

Problem Statement 12: Social Media Management Agent

AI-Powered Intelligent Social Media Management and Content Optimization Platform

Problem Overview

Modern businesses struggle with managing multiple social media platforms effectively, creating engaging content consistently, and optimizing their social media presence for maximum reach and engagement. The challenge involves content creation, scheduling, audience analysis, performance tracking, and real-time engagement across diverse platforms with varying algorithms and audience behaviors.

Key Requirements

Core Functionality

- **Multi-Platform Management:** Unified dashboard for Facebook, Instagram, Twitter/X, LinkedIn, TikTok, YouTube
- **AI Content Generation:** Automated creation of posts, captions, hashtags, and visual content
- **Intelligent Scheduling:** Optimal timing based on audience behavior and platform algorithms
- **Audience Analytics:** Deep insights into follower demographics, engagement patterns, and growth trends
- **Performance Optimization:** Real-time A/B testing and content optimization recommendations
- **Brand Voice Consistency:** AI-powered brand voice maintenance across all platforms
- **Crisis Management:** Automated sentiment monitoring and response recommendations
- **Influencer Collaboration:** Discovery, outreach, and campaign management tools

Advanced Features

- **Visual Content AI:** Automated image/video generation, editing, and optimization
- **Trend Analysis:** Real-time trend detection and content adaptation
- **Competitor Intelligence:** Automated competitor analysis and benchmarking
- **Social Listening:** Brand mention monitoring and sentiment analysis
- **ROI Analytics:** Comprehensive social media ROI measurement and reporting
- **Compliance Management:** Platform policy adherence and content moderation

Data Requirements

Social Media Data

- **Platform APIs:** Facebook Graph API, Instagram Basic Display API, Twitter API v2, LinkedIn API, TikTok API, YouTube Data API
- **Content Performance:** Engagement metrics, reach, impressions, click-through rates, conversion data
- **Audience Data:** Demographics, interests, behavior patterns, optimal posting times
- **Competitor Data:** Public posts, engagement rates, follower growth, content strategies

Content Assets

- **Brand Assets:** Logos, color schemes, fonts, brand guidelines, voice and tone documentation
- **Media Library:** Images, videos, graphics, templates, previously successful content
- **Content Calendar:** Planned campaigns, seasonal content, product launches, events

Business Context

- **Industry Data:** Sector-specific trends, seasonal patterns, regulatory requirements
- **Customer Data:** Target audience personas, customer journey stages, conversion funnels
- **Business Metrics:** Sales data, website traffic, lead generation, customer acquisition costs

Technical Themes

AI/ML Components

- **Natural Language Processing:** Content generation, sentiment analysis, brand voice consistency
- **Computer Vision:** Image analysis, visual content generation, brand compliance checking
- **Recommendation Systems:** Content optimization, optimal posting times, hashtag suggestions
- **Predictive Analytics:** Trend forecasting, performance prediction, audience growth modeling
- **Reinforcement Learning:** Dynamic content optimization based on engagement feedback

Integration Architecture

- **Social Media APIs:** Multi-platform connectivity with rate limiting and error handling
- **Content Management:** Digital asset management, version control, approval workflows
- **Analytics Platforms:** Google Analytics, Facebook Analytics, platform-specific insights
- **CRM Integration:** Customer relationship management, lead tracking, sales attribution
- **Marketing Automation:** Email marketing, lead nurturing, cross-channel campaigns

Real-time Processing

- **Stream Processing:** Real-time social listening, mention monitoring, engagement tracking
- **Event-Driven Architecture:** Automated responses, alert systems, workflow triggers
- **Caching Strategies:** Content delivery optimization, API response caching, performance enhancement
- **Queue Management:** Content scheduling, bulk operations, background processing

Expected Business Outcomes

Efficiency Gains

- **Time Savings:** 70% reduction in manual social media management tasks
- **Content Production:** 5x increase in content output through AI automation
- **Response Time:** 90% faster response to customer inquiries and mentions
- **Campaign Setup:** 80% reduction in campaign creation and setup time

Performance Improvements

- **Engagement Rates:** 40% increase in average engagement across platforms
- **Follower Growth:** 60% improvement in organic follower acquisition
- **Content Performance:** 50% increase in content reach and impressions
- **Conversion Rates:** 35% improvement in social media to sales conversion

Strategic Benefits

- **Brand Consistency:** 95% brand voice consistency across all platforms and content
- **Market Intelligence:** Real-time competitive insights and trend identification

- **ROI Measurement:** Comprehensive attribution and social media ROI tracking
- **Risk Mitigation:** Proactive crisis detection and management capabilities

Implementation Approach

Phase 1: Foundation (Months 1-2)

- Multi-platform API integration and authentication
- Basic content scheduling and publishing capabilities
- Core analytics dashboard and reporting
- User management and permission systems

Phase 2: Intelligence (Months 3-4)

- AI content generation and optimization
- Audience analysis and segmentation
- Automated scheduling optimization
- Basic social listening and monitoring

Phase 3: Automation (Months 5-6)

- Advanced AI features and personalization
- Comprehensive competitor analysis
- Influencer discovery and management
- Crisis management and response automation

Phase 4: Optimization (Months 7-8)

- Advanced analytics and ROI measurement
- A/B testing automation and optimization
- Cross-platform campaign orchestration
- Enterprise features and scalability enhancements

Success Metrics

Operational Metrics

- **Platform Coverage:** Support for 6+ major social media platforms
- **Content Volume:** 1000+ posts managed per month per client
- **Processing Speed:** <2 seconds for content generation and optimization
- **Uptime:** 99.9% platform availability and reliability

Business Impact Metrics

- **Client Retention:** 95% annual client retention rate
- **Time to Value:** 30 days to measurable improvement in social media performance
- **Cost Efficiency:** 60% reduction in social media management costs
- **Revenue Growth:** 25% increase in social media-driven revenue for clients

User Experience Metrics

- **User Adoption:** 90% feature adoption rate within 60 days
- **User Satisfaction:** 4.8/5.0 average user satisfaction score
- **Learning Curve:** <2 hours to platform proficiency for new users
- **Support Resolution:** 95% of issues resolved within 24 hours

This comprehensive social media management platform will transform how businesses approach social media marketing, providing AI-powered automation, deep analytics, and strategic optimization to maximize social media ROI while maintaining authentic brand engagement across all major platforms. # Product Requirements Document (PRD) ## Social Media Management Agent - AI-Powered Intelligent Social Media Management and Content Optimization Platform

Building upon README foundation for comprehensive business requirements and strategic objectives

ETVX Framework

ENTRY CRITERIA

- âœ… README completed with problem overview, key requirements, and technical approach
- âœ… Market research on social media management landscape and competitive analysis
- âœ… Stakeholder interviews with marketing teams, social media managers, and business owners
- âœ… Technical feasibility assessment for AI-powered social media automation

TASK

Define comprehensive business objectives, market analysis, user personas, success metrics, core features, technical requirements, constraints, assumptions, and risk assessment for the Social Media Management Agent platform.

VERIFICATION & VALIDATION

Verification Checklist: - [] Business objectives align with market opportunities and user needs - [] Success metrics are measurable and achievable within defined timelines - [] Core features address all major social media management pain points - [] Technical requirements support scalability and multi-platform integration

Validation Criteria: - [] PRD validated with marketing executives and social media professionals - [] Business case validated with financial stakeholders and investors - [] Technical requirements validated with AI/ML and platform integration experts - [] Market positioning validated with competitive intelligence and industry analysis

EXIT CRITERIA

- âœ… Complete business requirements ready for functional specification
- âœ… Clear success metrics and KPIs defined for product development
- âœ… Technical foundation established for architecture design
- âœ… Risk mitigation strategies identified for major project risks
- âœ… Foundation prepared for Functional Requirements Document (FRD) development

1. Business Objectives

1.1 Primary Objectives

Revenue Targets: - **Annual Recurring Revenue (ARR):** \$50M by Year 3 - **Customer Acquisition:** 10,000+ active business clients by Year 3 - **Market Share:** Capture 5% of the \$17.7B social media management market - **Average Revenue Per User (ARPU):** \$5,000 annually per enterprise client

Operational Excellence: - **Platform Efficiency:** 70% reduction in manual social media management tasks - **Content Performance:** 40% improvement in average engagement rates across platforms - **Time to Value:** Clients achieve measurable ROI within 30 days of onboarding - **Customer Success:** 95% annual client retention rate with 4.8/5.0 satisfaction score

Strategic Positioning: - **Market Leadership:** Establish as the leading AI-powered social media management platform - **Technology Innovation:** Pioneer advanced AI capabilities in social media automation - **Enterprise Adoption:** Become the preferred solution for Fortune 500 companies - **Global Expansion:** Support for 50+ countries and 20+ languages by Year 2

1.2 Secondary Objectives

Ecosystem Development: - **Partner Network:** Integrate with 100+ marketing and business tools - **Developer Platform:** Launch API ecosystem with 500+ third-party integrations - **Marketplace:** Create content template and automation marketplace - **Community Building:** Establish 50,000+ member professional community

Innovation Leadership: - **AI Research:** Publish 20+ research papers on social media AI applications - **Patent Portfolio:** File 50+ patents for social media automation technologies - **Industry Recognition:** Win 10+ major industry awards and recognitions - **Thought Leadership:** Establish executives as recognized industry experts

2. Market Analysis

2.1 Market Size and Opportunity

Total Addressable Market (TAM): - **Global Social Media Management Market:** \$17.7B (2024) â†’ \$41.6B (2030) - **AI in Marketing Market:** \$27.4B (2024) â†’ \$107.5B (2030) - **Content Marketing Automation:** \$5.8B (2024) â†’ \$14.2B (2030) - **Social Media Analytics Market:** \$8.2B (2024) â†’ \$24.1B (2030)

Serviceable Addressable Market (SAM): - **Enterprise Social Media Management:** \$8.9B - **AI-Powered Marketing Automation:** \$12.3B - **Multi-Platform Content Management:** \$4.7B - **Social Media Intelligence:** \$3.2B

Serviceable Obtainable Market (SOM): - **Target Market Segment:** \$2.1B (5-year opportunity) - **Realistic Market Capture:** \$885M (3% market share by Year 5) - **Primary Geographic Markets:** North America (60%), Europe (25%), Asia-Pacific (15%)

2.2 Competitive Landscape

Direct Competitors: - **Hootsuite:** Market leader, \$200M+ ARR, strong enterprise presence - **Sprout Social:** \$300M+ ARR, excellent analytics, public company - **Buffer:** \$20M+ ARR, SMB focus, simple interface - **Later:** \$50M+ ARR, visual content focus, Instagram specialization

Competitive Advantages: - **Advanced AI:** Superior content generation and optimization capabilities - **Multi-Platform Intelligence:** Unified analytics across all major platforms - **Real-Time Optimization:** Dynamic content adjustment based on performance - **Enterprise Security:** SOC 2, GDPR compliance with enterprise-grade security

Market Gaps: - **AI-Native Platform:** Most competitors added AI as feature, not core architecture - **Cross-Platform Intelligence:** Limited unified analytics across platforms - **Real-Time Adaptation:** Static scheduling vs.Â dynamic optimization - **Industry Specialization:** Generic solutions vs.Â industry-specific optimization

2.3 Target Market Segments

Primary Segment: Enterprise Businesses (500+ employees) - **Market Size:** 200,000+ companies globally - **Budget Range:** \$50,000 - \$500,000 annually for social media management - **Pain Points:** Brand consistency, compliance, ROI measurement, scale management - **Decision Makers:** CMOs, Marketing Directors, Digital Marketing Managers

Secondary Segment: Mid-Market Businesses (50-500 employees) - **Market Size:** 2M+ companies globally - **Budget Range:** \$5,000 - \$50,000 annually for social media management - **Pain Points:** Resource constraints, expertise gaps, time management - **Decision Makers:** Marketing Managers, Business Owners, Digital Specialists

Tertiary Segment: Agencies and Consultants - **Market Size:** 50,000+ marketing agencies globally - **Budget Range:** \$10,000 - \$100,000 annually per agency - **Pain Points:** Client management, scalability, reporting, white-labeling - **Decision Makers:** Agency Owners, Account Directors, Operations Managers

3. User Personas

3.1 Primary Persona: Enterprise Marketing Director

Demographics: - **Name:** Sarah Chen - **Age:** 35-45 - **Role:** Director of Digital Marketing at Fortune 500 company - **Experience:** 10+ years in digital marketing, 5+ years in social media - **Team Size:** Manages 8-12 marketing professionals

Goals and Motivations: - **Strategic Objectives:** Drive brand awareness and customer acquisition through social media - **Operational Goals:** Improve team efficiency and content performance - **Career Aspirations:** Demonstrate measurable ROI and business impact - **Innovation Focus:** Leverage cutting-edge technology for competitive advantage

Pain Points: - **Scale Challenges:** Managing multiple brands across numerous platforms - **Consistency Issues:** Maintaining brand voice across teams and platforms - **ROI Measurement:** Proving social media impact on business outcomes - **Resource Constraints:** Limited budget and personnel for comprehensive coverage

Technology Profile: - **Comfort Level:** High - experienced with marketing technology stack - **Current Tools:** Uses Salesforce, HubSpot, Google Analytics, Adobe Creative Suite - **Decision Process:** Evaluates ROI, security, integration capabilities, vendor stability - **Budget Authority:** \$100,000+ annual budget approval authority

3.2 Secondary Persona: SMB Marketing Manager

Demographics: - **Name:** Michael Rodriguez - **Age:** 28-38 - **Role:** Marketing Manager at 150-person SaaS company - **Experience:** 5-8 years in marketing, 3+ years in social media - **Team Size:** Manages 2-3 marketing team members

Goals and Motivations: - **Growth Objectives:** Scale social media presence to support business growth - **Efficiency Goals:** Maximize output with limited resources - **Skill Development:** Stay current with social media trends and best practices - **Business Impact:** Contribute directly to lead generation and sales

Pain Points: - **Time Constraints:** Wearing multiple hats, limited time for social media - **Expertise Gaps:** Need guidance on best practices and optimization - **Budget Limitations:** Must demonstrate clear ROI for tool investments - **Platform Complexity:** Difficulty managing multiple platforms effectively

Technology Profile: - **Comfort Level:** Medium-High - comfortable with SaaS tools - **Current Tools:** Uses Google Workspace, Slack, basic social media schedulers - **Decision Process:** Focuses on ease of use, cost-effectiveness, quick implementation - **Budget Authority:** \$5,000-\$25,000 annual budget approval

3.3 Tertiary Persona: Agency Account Manager

Demographics: - **Name:** Jessica Thompson - **Age:** 26-35 - **Role:** Senior Account Manager at digital marketing agency - **Experience:** 4-7 years in agency environment, 3+ years in social media - **Client Portfolio:** Manages 8-15 client accounts simultaneously

Goals and Motivations: - **Client Success:** Deliver exceptional results for diverse client portfolio - **Operational Efficiency:** Streamline workflows to handle more clients - **Professional Growth:** Build expertise in emerging social media technologies - **Agency Growth:** Contribute to agencyâ€™s competitive positioning

Pain Points: - **Client Diversity:** Managing different industries, brand voices, and requirements - **Reporting Burden:** Time-intensive client reporting and performance analysis - **Scalability Issues:** Difficulty scaling personalized service across clients - **Tool Fragmentation:** Using multiple tools creates inefficiency and errors

Technology Profile: - **Comfort Level:** High - power user of multiple marketing platforms - **Current Tools:** Uses project management tools, multiple social platforms, reporting tools - **Decision Process:** Evaluates white-labeling, client reporting, scalability features - **Influence Level:** Strong influence on agency tool

selection and recommendations

4. Success Metrics and KPIs

4.1 Business Performance Metrics

Revenue Metrics: - **Monthly Recurring Revenue (MRR):** Target \$4.2M by Year 3 - **Annual Contract Value (ACV):** Average \$60,000 for enterprise, \$12,000 for mid-market - **Customer Lifetime Value (CLV):** \$180,000 enterprise, \$36,000 mid-market - **Revenue Growth Rate:** 150% year-over-year for first 3 years

Customer Metrics: - **Customer Acquisition Cost (CAC):** <\$15,000 for enterprise, <\$3,000 for mid-market - **Customer Churn Rate:** <5% annually for enterprise, <10% for mid-market - **Net Promoter Score (NPS):** >70 across all customer segments - **Customer Satisfaction (CSAT):** >4.8/5.0 average rating

Market Metrics: - **Market Share:** 3% of addressable market by Year 3 - **Brand Recognition:** 80% aided awareness in target market - **Competitive Win Rate:** >60% in competitive evaluations - **Geographic Expansion:** 25 countries with localized offerings

4.2 Product Performance Metrics

Platform Metrics: - **System Uptime:** 99.9% availability with <1 hour monthly downtime - **Response Time:** <2 seconds for all user interactions - **Processing Speed:** <30 seconds for AI content generation - **Scalability:** Support 1M+ social media posts per day

Feature Adoption Metrics: - **AI Content Generation:** 85% of users actively using within 30 days - **Multi-Platform Publishing:** 90% of users publishing to 3+ platforms - **Analytics Dashboard:** 95% of users accessing weekly reports - **Automation Features:** 70% of users setting up automated workflows

User Experience Metrics: - **Time to First Value:** <2 hours from signup to first successful post - **Feature Discovery:** 80% of features used within 60 days - **User Engagement:** 4+ logins per week average - **Support Ticket Volume:** <2% of users requiring support monthly

4.3 Customer Impact Metrics

Efficiency Improvements: - **Time Savings:** 70% reduction in manual social media management tasks - **Content Volume:** 5x increase in content output through automation - **Response Time:** 90% faster response to social media interactions - **Campaign Setup:** 80% reduction in campaign creation time

Performance Improvements: - **Engagement Rates:** 40% average improvement across all platforms - **Follower Growth:** 60% improvement in organic follower acquisition - **Content Reach:** 50% increase in average content impressions - **Conversion Rates:** 35% improvement in social-to-sales conversion

Business Impact: - **ROI Measurement:** 300% average ROI within 12 months - **Brand Consistency:** 95% brand voice consistency across platforms - **Crisis Response:** 95% of potential issues identified and addressed proactively - **Competitive Intelligence:** 100% of major competitor activities tracked and analyzed

5. Core Features and Capabilities

5.1 Multi-Platform Management

Platform Integration: - **Supported Platforms:** Facebook, Instagram, Twitter/X, LinkedIn, TikTok, YouTube, Pinterest - **Unified Dashboard:** Single interface for managing all social media accounts - **Cross-Platform Publishing:** Simultaneous posting with platform-specific optimization - **Account Management:** Unlimited social media accounts per subscription

Content Management: - **Digital Asset Management:** Centralized media library with version control - **Content Calendar:** Visual planning interface with drag-and-drop scheduling - **Approval Workflows:** Multi-level content approval with stakeholder notifications - **Brand Asset Library:** Centralized storage for logos, templates, brand guidelines

5.2 AI-Powered Content Generation

Content Creation: - **Text Generation:** AI-powered captions, posts, and hashtag recommendations - **Visual Content:** Automated image creation, editing, and optimization - **Video Generation:** AI-assisted video creation and editing capabilities - **Personalization:** Content adaptation based on audience segments and platforms

Content Optimization: - **A/B Testing:** Automated testing of content variations with performance analysis - **Performance Prediction:** AI models predicting content performance before publishing - **Trend Integration:** Real-time trend detection and content adaptation - **Brand Voice Consistency:** AI monitoring and maintenance of brand voice across content

5.3 Intelligent Scheduling and Automation

Smart Scheduling: - **Optimal Timing:** AI-determined best posting times based on audience behavior - **Platform Algorithms:** Scheduling optimization for each platform's algorithm - **Time Zone Management:** Global scheduling with automatic time zone adjustments - **Frequency Optimization:** AI-recommended posting frequency per platform

Automation Workflows: - **Trigger-Based Actions:** Automated responses to mentions, comments, and messages - **Content Series:** Automated publication of content series and campaigns - **Cross-Platform Promotion:** Automatic content adaptation and cross-promotion - **Performance-Based Adjustments:** Dynamic scheduling based on real-time performance

5.4 Advanced Analytics and Insights

Performance Analytics: - **Unified Reporting:** Cross-platform analytics with consolidated metrics - **ROI Measurement:** Attribution tracking from social media to business outcomes - **Competitive Analysis:** Automated competitor monitoring and benchmarking - **Audience Insights:** Deep demographic and behavioral analysis

Predictive Intelligence: - **Trend Forecasting:** AI-powered prediction of emerging trends and topics - **Performance Forecasting:** Predictive models for content and campaign performance - **Audience Growth Modeling:** Forecasting follower growth and engagement trends - **Budget Optimization:** AI recommendations for social media budget allocation

5.5 Social Listening and Engagement

Monitoring Capabilities: - **Brand Mention Tracking:** Real-time monitoring across all major platforms - **Sentiment Analysis:** AI-powered sentiment tracking and trend analysis - **Crisis Detection:** Automated identification of potential PR issues - **Competitor Intelligence:** Comprehensive competitor activity monitoring

Engagement Management: - **Unified Inbox:** Centralized management of all social media interactions - **Automated Responses:** AI-powered response suggestions and automation - **Escalation Workflows:** Automatic routing of complex issues to appropriate team members - **Response Time Tracking:** Monitoring and optimization of response times

6. Technical Requirements

6.1 Platform Architecture

Scalability Requirements: - **User Capacity:** Support 100,000+ concurrent users - **Data Processing:** Handle 10M+ social media posts daily - **API Throughput:** 100,000+ API calls per minute - **Storage Capacity:** Petabyte-scale data storage with 99.99% durability

Performance Requirements: - **Response Time:** <2 seconds for all user interface interactions - **Content Generation:** <30 seconds for AI-powered content creation - **Report Generation:** <10 seconds for standard analytics reports - **Real-Time Processing:** <5 seconds for social listening and monitoring

Reliability Requirements: - **System Uptime:** 99.9% availability with planned maintenance windows - **Data Backup:** Real-time replication with 99.99% data durability - **Disaster Recovery:** <4 hour RTO and <1 hour RPO - **Error Handling:** Graceful degradation with user-friendly error messages

6.2 Integration Requirements

Social Media APIs: - **Platform Coverage:** Native integration with 8+ major social media platforms - **API Management:** Rate limiting, error handling, and automatic retry mechanisms - **Real-Time Webhooks:** Support for real-time notifications and updates - **Data Synchronization:** Bi-directional sync with platform-specific data models

Third-Party Integrations: - **Marketing Tools:** CRM, email marketing, marketing automation platforms - **Analytics Platforms:** Google Analytics, Adobe Analytics, custom analytics tools - **Content Tools:** Design software, video editing tools, stock photo services - **Business Systems:** ERP, customer support, project management tools

API and Webhook Support: - **RESTful APIs:** Comprehensive API for third-party integrations - **Webhook Notifications:** Real-time event notifications for external systems - **SDK Development:** Software development kits for popular programming languages - **API Documentation:** Comprehensive documentation with interactive examples

6.3 Security and Compliance

Data Security: - **Encryption:** AES-256 encryption for data at rest and TLS 1.3 for data in transit - **Access Controls:** Role-based access control with multi-factor authentication - **API Security:** OAuth 2.0, API key management, and rate limiting - **Network Security:** VPC isolation, firewall rules, and intrusion detection

Compliance Requirements: - **Data Privacy:** GDPR, CCPA, and regional privacy regulation compliance - **Security Standards:** SOC 2 Type II, ISO 27001 certification - **Industry Compliance:** HIPAA compliance for healthcare clients - **Audit Trails:** Comprehensive logging and audit trail capabilities

Platform Security: - **Social Media Security:** Secure token management and credential protection - **User Authentication:** Single sign-on (SSO) with enterprise identity providers - **Data Retention:** Configurable data retention policies with secure deletion - **Incident Response:** 24/7 security monitoring with automated threat detection

7. Constraints and Assumptions

7.1 Technical Constraints

Platform Limitations: - **API Rate Limits:** Social media platform API restrictions and quotas - **Data Access:** Limited historical data availability from some platforms - **Feature Parity:** Platform-specific features may not be available across all networks - **Real-Time Limitations:** Varying degrees of real-time data availability

Resource Constraints: - **Development Timeline:** 18-month initial development cycle - **Budget Limitations:** \$25M initial development and launch budget - **Team Size:** 50-person development team across multiple disciplines - **Infrastructure Costs:** Cloud infrastructure scaling costs and optimization

7.2 Business Assumptions

Market Assumptions: - **Market Growth:** Social media management market continues 15%+ annual growth - **Platform Stability:** Major social media platforms maintain current API access - **Regulatory Environment:** No major regulatory changes affecting social media automation - **Competitive Landscape:** Current competitive positioning remains stable

Customer Assumptions: - **Adoption Rate:** Enterprise customers willing to adopt AI-powered automation - **Budget Allocation:** Businesses increasing social media management budgets - **Technology Readiness:** Target customers have necessary technical infrastructure - **Change Management:** Organizations willing to modify existing workflows

7.3 Regulatory and Compliance Assumptions

Data Privacy: - **Regulation Stability:** Current data privacy regulations remain stable - **Cross-Border Data:** Continued ability to process data across international boundaries - **Platform Compliance:** Social media platforms maintain current data sharing policies - **User Consent:** Ability to obtain necessary user consents for data processing

Content Regulations: - **Platform Policies:** Social media platforms maintain current content policies - **Automated Content:** Continued acceptance of AI-generated content by platforms - **Advertising Regulations:** Stable regulatory environment for social media advertising - **Industry Standards:** Continued evolution of industry best practices and standards

8. Risk Assessment and Mitigation

8.1 Technical Risks

High Risk: Platform API Changes - Risk: Social media platforms modify or restrict API access - **Impact:** Core functionality disruption, feature limitations - **Probability:** Medium (30-40% over 3 years) - **Mitigation:** Diversified platform portfolio, direct platform partnerships, alternative data sources

Medium Risk: AI Model Performance - Risk: AI models fail to meet performance expectations - **Impact:** Reduced content quality, customer dissatisfaction - **Probability:** Low-Medium (20-30%) - **Mitigation:** Multiple model approaches, continuous training, human oversight options

Medium Risk: Scalability Challenges - Risk: Platform cannot scale to meet demand - **Impact:** Performance degradation, customer churn - **Probability:** Low (15-20%) - **Mitigation:** Cloud-native architecture, load testing, gradual rollout strategy

8.2 Business Risks

High Risk: Competitive Response - Risk: Major competitors launch similar AI-powered features - **Impact:** Reduced competitive advantage, pricing pressure - **Probability:** High (70-80% over 2 years) - **Mitigation:** Continuous innovation, patent protection, first-mover advantage

Medium Risk: Market Adoption - Risk: Slower than expected enterprise adoption of AI automation - **Impact:** Revenue shortfall, extended payback period - **Probability:** Medium (25-35%) - **Mitigation:** Phased rollout, customer education, proof-of-concept programs

Low Risk: Regulatory Changes - Risk: New regulations restrict social media automation - **Impact:** Feature limitations, compliance costs - **Probability:** Low (10-15%) - **Mitigation:** Regulatory monitoring, compliance-first design, legal partnerships

8.3 Operational Risks

Medium Risk: Talent Acquisition - Risk: Difficulty hiring qualified AI and social media experts - **Impact:** Development delays, increased costs - **Probability:** Medium (30-40%) - **Mitigation:** Competitive compensation, remote work options, university partnerships

Low Risk: Data Security Breach - Risk: Security incident compromising customer data - **Impact:** Reputation damage, legal liability, customer churn - **Probability:** Low (5-10%) - **Mitigation:** Security-first architecture, regular audits, incident response plan

This comprehensive PRD establishes the foundation for developing an AI-powered social media management platform that addresses real market needs while maintaining realistic expectations for technical implementation and business outcomes. # Functional Requirements Document (FRD) ## Social Media Management Agent - AI-Powered Intelligent Social Media Management and Content Optimization Platform

Building upon README and PRD foundations for detailed system behaviors and user interactions

ETVX Framework

ENTRY CRITERIA

- áœ... README completed with problem overview and technical approach
- áœ... PRD completed with business objectives, market analysis, and success metrics
- áœ... User personas defined with specific needs and pain points
- áœ... Core features and capabilities identified from business requirements

TASK

Define detailed functional requirements covering system behaviors, user interactions, AI/ML capabilities, integration interfaces, and acceptance criteria for all platform features.

VERIFICATION & VALIDATION

Verification Checklist: - [] Functional requirements align with PRD business objectives and user personas - [] System behaviors support all identified core features and capabilities - [] AI/ML requirements enable content generation and optimization goals - [] Integration requirements support multi-platform social media management

Validation Criteria: - [] FRD validated with social media marketing experts and platform users - [] AI/ML requirements validated with data science and machine learning teams - [] Integration requirements validated with social media platform specialists - [] User experience requirements validated with UX/UI design teams

EXIT CRITERIA

- Complete functional specifications ready for technical architecture design
- User interaction flows defined for all major platform features
- AI/ML processing requirements specified for implementation
- Integration interfaces defined for all supported social media platforms
- Foundation prepared for Non-Functional Requirements Document (NFRD) development

Reference to Previous Documents

This FRD builds upon **README** and **PRD** foundations: - **README Key Requirements** Detailed functional specifications with acceptance criteria - **PRD Core Features** Comprehensive system behaviors and user interaction flows - **PRD User Personas** User-centric functional requirements and experience design - **PRD Success Metrics** Measurable functional outcomes and performance criteria

1. Multi-Platform Social Media Management

1.1 Platform Integration and Authentication

FR-001: Multi-Platform Account Connection - Description: Users must be able to connect and manage multiple social media accounts across supported platforms - **Supported Platforms:** Facebook, Instagram, Twitter/X, LinkedIn, TikTok, YouTube, Pinterest - **User Story:** As a marketing manager, I want to connect all my brand's social media accounts in one place so that I can manage them efficiently from a unified dashboard - **Functional Behavior:** - System shall provide OAuth 2.0 authentication for each supported platform - System shall support multiple accounts per platform (e.g., multiple Facebook pages) - System shall automatically refresh authentication tokens before expiration - System shall notify users of authentication failures and provide re-authentication flows - **Acceptance Criteria:** - User can connect unlimited social media accounts across all supported platforms - Authentication process completes within 60 seconds per platform - System maintains 99.9% authentication token validity - Failed authentications trigger immediate user notifications

FR-002: Account Status Monitoring - Description: System must continuously monitor the health and status of connected social media accounts - **User Story:** As a social media manager, I want to be immediately notified if any of my connected accounts have issues so that I can resolve them quickly - **Functional Behavior:** - System shall perform health checks on all connected accounts every 15 minutes - System shall detect account suspensions, API limit violations, and permission changes - System shall provide real-time status indicators for each connected account - System shall automatically retry failed operations with exponential backoff - **Acceptance Criteria:** - Account status updates reflect within 15 minutes of platform changes - Users receive notifications within 5 minutes of account issues - System maintains 95% accuracy in account status detection - Automatic retry mechanisms resolve 80% of temporary failures

1.2 Unified Content Publishing

FR-003: Cross-Platform Content Publishing - Description: Users must be able to publish content simultaneously across multiple platforms with platform-specific optimizations - **User Story:** As a content creator, I want to publish the same message across all my social media platforms while ensuring each platform receives optimized content - **Functional Behavior:** - System shall allow users to compose content once and publish to selected platforms - System shall automatically optimize content format, length, and hashtags per platform - System shall handle platform-specific requirements (character limits, image sizes, video formats) - System shall provide preview functionality showing how content appears on each platform - **Acceptance Criteria:** - Content publishes successfully to 95% of selected platforms within 30 seconds - Platform-specific optimizations maintain content intent and brand voice - Preview accuracy matches actual published content 98% of the time - Failed publications trigger automatic retry with user notification

FR-004: Content Scheduling and Calendar Management - Description: System must provide comprehensive scheduling capabilities with visual calendar interface - **User Story:** As a marketing coordinator, I want to schedule posts in advance and visualize my content calendar so that I can maintain consistent posting schedules - **Functional Behavior:** - System shall provide drag-and-drop calendar interface for content scheduling - System shall support recurring post schedules (daily, weekly, monthly) - System shall allow bulk scheduling operations for multiple posts - System shall provide timezone-aware scheduling for global audiences - **Acceptance Criteria:** - Calendar interface loads within 3 seconds and supports 1000+ scheduled posts - Scheduled posts publish within 60 seconds of designated time - Timezone calculations are 100% accurate across all supported regions - Bulk operations complete within 5 minutes for up to 100 posts

2. AI-Powered Content Generation and Optimization

2.1 Intelligent Content Creation

FR-005: AI Text Content Generation - Description: System must generate high-quality, brand-consistent text content for social media posts - **User Story:** As a small business owner, I want AI to help me create engaging social media posts so that I can maintain active social presence without spending hours writing content - **Functional Behavior:** - System shall generate post captions, descriptions, and copy based on user prompts - System shall maintain brand voice consistency across all generated content - System shall incorporate trending hashtags and keywords relevant to the content - System shall provide multiple content variations for user selection - **Acceptance Criteria:** - Content generation completes within 15 seconds of user request - Generated content maintains 90% brand voice consistency as measured by brand voice analysis - 85% of generated content requires minimal or no editing before publication - System provides 3-5 content variations per generation request

FR-006: Visual Content Generation and Editing - Description: System must create and edit visual content including images, graphics, and basic video content - **User Story:** As a marketing manager, I want to automatically generate branded visuals for my posts so that I can maintain professional appearance without hiring designers - **Functional Behavior:** - System shall generate images based on text descriptions and brand guidelines - System shall create branded graphics using company logos, colors, and fonts - System shall resize and optimize images for different platform requirements - System shall provide basic video editing capabilities including text overlays and transitions - **Acceptance Criteria:** - Image generation completes within 30 seconds for standard graphics - Generated visuals comply with brand guidelines 95% of the time - Automatic resizing maintains image quality across all platform formats - Video editing operations complete within 2 minutes for clips under 60 seconds

2.2 Content Performance Optimization

FR-007: AI-Powered Content Optimization - Description: System must analyze and optimize content for maximum engagement and reach - **User Story:** As a social media strategist, I want AI to optimize my content for better performance so that I can improve engagement rates and reach - **Functional Behavior:** - System shall analyze historical performance data to recommend content improvements - System shall suggest optimal posting times based on audience behavior analysis - System shall recommend hashtags, keywords, and content formats for maximum reach - System shall provide A/B testing capabilities for content variations - **Acceptance Criteria:** - Content optimization recommendations improve engagement by 25% on average - Posting time recommendations result in 30% higher reach compared to random timing - Hashtag recommendations achieve 90% relevance to content and audience - A/B tests provide statistically significant results within 48 hours

FR-008: Real-Time Performance Monitoring and Adjustment - Description: System must monitor content performance in real-time and suggest adjustments - **User Story:** As a digital marketing manager, I want to see how my content is performing immediately after posting and get suggestions for improvement - **Functional Behavior:** - System shall track engagement metrics (likes, shares, comments, clicks) in real-time - System shall compare performance against historical benchmarks and industry standards - System shall suggest content modifications for underperforming posts - System shall automatically boost high-performing content when budget allows - **Acceptance Criteria:** - Performance data updates within 5 minutes of platform reporting - Benchmark comparisons provide accurate context for performance evaluation - Improvement suggestions result in 20% average performance increase when implemented - Automatic boosting decisions align with campaign objectives 90% of the time

3. Advanced Analytics and Intelligence

3.1 Comprehensive Performance Analytics

FR-009: Cross-Platform Analytics Dashboard - Description: System must provide unified analytics across all connected social media platforms - **User Story:** As a marketing director, I want to see consolidated performance metrics across all platforms so that I can understand overall social media ROI - **Functional Behavior:** - System shall aggregate metrics from all connected platforms into unified reports - System shall provide customizable dashboards with drag-and-drop widgets - System shall calculate cross-platform engagement rates, reach, and conversion metrics - System shall generate automated insights and trend analysis - **Acceptance Criteria:** - Dashboard loads within 5 seconds with data from up to 20 connected accounts - Metric aggregation accuracy maintains 99% consistency with platform native analytics - Custom dashboards support 50+ widget types and unlimited configurations - Automated insights identify 90% of significant performance trends

FR-010: ROI and Attribution Tracking - Description: System must track social media impact on business outcomes and revenue generation - **User Story:** As a CMO, I want to understand how social media activities contribute to sales and leads so that I can justify marketing spend and optimize budget allocation - **Functional Behavior:** - System shall integrate with Google Analytics, CRM systems, and e-commerce platforms - System shall track customer journey from social media interaction to conversion - System shall calculate social media attribution for sales, leads, and other business outcomes - System shall provide ROI calculations with configurable attribution models - **Acceptance Criteria:** - Attribution tracking captures 85% of social media-influenced conversions - ROI calculations update within 24 hours of conversion events - Integration accuracy maintains 95% consistency with source systems - Attribution models support first-touch, last-touch, and multi-touch scenarios

3.2 Competitive Intelligence and Market Analysis

FR-011: Automated Competitor Monitoring - Description: System must monitor competitor social media activities and provide comparative analysis - **User Story:** As a brand manager, I want to track my competitors' social media performance so that I can identify opportunities and benchmark my own performance - **Functional Behavior:** - System shall automatically discover and track competitor social media accounts - System shall analyze competitor content strategies, posting frequency, and engagement rates - System shall identify trending content and successful campaigns from competitors - System shall provide comparative performance reports and market positioning analysis - **Acceptance Criteria:** - Competitor discovery identifies 90% of relevant competitors in user's industry - Performance tracking updates within 24 hours of competitor posts - Trend identification accuracy achieves 80% precision in identifying viral content - Comparative reports provide actionable insights for strategy improvement

FR-012: Industry Trend Analysis and Forecasting - Description: System must identify and predict social media trends relevant to user's industry and audience - **User Story:** As a content strategist, I want to stay ahead of social media trends so that I can create timely, relevant content that resonates with my audience - **Functional Behavior:** - System shall analyze trending topics, hashtags, and content formats across platforms - System shall predict emerging trends using machine learning algorithms - System shall filter trends by industry, audience demographics, and geographic location - System shall provide trend-based content recommendations and templates - **Acceptance Criteria:** - Trend detection identifies emerging topics 48-72 hours before mainstream adoption - Prediction accuracy achieves 70% success rate for trends lasting 7+ days - Industry filtering provides 85% relevance to user's business sector - Trend-based recommendations result in 40% higher engagement rates

4. Social Listening and Engagement Management

4.1 Brand Monitoring and Sentiment Analysis

FR-013: Comprehensive Social Listening - Description: System must monitor brand mentions, keywords, and conversations across social media platforms - **User Story:** As a customer service manager, I want to track all mentions of my brand across social media so that I can respond quickly to customer feedback and issues - **Functional Behavior:** - System shall monitor brand mentions, product names, and custom keywords in real-time - System shall track mentions across all supported platforms plus web and news sources - System shall categorize mentions by sentiment (positive, negative, neutral) and priority - System shall provide alert notifications for high-priority mentions requiring immediate attention - **Acceptance Criteria:** - Mention detection captures 95% of brand references within 15 minutes of posting - Sentiment analysis achieves 85% accuracy compared to human evaluation - Priority categorization correctly identifies 90% of crisis-level mentions - Alert notifications reach users within 5 minutes of high-priority mention detection

FR-014: Crisis Detection and Management - Description: System must identify potential PR crises and provide response recommendations - **User Story:** As a PR manager, I want to be alerted immediately when negative sentiment spikes so that I can respond quickly to prevent reputation damage - **Functional Behavior:** - System shall detect unusual spikes in negative sentiment or mention volume - System shall analyze crisis severity and provide escalation recommendations - System shall suggest response strategies based on crisis type and historical data - System shall track crisis resolution progress and sentiment recovery - **Acceptance Criteria:** - Crisis detection identifies 90% of significant reputation threats within 30 minutes - Severity assessment accuracy achieves 85% alignment with expert evaluation - Response recommendations result in 60% faster crisis resolution time - Sentiment recovery tracking provides accurate progress measurement

4.2 Engagement Automation and Response Management

FR-015: Intelligent Response Automation - Description: System must provide automated and semi-automated response capabilities for social media interactions - **User Story:** As a community manager, I want AI to help me respond to common questions and comments so that I can maintain high response rates without working 24/7 - **Functional Behavior:** - System shall analyze incoming comments, messages, and mentions for response opportunities - System shall generate appropriate response suggestions based on context and brand voice - System shall automatically respond to common questions using predefined knowledge base - System shall escalate complex issues to human team members with context and recommendations - **Acceptance Criteria:** - Response suggestions maintain 90% brand voice consistency and appropriateness - Automated responses resolve 70% of common customer inquiries without human intervention - Escalation accuracy identifies 95% of interactions requiring human attention - Response time averages under 15 minutes for automated responses

FR-016: Unified Engagement Inbox - Description: System must consolidate all social media interactions into a single management interface - **User Story:** As a social media manager, I want all comments, messages, and mentions in one place so that I can efficiently manage customer interactions across platforms - **Functional Behavior:** - System shall aggregate all social media interactions into unified inbox interface - System shall provide filtering, sorting, and search capabilities for efficient management - System shall track response times and team member assignments - System shall maintain conversation history and context across multiple interactions - **Acceptance Criteria:** - Inbox aggregation includes 100% of interactions from connected platforms - Filtering and search operations complete within 2 seconds for up to 10,000 interactions - Response time tracking accuracy maintains 99% precision - Conversation context preservation supports unlimited interaction history

5. Team Collaboration and Workflow Management

5.1 Multi-User Access and Permissions

FR-017: Role-Based Access Control - Description: System must support multiple team members with appropriate access levels and permissions - **User Story:** As a marketing director, I want to control what team members can access and modify so that I can maintain security while enabling collaboration - **Functional Behavior:** - System shall provide predefined roles (Admin, Manager, Creator, Viewer) with customizable permissions - System shall support custom role creation with granular permission settings - System shall track all user actions and changes with comprehensive audit logs - System shall provide single sign-on (SSO) integration with enterprise identity providers - **Acceptance Criteria:** - Role assignment takes effect within 60 seconds of configuration - Permission enforcement prevents 100% of unauthorized access attempts - Audit logs capture all user actions with timestamp and user identification - SSO integration supports major enterprise identity providers (Azure AD, Okta, Google)

FR-018: Collaboration and Approval Workflows - Description: System must support content approval processes and team collaboration features - **User Story:** As a brand manager, I want to review and approve content before it's published so that I can ensure brand consistency and quality - **Functional Behavior:** - System shall provide configurable approval workflows with multiple approval levels - System shall support content commenting, revision requests, and collaborative editing - System shall send notifications to approvers and track approval status - System shall maintain version history for all content with rollback capabilities - **Acceptance Criteria:** - Approval workflows support up to 5 approval levels with parallel and sequential options - Notification delivery achieves 99% success rate within 2 minutes of workflow events - Version history maintains unlimited revisions with complete change tracking - Collaborative editing supports real-time updates for up to 10 simultaneous users

6. Integration and API Capabilities

6.1 Third-Party System Integration

FR-019: CRM and Marketing Automation Integration - Description: System must integrate with popular CRM and marketing automation platforms - **User Story:** As a marketing operations manager, I want social media data to sync with our CRM so that we can track customer interactions across all touchpoints - **Functional Behavior:** - System shall integrate with major CRM platforms (Salesforce, HubSpot, Pipedrive) - System shall sync social media interactions with customer records - System shall support lead generation and qualification from social media activities - System shall provide webhook notifications for real-time

data synchronization - **Acceptance Criteria:** - CRM integration maintains 99% data synchronization accuracy - Lead qualification achieves 85% accuracy in identifying sales-ready prospects - Webhook delivery success rate exceeds 95% with automatic retry mechanisms - Data sync latency averages under 5 minutes for standard operations

FR-020: Analytics and Business Intelligence Integration - Description: System must integrate with analytics platforms and support data export for business intelligence - **User Story:** As a data analyst, I want to combine social media data with other business metrics so that I can create comprehensive performance reports - **Functional Behavior:** - System shall integrate with Google Analytics, Adobe Analytics, and other major platforms - System shall provide data export capabilities in multiple formats (CSV, JSON, API) - System shall support real-time data streaming to business intelligence platforms - System shall maintain data schema consistency for reliable reporting - **Acceptance Criteria:** - Analytics integration captures 100% of trackable social media events - Data export operations complete within 10 minutes for datasets up to 1M records - Real-time streaming maintains 99% uptime with sub-second latency - Schema consistency prevents data integration errors in 95% of use cases

6.2 API and Developer Platform

FR-021: Comprehensive REST API - Description: System must provide complete API access for third-party integrations and custom applications - **User Story:** As a developer, I want to access all platform functionality through APIs so that I can build custom integrations and applications - **Functional Behavior:** - System shall provide RESTful APIs for all platform features and data access - System shall support API authentication using OAuth 2.0 and API keys - System shall implement rate limiting and usage monitoring for API consumers - System shall provide comprehensive API documentation with interactive examples - **Acceptance Criteria:** - API coverage includes 100% of platform functionality available through user interface - Authentication success rate exceeds 99% with proper credential management - Rate limiting prevents abuse while supporting legitimate high-volume usage - API documentation maintains 95% accuracy with automated testing validation

FR-022: Webhook and Event System - Description: System must provide real-time event notifications through webhooks and event streaming - **User Story:** As an integration developer, I want to receive real-time notifications of platform events so that I can build responsive integrations - **Functional Behavior:** - System shall provide webhook notifications for all significant platform events - System shall support event filtering and subscription management - System shall guarantee webhook delivery with retry mechanisms and dead letter queues - System shall provide event streaming capabilities for high-volume consumers - **Acceptance Criteria:** - Webhook delivery success rate exceeds 95% on first attempt, 99% with retries - Event filtering reduces unnecessary notifications by 80% for targeted subscriptions - Retry mechanisms resolve 90% of temporary delivery failures - Event streaming supports throughput of 10,000+ events per second

This comprehensive FRD provides detailed functional specifications for all major platform capabilities, ensuring clear requirements for development teams while maintaining alignment with business objectives and user needs defined in the README and PRD documents. # Non-Functional Requirements Document (NFRD) ## Social Media Management Agent - AI-Powered Intelligent Social Media Management and Content Optimization Platform

Building upon README, PRD, and FRD foundations for performance, scalability, security, and operational requirements

ETVX Framework

ENTRY CRITERIA

- âœ… README completed with problem overview and technical approach
- âœ… PRD completed with business objectives, market analysis, and success metrics
- âœ… FRD completed with 22 detailed functional requirements across 6 modules
- âœ… User personas and success metrics defined for performance benchmarking

TASK

Define comprehensive non-functional requirements covering performance, scalability, reliability, security, usability, compliance, and operational requirements for the Social Media Management Agent platform.

VERIFICATION & VALIDATION

Verification Checklist: - [] Performance requirements align with user experience expectations and business objectives - [] Scalability requirements support projected user growth and platform expansion - [] Security requirements address social media data protection and enterprise compliance - [] Operational requirements enable 24/7 platform availability and support

Validation Criteria: - [] NFRD validated with infrastructure and DevOps teams for technical feasibility - [] Performance requirements validated with user experience and product teams - [] Security requirements validated with information security and compliance teams - [] Scalability requirements validated with business growth projections and capacity planning

EXIT CRITERIA

- âœ… Complete non-functional specifications ready for architecture design
- âœ… Performance benchmarks defined for system optimization and monitoring
- âœ… Security framework established for enterprise-grade data protection
- âœ… Operational requirements specified for production deployment and maintenance
- âœ… Foundation prepared for Architecture Diagram (AD) development

Reference to Previous Documents

This NFRD builds upon **README**, **PRD**, and **FRD** foundations: - **README Expected Outcomes** â†’ Quantified performance and efficiency targets - **PRD Success Metrics** â†’ Measurable non-functional performance indicators - **FRD Functional Requirements** â†’ Performance constraints for system behaviors - **PRD Technical Requirements** â†’ Infrastructure and scalability specifications

1. Performance Requirements

1.1 Response Time and Latency

NFR-001: User Interface Response Time - Requirement: All user interface interactions must complete within specified time limits - **Specifications:** - Dashboard loading: â‰¤3 seconds for initial load, â‰¤1 second for subsequent navigation - Content creation interface: â‰¤2 seconds for form rendering and data population - Analytics report generation: â‰¤5 seconds for standard reports, â‰¤15 seconds for complex queries - Search operations: â‰¤1 second for basic search, â‰¤3 seconds for advanced filtering - **Measurement:** 95th percentile response times measured across all user interactions - **Business Impact:** Ensures optimal user experience and productivity for social media management tasks

NFR-002: AI Content Generation Performance - Requirement: AI-powered content generation must meet real-time user expectations - **Specifications:** - Text content generation: â‰¤15 seconds for social media posts and captions - Image generation: â‰¤30 seconds for branded graphics and visual content - Video editing operations: â‰¤2 minutes for clips under 60 seconds - Content optimization recommendations: â‰¤5 seconds for performance analysis - **Measurement:** Average processing time measured across all AI operations - **Business Impact:** Enables efficient content creation workflows without productivity delays

NFR-003: Social Media API Response Time - Requirement: Social media platform integrations must maintain optimal performance - **Specifications:** - Content publishing: â‰¤30 seconds from user action to platform publication - Data synchronization: â‰¤5 minutes for social media metrics and engagement data - Real-time monitoring: â‰¤15 minutes for brand mention detection and alerts - Account authentication: â‰¤60 seconds for OAuth flows and token refresh - **Measurement:** End-to-end processing time including platform API latency - **Business Impact:** Ensures timely content publication and real-time social media monitoring

1.2 Throughput and Capacity

NFR-004: Content Processing Throughput - Requirement: System must handle high-volume content operations efficiently - **Specifications:** - Concurrent content publishing: 1,000+ posts per minute across all platforms - Bulk scheduling operations: 500+ posts processed per minute - Image processing: 100+ images optimized per minute - Analytics data processing: 10M+ social media interactions per hour - **Measurement:** Peak throughput sustained for 15-minute intervals during high-traffic periods - **Business Impact:** Supports large enterprise clients and high-volume content campaigns

NFR-005: User Concurrency - Requirement: Platform must support simultaneous users without performance degradation - **Specifications:** - Concurrent active users: 10,000+ users simultaneously using the platform - Peak concurrent sessions: 25,000+ during high-traffic periods - Database connections: 5,000+ concurrent

database connections - API requests: 100,000+ API calls per minute during peak usage - **Measurement:** Load testing with simulated user behavior patterns - **Business Impact:** Enables platform scalability for enterprise clients and global usage

2. Scalability Requirements

2.1 Horizontal and Vertical Scaling

NFR-006: Auto-Scaling Infrastructure - Requirement: System must automatically scale resources based on demand - **Specifications:** - Horizontal scaling: Auto-scale from 10 to 1,000+ application instances - Database scaling: Read replicas auto-scale based on query load - CDN scaling: Global content delivery with automatic edge location provisioning - Queue processing: Auto-scale background job processors from 5 to 500+ workers - **Measurement:** Resource utilization metrics and scaling response times - **Business Impact:** Maintains performance during traffic spikes while optimizing infrastructure costs

NFR-007: Data Storage Scalability - Requirement: Data storage must scale to accommodate growing user base and content volume - **Specifications:** - Database storage: Petabyte-scale data storage with automatic partitioning - Media storage: Unlimited file storage with intelligent tiering and archiving - Analytics data: Time-series data retention for 7 years with compression - Backup storage: 3x redundancy with geographic distribution - **Measurement:** Storage capacity utilization and data retrieval performance - **Business Impact:** Supports long-term business growth and comprehensive historical analytics

2.2 Geographic and Multi-Region Scaling

NFR-008: Global Infrastructure Distribution - Requirement: Platform must provide optimal performance for global user base - **Specifications:** - Geographic regions: 6+ AWS/Azure regions with active-active deployment - CDN coverage: 200+ edge locations for content delivery optimization - Database replication: Cross-region replication with <5 second lag - Failover capability: Automatic regional failover within 60 seconds - **Measurement:** Regional response times and failover recovery times - **Business Impact:** Ensures consistent global user experience and business continuity

3. Reliability and Availability

3.1 System Uptime and Availability

NFR-009: Platform Availability - Requirement: System must maintain high availability for business-critical operations - **Specifications:** - Overall uptime: 99.9% availability (8.77 hours downtime per year maximum) - Planned maintenance windows: â‰ƒ4 hours monthly, scheduled during low-usage periods - Unplanned downtime: â‰ƒ2 hours per incident, â‰ƒ6 hours total per year - Service degradation: Graceful degradation with core functions maintained - **Measurement:** Uptime monitoring with third-party validation and SLA reporting - **Business Impact:** Ensures reliable social media management operations for enterprise clients

NFR-010: Disaster Recovery and Business Continuity - Requirement: System must recover quickly from disasters and maintain business operations - **Specifications:** - Recovery Time Objective (RTO): â‰ƒ4 hours for full service restoration - Recovery Point Objective (RPO): â‰ƒ1 hour maximum data loss - Backup frequency: Real-time replication with hourly backup snapshots - Geographic redundancy: Multi-region deployment with automatic failover - **Measurement:** Disaster recovery testing and failover exercise results - **Business Impact:** Protects business operations and customer data during major incidents

3.2 Error Handling and Fault Tolerance

NFR-011: System Resilience - Requirement: System must handle failures gracefully and maintain service quality - **Specifications:** - Circuit breaker patterns: Automatic isolation of failing services - Retry mechanisms: Exponential backoff with jitter for transient failures - Graceful degradation: Core functionality maintained during partial system failures - Error recovery: Automatic recovery from 90% of transient errors - **Measurement:** Error rates, recovery times, and service availability during failures - **Business Impact:** Maintains user productivity and platform reliability during system stress

4. Security Requirements

4.1 Data Protection and Encryption

NFR-012: Data Encryption and Security - Requirement: All data must be protected with enterprise-grade encryption - **Specifications:** - Data at rest: AES-256 encryption for all stored data including databases and files - Data in transit: TLS 1.3 encryption for all network communications - Key management: Hardware Security Module (HSM) for encryption key storage - Social media tokens: Encrypted storage with automatic rotation and expiration - **Measurement:** Security audits and penetration testing validation - **Business Impact:** Protects sensitive social media data and maintains customer trust

NFR-013: Access Control and Authentication - Requirement: System must implement comprehensive access control mechanisms - **Specifications:** - Multi-factor authentication: Required for all user accounts with enterprise SSO support - Role-based access control: Granular permissions with principle of least privilege - Session management: Automatic session timeout and concurrent session limits - API security: OAuth 2.0, API key management, and rate limiting - **Measurement:** Security compliance audits and access control effectiveness testing - **Business Impact:** Prevents unauthorized access and maintains data security compliance

4.2 Compliance and Privacy

NFR-014: Data Privacy Compliance - Requirement: System must comply with global data privacy regulations - **Specifications:** - GDPR compliance: Data subject rights, consent management, and data portability - CCPA compliance: Consumer privacy rights and data disclosure requirements - Data residency: Geographic data storage controls for regulatory compliance - Privacy by design: Minimal data collection with automatic data retention policies - **Measurement:** Compliance audits and privacy impact assessments - **Business Impact:** Enables global operations while maintaining regulatory compliance

NFR-015: Security Monitoring and Incident Response - Requirement: System must provide comprehensive security monitoring and response capabilities - **Specifications:** - Security monitoring: 24/7 SOC with automated threat detection - Incident response: â‰ƒ15 minutes for security incident acknowledgment - Audit logging: Comprehensive audit trails with tamper-proof storage - Vulnerability management: Monthly security scans with 48-hour critical patch deployment - **Measurement:** Security incident response times and vulnerability remediation metrics - **Business Impact:** Proactive security threat management and rapid incident resolution

5. Usability and User Experience

5.1 User Interface and Accessibility

NFR-016: User Experience Standards - Requirement: Platform must provide intuitive and efficient user experience - **Specifications:** - Learning curve: New users achieve basic proficiency within 2 hours - Task completion: 95% of common tasks completed without documentation - Error prevention: Proactive validation and user guidance to prevent errors - Mobile responsiveness: Full functionality on tablets and mobile devices - **Measurement:** User experience testing and task completion analytics - **Business Impact:** Reduces training costs and improves user adoption rates

NFR-017: Accessibility Compliance - Requirement: Platform must be accessible to users with disabilities - **Specifications:** - WCAG 2.1 AA compliance: Full accessibility standard compliance - Screen reader support: Compatible with major screen reading software - Keyboard navigation: Complete functionality accessible via keyboard - Color contrast: Minimum 4.5:1 contrast ratio for all text elements - **Measurement:** Accessibility audits and assistive technology testing - **Business Impact:** Ensures inclusive access and compliance with accessibility regulations

5.2 Internationalization and Localization

NFR-018: Global User Support - Requirement: Platform must support international users and markets - **Specifications:** - Language support: 20+ languages with professional translation quality - Cultural adaptation: Region-specific content formats and social media practices - Timezone handling: Accurate timezone conversion and scheduling across all regions - Currency support: Multi-currency pricing and analytics reporting - **Measurement:** Localization quality assessments and international user feedback - **Business Impact:** Enables global market expansion and international customer acquisition

6. Operational Requirements

6.1 Monitoring and Observability

NFR-019: System Monitoring and Alerting - Requirement: Comprehensive monitoring must provide visibility into system health and performance - **Specifications:** - Application monitoring: Real-time performance metrics with 1-minute granularity - Infrastructure monitoring: Server, database, and network

NFR-020: Logging and Audit Trails - Requirement: System must maintain comprehensive logs for troubleshooting and compliance - **Specifications:** - Application logs: Structured logging with correlation IDs and contextual information - Audit logs: Immutable audit trails for all user actions and system changes - Log retention: 7-year retention for compliance with automatic archiving - Log analysis: Real-time log analysis with automated anomaly detection - **Measurement:** Log completeness, searchability, and analysis effectiveness - **Business Impact:** Enables effective troubleshooting and regulatory compliance

NFR-021: Deployment and Release Management - Requirement: System must support efficient and reliable deployment processes - **Specifications:** - Deployment frequency: Daily deployments with zero-downtime deployment capability - Rollback capability: Automated rollback within 5 minutes if deployment issues detected - Environment parity: Identical staging and production environments for testing - Feature flags: Gradual feature rollout with A/B testing capabilities - **Measurement:** Deployment success rates, rollback frequency, and deployment duration - **Business Impact:** Enables rapid feature delivery while maintaining system stability

7. Integration and API Requirements

NFR-023: API Service Level Agreements - Requirement: APIs must meet stringent performance and reliability standards - **Specifications:** - API response time: 95th percentile response time ≤500ms for standard operations - API availability: 99.95% uptime for all public APIs - Rate limiting: Fair usage policies with burst capacity for legitimate high-volume usage - API versioning: Backward compatibility maintained for minimum 2 years - **Measurement:** API performance monitoring and SLA compliance reporting - **Business Impact:** Enables reliable third-party integrations and developer ecosystem growth

8. Compliance and Regulatory Requirements

NFR-025: Security and Compliance Certifications - Requirement: Platform must achieve and maintain industry-standard security certifications - **Specifications:** - SOC 2 Type II: Annual compliance audit with clean opinion - ISO 27001: Information security management system certification - PCI DSS: Payment card industry compliance for payment processing - HIPAA: Healthcare data protection compliance for healthcare clients - **Measurement:** Annual compliance audits and certification maintenance - **Business Impact:** Enables enterprise sales and builds customer trust

This comprehensive NFRD establishes the performance, security, and operational foundation required for an enterprise-grade social media management platform that can scale globally while maintaining high availability, security, and user experience standards. # Architecture Diagram (AD) ## Social Media Management Agent - AI-Powered Intelligent Social Media Management and Content Optimization Platform

ETVX Framework

- â€¦ README completed with problem overview and technical approach
- â€¦ PRD completed with business objectives, market analysis, and success metrics
- â€¦ FRD completed with 22 detailed functional requirements across 6 modules
- â€¦ NFRD completed with 26 non-functional requirements covering performance, security, and scalability

Design comprehensive system architecture including microservices design, AI/ML pipeline, data architecture, integration patterns, security framework, and cloud native deployment strategy.

Verification Checklist: - [] Architecture supports all functional requirements from FRD - [] Design meets all non-functional requirements from NFRD - [] Microservices design enables scalability and maintainability - [] Security architecture addresses enterprise compliance requirements

EXIT CRITERIA

- Complete system architecture ready for detailed design
- Technology stack decisions documented and justified
- Integration patterns defined for all external systems
- Security and compliance framework established
- Foundation prepared for High Level Design (HLD) development

This AD builds upon **README**, **PRD**, **FRD**, and **NFRD** foundations: - **FRD Functional Requirements** â†’ Architectural components and service boundaries - **NFRD Performance Requirements** â†’ Scalability and performance architecture decisions - **NFRD Security Requirements** â†’ Security architecture and compliance framework - **PRD Technical Requirements** â†’ Technology stack and integration architecture

1. System Architecture Overview

1.1 High-Level Architecture Pattern

Cloud-Native Microservices Architecture

Architecture Principles: - **Microservices Design:** Domain-driven service boundaries with independent deployment - **Event-Driven Architecture:** Asynchronous communication using message queues and event streams - **API-First Design:** RESTful APIs with OpenAPI specifications and GraphQL for complex queries - **Cloud-Native:** Containerized services with Kubernetes orchestration and auto-scaling - **Security by Design:** Zero-trust architecture with comprehensive security controls

Service Mesh Architecture: - **API Gateway:** Kong/AWS API Gateway for request routing, authentication, and rate limiting - **Service Discovery:** Consul/Kubernetes DNS for dynamic service registration and discovery - **Load Balancing:** NGINX/AWS ALB with health checks and circuit breaker patterns - **Inter-Service Communication:** gRPC for internal services, REST for external APIs

2. Microservices Design

User Management Service - Responsibilities: Authentication, authorization, user profiles, team management - **Technology Stack:** Node.js/Express, PostgreSQL, Redis, Auth0/Keycloak - **API Endpoints:** /auth, /users, /teams, /permissions - **Data Models:** Users, Teams, Roles, Permissions, Sessions - **Integration:** SSO providers, audit logging, notification service

Social Media Integration Service - Responsibilities: Platform API management, content publishing, data synchronization - **Technology Stack:** Python/FastAPI, Redis, PostgreSQL, Celery - **API Endpoints:** /platforms, /accounts, /publish, /sync - **Data Models:** Platform Accounts, API Tokens, Publishing Queue, Sync Status - **Integration:** Facebook Graph API, Twitter API, LinkedIn API, Instagram API

2.2 AI/ML Services

Intelligence and Optimization Service - Responsibilities: Performance optimization, trend analysis, predictive analytics - **Technology Stack:** Python/FastAPI, Scikit-learn, TensorFlow, Apache Kafka - **Models:** Recommendation systems, time series forecasting, sentiment analysis - **API Endpoints:** /optimize, /predict, /trends, /sentiment - **Infrastructure:** Kubernetes with auto-scaling, real-time stream processing - **Integration:** Analytics service, social listening service, content service

2.3 Supporting Services

Workflow Engine Service - Responsibilities: Business process automation, approval workflows, scheduling - **Technology Stack:** Java/Spring Boot, PostgreSQL, Apache Airflow - **Features:** Visual workflow designer, conditional logic, parallel processing - **API Endpoints:** /workflows, /tasks, /schedules, /approvals - **Integration:** Content service, user management, notification service

3. Data Architecture

Primary Databases:

Specialized Databases:

```
InfluxDB (Time Series)
"â€œâ€" Social media metrics and KPIs
"â€œâ€" Performance monitoring data
"â€œâ€" User engagement analytics
"â€œâ€" System performance metrics
```

- Elasticsearch (Search and Analytics)
 - Content search and discovery
 - Social media mention indexing
 - Log aggregation and analysis
 - Full-text search capabilities

Social Engagement â†’ Event Processing â†’ Marketing Automation â†’ Campaigns
â†’

Analytics Data → ETL Pipeline → Data Warehouse → BI Tools (Tableau/PowerBI)
→
Executive Dashboards → Business Insights

6. Security Architecture

6.1 Zero-Trust Security Model

Authentication and Authorization:

User Request → API Gateway → JWT Validation → Service Authorization
→ Rate Limiting → WAF → Service Mesh → Microservice

Security Layers: - **Perimeter Security:** Web Application Firewall (WAF) with DDoS protection - **Network Security:** VPC isolation with security groups and network ACLs - **Application Security:** OAuth 2.0, JWT tokens, API key management - **Data Security:** Encryption at rest (AES-256) and in transit (TLS 1.3) - **Infrastructure Security:** Container scanning, vulnerability management

6.2 Compliance and Data Protection

Data Privacy Architecture:

Data Collection Privacy Controls Consent Management Data Processing
Audit Logging Compliance Reporting

Compliance Framework: - **GDPR Compliance:** Data subject rights, consent management, data portability - **SOC 2 Type II:** Security controls, access management, incident response - **ISO 27001:** Information security management system - **PCI DSS:** Payment card data protection (if applicable)

7. Deployment Architecture

7.1 Cloud-Native Deployment

Kubernetes Architecture:

[illegible]

Container Strategy: - **Base Images:** Distrosless containers for security and minimal attack surface - **Multi-Stage Builds:** Optimized container sizes with build-time dependencies removed - **Image Scanning:** Automated vulnerability scanning with Trivy/Snyk - **Registry Management:** Private container registry with image signing and verification

7.2 Multi-Region Deployment

Global Architecture:

[illegible]

Regional Distribution: - **Primary Region:** US-East with full service deployment - **Secondary Regions:** EU-West, Asia-Pacific with read replicas and caching - **Edge Locations:** Global CDN with 200+ edge locations for content delivery - **Disaster Recovery:** Cross-region backup and failover capabilities

7.3 DevOps and CI/CD Pipeline

Deployment Pipeline:

Code Commit â†’ CI Pipeline â†’ Security Scan â†’ Build â†’ Test â†’ Deploy
â†’
Automated Testing â†’ Staging â†’ Production â†’ Monitoring

Infrastructure as Code: - **Terraform:** Infrastructure provisioning and management - **Helm Charts:** Kubernetes application deployment and configuration - **GitOps:** ArgoCD for continuous deployment and configuration management - **Policy as Code:** Open Policy Agent (OPA) for security and compliance policies

8. Monitoring and Observability

8.1 Comprehensive Monitoring Stack

Observability Architecture:

[illegible]

Monitoring Components:

- **Metrics Collection:** Prometheus with custom metrics and business KPIs
- **Visualization:** Grafana dashboards for technical and business metrics
- **Distributed Tracing:** Jaeger for request tracing across microservices
- **Log Management:** ELK Stack (Elasticsearch, Logstash, Kibana) for centralized logging
- **Alerting:** AlertManager with PagerDuty integration for incident response

8.2 Performance and Health Monitoring

Health Check Strategy:

Load Balancer → Health Endpoints → Service Health → Database Health
→
Kubernetes Probes → Auto-healing → Service Recovery

Performance Monitoring: - **Application Performance:** Response times, throughput, error rates - **Infrastructure Performance:** CPU, memory, disk, network utilization - **Business Metrics:** User engagement, content performance, revenue impact - **AI/ML Monitoring:** Model performance, prediction accuracy, data drift

detection

This comprehensive architecture provides a scalable, secure, and maintainable foundation for the Social Media Management Agent platform, supporting all functional and non-functional requirements while enabling future growth and innovation. # High Level Design (HLD) ## Social Media Management Agent - AI-Powered Intelligent Social Media Management and Content Optimization Platform

Building upon README, PRD, FRD, NFRD, and AD foundations for detailed component specifications

ETVX Framework

ENTRY CRITERIA

- âœ… README completed with problem overview and technical approach
- âœ… PRD completed with business objectives, market analysis, and success metrics
- âœ… FRD completed with 22 detailed functional requirements across 6 modules
- âœ… NFRD completed with 26 non-functional requirements covering performance, security, and scalability
- âœ… AD completed with microservices architecture and cloud-native deployment strategy

TASK

Define detailed component specifications, API designs, data models, processing workflows, and AI/ML architectures for all system components.

VERIFICATION & VALIDATION

Verification Checklist: - [] Component specifications align with microservices architecture from AD - [] API designs support all functional requirements with performance targets - [] Data models accommodate social media management and analytics requirements - [] AI/ML workflows meet content generation and optimization objectives

Validation Criteria: - [] HLD validated with social media platform experts and AI/ML specialists - [] API designs validated with frontend and integration teams - [] Data models validated with database architects and performance teams - [] AI/ML workflows validated with data science and machine learning teams

EXIT CRITERIA

- âœ… Complete component specifications ready for implementation
- âœ… API designs with detailed interface definitions
- âœ… Data models supporting all functional requirements
- âœ… Foundation prepared for Low Level Design (LLD) development

1. Core Service Components

1.1 User Management Service

Component Architecture:

```
class UserManagementService:
    def __init__(self):
        self.auth_provider = AuthProvider()
        self.user_repository = UserRepository()
        self.team_manager = TeamManager()
        self.permission_engine = PermissionEngine()
        self.audit_logger = AuditLogger()

    async def authenticate_user(self, credentials: AuthCredentials) -> AuthResult:
        # Multi-factor authentication with SSO support

    async def manage_permissions(self, user_id: str, permissions: List[Permission]) -> PermissionResult:
        # Role-based access control with granular permissions
```

API Specifications:

```
# User Management API
/api/v1/auth:
  post:
    summary: Authenticate user
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            properties:
              email: { type: string, format: email }
              password: { type: string, minLength: 8 }
              mfa_token: { type: string }
    responses:
      200:
        description: Authentication successful
        content:
          application/json:
            schema:
              type: object
              properties:
                access_token: { type: string }
                refresh_token: { type: string }
                expires_in: { type: integer }
                user_profile: { $ref: '#/components/schemas/UserProfile' }

/api/v1/users/{userId}/permissions:
  get:
    summary: Get user permissions
    parameters:
      - name: userId
        in: path
        required: true
        schema: { type: string, format: uuid }
    responses:
      200:
        description: User permissions retrieved
        content:
          application/json:
            schema:
              type: object
              properties:
                permissions:
                  type: array
                  items: { $ref: '#/components/schemas/Permission' }
```

Data Models:

```
-- User Management Schema
CREATE TABLE users (
```

```

    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    profile_image_url VARCHAR(500),
    timezone VARCHAR(50) DEFAULT 'UTC',
    language VARCHAR(10) DEFAULT 'en',
    status user_status_enum DEFAULT 'active',
    last_login TIMESTAMPT WITH TIME ZONE,
    created_at TIMESTAMPT WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMPT WITH TIME ZONE DEFAULT NOW()
);

CREATE TABLE teams (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(255) NOT NULL,
    description TEXT,
    owner_id UUID NOT NULL REFERENCES users(id),
    settings JSONB DEFAULT '{}',
    created_at TIMESTAMPT WITH TIME ZONE DEFAULT NOW()
);

CREATE TABLE user_teams (
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    team_id UUID REFERENCES teams(id) ON DELETE CASCADE,
    role team_role_enum NOT NULL,
    permissions JSONB DEFAULT '{}',
    joined_at TIMESTAMPT WITH TIME ZONE DEFAULT NOW(),
    PRIMARY KEY (user_id, team_id)
);

CREATE TYPE user_status_enum AS ENUM ('active', 'inactive', 'suspended');
CREATE TYPE team_role_enum AS ENUM ('owner', 'admin', 'manager', 'creator', 'viewer');

```

1.2 Content Management Service

Component Architecture:

```

class ContentManagementService:
    def __init__(self):
        self.content_repository = ContentRepository()
        self.media_processor = MediaProcessor()
        self.workflow_engine = WorkflowEngine()
        self.brand_validator = BrandValidator()
        self.scheduler = ContentScheduler()

    async def create_content(self, content_data: ContentData) -> Content:
        # Content creation with brand validation and workflow routing

    async def schedule_content(self, content_id: str, schedule: ScheduleData) -> ScheduleResult:
        # Intelligent scheduling with optimization recommendations

```

Content Processing Workflow:

```

graph TD
    A[Content Creation] --> B[Brand Validation]
    B --> C[Media Processing]
    C --> D[Workflow Routing]
    D --> E[Approval Process]
    E --> F[Scheduling Queue]
    F --> G[Platform Publishing]
    G --> H[Performance Tracking]

```

API Specifications:

```

/api/v1/content:
  post:
    summary: Create new content
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            properties:
              title: { type: string, maxLength: 255 }
              body: { type: string, maxLength: 10000 }
              media_ids:
                type: array
                items: { type: string, format: uuid }
              platforms:
                type: array
                items: { type: string, enum: [facebook, instagram, twitter, linkedin] }
              tags:
                type: array
                items: { type: string }
    responses:
      201:
        description: Content created successfully
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Content'

/api/v1/content/{contentId}/schedule:
  post:
    summary: Schedule content for publishing
    parameters:
      - name: contentId
        in: path
        required: true
        schema: { type: string, format: uuid }
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            properties:
              publish_at: { type: string, format: date-time }
              platforms:
                type: array
                items: { type: string }
              optimization_enabled: { type: boolean, default: true }

```

1.3 Social Media Integration Service

Component Architecture:

```
class SocialMediaIntegrationService:
    def __init__(self):
        self.platform_connectors = {
            'facebook': FacebookConnector(),
            'instagram': InstagramConnector(),
            'twitter': TwitterConnector(),
            'linkedin': LinkedInConnector(),
            'tiktok': TikTokConnector(),
            'youtube': YouTubeConnector()
        }
        self.rate_limiter = RateLimiter()
        self.circuit_breaker = CircuitBreaker()
        self.retry_handler = RetryHandler()

    async def publish_content(self, content: Content, platforms: List[str]) -> PublishResult:
        # Multi-platform publishing with error handling and retry logic

    async def sync_metrics(self, account_id: str, platform: str) -> MetricsResult:
        # Real-time metrics synchronization with rate limiting
```

Platform Connector Interface:

```
from abc import ABC, abstractmethod

class PlatformConnector(ABC):
    @abstractmethod
    async def authenticate(self, credentials: dict) -> AuthResult:
        pass

    @abstractmethod
    async def publish_post(self, content: Content) -> PublishResult:
        pass

    @abstractmethod
    async def get_metrics(self, post_id: str) -> MetricsData:
        pass

    @abstractmethod
    async def get_account_info(self) -> AccountInfo:
        pass

class FacebookConnector(PlatformConnector):
    def __init__(self):
        self.graph_api = FacebookGraphAPI()
        self.webhook_handler = FacebookWebhookHandler()

    async def publish_post(self, content: Content) -> PublishResult:
        # Facebook-specific publishing logic with Graph API
        try:
            response = await self.graph_api.post(
                endpoint=f"/{content.account_id}/feed",
                data={
                    "message": content.text,
                    "link": content.link,
                    "published": content.published
                }
            )
            return PublishResult(
                success=True,
                platform_post_id=response["id"],
                published_at=datetime.utcnow()
            )
        except FacebookAPIException as e:
            return PublishResult(
                success=False,
                error=str(e),
                retry_after=e.retry_after if hasattr(e, 'retry_after') else None
            )
```

2. AI/ML Service Architecture

2.1 Content Generation Service

AI Model Pipeline:

```
class ContentGenerationService:
    def __init__(self):
        self.text_generator = TextGenerator()
        self.image_generator = ImageGenerator()
        self.brand_adapter = BrandAdapter()
        self.quality_validator = QualityValidator()
        self.performance_predictor = PerformancePredictor()

    async def generate_text_content(self, prompt: ContentPrompt) -> GeneratedContent:
        # Multi-model text generation with brand consistency

    async def generate_visual_content(self, description: str, brand_guidelines: BrandGuidelines) -> GeneratedImage:
        # AI-powered image generation with brand compliance
```

Text Generation Pipeline:

```
class TextGenerator:
    def __init__(self):
        self.primary_model = GPT4Model()
        self.fallback_model = GPT35TurboModel()
        self.brand_fine_tuned_models = {}
        self.prompt_optimizer = PromptOptimizer()

    async def generate_content(self, prompt: ContentPrompt, brand_context: BrandContext) -> List[GeneratedText]:
        # Enhanced prompt with brand context
        enhanced_prompt = await self.prompt_optimizer.enhance_prompt(
            base_prompt=prompt.text,
            brand_voice=brand_context.voice_guidelines,
            target_audience=prompt.target_audience,
            platform_requirements=prompt.platform_specs
        )

        # Primary model generation
        try:
            primary_results = await self.primary_model.generate(
                prompt=enhanced_prompt,
                max_tokens=prompt.max_length,
                temperature=0.7,
                n=3 # Generate 3 variations
            )
```



```

        # Brand consistency validation
        validated_results = []
        for result in primary_results:
            brand_score = await self.brand_adapter.validate_consistency(
                content=result.text,
                brand_guidelines=brand_context.guidelines
            )
            if brand_score >= 0.8: # 80% brand consistency threshold
                validated_results.append(result)

        return validated_results

    except ModelException as e:
        # Fallback to secondary model
        return await self.fallback_model.generate(enhanced_prompt)

```

Image Generation Pipeline:

```

class ImageGenerator:
    def __init__(self):
        self.dalle_model = DALLEModel()
        self.stable_diffusion = StableDiffusionModel()
        self.brand_style_adapter = BrandStyleAdapter()
        self.image_optimizer = ImageOptimizer()

    async def generate_branded_image(self, description: str, brand_guidelines: BrandGuidelines) -> GeneratedImage:
        # Brand-adapted prompt generation
        branded_prompt = await self.brand_style_adapter.adapt_prompt(
            base_description=description,
            color_palette=brand_guidelines.colors,
            style_preferences=brand_guidelines.visual_style,
            logo_placement=brand_guidelines.logo_usage
        )

        # Multi-model generation with quality selection
        dalle_result = await self.dalle_model.generate(branded_prompt)
        sd_result = await self.stable_diffusion.generate(branded_prompt)

        # Quality assessment and selection
        best_image = await self.select_best_image([dalle_result, sd_result])

        # Platform-specific optimization
        optimized_images = await self.image_optimizer.optimize_for_platforms(
            image=best_image,
            platforms=["facebook", "instagram", "twitter", "linkedin"]
        )

        return GeneratedImage(
            original=best_image,
            platform_optimized=optimized_images,
            brand_compliance_score=await self.validate_brand_compliance(best_image, brand_guidelines)
        )

```

2.2 Performance Optimization Service

Optimization Engine:

```

class PerformanceOptimizationService:
    def __init__(self):
        self.engagement_predictor = EngagementPredictor()
        self.timing_optimizer = TimingOptimizer()
        self.hashtag_recommender = HashtagRecommender()
        self.ab_test_manager = ABTestManager()
        self.performance_analyzer = PerformanceAnalyzer()

    async def optimize_content(self, content: Content, historical_data: HistoricalData) -> OptimizationRecommendations:
        # Comprehensive content optimization with ML predictions

```

Engagement Prediction Model:

```

class EngagementPredictor:
    def __init__(self):
        self.model = self.load_trained_model()
        self.feature_extractor = FeatureExtractor()
        self.scaler = StandardScaler()

    async def predict_engagement(self, content: Content, context: PostingContext) -> EngagementPrediction:
        # Extract features from content and context
        features = await self.feature_extractor.extract_features(
            text=content.text,
            media=content.media,
            posting_time=context.posting_time,
            audience_size=context.audience_size,
            historical_performance=context.historical_data
        )

        # Normalize features
        normalized_features = self.scaler.transform(features)

        # Generate predictions
        predictions = self.model.predict(normalized_features)

        return EngagementPrediction(
            expected_likes=predictions[0],
            expected_shares=predictions[1],
            expected_comments=predictions[2],
            engagement_rate=predictions[3],
            confidence_interval=predictions[4]
        )

```

2.3 Social Listening Service

Real-Time Monitoring Pipeline:

```

class SocialListeningService:
    def __init__(self):
        self.stream_processor = StreamProcessor()
        self.sentiment_analyzer = SentimentAnalyzer()
        self.entity_extractor = EntityExtractor()
        self.trend_detector = TrendDetector()
        self.crisis_detector = CrisisDetector()

    async def process_social_stream(self, stream_data: SocialStreamData) -> ProcessedMention:
        # Real-time social media mention processing and analysis

```

Sentiment Analysis Pipeline:

```

class SentimentAnalyzer:

```

```

def __init__(self):
    self.transformer_model = BERTSentimentModel()
    self.lexicon_analyzer = VaderSentiment()
    self.ensemble_weights = [0.7, 0.3] # Transformer: 70%, Lexicon: 30%

async def analyze_sentiment(self, text: str, context: MentionContext) -> SentimentResult:
    # Ensemble sentiment analysis with context awareness
    transformer_result = await self.transformer_model.predict(text)
    lexicon_result = self.lexicon_analyzer.polarity_scores(text)

    # Weighted ensemble prediction
    final_sentiment = (
        transformer_result.sentiment * self.ensemble_weights[0] +
        lexicon_result['compound'] * self.ensemble_weights[1]
    )

    # Context-aware adjustment
    if context.is_customer_service:
        final_sentiment = self.adjust_for_customer_service_context(final_sentiment)

    return SentimentResult(
        sentiment_score=final_sentiment,
        sentiment_label=self.score_to_label(final_sentiment),
        confidence=transformer_result.confidence,
        emotions=transformer_result.emotions
    )

```

3. Data Processing Workflows

3.1 Real-Time Analytics Pipeline

Stream Processing Architecture:

```

class RealTimeAnalyticsProcessor:
    def __init__(self):
        self.kafka_consumer = KafkaConsumer()
        self.stream_processor = StreamProcessor()
        self.metrics_aggregator = MetricsAggregator()
        self.alert_manager = AlertManager()

    async def process_engagement_events(self, event_stream: EventStream) -> ProcessedMetrics:
        # Real-time engagement event processing and aggregation

```

Event Processing Flow:

```

graph LR
    A[Social Media APIs] --> B[Kafka Topics]
    B --> C[Stream Processing]
    C --> D[Metrics Aggregation]
    D --> E[Real-time Dashboard]
    D --> F[Alert Generation]
    D --> G[Data Storage]

```

3.2 Batch Analytics Processing

ETL Pipeline:

```

class BatchAnalyticsProcessor:
    def __init__(self):
        self.data_extractor = DataExtractor()
        self.data_transformer = DataTransformer()
        self.data_loader = DataLoader()
        self.report_generator = ReportGenerator()

    async def process_daily_analytics(self, date: datetime) -> AnalyticsReport:
        # Daily batch processing for comprehensive analytics

        # Extract data from multiple sources
        social_media_data = await self.data_extractor.extract_social_media_metrics(date)
        user_engagement_data = await self.data_extractor.extract_user_engagement(date)
        content_performance_data = await self.data_extractor.extract_content_performance(date)

        # Transform and enrich data
        transformed_data = await self.data_transformer.transform_and_enrich(
            social_media_data,
            user_engagement_data,
            content_performance_data
        )

        # Load into analytics database
        await self.data_loader.load_to_analytics_db(transformed_data)

        # Generate insights and reports
        insights = await self.generate_insights(transformed_data)
        report = await self.report_generator.generate_daily_report(insights)

        return report

```

4. Integration Layer Design

4.1 API Gateway Architecture

Gateway Configuration:

```

class APIGateway:
    def __init__(self):
        self.rate_limiter = RateLimiter()
        self.auth_validator = AuthValidator()
        self.request_router = RequestRouter()
        self.response_transformer = ResponseTransformer()
        self.circuit_breaker = CircuitBreaker()

    async def handle_request(self, request: HTTPRequest) -> HTTPResponse:
        # Comprehensive request handling with security and performance controls

```

Rate Limiting Strategy:

```

class RateLimiter:
    def __init__(self):
        self.redis_client = RedisClient()
        self.rate_limit_configs = {
            'free_tier': {'requests_per_minute': 100, 'burst': 20},
            'pro_tier': {'requests_per_minute': 1000, 'burst': 200},
            'enterprise_tier': {'requests_per_minute': 10000, 'burst': 2000}
        }

```

```

async def check_rate_limit(self, user_id: str, tier: str) -> RateLimitResult:
    config = self.rate_limit_configs[tier]
    current_usage = await self.redis_client.get_usage(user_id)

    if current_usage >= config['requests_per_minute']:
        return RateLimitResult(
            allowed=False,
            retry_after=60,
            remaining=0
        )

    await self.redis_client.increment_usage(user_id)
    return RateLimitResult(
        allowed=True,
        remaining=config['requests_per_minute'] - current_usage - 1
    )

```

4.2 Webhook Management System

Webhook Processing:

```

class WebhookManager:
    def __init__(self):
        self.webhook_registry = WebhookRegistry()
        self.event_processor = EventProcessor()
        self.delivery_manager = DeliveryManager()
        self.retry_handler = RetryHandler()

    async def process_webhook_event(self, event: WebhookEvent) -> ProcessingResult:
        # Reliable webhook event processing with retry logic

        # Validate webhook signature
        if not await self.validate_signature(event):
            return ProcessingResult(success=False, error="Invalid signature")

        # Process event
        processing_result = await self.event_processor.process(event)

        # Deliver to registered endpoints
        delivery_results = await self.delivery_manager.deliver_to_subscribers(
            event=event,
            processing_result=processing_result
        )

        return ProcessingResult(
            success=True,
            delivery_results=delivery_results
        )

```

5. Security Implementation

5.1 Authentication and Authorization

JWT Token Management:

```

class JWTTokenManager:
    def __init__(self):
        self.private_key = self.load_private_key()
        self.public_key = self.load_public_key()
        self.token_blacklist = TokenBlacklist()

    async def generate_token(self, user: User, permissions: List[Permission]) -> JWTToken:
        payload = {
            'user_id': user.id,
            'email': user.email,
            'permissions': [p.name for p in permissions],
            'iat': datetime.utcnow(),
            'exp': datetime.utcnow() + timedelta(hours=24),
            'iss': 'social-media-agent',
            'aud': 'api-clients'
        }

        token = jwt.encode(payload, self.private_key, algorithm='RS256')

        # Store token metadata for tracking
        await self.store_token_metadata(user.id, token)

        return JWTToken(
            access_token=token,
            token_type='Bearer',
            expires_in=86400, # 24 hours
            refresh_token=await self.generate_refresh_token(user.id)
        )

```

5.2 Data Encryption and Protection

Encryption Service:

```

class EncryptionService:
    def __init__(self):
        self.encryption_key = self.load_encryption_key()
        self.cipher_suite = Fernet(self.encryption_key)

    async def encrypt_sensitive_data(self, data: str) -> EncryptedData:
        encrypted_bytes = self.cipher_suite.encrypt(data.encode())
        return EncryptedData(
            encrypted_value=base64.b64encode(encrypted_bytes).decode(),
            encryption_algorithm='AES-256',
            key_version=self.get_key_version()
        )

    async def decrypt_sensitive_data(self, encrypted_data: EncryptedData) -> str:
        encrypted_bytes = base64.b64decode(encrypted_data.encrypted_value)
        decrypted_bytes = self.cipher_suite.decrypt(encrypted_bytes)
        return decrypted_bytes.decode()

```

This comprehensive HLD provides detailed component specifications, API designs, and processing workflows that enable the development team to implement a robust, scalable, and secure social media management platform. # Low Level Design (LLD) ## Social Media Management Agent - AI-Powered Intelligent Social Media Management and Content Optimization Platform

Building upon README, PRD, FRD, NFRD, AD, and HLD foundations for implementation-ready specifications

ETVX Framework

ENTRY CRITERIA

- â€¦ README completed with problem overview and technical approach
- â€¦ PRD completed with business objectives, market analysis, and success metrics
- â€¦ FRD completed with 22 detailed functional requirements across 6 modules
- â€¦ NFRD completed with 26 non-functional requirements covering performance, security, and scalability
- â€¦ AD completed with microservices architecture and cloud-native deployment strategy
- â€¦ HLD completed with component specifications and API designs

TASK

Develop implementation-ready detailed class structures, database schemas, API implementations, algorithm specifications, configuration files, and deployment scripts for all system components.

VERIFICATION & VALIDATION

Verification Checklist: - [] Class structures implement all HLD component specifications - [] Database schemas support all data models and performance requirements - [] API implementations include validation, error handling, and security - [] Algorithm specifications provide step-by-step implementation guidance

Validation Criteria: - [] LLD validated with senior developers and technical leads - [] Database schemas validated with DBA and performance teams - [] API implementations validated with security and integration teams - [] Configuration files validated with DevOps and infrastructure teams

EXIT CRITERIA

- â€¦ Complete implementation-ready class structures and database schemas
- â€¦ API implementations with comprehensive error handling and validation
- â€¦ Algorithm specifications for all AI/ML processing components
- â€¦ Configuration files and deployment scripts for production deployment
- â€¦ Foundation prepared for Pseudocode development

1. Core Service Implementation

1.1 User Management Service

Class Structure:

```
# src/services/user_management/models.py
from sqlalchemy import Column, String, DateTime, Enum
from sqlalchemy.dialects.postgresql import UUID, JSONB
from sqlalchemy.ext.declarative import declarative_base
import uuid
from datetime import datetime
from enum import Enum as PyEnum

Base = declarative_base()

class UserStatus(PyEnum):
    ACTIVE = "active"
    INACTIVE = "inactive"
    SUSPENDED = "suspended"

class User(Base):
    __tablename__ = "users"

    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid.uuid4)
    email = Column(String(255), unique=True, nullable=False, index=True)
    password_hash = Column(String(255), nullable=False)
    first_name = Column(String(100), nullable=False)
    last_name = Column(String(100), nullable=False)
    profile_image_url = Column(String(500))
    timezone = Column(String(50), default='UTC')
    language = Column(String(10), default='en')
    status = Column(Enum(UserStatus), default=UserStatus.ACTIVE)
    last_login = Column(DateTime(timezone=True))
    preferences = Column(JSONB, default={})
    created_at = Column(DateTime(timezone=True), default=datetime.utcnow)
    updated_at = Column(DateTime(timezone=True), default=datetime.utcnow)

class UserManagementService:
    def __init__(self, db: Session, redis_client):
        self.db = db
        self.redis = redis_client
        self.pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")

    async def authenticate_user(self, email: str, password: str) -> Optional[User]:
        user = self.db.query(User).filter(User.email == email).first()
        if not user or not self.verify_password(password, user.password_hash):
            return None
        user.last_login = datetime.utcnow()
        self.db.commit()
        return user

    def verify_password(self, plain_password: str, hashed_password: str) -> bool:
        return self.pwd_context.verify(plain_password, hashed_password)
```

1.2 Content Management Service

Implementation:

```
# src/services/content_management/service.py
from typing import List, Dict, Optional
from datetime import datetime

class ContentManagementService:
    def __init__(self, db: Session, media_service, ai_service):
        self.db = db
        self.media_service = media_service
        self.ai_service = ai_service

    async def create_content(self, content_data: dict, author_id: str) -> Content:
        """Create new content with validation and processing"""
        await self._validate_content_data(content_data)

        processed_media = []
        if content_data.get("media_files"):
            processed_media = await self.media_service.process_media_files(
                content_data["media_files"]
            )

        content = Content(
            title=content_data["title"],
            body=content_data["body"],
```

```

        media_urls=processed_media,
        hashtags=content_data.get("hashtags", []),
        platforms=content_data.get("platforms", []),
        author_id=author_id,
        team_id=content_data.get("team_id"),
        metadata=content_data.get("metadata", {})
    )

    self.db.add(content)
    self.db.commit()
    self.db.refresh(content)
    return content

async def schedule_content(self, content_id: str, schedule_data: dict) -> List[ContentSchedule]:
    """Schedule content for publishing across platforms"""
    content = self.db.query(Content).filter(Content.id == content_id).first()
    if not content:
        raise HTTPException(status_code=404, detail="Content not found")

    schedules = []
    for platform in schedule_data["platforms"]:
        optimal_time = schedule_data["scheduled_at"]
        if schedule_data.get("optimize_timing", False):
            optimal_time = await self.ai_service.get_optimal_posting_time(
                platform=platform,
                content=content,
                target_time=schedule_data["scheduled_at"]
            )

        schedule = ContentSchedule(
            content_id=content_id,
            platform=platform,
            scheduled_at=optimal_time,
            status="scheduled"
        )

        self.db.add(schedule)
        schedules.append(schedule)

    self.db.commit()
    return schedules

```

1.3 Social Media Integration

Platform Connector:

```

# src/services/social_media/connectors/facebook.py
import aiohttp
from typing import Dict

class FacebookConnector:
    def __init__(self, app_id: str, app_secret: str):
        self.app_id = app_id
        self.app_secret = app_secret
        self.base_url = "https://graph.facebook.com/v18.0"

    async def publish_post(self, page_id: str, access_token: str, content: Dict) -> Dict:
        """Publish content to Facebook page"""
        url = f"{self.base_url}/{page_id}/feed"

        post_data = {
            "access_token": access_token,
            "message": content["text"]
        }

        if content.get("link"):
            post_data["link"] = content["link"]

        if content.get("media_urls"):
            post_data["url"] = content["media_urls"][0]

        async with aiohttp.ClientSession() as session:
            async with session.post(url, data=post_data) as response:
                if response.status == 200:
                    result = await response.json()
                    return {
                        "success": True,
                        "platform_post_id": result["id"],
                        "published_at": datetime.utcnow().isoformat()
                    }
                else:
                    error_data = await response.json()
                    return {
                        "success": False,
                        "error": error_data.get("error", {}).get("message", "Publishing failed")
                    }

```

2. Database Schema

PostgreSQL Implementation:

```

-- Core tables
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    profile_image_url VARCHAR(500),
    timezone VARCHAR(50) DEFAULT 'UTC',
    language VARCHAR(10) DEFAULT 'en',
    status user_status_enum DEFAULT 'active',
    preferences JSONB DEFAULT '{}',
    last_login TIMESTAMPTZ WITH TIME ZONE,
    created_at TIMESTAMPTZ WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMPTZ WITH TIME ZONE DEFAULT NOW()
);

CREATE TABLE content (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    title VARCHAR(255) NOT NULL,
    body TEXT,
    media_urls TEXT[],
    hashtags TEXT[],
    platforms TEXT[],
    status content_status_enum DEFAULT 'draft',
    author_id UUID NOT NULL REFERENCES users(id),
    team_id UUID REFERENCES teams(id),

```

```

        metadata JSONB DEFAULT '{}',
        created_at TIMESTAMPT WITH TIME ZONE DEFAULT NOW(),
        updated_at TIMESTAMPT WITH TIME ZONE DEFAULT NOW()
    );

CREATE TABLE content_schedules (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    content_id UUID NOT NULL REFERENCES content(id) ON DELETE CASCADE,
    platform VARCHAR(50) NOT NULL,
    scheduled_at TIMESTAMPT WITH TIME ZONE NOT NULL,
    published_at TIMESTAMPT WITH TIME ZONE,
    platform_post_id VARCHAR(255),
    status schedule_status_enum DEFAULT 'scheduled',
    error_message TEXT,
    created_at TIMESTAMPT WITH TIME ZONE DEFAULT NOW()
);

-- Enums
CREATE TYPE user_status_enum AS ENUM ('active', 'inactive', 'suspended');
CREATE TYPE content_status_enum AS ENUM ('draft', 'pending_approval', 'approved', 'scheduled', 'published', 'failed');
CREATE TYPE schedule_status_enum AS ENUM ('scheduled', 'publishing', 'published', 'failed', 'cancelled');

-- Indexes
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_content_author_created ON content(author_id, created_at);
CREATE INDEX idx_schedules_scheduled_at ON content_schedules(scheduled_at);

```

3. API Implementation

FastAPI Service:

```

# src/api/main.py
from fastapi import FastAPI, Depends, HTTPException
from pydantic import BaseModel, Field
from typing import List, Optional

app = FastAPI(title="Social Media Management Agent API", version="1.0.0")

class ContentCreateRequest(BaseModel):
    title: str = Field(..., min_length=1, max_length=255)
    body: str = Field(..., min_length=1, max_length=10000)
    platforms: List[str] = Field(..., min_items=1)
    hashtags: Optional[List[str]] = Field(default=[])

class ContentResponse(BaseModel):
    id: str
    title: str
    body: str
    platforms: List[str]
    status: str
    created_at: str

@app.post("/api/v1/content", response_model=ContentResponse)
async def create_content(
    request: ContentCreateRequest,
    current_user: str = Depends(get_current_user),
    content_service: ContentManagementService = Depends(get_content_service)
):
    """Create new social media content"""
    content = await content_service.create_content(
        content_data=request.dict(),
        author_id=current_user
    )

    return ContentResponse(
        id=str(content.id),
        title=content.title,
        body=content.body,
        platforms=content.platforms,
        status=content.status.value,
        created_at=content.created_at.isoformat()
    )

@app.get("/api/v1/content", response_model=List[ContentResponse])
async def list_content(
    skip: int = 0,
    limit: int = 20,
    current_user: str = Depends(get_current_user),
    content_service: ContentManagementService = Depends(get_content_service)
):
    """List user's content with pagination"""
    content_list = await content_service.list_user_content(
        user_id=current_user,
        skip=skip,
        limit=limit
    )

    return [
        ContentResponse(
            id=str(content.id),
            title=content.title,
            body=content.body,
            platforms=content.platforms,
            status=content.status.value,
            created_at=content.created_at.isoformat()
        )
        for content in content_list
    ]

```

4. Configuration Files

Docker Compose:

```

# docker-compose.yml
version: '3.8'

services:
    api:
        build: .
        ports:
            - "8000:8000"
        environment:
            - DATABASE_URL=postgres://postgres:password@postgres:5432/social_media_db
            - REDIS_URL=redis://redis:6379
        depends_on:
            - postgres
            - redis

```

```
postgres:
  image: postgres:14
  environment:
    - POSTGRES_DB=social_media_db
    - POSTGRES_USER=postgres
    - POSTGRES_PASSWORD=password
  volumes:
    - postgres_data:/var/lib/postgresql/data
  ports:
    - "5432:5432"

redis:
  image: redis:7-alpine
  ports:
    - "6379:6379"
  volumes:
    - redis_data:/data

volumes:
  postgres_data:
  redis_data:
```

Environment Configuration:

```
# config/production.yaml
database:
  url: ${DATABASE_URL}
  pool_size: 20
  max_overflow: 30

redis:
  url: ${REDIS_URL}
  max_connections: 50

social_media:
  facebook:
    app_id: ${FACEBOOK_APP_ID}
    app_secret: ${FACEBOOK_APP_SECRET}
  twitter:
    api_key: ${TWITTER_API_KEY}
    api_secret: ${TWITTER_API_SECRET}

ai_services:
  openai:
    api_key: ${OPENAI_API_KEY}
    model: "gpt-4"

security:
  jwt_secret: ${JWT_SECRET}
  jwt_algorithm: "HS256"
  access_token_expire_minutes: 30

logging:
  level: INFO
  format: json
```

This LLD provides implementation-ready specifications with detailed class structures, database schemas, API implementations, and configuration files for direct development implementation. # Pseudocode ## Social Media Management Agent - AI-Powered Intelligent Social Media Management and Content Optimization Platform

Building upon README, PRD, FRD, NFRD, AD, HLD, and LLD foundations for executable implementation algorithms

ETVX Framework

ENTRY CRITERIA

- âœ… README completed with problem overview and technical approach
- âœ… PRD completed with business objectives, market analysis, and success metrics
- âœ… FRD completed with 22 detailed functional requirements across 6 modules
- âœ… NFRD completed with 26 non-functional requirements covering performance, security, and scalability
- âœ… AD completed with microservices architecture and cloud-native deployment strategy
- âœ… HLD completed with component specifications and API designs
- âœ… LLD completed with implementation-ready class structures and database schemas

TASK

Develop executable pseudocode algorithms for all core system components including content management, AI-powered generation, social media integration, analytics processing, and performance optimization.

VERIFICATION & VALIDATION

Verification Checklist: - [] Pseudocode algorithms align with LLD class implementations - [] Processing workflows meet performance requirements (<2s UI response) - [] AI/ML algorithms implement content generation and optimization features - [] Integration algorithms support all social media platform connectors

Validation Criteria: - [] Pseudocode validated with social media platform experts and AI/ML specialists - [] Algorithms validated with frontend and integration development teams - [] Performance algorithms validated with scalability and optimization teams - [] Security algorithms validated with information security teams

EXIT CRITERIA

- âœ… Complete executable pseudocode for all system components
- âœ… AI/ML processing algorithms for content generation and optimization
- âœ… Social media integration workflows for multi-platform publishing
- âœ… Analytics and performance monitoring algorithms
- âœ… Implementation-ready foundation for development teams

1. Content Management Algorithms

1.1 Content Creation and Validation

```
ALGORITHM: CreateContent
INPUT: content_data (object), author_id (string)
OUTPUT: Content (object)

BEGIN CreateContent
  // Input validation
  IF NOT ValidateContentData(content_data) THEN
    THROW ValidationException("Invalid content data")
  END IF
```

```

// Platform-specific validation
FOR each platform IN content_data.platforms DO
  validation_result = ValidatePlatformRequirements(content_data, platform)
  IF NOT validation_result.valid THEN
    THROW ValidationException("Content violates " + platform + " requirements")
  END IF
END FOR

// Process media files
processed_media = []
IF content_data.media_files IS NOT EMPTY THEN
  FOR each media_file IN content_data.media_files DO
    processed_file = ProcessMediaFile(media_file)
    processed_media.append(processed_file.url)
  END FOR
END IF

// Create content record
content = Content{
  id: GenerateUUID(),
  title: content_data.title,
  body: content_data.body,
  media_urls: processed_media,
  hashtags: content_data.hashtags,
  platforms: content_data.platforms,
  status: "draft",
  author_id: author_id,
  created_at: getCurrentTime()
}

// Save to database
SaveContentToDatabase(content)
LogContentActivity(content.id, "created", author_id)

RETURN content
END CreateContent

```

1.2 Content Scheduling Algorithm

```

ALGORITHM: ScheduleContent
INPUT: content_id (string), schedule_data (object)
OUTPUT: List<ContentSchedule>

BEGIN ScheduleContent
  content = GetContentById(content_id)
  IF content IS NULL THEN
    THROW NotFoundException("Content not found")
  END IF

  schedules = []

  FOR each platform IN schedule_data.platforms DO
    // Optimize timing if requested
    optimal_time = schedule_data.scheduled_at
    IF schedule_data.optimize_timing THEN
      optimal_time = OptimizePostingTime(platform, content, schedule_data.scheduled_at)
    END IF

    // Create schedule
    schedule = ContentSchedule{
      id: GenerateUUID(),
      content_id: content_id,
      platform: platform,
      scheduled_at: optimal_time,
      status: "scheduled"
    }

    SaveScheduleToDatabase(schedule)
    AddToSchedulingQueue(schedule)
    schedules.append(schedule)
  END FOR

  RETURN schedules
END ScheduleContent

```

2. AI Content Generation Algorithms

2.1 Text Content Generation

```

ALGORITHM: GenerateTextContent
INPUT: prompt (ContentPrompt), brand_context (BrandContext)
OUTPUT: List<GeneratedContent>

BEGIN GenerateTextContent
  // Enhance prompt with brand context
  enhanced_prompt = EnhancePromptWithBrandContext(
    base_prompt: prompt.text,
    brand_voice: brand_context.voice_guidelines,
    target_audience: prompt.target_audience,
    platform_specs: prompt.platform_requirements
  )

  generated_variations = []

  // Primary AI model generation
  TRY
    primary_results = CallPrimaryAIModel(
      prompt: enhanced_prompt,
      max_tokens: prompt.max_length,
      temperature: 0.7,
      variations: 3
    )

    // Validate brand consistency
    FOR each result IN primary_results DO
      brand_score = ValidateBrandConsistency(result.text, brand_context.guidelines)

      IF brand_score >= 0.8 THEN
        generated_content = GeneratedContent{
          text: result.text,
          brand_consistency_score: brand_score,
          quality_score: AssessContentQuality(result.text),
          suggested_hashtags: GenerateHashtags(result.text, prompt.platforms),
          engagement_prediction: PredictEngagement(result.text, brand_context)
        }
        generated_variations.append(generated_content)
      END IF
    END FOR
  END TRY

```



```

        END FOR

    CATCH AIModelException e
        // Fallback to secondary model
        fallback_results = CallFallbackAIModel(enhanced_prompt)
        FOR each result IN fallback_results DO
            generated_content = GeneratedContent{
                text: result.text,
                brand_consistency_score: 0.7,
                quality_score: AssessContentQuality(result.text),
                suggested_hashtags: GenerateHashtags(result.text, prompt.platforms)
            }
            generated_variations.append(generated_content)
        END FOR
    END TRY

    // Sort by combined score and return top 3
    SortByScore(generated_variations)
    RETURN Take(generated_variations, 3)
END GenerateTextContent

```

2.2 Visual Content Generation

ALGORITHM: GenerateVisualContent
 INPUT: description (string), brand_guidelines (BrandGuidelines), platforms (List<string>)
 OUTPUT: GeneratedVisualContent

```

BEGIN GenerateVisualContent
    // Adapt prompt for brand consistency
    branded_prompt = AdaptPromptForBrand(
        base_description: description,
        color_palette: brand_guidelines.colors,
        style_preferences: brand_guidelines.visual_style
    )

    // Generate using multiple AI models
    generation_results = []

    TRY
        dalle_result = CallDALLEModel(branded_prompt, "1024x1024", "hd")
        generation_results.append(dalle_result)
    CATCH AIModelException e
        LogWarning("DALL-E generation failed: " + e.message)
    END TRY

    TRY
        sd_result = CallStableDiffusionModel(branded_prompt, 1024, 1024)
        generation_results.append(sd_result)
    CATCH AIModelException e
        LogWarning("Stable Diffusion generation failed: " + e.message)
    END TRY

    // Select best image
    best_image = SelectBestImage(generation_results, brand_guidelines)

    // Generate platform-optimized versions
    platform_optimized = {}
    FOR each platform IN platforms DO
        platform_specs = GetPlatformImageSpecs(platform)
        optimized_image = OptimizeImageForPlatform(best_image, platform_specs)
        platform_optimized[platform] = optimized_image
    END FOR

    RETURN GeneratedVisualContent{
        original_image: best_image,
        platform_optimized: platform_optimized,
        brand_compliance_score: ValidateVisualBrandCompliance(best_image, brand_guidelines)
    }
END GenerateVisualContent

```

3. Social Media Integration Algorithms

3.1 Multi-Platform Publishing

ALGORITHM: PublishToMultiplePlatforms
 INPUT: content (Content), schedules (List<ContentSchedule>)
 OUTPUT: List<PublishResult>

```

BEGIN PublishToMultiplePlatforms
    publish_results = []

    FOR each schedule IN schedules DO
        TRY
            // Check timing
            IF getCurrentTime() < schedule.scheduled_at THEN
                CONTINUE
            END IF

            // Update status
            schedule.status = "publishing"
            UpdateScheduleInDatabase(schedule)

            // Get platform connector and account
            connector = GetPlatformConnector(schedule.platform)
            account = GetSocialAccount(schedule.platform, content.author_id)

            // Check rate limits
            rate_limit_check = CheckRateLimit(schedule.platform, account.id)
            IF NOT rate_limit_check.allowed THEN
                RescheduleForLater(schedule, rate_limit_check.retry_after)
                CONTINUE
            END IF

            // Adapt content for platform
            platform_content = AdaptContentForPlatform(content, schedule.platform, account)

            // Publish content
            publish_result = connector.PublishPost(
                account_id: account.platform_account_id,
                access_token: account.access_token,
                content: platform_content
            )

            // Update schedule based on result
            IF publish_result.success THEN
                schedule.status = "published"
                schedule.published_at = getCurrentTime()
            END IF
        END TRY
    END FOR

    RETURN publish_results
END PublishToMultiplePlatforms

```

```

        schedule.platform_post_id = publish_result.platform_post_id
        StartMetricsCollection(schedule.id, publish_result.platform_post_id)
    ELSE
        schedule.status = "failed"
        schedule.error_message = publish_result.error

        IF IsRetryableError(publish_result.error) AND schedule.retry_count < 3 THEN
            schedule.retry_count += 1
            schedule.scheduled_at = getCurrentTime() + CalculateRetryDelay(schedule.retry_count)
            schedule.status = "scheduled"
        END IF
    END IF

    UpdateScheduleInDatabase(schedule)

    result = PublishResult{
        schedule_id: schedule.id,
        success: publish_result.success,
        platform_post_id: publish_result.platform_post_id,
        error: publish_result.error
    }
    publish_results.append(result)

    CATCH Exception e
        schedule.status = "failed"
        schedule.error_message = "Unexpected error: " + e.message
        UpdateScheduleInDatabase(schedule)

        publish_results.append(PublishResult{
            schedule_id: schedule.id,
            success: false,
            error: e.message
        })
    END TRY
END FOR

RETURN publish_results
END PublishToMultiplePlatforms

```

4. Analytics and Performance Monitoring

4.1 Real-Time Metrics Collection

```

ALGORITHM: CollectRealTimeMetrics
INPUT: platform_posts (List<PlatformPost>)
OUTPUT: MetricsCollectionResult

BEGIN CollectRealTimeMetrics
    collection_results = []

    FOR each post IN platform_posts DO
        TRY
            connector = GetPlatformConnector(post.platform)
            account = GetSocialAccount(post.platform, post.author_id)

            // Fetch metrics from platform
            metrics_data = connector.GetPostMetrics(
                post_id: post.platform_post_id,
                access_token: account.access_token
            )

            IF metrics_data.success THEN
                // Calculate engagement rate
                engagement_rate = CalculateEngagementRate(
                    likes: metrics_data.likes,
                    comments: metrics_data.comments,
                    shares: metrics_data.shares,
                    impressions: metrics_data.impressions
                )

                // Store metrics
                metrics_record = PostMetrics{
                    content_schedule_id: post.schedule_id,
                    platform: post.platform,
                    platform_post_id: post.platform_post_id,
                    likes: metrics_data.likes,
                    shares: metrics_data.shares,
                    comments: metrics_data.comments,
                    impressions: metrics_data.impressions,
                    engagement_rate: engagement_rate,
                    collected_at: getCurrentTime()
                }

                SaveMetricsToDatabase(metrics_record)

                collection_results.append(MetricsResult{
                    post_id: post.id,
                    success: true,
                    metrics: metrics_record
                })
            ELSE
                collection_results.append(MetricsResult{
                    post_id: post.id,
                    success: false,
                    error: metrics_data.error
                })
            END IF

        CATCH Exception e
            collection_results.append(MetricsResult{
                post_id: post.id,
                success: false,
                error: e.message
            })
        END TRY
    END FOR

    RETURN MetricsCollectionResult{
        results: collection_results,
        success_rate: CalculateSuccessRate(collection_results)
    }
END CollectRealTimeMetrics

```

5. Performance Optimization Algorithms

5.1 Content Performance Optimization

```

ALGORITHM: OptimizeContentPerformance
INPUT: content (Content), historical_data (HistoricalData)
OUTPUT: OptimizationRecommendations

BEGIN OptimizeContentPerformance
    recommendations = OptimizationRecommendations{}

    // Analyze current performance
    current_performance = AnalyzeContentPerformance(content, historical_data)

    // Text optimization
    text_analysis = AnalyzeTextContent(content.body)
    IF text_analysis.readability_score < 70 THEN
        recommendations.text_improvements.append("Improve readability")
    END IF

    // Hashtag optimization
    hashtag_analysis = AnalyzeHashtagPerformance(content.hashtags, historical_data)
    better_hashtags = RecommendBetterHashtags(content.body, content.platforms)
    recommendations.hashtag_improvements = better_hashtags

    // Timing optimization
    optimal_times = CalculateOptimalPostingTimes(content.author_id, content.platforms)
    recommendations.timing_suggestions = optimal_times

    // Platform-specific recommendations
    FOR each platform IN content.platforms DO
        platform_recommendations = GeneratePlatformRecommendations(content, platform)
        recommendations.platform_specific[platform] = platform_recommendations
    END FOR

    RETURN recommendations
END OptimizeContentPerformance

```

This comprehensive pseudocode provides executable algorithms for all core system components, enabling direct implementation of the Social Media Management Agent platform while maintaining alignment with all previous requirements and architectural decisions.