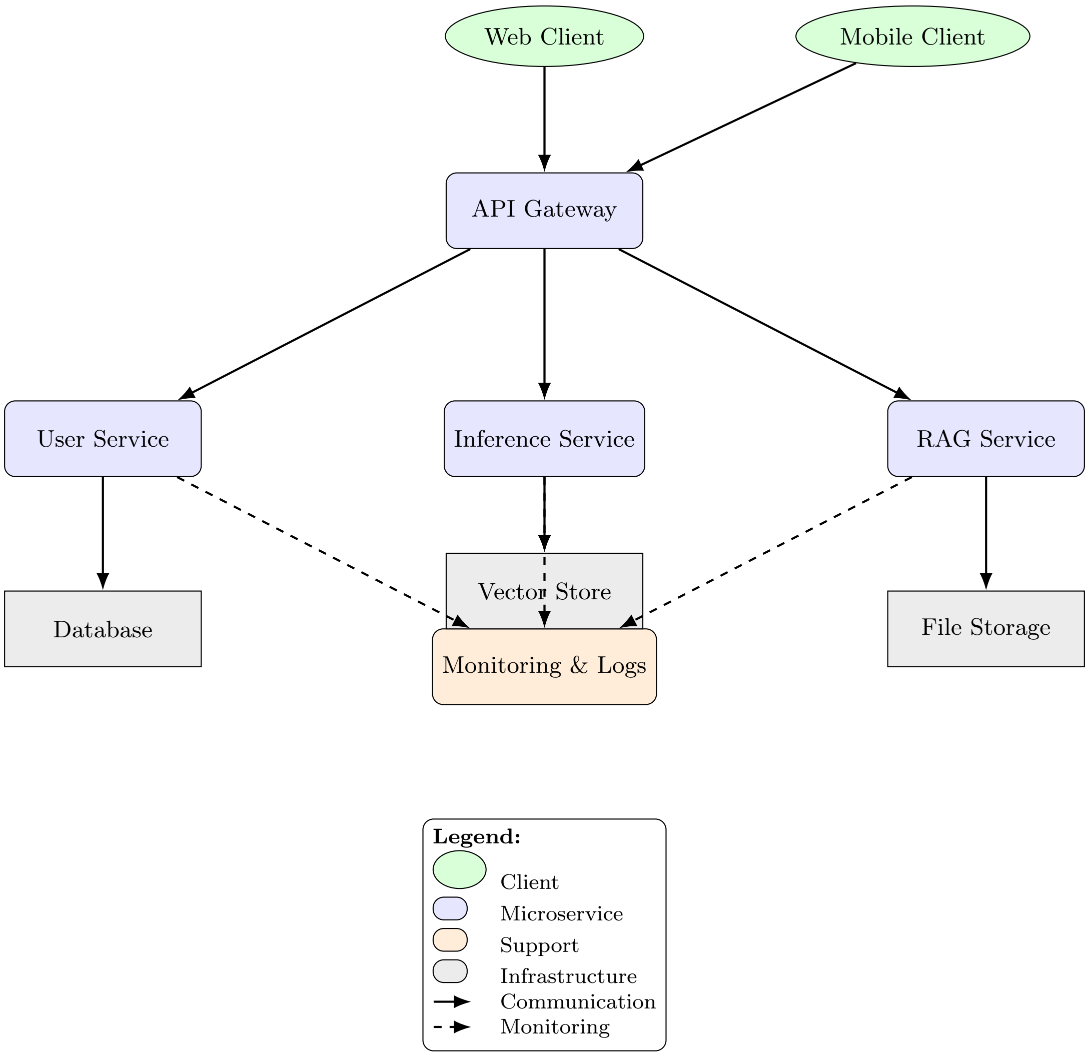
Simplified Microservices Architecture – DeepSeek



1 API Gateway Analysis

Role and Definition

The API Gateway acts as a single entry point for clients. It centralizes access to microservices and ensures critical functions such as routing, security, and data aggregation.

Main Responsibilities

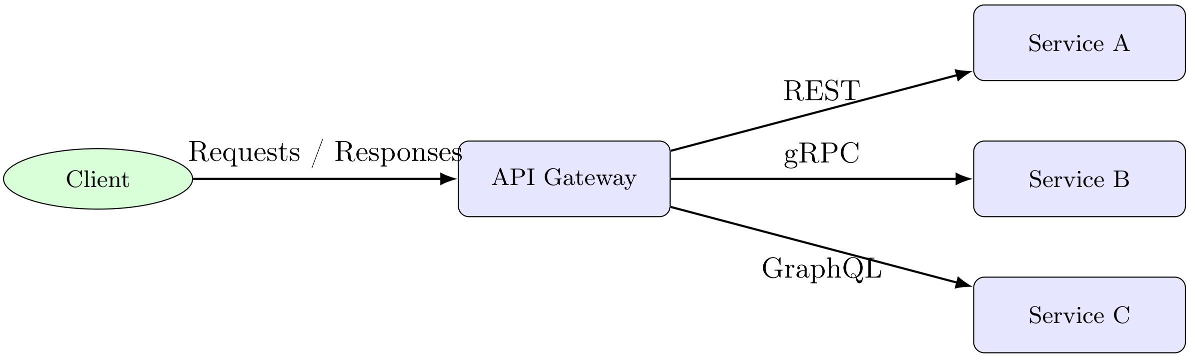
- Request routing to microservices.
- Aggregation of responses from multiple services.
- Authentication and authorization (OAuth2, JWT).
- Protocol transformation (REST, gRPC, WebSocket).
- Metrics and logs collection for monitoring.
- Rate limiting and throttling.

Advantages

- Simplifies client access with a single entry point.
- Centralizes security and global policies.
- Supports response aggregation and improves performance.
- Facilitates scalability and modularity.

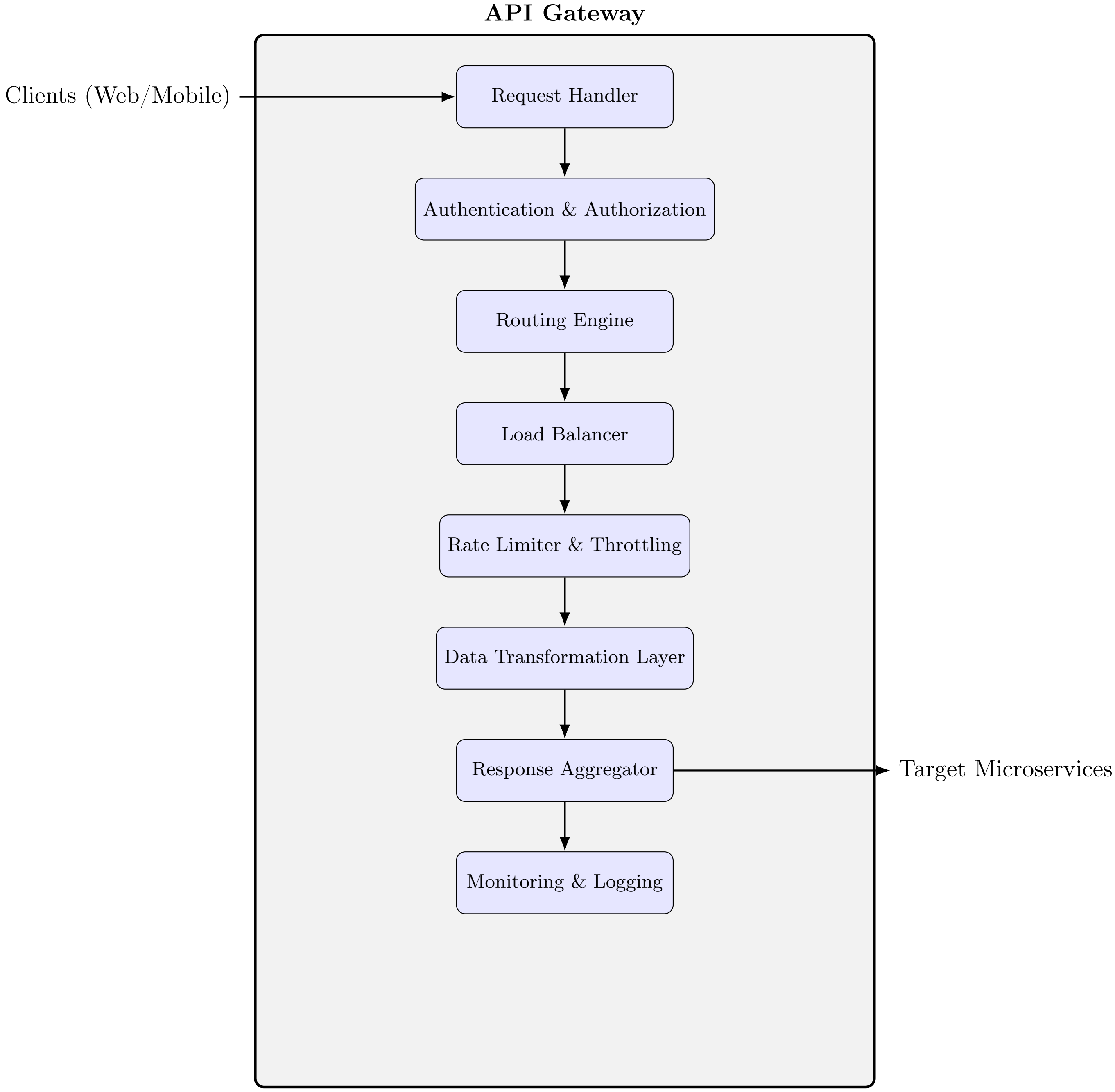
Limitations

- Risk of **Single Point of Failure**.
- Additional latency.
- Complexity in configuration and maintenance.



figureUML diagram of the API Gateway and its interactions.

2 Internal Architecture of the API Gateway



3 Detailed Analysis — Internal Architecture of the API Gateway

3.1 Objectives and Requirements

The API Gateway plays a central role in controlling and orchestrating client access to DeepSeek's microservices.

Functional Objectives:

- Provide a unified and stable entry point for all clients (Web, Mobile, Third-party APIs).
- Authenticate and authorize requests.
- Route requests to the appropriate services and aggregate responses when necessary.
- Perform protocol and data format transformations (REST ↔ gRPC, JSON, GraphQL).

- Enforce policies (rate limiting, quotas, canary routing).

Non-Functional Requirements:

- High availability (replicas, multi-zone deployment).
- Low latency and ability to handle traffic spikes.
- Observability (metrics, logs, traces).
- Security (TLS, validation, WAF).
- Easy configuration and policy enforcement.

3.2 Internal Components — Detailed Description

Request Handler : entry point receiving HTTP/WS requests, extracting headers, body, correlation IDs; performs syntactic validations (size, JSON schema) and applies basic protections (payload limits).

Authentication & Authorization : verifies JWT/OAuth2 (signature via JWKS) or opaque tokens (introspection with an Auth Service). Enforces scope/role checks (RBAC/ABAC).

Rate Limiter & Throttling : applies quotas per key (IP, API key, userId). Common implementations: token-bucket or leaky-bucket, atomic counters in Redis (Lua scripts) for cluster safety.

Input Validation & Sanitization : schema validation, header sanitization, normalization (unicode, trim), injection protection.

Routing Engine / Service Discovery : maps requests (path, host, header) to target services. Integrates discovery (Consul / Kubernetes API) and routing rules (versions, canary, A/B testing).

Load Balancer / Upstream Manager : selects upstream instance (RoundRobin, least-connections, weighted), manages persistent connections (keep-alive), handles timeouts.

Circuit Breaker & Retry Policy : protects against degraded upstreams. Key parameters: failure threshold, time window, reset timeout, exponential backoff retry logic.

Protocol / Data Transformation : converts REST ↔ gRPC, shapes JSON, maps schemas, enriches responses, strips sensitive fields before sending.

Response Aggregator / Composer : orchestrates parallel service calls, merges results, supports partial responses and fallback strategies.

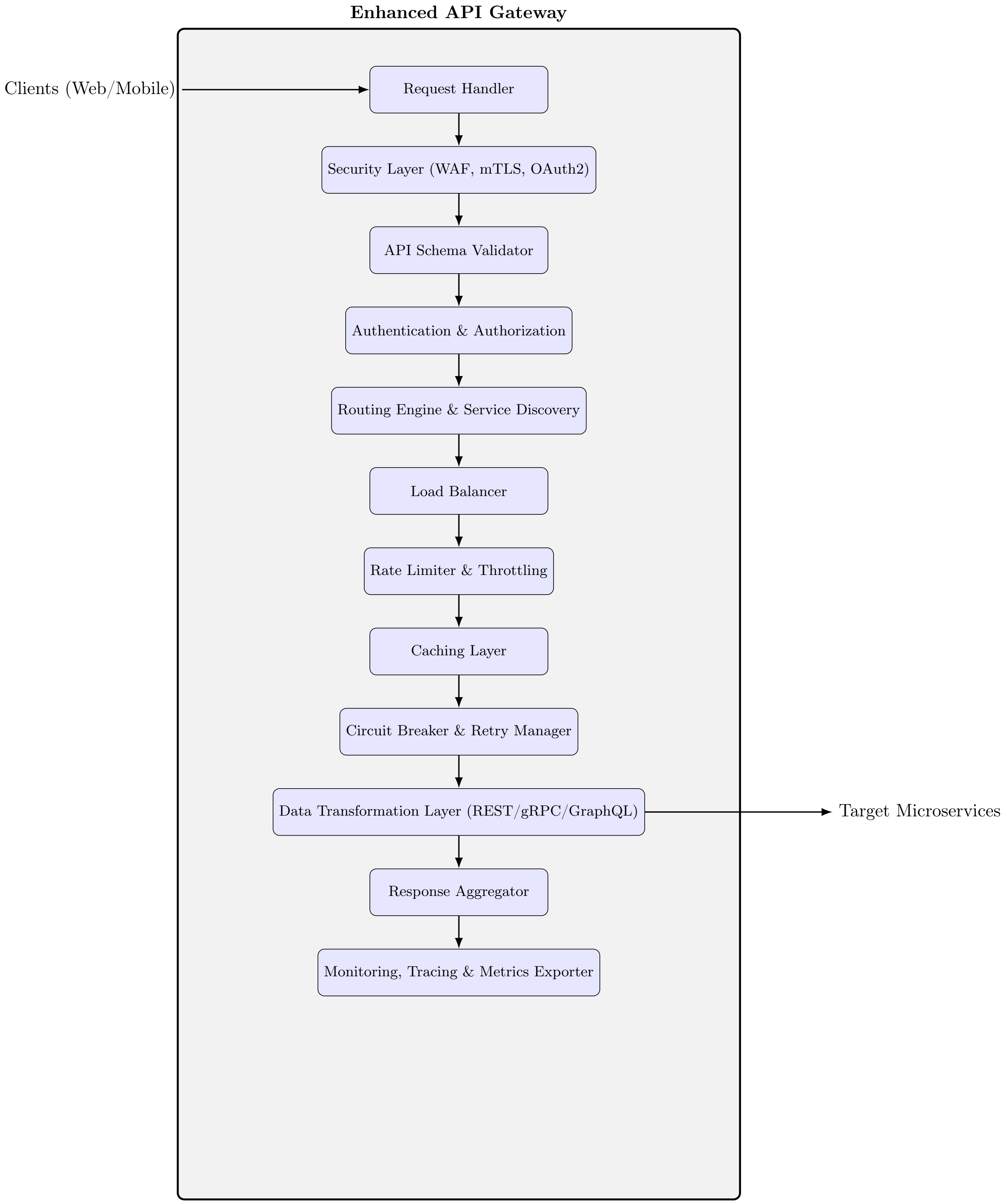
Monitoring & Logging : exports structured logs, traces, and metrics (Prometheus, OpenTelemetry) for observability.

4 Proposed Improvements for the API Gateway

The **DeepSeek API Gateway** can be enhanced to improve security, performance, and governance. Key additions include:

- **Security:** WAF (Web Application Firewall), mTLS support, OAuth2 validation.
- **Performance:** distributed caching, intelligent load balancing, circuit breaker and retry logic.
- **Observability:** distributed tracing (OpenTelemetry), advanced monitoring, Prometheus metrics.
- **Flexibility:** multi-protocol support (REST, gRPC, GraphQL), data aggregation and advanced transformation.
- **Governance:** API schema validation (OpenAPI/JSON Schema), quotas, versioning, and service discovery.

Enhanced Internal Architecture of the API Gateway


Enhanced API Gateway
Clients (Web/Mobile)
Request Handler
Security Layer (WAF, mTLS, OAuth2)
API Schema Validator
Authentication & Authorization
Routing Engine & Service Discovery
Load Balancer
Rate Limiter & Throttling
Caching Layer
Circuit Breaker & Retry Manager
Data Transformation Layer (REST/gRPC/GraphQL)
Target Microservices
Response Aggregator
Monitoring, Tracing & Metrics Exporter

5 Client Request Execution Scenario - DeepSeek API Gateway

Step-by-Step Scenario

1. **Client Request:** Web/Mobile client sends a request (REST/GraphQL) with JWT/OAuth2 token.

2. **Request Handler:** Validates headers, payload, and assigns correlation ID.
3. **Security Layer:** WAF blocks malicious patterns, mTLS ensures client identity, OAuth2 token validation.
4. **API Schema Validator:** Validates request against OpenAPI/JSON Schema.
5. **Authentication & Authorization:** Checks RBAC/ABAC, denies unauthorized requests.
6. **Routing Engine & Service Discovery:** Maps request to appropriate microservices.
7. **Load Balancer:** Selects upstream instances, manages connections and timeouts.
8. **Rate Limiter & Throttling:** Applies per-user/API key quotas.
9. **Caching Layer:** Returns cached response if available, otherwise forwards request.
10. **Circuit Breaker & Retry:** Protects upstream services, retries failed calls, applies fallback if needed.
11. **Data Transformation Layer:** Converts REST ↔ gRPC/GraphQL, enriches request.
12. **Microservice Calls:** Calls User, Inference, and RAG services in parallel.
13. **Response Aggregator:** Combines results into a unified response, handles partial failures.
14. **Monitoring & Metrics:** Logs request/response, updates traces and metrics.
15. **Client Response:** Sends aggregated response back to the client securely (TLS).