Software Requirements Specification (SRS) Template

Project: API Rate Limiter
Version: 1.0
Authors: Yeswant padwala (PES1UG23CS721),Vishal Naik (PES1UG23CS695),Nigam reddy (PES1UG23CS904)

Revision history

Version	Date	Author	Change summary	Approval
1.0.0	30-08-2025	Yeswant padwala	Section 6,7,8	Vishal naik
1.0.0	30-08-2025	Vishal Naik	Section 4	Yeswant padwala
1.0.0	30-08-2025	Nigam Reddy	Section 3,5	Vishal Naik

Date: 30-Aug-2025
Status: Draft / Under Review

Table of Contents

1. Introduction

1.1 Purpose

This document is a Software Requirements Specification (SRS) for an API Rate Limiter system. It defines the functional and non-functional requirements, operational boundaries, and verification criteria intended for developers, system architects, QA engineers, and platform operators to use as a reference.

1.2 Scope

The system regulates and controls the number of API requests a client or user can make within a specified time window. It supports multiple rate-limiting strategies (fixed window, sliding window, token bucket, leaky bucket) and integrates with APIs via middleware or gateway layers. It covers enforcement, monitoring, logging, and alerting functions. Excludes backend business logic of APIs themselves, and does not handle network-level throttling outside the application scope.

1.3 Audience

Developers, API Gateway Administrators, System Architects, QA Engineers, DevOps Engineers, and Assessment Evaluators.

1.4 Definitions

List of acronyms:

- API – Application Programming Interface
- RPS – Requests Per Second
- QPS – Queries Per Second
- TTL – Time To Live
- IP – Internet Protocol
- JWT – JSON Web Token
- UI – User Interface
- DoS – Denial of Service
- SLA – Service Level Agreement

2. Overall description

2.1 Product Perspective

The API Rate Limiter sits between users and backend services. It controls how many requests a user or system can send in a given time. It can run as part of an API Gateway, proxy, or a small service.

2.2 Major Product Functions
- Count and track API requests.
- Apply rules (per user, per token, per IP).
- Block extra requests and return error (HTTP 429).
- Allow burst traffic but control average usage.
- Show usage stats and logs for monitoring.
- Work across multiple servers in a cluster.
- Let admins set or update limits easily.

2.3 User Roles and Characteristics
API User (Developer/App): Just uses APIs, expects clear errors when limits are hit.

- Admin/Operator: Configures limits, checks logs, and monitors usage.
- Business Owner: Decides limits for free, paid, or premium users.
- Security Officer: Ensures no abuse or overload of systems**.**

2.4 Operating Environment
Runs on servers, VMs, or containers. Works in cloud or on-prem systems.
Needs:

- Secure network (HTTPS/TLS).
- Connection to cache or database (like Redis).
- Works with monitoring tools (Grafana, Prometheus).

2.5 Constraints
- Must support TLS 1.2+ for secure connections.
- Should add very little delay (<10 ms per request).
- Must work reliably in distributed systems.
- Needs to scale for traffic spikes.
- Should integrate with existing API gateways.

3. External interface requirements

3.1 User Interfaces
- Web dashboard or command-line tool for administrators to configure limits and view usage.
- API documentation (Swagger/OpenAPI) for developers to understand how limits are applied.

3.2 Hardware Interfaces
- Runs on standard servers or cloud instances.
- Can connect with load balancers or HSMs (optional).

3.3 Software Interfaces
- Works with API Gateway middleware (e.g., NGINX, Kong).
- Connects to authentication services (API keys, OAuth, JWT).
- Uses cache/databases (e.g., Redis) to store request counters.
- Sends logs/metrics to monitoring tools (e.g., Prometheus, ELK).

3.4 Communications
- Uses HTTPS (TLS 1.2+) for all API traffic.
- Returns HTTP 429 status code when limits are exceeded.
- Provides Retry-After header so clients know when to retry.

- Supports clustering to share limits across multiple servers.

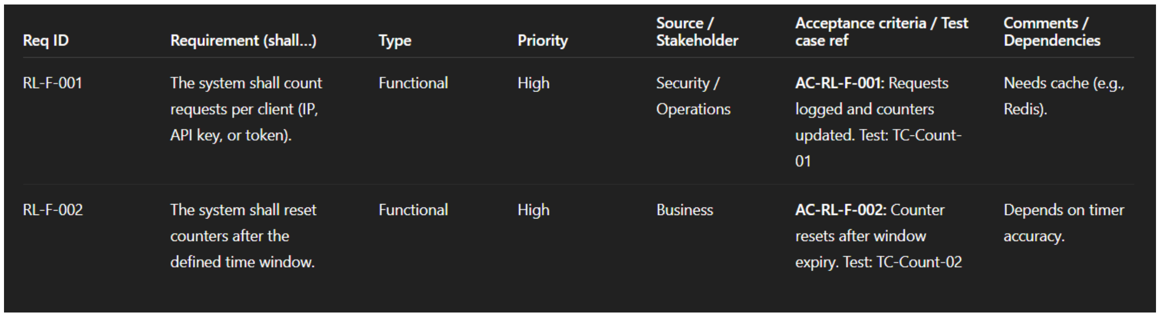
4. System features (detailed)

Each requirement below includes acceptance criteria and a reference test case. IDs follow RL-F-### (Rate Limiter – Functional).

4.1 Request Counting & Tracking

Description: Track the number of API requests per user/token/IP.

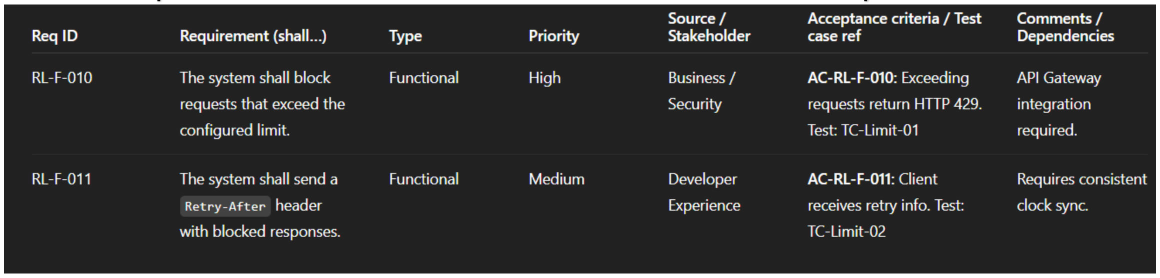
.



Req ID	Requirement (shall...)	Type	Priority	Source / Stakeholder	Acceptance criteria / Test case ref	Comments / Dependencies
RL-F-001	The system shall count requests per client (IP, API key, or token).	Functional	High	Security / Operations	**AC-RL-F-001:** Requests logged and counters updated. Test: TC-Count-01	Needs cache (e.g., Redis).
RL-F-002	The system shall reset counters after the defined time window.	Functional	High	Business	**AC-RL-F-002:** Counter resets after window expiry. Test: TC-Count-02	Depends on timer accuracy.

4.2 Limit Enforcement

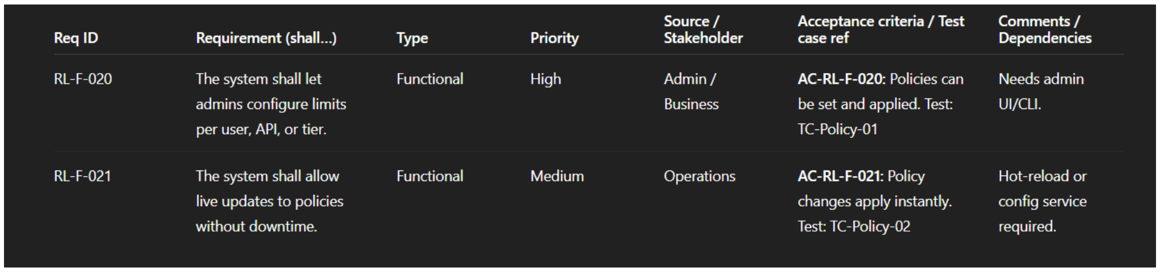
Description: Enforce rate limit rules and block excess requests.



Req ID	Requirement (shall...)	Type	Priority	Source / Stakeholder	Acceptance criteria / Test case ref	Comments / Dependencies
RL-F-010	The system shall block requests that exceed the configured limit.	Functional	High	Business / Security	**AC-RL-F-010:** Exceeding requests return HTTP 429. Test: TC-Limit-01	API Gateway integration required.
RL-F-011	The system shall send a `Retry-After` header with blocked responses.	Functional	Medium	Developer Experience	**AC-RL-F-011:** Client receives retry info. Test: TC-Limit-02	Requires consistent clock sync.

4.3 Policy Management

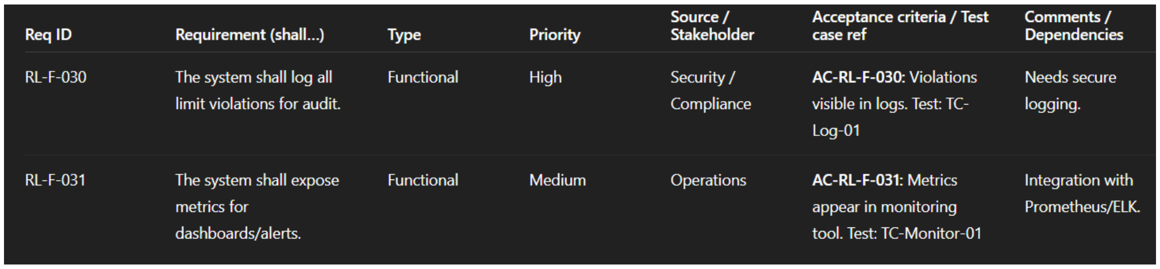
Description: Allow admins to define, update, and remove rate limit policies.



Req ID	Requirement (shall...)	Type	Priority	Source / Stakeholder	Acceptance criteria / Test case ref	Comments / Dependencies
RL-F-020	The system shall let admins configure limits per user, API, or tier.	Functional	High	Admin / Business	**AC-RL-F-020:** Policies can be set and applied. Test: TC-Policy-01	Needs admin UI/CLI.
RL-F-021	The system shall allow live updates to policies without downtime.	Functional	Medium	Operations	**AC-RL-F-021:** Policy changes apply instantly. Test: TC-Policy-02	Hot-reload or config service required.

4.4 Monitoring & Logging

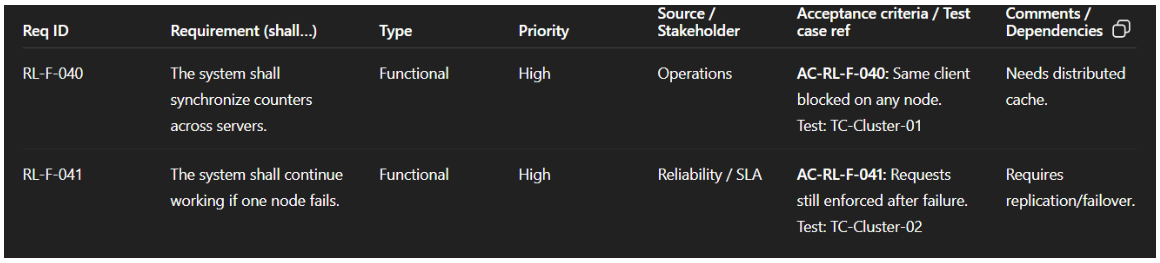
Description: Record usage and provide metrics for operators.



Req ID	Requirement (shall...)	Type	Priority	Source / Stakeholder	Acceptance criteria / Test case ref	Comments / Dependencies
RL-F-030	The system shall log all limit violations for audit.	Functional	High	Security / Compliance	**AC-RL-F-030:** Violations visible in logs. Test: TC-Log-01	Needs secure logging.
RL-F-031	The system shall expose metrics for dashboards/alerts.	Functional	Medium	Operations	**AC-RL-F-031:** Metrics appear in monitoring tool. Test: TC-Monitor-01	Integration with Prometheus/ELK.

4.5 High Availability & Clustering

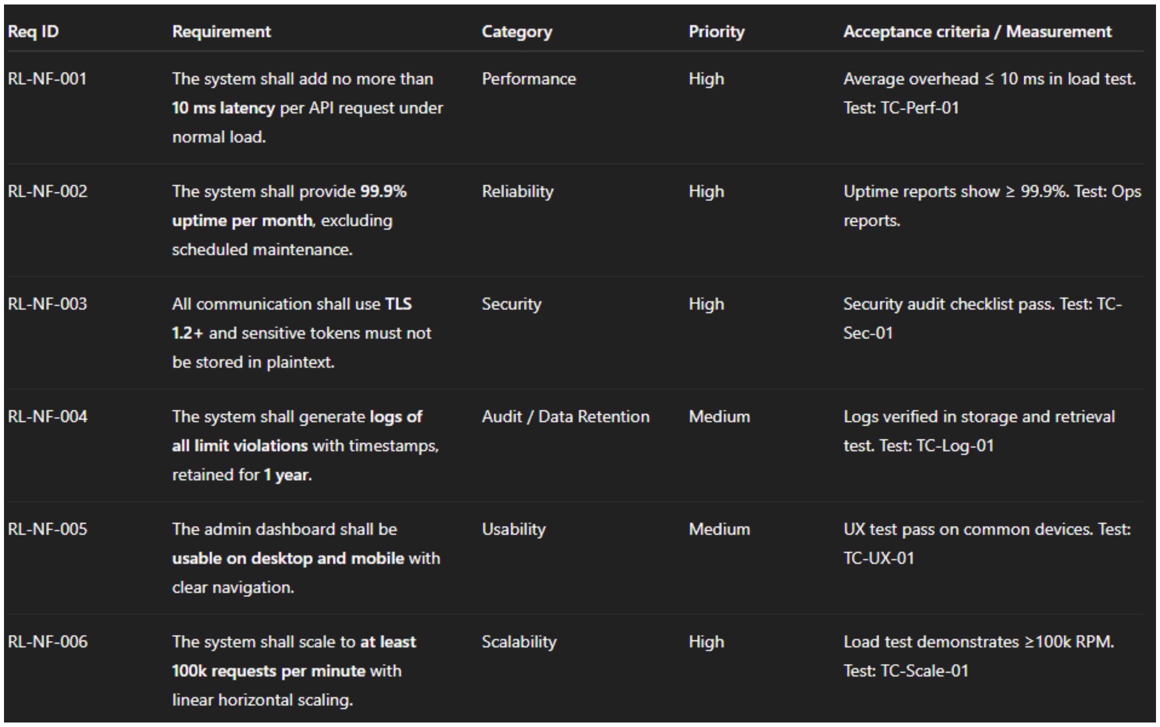
Description: Work across multiple nodes to enforce limits consistently.



Req ID	Requirement (shall...)	Type	Priority	Source / Stakeholder	Acceptance criteria / Test case ref	Comments / Dependencies
RL-F-040	The system shall synchronize counters across servers.	Functional	High	Operations	**AC-RL-F-040:** Same client blocked on any node. Test: TC-Cluster-01	Needs distributed cache.
RL-F-041	The system shall continue working if one node fails.	Functional	High	Reliability / SLA	**AC-RL-F-041:** Requests still enforced after failure. Test: TC-Cluster-02	Requires replication/failover.

5. Non-functional requirements (detailed)

NFRs below are measurable and tied to test plans. IDs follow RL-NF-###.



Req ID	Requirement	Category	Priority	Acceptance criteria / Measurement
RL-NF-001	The system shall add no more than **10 ms latency** per API request under normal load.	Performance	High	Average overhead ≤ 10 ms in load test. Test: TC-Perf-01
RL-NF-002	The system shall provide **99.9% uptime per month**, excluding scheduled maintenance.	Reliability	High	Uptime reports show ≥ 99.9%. Test: Ops reports.
RL-NF-003	All communication shall use **TLS 1.2+** and sensitive tokens must not be stored in plaintext.	Security	High	Security audit checklist pass. Test: TC-Sec-01
RL-NF-004	The system shall generate **logs of all limit violations** with timestamps, retained for **1 year.**	Audit / Data Retention	Medium	Logs verified in storage and retrieval test. Test: TC-Log-01
RL-NF-005	The admin dashboard shall be **usable on desktop and mobile** with clear navigation.	Usability	Medium	UX test pass on common devices. Test: TC-UX-01
RL-NF-006	The system shall scale to **at least 100k requests per minute** with linear horizontal scaling.	Scalability	High	Load test demonstrates ≥100k RPM. Test: TC-Scale-01

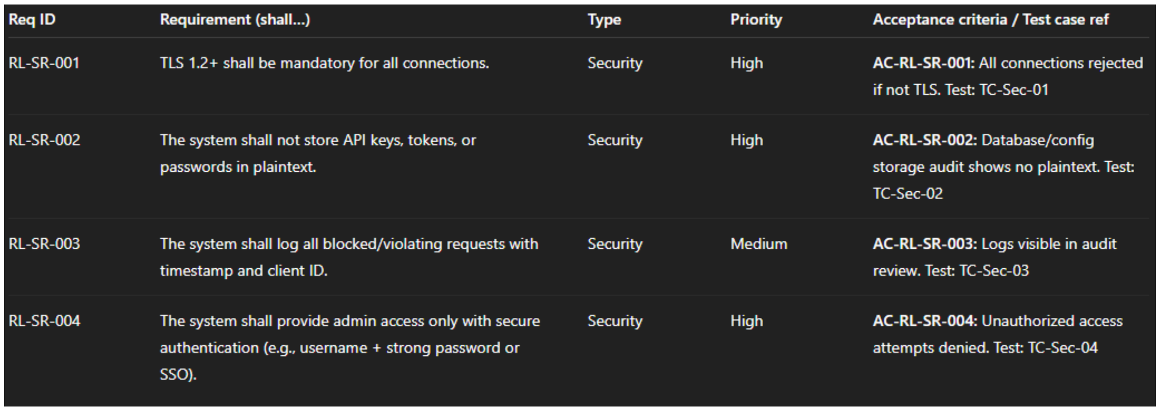
5.1 Security

5.1.1 Security Objectives

- Ensure all API traffic is encrypted in transit using TLS 1.2+.
- Prevent abuse or denial-of-service (DoS) through strong request limiting.
- Protect stored credentials and tokens from unauthorized access.
- Provide audit logs for all limit violations and admin actions.

.

5.1.2 Security Requirements



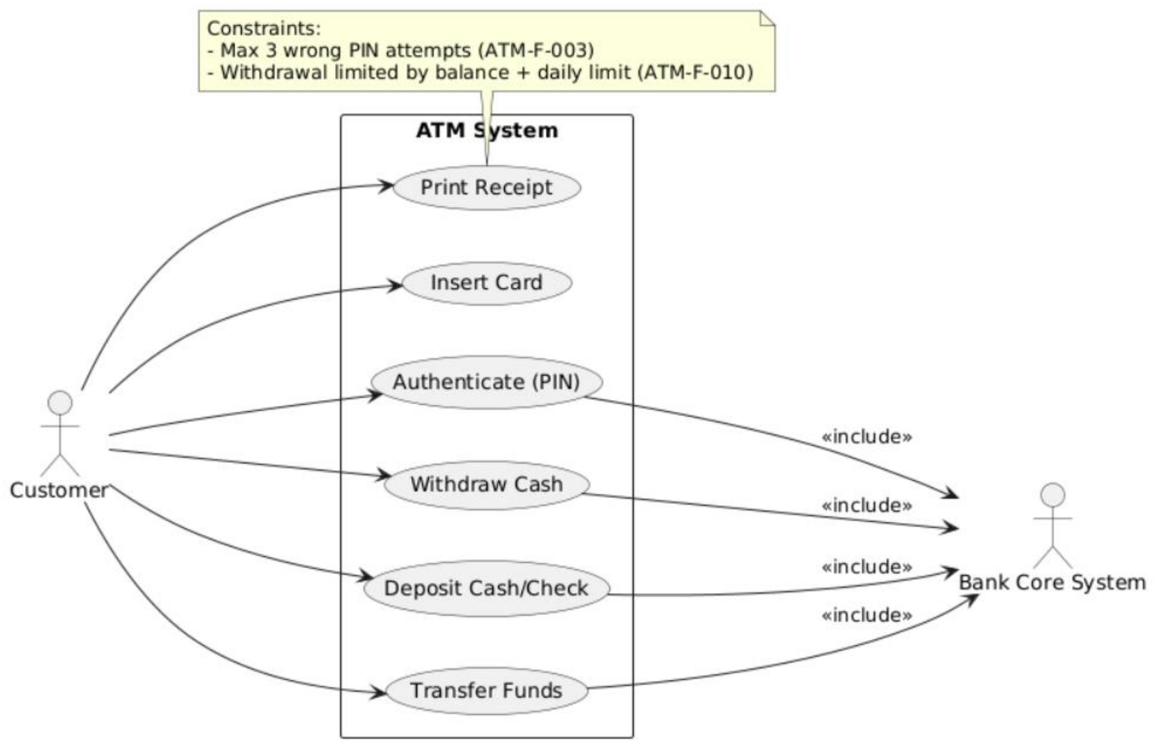
Req ID	Requirement (shall...)	Type	Priority	Acceptance criteria / Test case ref
RL-SR-001	TLS 1.2+ shall be mandatory for all connections.	Security	High	**AC-RL-SR-001:** All connections rejected if not TLS. Test: TC-Sec-01
RL-SR-002	The system shall not store API keys, tokens, or passwords in plaintext.	Security	High	**AC-RL-SR-002:** Database/config storage audit shows no plaintext. Test: TC-Sec-02
RL-SR-003	The system shall log all blocked/violating requests with timestamp and client ID.	Security	Medium	**AC-RL-SR-003:** Logs visible in audit review. Test: TC-Sec-03
RL-SR-004	The system shall provide admin access only with secure authentication (e.g., username + strong password or SSO).	Security	High	**AC-RL-SR-004:** Unauthorized access attempts denied. Test: TC-Sec-04

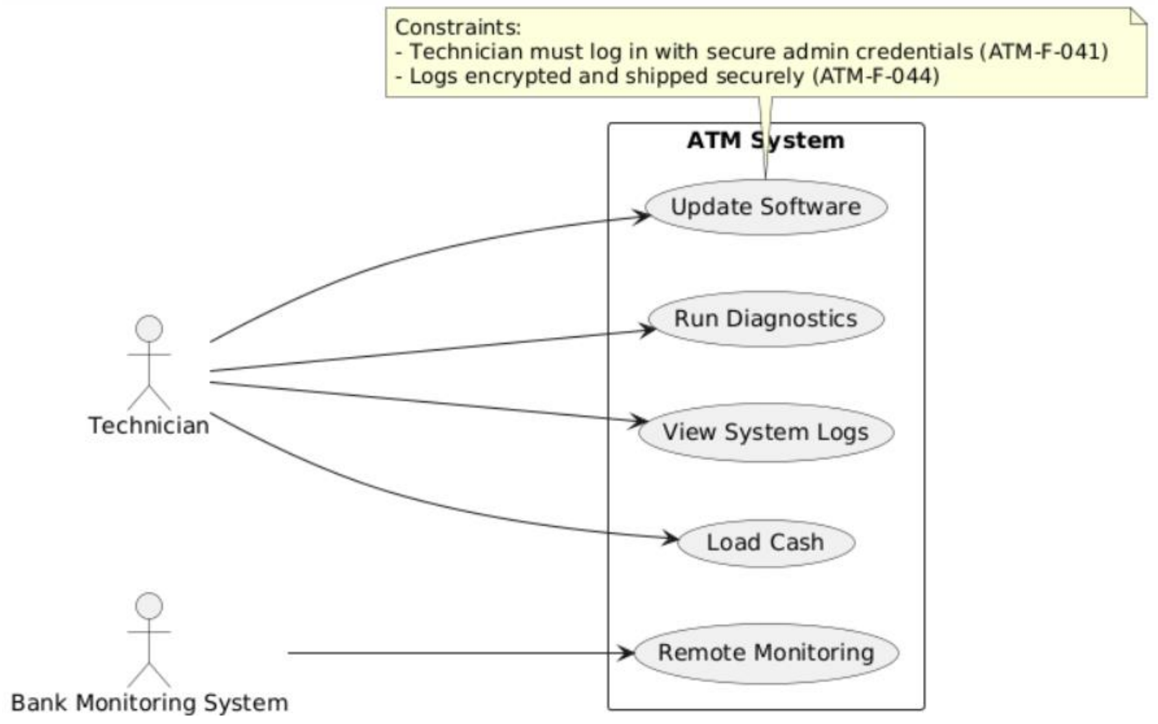
6. Quality Attributes & Acceptance Tests

- **Exit criteria for acceptance:**
    - All high-priority functional requirements (request counting, limit enforcement, policy management, monitoring, clustering) are implemented and verified.
    - No critical non-functional requirement failures (performance, security, availability).
    - Requirements Traceability Matrix (RTM) shows all mapped test cases passed.
- **Acceptance test suites:**
    - Counting & Tracking tests (validate request counters reset correctly).
    - Limit Enforcement tests (ensure HTTP 429 and Retry-After headers work).
    - Policy Management tests (verify admins can configure/update policies).
    - Monitoring & Logging tests (logs and metrics are accurate and retrievable).
    - Performance & Scalability tests (measure latency and throughput under load).
    - Security tests (TLS 1.2+, no plaintext secrets, secure admin access).
    - Usability tests (admin dashboard simple to use on desktop/mobile).

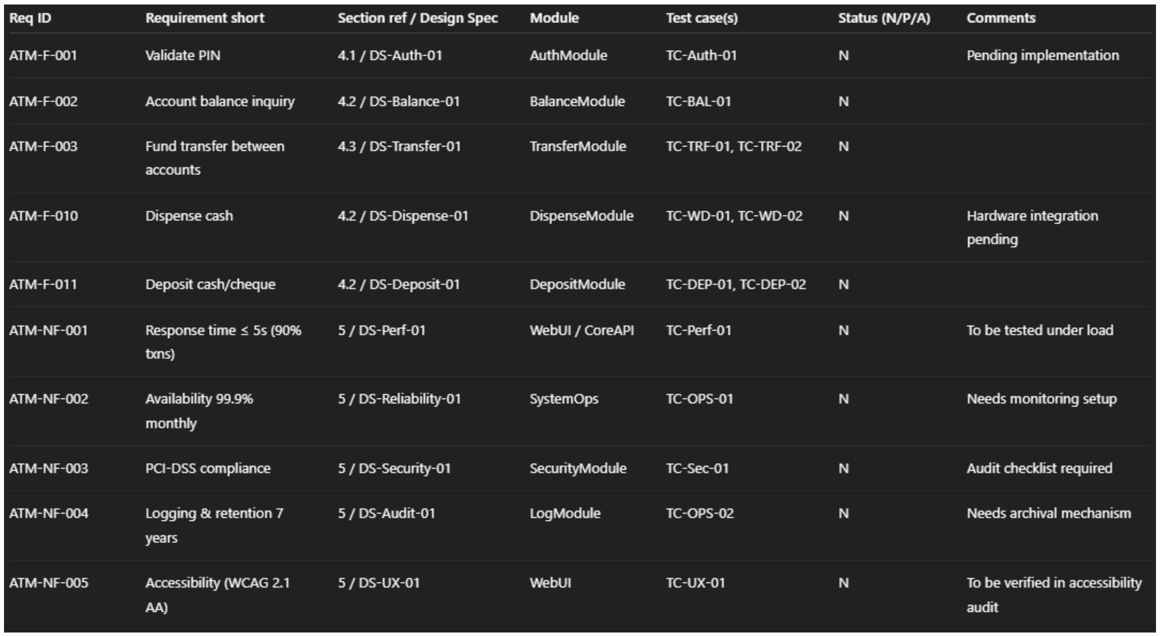
7. System models and diagrams

7.1 UML Use-Case diagram





8. Requirements Traceability Matrix (RTM)



Req ID	Requirement short	Section ref / Design Spec	Module	Test case(s)	Status (N/P/A)	Comments
ATM-F-001	Validate PIN	4.1 / DS-Auth-01	AuthModule	TC-Auth-01	N	Pending implementation
ATM-F-002	Account balance inquiry	4.2 / DS-Balance-01	BalanceModule	TC-BAL-01	N	
ATM-F-003	Fund transfer between accounts	4.3 / DS-Transfer-01	TransferModule	TC-TRF-01, TC-TRF-02	N	
ATM-F-010	Dispense cash	4.2 / DS-Dispense-01	DispenseModule	TC-WD-01, TC-WD-02	N	Hardware integration pending
ATM-F-011	Deposit cash/cheque	4.2 / DS-Deposit-01	DepositModule	TC-DEP-01, TC-DEP-02	N	
ATM-NF-001	Response time ≤ 5s (90% txns)	5 / DS-Perf-01	WebUI / CoreAPI	TC-Perf-01	N	To be tested under load
ATM-NF-002	Availability 99.9% monthly	5 / DS-Reliability-01	SystemOps	TC-OPS-01	N	Needs monitoring setup
ATM-NF-003	PCI-DSS compliance	5 / DS-Security-01	SecurityModule	TC-Sec-01	N	Audit checklist required
ATM-NF-004	Logging & retention 7 years	5 / DS-Audit-01	LogModule	TC-OPS-02	N	Needs archival mechanism
ATM-NF-005	Accessibility (WCAG 2.1 AA)	5 / DS-UX-01	WebUI	TC-UX-01	N	To be verified in accessibility audit