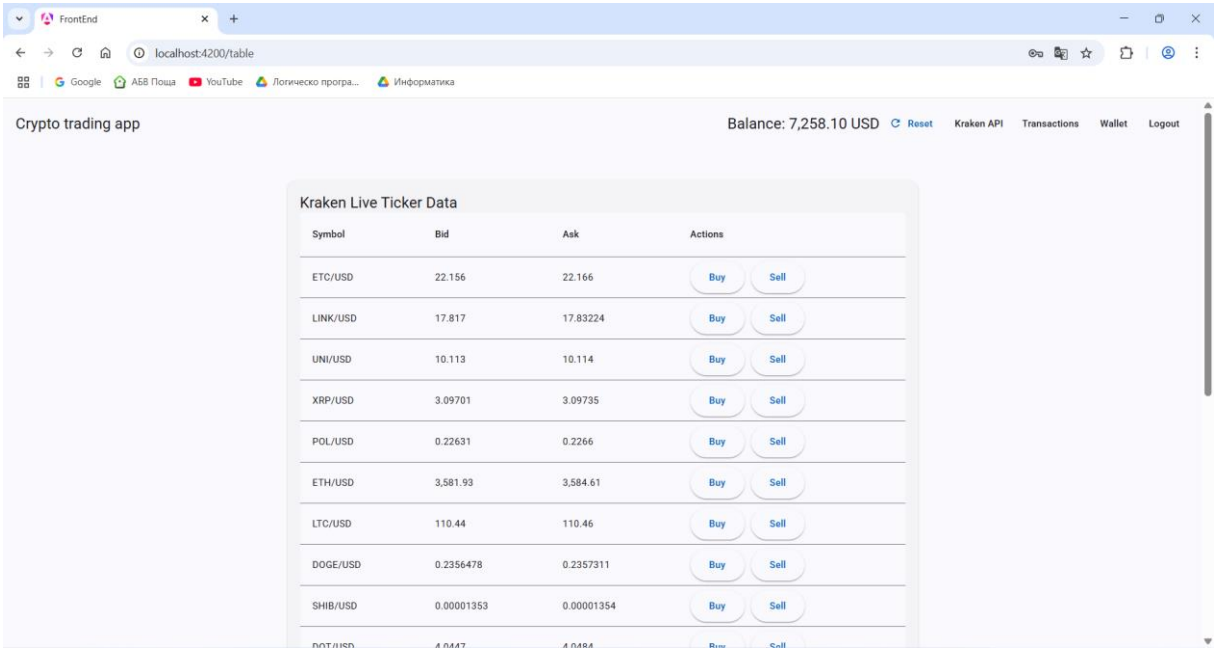
Buy/Sell Interface after Login: Illustrates the trading screen, accessible after successful login, with buy and sell options.



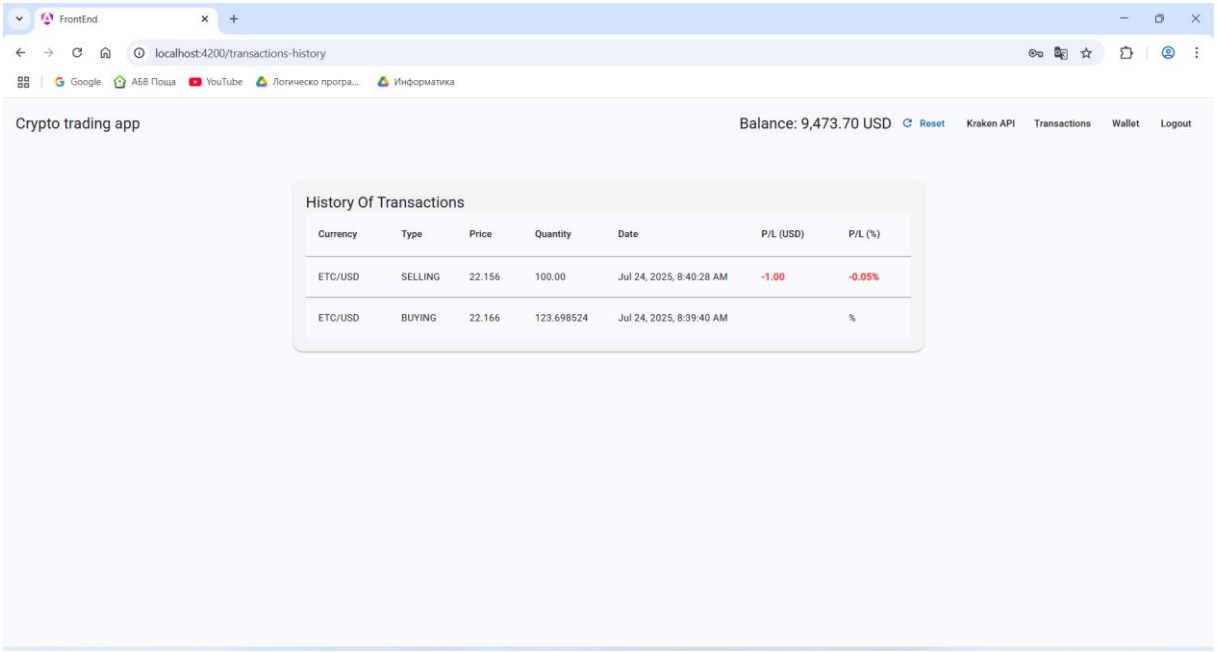

FrontEnd
localhost:4200/table
Google
АБВ Поща
YouTube
Логическо програ...
Информатика
Crypto trading app
Balance: 10,000.00 USD
Reset
Kraken API
Transactions
Wallet
Logout
Kraken Live T
Symbol
ETC/USD
LINK/USD
UNI/USD
XRP/USD
POL/USD
ETH/USD
LTC/USD
DOGE/USD
0.2356478
0.2357311
Buy
Sell
SHIB/USD
0.00001353
0.00001354
Buy
Sell
Buy ETC/USD
Currency Pair
ETC/USD
Price
22.166
Quantity*
Buy
Cancel


FrontEnd
localhost:4200/table
Google
АБВ Поща
YouTube
Логическо програ...
Информатика
Crypto trading app
Balance: 9,473.70 USD
Reset
Kraken API
Transactions
Wallet
Logout
Kraken Live T
Symbol
ETC/USD
LINK/USD
UNI/USD
XRP/USD
POL/USD
ETH/USD
LTC/USD
DOGE/USD
0.2363082
0.2363083
Buy
Sell
SHIB/USD
0.00001354
0.00001355
Buy
Sell
Sell ETC/USD
Currency Pair
ETC/USD
Price
22.156
Quantity*
Sell
Cancel

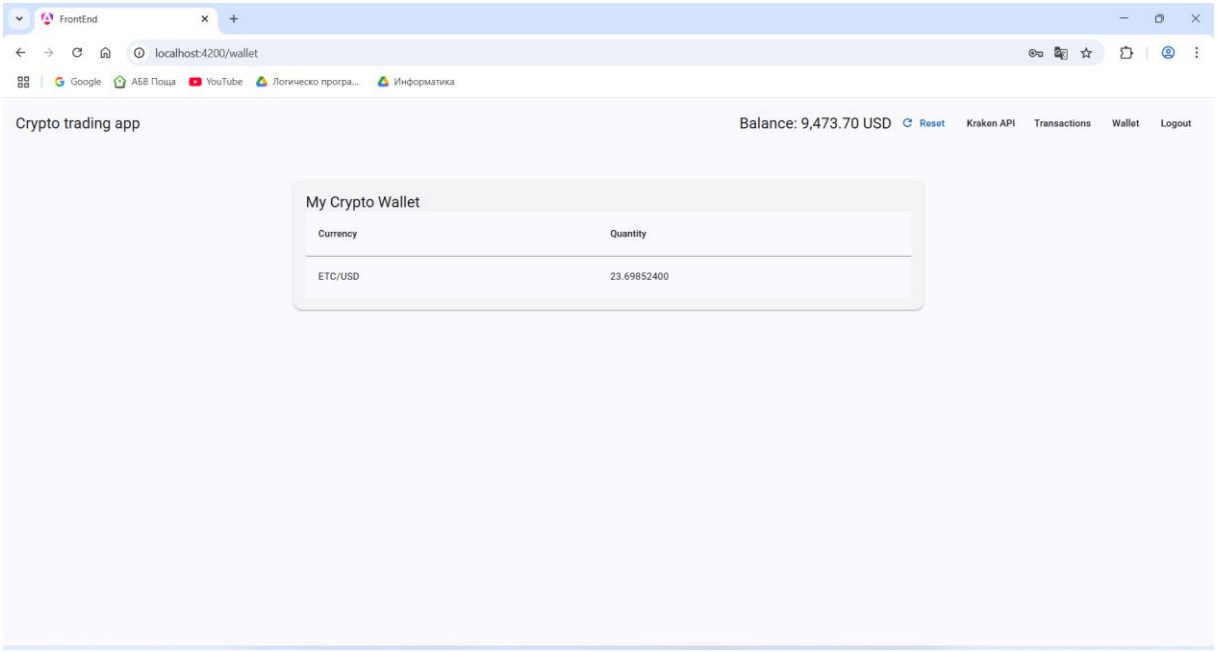
Updated Account Balance after Transaction: Shows how the user's balance and holdings change after performing a buy or sell operation.



Transaction History: Presents the log of all past transactions, including details like type, symbol, quantity, and price.



All Cryptocurrencies Owned by the User: Shows a comprehensive list of all digital assets currently held by the authenticated user in their virtual wallet.



8. Testing

Several tests of the services