Project - Marketing Content Generation

High Level Design

Course Name: Generative AI

Institution Name: Medicpas University - Datagami Skill Based Course

Student Name(s) & Enrolment Number(s)

S. No.	Student Name	Enrolment No.
1	Bhavya Rathore	EN22CS301269
2	Bhoomika Sanodiya	EN22CS301271
3	Bhumika Chouhan	EN22CS301275
4	Devi Sanjaykumar Modi	EN23CS3T1008
5	Dollyna Sahu	EN22CS301354

Group Name: **Group G2D3**

Project Number: **GAI-26**

Industry Mentor Name: **Mr. Suraj Nayak**

University Mentor Name: **Prof. Veenita Rathore**

Academic Year: **2025-2026**

Table of Content

1. Introduction

1.1 Scope of the Document

This High-Level Design (HLD) document provides a comprehensive architectural blueprint for the Marketing Content Generator system. It defines the system architecture, component design, data flow, interfaces, and non-functional requirements. This document serves as a technical reference for developers, architects, and stakeholders to understand how the system will be built and how different components interact with each other.

➢ **Scope Includes:**

- System architecture and component breakdown
- Application design patterns and frameworks
- Data models and storage mechanisms
- API specifications and interfaces
- Security and performance considerations

1.2 Intended Audience

- **Development Team**: To understand system architecture and implementation details
- **Project Managers:** To track technical progress and resource allocation
- **Technical Architects:** To review design decisions and ensure best practices
- **QA Team:** To understand system behavior for test planning
- **Stakeholders:** To validate that business requirements are addressed

1.3 System Overview

The Marketing Content Generator is a specialized GenAI-powered application that automates the creation of professional marketing content. The system leverages Large Language Models (LLMs) and sophisticated prompt engineering to transform basic user inputs into high-quality, industry-standard marketing materials such as ad copies, social media posts, email campaigns, and product descriptions.

➢ **Key Capabilities:**

- AI-powered content generation using LLM APIs
- Template-based output formatting
- Tone and style consistency maintenance
- Vector database for context storage and retrieval
- User-friendly web interface for content creation

2. System Design

2.1 Application Design

Architecture Pattern: Microservices-based Architecture with Layered Design. The important elements, their connections, interactions, and manner in which these interact with each other in order to create a complete system are identified. The complex ideas are communicated in a simplified form in a manner that is understandable, making these a blueprint that can be shared with stakeholders that include project managers, developers, architects, as well as non-technical stakeholders.

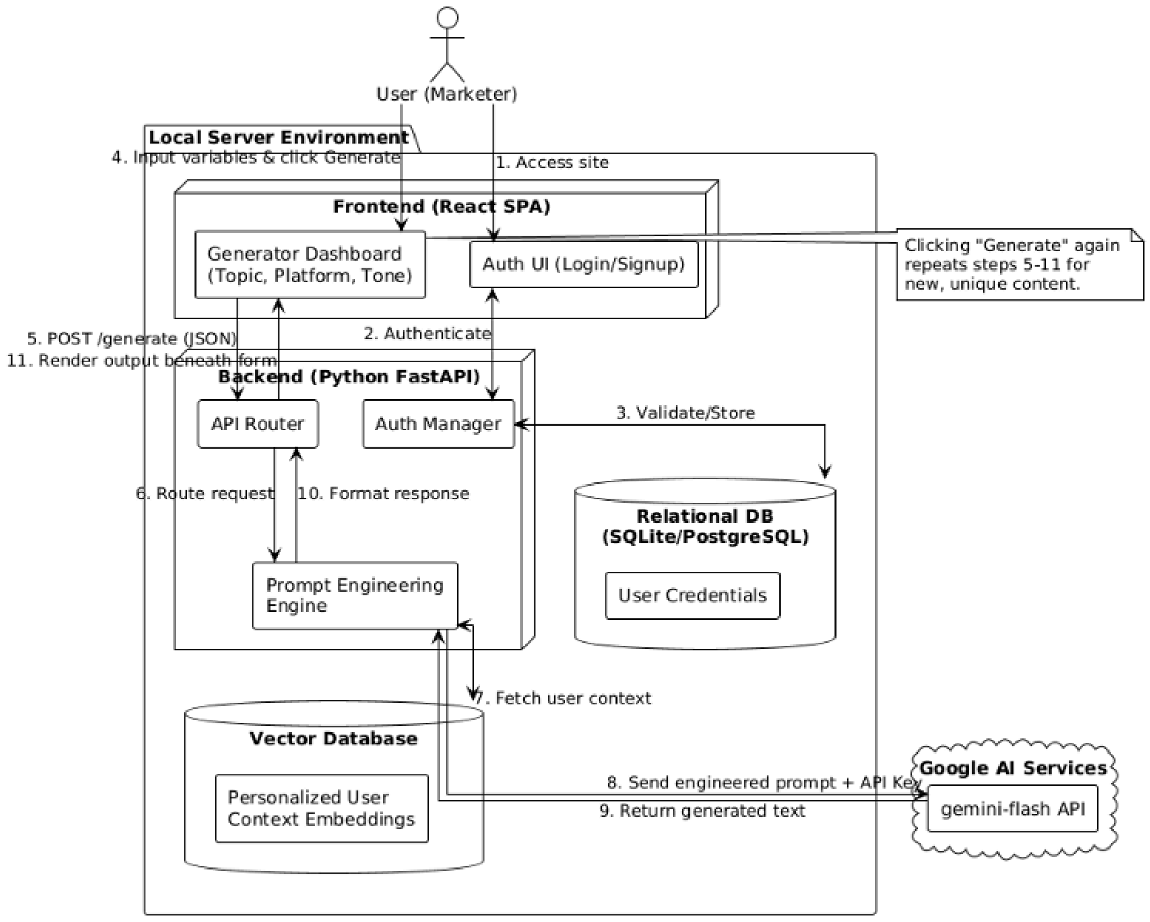


Figure 1.1 Architecture Diagram

➢ **Technology Stack:**

- Frontend: React.js / Streamlit (Python-based UI)
- Backend: Python (FastAPI/Flask)
- AI/ML Layer: OpenAI GPT-4 / Claude / Local LLM integration
- Vector Database: Pinecone / ChromaDB / FAISS
- Database: PostgreSQL / MongoDB (for metadata and user data)
- Cache: Redis
- Deployment: Docker containers on AWS/GCP/Azure

```
┌─────────────────────────────────────────────┐
│        Presentation Layer          │
│   (React.js UI / Streamlit Interface)  │
├─────────────────────────────────────────────┤
│        Application Layer           │
│   (API Gateway, Authentication,        │
│    Request Validation, Rate Limiting)   │
├─────────────────────────────────────────────┤
│        Business Logic Layer         │
│   (Content Generation Service,        │
│    Prompt Engineering Engine,         │
│    Template Management)             │
├─────────────────────────────────────────────┤
│        Data Access Layer           │
│   (Vector DB, Relational DB,          │
│    Cache Layer, LLM API Client)        │
├─────────────────────────────────────────────┤
│        External Services           │
│   (OpenAI API, Vector Database,        │
│    Authentication Service)          │
└─────────────────────────────────────────────┘
```

Figure 1.2 Architecture Work Flow

2.2 Process Flow

Use Case 1: Content Generation Flow

1. User Input: User enters topic, content type (ad copy, blog, social post), tone preference, and target audience
2. Validation: System validates input parameters and user authentication
3. Context Retrieval: System queries Vector DB for relevant templates and examples
4. Prompt Construction: Prompt Engineering module constructs optimized prompt with context
5. LLM Processing: Request sent to LLM API with constructed prompt
6. Post-Processing: Generated content formatted according to industry standards
7. Storage: Content saved to database with metadata

Use Case 2: Template Management Flow

1. Admin uploads new marketing templates with style guidelines
2. Templates processed and embedded into Vector DB
3. System indexes templates by category, tone, and use case
4. Templates available for future content generation

2.3 Information Flow

Data Flow Diagram Description:

Input Data Flow:

User Request → API Gateway → Authentication → Input Parser → Validation Engine → Context Assembler → Prompt Builder

Output Data Flow:

Quality Checker → Content Storage → Cache Layer → Response Builder → API Gateway → User Interface

Key Data Elements:

- User Profile Data (preferences, history, subscription tier)
- Content Templates (marketing frameworks, brand guidelines)
- Generated Content (ad copies, social posts, email templates)
- Analytics Data (usage metrics, content performance)

2.4 Components Design

Component-based diagrams are essential tools in software engineering, providing a visual representation of a system's structure by showcasing its various components and their interactions. These diagrams simplify complex systems, making it easier for developers to design, understand, and communicate the architecture.

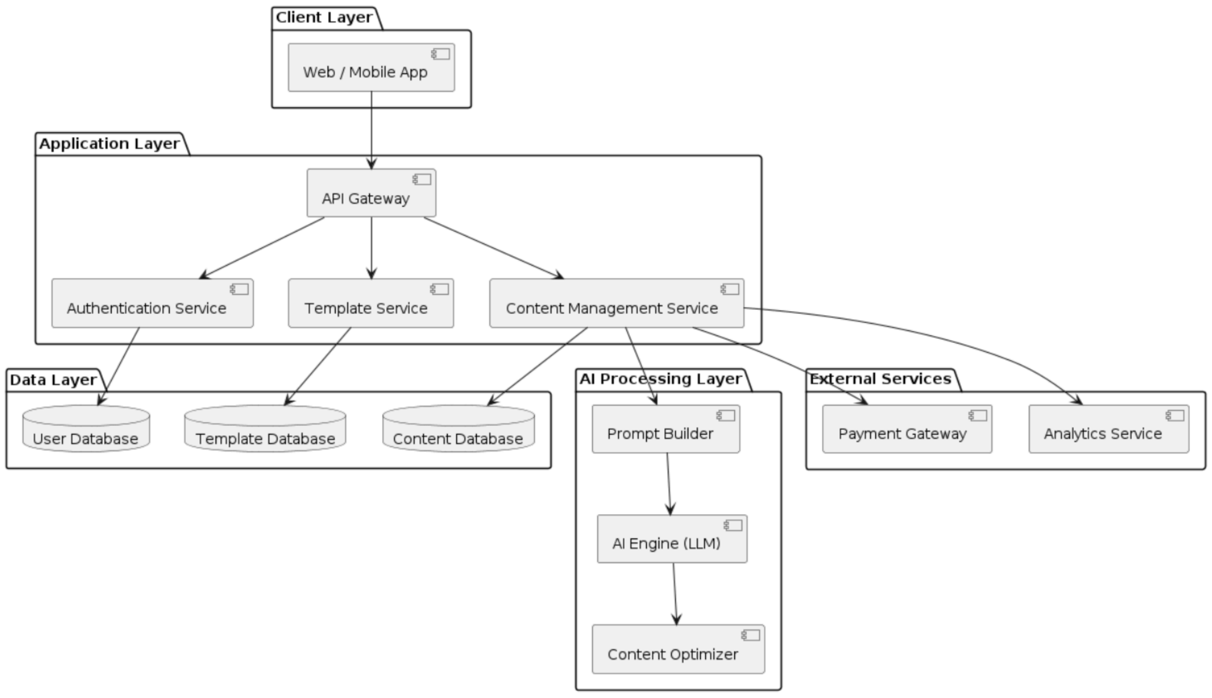


Figure 2.1 Component Diagram

Component 1: User Interface Module

- Purpose: Web-based interface for user interaction
- Features: Input forms, template selection, content preview, history view
- Tech: React.js with Material-UI or Streamlit for rapid prototyping

Component 2: API Gateway

- Purpose: Single entry point for all client requests
- Features: Rate limiting, request routing, authentication, logging
- Tech: FastAPI with middleware

Component 3: Authentication Service

- Purpose: User identity management and access control
- Features: JWT token-based auth, OAuth integration, role management
- Tech: Auth0 / Firebase Auth / Custom JWT implementation

Component 4: Content Generation Engine

- Purpose: Core AI content creation logic
- Features: Prompt templates, context management, output formatting
- Tech: Python with LangChain framework

Component 5: Prompt Engineering Module

- Purpose: Optimize prompts for high-quality output
- Features: Dynamic prompt construction, few-shot learning, chain-of-thought prompting
- Tech: Custom prompt templates with Jinja2

2.5 Key Design Considerations

Scalability:

- Horizontal scaling using container orchestration (Kubernetes)
- Stateless application design for easy replication
- Async processing for LLM API calls using Celery/RabbitMQ

Reliability:

- Circuit breaker pattern for external API failures
- Retry mechanisms with exponential backoff
- Fallback to cached templates if LLM service unavailable

Maintainability:

- Modular component design with clear interfaces

- Comprehensive logging and monitoring
- CI/CD pipeline for automated testing and deployment

Extensibility:

- Plugin architecture for new content types
- Configurable prompt templates
- Support for multiple LLM providers (OpenAI, Anthropic, Local models)

Security:

- Input sanitization to prevent prompt injection
- API key management using environment variables/secrets manager
- HTTPS/TLS for all communications
- Data encryption at rest and in transit

3. Data Design

3.1 Data Model

A Data Flow Diagram (DFD) is a graphical tool used to represent how data moves through a system. It shows data inputs, outputs, data stores, and the processes that transform the data. DFDs provide a high-level overview of system functionality and are widely used in structured analysis due to their simplicity and clarity for both technical and non-technical users.

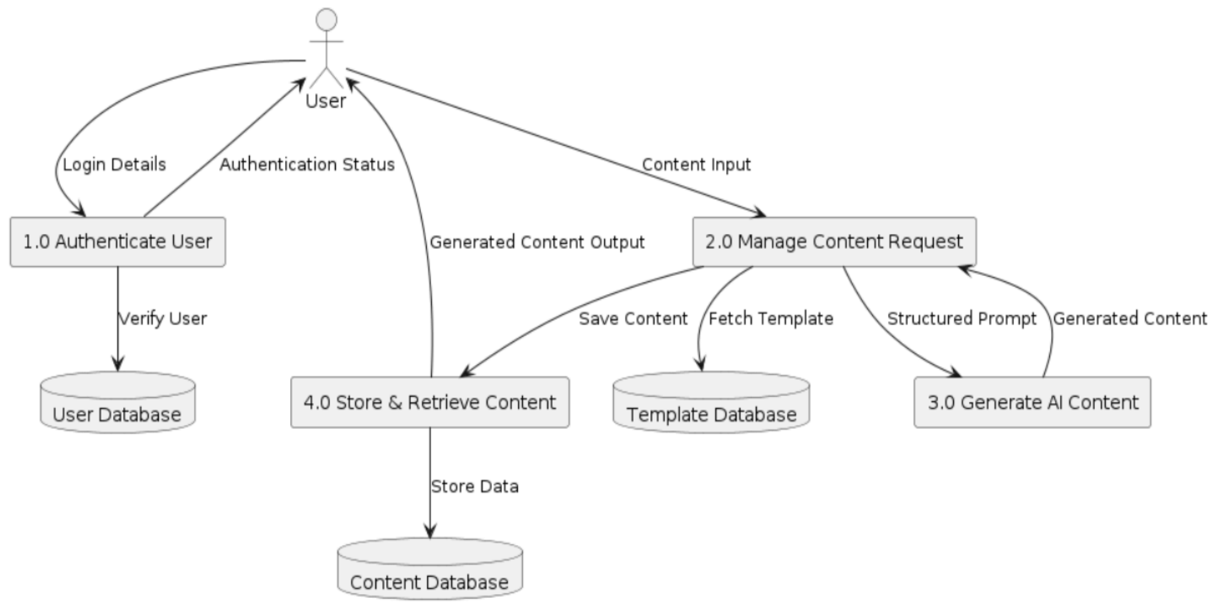


Figure 3.1 Data Flow Diagram

Entity Relationship Overview:

➢ **User Entity:**

- user_id (PK)
- email
- password_hash
- subscription_tier
- created_at

➢ **Content Generation Entity:**

- content_id (PK)
- user_id (FK)
- content_type
- input_parameters (JSON)
- generated_output (JSON)

➢ **Template Entity:**

- template_id (PK)
- template_name
- embedding_vector
- created_by
- is_active

➢ **Usage Analytics Entity:**

- log_id (PK)
- response_time
- status_code
- tokens_consumed

➢ **Vector DB Schema (ChromaDB/Pinecone):**

- ID: Unique identifier
- Embedding: 1536-dimensional vector (OpenAI text-embedding-ada-002)

3.2 Data Access Mechanism

➢ **Relational Database (PostgreSQL):**

- SQLAlchemy ORM for Python integration
- Connection pooling (PgBouncer)
- Read replicas for scaling read operations
- Automated backups with point-in-time recovery

➢ **Vector Database (ChromaDB):**

- Similarity search using cosine distance
- Metadata filtering for category-based retrieval

- Batch embedding generation for efficiency
- Persistent storage with optional cloud backup

➢ **Data Access Patterns:**

- Write-heavy: Content generation logs, user analytics
- Read-heavy: Template retrieval, user history
- Search-heavy: Vector similarity search for context

3.3 Data Retention Policies

➢ **User Data:**

- Active accounts: Retain indefinitely until account deletion
- Deleted accounts: Soft delete for 30 days, then hard delete
- Inactive accounts (no login for 2 years): Archive and notify user

➢ **Generated Content:**

- Free tier: Retain for 30 days
- Premium tier: Retain for 2 years
- Enterprise: Custom retention as per contract

➢ **Analytics Data:**

- Raw logs: 90 days
- Aggregated metrics: 2 years
- Usage statistics: Indefinite (anonymized)

➢ **Vector DB Data:**

- Active templates: Indefinite
- Deprecated templates: Archive after 6 months
- Temporary context: 7 days

3.4 Data Migration

Data migration from Version 1 to Version 2 will be handled in a structured and controlled manner to ensure data integrity and zero downtime.
During the migration process, new template categories will be created in parallel while existing templates will be backfilled with the required new metadata.
A gradual rollout strategy will be implemented using feature flags to minimize risk and allow controlled exposure of new features.

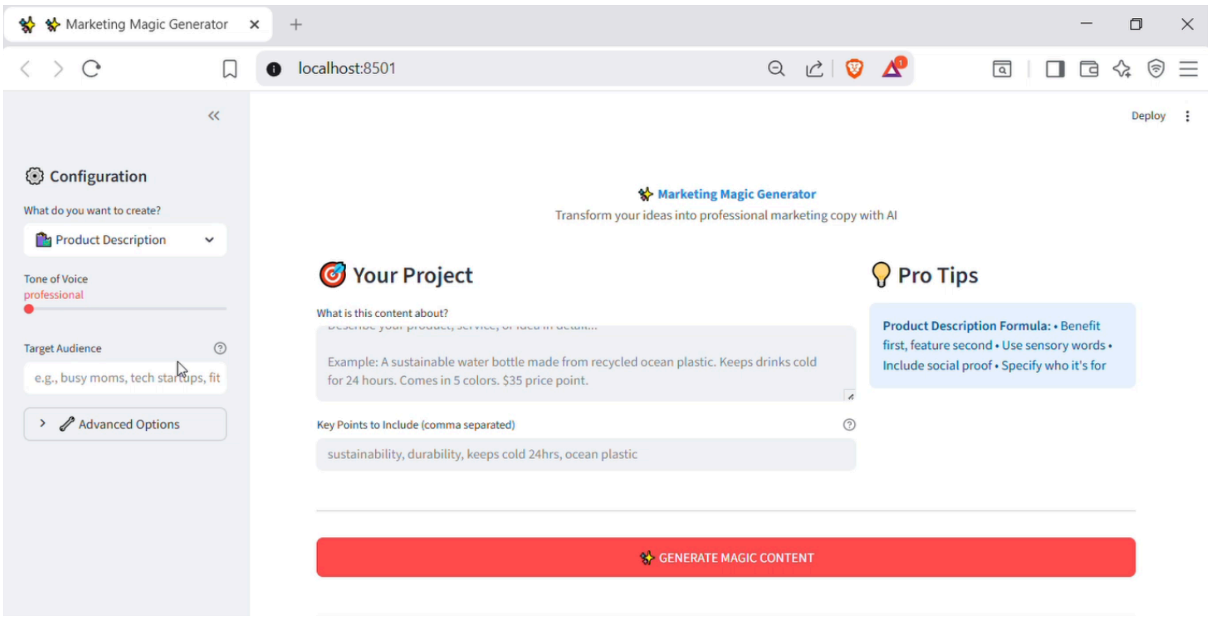
Key Points :

- Parallel creation of new template categories with metadata backfilling
- Alembic-based schema migrations with blue-green deployment

- Pre and post migration data validation scripts
- Backup and parallel rebuilding of vector database indexes
- Snapshot-based and automated rollback mechanisms with feature flags

4. Interfaces

User Interface (UI) 1:



Marketing Magic Generator
Transform your ideas into professional marketing copy with AI

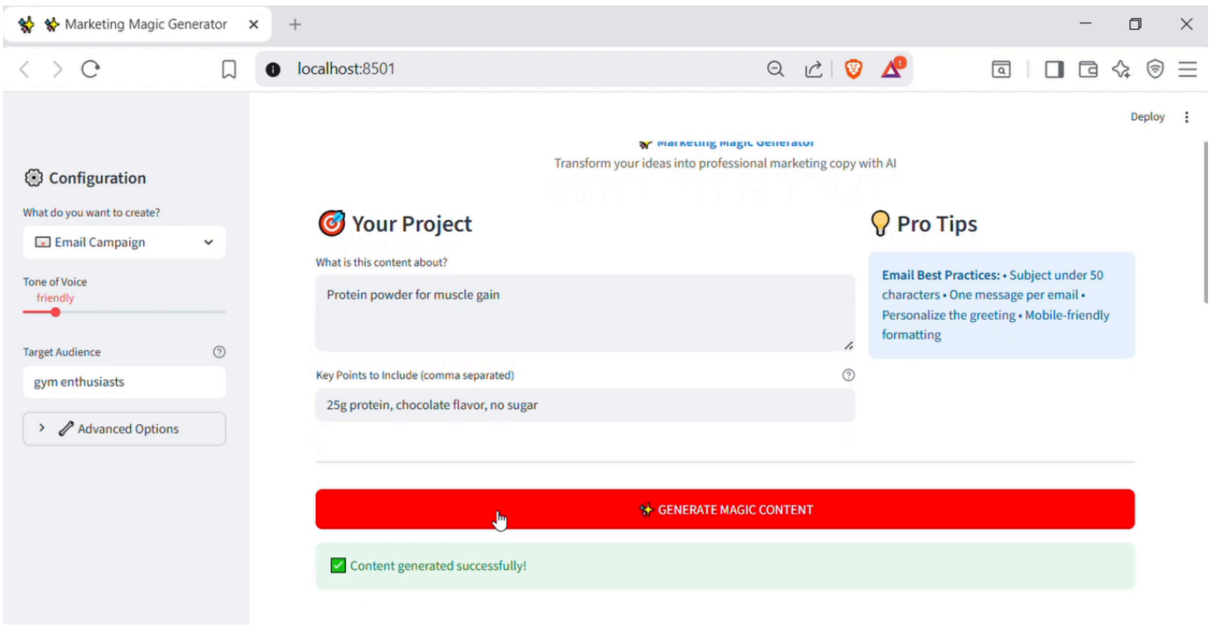
What do you want to create?: Product Description
Tone of Voice: professional
Target Audience: e.g., busy moms, tech startups, fit

Your Project
What is this content about?
Example: A sustainable water bottle made from recycled ocean plastic. Keeps drinks cold for 24 hours. Comes in 5 colors. $35 price point.
Key Points to Include (comma separated): sustainability, durability, keeps cold 24hrs, ocean plastic

Pro Tips
Product Description Formula: • Benefit first, feature second • Use sensory words • Include social proof • Specify who it's for

Figure 4.1 Snapshot of Dashboard

User Interface (UI) 2:



Marketing Magic Generator
Transform your ideas into professional marketing copy with AI

What do you want to create?: Email Campaign
Tone of Voice: friendly
Target Audience: gym enthusiasts

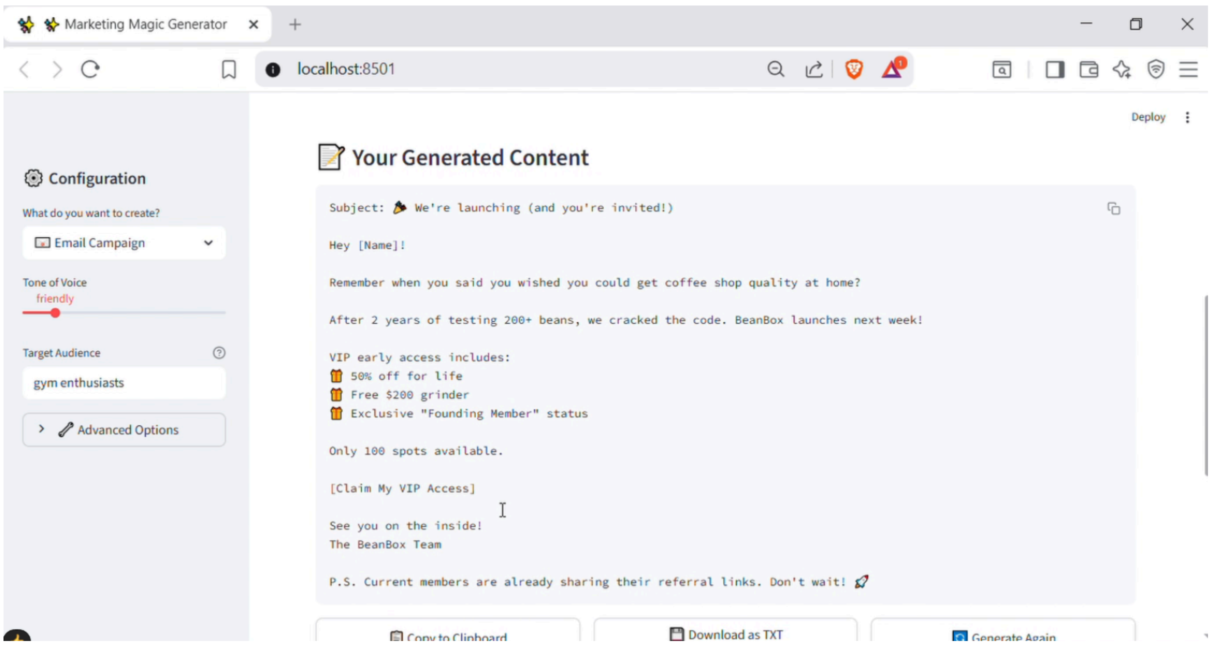
Your Project
What is this content about?: Protein powder for muscle gain
Key Points to Include (comma separated): 25g protein, chocolate flavor, no sugar

Pro Tips
Email Best Practices: • Subject under 50 characters • One message per email • Personalize the greeting • Mobile-friendly formatting

Content generated successfully!

Figure 4.2 Snapshot of Input Entered

User Interface (UI) 3:



```
Subject: 🎉 We're launching (and you're invited!)

Hey [Name]!

Remember when you said you wished you could get coffee shop quality at home?

After 2 years of testing 200+ beans, we cracked the code. BeanBox launches next week!

VIP early access includes:
🎁 50% off for life
🎁 Free $200 grinder
🎁 Exclusive "Founding Member" status

Only 100 spots available.

[Claim My VIP Access]

See you on the inside!
The BeanBox Team

P.S. Current members are already sharing their referral links. Don't wait! 🚀
```

Figure 4.3 Snapshot of Content Generated Output

➢ **Web Application: Responsive design supporting desktop and mobile**
- Dashboard: Content creation interface with input forms
- Template Gallery: Browse and select marketing templates
- History: View and manage previously generated content
- Settings: User preferences and API key management
- Admin Panel: Template management and analytics (Admin only)

➢ **External System Interfaces:**

- LLM API Interface: RESTful API integration with OpenAI/Anthropic
- Authentication: API Key-based
- Protocol: HTTPS with JSON payloads
- Rate limiting: Handled via exponential backoff

➢ **Database: SQLAlchemy ORM for relational, custom client for Vector DB**

- API Interface Specifications:
- RESTful architecture
- JSON data format
- HTTP status codes for error handling
- Versioning in URL (/api/v1/)

5. State and Session Management

➢ **Session Management:**

- Stateless Architecture: JWT tokens for authentication state
- Token Storage: Client-side (HTTP-only cookies or localStorage)
- Token Refresh: Short-lived access tokens (15 min), long-lived refresh tokens (7 days)
- Session Invalidation: Token blacklist on logout

➢ **Application State:**

- Client State: React Context API or Redux for UI state
- Server State: Database as source of truth
- Cache State: Redis for temporary state (drafts, partial inputs)

➢ **Content Generation State:**

- Pending: Request received, queued for processing
- Processing: Active LLM API call
- Completed: Content generated and stored
- Failed: Error occurred, retry or manual intervention needed

➢ **State Persistence:**

- Auto-save user inputs to localStorage (draft feature)
- Server-side session storage for multi-step workflows
- Distributed session store (Redis) for horizontal scaling

6. Caching

➢ **Layer 1: Browser Cache (Client-Side**)

- Static assets (JS, CSS, Images) → 1 week
- Template metadata → 1 hour
- User preferences → Session duration

➢ **Layer 2: CDN Cache (Cloudflare / AWS CloudFront)**

- Static content fast delivery
- Public template API responses → 1 hour

➢ **Layer 3: Application Cache (Redis)**

- Generated content cache (TTL: 12 hours, key = input hash)
- Frequently used templates (TTL: 1 hour)
- User sessions (TTL: 24 hours)
- Rate limiting tracking (1 minute sliding window)

➢ **Layer 4: Database Cache**

- PostgreSQL shared buffers
- Vector DB in-memory index for frequently used vectors

➢ **Cache Management**

- TTL-based automatic expiry
- Event-based invalidation (template update par clear)
- Version-based keys during deployments
- Cache warming: popular templates preload + background refresh

7. Non-Functional Requirements

7.1 Security Aspects

Authentication & Authorization:

- JWT-based stateless authentication
- Role-based access control (RBAC): User, Admin, Super Admin
- OAuth 2.0 integration for social login (Google, LinkedIn)
- Multi-factor authentication (MFA) for admin accounts

Data Security:

- Encryption at Rest: AES-256 for database storage
- Encryption in Transit: TLS 1.3 for all communications
- API Key Management: AWS Secrets Manager or HashiCorp Vault
- PII Protection: Masking of sensitive data in logs

Input Validation:

- Strict input sanitization to prevent SQL injection and XSS
- Prompt injection prevention: Input filtering and output encoding
- File upload validation (if supporting image uploads)
- Rate limiting to prevent abuse

API Security:

- API key validation for all endpoints
- CORS policy configuration
- Request size limits (prevent DoS)
- IP whitelisting for admin endpoints

Audit & Compliance:

- Comprehensive audit logging for all data access

- GDPR compliance: Right to erasure, data portability
- Regular security audits and penetration testing
- Dependency scanning for vulnerabilities

Infrastructure Security:

- Network segmentation (VPC, private subnets)
- DDoS protection (AWS Shield, CloudFlare)
- Web Application Firewall (WAF) rules
- Regular OS and dependency patching

7.2 Performance Aspects

Response Time Targets:

- UI Load Time: < 2 seconds (95th percentile)
- API Response Time: < 500ms for cached responses
- Content Generation: < 5 seconds (dependent on LLM API)
- Database Queries: < 100ms for simple queries

Throughput Requirements:

- Concurrent Users: Support 1000+ simultaneous users
- Requests per Second: 500+ RPS sustained
- Content Generation: 100+ generations per minute

Scalability:

- Horizontal Scaling: Auto-scaling based on CPU/memory metrics
- Database Scaling: Read replicas for query load, sharding for write load
- Caching Scaling: Redis Cluster for distributed caching

Resource Optimization:

- Lazy Loading: Load templates on demand
- Pagination: API responses paginated (20 items per page)
- Compression: Gzip/Brotli for API responses
- Connection Pooling: Database and HTTP connection reuse

Disaster Recovery:

- RTO (Recovery Time Objective): < 4 hours
- RPO (Recovery Point Objective): < 15 minutes
- Multi-Region Deployment: Active-passive setup for critical components
- Automated Backups: Daily snapshots with cross-region replication

Load Testing:

- Simulate 10x expected traffic for peak load testing
- Chaos engineering practices (random failure injection)
- Regular performance benchmarking

8. References

1. OpenAI API Documentation: https://platform.openai.com/docs
2. LangChain Framework: https://langchain.readthedocs.io/
3. ChromaDB Documentation: https://docs.trychroma.com/
4. FastAPI Documentation: https://fastapi.tiangolo.com/
5. React.js Documentation: https://reactjs.org/docs/
6. PostgreSQL Documentation: https://www.postgresql.org/docs/
7. OWASP Security Guidelines: https://owasp.org/www-project-top-ten/
8. GDPR Compliance Guide: https://gdpr.eu/
9. Microservices Patterns: "Building Microservices" by Sam Newman

Technical Standards:

- REST API Design: RFC 7231
- JSON Schema: Draft 2020-12
- JWT: RFC 7519
- OAuth 2.0: RFC 6749