Retrospective: API Rate Limiter (ARL)

Sprint 1

1. What Went Well

- **Clear User Stories & Planning:** The project benefited from clear and well-defined user stories due to detailed description and planning. This allowed the team to implement features effectively.
- **Successful Sprint 1 Core Delivery:** The team successfully delivered the core functionalities in Sprint 1, completing key stories for client identification and the rate limiting engine.

2. What Didn't Go Well

- **Initial Design Blockers:** The team encountered unexpected issues with the design and architecture plan. We had to move from Pipe & Filter architecture to Multi Layered architecture. This required extra time to resolve and impacted the initial flow of work.
- **Unclear Testing Strategy:** The team was initially operating as 2 developers and 2 testers, but soon realised that it would be more efficient for every developer to write their own unit tests.
- **Sprint 1 Re-planning:** Due to the realization about testing, the team had to pivot mid-sprint and make the decision to push all integration testing to Sprint 2. This meant the original Sprint 1 commitment was not fully met as planned. Due to this, we also could not implement one user story (ARL-20), which had to be moved to Sprint 2.
- **Backlog Clutter:** Duplicate testing tasks (ARL-36, ARL-37) were left in the backlog as Jira was not allowing us to delete those redundant tasks.

3. What Should We Do to Fix It

- **Refine Testing Strategy:** For future projects, define the testing strategy (e.g., unit tests with each story, system integration test at the end) *before* the first sprint. This will prevent creating duplicate or poorly timed testing tasks and lead to greater efficiency.
- **Account for Design/Architecture Time:** When planning, explicitly block time or create a task for complex design/architecture work before committing to the implementation stories that depend on it.

Sprint 2

1. What Went Well

- **High Velocity :** Sprint 2 was highly productive, with the team completing all 8 planned tasks/stories. This success was built on the learnings and stable core code established in Sprint 1.

- **Successful Integration Testing:** The team successfully completed the "System Integration Testing" (ARL-40) in Sprint 2, validating the end-to-end functionality of the system.

2. What Didn’t Go Well

- **Incomplete CI/CD Pipeline:** We had only 2 stages in CI/CD : build and test. The test stage included the linting which should have been a separate stage. It was quite late into the sprint that we realised this, since the detailed evaluation rubrics that mentioned this requirement were released halfway through Sprint 2. This was fixed after the sprint officially ended.
- **Missing code coverage quality gates:** For similar reasons, we did not know about the coverage stage requirement, and hence this was missing. This was also added later after the code coverage lab in class.

3. What Should We Do to Fix It

- Each sequential stage of the CI/CD pipeline must be planned well in advance with clarity.
- Include a dedicated CI/CD task in Jira to properly accommodate and prioritize accordingly.