Fox of Hood: Programmer's Guide

Introduction to the application

Fox of Hood (abbreviated FOH) is a stock portfolio and trading web application upon which users can buy, sell, adjust, and view stocks in their personal portfolio. This documentation is intended to provide information and instruction as to the proper development and maintenance of the product. As such, the tech stack used for the application includes Flask (python), SQL, HTML, CSS, JavaScript, and the Yahoo Finance API.

Overview

1. *Application architecture*

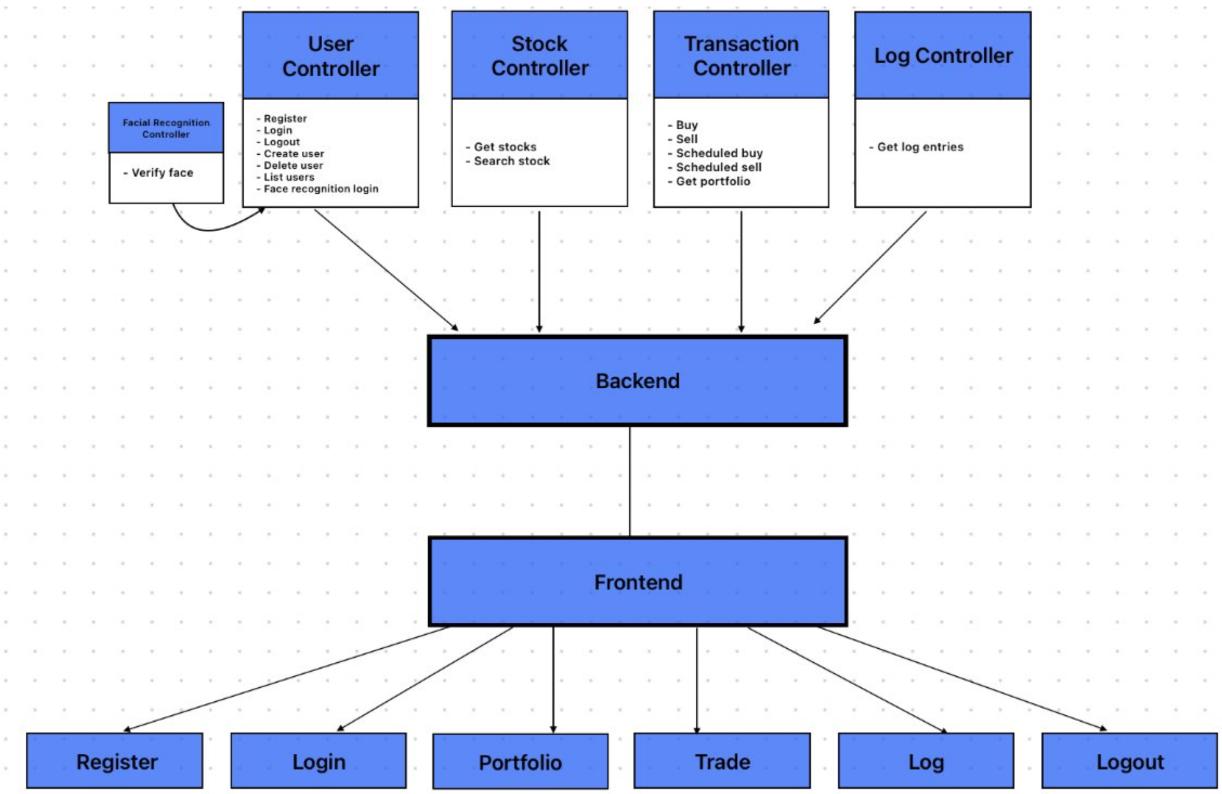


Figure 1. Application architecture

2. *Key features*

The application features main functionalities including the following: hosting user accounts, buying stocks, selling stocks, stock adjustments (scheduled buying and selling of stocks), and the logging of user actions.

3. Directory structure

```
-backend
        -app
                -controllers
                -logs
        -_init_.py
        -db.py
        -schema.sql
        -uploads
        -.env
        -requirements.txt
        -server.py
-frontend
        -app
                -static
                -templates
                -_init_.py
                -.env
                -interface.py
                -requirements.txt
```

Getting started

Set-up and installation

1. Clone the repository from Github.
   a. git clone https://github.com/ettany/FOH_PSE
2. Install dependencies.
   a. pip install -r requirements.txt
3. Run the backend.
   a. export FLASK_APP=backend/app
   b. flask init-db
   c. flask --app backend/server.py run
4. Run the front end.
   a. python frontend/interface.py

Configuration

Environment variables

.env Backend

```
SECRET_KEY=●●●●●●●

DATABASE_URL=●●●●●●●

ALPHA_VANTAGE_API_KEY=HYZ6LXZ02KV3N6VH

FLASK_APP=development

UI_URL=http://127.0.0.1:5001

RECAPTCHA_SECRET_KEY=●●●●●●●
```

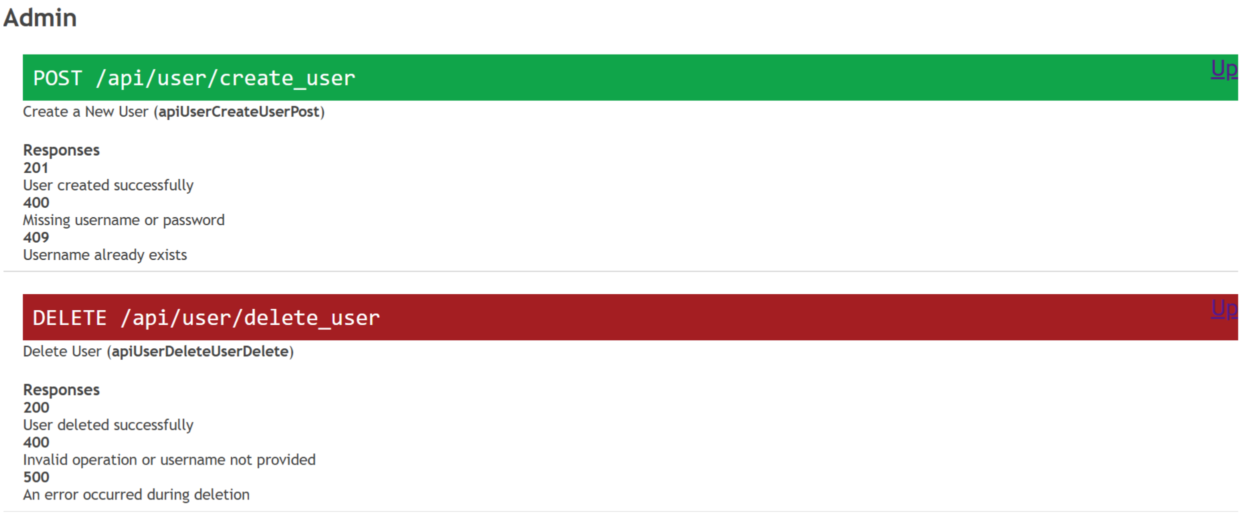
These variables should be stored in an environment file in the front end of the application.

.env Frontend

```
BACKEND_URL = 'http://localhost:5000'
```

These variables should be stored in an environment file in the back end of the application.

API documentation



Admin

`POST /api/user/create_user`

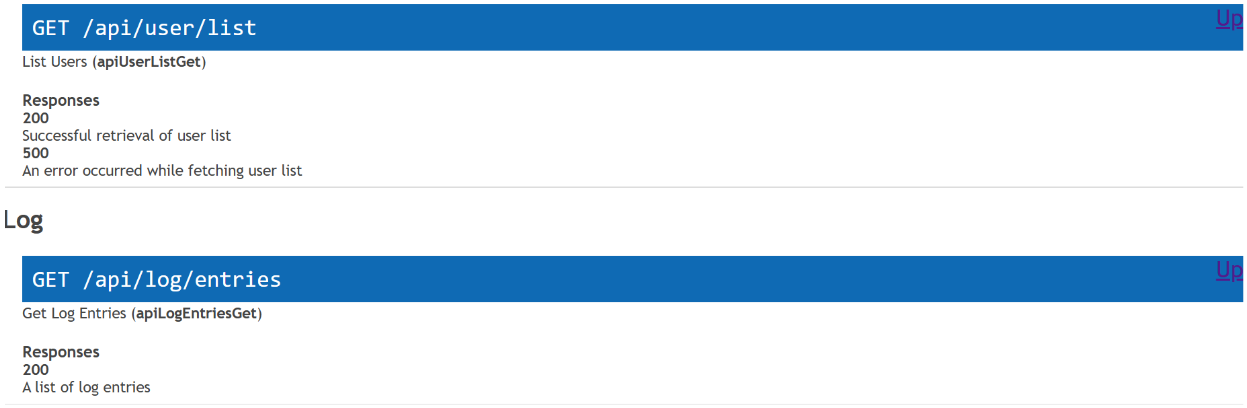
Create a New User (**apiUserCreateUserPost**)

Responses
201
User created successfully
400
Missing username or password
409
Username already exists

`DELETE /api/user/delete_user`

Delete User (**apiUserDeleteUserDelete**)

Responses
200
User deleted successfully
400
Invalid operation or username not provided
500
An error occurred during deletion



`GET /api/user/list`

List Users (**apiUserListGet**)

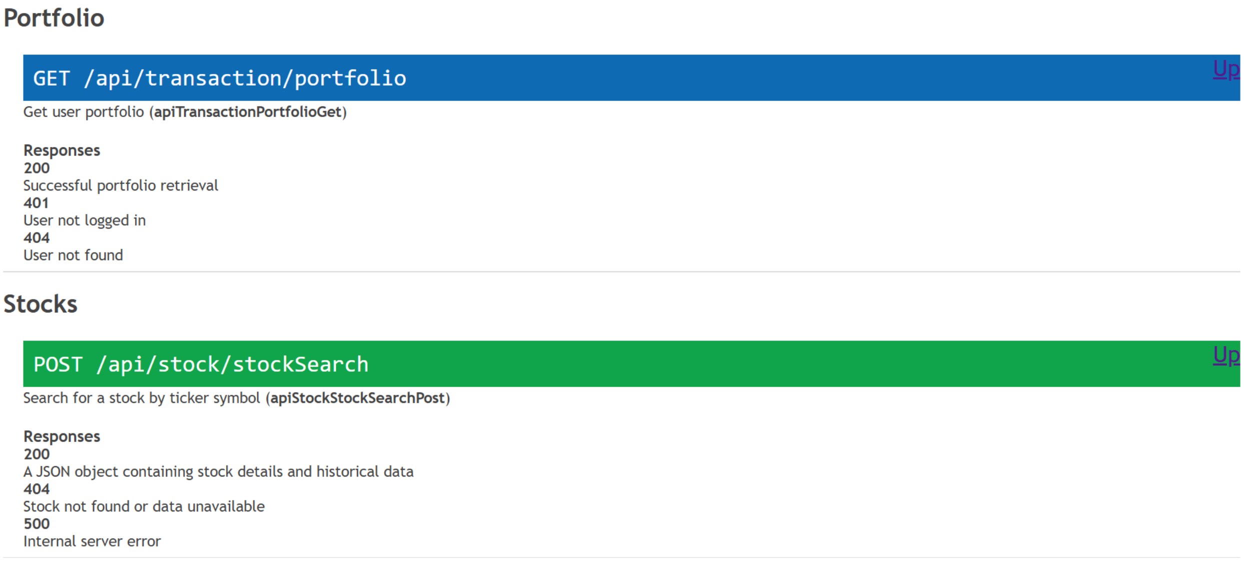
Responses
200
Successful retrieval of user list
500
An error occurred while fetching user list

Log

`GET /api/log/entries`

Get Log Entries (**apiLogEntriesGet**)

Responses
200
A list of log entries



Portfolio

`GET /api/transaction/portfolio`

Get user portfolio (**apiTransactionPortfolioGet**)

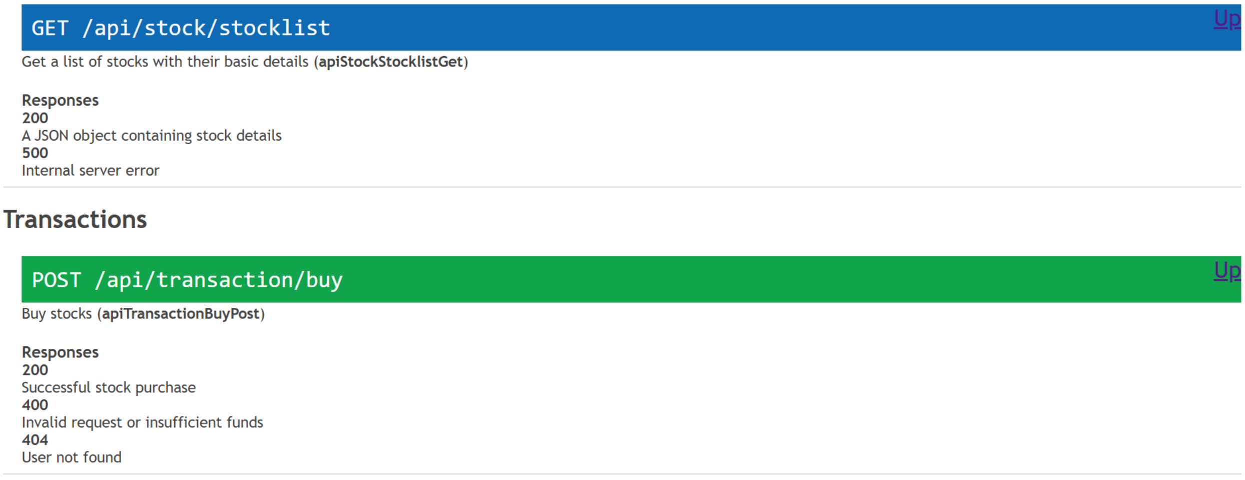
Responses
200
Successful portfolio retrieval
401
User not logged in
404
User not found

Stocks

`POST /api/stock/stockSearch`

Search for a stock by ticker symbol (**apiStockStockSearchPost**)

Responses
200
A JSON object containing stock details and historical data
404
Stock not found or data unavailable
500
Internal server error



GET /api/stock/stocklist

Get a list of stocks with their basic details (**apiStockStocklistGet**)

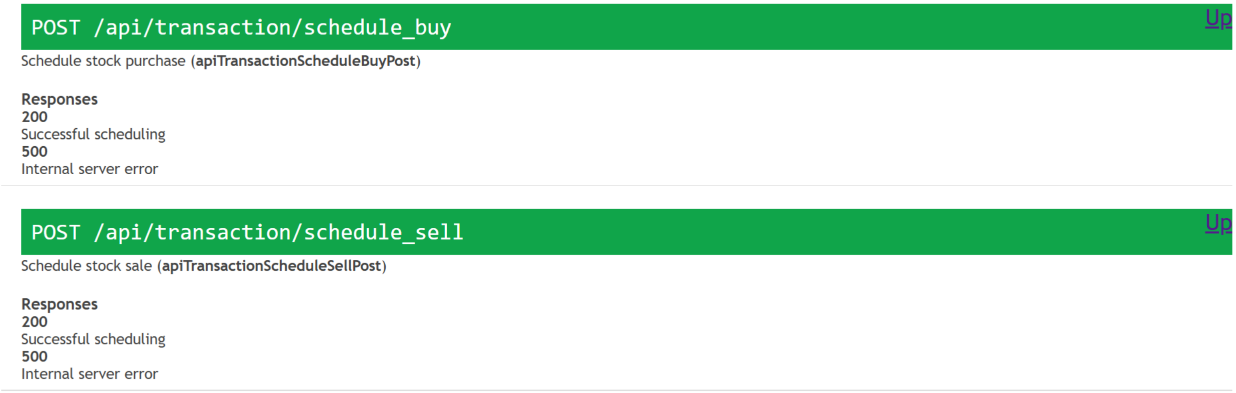
Responses
200
A JSON object containing stock details
500
Internal server error

Transactions

POST /api/transaction/buy

Buy stocks (**apiTransactionBuyPost**)

Responses
200
Successful stock purchase
400
Invalid request or insufficient funds
404
User not found



POST /api/transaction/schedule_buy

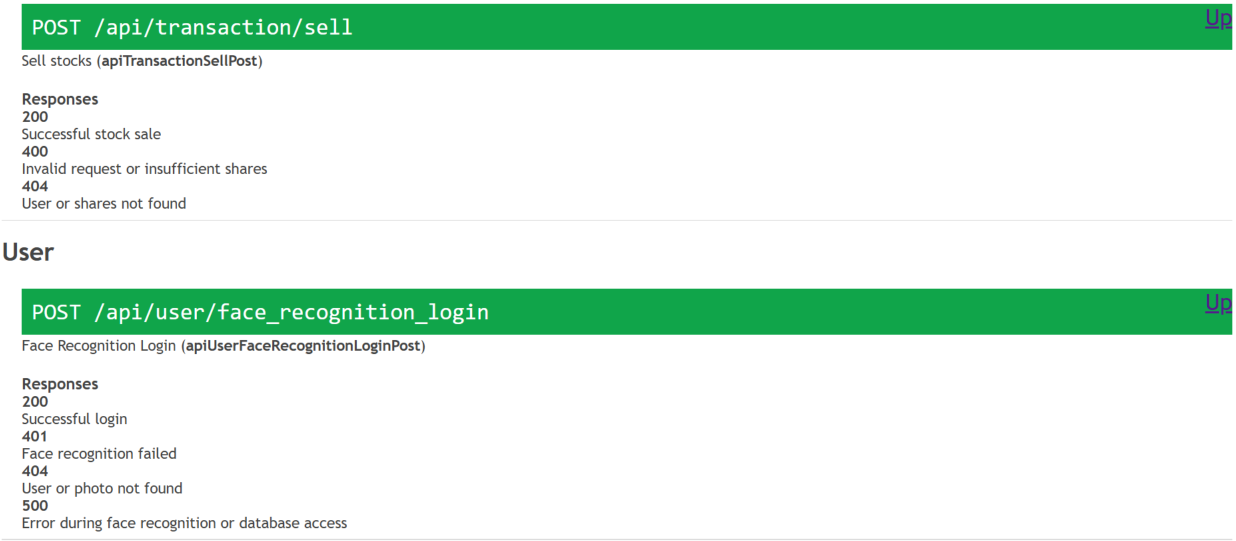
Schedule stock purchase (**apiTransactionScheduleBuyPost**)

Responses
200
Successful scheduling
500
Internal server error

POST /api/transaction/schedule_sell

Schedule stock sale (**apiTransactionScheduleSellPost**)

Responses
200
Successful scheduling
500
Internal server error



POST /api/transaction/sell

Sell stocks (**apiTransactionSellPost**)

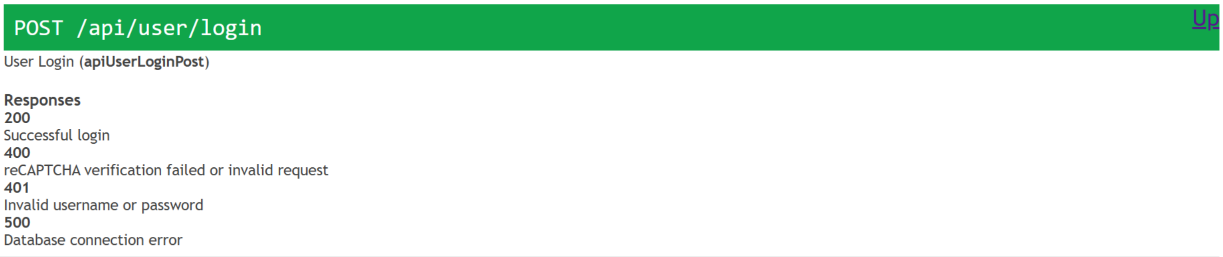
Responses
200
Successful stock sale
400
Invalid request or insufficient shares
404
User or shares not found

User

POST /api/user/face_recognition_login

Face Recognition Login (**apiUserFaceRecognitionLoginPost**)

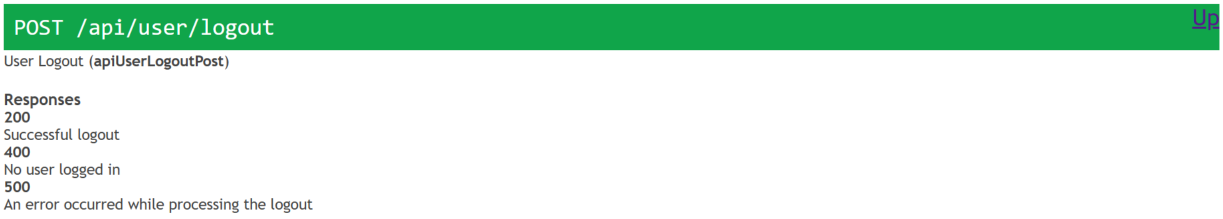
Responses
200
Successful login
401
Face recognition failed
404
User or photo not found
500
Error during face recognition or database access



POST /api/user/login Up

User Login (**apiUserLoginPost**)

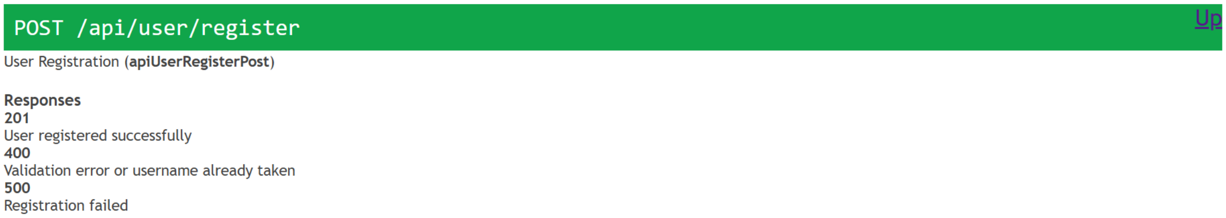
Responses
200
Successful login
400
reCAPTCHA verification failed or invalid request
401
Invalid username or password
500
Database connection error



POST /api/user/logout Up

User Logout (**apiUserLogoutPost**)

Responses
200
Successful logout
400
No user logged in
500
An error occurred while processing the logout



POST /api/user/register Up

User Registration (**apiUserRegisterPost**)

Responses
201
User registered successfully
400
Validation error or username already taken
500
Registration failed

Frontend

1. *Templates*

Description of which controllers and functions each page uses:

register.html

user_controller.py:

- register

login.html

user_controller.py (face_verification.py)

- login
- face_recognition_login

admin.html

user_controller.py:

- create_user
- delete_user
- list

index.html

chartUtils.js:

- drawScatterPlotChart

transaction_controller.py:

- portfolio

user_controller.py:

- index

trade.html

chartUtils.js:

- drawBarChart
- drawLineChart

stock_controller.py:

- stocklist
- StockSearch

transaction_controller.py:

- buy
- sell
- schedule_buy
- schedule_sell

log.html

log_controller.py:

- entries

logout.html

2. *Static files*

css – main styling (in addition to each page having a separate style section for internal css)

utils – chart functions and cookies (JavaScript)

3. *UI*

See user guide and demo video for more details:

User Guide: https://github.com/ettany/FOH_PSE/blob/main/UserGuide_FOH.pdf

Demo Video: https://www.youtube.com/watch?v=7XaVBbRt8u4