# First Project Overview

## Phased Roadmap

### Stage 1 – MVP Web Scraper (Your Immediate Need)

* **Goal:** Automate daily collection of price data for two brands.
* **Tools:** Selenium, pandas, datetime, csv/json storage
* **Milestones:**
  1. Setup project structure & venv.
  2. Build Selenium scraper.
  3. Store data with timestamp.
  4. Validate data accuracy.
* **Testing:** Run manually, validate stored data matches page.

### Stage 2 – Automated Scheduling

* **Goal:** Run daily without manual action.
* **Tools:**
  + Local: cron (Linux/Mac), Task Scheduler (Windows)
  + Cloud: AWS Lambda + EventBridge or any server you have.
* **Milestones:**
  + Wrap scraper in a script that logs daily.
  + Setup scheduling on your machine or cloud.
  + Confirm daily data is added.

### Stage 3 – Data Visualization Dashboard

* **Goal:** View price history, trends, highs/lows.
* **Tools:**
  + matplotlib, pandas, plotly, or
  + Web app with Streamlit or Dash
* **Milestones:**
  + Load historical data.
  + Build plots showing price trends.
  + Add simple filters for brands/dates.

### Stage 4 – Price Alert Notifications

* **Goal:** Get notified when price drops below your target.
* **Tools:**
  + smtplib (email), twilio (SMS/WhatsApp), or Pushbullet
* **Milestones:**
  + Define target price threshold.
  + Check latest price after scraping.
  + Send notification if condition met.

### Stage 5 – Predictive Price Analysis (Stretch Goal)

* **Goal:** Estimate future low-price periods using ML.
* **Tools:**
  + scikit-learn, pandas, numpy, matplotlib
* **Milestones:**
  + Gather enough historical data (1-3 months min).
  + Clean and explore dataset.
  + Build regression or time series models.
  + Evaluate and improve predictions.

## General Recommendations

* Use **venv** from day one for clean dependency management.
* Save data in **CSV** or **JSON** for easy integration later.
* Keep your **scraper modular** so you can expand functionality easily.
* **Document** everything as you go.
* Commit to **GitHub** to make it a portfolio project.

# What Else Can We Throw In?

* **Docker** (Containerize scraper & dashboard for local/cloud portability).
* **AWS Free Tier**
  + **Lambda** for scheduled scraping
  + **S3** for storing CSV/JSON data
  + **EventBridge** for triggers
* **Streamlit** or **Dash** (Free dashboards, no need for PowerBI licenses).
* **Github Actions** for CI/CD or automated tests.
* **SQLLite** or **PostgreSQL** with Dockerized DB for relational storage.
* **Makefile** or **Bash Scripts** for automation.
* **Jupyter Notebooks** to document EDA (Exploratory Data Analysis).
* **VSCode Dev Containers** (Perfect for showing Docker + Dev workflow).

**Free Project Management Stack**

* **Trello** (for tasks and board management).
* **GanttProject** (free Gantt chart app).
* **Draw.io** (for architecture diagrams).
* **Markdown** on GitHub (for all documentation, README, etc.).

# Project Refinement

**Stage 1 – MVP Web Scraper**

* **Tech:** Python, Selenium, Docker, venv
* **Deliverable:** Dockerized daily scraper, outputs to CSV/JSON.

**Stage 2 – Data Storage Layer**

* **Tech:** SQLite or PostgreSQL in Docker.
* **Deliverable:** Store data relationally, queryable schema.

**Stage 3 – Automation & Scheduling**

* **Tech:** Cron (Local) or AWS Lambda + EventBridge (Cloud).
* **Deliverable:** Fully automated daily run.

**Stage 4 – Data Visualization Dashboard**

* **Tech:** Streamlit or Dash, Matplotlib/Plotly, Docker.
* **Deliverable:** Web dashboard showing trends, highs, lows.

**Stage 5 – Alerts System**

* **Tech:** SMTP (Email), Twilio Free Tier (SMS/WhatsApp).
* **Deliverable:** Notifications when price drops below threshold.

**Stage 6 – ML-Based Price Predictions**

* **Tech:** Pandas, Scikit-learn, Jupyter Notebooks.
* **Deliverable:** Predictive model + documented analysis.

**Stage 7 – Documentation & Portfolio**

* **Tech:** Markdown, Diagrams, GitHub Pages (optional).
* **Deliverable:** Fully documented project repo, Gantt, diagrams.