Assignment

Objective

Build a small Python service that fetches stock market data, stores it locally, and exposes it through a web server.

This assignment tests:

- Code clarity & structure
- API usage & data handling
- Basic architecture instincts
- Ability to think like an engineer, not a script writer

📌 Core Requirements

1 Fetch Market Data

- Use **Yahoo Finance** (`yfinance` recommended)
- Fetch **OHLCV** (Open, High, Low, Close, Volume) for a given ticker
- Fetch either:
  - at a **fixed interval**, OR
  - **on demand** via an endpoint (your choice)

2 Store the Data

- Storage must be **local** using either:
  - **SQLite**, or
  - **CSV**
- Handle duplicates and bad data sensibly
- Clearly document schema/format

3 Serve the Data

Create a web server (FastAPI preferred, Flask acceptable) with:

Endpoint	Description
`GET /last`	Returns the latest stored data point

`GET /history`	Returns all stored data
`POST /fetch?ticker=TSLA`	Triggers a fetch & store operation

Responses must be in **JSON**.

🧱 Architectural Expectations

We're not dictating a specific design, but we expect:

- **Separation of concerns**
  Fetching, storage, and API logic should not live in one tangled file.
- **Configurable behavior**
  Tickers, intervals, DB path, etc. should be changeable without modifying code logic (env vars, config file, arguments, etc.).
- **Maintainability**
  If the project grew 3× in size, the structure should still make sense.
- **Justify your decisions** in the README.

❌ Avoid

- Single-file, 500+ line scripts
- Copy-pasting half of StackOverflow
- Hardcoded paths / magic values everywhere

🧪 Testing Requirements

We expect **2–5 unit tests**, focusing on **core logic**, not the entire project.

Suggested areas:

- Data parsing & validation
- Handling invalid ticker or bad inputs
- DB/CSV write & read logic

✅ Good Practices

- Mock external API calls

- Name tests clearly
- Test behavior, not just happy paths

❌ Avoid

- Calling the real API in tests
- “assert True” placeholders
- Tests that require internet connection

Use **pytest** or **unittest**, whichever you prefer.

📁 Deliverables

Submit the following:

- Source code (Git repo preferred)
- **README** with:
  - Setup + run instructions
  - Explanation of decisions & trade-offs
  - Assumptions & limitations
- Optional bonus:
  - Dockerfile
  - Simple ASCII architecture diagram

🤔 Extension Questions (Optional but Impressive)

Answer briefly in README if you want to stand out:

1. How would this scale to handle **10 tickers** concurrently?
2. How would you avoid **API rate limits**?
3. What’s the first architectural change you'd make for **production**?
4. What’s a trading-related pitfall of using this setup as-is?

⏳ Expected Time

Ideally, the assignment should take around 1–2 hours to complete. However, you will have up to 24 hours to submit it.

📫 Submission

Please share your completed assignment via:

- **GitHub/GitLab link**, and
- Send it to **apurv@cirqllabs.com**

For any queries or clarification regarding the assignment, feel free to reach out to **Apurv Salunke**.

🟢 Final Note

We don't expect perfection.
We expect **ownership**, **reasoning**, and **effort**.

If you've never touched trading systems or yfinance before — that's fine.
Show us how you think, learn, and build.
