Sprint 1 Retrospective Report

API Rate Limiter Project

Sprint: Sprint 1 - Core Features + Basic Infrastructure
Date: End of Week 2
Team ID: 73
Team members: Yeswant Padavala, Vishal Naik, Nigam Reddy
Sprint Duration: 1 and half weeks
Participants: All 3 team members present

What Went Well ✓

Strong Technical Foundation

- Successfully established Redis-based counter storage with automatic TTL resets, providing a solid foundation for rate limiting functionality
- HTTP 429 response handling with Retry-After headers works smoothly and meets basic performance requirements
- Graceful shutdown mechanism handles in-flight requests properly, ensuring zero data loss during deployment
- Security headers implementation using helmet.js added protection against common web vulnerabilities from day one

Good Progress Across Epics

- Covered 9 out of 10 planned epics in the first sprint, demonstrating broad system coverage
- Achieved 34 story points as planned, meeting our velocity target
- JSON structured logging provides excellent visibility into system behavior and rate limit violations
- Basic API key authentication middleware integrated smoothly with the rate limiting logic

---

What Needs to Be Changed ⚠

Technical Challenges

- Redis connection pooling configuration took longer than expected - initial settings caused connection timeouts under load testing
- Performance testing revealed response times averaging 35-40ms, which is above our stretch goal of sub-10ms (though acceptable for MVP)
- Input validation logic became repetitive across endpoints - needs refactoring into reusable middleware
- Load testing scripts were created too late in the sprint (Day 10), limiting our ability to identify bottlenecks early

Scope and Planning Issues

- Policy management epic was reduced to just REST API, leaving gap in user-friendly configuration - team felt web UI was needed sooner
- Documentation was rushed in the final day - README lacks detailed examples and troubleshooting steps
- Several "nice-to-have" features were cut mid-sprint to meet deadlines, affecting feature completeness
- Testing coverage reached only 62%, falling short of our 70% target

Time Management

- College midterm exams in Week 1 reduced actual available working hours by approximately 30%
- Integration day (Day 7) revealed that components weren't as compatible as assumed, requiring 4 additional hours of rework
- Nigam had to handle multiple epics (counters, shutdown, auth) with limited overlap, causing stress in the final days
- Team underestimated the complexity of authentication middleware integration

Communication Gaps

- Assumptions about Redis schema design weren't documented early, causing confusion during integration
- Vishal and Nigam worked on related components (enforcement + counters) but didn't sync until Day 7
- Missing context about burst traffic requirements until late in sprint affected counter design

---

What Needs to Be Started 🚀

Technical Improvements

- Begin implementing automated integration tests to catch compatibility issues before mid-sprint review
- Start using Redis connection pooling best practices documentation as a reference for configuration
- Create shared utility library for common functions like validation, error handling, and response formatting
- Set up continuous performance monitoring to track latency trends throughout Sprint 2

Process Enhancements

- Implement a "definition of ready" checklist for stories before sprint planning to ensure requirements are clear
- Schedule a 30-minute architecture sync every Monday and Thursday to align on design decisions
- Start tracking "unplanned work" separately to understand actual capacity vs. planned capacity
- Create a shared knowledge document for Redis patterns, security best practices, and common issues

Testing and Quality

- Begin writing tests alongside code (TDD approach) rather than at the end of tasks
- Start automated test runs on every commit using GitHub Actions or similar CI/CD tool
- Implement code coverage tracking visible to all team members to maintain accountability
- Create integration test suite structure in Sprint 2 Day 1 to avoid last-minute rush

Documentation Practices

- Start maintaining API documentation using Swagger/OpenAPI from Sprint 2 Day 1
- Create architecture decision records (ADRs) for major design choices like algorithm selection
- Begin keeping a team wiki with setup instructions, common errors, and solutions
- Record short demo videos for each major feature to supplement written documentation

---

Action Items

Action Item	Owner	Due Date	Priority
Refactor Redis connection pooling configuration	Nigam	Sprint 2 Day 3	High
Create shared validation middleware	Vishal	Sprint 2 Day 4	High
Set up GitHub Actions for automated testing	Yeswant	Sprint 2 Day 2	Medium
Document Redis schema and design patterns	Nigam	Sprint 2 Day 2	High
Improve README with detailed examples	All	Sprint 2 Day 5	Medium
Create "definition of ready" checklist	Yeswant	Before Sprint 2 planning	High
Schedule bi-weekly architecture syncs	Vishal	Sprint 2 Day 1	Medium
Increase test coverage to 70%+	All	Sprint 2 Day 8	High

Sprint 1 Metrics

Delivery Performance

- **Planned Story Points:** 34 SP
- **Completed Story Points:** 34 SP
- **Velocity:** 34 SP (100% completion)
- **Epics Touched:** 9 out of 10
- **Carryover Items:** 0 stories (though some technical debt identified)

Quality Metrics

- **Test Coverage:** 62% (target: 70%)
- **Bugs Found:** 5 (3 minor, 2 moderate - mostly in Redis connection handling)
- **Code Review Completion:** 100% of PRs reviewed
- **Performance:** P95 latency 38ms (target: <50ms for MVP - ACHIEVED)

Team Metrics

- **Daily Standup Attendance:** 95% (missed 2 due to exams)
- **Mid-Sprint Review Attendance:** 100%
- **Actual Hours vs. Planned:** ~70% (due to midterm exams)

Sprint Health Assessment

Overall Sprint Health: 🟡 Good with Room for Improvement

Technical Delivery: 🟢 Strong - All planned features delivered
Quality: 🟡 Acceptable - Test coverage below target but functionality solid
Team Collaboration: 🟢 Excellent - Good communication and support
Process Adherence: 🟡 Moderate - Some planning gaps and rushed testing

Team Morale: 😊 Positive but stretched

- Team feels accomplished covering 9 epics in 2 weeks
- Some stress around exam conflicts and last-minute integration
- Excited about Sprint 2 to add algorithms and complete all epics

---

Key Learnings

What We Learned About Our System

- Redis is extremely fast but connection management requires careful configuration
- Rate limiting enforcement is straightforward, but burst handling will need sophisticated algorithms in Sprint 2
- JSON logging provides excellent debugging capabilities - worth the setup time

What We Learned About Our Team

- Our velocity estimate of 30-35 SP was accurate when accounting for college commitments
- Integration issues are inevitable - mid-sprint check was crucial and should continue
- Code reviews catch most issues, but integration testing catches the rest

What We Learned About Our Process

- Need better "definition of ready" to avoid mid-sprint scope clarifications
- Testing should be continuous, not end-of-sprint activity
- Documentation takes longer than expected and should be incremental

---

Looking Ahead to Sprint 2

Key Focus Areas

1. **Complete EPIC-10 (Testing)** - Bring coverage above 70% and add comprehensive integration tests
2. **Add Multiple Algorithms** - Fixed window and token bucket implementations for EPIC-1
3. **Implement TLS Security** - Critical for production readiness
4. **Polish Documentation** - Complete admin guide and improve developer quick start

Risk Mitigation

- Start Sprint 2 with architecture alignment session to avoid integration issues
- Allocate 20% of sprint capacity for testing and quality improvements
- Set up CI/CD on Day 1 to catch issues continuously

Confidence Level: High - Team has demonstrated ability to deliver, and Sprint 2 work builds on solid Sprint 1 foundation

---

Additional Notes

The team demonstrated excellent resilience managing college coursework alongside project development. While midterms impacted available hours, the quality of delivered work was high. The decision to cover 9 epics broadly rather than 5 epics deeply was validated - we have a working system demonstrating all major capabilities.

The Redis connection issue discovered during load testing was frustrating but provided valuable learning about production-readiness considerations. Looking forward to Sprint 2 with optimism and better preparation!
