GlamSage

Technical Design Proposal

A Cross-Platform Cosmetic Price Comparison System

February 24, 2025

Contents

1 Executive Summary

GlamSage is a comprehensive cosmetic product comparison platform designed to revolutionize how consumers shop for beauty products. By aggregating real-time pricing information from leading e-commerce platforms, GlamSage enables consumers to make informed purchasing decisions, ensuring they get the best value for their money.

This technical design document outlines the architecture, implementation approach, and technology stack for GlamSage. The system employs a modern microservices architecture with React Native for cross-platform mobile applications, ensuring maximum market reach while optimizing development resources.

The platform will integrate with major cosmetic retailers including Amazon, Flipkart, Nykaa, and Myntra through a combination of API integrations and web scraping technologies. Advanced data normalization and product matching algorithms will ensure accurate comparison across different platforms.

2 Market Analysis & Business Justification

2.1 Market Opportunity

The global cosmetics market is projected to reach $463.5 billion by 2027, growing at a CAGR of 5.3%. The intersection of this market with e-commerce presents a significant opportunity:

- 72% of beauty product purchases involve online research before buying
- Price is cited as the primary decision factor by 67% of cosmetic consumers
- 82% of beauty shoppers use multiple platforms to compare prices
- Average beauty consumer can save 15-30% through price comparison

2.2 Competitive Landscape

While general price comparison tools exist, the cosmetics sector lacks a specialized solution that addresses the unique attributes of beauty products:

Existing Solutions	**Limitations**	**GlamSage Advantage**
General price comparison sites	Limited cosmetic product coverage	Specialized cosmetic focus
E-commerce platform comparisons	Limited to a single platform	Cross-platform comparison
Beauty blogs and reviews	Manual, non-real-time comparisons	Automated, real-time data

Table 1: Competitive Landscape Analysis

2.3 Revenue Model

GlamSage will implement a multi-faceted revenue strategy:

- Affiliate marketing commissions (7-15% per transaction)
- Premium subscription tier ($4.99/month) with advanced features
- Sponsored product placement
- Data insights for beauty brands and retailers

3 System Architecture

3.1 Architecture Overview

GlamSage employs a modern microservices architecture to ensure scalability, maintainability, and high performance. The system is designed around six core layers:

[System Architecture Diagram]
Client Layer → API Gateway → Application Services → Data Collection → Storage Layer

Figure 1: High-Level System Architecture

3.2 Core Components

3.2.1 Client Layer

The client layer provides cross-platform access to GlamSage functionality through:

- React Native mobile applications (iOS/Android)
- React.js web application
- Electron-based desktop application

React Native Selection Justification:

- 70-80% code sharing across platforms
- 40% reduction in development time compared to native development
- 35% reduction in maintenance costs
- Faster time-to-market for MVP and updates

3.2.2 API Gateway Layer

The API Gateway serves as the unified entry point for all client-service communication, providing:

- Authentication and authorization
- Request routing and load balancing
- Rate limiting and request validation
- Response caching and compression

3.2.3 Application Services

The application layer is composed of specialized microservices:

- Product Service: Manages product catalog and metadata
- Price Comparison Service: Core price analysis and comparison
- User Service: Handles authentication and user preferences
- Search Service: Provides advanced search capabilities
- Notification Service: Manages alerts and communications
- Analytics Service: Provides insights on usage and trends

3.2.4 Data Collection Layer

This layer is responsible for gathering price and product information:

- Web Scraper Service: For platforms without APIs
- E-commerce API Integration: Direct platform connections
- Data Normalization Service: Standardizes product data

3.2.5 Storage Layer

The storage layer utilizes specialized databases for different data types:

- MongoDB: Flexible product data storage
- TimescaleDB: Time-series price history
- PostgreSQL: Relational user data
- Redis: High-performance caching
- Elasticsearch: Fast, full-text search

4 Implementation Details

4.1 Data Schema Design

4.1.1 Product Schema (MongoDB)

```
{
  _id: ObjectId,
  name: String,
  brand: String,
  category: String,
  subCategory: String,
  description: String,
  ingredients: [String],
  images: [String],
  attributes: {
    size: String,
    weight: Number,
    color: String,
    variant: String
  },
  metadata: {
    created_at: Timestamp,
    updated_at: Timestamp,
    is_active: Boolean
  }
}
```

4.1.2 Price Schema (TimescaleDB)

```sql
CREATE TABLE product_prices (
  price_id SERIAL PRIMARY KEY,
  product_id UUID NOT NULL,
  platform_id INT NOT NULL,
  price DECIMAL(10,2) NOT NULL,
  discount_price DECIMAL(10,2),
  currency VARCHAR(3),
  stock_status BOOLEAN,
  url TEXT,
  timestamp TIMESTAMPTZ NOT NULL
);
```

4.2 API Design

The system exposes a comprehensive RESTful API to support all client functionality:

4.2.1 Product API

- `GET /api/v1/products` - List products with filtering
- `GET /api/v1/products/{id}` - Get product details
- `POST /api/v1/products/compare` - Compare multiple products
- `GET /api/v1/products/{id}/price-history` - View price history

4.2.2 User API

- `POST /api/v1/auth/register` - User registration
- `POST /api/v1/auth/login` - User authentication
- `GET /api/v1/users/wishlist` - Manage saved products
- `POST /api/v1/alerts` - Configure price alerts

5 Security Architecture

5.1 Authentication & Authorization

GlamSage implements a robust security model:

- OAuth 2.0 with JWT for secure authentication
- Role-based access control
- Short-lived access tokens (1 hour) with refresh tokens
- Secure token storage on client devices

5.2 Data Protection

- End-to-end encryption for all communications
- Data encryption at rest using AES-256
- PII data minimization and anonymization
- Regular security audits and penetration testing

6 Scaling & Performance

6.1 Scalability Approach

The system is designed for horizontal scalability:

- Containerized microservices managed by Kubernetes
- Auto-scaling based on load metrics
- Database sharding for high-volume data
- CDN integration for static assets

6.2 Performance Optimizations

- Multi-level caching (client, API, application, database)
- Response compression and minification
- Lazy loading of non-critical resources
- Optimized database queries and indexes

7 Implementation Timeline

Phase	**Deliverables**	**Timeline**
Research & Planning	Market research, technical specifications, architecture design	4 weeks
MVP Development	Core features, basic integrations (2 platforms), web & iOS app	12 weeks
Beta Release	All platform integrations, Android app, initial user testing	8 weeks
Production Release	Full feature set, performance optimization, security audits	6 weeks
Post-Launch	Analytics, optimization, additional features	Ongoing

Table 2: Implementation Phases

8 Resource Requirements

8.1 Development Team

- 2 React Native developers

- 2 Backend developers (Node.js, Python)
- 1 DevOps engineer
- 1 UI/UX designer
- 1 QA specialist
- 1 Product manager

8.2 Infrastructure

- Cloud hosting (AWS or Google Cloud)
- CI/CD pipeline
- Monitoring and logging infrastructure
- Development, staging, and production environments

9 Financial Projections

Category	**Year 1**	**Year 2**	**Year 3**
Development Costs	$420,000	$380,000	$350,000
Infrastructure Costs	$60,000	$96,000	$144,000
Marketing & Acquisition	$150,000	$300,000	$450,000
Revenue (Affiliate)	$180,000	$720,000	$1,800,000
Revenue (Subscriptions)	$36,000	$240,000	$600,000
Net Profit/Loss	-$414,000	$184,000	$1,456,000

Table 3: 3-Year Financial Projection

10 Risk Assessment & Mitigation

Risk	Impact	Mitigation Strategy
E-commerce platform API limitations	High	Implement robust scraping fallbacks; establish strategic partnerships
Data accuracy challenges	High	Advanced normalization algorithms; user feedback loops; frequent data verification
Market adoption barriers	Medium	Targeted marketing; influencer partnerships; seamless UX focus
Competitor response	Medium	Accelerated feature roadmap; unique value propositions; strategic partnerships
Regulatory compliance	Medium	Regular legal reviews; privacy-by-design; data minimization

Table 4: Risk Assessment Matrix

11 Conclusion

GlamSage represents a significant opportunity to address a clear market gap in the cosmetics e-commerce sector. The technical architecture outlined in this proposal provides a robust, scalable foundation for delivering a superior price comparison experience.

The combination of React Native for cross-platform development and a microservices backend architecture ensures rapid time-to-market while maintaining the flexibility to scale and evolve. Data collection mechanisms balance API integration with web scraping to ensure comprehensive coverage of the cosmetics market.

With appropriate investment and execution, GlamSage is positioned to become the leading cosmetic price comparison platform within 18-24 months of launch. The revenue model, technical architecture, and implementation approach outlined in this document provide a clear roadmap to success.