User Manual

Owlgorithmic Traders

Table of Contents

Part I: User Guide

Owlgorithmic Traders Financial Disclosure Viewer

A web application that allows users to download U.S. Congressional financial disclosures by year (2021–2025) via a simple GUI. Will then provide raw data as well as visual representation via graphs.

Project Source Code

https://github.com/rxzvdx/Owlgorithmic-Traders.git

The process is as follows:

1. Log in with Google (via OAuth)
2. Option to download U.S. House of Representatives disclosures by year (2021-2025)
3. View raw metadata (txt/xml/zip)
4. Explore real-time stock charts with company overviews
5. Filter and browse disclosures in interactive dashboard
6. Contact support via in-app form (entries → Google Sheets)

How to Use

1. Start the App:
   a. "**python app.py**" in terminal OR
   b. flask run
   c. Go to: http://localhost:5000 (or server IP if deployed on AWS)
2. Log in/Sign up
   a. Click "**log in with Google**"
   b. Grant permission for email and profile
   c. On success you'll see a green flash "Successfully signed in as…"
3. Download Raw Disclosures (optional)
   a. Click **Download** under **Select a Year** (2021–2025).

   b. A ZIP ({year}.zip) and TXT ({year}FD.txt) will download to your browser.
4. View Live Charts
   a. Click **View Live Charts** in the navbar
   b. Select any company or index button to see an embedded TradingView chart plus a company overview
5. Dashboard (interactive table)
   a. Open **Dashboard** via nav bar
   b. Use the Name, State, Transaction Type, and Year filters
6. Contact and Support
   a. Navigate to Contact in the nav bar
   b. Submit an inquiry

Where Files Go

Artifact	**Delivered To:**
{year}.zip	Browser download
{year}FD.txt	Browser download
/raw_data/{year}_data/*.xml	Server-side extracted XML disclosures
/house/{Last_First}/{year}/{DocID}.pdf	Per-Rep. PDF folder on server
/term_logs	download_log.txt & failed_downloads.txt

Troubleshooting

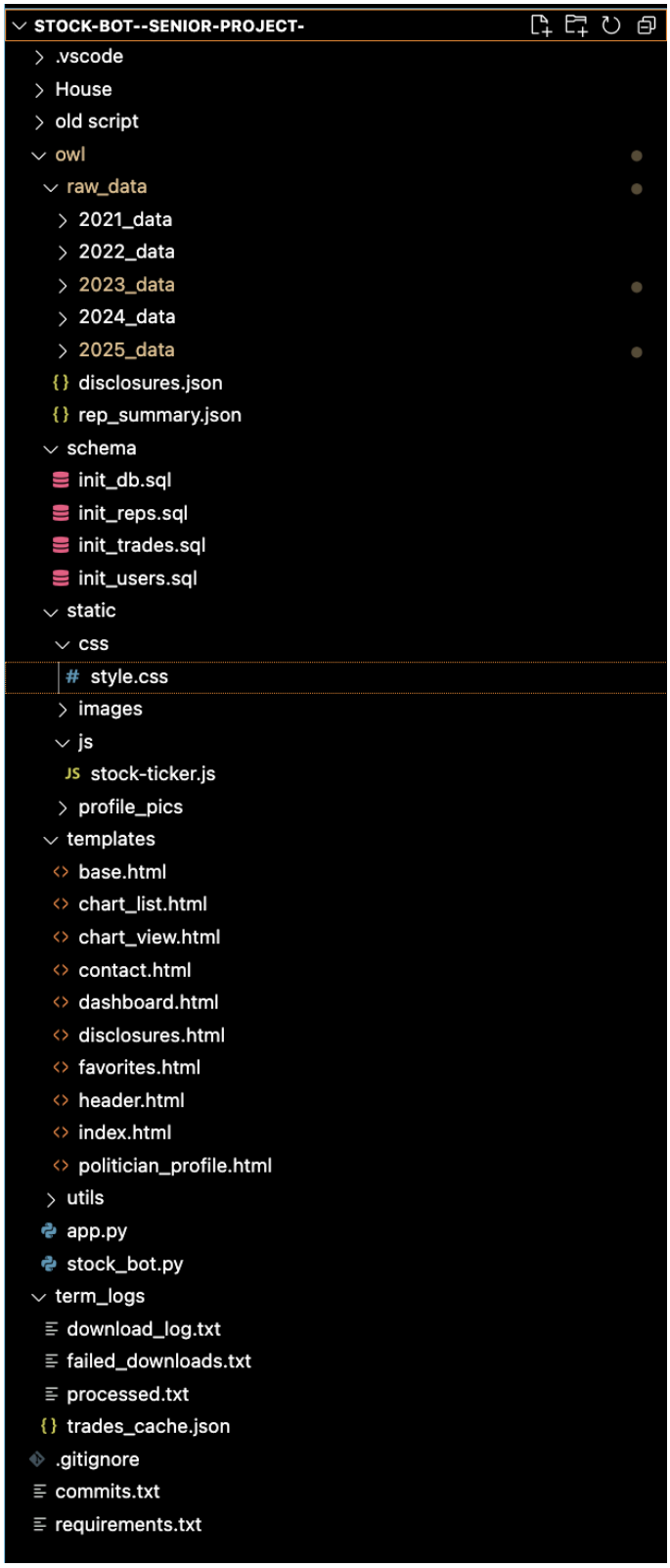
Issue	Solution
“Download failed” flash	Check internet connection or year availability. Verify year availability. Ensure you’re logged in.
“Session expired” during Google OAuth	Clear cookies, re-login; make sure OAUTHLIB_INSECURE_TRANSPORT=1 in dev.
Charts won’t load (TradingView)	Confirm secure HTTP(S); check console for widget errors.
Dashboard filters not responding	Ensure JS is loading (static/js/); check browser console.
App does not start	Run **pip install -r requirements.txt**, then **python app.py**

Part II: Developer Guide

Tech Stack & Dependencies

- Python 3.13
- Flask + Flask-Dance (Google OAuth)
- Requests, zipfile, csv, xml.etree
- MySQL Connector (or SQLAlchemy)
- PyMuPDF & PyPDF2 (PDF parsing)
- finance (stock API)
- Pandas (optional dataframes)
- Plyer (desktop notifications)
- oauthlib, Werkzeug, Jinja2, etc.

Project Structure



```
Owlgorithmic Traders/
├── .vscode/
│ └── … (VS Code settings)
├── house/                              ← Per-representative PDF folders
├── old script/                         ← Archive of legacy code
├── owl/                                ← Main application package
│ ├── plans/                            ← Project notes & roadmaps
│ ├── raw_data/                         ← Extracted metadata & XML disclosures
│ ├── schema/
│ │ ├── init_db.sql                     ← Create database
│ │ ├── init_reps.sql                   ← Create representatives table
│ │ ├── init_trades.sql                 ← Create trades table
│ │ └── init_users.sql                  ← Create users table
│ ├── static/
│ │ ├── css/
│ │ │ └── style.css
│ │ ├── images/                         ← Logos & icons
│ │ ├── js/
│ │ │ └── stock-ticker.js
│ │ └── profile_pics/                   ← Legislator avatars
│ ├── templates/
│ │ ├── base.html
│ │ ├── chart_list.html
│ │ ├── chart_view.html
│ │ ├── contact.html
│ │ ├── dashboard.html
│ │ ├── disclosures.html
│ │ ├── favorites.html
│ │ ├── header.html
│ │ ├── index.html
│ │ └── politician_profile.html
│ ├── utils/
│ │ ├── pycache/
│ │ ├── desktop_notifs.py               ← Desktop notification helper
│ │ ├── download_politician_images.py   ← Bulk portrait fetcher
│ │ ├── downloader.py                   ← ZIP/TXT downloader & extractor
│ │ ├── fetch.py                        ← Catalog downloaded PDFs
│ │ ├── load_reps.py                    ← Populate representatives table
│ │ ├── load_trades.py                  ← Populate trades table
│ │ ├── login.py                        ← Google OAuth blueprint
│ │ ├── pdf_parser.py                   ← Extract trade data from PDFs
│ │ └── process_data.py                 ← Clean & transform disclosures
│ ├── app.py ← Flask application entry point
│ └── stock_bot.py                      ← (Legacy launcher script)
├── term_logs/
│ ├── download_log.txt                  ← Download activity log
│ ├── failed_downloads.txt              ← Download error log
│ └── processed.txt                     ← Pre-parsed CSV tracker
├── trades_cache.json                   ← Cached trade/price data
├── .gitignore
└── requirements.txt                    ← Python dependencies
```

Setup Instructions

Bash

- git clone https://github.com/rxzvdx/Owlgorithmic-Traders.git
- cd owl
- python3 -m venv venv && source venv/bin/activate
- pip install -r requirements.txt
- export OAUTHLIB_INSECURE_TRANSPORT=1
- export FLASK_APP=app.py
- flask run
- # or
- python app.py

AWS EC2 UBUNTU

- sudo apt update && sudo apt install python3-pip python3-venv -y
- python3 -m venv venv && source venv/bin/activate
- pip install -r requirements.txt
- ufw allow 5000
- flask run --host 0.0.0.0

How It Works

Green	🟩	= directory
Blue	🟦	= file
Purple	🟪	= database table
Red	🟥	= flask route

1. **Database Setup**
   a. Run the SQL scripts in **owl/schema** to create the db and tables:
   b. **init_db.sql**
   c. **init_reps.sql**
   d. **init_trades.sql**
   e. **init_users.sql**
2. **Data Ingestion (Command-line Utilities)**
   a. Download & extract disclosures via **utils/downloader.py**
      i. Downloads **{year}FD.zip** → extracts **{year}FD.txt** into **owl/raw_data**
   b. Populate Representatives via **utils/load_reps.py**:
      i. Reads **{year}FD.txt** and inserts into the **representatives** table
   c. Populate Trades via **utils/load_trades.py**
      i. Walks the **house/** PDF directories, extracts trades with **utils/pdf_parser.py** and inserts into the **trades** table
3. **App Startup**
   a. **owl/app.py** initializes Flask, sets up Google OAuth (**utils/login.py**), and desktop notifications (**utils/desktop_notifs.py**).
4. **HTTP Routes & Logic**
   a. **GET /** → **index.html**
      i. Shows login button or "Select a Year" download UI.
   b. **GET /auth** & **OAuth callback** (Flask-Dance)
      i. Handles Google sign-in and flashes success/error messages.
   c. **POST /download**
      i. Downloads ZIP via downloader util & returns as attachment.
   d. **GET /api/stock/<symbol>**

    i. Returns JSON {price, change} from yfinance.
  e. **GET /chart_view** → **chart_list.html**
    i. lists top 50 stocks.
  f. **GET /chart_view/<symbol>** → **chart_view.html**
    i. embeds TradingView widget + overview.
  g. **GET /contact** → **contact.html**
    i. form posts to Google Sheets.
  h. **GET /dashboard** → **dashboard.html**
    i. dynamically filters XML under **owl/raw_data/**.
  i. **GET /api/disclosures**
    i. Returns JSON array of parsed XML disclosures.
  j. **POST /create_plan**
    i. Reads opt-in checkbox; fires desktop notification if enabled.

5. **Static Assets & Templates**
  a. CSS: **owl/static/css/style.css**
  b. JS Ticker: **owl/static/js/stock_ticker.js**
  c. Templates: **owl/templates** (HTML files)
6. **Logs & Downloads**
  a. Logs: **term_logs/download_log.txt** & **term_logs/failed_downloads.txt** track CLI downloads
  b. Raw metadata & XML: **owl/raw_data/{year}_data/*.xml**
  c. PDFs: Organized under **house/{Last_First}/{year}/{DocID}.pdf**