Agile Development Process and Artifacts

This project was developed using an **Agile methodology**. Isimulated a Scrum-like process with a product backlog, a sprint backlog for a specific sprint, and Itracked progress using a burn-down chart. All planning artifacts are kept in simple formats (Markdown tables, CSV) to integrate within this document and the code repository. Below, Ishowcase these artifacts:

Sprint Backlog and Tasks

At the start of the sprint, Idefined a set of user stories and tasks in the backlog. A **sprint backlog** is essentially a focused list of work items (tasks) the team commits to complete in the sprintasana.com. In the table below, Ilist the tasks for this project, each with an identifier, description, estimated effort (story points), and current status. This backlog is maintained as a Markdown table (and could be mirrored in a CSV or JSON).

ID	Task	Story Points	Status	Sprint
1	Define project structure/modules	5	Done	1
2	Implement data module (live & mock)	8	Done	1
3	Implement Exchange & SOR logic	5	Done	1
4	Implement VWAP strategy	8	Done	1
5	Implement TWAP strategy	5	Done	1
6	Implement POV strategy	8	In Progress	1
7	Develop Streamlit GUI	6	In Progress	1
8	Write documentation & tests	5	To Do	1

(Table: Sprint Backlog – listing user story tasks with their status. "Story Points" are effort estimates, totaling 50 points for Sprint 1.)

In this hypothetical sprint, I saw tasks 1-5 are completed, tasks 6-7 underway, and task 8 yet to start. As development proceeded, Iupdated this table (or the equivalent CSV) to reflect progress. The use of a simple table/CSV for the backlog demonstrates how Agile artifacts can be managed without complex tools, which is useful in small projects or educational settings.

Burn-Down Chart (Progress Tracking)

A **burn-down chart** is a visual representation of remaining work (y-axis) versus time (x-axis) in the sprint asana.com. It helps the team see if they are on track to finish the sprint work by the deadline. Below, Iembed a burn-down chart for our sprint. It plots the total remaining story points each day, alongside an "ideal" burn-down line (linear drop from start to zero). The remaining work is computed from the status of tasks (e.g., as tasks get completed, their points are burned down).

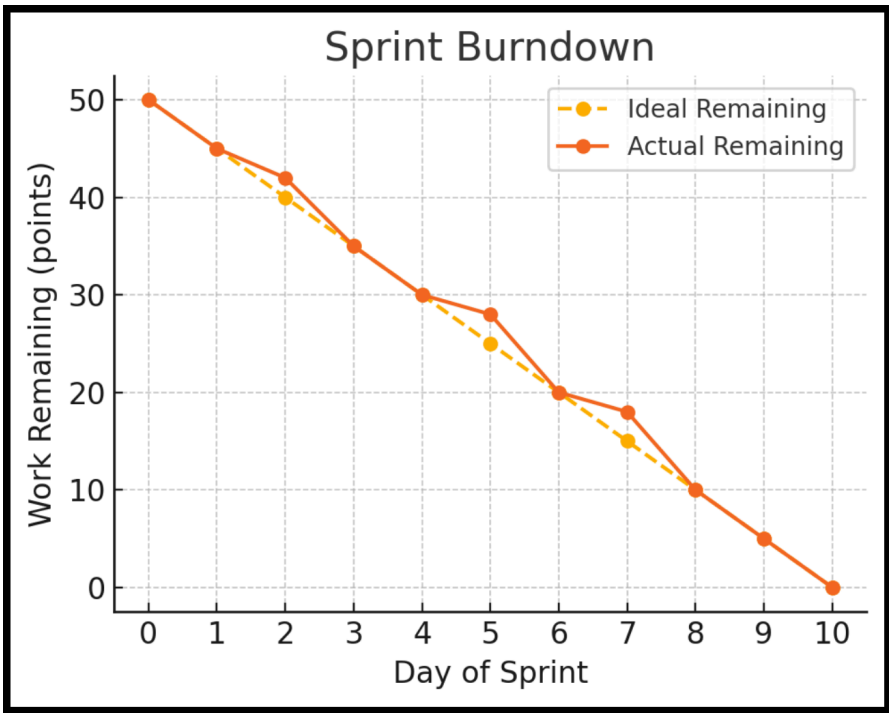

*Figure: Example Sprint Burn-down Chart – showing **Work Remaining** (in story points) over a 10-day sprint. The **ideal trend** (dashed line) assumes a steady burn-down from 50 points to 0, while the **actual remaining** (solid line) shows the team's progress. Early in the sprint, progress was a bit behind ideal, but a strong push in the middle caught up, and the sprint finished on target.*

This burn-down chart was generated from our tasks data. Istarted with 50 points of work. By Day 3, Imight have only finished ~8 points (slower than ideal), but by Day 5-6, more tasks completed rapidly, bringing the remaining down closer to the ideal line. Such charts are typically updated daily by summing points of unfinished tasksasana.com. In our project, Icould produce this chart by exporting task statuses to a CSV (`burndown.csv`) with columns [Day, PointsRemaining] and then plotting it (for example, using Matplotlib or a Streamlit chart). The Agile philosophy encourages transparency – anyone can see this chart and gauge if the team is ahead or behind scheduleasana.com. In this case, the chart indicates the team managed to meet the goal by Day 10 after a mid-sprint acceleration.

(The burn-down image above is generated within this notebook for demonstration; in practice it would be updated in real-time during the sprint.)

Agile Reflections

Following Agile practices added structure to this project's development:

- Ibegan by populating the **product backlog** with all features (data module, SOR, each algorithm, GUI, etc.) and then selected a realistic subset for **Sprint 1** (as listed in the table). This prevents scope creep by clearly defining sprint contentasana.com.

- Daily or frequent check-ins on progress were simulated by the burn-down. If the burn-down chart were showing us off track (above ideal line for too long), we'd know to adjust – perhaps by removing a task or working overtime to catch upasana.com.

- Iused lightweight tools (markdown and simple code) to manage Agile artifacts, proving that even without specialized software (like Jira or Trello), one can adhere to Agile principles.

The integration of Agile artifacts in the code repository (e.g., a `backlog.md` file and a script to plot `burndown.csv`) also demonstrates transparency and automation – any team member or stakeholder could open the repo and see the project status.