Setup and Running the Project

To run this project on your local machine, follow these steps:

1. **Prerequisites:** Install Python 3 and pip. Ensure you have an internet connection if you plan to use live data APIs.
2. **Project Files:** Organize the files as per the structure described (or clone the repository if provided). You should have `app.py`, the module files (`data.py`, `exchange.py`, `sor.py`, `strategies/` package), and optionally the `docs/`folder containing agile artifact files (like `backlog.csv`).
3. **Install Dependencies:** Install required Python libraries. Mainly:
   - Streamlit (for the GUI): `pip install streamlit`streamlit.io
   - Pandas (for data handling, if not already installed with Streamlit).
   - yfinance (if using live Yahoo Finance data): `pip install yfinance`.
     (Alpha Vantage usage would require their `alpha_vantage` library or `requests` to call their API, plus getting a free API key.)
4. **(Optional) API Keys:** If using Alpha Vantage or other APIs that require keys, obtain a free API key from their website and set it in `data.py` or as an environment variable. For Yahoo (yfinance), no key is needed. For Binance public endpoints, no key needed for public datadevelopers.binance.com.
5. **Run the App:** In a terminal, navigate to the project directory and run:
   `streamlit run app.py`

   This will launch the Streamlit app in your web browser. Use the sidebar to configure the simulation and press **Run Simulation**.
6. **Interpreting Results:** After running, observe the output:
   - The **Executed Trades** table will show each trade's time step, which exchange was chosen by the SOR, price, and quantity.
   - The **Average Execution Price** is calculated for your order – you can compare this to the price trend to see if you achieved a favorable price. For instance, check if it's close to the overall VWAP of the price series.
   - The **Price chart** will display each exchange's price over time. You should see that trades (from the table) occurred at the lowest (for buys) or highest (for sells) price points at those times – validating the SOR's function. If using VWAP, you'll notice larger trades at beginning/end; TWAP will show uniform trade timing; POV's trade sizes will vary with volume.
   - If something looks off (e.g., no trades executed because schedule was all zeros), adjust parameters and re-run. The interactivity of Streamlit lets you iterate quickly.
8. **Further Exploration:** You can modify the code to experiment:
   - Try changing the volume profile in VWAP (e.g., make it more extreme) and see the effect.

- Use a different stock or a cryptocurrency for live data to see how the algorithms perform on various market conditions.
- Increase the number of exchanges in synthetic mode; perhaps make one exchange consistently cheaper by adding a bias to its price – the SOR should then mostly route to that exchange.
- Inspect the Agile artifact files (`backlog` or `burndown`) to see how the project was planned. You can update the backlog (e.g., mark remaining tasks as done) and regenerate the burn-down chart by running the plotting code (the one used to produce the embedded image earlier).

This project provides a comprehensive look at how a trading execution system can be built and documented. By following an Agile approach, we ensured each feature (data feed, SOR, each algorithm, GUI) was tackled in increments, and progress was visible. The final result is a working Python application where one can learn about SOR and algorithms hands-on. The modular code can serve as a foundation for more complex extensions, such as adding an order book simulation per exchange or including more sophisticated strategies (implementation shortfall, adaptive algorithms, etc.). We hope this serves as a useful educational tool for both algorithmic trading concepts and Agile software practice.

Sources: The definitions and concepts utilized in this project were informed by external references for accuracy. For instance, the SOR description and importance of liquidity fragmentation were based on industry explanations medium.comen.wikipedia.org. The algorithm strategies' purposes were cross-checked with trading literature: VWAP as a popular execution benchmark empirica.io, TWAP as a uniform time-based strategychain.link, and POV as a participation strategy targeting a percentage of volume ibkrguides.com. Agile artifact definitions (sprint backlog and burn-down) are per standard Agile guides asana.comasana.com. The data sources mentioned (Yahoo Finance, Alpha Vantage, Binance API) are well-known free resources alphavantage.corowzero.iodevelopers.binance.com. All these helped ensure the project's educational content is grounded in real-world practices and terminology.