Software Test Plan (STP) – API Rate Limiter

Project: API Rate Limiter
Version: 1.0
Authors: Yeswant Padavala (PES1UG23CS721), Vishal Naik (PES1UG23CS695), Nigam Reddy (PES1UG23CS904)
Date: 07-09-2025
Status: Sample / Draft

1. Introduction

Purpose: This document defines the test plan for the API Rate Limiter System v1.0. It outlines objectives, scope, strategy, resources, schedule, and responsibilities for testing.

Scope: Testing covers all testing activities for API Rate Limiter System such as functional testing, performance testing, security testing, integration testing, and user acceptance testing

References: API Rate Limiter SRS, Design Specifications v1.0

Definitions: API (Application Programming Interface)

2.1. Test Levels

- **Unit Testing:** Individual component testing
- **Integration Testing:** Component interaction testing
- **System Testing:** End-to-end functionality testing
- **Acceptance Testing:** User requirement validation

2.2. Test Types

- Functional Testing
- Performance Testing
- Security Testing
- Compatibility Testing
- Usability Testing
- Stress Testing

- Load Testing

3. Features to be Tested

3.1. Functional Features

- Request counting and tracking (RL-F-001 to RL-F-003)
- Rate limit enforcement (RL-F-004 to RL-F-006)
- Policy management (RL-F-007 to RL-F-009)
- Monitoring and logging (RL-F-010 to RL-F-012)
- High availability and clustering (RL-F-013 to RL-F-015)

3.2. Non-Functional Features

- Performance requirements (< 10ms latency)
- Security requirements (TLS 1.2+, DoS protection)
- Scalability requirements
- Reliability requirements (99.9% uptime)

4. Features Not to be Tested

- Backend business logic of protected APIs
- Network-level throttling outside application scope
- Third-party authentication service internals
- Operating system level configurations

5. Test Approach / Strategy

- **Black-box Testing:** Focus on input-output behavior without internal code knowledge
- **White-box Testing:** Code coverage analysis and internal logic testing
- **Gray-box Testing:** Combination approach for integration testing
- **Automated Testing:** Unit tests, integration tests, and performance tests
- **Manual Testing:** User interface testing and exploratory testing

6. Test Environment

6.1. Hardware

- **Test Servers:** Minimum 4 GB RAM, 2 CPU cores
- **Load Testing:** Dedicated load generation servers
- **Network:** Minimum 100 Mbps bandwidth

6.2. Software

- **Operating System:** Linux (Ubuntu 20.04+) or Windows Server 2019+
- **Container Platform:** Docker, Kubernetes (optional)
- **Database/Cache:** Redis 6.0+
- **Monitoring:** Prometheus, Grafana
- **Load Testing Tools:** Apache JMeter, Artillery
- **Security Testing:** OWASP ZAP

6.3. Test Data

- Valid API keys and tokens
- Invalid/expired credentials
- Various IP addresses for testing
- Policy configuration templates
- Load testing datasets

7. Test Schedule

Milestones:
- Test case design: 07-Sep-2025
- Environment setup: 10-Sep-2025
- Test execution start: 12-Sep-2025
- Test execution end: 05-Oct-2025
- UAT: 07-Oct-2025 to 10-Oct-2025

8. Test Deliverables

- Test Plan (this document)
- Test Cases (manual & automated)
- Test Scripts
- Test Data

- Test Execution Logs
- Defect Reports
- Test Summary Report

9. Roles and Responsibilities

Role	Name	Responsibility
QA Lead	Nigam Reddy	Prepare plan, coordinate execution
Test Engineer	Yeswant Padavala	Design & execute test cases, log defects
Developer	Yeswant Padavala	Support defect fixes and triage
Product Owner	Vishal Naik	Approve test results, sign-off readiness

10. Risks and Mitigation

Risk	**Level of Risk**	**Mitigation**
Performance degradation: Rate limiter may add excessive latency	High	Extensive performance testing and optimization
Distributed Counter Consistency: Race conditions in clustered environment	High	Thorough integration testing with multiple nodes
Security Vulnerabilities: Bypass attempts or credential exposure	High	Security testing and code review
Configuration Errors: Incorrect policy settings	Medium	Validation testing and user training

Third-party Integration Issues: Problems with Redis, monitoring tools	Medium	Integration testing with various versions

11. Assumptions & Dependencies

- SRS document approved and baselined
- Test environment setup completed
- Test data prepared and validated
- Development code ready for testing
- Unit testing completed with > 80% code coverage

12. Suspension & Resumption Criteria

Suspend testing if:

- Critical defects that prevent testing
- Test environment unavailable for > 24 hours
- Major changes to requirements
- Resource unavailability affecting timeline

Resume testing if:

- Critical defects fixed and verified
- Test environment restored
- Updated requirements approved
- Required resources available

13. Test Case Management & Traceability

6.1 Functional Test Cases

6.1.1 Request Counting & Tracking (TC-001 to TC-010)

TC-001: Basic Request Counting

- **Objective:** Verify system accurately counts API requests per user

- **Prerequisites:** Rate limiter deployed, user authenticated
- **Steps:**
    1. Send 5 API requests from user A
    2. Check request counter for user A
    3. Send 3 API requests from user B
    4. Check request counter for user B
- **Expected Result:** User A counter = 5, User B counter = 3
- **Priority:** High

TC-002: Time Window Reset

- **Objective:** Verify request counters reset after time window expires
- **Prerequisites:** Fixed window policy configured (1 minute window)
- **Steps:**
    1. Send 10 requests within 1 minute
    2. Wait for window to expire
    3. Check counter value
    4. Send 1 new request
    5. Check counter value
- **Expected Result:** Counter resets to 0 after window expiry, then increments to 1
- **Priority:** High

6.1.2 Limit Enforcement (TC-011 to TC-020)

TC-011: Rate Limit Exceeded

- **Objective:** Verify HTTP 429 response when limit exceeded
- **Prerequisites:** Rate limit set to 10 requests per minute
- **Steps:**
    1. Send 10 requests (should succeed)
    2. Send 11th request
    3. Check response status and headers
- **Expected Result:** 11th request returns HTTP 429 with Retry-After header
- **Priority:** High

TC-012: Multiple Rate Limiting Strategies

- **Objective:** Test different algorithms (fixed window, sliding window, token bucket)
- **Prerequisites:** All algorithms implemented
- **Steps:** Test each algorithm with same traffic pattern
- **Expected Result:** Each algorithm behaves according to its specifications
- **Priority:** Medium

6.1.3 Policy Management (TC-021 to TC-030)

TC-021: Create Rate Limit Policy

- **Objective:** Admin can create new rate limit policies
- **Prerequisites:** Admin dashboard accessible
- **Steps:**
    1. Login as admin
    2. Navigate to policy creation
    3. Define new policy (100 RPS for premium users)
    4. Save policy
- **Expected Result:** Policy created successfully and applied
- **Priority:** High

6.2 Performance Test Cases

6.2.1 Latency Testing (TC-031 to TC-035)

TC-031: Response Time Measurement

- **Objective:** Verify rate limiter adds < 10ms latency
- **Prerequisites:** Performance testing environment ready
- **Steps:**
    1. Measure baseline API response time without rate limiter
    2. Enable rate limiter
    3. Measure API response time with rate limiter
    4. Calculate difference
- **Expected Result:** Added latency < 10ms for 95% of requests
- **Priority:** High

6.2.2 Throughput Testing (TC-036 to TC-040)

TC-036: High Load Testing

- **Objective:** System handles expected load without degradation
- **Prerequisites:** Load testing tools configured
- **Steps:**
    1. Generate 1000 RPS for 10 minutes
    2. Monitor system performance
    3. Check error rates and response times
- **Expected Result:** No errors, response times within SLA
- **Priority:** High

6.3 Security Test Cases

6.3.1 Encryption Testing (TC-041 to TC-045)

TC-041: TLS Version Verification

- **Objective:** Verify only TLS 1.2+ connections accepted
- **Prerequisites:** Security testing tools available
- **Steps:**
    1. Attempt connection with TLS 1.1
    2. Attempt connection with TLS 1.2
    3. Attempt connection with TLS 1.3
- **Expected Result:** TLS 1.1 rejected, TLS 1.2+ accepted
- **Priority:** High

6.3.2 DoS Protection Testing (TC-046 to TC-050)

TC-046: DoS Attack Simulation

- **Objective:** Rate limiter prevents DoS attacks
- **Prerequisites:** Attack simulation tools ready
- **Steps:**
    1. Launch simulated DoS attack (10000 RPS)
    2. Monitor system behavior
    3. Verify legitimate traffic still served
- **Expected Result:** Attack requests blocked, system remains responsive
- **Priority:** High

14. Test Metrics & Reporting

Metrics collected:
- % test cases executed
- % passed/failed
- Defect density
- Defect aging
- Requirement coverage

Reports:
- Daily execution status
- Final Test Summary Report

15. Approvals

Role	Name	Signature / Date
QA Lead	Nigam Reddy	Nigam 06/09/2025
Dev Lead	Yeswant Padavala	Yeswant 06/09/2025
Product Owner	Vishal Naik	Vishal 06/09/2025

16. Test Cases

Test Case ID	Test Scenario / Description	Preconditions	Test Steps / Input Data	Expected Result	Postconditions / Remarks
TC-Auth-01	Validate user authentication using valid API credentials	API server and database are running; valid user credentials exist	1. Send POST /login request with correct username and password.2. Observe API response.	Response code **200 OK**; returns valid API token.	User successfully authenticated; token stored in logs.
TC-Auth-02	Validate authentication with invalid credentials	API running	1. Send POST /login with incorrect password.2. Observe response.	Response code 401 Unauthorized; JSON message: "Invalid credentials".	Request rejected; no token generated.
TC-LIMIT-01	Verify API rate limit enforcement per user	User logged in; rate limit set to 100 requests/hour	1. Send 101 API requests to /api/data within one hour.2. Observe final response.	101st request returns 429 Too Many Requests.	API blocks additional requests for cooldown period.
TC-LIMIT-02	Verify rate limit reset after cooldown period	User recently exceeded request quota	1. Wait for one-hour cooldown.2. Send another API request to /api/data.	Request accepted; response code 200 OK.	Counter reset successfully; normal flow resumed.
TC-LIMIT-03	Validate different rate limits for user tiers (Free vs Premium)	Free and Premium users exist; rate limits set differently	1. Send multiple requests as both users.2. Track number of allowed requests.	Free user blocked after 100; Premium after 1000 requests.	Rate limit applied correctly per plan.
TC-LOG-01	Validate logging of each API request	Logging service active	1. Send multiple requests from different IPs.2. Check /logs endpoint or DB table.	Each request logged with timestamp, IP, endpoint, and status.	Request history visible in admin dashboard.
TC-MON-	Verify monitoring	Admin dashboard	1. Open /admin/dashb	Counts and charts	Dashboard data

01	dashboar d displays correct request statistics	accessible	oard.2. View total requests and blocked requests count.	match log database records.	synchroni zed correctly.
TC-ALE RT-01	Check if alert triggers when threshold breached	Alert system active; admin email configure d	1. Send repeated requests exceeding 500 per minute.2. Check alert system/email.	Alert email sent to admin: "High traffic detected."	Alerts functionin g correctly.
TC-SEC -01	Validate secure HTTPS communi cation	TLS/SSL enabled on server	1. Attempt API call over HTTP.2. Attempt API call via HTTPS.	HTTP returns 301 Redirect; HTTPS responds with 200 OK.	All data transmitte d securely.
TC-ERR -01	Validate proper error message format for blocked requests	API gateway running	1. Send request exceeding limit.2. Check error response JSON.	JSON contains {"error": "Rate limit exceeded" , "retry_aft er": 60}.	Error handled gracefully; no crash.
TC-PER F-01	Measure average API response time under load	JMeter test configure d	1. Send 500 concurrent requests to /api/data.2. Record response time.	95% of requests complete within 2 seconds.	Meets performan ce requireme nt.
TC-AV AIL-01	Verify API uptime monitorin g	Monitoring tool integrate d	1. Observe uptime status for 24 hours.	Uptime ≥ 99.9%.	Service reliability confirmed.
TC-DB-01	Validate rate limit counters update correctly in database	DB active and connecte d	1. Send API request.2. Query rate_limit table.	Request counter increment ed correctly per user.	DB synchroni zation verified.
TC-UI- 01	Verify dashboar d paginatio n and data filters	Admin UI operation al	1. Open dashboard logs section.2. Apply filters by user and date.	Filtered data displays correctly with paginatio n.	UI usability confirmed.

Notes for Students:

• Maintain a consistent test case ID format (e.g., TC-Module-XX)

- Each test case should trace back to a requirement ID from the SRS
- Include both functional and non-functional test cases
- Modify the sample table according to your project’s domain and functionality
- Add columns such as 'Actual Result' and 'Status (Pass/Fail)' during execution