ChatGPT

Inspire Ecom Master - Trending Product Tracker

Inspire Ecom Master is a production-ready application that tracks trending products across multiple platforms and provides a unified dashboard. It features a FastAPI backend with asynchronous "trend bot" tasks for each platform, an optional Shopify integration to sync products, a visually styled dashboard, and packaging for both Docker deployment and a standalone desktop application.

Features

- **FastAPI Backend**: A high-performance Python backend using FastAPI serves both API endpoints and a web-based dashboard. It organizes code into an `app` package with clear separation of concerns.
- **Trend Bots for Multiple Platforms**: Asynchronous background tasks ("bots") fetch trending product data from Amazon, Shopify, TikTok Shop, and Walmart at regular intervals. In this demo, each bot simulates trending items (e.g. popular items like the Stanley Quencher tumbler [1] or barrel fit jeans [2]) to mimic real trends. The architecture supports replacing these with real API calls or web scrapers in the future.
- **Shopify Sync Integration**: Optionally syncs trending products to your Shopify store. If you provide a Shopify store domain and API access token in the environment (`.env` file), the app will create new products on your Shopify store for each new trending item discovered from external sources (Amazon, TikTok, Walmart). This integration is disabled if no credentials are provided. (The code uses Shopify's Admin API – a new product is created by simply supplying a title via the REST endpoint [3].)
- **Visual Dashboard**: A front-end dashboard is served at the root URL (`/`) to display the latest trends and bot status. It is styled with a custom CSS and includes your provided visual assets (logo and background pattern). The dashboard shows each platform's trending products, last update time, and status (e.g. success or failure). Shopify integration status (enabled/disabled and sync results) is also displayed at the top. The UI uses the `logo_inspire.png` and `pattern_inspire.png` visuals for branding.
- **Docker Deployment**: A Dockerfile is included to containerize the application. The Docker image uses Uvicorn to run the FastAPI app, making deployment straightforward. An example environment file (`.env.example`) is provided for configuration.
- **Desktop Application Packaging**: For convenience, the project can be compiled into a standalone Windows executable. A launcher script (`launch_app.py`) starts the FastAPI server and automatically opens the dashboard in the user's default web browser. A Windows batch script (`build_windows.bat`) uses PyInstaller to produce a single-file `.exe` so non-technical users can run the dashboard without setting up a Python environment.

Project Structure

```
inspire_ecom_master/
├── app/
│   ├── __init__.py           # Package initializer
```

```
│   ├── main.py               # FastAPI application instance and routes
│   ├── bots.py               # Trend bot logic for fetching trends and syncing
to Shopify
│   ├── templates/
│   │   └── dashboard.html    # Jinja2 template for the dashboard UI
│   └── static/
│       ├── css/
│       │   └── style.css     # CSS styling for the dashboard
│       └── images/
│           ├── logo_inspire.png       # Logo image used in the UI (custom
provided asset)
│           └── pattern_inspire.png    # Background pattern image for UI styling
(custom provided asset)
├── launch_app.py             # Desktop launch script to start server and open
the browser
├── Dockerfile                # Dockerfile for containerizing the application
├── .env.example              # Example environment variables for configuration
├── build_windows.bat         # Windows script to build the standalone
executable
├── requirements.txt          # Python dependencies
└── README.md                 # Project documentation (this file)
```

Setup and Installation

Prerequisites

- **Python 3.10+** (for running from source or building the exe)
- **Pip** for installing dependencies
- **Node.js** is **not required** (the dashboard is server-rendered, no separate Node frontend)
- *(Optional)* **Docker** if you plan to use the Docker container
- *(Optional)* **PyInstaller** (will be installed via the build script if building the Windows exe)

Installation Steps

1. **Clone or Extract the Project**: Obtain the project source code. If you received a ZIP archive (`InspireEcomMaster.zip`), extract it. Ensure the custom images `logo_inspire.png` and `pattern_inspire.png` are placed in the `app/static/images` directory (they should be included if provided).
2. **Python Environment**: It is recommended to create a virtual environment:

   ```
   python -m venv venv
   source venv/bin/activate   # on Windows: venv\Scripts\activate
   ```

3. **Install Dependencies**: Use pip to install required packages:

```
pip install -r requirements.txt
```

This will install FastAPI, Uvicorn, Jinja2, httpx, python-dotenv, aiofiles, etc.

4. **Configuration**: Copy `.env.example` to `.env` (or create a new `.env`) if you plan to enable Shopify integration. Then edit the `.env` file:

5. `SHOPIFY_STORE`: Your Shopify store's domain (e.g. `mystore.myshopify.com`).
6. `SHOPIFY_TOKEN`: A Shopify Admin API access token for that store. (If you don't have one, you can create a private app or custom app in Shopify to get an API key/password with permissions to create products.)

If you leave these blank or do not create a `.env`, the Shopify sync features will remain inactive (which is fine for running the dashboard with mock data).

Running the Application (Development/Testing)

You can run the app either directly via Python or using Docker:

Running with Python (FastAPI/Uvicorn)

After installing dependencies and configuring the environment variables as needed, you have two options:

- **Option 1: Using Uvicorn CLI** – Start the FastAPI server directly:

```
uvicorn app.main:app --reload --host 127.0.0.1 --port 8000
```

This will serve the app at http://127.0.0.1:8000. The `--reload` flag will auto-reload the server on code changes (useful during development).

- **Option 2: Using the Launcher Script** – Run the provided launch script which also opens your web browser:

```
python launch_app.py
```

This will start the Uvicorn server in the background and automatically open the dashboard in your default browser. The console will show server logs. Press `Ctrl+C` in the terminal to stop the server. (If running on Windows by double-clicking the script, a console window will appear; closing that window will stop the server.)

Once running, open your browser to `http://127.0.0.1:8000` (if it didn't open automatically). You should see the **Inspire Ecom Master Dashboard** with sections for Amazon, Shopify, TikTok, and Walmart trends, as well as a Shopify integration status panel at the top.

Running with Docker

If you prefer to use Docker (e.g., for deployment or to avoid installing Python locally):

1. Build the Docker image:

   ```
   docker build -t inspire-ecom .
   ```

   This will create a Docker image named "inspire-ecom".

2. Run a container from the image:

   ```
   docker run -d -p 8000:8000 --name inspire_app --env-file .env inspire-ecom
   ```

3. The app listens on port 8000 inside the container, so we publish it to port 8000 on the host.
4. The `--env-file .env` option passes your configuration (Shopify credentials) into the container. Alternatively, you can use `-e SHOPIFY_STORE=... -e SHOPIFY_TOKEN=...` to set them.
5. The container will run in detached mode (`-d`). You can remove `-d` to run it in the foreground to see logs.

After the container starts, visit http://localhost:8000 in your browser to access the dashboard. (If deploying to a remote server, use that server's address or set up appropriate port forwarding.)

Dashboard and API Usage

- **Dashboard Interface**: The dashboard displays each platform's trending products in a separate section. For each source (Amazon, Shopify, TikTok, Walmart) you will see:
- A heading (e.g., "Amazon Trends").
- The last update timestamp and status of the bot. If a fetch failed, it will display an error message and the time of failure (in red). If it's still fetching for the first time, it shows "Fetching data...".
- A list of trending products (product name and price). These are sample data points; in a real scenario, they would be populated by actual trending items fetched via APIs or scraping. For instance, Amazon trends might include popular electronics or home gadgets, TikTok Shop might show viral products like LED lights or beauty tools, etc.
- The list updates automatically in the background every few minutes. To see the latest data, use the "Refresh" button at the bottom of the dashboard or simply reload the page.
- **Shopify Integration Status**: At the top, the dashboard indicates whether Shopify integration is enabled.
- If **enabled**, it shows the number of products that have been synced to your Shopify store so far, the last sync time, and whether the last sync encountered any error. If an error occurred (e.g., invalid credentials), it will be highlighted in red.
- If **disabled** (no credentials provided), it simply states that the integration is disabled.
- **API Endpoints**: This project also exposes a few REST API endpoints for programmatic access:

- `GET /api/trends` – Retrieve the latest trending products from all sources in JSON format. The response includes keys for each source (amazon, shopify, tiktok, walmart), each containing a list of product items.
- `GET /api/trends/{source}` – Retrieve trending products for a specific source (e.g., `/api/trends/amazon` or `/api/trends/tiktok`). The response will include the source name and its list of trends, or a 404 error if the source name is invalid.
- `GET /api/status` – Get the current status of all trend bots and the Shopify sync. It returns a JSON with `trends` (containing status info for each source's bot: last run time, status, and error if any) and `shopify_sync` (with last run time, status, error, and count of products synced).

These endpoints can be used to integrate the trending data with other systems or front-ends if needed.

Building the Desktop Application (Windows)

If you want to compile the project into a standalone Windows executable (so that end users can run the app without installing Python):

1. Ensure you're on a Windows environment with Python and pip available. All project files should be in place (especially include the `app/static` and `app/templates` folders and their contents, as the build will package these).
2. Edit `build_windows.bat` if you want to customize build options (for example, adding an icon or changing the output name). By default, it will create a one-file executable using PyInstaller.
3. Run the build script:

   ```
   build_windows.bat
   ```

   This will install PyInstaller (if not already installed) and then bundle the application. The static files and templates are included in the bundle via the `--add-data` flags in the script. After a successful build, you should find an `dist\launch_app.exe` file.
4. Distribute the `launch_app.exe` to users along with the `logo_inspire.png` and `pattern_inspire.png` files in case they are needed (if the visuals are properly packaged inside the exe, users will see the styled UI without needing separate image files). The user can double-click the exe to run the application.

Note: The executable when run will open a console window and launch the browser. The console window is necessary to host the server (and to allow easy termination of the app by closing the window). If you prefer to hide the console window, you could modify the PyInstaller command to use `--noconsole`, but then you'd need another mechanism to stop the server (since it would run in the background).

Extending and Customization

- **Using Real Trend Data**: Currently, the trend bots use mock data for demonstration. To integrate real data:
- *Amazon*: You might use the Amazon Advertising API or scrape the "Best Sellers" pages to get current top-selling products.

- *TikTok Shop*: If available, use TikTok's APIs or third-party services to fetch trending items. Alternatively, this could be replaced with trending social media product data.
- *Walmart*: Use Walmart's Open API or scrape their trending/best-seller listings.
- *Shopify Trends*: If you want to show trending products from your own Shopify store, you could query your store's order history to find best-selling products. (This project currently uses Shopify integration only for syncing products, not for fetching trending data from Shopify.)

You can modify the asynchronous functions in `app/bots.py` to call real APIs instead of using the dummy `_dummy_pools`. Make sure to handle authentication and rate limiting as needed, and update the data parsing logic to fit the API responses.

- **Data Persistence**: By default, trending data and sync status are stored in-memory (and reset on each app restart). For a production system, you might integrate a database to persist trending history or use an external cache.

- **UI Enhancements**: The current dashboard is minimalistic and uses server-side rendering. You can enhance it with more interactive features (e.g., auto-refresh via AJAX, charts for trends over time, etc.). Since FastAPI can serve JSON, it would be straightforward to build a more dynamic front-end (for example, using a JavaScript framework or just vanilla JS to periodically fetch `/api/trends`).

Conclusion

The Inspire Ecom Master project brings together trending product tracking and e-commerce integration in one package. With a clean architecture, async capabilities, and multiple deployment options (API service, web app, or desktop app), it is designed to be flexible and easy to use. Whether used as a personal dashboard for product research or as a foundation to build a more advanced analytics tool, the project demonstrates how FastAPI can power a full-stack application integrating third-party data and services. [1] [2] [3]

---

[1] 18 high-demand and trending products to sell online in 2025 | Sell on Amazon - Sell on Amazon
https://sell.amazon.com/blog/products-to-sell

[2] 20 Trending Products and Things To Sell Online (2026) - Shopify
https://www.shopify.com/blog/trending-products

[3] REST Admin API reference
https://shopify.dev/docs/api/admin-rest