CuratorAI MVP - System Architecture Documentation

Version 1.0 | October 2025
Project: CuratorAI Fashion Recommendation Platform
Client: K&O Curator Technologies Group Ltd.
Developer: Sumic IT Solutions Ltd.

Table of Contents

Executive Summary

CuratorAI is an AI-powered fashion recommendation platform that combines computer vision, machine learning, and virtual try-on technologies to deliver personalized outfit recommendations. The system is designed for scalability, handling 10,000+ concurrent users with sub-200ms response times.

Key Features

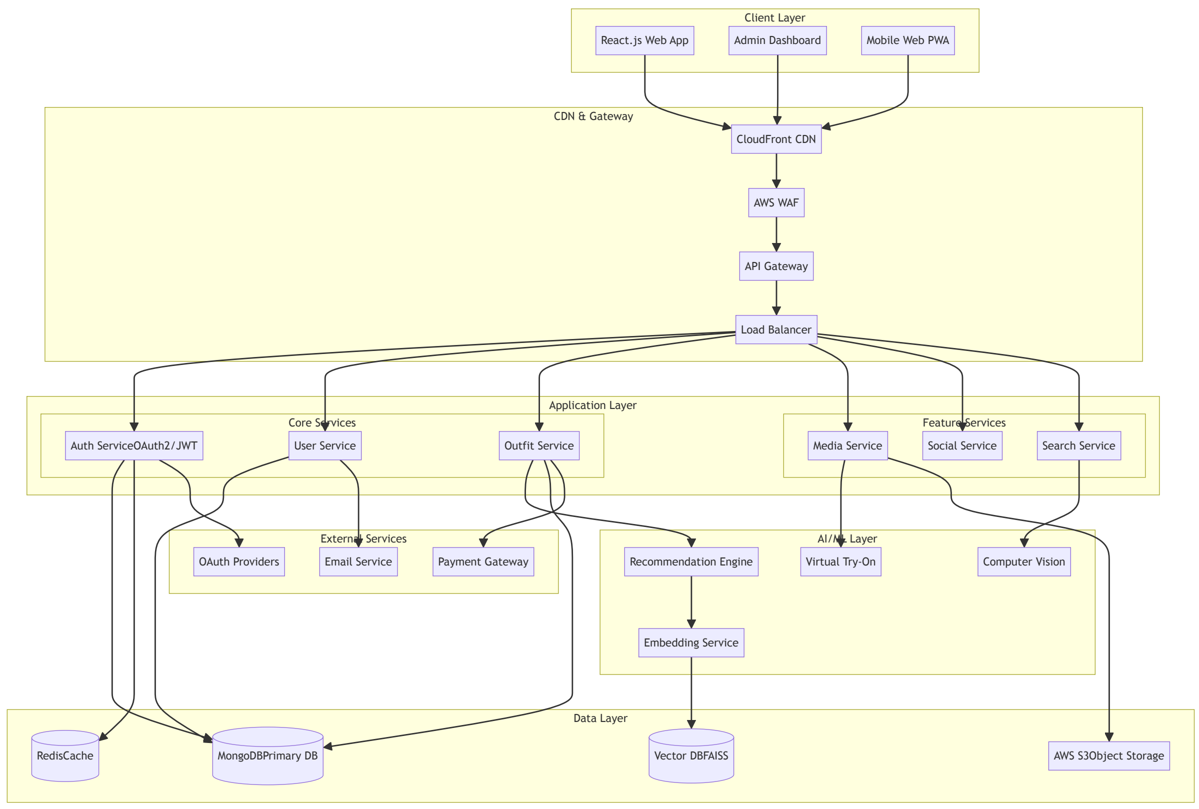
- AI-powered outfit recommendations
- Virtual try-on using AR/Computer Vision
- Visual search for similar outfits
- Wardrobe tracking and management
- Social feed with shoppable lookbooks
- Comprehensive admin dashboard

Technical Highlights

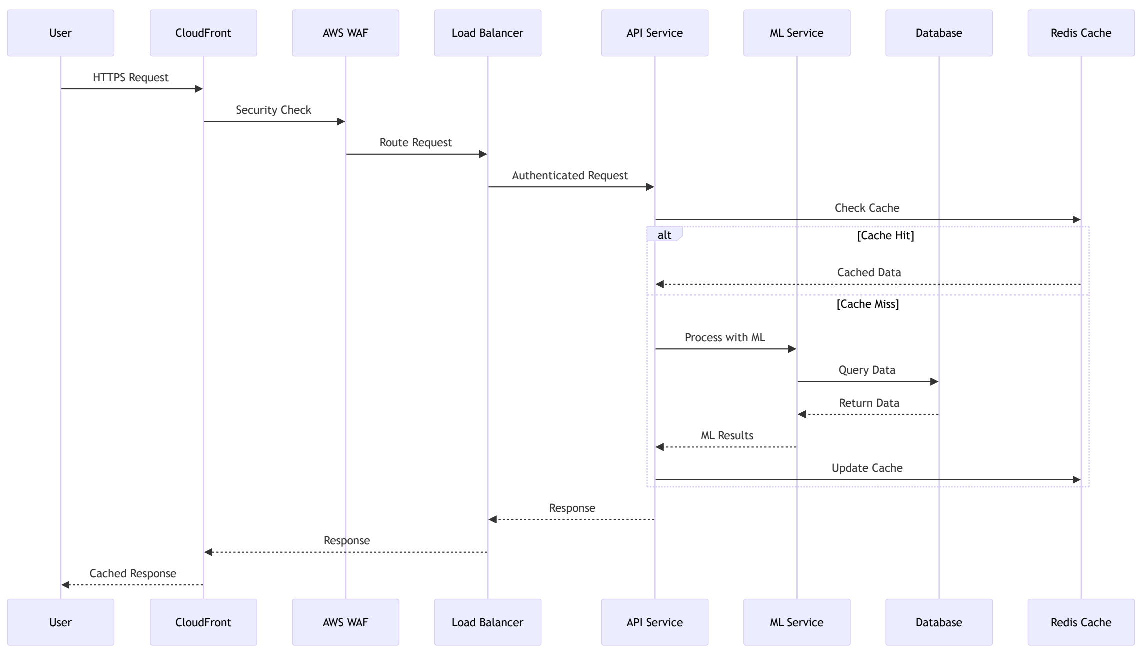
- **Stack**: React.js, Node.js, MongoDB, TensorFlow
- **Cloud**: AWS Multi-region architecture
- **Performance**: <200ms API response, <500ms ML inference
- **Scale**: 10,000 concurrent users, 5,000 RPS
- **Cost**: $4,700-7,000/month estimated

System Overview

High-Level Architecture

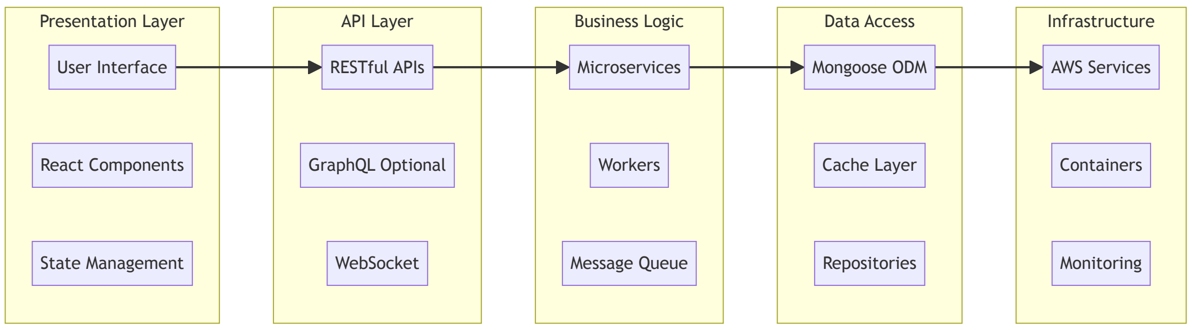


Request Flow



Architecture Layers

System Layers Overview



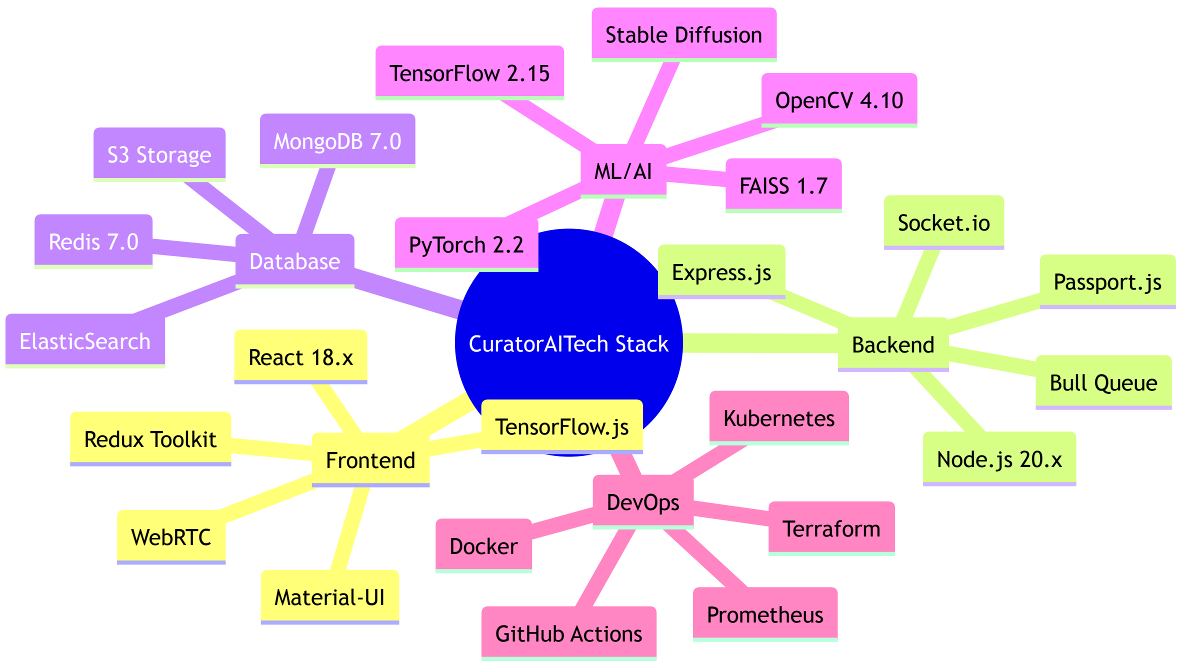
Layer Responsibilities

Layer	Responsibility	Technologies
Presentation	User interface, interactions	React, Redux, Material-UI
API Gateway	Routing, rate limiting, auth	Express, Nginx
Application	Business logic, processing	Node.js, Express
AI/ML	Intelligence, recommendations	TensorFlow, Python
Data	Persistence, caching	MongoDB, Redis

Layer	Responsibility	Technologies
Infrastructure	Cloud resources, scaling	AWS, Docker, K8s

Technology Stack

Core Technologies



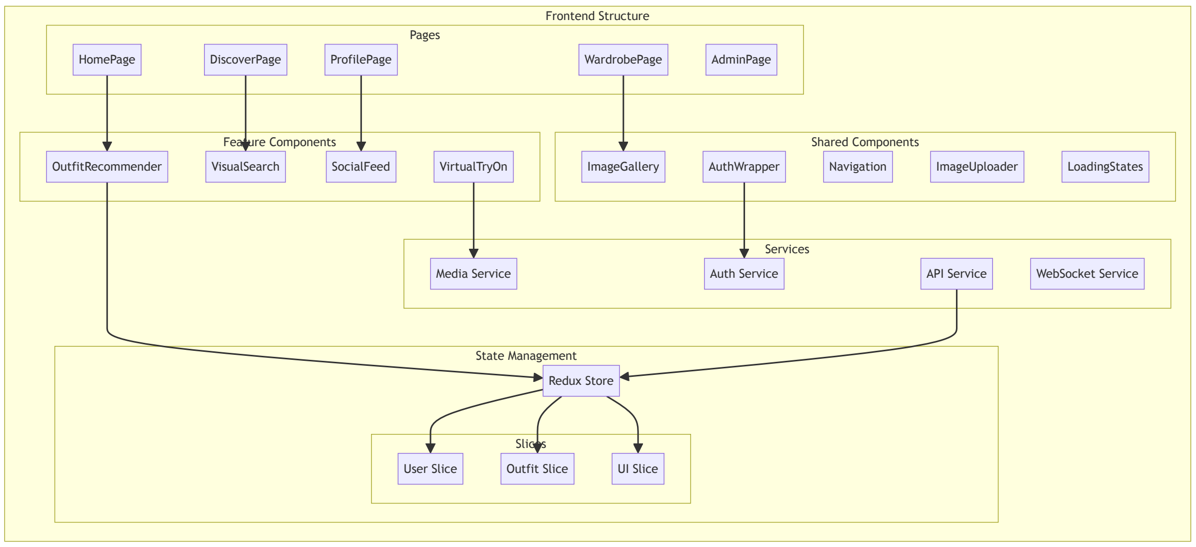
Technology Decision Matrix

Component	Technology	Alternative	Rationale
Frontend Framework	React.js	Vue.js, Angular	Team expertise, ecosystem
Backend Runtime	Node.js	Python, Go	JavaScript consistency
Primary Database	MongoDB	PostgreSQL	Flexibility for schema evolution
Cache Layer	Redis	Memcached	Persistence, data structures
ML Framework	TensorFlow	PyTorch	Production maturity

Component	Technology	Alternative	Rationale
Container Platform	Docker	Podman	Industry standard
Orchestration	ECS	Kubernetes	AWS integration
CI/CD	GitHub Actions	Jenkins	GitHub integration

Frontend Architecture

Component Architecture



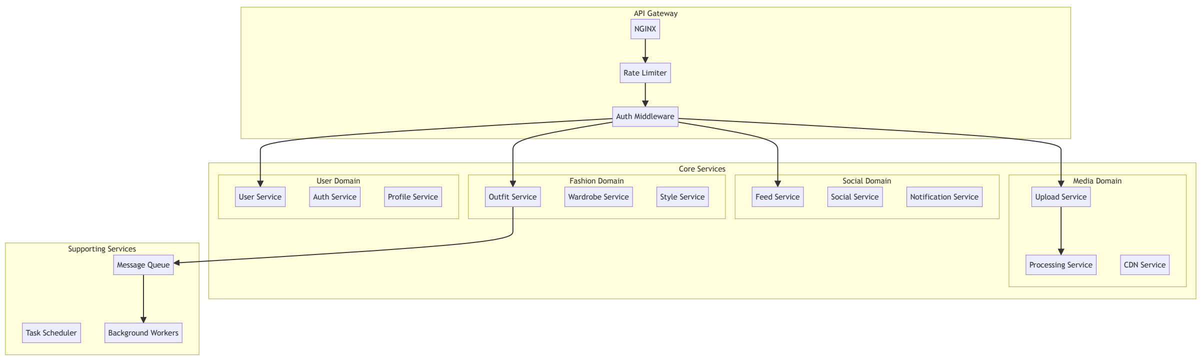
Frontend File Structure

```
frontend/
├── public/
│   ├── index.html
│   └── manifest.json
├── src/
│   ├── components/
│   │   ├── common/
│   │   │   ├── Button/
│   │   │   ├── Modal/
│   │   │   └── Form/
│   │   ├── outfit/
│   │   │   ├── OutfitCard/
│   │   │   ├── OutfitGrid/
```

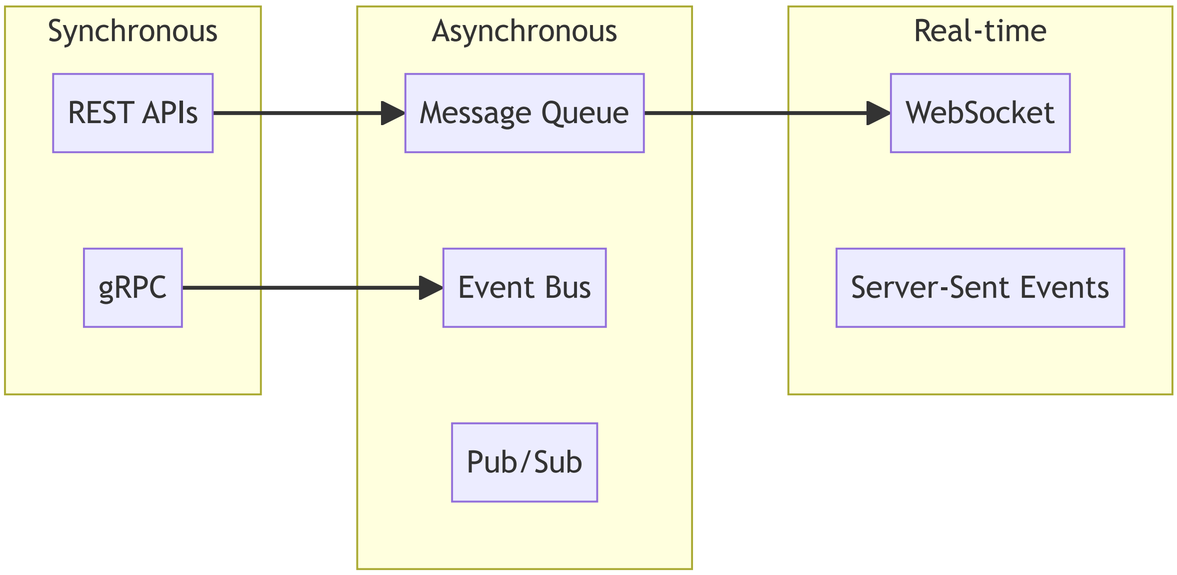
```
|   |   |   └── Recommender/
|   |   ├── wardrobe/
|   |   |   ├── WardrobeItem/
|   |   |   └── CategoryFilter/
|   |   └── tryon/
|   |       ├── Camera/
|   |       └── AROverlay/
|   ├── pages/
|   |   ├── Home/
|   |   ├── Profile/
|   |   └── Admin/
|   ├── services/
|   |   ├── api.js
|   |   ├── auth.js
|   |   └── websocket.js
|   ├── store/
|   |   ├── index.js
|   |   └── slices/
|   ├── hooks/
|   |   ├── useAuth.js
|   |   └── useOutfit.js
|   └── utils/
|       ├── constants.js
|       └── helpers.js
```

Backend Architecture

Microservices Architecture



Service Communication

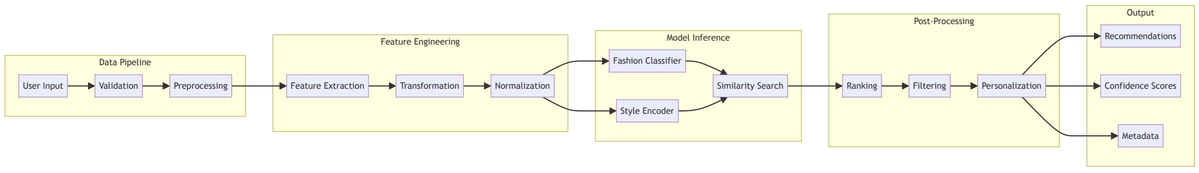


API Endpoints Structure

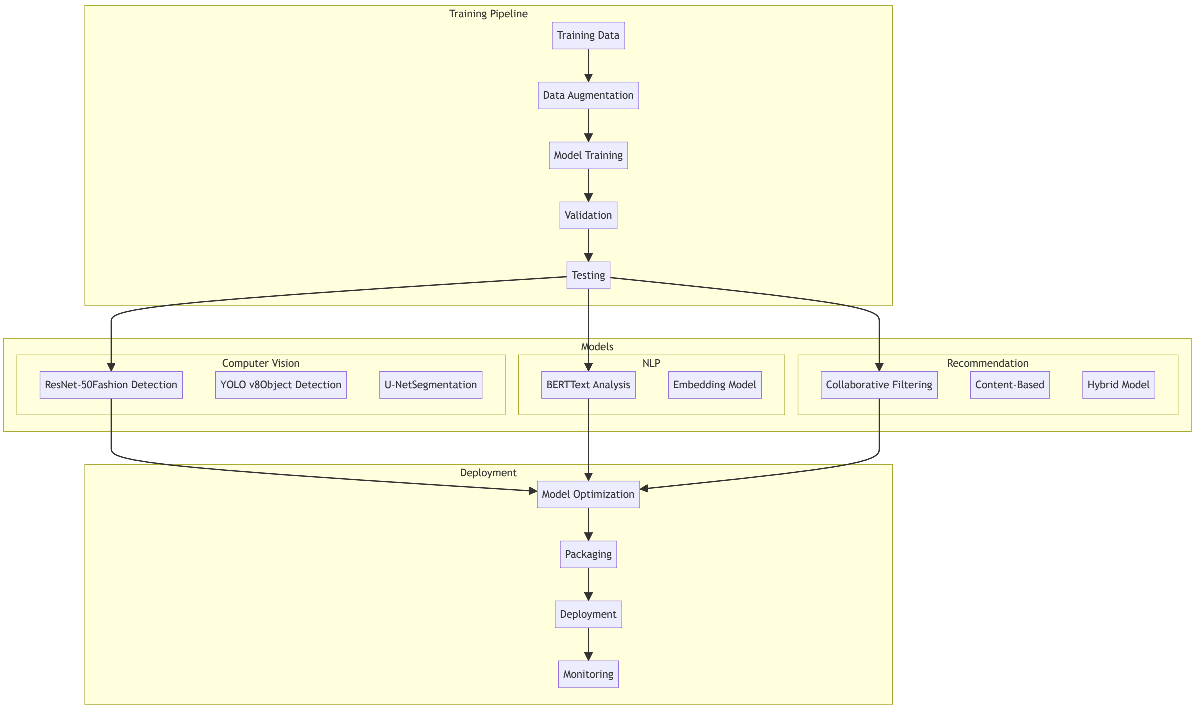
```
/api/v1:
  /auth:
    POST /register: User registration
    POST /login: User login
    POST /logout: User logout
    POST /refresh: Token refresh
    GET /profile: Get user profile

  /users:
    GET /{userId}: Get user details
    PUT /{userId}: Update user
    DELETE /{userId}: Delete user
    GET /{userId}/wardrobe: Get user wardrobe
    POST /{userId}/preferences: Update preferences

  /outfits:
    GET /recommendations: Get recommendations
    POST /generate: Generate outfit
    GET /{outfitId}: Get outfit details
    POST /{outfitId}/like: Like outfit
    POST /{outfitId}/save: Save outfit

  /wardrobe:
    GET /items: List wardrobe items
    POST /items: Add item
    PUT /items/{itemId}: Update item
    DELETE /items/{itemId}: Delete item
    POST /items/{itemId}/upload: Upload image

  /search:
    POST /visual: Visual search
    GET /similar/{imageId}: Find similar
    POST /filters: Search with filters

  /social:
    GET /feed: Get social feed
    POST /posts: Create post
    GET /posts/{postId}: Get post
    POST /posts/{postId}/comment: Add comment

  /admin:
    GET /dashboard: Dashboard data
    GET /analytics: Analytics
    GET /users: User management
    POST /content: Content management
```

AI/ML Architecture

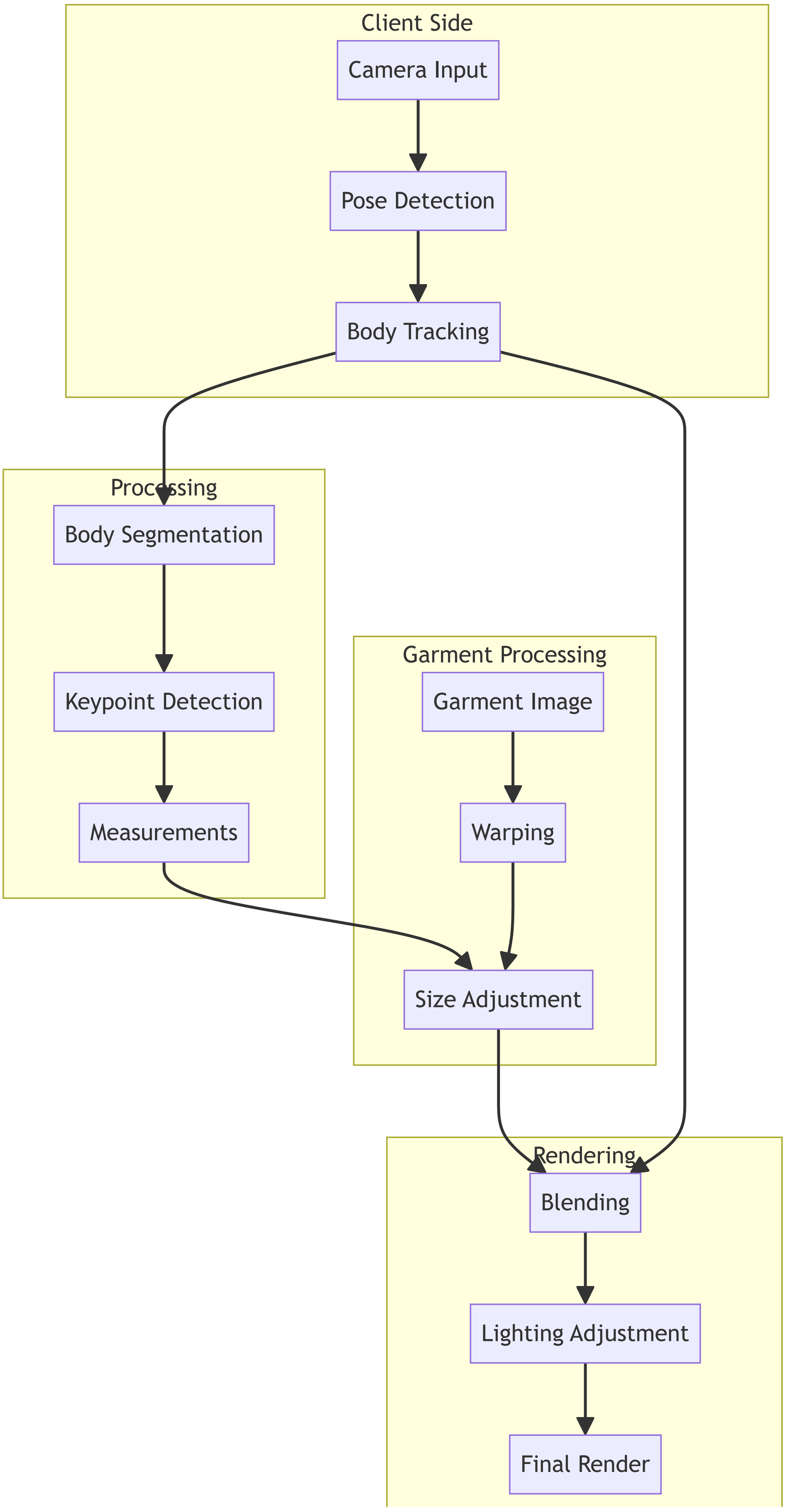
ML Pipeline



Model Components

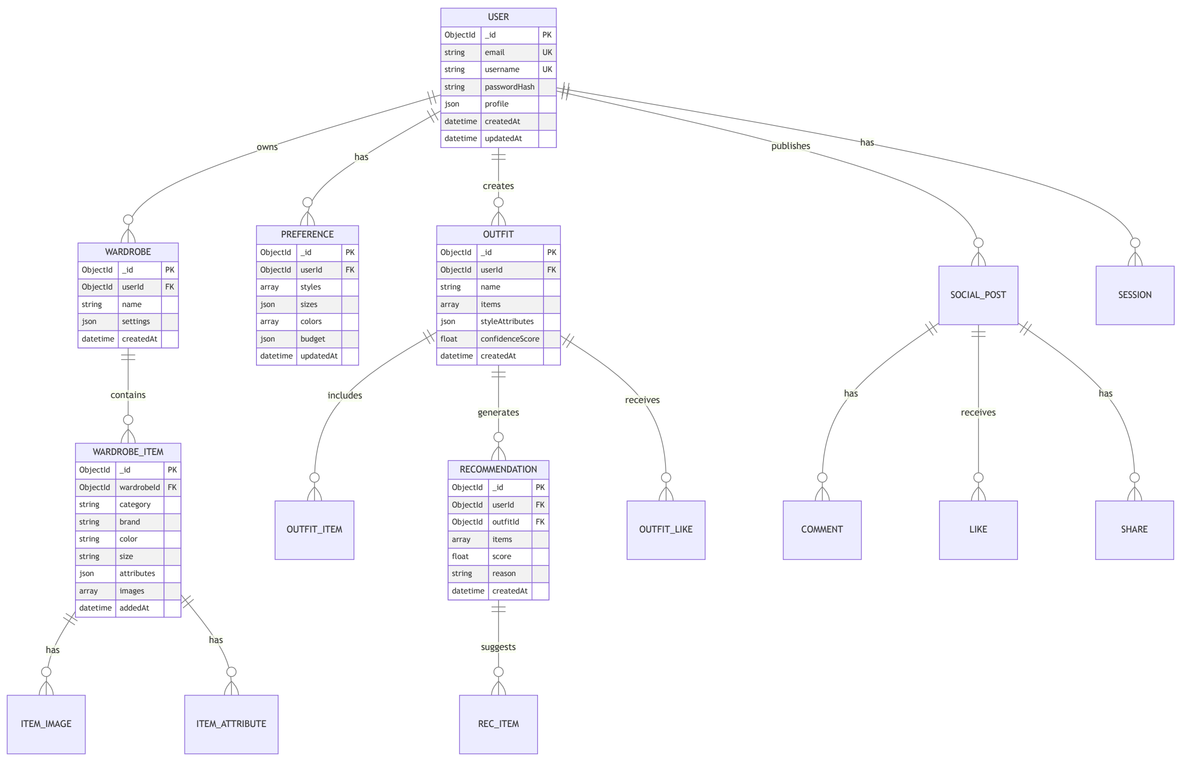


Virtual Try-On Architecture

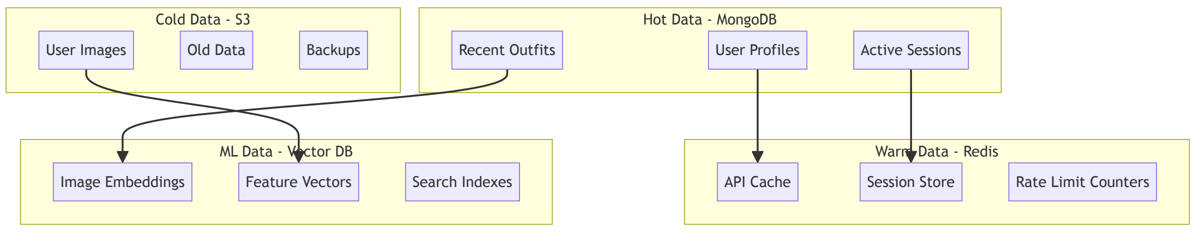

Client Side
Camera Input
Pose Detection
Body Tracking
Processing
Body Segmentation
Keypoint Detection
Measurements
Garment Processing
Garment Image
Warping
Size Adjustment
Rendering
Blending
Lighting Adjustment
Final Render

Database Architecture

Database Schema

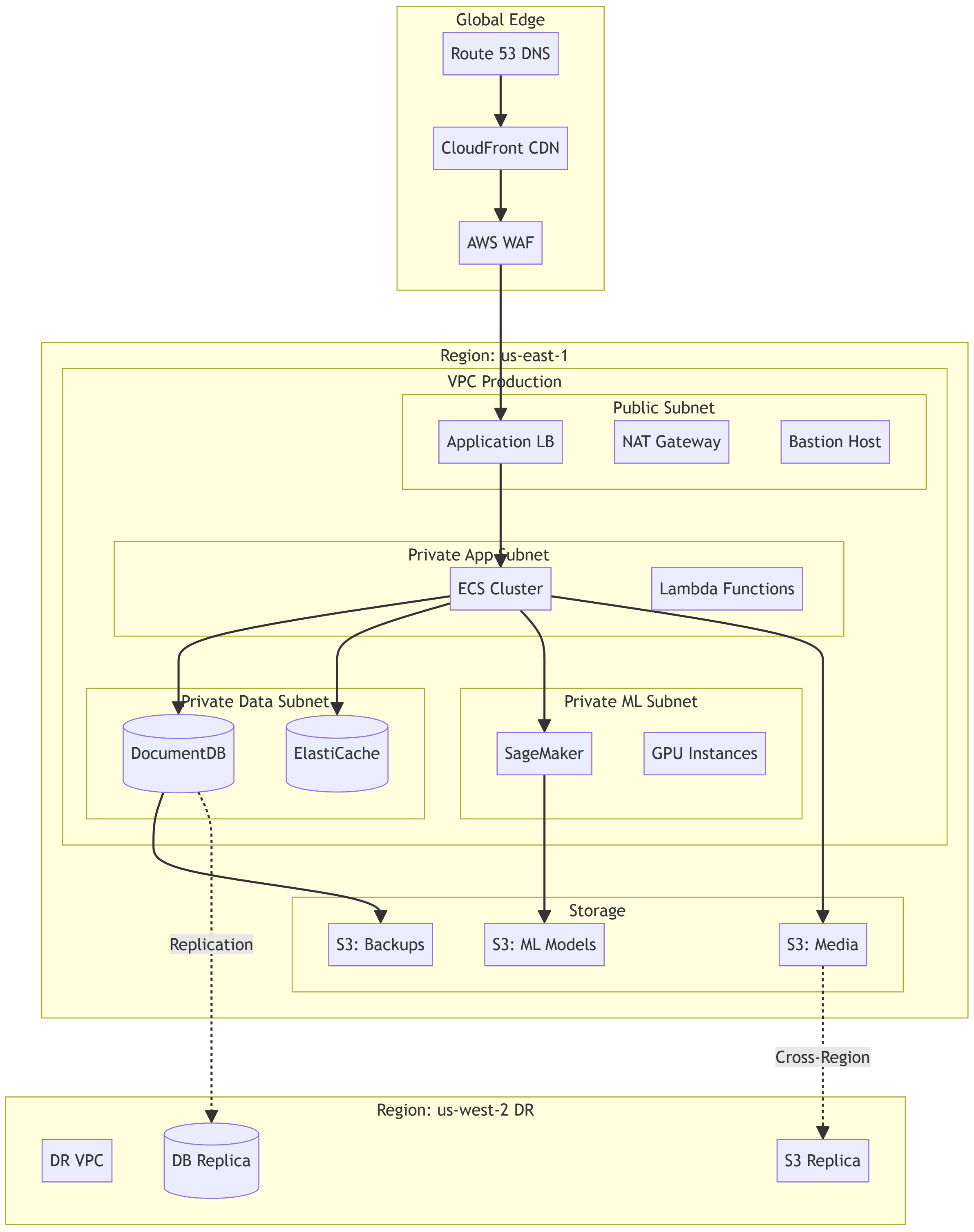


Data Storage Strategy

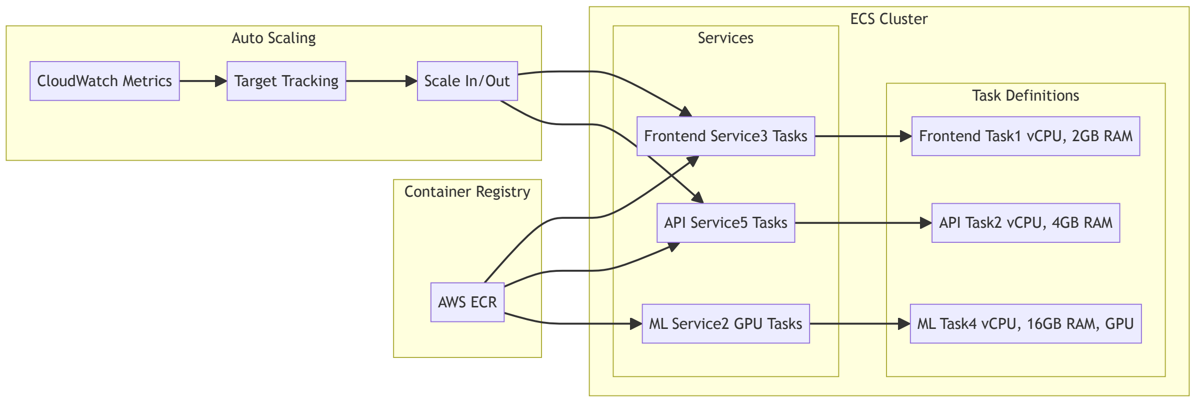


Cloud Infrastructure

AWS Architecture

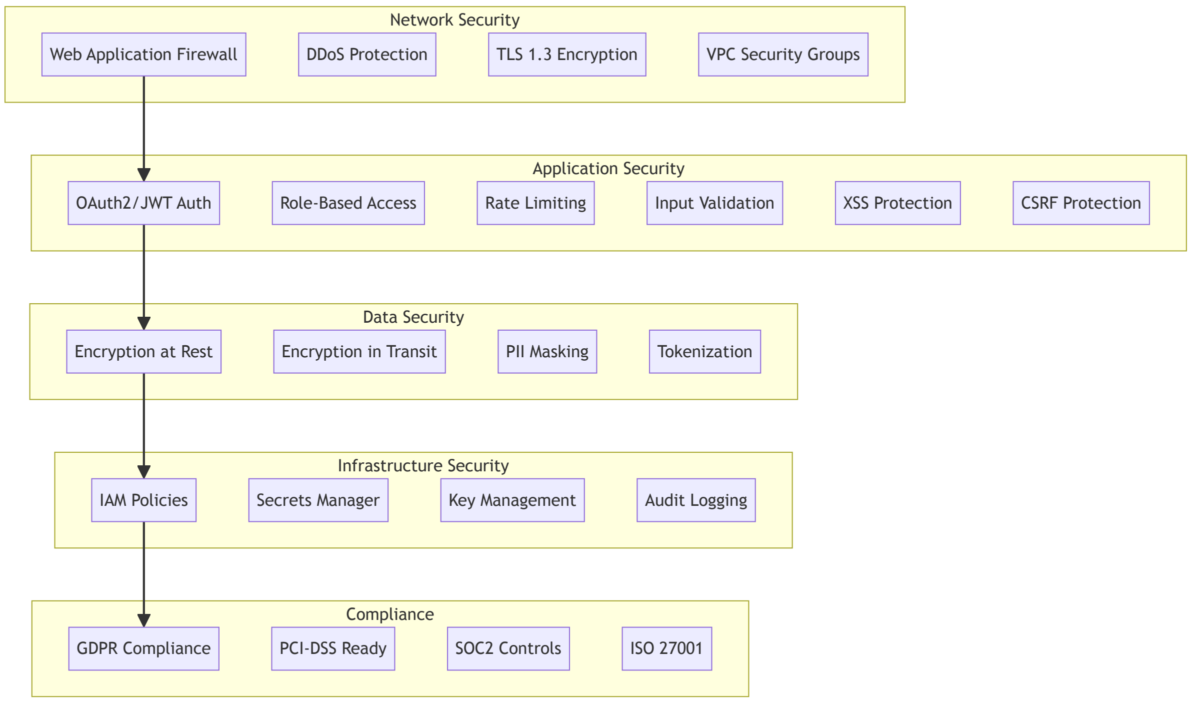


Container Orchestration

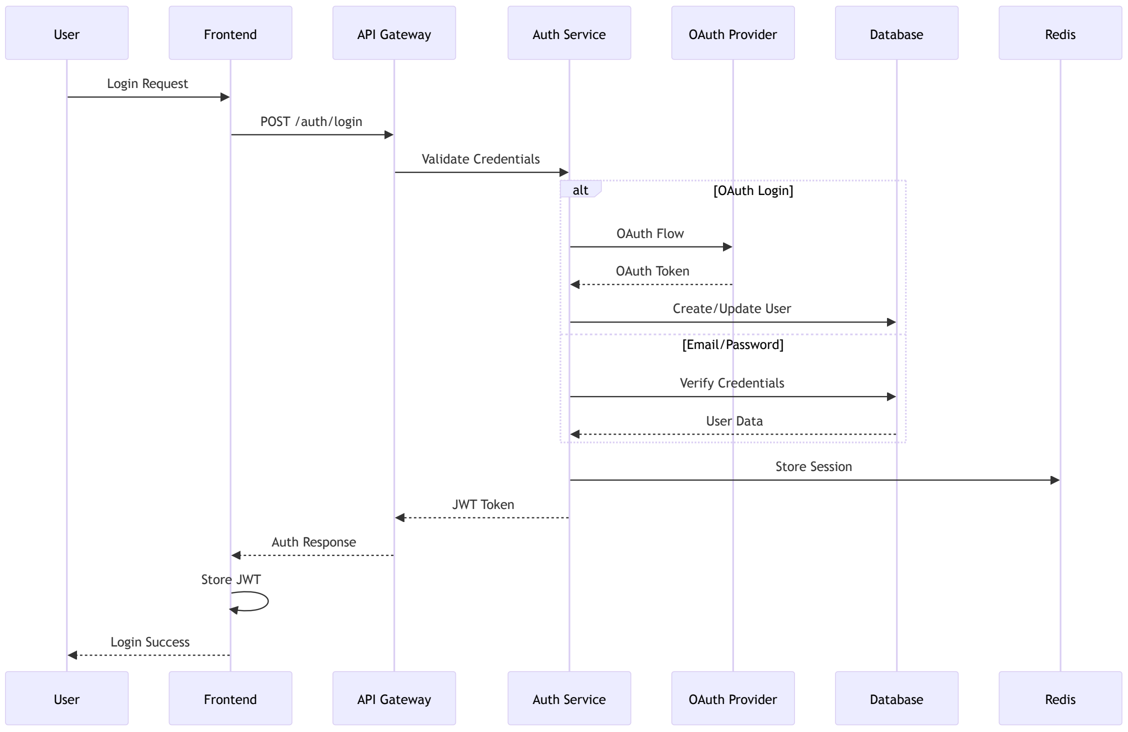


Security Architecture

Security Layers



Authentication Flow

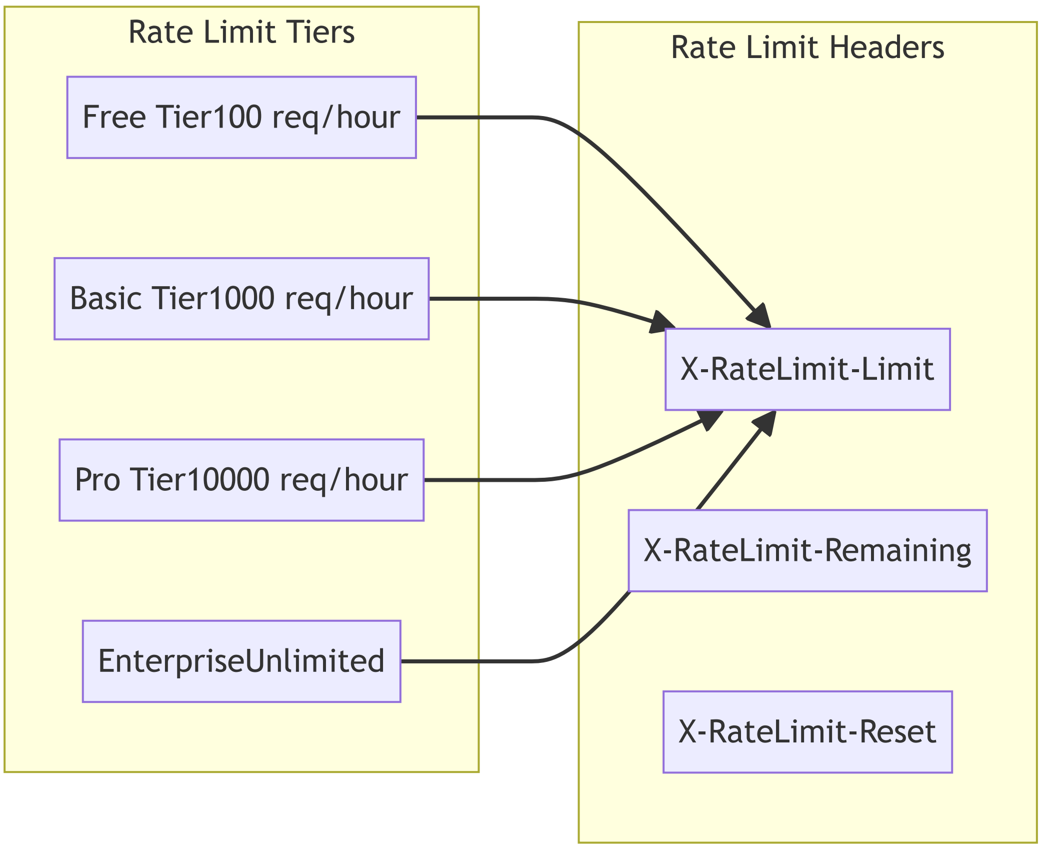


API Design

RESTful API Standards

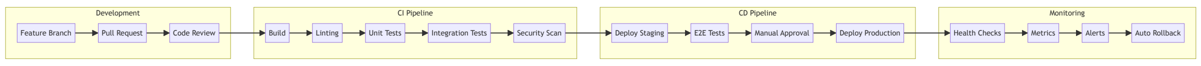
```
API Standards:
  Version: v1
  Base URL: https://api.curatorai.com/v1

  Authentication:
    Type: Bearer Token (JWT)
    Header: Authorization: Bearer {token}

  Response Format:
    Content-Type: application/json
    Structure:
      success:
        status: success
        data: object/array
        metadata:
          timestamp: ISO8601
          version: string
          pagination: object (if applicable)
      error:
        status: error
        error:
          code: string
          message: string
          details: object
        metadata:
          timestamp: ISO8601
          request_id: uuid

  Status Codes:
    200: OK
    201: Created
    204: No Content
    400: Bad Request
    401: Unauthorized
    403: Forbidden
    404: Not Found
    429: Too Many Requests
    500: Internal Server Error
    503: Service Unavailable
```

API Rate Limiting

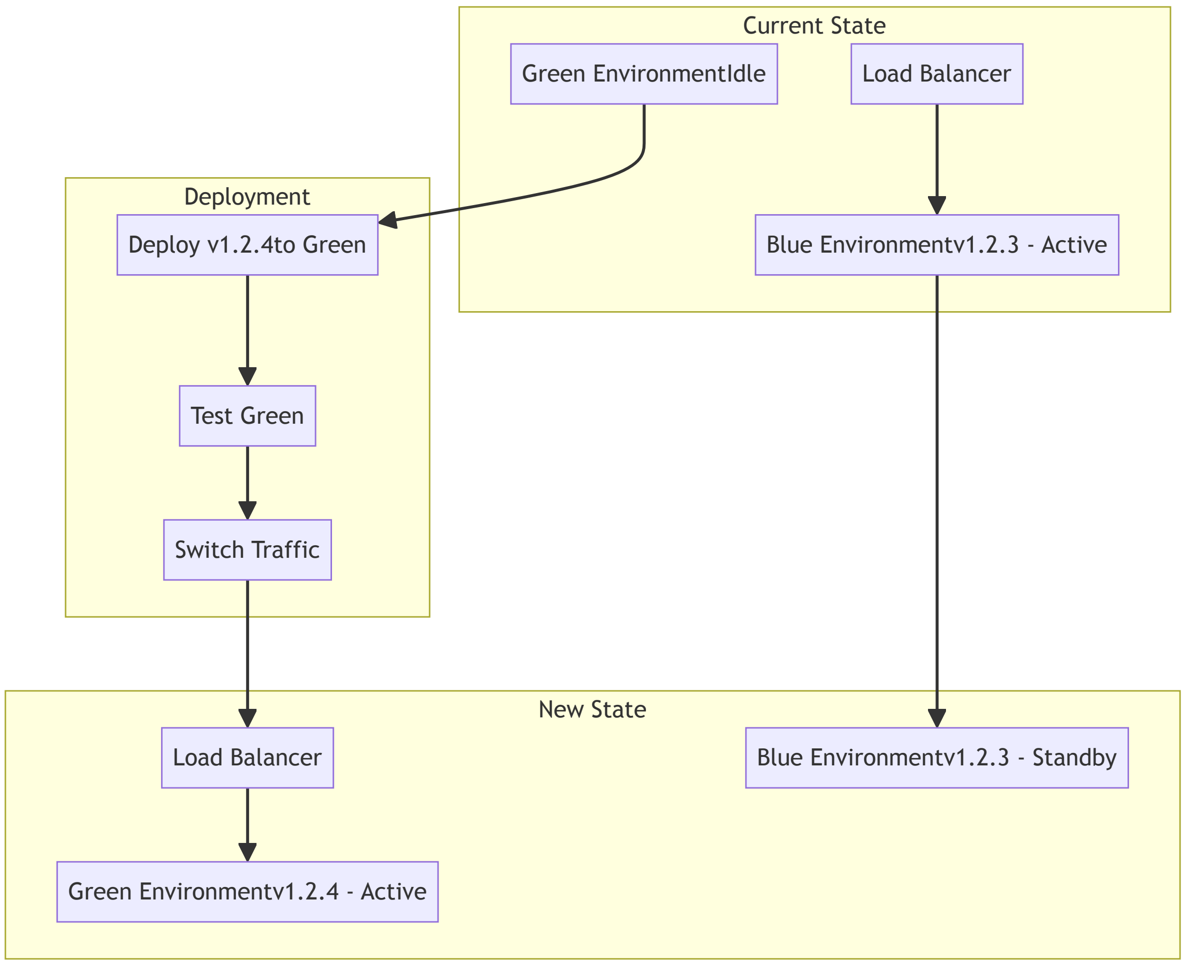


Deployment Strategy

CI/CD Pipeline

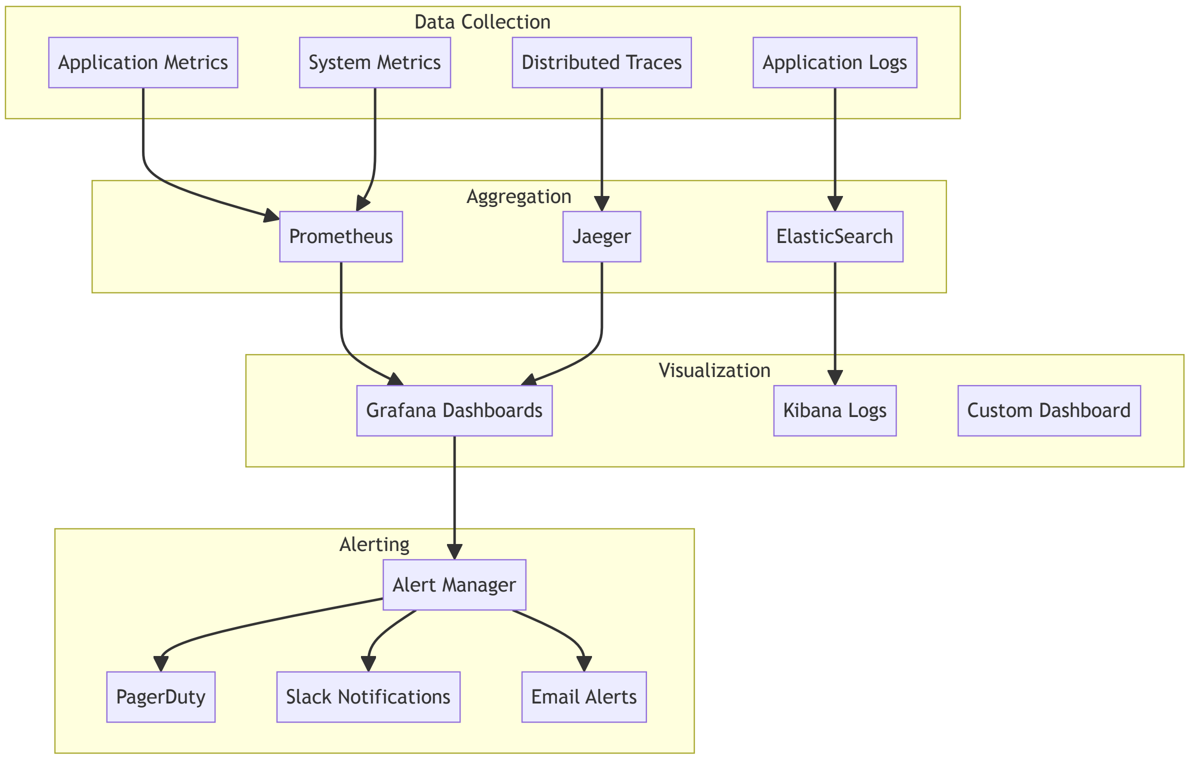


Blue-Green Deployment



Monitoring & Observability

Monitoring Stack



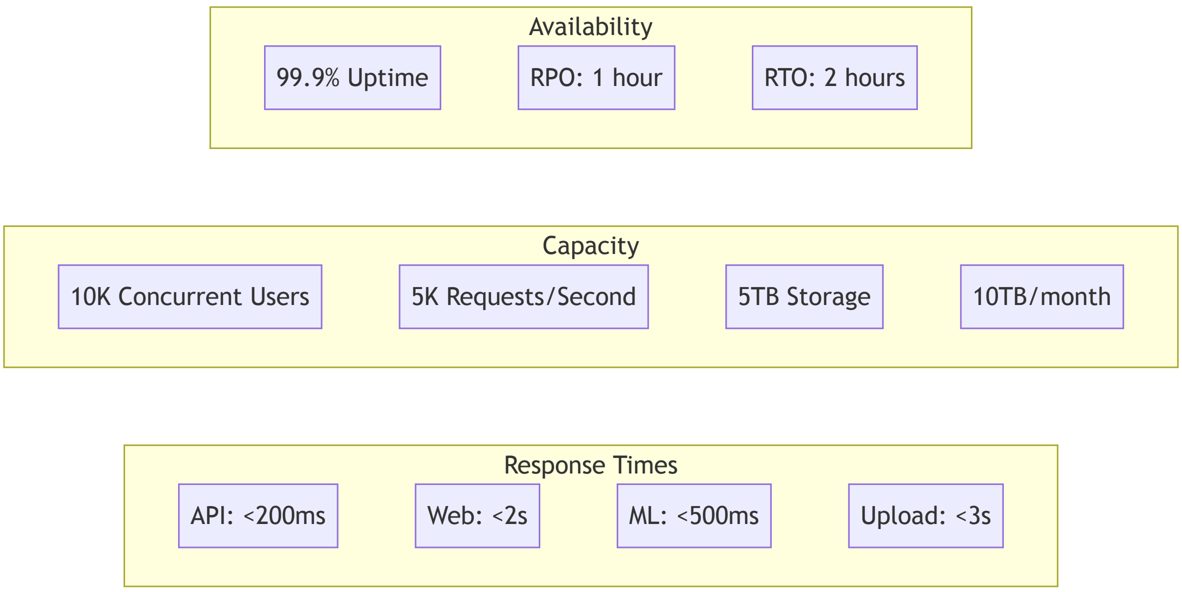
Key Performance Indicators

Category	Metric	Target	Alert Threshold
Availability	Uptime	99.9%	< 99.5%
Performance	API Response Time (p50)	< 100ms	> 150ms
	API Response Time (p95)	< 200ms	> 500ms
	API Response Time (p99)	< 500ms	> 1000ms
Throughput	Requests per Second	5000	< 1000
Error Rate	4xx Errors	< 2%	> 5%
	5xx Errors	< 0.1%	> 1%
ML Performance	Model Accuracy	> 90%	< 85%

Category	Metric	Target	Alert Threshold
	Inference Time	< 500ms	> 1000ms
Business Metrics	Daily Active Users	10,000	< 5,000
	Outfit Generation Rate	500/hour	< 100/hour

Performance Requirements

System Performance Targets

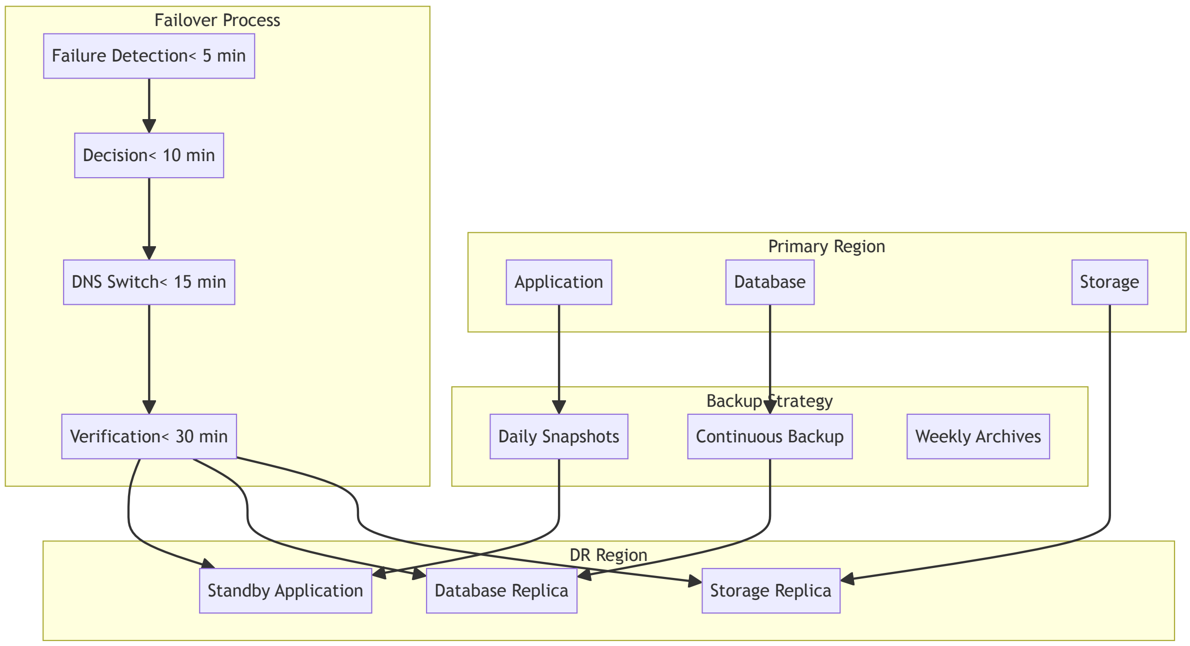


Performance Optimization Strategies

Area	Strategy	Expected Improvement
Frontend	Code splitting, lazy loading	40% faster load
API	Response caching, pagination	60% latency reduction
Database	Indexing, query optimization	50% faster queries
ML Models	Model quantization, caching	70% inference speedup
Infrastructure	CDN, auto-scaling	80% better response

Disaster Recovery

DR Strategy



Recovery Procedures

Scenario	RPO	RTO	Procedure
Database Failure	1 hour	30 min	Promote read replica
Region Outage	1 hour	2 hours	Failover to DR region
Data Corruption	24 hours	4 hours	Restore from backup
Service Failure	0	5 min	Auto-scaling recovery
Complete Disaster	24 hours	4 hours	Full DR activation

Document Control

- **Version:** 1.0
- **Created:** October 1, 2025
- **Last Updated:** October 1, 2025
- **Author:** Team Lead, Sumic IT Solutions
- **Status:** Living Document

- **Review Cycle:** Bi-weekly during development
