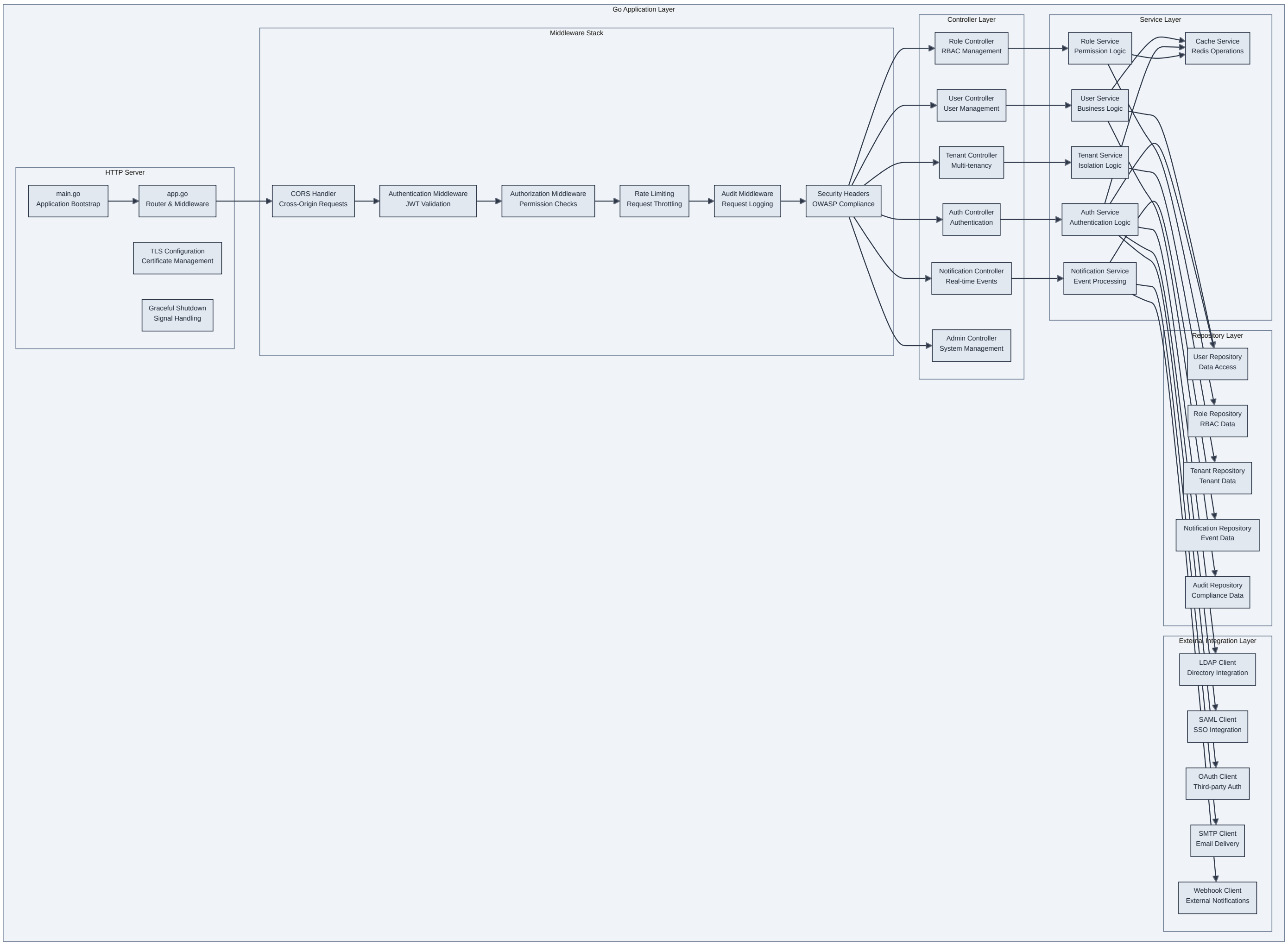
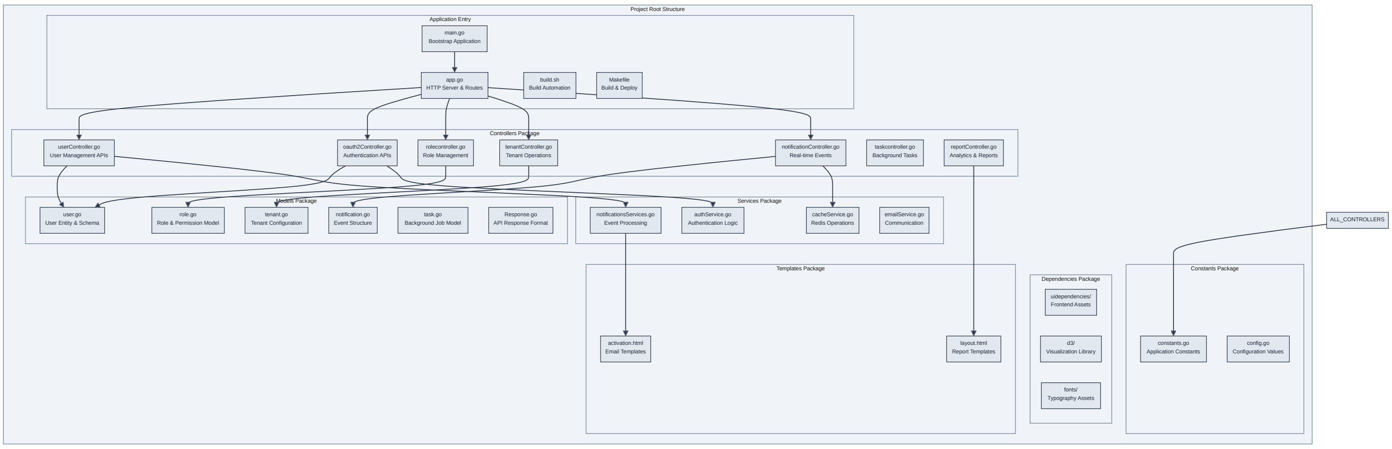
Securaa User Service - Low Level Design

🔧 TECHNICAL ARCHITECTURE OVERVIEW

Service Implementation Architecture

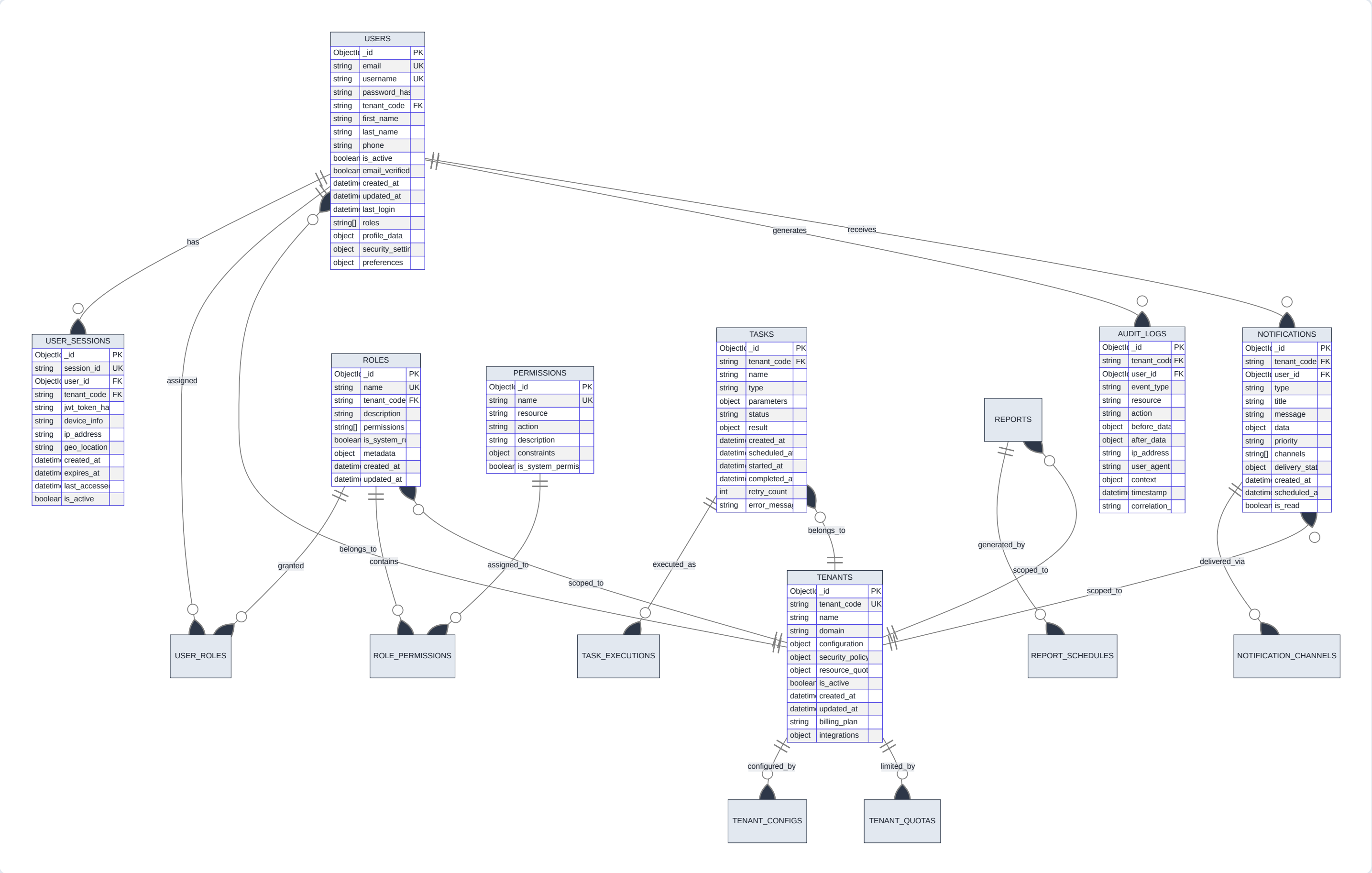


Code Structure & Package Organization



🏗️ DATABASE DESIGN & DATA MODELS

MongoDB Collection Schema Architecture



USERS

ObjectI	_id	PK
string	email	UK
string	username	UK
string	password_has	
string	tenant_code	FK
string	first_name	
string	last_name	
string	phone	
boolean	is_active	
boolean	email_verified	
datetim	created_at	
datetim	updated_at	
datetim	last_login	
string[]	roles	
object	profile_data	
object	security_settir	
object	preferences	

USER_SESSIONS

ObjectI	_id	PK
string	session_id	UK
ObjectI	user_id	FK
string	tenant_code	FK
string	jwt_token_ha	
string	device_info	
string	ip_address	
string	geo_location	
datetim	created_at	
datetim	expires_at	
datetim	last_accesse	
boolean	is_active	

ROLES

ObjectI	_id	PK
string	name	UK
string	tenant_code	FK
string	description	
string[]	permissions	
boolean	is_system_r	
object	metadata	
datetim	created_at	
datetim	updated_at	

PERMISSIONS

ObjectI	_id	PK
string	name	UK
string	resource	
string	action	
string	description	
object	constraints	
boolear	is_system_permis	

TASKS

ObjectI	_id	PK
string	tenant_code	FK
string	name	
string	type	
object	parameters	
string	status	
object	result	
datetim	created_at	
datetim	scheduled_a	
datetim	started_at	
datetim	completed_a	
int	retry_count	
string	error_messa	

AUDIT_LOGS

ObjectI	_id	PK
string	tenant_code	FK
ObjectI	user_id	FK
string	event_type	
string	resource	
string	action	
object	before_data	
object	after_data	
string	ip_address	
string	user_agent	
object	context	
datetim	timestamp	
string	correlation_	

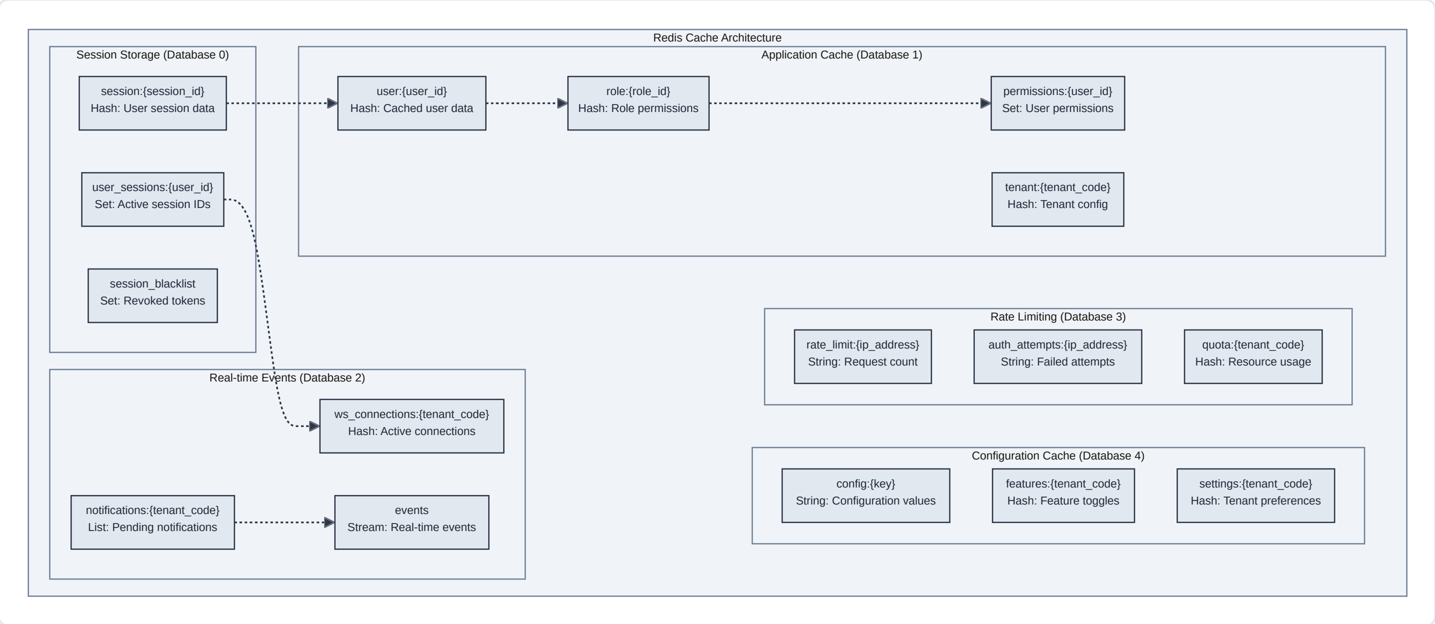
NOTIFICATIONS

ObjectI	_id	PK
string	tenant_code	FK
ObjectI	user_id	FK
string	type	
string	title	
string	message	
object	data	
string	priority	
string[]	channels	
object	delivery_stat	
datetim	created_at	
datetim	scheduled_a	
boolean	is_read	

TENANTS

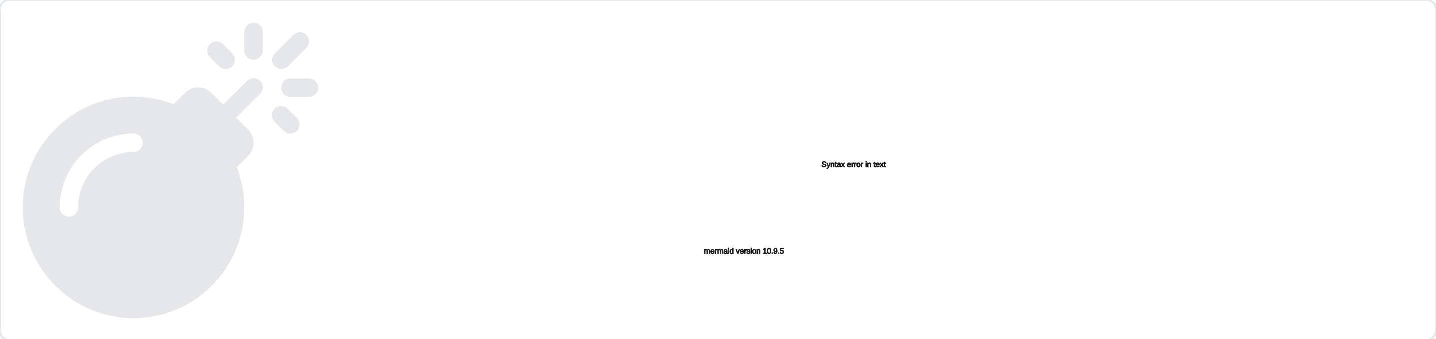
ObjectI	_id	PK
string	tenant_code	UK
string	name	
string	domain	
object	configuration	
object	security_policy	
object	resource_quot	
boolean	is_active	
datetim	created_at	
datetim	updated_at	
string	billing_plan	
object	integrations	

Redis Cache Schema Design

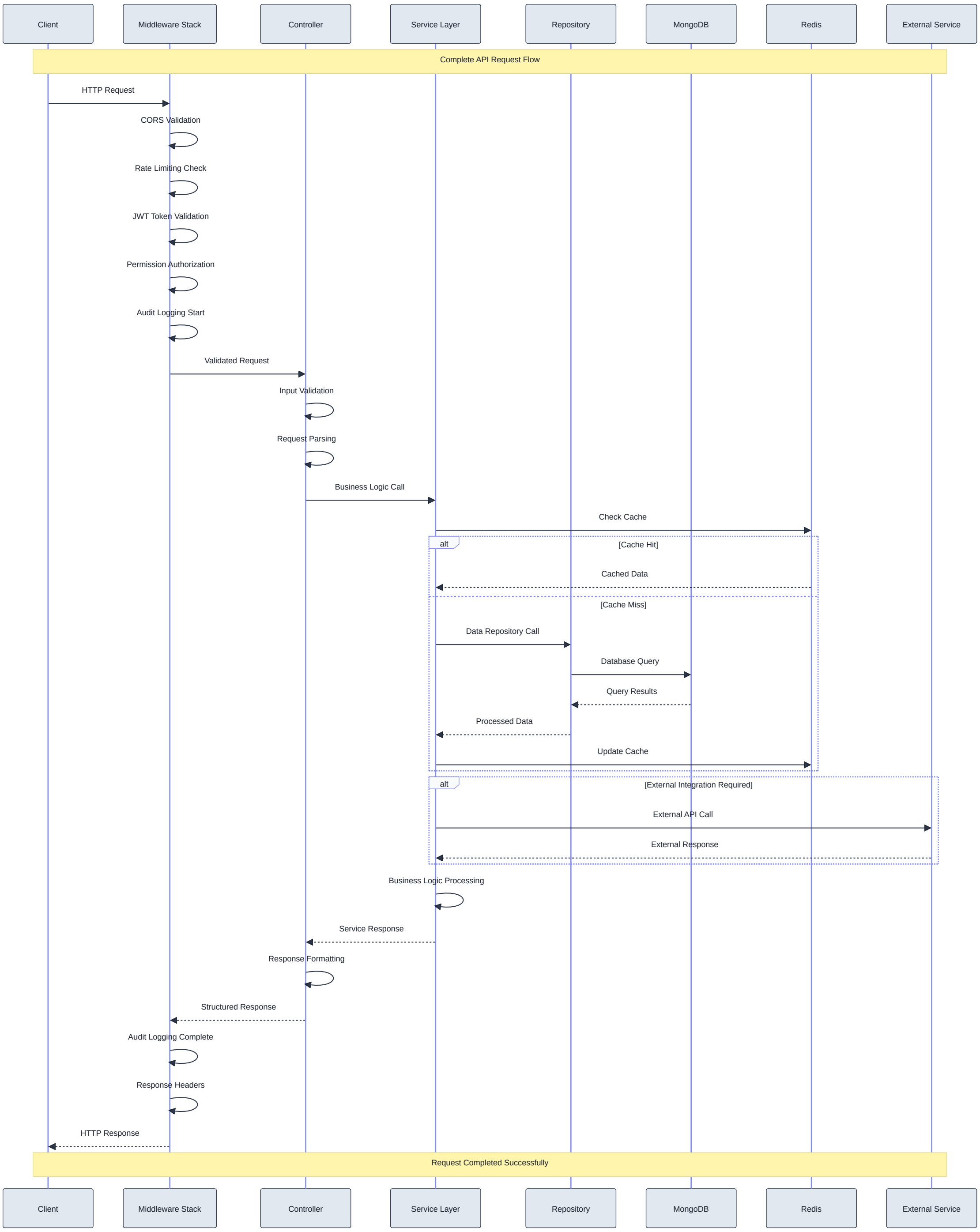


🔗 API DESIGN & ENDPOINT SPECIFICATIONS

RESTful API Architecture

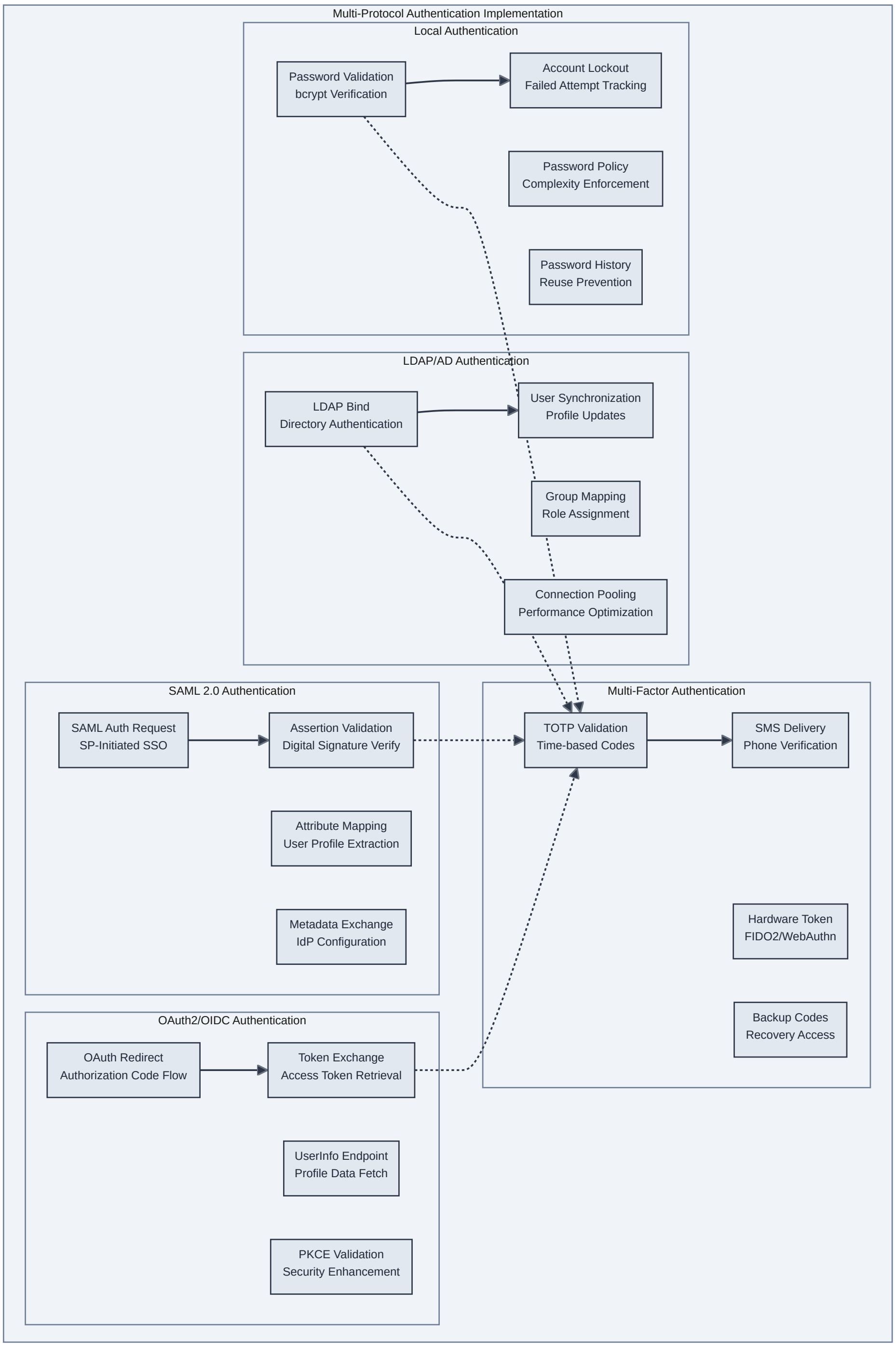


API Request/Response Flow Diagram

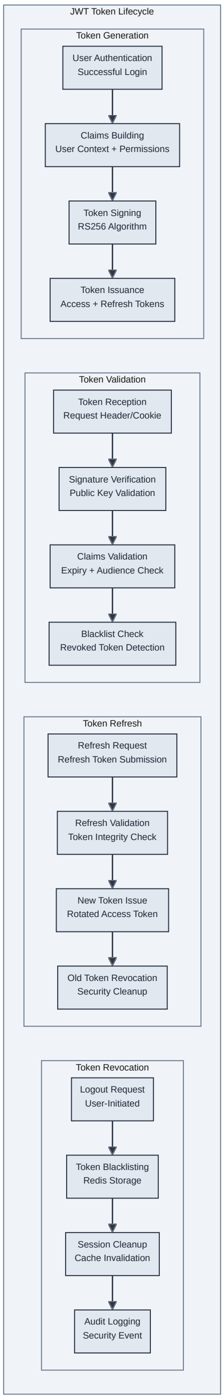


🔒 SECURITY IMPLEMENTATION DETAILS

Authentication & Authorization Flow



JWT Token Management Implementation


JWT Token Lifecycle
Token Generation
User Authentication Successful Login
Claims Building User Context + Permissions
Token Signing RS256 Algorithm
Token Issuance Access + Refresh Tokens
Token Validation
Token Reception Request Header/Cookie
Signature Verification Public Key Validation
Claims Validation Expiry + Audience Check
Blacklist Check Revoked Token Detection
Token Refresh
Refresh Request Refresh Token Submission
Refresh Validation Token Integrity Check
New Token Issue Rotated Access Token
Old Token Revocation Security Cleanup
Token Revocation
Logout Request User-Initiated
Token Blacklisting Redis Storage
Session Cleanup Cache Invalidation
Audit Logging Security Event

Role-Based Access Control (RBAC) Implementation

Syntax error in text

mermaid version 10.9.5

🚀 DEPLOYMENT & INFRASTRUCTURE IMPLEMENTATION

Traditional Server Deployment Architecture

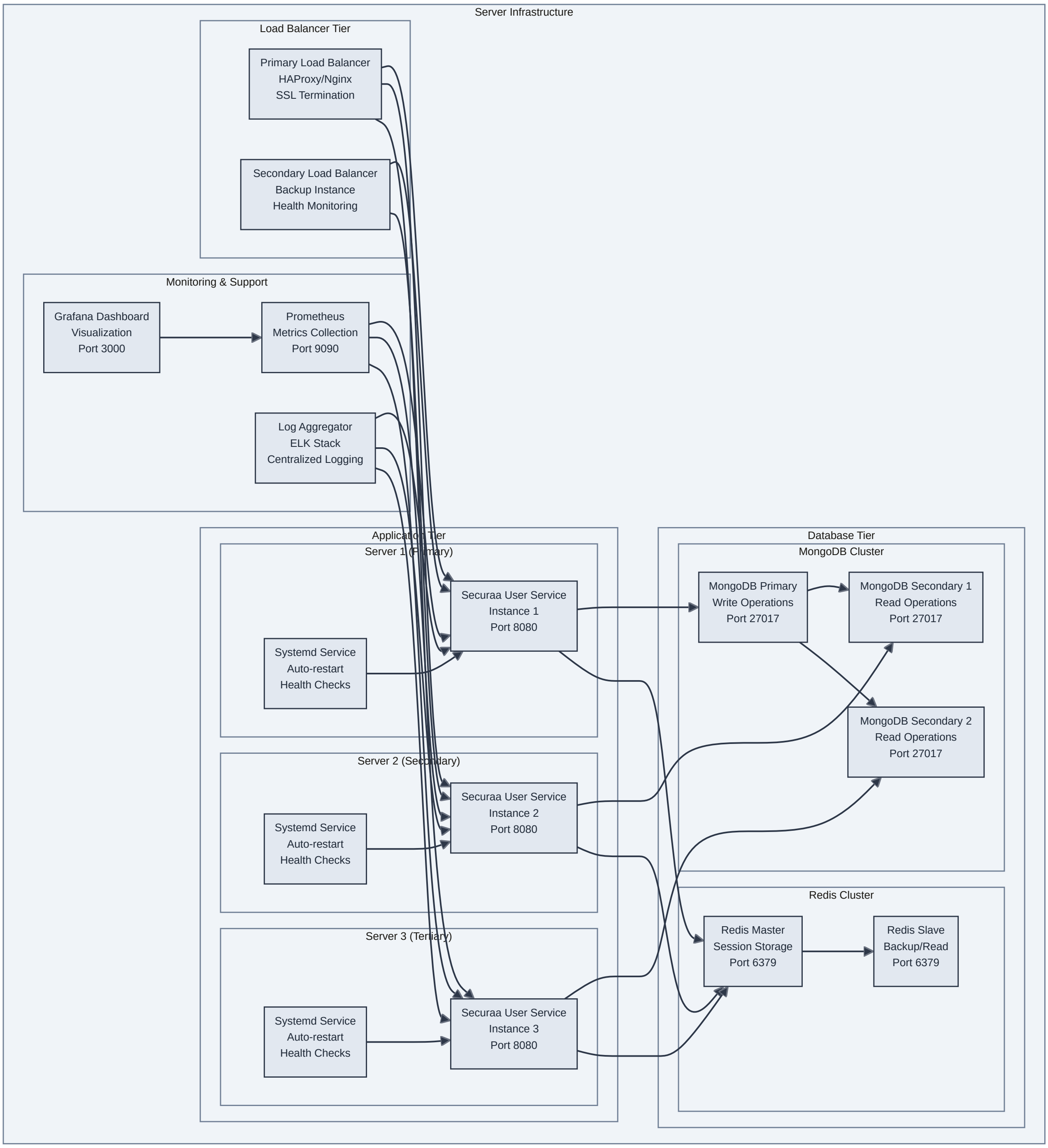
The Securaa User Service is deployed using traditional server architecture with load balancing and service clustering for high availability and scalability.

Server Deployment Strategy:

Load Balancer Configuration: - **Primary Load Balancer**: HAProxy or Nginx for traffic distribution and SSL termination - **Health Check Integration**: Application-level health endpoints for intelligent routing - **Session Management**: Sticky sessions support for WebSocket connections - **SSL/TLS Termination**: Certificate management and encryption handling at the edge

Service Instance Management: - **Multiple Instances**: 3+ service instances for high availability - **Process Management**: Systemd service files for automatic restart and lifecycle management - **Configuration Management**: Environment-specific configuration files and secrets - **Log Management**: Structured logging with centralized log aggregation

Database Deployment: - **MongoDB Replica Set**: Primary and secondary instances for read scaling and failover - **Redis Cluster**: Master-slave configuration for caching and session storage - **Connection Pooling**: Optimized database connections with configurable pool sizes - **Backup Strategy**: Automated backup schedules with point-in-time recovery



Deployment Process:

Service Installation: 1. **Binary Deployment**: Go binary compilation and distribution to target servers 2. **Configuration Setup**: Environment-specific configuration files and secrets deployment 3. **Service Registration**: Systemd service file installation and enablement 4. **Health Verification**: Automated health checks during deployment process

Database Setup: 1. **MongoDB Replica Set**: Primary and secondary node configuration with authentication 2. **Redis Configuration**: Master-slave setup with persistence and clustering 3. **Index Creation**: Database index optimization for query performance 4. **Data Migration**: Automated migration scripts for schema updates

Load Balancer Configuration: 1. **SSL Certificate Installation**: TLS certificate deployment and renewal automation 2. **Health Check Configuration**: Application-level health endpoint configuration 3. **Traffic Routing Rules**: Request routing based on URL patterns and headers 4. **Failover Configuration**: Automatic failover to healthy instances

Monitoring Setup: 1. **Metrics Collection**: Prometheus metrics endpoint configuration 2. **Dashboard Deployment**: Grafana dashboard installation and configuration 3. **Alerting Rules**: Alert configuration for critical system events 4. **Log Aggregation**: Centralized logging setup with retention policies

Production Deployment Configuration

Systemd Service Configuration:

```
# /etc/systemd/system/securaa-user-service.service
[Unit]
Description=Securaa User Service
Documentation=https://docs.securaa.com/securaa-user
After=network.target mongodb.service redis.service
Wants=mongodb.service redis.service

[Service]
Type=exec
User=securaa-user
Group=securaa-user
WorkingDirectory=/opt/securaa-user
ExecStart=/opt/securaa-user/bin/securaa-user-service
ExecReload=/bin/kill -HUP $MAINPID
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal
SyslogIdentifier=securaa-user-service

# Security Settings
NoNewPrivileges=true
PrivateTmp=true
ProtectSystem=strict
ProtectHome=true
ReadWritePaths=/opt/securaa-user/logs /opt/securaa-user/data

# Environment Variables
Environment=ENVIRONMENT=production
Environment=LOG_LEVEL=info
Environment=HTTP_PORT=8080
Environment=METRICS_PORT=9090
Environment=HEALTH_CHECK_PORT=8081

# Resource Limits
LimitNOFILE=65536
LimitNPROC=4096

[Install]
WantedBy=multi-user.target
```

Environment Configuration:

```
# /opt/securaa-user/config/production.env
# Database Configuration
MONGO_URI=mongodb://securaa-user:${MONGO_PASSWORD}@mongo-primary:27017,mongo-secondary1:27017,mongo-secondary2:27017/zona_user?replicaSet=rs0&authSource=admin
REDIS_URI=redis://:${REDIS_PASSWORD}@redis-master:6379/0

# Security Configuration
JWT_SECRET_KEY=${JWT_SECRET_KEY}
ENCRYPTION_KEY=${ENCRYPTION_KEY}
SAML_CERT_PATH=/opt/securaa-user/certs/saml.crt
SAML_KEY_PATH=/opt/securaa-user/certs/saml.key

# Integration Configuration
EMAIL_SMTP_HOST=smtp.securaa.com
EMAIL_SMTP_PORT=587
EMAIL_FROM=noreply@securaa.com

# Performance Configuration
MAX_CONNECTIONS=1000
CONNECTION_TIMEOUT=30s
IDLE_TIMEOUT=60s
READ_TIMEOUT=30s
WRITE_TIMEOUT=30s

# Monitoring Configuration
METRICS_ENABLED=true
PROMETHEUS_ENDPOINT=/metrics
HEALTH_CHECK_ENDPOINT=/health
LOG_FORMAT=json
LOG_OUTPUT=file
LOG_FILE_PATH=/opt/securaa-user/logs/securaa-user.log
```

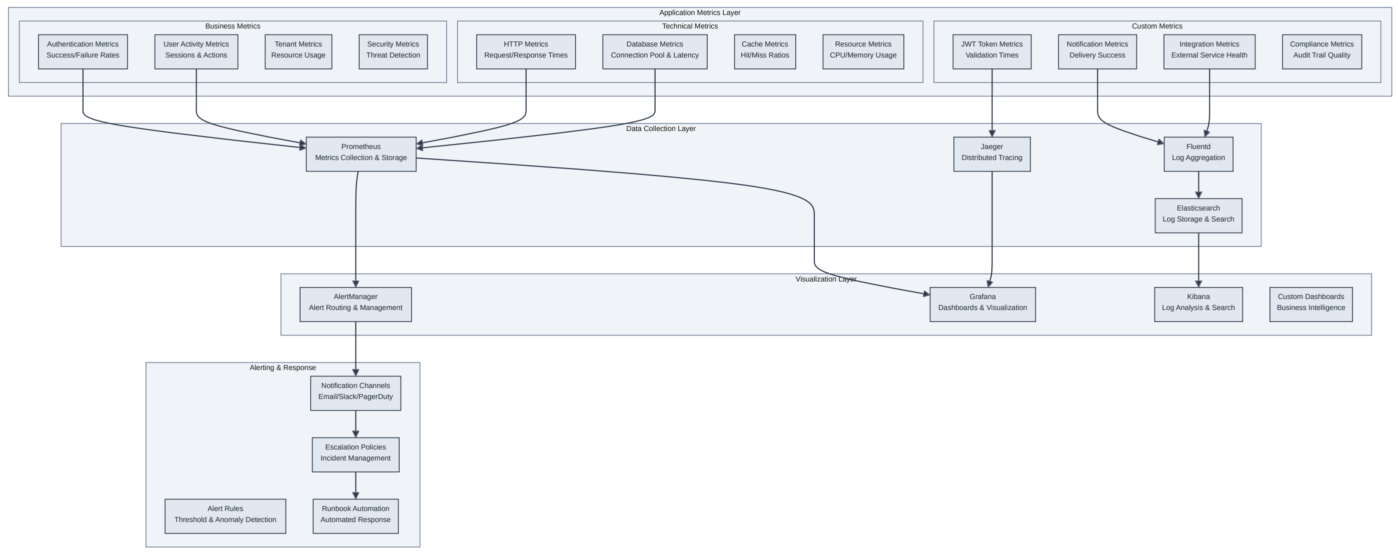
Deployment Script:

```bash
#!/bin/bash
# /opt/securaa-user/scripts/deploy.sh

set -euo pipefail

DEPLOY_DIR="/opt/securaa-user"
SERVICE_NAME="securaa-user-service"
BACKUP_DIR="/opt/securaa-user/backups"
LOG_DIR="/opt/securaa-user/logs"

# Pre-deployment checks
echo "Starting deployment of Securaa User Service..."
echo "Checking system requirements..."

# Verify dependencies
systemctl is-active --quiet mongodb || { echo "MongoDB not running"; exit 1; }
systemctl is-active --quiet redis || { echo "Redis not running"; exit 1; }

# Create backup of current version
if systemctl is-active --quiet $SERVICE_NAME; then
    echo "Creating backup of current deployment..."
    mkdir -p $BACKUP_DIR/$(date +%Y%m%d_%H%M%S)
    cp -r $DEPLOY_DIR/bin $BACKUP_DIR/$(date +%Y%m%d_%H%M%S)/
    cp -r $DEPLOY_DIR/config $BACKUP_DIR/$(date +%Y%m%d_%H%M%S)/
fi

# Deploy new binary
echo "Deploying new binary..."
cp ./securaa-user-service $DEPLOY_DIR/bin/
chmod +x $DEPLOY_DIR/bin/securaa-user-service
chown securaa-user:securaa-user $DEPLOY_DIR/bin/securaa-user-service

# Update configuration
echo "Updating configuration..."
cp ./config/* $DEPLOY_DIR/config/
chown -R securaa-user:securaa-user $DEPLOY_DIR/config/

# Restart service
echo "Restarting service..."
systemctl restart $SERVICE_NAME

# Health check
echo "Performing health check..."
sleep 10

for i in {1..30}; do
    if curl -f http://localhost:8081/health > /dev/null 2>&1; then
        echo "Health check passed!"
        break
    fi
    if [ $i -eq 30 ]; then
        echo "Health check failed! Rolling back..."
        systemctl stop $SERVICE_NAME
        # Restore from backup
        LATEST_BACKUP=$(ls -t $BACKUP_DIR | head -1)
        cp -r $BACKUP_DIR/$LATEST_BACKUP/* $DEPLOY_DIR/
        systemctl start $SERVICE_NAME
        exit 1
    fi
    echo "Waiting for service to start... ($i/30)"
    sleep 2
done

echo "Deployment completed successfully!"
```

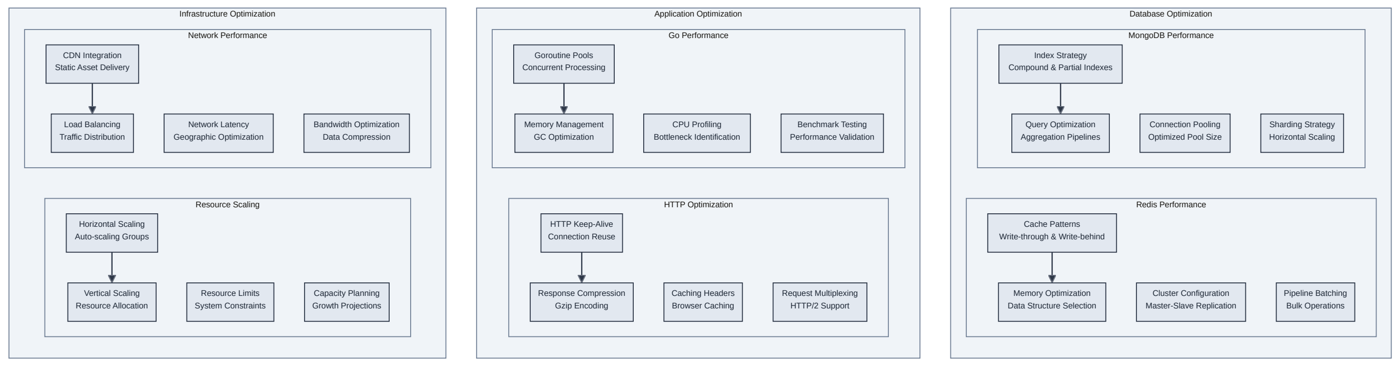
```yaml
  - name: JWT_SECRET
    valueFrom:
      secretKeyRef:
        name: auth-secrets
        key: jwt-secret
  resources:
    requests:
      memory: "512Mi"
      cpu: "500m"
    limits:
      memory: "2Gi"
      cpu: "2000m"
  securityContext:
    allowPrivilegeEscalation: false
    readOnlyRootFilesystem: true
    capabilities:
      drop:
      - ALL
  livenessProbe:
    httpGet:
      path: /health/live
      port: 8000
      scheme: HTTPS
    initialDelaySeconds: 60
    periodSeconds: 30
    timeoutSeconds: 10
    failureThreshold: 3
  readinessProbe:
    httpGet:
      path: /health/ready
      port: 8000
      scheme: HTTPS
    initialDelaySeconds: 30
    periodSeconds: 10
    timeoutSeconds: 5
    failureThreshold: 3
  volumeMounts:
  - name: tls-certificates
    mountPath: /etc/ssl/certs
    readOnly: true
  - name: temp-storage
    mountPath: /tmp
  - name: config-volume
    mountPath: /app/config
    readOnly: true
volumes:
- name: tls-certificates
  secret:
    secretName: securaa-user-tls
- name: temp-storage
  emptyDir: {}
- name: config-volume
  configMap:
    name: securaa-user-config
```

📊 MONITORING & OBSERVABILITY IMPLEMENTATION

Comprehensive Monitoring Architecture

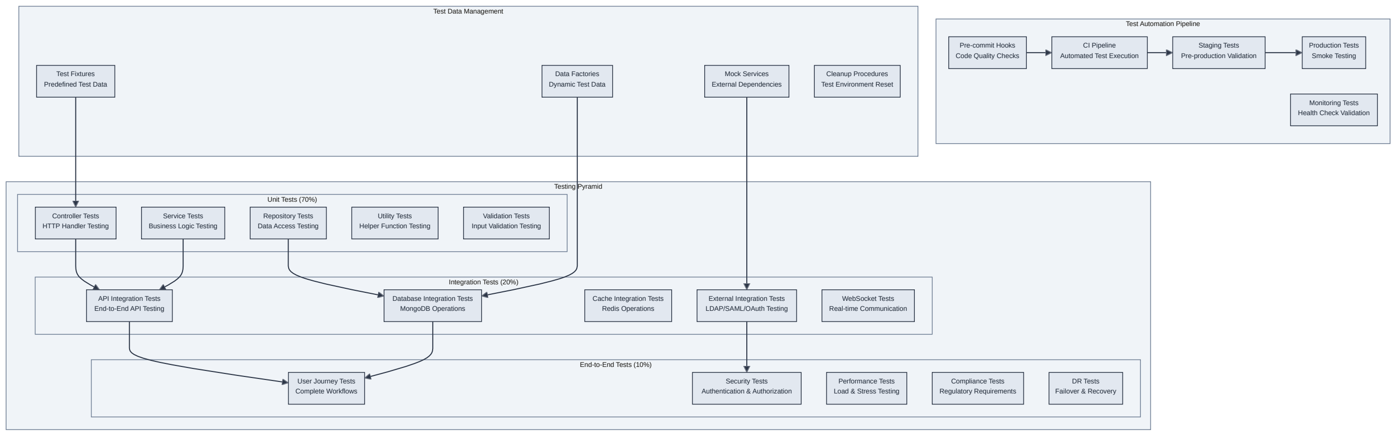


Performance Optimization Strategies

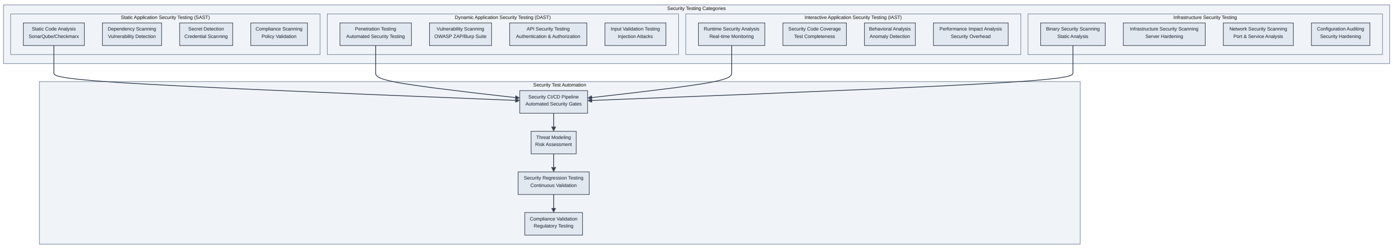


🧪 TESTING STRATEGY & IMPLEMENTATION

Comprehensive Testing Pyramid



Security Testing Implementation



🔧 CONFIGURATION MANAGEMENT

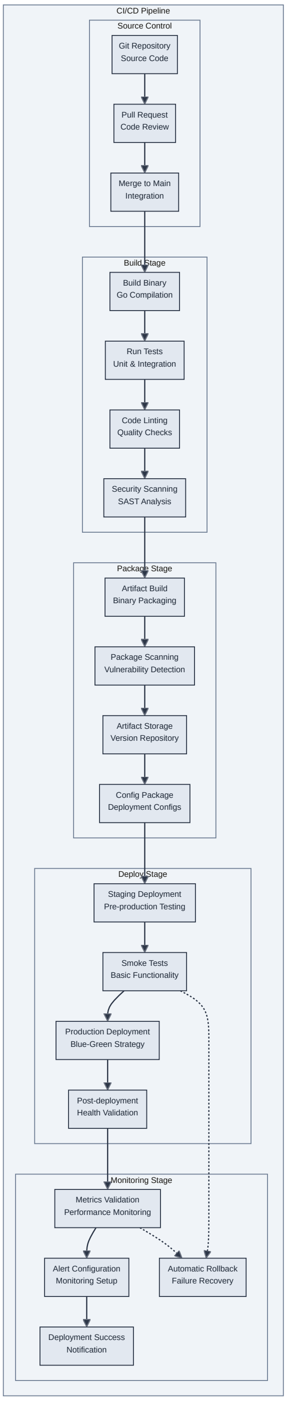
Environment Configuration Strategy

```
# Production Configuration Template
production:
  server:
    port: 8000
    tls:
      enabled: true
      cert_file: "/etc/ssl/certs/server.crt"
      key_file: "/etc/ssl/private/server.key"
      min_version: "1.2"
      max_version: "1.3"
      cipher_suites:
        - "TLS_AES_256_GCM_SHA384"
        - "TLS_CHACHA20_POLY1305_SHA256"
        - "TLS_AES_128_GCM_SHA256"

  database:
    mongodb:
      uri: "${MONGO_URI}"
      database: "securaa_production"
      max_pool_size: 200
      min_pool_size: 50
      connect_timeout_ms: 30000
      server_selection_timeout_ms: 30000
      socket_timeout_ms: 60000
      max_idle_time_ms: 300000

  cache:
    redis:
      cluster_endpoints:
        - "${REDIS_CLUSTER_1}"
        - "${REDIS_CLUSTER_2}"
        - "${REDIS_CLUSTER_3}"
      password: "${REDIS_PASSWORD}"
      max_retries: 3
      retry_delay_ms: 1000
      dial_timeout_ms: 5000
      read_timeout_ms: 3000
      write_timeout_ms: 3000
      pool_size: 100
      min_idle_connections: 20

  security:
    jwt:
      secret: "${JWT_SECRET}"
      issuer: "securaa-user-service"
      audience: "securaa-platform"
      access_token_ttl: "15m"
      refresh_token_ttl: "7d"
      signing_method: "RS256"

    encryption:
      key: "${ENCRYPTION_KEY}"
      algorithm: "AES-256-GCM"

    session:
      timeout: "4h"
      max_concurrent_sessions: 3
      secure_cookies: true
      same_site: "strict"

    password_policy:
      min_length: 12
      require_uppercase: true
      require_lowercase: true
      require_numbers: true
      require_special_chars: true
      prevent_reuse_count: 12
      max_age_days: 90

  integrations:
    ldap:
      servers:
        - "${LDAP_SERVER_1}"
        - "${LDAP_SERVER_2}"
      base_dn: "${LDAP_BASE_DN}"
      bind_dn: "${LDAP_BIND_DN}"
      bind_password: "${LDAP_BIND_PASSWORD}"
      user_search_filter: "(uid=%s)"
      group_search_filter: "(memberUid=%s)"
      tls_enabled: true
      skip_cert_verify: false

    saml:
      entity_id: "securaa-user-service"
      assertion_consumer_service_url: "https://api.securaa.com/auth/saml/acs"
      single_logout_service_url: "https://api.securaa.com/auth/saml/sls"
      metadata_url: "${SAML_METADATA_URL}"
      certificate_file: "/etc/ssl/certs/saml.crt"
      private_key_file: "/etc/ssl/private/saml.key"

    oauth2:
      providers:
        google:
          client_id: "${GOOGLE_CLIENT_ID}"
          client_secret: "${GOOGLE_CLIENT_SECRET}"
          redirect_url: "https://api.securaa.com/auth/oauth/google/callback"
          scopes: ["openid", "profile", "email"]
        microsoft:
          client_id: "${MICROSOFT_CLIENT_ID}"
          client_secret: "${MICROSOFT_CLIENT_SECRET}"
          redirect_url: "https://api.securaa.com/auth/oauth/microsoft/callback"
          scopes: ["openid", "profile", "email"]

    smtp:
      host: "${SMTP_HOST}"
      port: 587
      username: "${SMTP_USERNAME}"
      password: "${SMTP_PASSWORD}"
      from_address: "noreply@securaa.com"
      use_tls: true
      connection_pool_size: 10

    sms:
      provider: "twilio"
      account_sid: "${TWILIO_ACCOUNT_SID}"
      auth_token: "${TWILIO_AUTH_TOKEN}"
      from_number: "${TWILIO_FROM_NUMBER}"

  monitoring:
    metrics:
      enabled: true
      port: 9090
      path: "/metrics"

    logging:
      level: "info"
      format: "json"
      output: "stdout"

    tracing:
      enabled: true
      jaeger:
        endpoint: "${JAEGER_ENDPOINT}"
        sampler_type: "probabilistic"
        sampler_param: 0.1

    health_checks:
      enabled: true
      endpoints:
        - path: "/health"
          method: "GET"
        - path: "/health/ready"
          method: "GET"
        - path: "/health/live"
          method: "GET"

  compliance:
    audit_logging:
      enabled: true
      retention_days: 2555  # 7 years
      encryption_enabled: true
      remote_storage: true

    data_protection:
      gdpr_enabled: true
      data_retention_days: 1825  # 5 years
      anonymization_enabled: true
      right_to_be_forgotten: true

    regulatory:
```

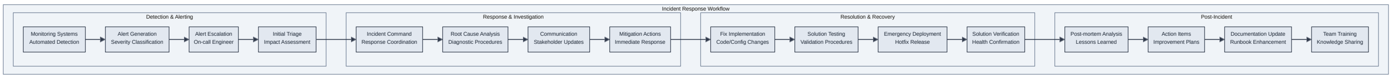
```
    soc2_mode: true
    hipaa_mode: true
    pci_dss_mode: true
    iso27001_mode: true
```

📋 OPERATIONAL PROCEDURES

Deployment Automation Pipeline



Incident Response Procedures

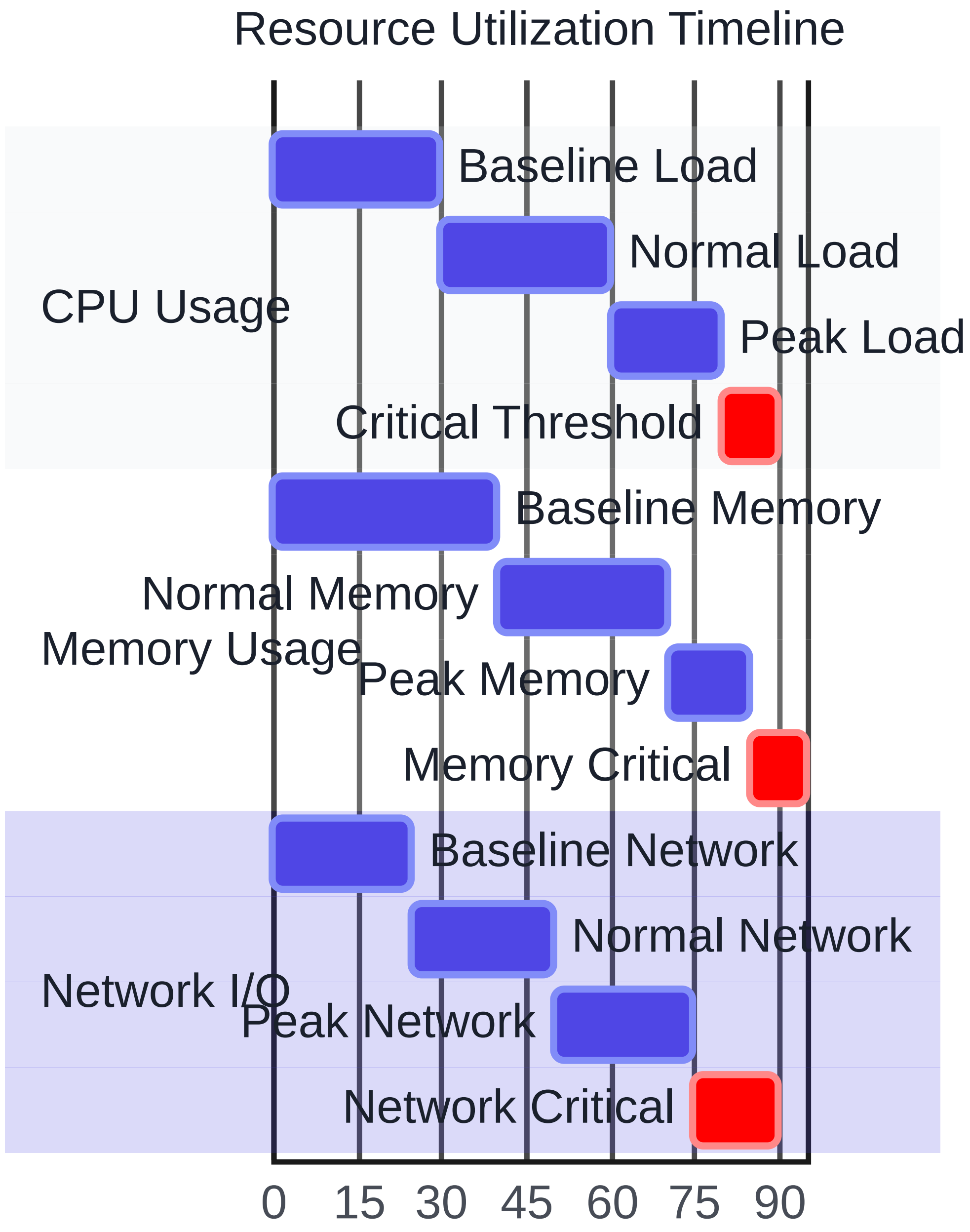


🎯 PERFORMANCE BENCHMARKS & TARGETS

Service Level Objectives (SLOs)

METRIC CATEGORY	OBJECTIVE	TARGET	MEASUREMENT
Availability	Service Uptime	99.99%	Monthly rolling window
Performance	Authentication Response	< 100ms (p95)	Request duration
Performance	API Response Time	< 200ms (p99)	End-to-end latency
Scalability	Concurrent Users	10,000+	Peak load capacity
Reliability	Error Rate	< 0.1%	Request success ratio
Security	Auth Success Rate	> 99.5%	Authentication attempts
Recovery	RTO (Recovery Time)	< 15 minutes	Incident response
Recovery	RPO (Recovery Point)	< 5 minutes	Data loss window

Resource Utilization Targets



🔒 SECURITY IMPLEMENTATION CHECKLIST

Security Controls Implementation Status

SECURITY CONTROL	IMPLEMENTATION STATUS	VALIDATION METHOD
Multi-Factor Authentication	✅ Implemented	Automated testing + Manual verification
JWT Token Security	✅ Implemented	Security scanning + Penetration testing
Role-Based Access Control	✅ Implemented	Permission matrix testing
Tenant Data Isolation	✅ Implemented	Cross-tenant access testing
Encryption at Rest	✅ Implemented	Database encryption verification
Encryption in Transit	✅ Implemented	TLS configuration validation
Input Validation	✅ Implemented	Injection attack testing
Audit Logging	✅ Implemented	Log completeness verification
Session Management	✅ Implemented	Session security testing
Rate Limiting	✅ Implemented	Load testing + DoS simulation
CORS Protection	✅ Implemented	Cross-origin request testing
CSRF Protection	✅ Implemented	CSRF attack simulation
SQL/NoSQL Injection Prevention	✅ Implemented	Injection attack testing
XSS Protection	✅ Implemented	Script injection testing
Security Headers	✅ Implemented	Header configuration validation

This comprehensive Low Level Design document provides detailed technical specifications, implementation details, code structure, database schemas, API designs, security implementations, deployment configurations, and operational procedures for the Securaa User Service. The document includes extensive diagrams, code examples, and configuration templates to guide development and operations teams in implementing and maintaining this critical security service.