Amazon Price Tracker

A Flask-based web application that tracks Amazon product prices and notifies users when prices drop below their target thresholds.

Features

- **Product URL Tracking**: Add Amazon product URLs to monitor
- **Price Monitoring**: Automatically scrape and track price changes
- **Target Price Alerts**: Set desired price thresholds for notifications
- **Price History**: View historical price data and trends
- **Responsive Design**: Mobile-friendly Bootstrap interface
- **Real-time Updates**: Periodic background scraping with APScheduler

Technology Stack

- **Backend**: Flask, SQLAlchemy
- **Frontend**: HTML5, CSS3, Bootstrap 5
- **Database**: SQLite
- **Web Scraping**: BeautifulSoup4, Requests
- **Scheduling**: APScheduler
- **Deployment**: Gunicorn, Render-ready

Project Structure

```
amazon-price-tracker/
├── app.py                 # Main Flask application
├── models.py             # Database models
├── scraper.py            # Web scraping logic
├── scheduler.py          # Background task scheduler
├── requirements.txt      # Python dependencies
├── Procfile             # Render deployment config
├── README.md            # Project documentation
├── static/
│   └── style.css        # Custom stylesheets
├── templates/
│   ├── base.html        # Base template
│   ├── index.html       # Homepage
│   ├── track_products.html # Add product form
│   ├── all_products.html   # Product listing
│   ├── 404.html         # Error pages
│   └── 500.html
└── instance/
    └── app.db           # SQLite database (auto-generated)
```

Setup Instructions

Prerequisites

- Python 3.8 or higher
- pip package manager
- Git (for cloning)

Local Development

1. **Clone the repository**

   ```
   git clone <repository-url>
   cd amazon-price-tracker
   ```

2. **Create virtual environment**

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

3. **Install dependencies**

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

4. **Set environment variables** (optional)

   ```
   export FLASK_APP=app.py
   export FLASK_ENV=development
   ```

5. **Initialize database**

   ```
   python -c "from app import app, db; app.app_context().push(); db.create_all()"
   ```

6. **Run the application**

   ```
   python app.py
   ```

7. **Access the application**

   - Open your browser to `http://localhost:5000`

Running with Scheduler

To enable automatic price monitoring:

```
python scheduler.py
```

This will start the background scheduler that periodically scrapes product prices.

Deployment

Render Deployment

1. **Fork/Clone to GitHub**
   - Push your code to a GitHub repository
2. **Create Render Account**
   - Sign up at render.com
3. **Create New Web Service**
   - Connect your GitHub repository
   - Use the following settings:

        - **Build Command**: `pip install -r requirements.txt`
        - **Start Command**: `gunicorn app:app`
        - **Environment**: Python 3

4. **Environment Variables**
    - Set `PYTHON_VERSION` to `3.8.10` or higher
    - Add any custom environment variables if needed

5. **Deploy**
    - Render will automatically build and deploy your application

Heroku Deployment

1. **Install Heroku CLI**

    ```
    # Install Heroku CLI from https://devcenter.heroku.com/articles/heroku-cli
    ```

2. **Login and Create App**

    ```
    heroku login
    heroku create your-app-name
    ```

3. **Set Environment Variables**

    ```
    heroku config:set FLASK_APP=app.py
    ```

4. **Deploy**

    ```
    git push heroku main
    ```

Usage

Adding Products

1. Navigate to "Track Products" page
2. Enter the Amazon product URL
3. Set your target price threshold
4. Click "Start Tracking"

Viewing Products

- **Homepage**: Shows recently added products
- **All Products**: Complete list with current prices
- **Price History**: Track price changes over time

Product URL Format

Ensure Amazon URLs are in the correct format:

```
https://www.amazon.com/dp/PRODUCT_ID
https://www.amazon.com/PRODUCT-NAME/dp/PRODUCT_ID
```

Configuration

Database Configuration

The app uses SQLite by default. To use a different database:

1. Update the `SQLALCHEMY_DATABASE_URI` in `app.py`
2. Install appropriate database drivers
3. Update `requirements.txt` accordingly

Scraping Configuration

Modify scraping settings in `scraper.py`:

- **Headers**: Customize user agent strings
- **Delays**: Adjust request intervals
- **Selectors**: Update CSS selectors if Amazon changes their layout

Scheduler Configuration

Adjust monitoring frequency in `scheduler.py`:

```
scheduler.add_job(
    func=scrape_all_products,
    trigger="interval",
    hours=1,  # Change interval as needed
    id='price_scraper'
)
```

Troubleshooting

Common Issues

1. **403 Forbidden Errors**
   - Amazon may block requests; try different user agents
   - Add delays between requests
   - Use proxy servers if necessary
2. **Database Errors**
   - Ensure database is properly initialized
   - Check file permissions for SQLite
3. **Scraping Failures**
   - Verify Amazon URL format
   - Check if product page layout changed
   - Update CSS selectors in scraper

Debugging

Enable debug mode for development:

```
app.run(debug=True)
```

Check logs for error details:

```
heroku logs --tail  # For Heroku
```

Contributing

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Add tests if applicable
5. Submit a pull request

License

This project is for educational purposes. Please comply with Amazon's Terms of Service and robots.txt when scraping.

Disclaimer

This tool is for personal use only. Users are responsible for complying with Amazon's Terms of Service and any applicable laws regarding web scraping.

Support

For issues and questions:

1. Check the troubleshooting section
2. Review error logs
3. Create an issue in the repository

Roadmap

- [ ] Email notifications for price drops
- [ ] Multiple retailer support
- [ ] Advanced price analytics
- [ ] User authentication
- [ ] Mobile app
- [ ] API endpoints

Note: The SQLite database (`instance/app.db`) is automatically created when you first run the application. This file will contain all your tracked products and price history.