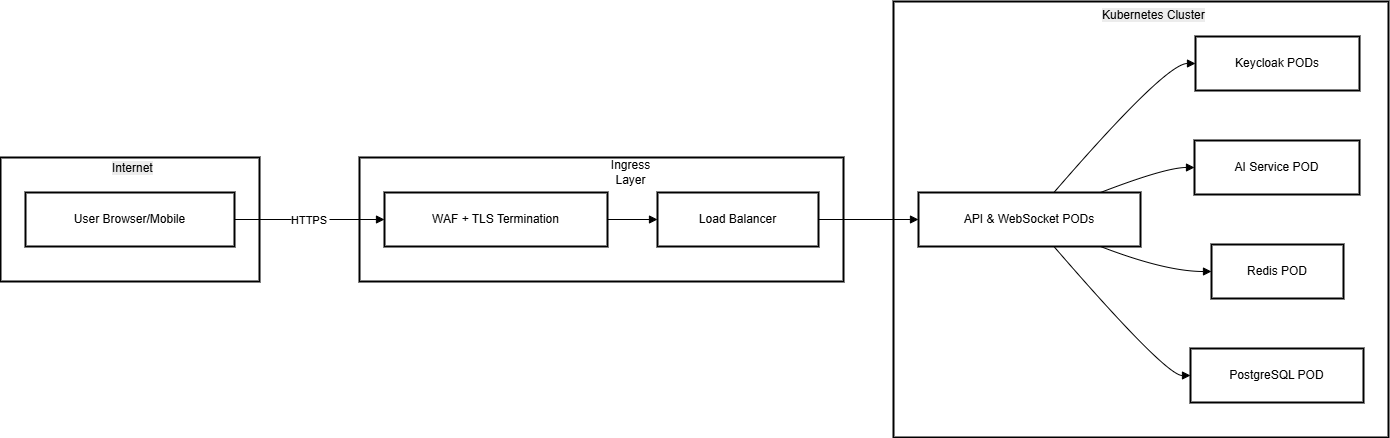
**Low‑Level Technical Specification: Security** Includes network, application, data, and operational security controls with illustrative Mermaid.js diagrams.

## **1. Network Security**

* **TLS Everywhere**
  + Enforce HTTPS/TLS 1.2+ on all ingress points (API, WebSocket, Frontend, Keycloak).
  + Use Let’s Encrypt for automatic certificate issuance and renewal (Cert‑Manager on Kubernetes or Certbot on NGINX).
* **Ingress/Egress Controls**
  + Kubernetes NetworkPolicies or Security Groups to restrict inbound traffic to:  
    - Ports 80/443 (ingress) to frontend/API load balancer.
    - Port 6379 (Redis) only from backend pods.
    - PostgreSQL port 5432 only from app server pods or pgbouncer.
* **WAF (Optional)**
  + Deploy OWASP CRS–enabled ModSecurity in front of API.



## **2. Authentication & Authorization Security**

* **JWT Validation**
  + Validate signature using JWKS endpoint from Keycloak.
  + Enforce aud, iss, and expire (exp) claims.
* **Role Checks**
  + Implement requireRole() middleware for each API route.
* **Secret Management**
  + Store JWKS URI, JWT public keys, JWT\_SECRET, Keycloak client secrets in Kubernetes Secrets or Vault.
* **PKCE & OIDC**
  + Enforce Proof Key for Code Exchange on public clients.

## **3. Data Security**

* **Encryption at Rest**
  + PostgreSQL: enable pgcrypto or TDE for sensitive columns (emails, secrets).
  + Redis: enable rdb-encryption or run in a private network only.
* **Encryption in Transit**
  + Use TLS for all intra‑cluster communication (mTLS via service mesh, e.g. Istio, optional).
* **Database Hardening**
  + Use separate DB roles (app\_readwrite, app\_readonly).
  + Disable default postgres remote login.

## **4. Application Security**

* **Input Validation & Sanitization**
  + Validate all REST/WS payloads with Joi/Zod.
  + Sanitize text fields to prevent injection (e.g., HTML escape in comments).
* **OWASP Top 10 Protections**
  + **XSS**: Content Security Policy; escape user input.
  + **CSRF**: stateless API, JWT in header; no cookies.
  + **Injection**: use parameterized queries (ORM/Knex).
  + **Security Headers**: CSP, X-Frame-Options, HSTS, X-Content-Type-Options.

## **5. Container & CI/CD Security**

* **Container Hardening**
  + Use minimal base images (e.g., Alpine).
  + Run processes as non‑root user.
  + Scan images with Trivy in CI pipeline.
* **CI/CD Pipeline**
  + Require signed commits or PR approvals.
  + Automated vulnerability scanning (npm audit, snyk).
  + Deploy only on passing all security gates.

## **6. Secrets Management**

* **Kubernetes Secrets** or HashiCorp Vault for:  
  + Database passwords, OAuth client secrets, JWT keys, Redis passwords.
* **Secret Rotation**
  + Rotate every 90 days; automate via Vault dynamic secrets (if Vault used).

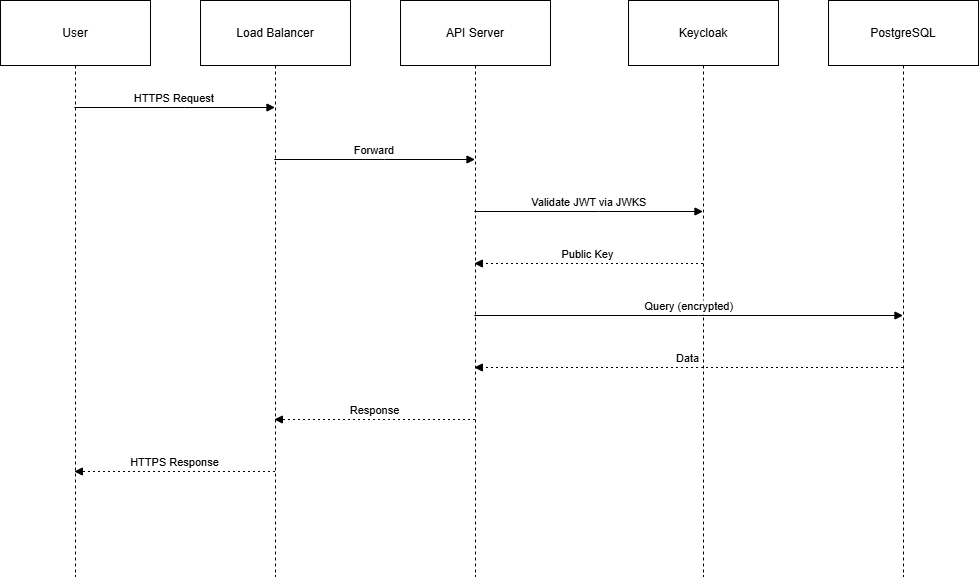
## **7. Logging & Monitoring**

* **Audit Logging**
  + Log security-relevant events: login, logout, token refresh, failed auth attempts, permission denials to audit\_logs in DB.
* **Centralized Logs**
  + Use Fluentd/Promtail to ship logs to Loki or ELK.
* **Alerting**
  + Prometheus Alertmanager rules for:  
    - High 401/403 rates.
    - Suspicious number of failed login attempts.

## **8. Incident Response & Recovery**

* **WAF Blocking**
  + Automated block on detected OWASP CRS rules.
* **Rate Limiting**
  + API Gateway rate limits (e.g., 100 requests/min per IP).
* **Fallback & Graceful Degradation**
  + Read-only mode if write DB unreachable; queue writes in Redis for replay.
* **Forensics**
  + Retain logs for 90 days; enable audit trail in DB.

## **9. Diagram: Security Event Flow**



*End of Security Specification*