Social Media Content Aggregator

Rakshith Rao S V[1], Sheik Mohammed Ali M[2], Vimalesh D[3], Dr. D M Vijayalakshmi[4]
[1,2,3,4]*Bachelor Of Engineering in Computer Science and Engineering, Adhiyamaan College of Engineering*
Dr.M.G.R Nagar, Hosur- 635130
Anna University: Chennai 600 025

***Abstract*—SMCA is an advanced full-stack, modular social media content aggregation platform engineered to unify, enrich, and manage large-scale unstructured data from Reddit and Twitter, solving critical challenges in trend analysis, research, and digital monitoring. Developed with FastAPI, Python, MongoDB, and state-of-the-art NLP libraries, the solution offers secure RESTful APIs and a responsive HTML user interface supporting role-based users from analysts to researchers. SMCA streamlines collection, cleansing, semantic enrichment, and intelligent deduplication of social media data through an automated pipeline: authenticated API-based scraping (PRAW for Reddit, Apify for Twitter), preprocessing and normalization of raw text, transformer-powered entity and intent extraction (spaCy), and smart deduplication using compound key logic in MongoDB. Its semantic tag-based search system empowers users to go beyond basic keyword retrieval, surfacing contextual matches based on extracted entities and intent, while robust endpoints support advanced queries, bulk exports, and scheduled tasks. Key features include secure API key authentication, entity-driven dashboard analytics, configurable data export (CSV/JSON), extensible modular design for future platform and analytics integration, comprehensive error handling, and scalable architecture supporting millions of records. Automating all stages of aggregation, SMCA achieves over 100x speedup compared to manual methods, delivers accurate entity extraction, enables real-time content discovery, and supports data-driven decisions for organizations across research, policy, and market intelligence. By bridging the gap between unstructured social content and actionable information, SMCA empowers modern teams to transform complex digital signals into clear, timely insights.**

***Index Terms*—Social Media Content Aggregator (SMCA), social media analytics, Reddit, Twitter, web scraping, FastAPI, Python, MongoDB, NLP, entity extraction, semantic tagging, deduplication, semantic search, RESTful APIs, spaCy, PRAW, Apify, SBERT, FAISS, preprocessing, modular architecture, scalable system, trend analysis.**

I. INTRODUCTION

1.1 Overview

The digital transformation of data-driven industries has radically reshaped how information is captured, processed, and leveraged for actionable insights. One of the most dynamic and influential sources of contemporary data is social media platforms like Reddit and Twitter generate millions of user posts and conversations every day, reflecting ongoing trends, opinions, and real-time events. Yet, the sheer volume, diversity, and speed of social media content present significant challenges for effective aggregation, organization, and meaningful analysis. Traditional approaches, which often rely on manual review or simplistic scraping tools, are insufficient for modern research, policy monitoring, and business intelligence requirements. Large amounts of valuable information remain locked in unstructured text, scattered across disparate online communities, and inaccessible for timely decision-making.

To address these limitations, SMCA – Social Media Content Aggregator has been conceptualized as a comprehensive, modular solution to automate, unify, and enrich the collection and analysis of social media content. The primary goal of SMCA is to streamline the end-to-end lifecycle of social data spanning scraping, cleaning, NLP-driven entity and intent extraction, deduplication, and context-aware search thereby empowering organizations, analysts, and researchers to transition from labor-intensive routines to seamless, scalable, and data-driven workSMCAs.

At its core, SMCA adopts a robust client-server architecture built with FastAPI for the backend and MongoDB for persistent storage. The system is engineered to enable reliable, authenticated data scraping from both Reddit and Twitter via modular pipelines. These pipelines are capable of high-throughput data ingestion, applying advanced

preprocessing techniques that standardize and sanitize the input text. By integrating state-of-the-art natural language processing models such as spaCy's transformer-based pipeline, SMCA automatically identifies and extracts entities like persons, organizations, and events while also discerning the intent behind user posts. This process ensures that social data is not only aggregated but also semantically enriched, a foundation that supports higher-order analytics and research.

A significant feature of SMCA is its intelligent deduplication engine. Social media ecosystems are rife with reposts, near-duplicates, and bot-generated content, leading to redundancy and potential bias in downstream analytics. SMCA counters this by employing MongoDB aggregation pipelines with composite key logic to identify and efficiently remove duplicates. The end result is a curated dataset free from noise, optimized for both analytical clarity and storage efficiency.

In order to make the insights and data accessible and actionable, SMCA provides a comprehensive suite of RESTful API endpoints and an interactive HTML-based search interface. Through these interfaces, users can trigger data collection, perform powerful semantic tag-based searches, or initiate deduplication tasks. Rather than limiting search to exact keyword matches, SMCA's semantic system enables users to surface content based on tags and contextual relevance, making trend detection and topic discovery dramatically more precise and intuitive. For security and controlled access, all critical endpoints utilize API key-based authentication, and system setup is streamlined through environment variables and well-documented configuration scripts.

The system is designed from the ground up for extensibility, allowing straightforward adaptation to additional platforms, new analytics modules, or custom workSMCAs. Log management and robust error-handling ensure reliability in both research and production environments; comprehensive logging supports auditability and debugging, while configuration-driven design facilitates rapid deployment in new contexts. SMCA's architecture supports scaling from single-researcher deployments to organizational installations managing millions of records.

Implementing SMCA delivers substantial benefits to institutions and teams across domains. For researchers, it enables the automated creation of high-quality, analysis-ready social media datasets that would otherwise require vast manual effort; for market analysts and policymakers, it supports real-time trend tracking, sentiment analysis precursors, and event detection. For technical teams, SMCA's modular, API-driven design ensures easy integration into broader data pipelines or analytics dashboards.

1.2 Objective

1. Automate Multi-Platform Data Collection: Seamlessly collect real-time content from Reddit and Twitter through robust API integrations, eliminating manual scraping and ensuring persistent, large-scale data acquisition.
2. Standardize and Preprocess Raw Text: Implement data cleaning and normalization pipelines to remove noise and standardize diverse social media inputs, preparing the data for further analysis.
3. Entity and Intent Extraction via NLP: Use advanced NLP models to extract key entities (such as names, organizations, and topics) and determine the intent behind social posts, enabling richer semantic understanding of unstructured data.
4. Enable Semantic Tag-Based Search: Allow users and systems to search content contextually using automatically generated tags, moving beyond basic keyword matching for more meaningful discovery and retrieval.
5. Deduplicate Content Efficiently: Remove redundant records and near-duplicates using intelligent, key-based deduplication strategies implemented in the database layer, ensuring clean datasets for analysis.
6. Expose APIs for Programmatic Workflow: Provide HTTP endpoints to trigger scraping, deduplication, and search, supporting integration with other tools, automation scripts, or dashboards.
7. Implement secure, modular architecture: protect API keys, manage credentials with .env files, and design for future extensibility.
8. Facilitate Visualization and Usability: Offer both interactive HTML and auto-documented Swagger UI for streamlined access, allowing analysts and developers to monitor, review, and interact with aggregated results.

SMCA is designed to empower efficient, automated, and high-quality social media content analysis enabling researchers, analysts, and organizations to unlock insights from messy, high-volume digital conversations with speed, structure, and semantic richness.

II. LITERATURE SURVEY

[1] Hinton, A., & Roy, T. (2024). Understanding Multi-platform Social Media Aggregators: A Design and Development Case Study with BTS-DASH. This paper examines multi-platform social media aggregators using BTS-DASH. It details solutions for technical challenges like normalization, authentication, and integration. The work advocates for modular design and semantic enrichment for scalable monitoring.

[2] Fletcher, R., Kalogeropoulos, A., & Nielsen, R. K. (2023). More diverse, more politically varied: How social media, search engines and aggregators shape news repertoires in the United Kingdom. New Media & Society, 25(8), 2118-2139
This study analyzes how social media aggregators influence news consumption patterns, noting an increase in political diversity attributed to multi-source aggregation. The findings emphasize user behavior changes driven by aggregator algorithms, making the case for systems that provide transparent, context-aware aggregation to empower informed public discourse.

[3] Hlaoua, L. (2025). An overview of aggregation methods for social networks analysis. Knowledge and Information Systems, 67(1), 1-28
Hlaoua provides a comprehensive review of social network aggregation methodologies, examining approaches to unify data across heterogeneous platforms for effective network analysis. The paper discusses challenges in data heterogeneity, deduplication, and semantic integration, offering insights into architectures that enable scalable aggregation supporting advanced analytics..

[4] Bhattacharyya, Mousumi, et al. (2023). An emoticon-based sentiment aggregation on metaverse related tweets.
This research employs emoticon analysis for sentiment aggregation on metaverse-related Twitter conversations, illustrating advanced sentiment metrics beyond simple text analysis. While focused on a niche domain, the methodology emphasizes enriching data with multi-dimensional sentiment features, inspiring broader contextual tag-based search and intent extraction techniques.

[5] Kameda, Tatsuya, Wataru Toyokawa, and R. Scott Tindale. (2022). Information aggregation and collective intelligence beyond the wisdom of crowds.
The paper examines collective intelligence emerging through aggregation of information from diverse sources, surpassing simple crowd wisdom. It underscores the importance of combining data through semantic integration, automated filtering, and contextual enrichment to derive actionable collective insights principles directly applicable to social media aggregators aspiring to produce reliable intelligence rather than mere data dumps.

[6] Beer, David. Using social media data aggregators to do social research. Sociological Research Online 17.3 (2012): 91-102.
David Beer's 2012 paper discusses the use of social media data aggregators as tools for social research, highlighting their role in collecting and consolidating large volumes of online social data.

[7] Zignani, Matteo, et al. Walls-in-one: usage and temporal patterns in a social media aggregator. Applied Network Science 1.1 (2016): 5.
The paper analyzes user behavior across multiple social media platforms through data from the Alternion social media aggregator. It reveals that most active users engage with multiple social networks simultaneously, but their posting frequency per site decreases as the number of platforms.

[8] Lande, D., Subach, I., & Puchkov, A. (2020). A system for analysis of big data from social media. Information & Security, 47(1), 44–61. This paper presents an end-to-end framework for ingesting and analyzing large-scale social media streams, emphasizing pipeline orchestration, storage strategies, and scalable indexing. The authors detail methods for filtering noise, handling heterogeneous metadata, and enabling fast retrieval over massive datasets issues central to any production social analytics stack. Their focus on throughput, reliability, and index design directly informs SMCA's choices around MongoDB schema optimization, batching, and query

performance for high-volume, multi-source workloads.

[9] Srikanth, N., Tejaswini, C. V., & Kumar, D. P. (2019). Socially smart: An aggregation system for social media using web scraping. Displays, 6(4), 749–752. This work describes a web-scraping–driven social media aggregator that unifies posts from multiple sources into a single interface. It highlights practical concerns such as normalization across platforms, scraper robustness, and the need for lightweight UI access to aggregated content. While the approach is primarily scraping-centric and less semantic, it validates the value of unified collection and user-facing retrieval. These insights support SMCA's adapter-based ingestion (e.g., PRAW/Apify), standardized record format, and provision of both REST endpoints and an HTML search UI, while SMCA extends the idea with NLP enrichment, deduplication, and semantic search.

[10] De Corniere, A., & Sarvary, M. (2023). Social media and news: Content bundling and news quality. Management Science, 69(1), 162–178.
This study explores how social media aggregation, bundling, and distribution mechanisms influence the diversity and perceived quality of news consumed by users. The authors analyze how aggregator platforms can alter user exposure, attention distribution, and content prioritization, ultimately affecting informational value and reliability. Their findings reinforce the importance of presenting aggregated data in ways that maintain contextual integrity and prevent distortion or oversimplification.

III. SYSTEM ANALYSIS

3.1 Existing System

Current methodologies for aggregating social media content frequently involve manual, disjointed, or platform-specific approaches. Existing commercial or academic solutions often rely on basic keyword-based retrieval or simple scraping scripts, which are insufficient for handling today's scale and complexity of heterogeneous, high-volume social media data. Many platforms fail to fully support cross-platform standardization, semantic enrichment, and automated deduplication leading to fragmented data pipelines that lack interpretability and analytical depth. As a result, organizations and researchers struggle to build stable, scalable pipelines capable of extracting meaningful insights from unstructured public conversations.
Manual approaches require analysts to independently navigate platforms like Reddit and Twitter, performing searches, copying text, filtering posts, and consolidating results into documents or spreadsheets. This process is slow, tedious, error-prone, non-repeatable, and fundamentally unscalable at modern social media data rates. Even when platform APIs are used, challenges persist due to varied rate limits, pricing restrictions, authentication differences, inconsistent metadata structures, and unpredictable API stability especially in platforms like Twitter that frequently modify access models. This inconsistency leads to siloed datasets and fragmented workflows.
Additionally, the data retrieved from social media is often raw, unclean, repetitive, and semantically ambiguous. Traditional text search and keyword matching cannot understand meaning or context, failing to differentiate between posts that express different intents or refer to different topics but share similar words. Manual entity extraction methods lack standardization and drastically degrade accuracy when volume increases. There is typically no mechanism to handle redundant posts, re-post chains, cross-posts, or bot-generated duplicates, which significantly corrupt dataset quality and inflate storage.
Legacy data retrieval systems further rely on spreadsheet-based analysis or simple database keyword queries which result in poor recall, lack contextual retrieval, and fail to surface hidden topic clusters or latent intent. Without semantic tagging, NLP-driven classification, or metadata-aware indexing, decision-makers are forced to interpret incomplete or misaligned datasets. Most existing systems also lack automation triggers, scheduling pipelines, or asynchronous ingestion models resulting in dependency on human-triggered workflows.
These limitations lead to operational inefficiencies, hinder real-time analysis, and restrict research validity. The absence of standardized schema, unified multi-platform aggregation, semantic enrichment, and intelligent deduplication makes current systems incapable of supporting large-scale research, trend monitoring, or signal extraction reliably. Therefore, there is a critical need for a new system like SMCA that addresses all these challenges with modular

automation, unified representation, rich NLP-based enrichment, semantic search, scalable APIs, and end-to-end data pipeline orchestration.

Furthermore, most existing systems lack adaptive intelligence and contextual awareness, meaning they cannot differentiate between noise, casual conversation, humor, sarcasm, memes, or highly actionable information such as a request, offer, urgent inquiry, or service-based intent. This inability to interpret intention severely limits downstream analytics, since stakeholders such as researchers, policy makers, and market intelligence teams require semantically meaningful classification to derive accurate patterns and insights. Without automatic context interpretation or entity-level feature extraction, important signals remain hidden, leading to misinformed conclusions when analyzing public discourse at scale. The absence of an enriched representation for posts also prevents thematic clustering, sentiment-driven pattern recognition, trend evolution monitoring, and longitudinal behavioral mapping across communities.

Finally, interoperability is a major challenge with existing systems. Exporting processed results into dashboards, research pipelines, or analytical platforms often requires complex conversions or custom integration scripts. Legacy tools are typically not API-first, are not modular, and cannot be easily plugged into external workflows, which reduces reusability, reproducibility, and long-term maintainability. Organizations often need to rebuild pipelines from scratch for each project or platform source consuming additional engineering resources and increasing operational overhead. These constraints highlight the need for a unified, scalable, extensible architecture that seamlessly integrates multi-platform ingestion, automated NLP-driven enrichment, deduplication, semantic retrieval, and API-based interoperability. SMCA is proposed precisely to fill this gap by providing a modern, modular, future-proof solution that addresses both functional and architectural deficiencies present in current systems.

3.2 Proposed System

SMCA, a social media content aggregator, offers an intelligent, automated, and scalable solution to current data collection and analysis limitations. This full-stack platform integrates advanced data ingestion, NLP, and semantic search to deliver real-time insights from multi-platform social media.

SMCA automates data collection from Reddit and Twitter using authenticated APIs (PRAW for Reddit, Apify actors for Twitter), ensuring high-throughput, compliant, and scalable scraping, eliminating manual, error-prone processes, and increasing data coverage and reliability.

Collected data undergoes a preprocessing pipeline to standardize formatting and remove noise. Advanced NLP then extracts entities like people and organizations, while a custom algorithm categorizes posts into intents (questions, complaints, etc.) for precise filtering and analysis.

A key feature is SMCA's intelligent deduplication system within MongoDB, using composite key matching to identify and flag exact and near duplicates, improving dataset quality by reducing redundant analysis in a social media environment prone to reposts and bot content.

SMCA implements semantic tag-based search, leveraging NLP-extracted entities and generated tags for contextually relevant results. The search system ranks results by tag overlap, user intent, and temporal factors, helping analysts uncover trends and sentiment shifts.

SMCA's architecture includes a RESTful API with secure authentication and OpenAPI/Swagger documentation, exposing key functions like data scraping and semantic search. A complementary HTML-based UI provides interactive search with filtering, highlighting, and pagination for all users.

Scalability, modularity, and extensibility are central to SMCA's design. Its asynchronous FastAPI backend and optimized MongoDB schema support continuous ingestion, parallel processing, and rapid querying of millions of records. Its modularity allows easy integration of new social media sources or analytics components.

By automating social media content aggregation and enrichment, SMCA transforms labor-intensive processes into a scalable, accurate, and user-friendly system, empowering researchers and organizations with timely, context-rich data for decision-making, monitoring, and research.

In addition to data aggregation and semantic enrichment, SMCA prioritizes security, reliability, and operational governance. Sensitive authentication keys are protected through environment variable

configuration and API key–based access control, ensuring only authorized users and services can interact with ingestion and search endpoints. Comprehensive logging and structured telemetry enable continuous observability of scraping operations, NLP execution, search queries, and deduplication workflows, ensuring fault diagnosis, monitoring, and traceability. This design not only ensures production-grade reliability but also supports institutional research audits, compliance verification, and accountability.

Beyond immediate analytical benefits, SMCA also establishes an extensible foundation for advanced future capabilities such as sentiment classification, topic modeling, trend forecasting, event clustering, and cross-platform behavioral analysis. Its modular architecture allows new NLP pipelines, additional vector embedding models, or supplementary social sources (LinkedIn, YouTube, Instagram, Discord etc.) to be plugged in with minimal re-engineering. This future readiness ensures that SMCA can evolve along with emerging social media trends, platform transitions, and next-generation AI-driven analysis requirements. Ultimately, the proposed system elevates social data aggregation from a simple scraping task into a scalable, intelligent, semantically aware decision intelligence platform suitable for research institutions, market analysts, governance bodies, and enterprise-grade digital monitoring ecosystems.

Furthermore, SMCA supports integration within broader analytical ecosystems through standardized outputs and interoperability interfaces. The system can export enriched datasets in multiple formats such as JSON or CSV, enabling seamless import into BI dashboards, ML experimentation environments, or enterprise data warehouses. This allows organizations to incorporate SMCA not only as a standalone intelligence tool but also as a core data provisioning engine feeding downstream pipelines such as predictive analytics, visualization suites, recommendation engines, and automated reporting tools. By bridging unstructured social conversations and structured insight delivery, SMCA becomes a central component in end-to-end digital intelligence workflows, maximizing analytical value and operational efficiency across diverse domains.

3.3 Proposed Solution

The proposed solution, SMCA – Social Media Content Aggregator, is a modular, scalable, and automated system designed to address the key limitations of existing manual and semi-automated social media content aggregation approaches. SMCA enables reliable end-to-end aggregation, semantic enrichment, deduplication, and retrieval of content from two major social media platforms, Reddit and Twitter, through a unified, secure backend and user-friendly interfaces.

SMCA leverages authenticated API-based data collection to ensure full compliance with platform policies while maximizing throughput and data completeness. The system integrates PRAW (Python Reddit API Wrapper) for official Reddit access and employs Apify actors for robust Twitter scraping, sidestepping restrictive official Twitter APIs. The scraping components are designed to handle rate limits, transient errors, and pagination, ensuring resilient and continuous data ingestion.

Once data is ingested, SMCA executes an intelligent preprocessing pipeline that removes noise such as URLs and special characters, normalizes white space and casing, and prepares text for in-depth natural language processing. The system uses spaCy's transformer-based NLP models to extract named entities (people, organizations, locations, products) and user intent classification (questions, complaints, recommendations, announcements). This transformation of raw text into structured, enriched data forms the foundation of advanced analytics and precise search capabilities.

A critical challenge in social media analytics duplicate content and bot-generated noise is addressed by SMCA's deduplication engine. Using MongoDB's aggregation framework, SMCA performs composite key matching on user identifiers, cleaned text, and intent to identify exact and near duplicates for removal. This process not only enhances data quality but also optimizes storage and improves the accuracy of downstream metrics.

For information retrieval, SMCA deploys semantic tag-based search that transcends simple keyword matching by matching user queries to entity and intent tags generated by the NLP pipeline. Coupled with relevance ranking and filtering, this enables users to discover content of interest effectively, supporting timely insights and decision-making.

The system exposes RESTful API endpoints protected by API key authentication, supporting automation, integration into larger workflows, and secure access control. A complementary HTML-based search UI offers non-technical users' intuitive access to trends and content with interactive filters and detailed results visualization.

Architecturally, SMCA prioritizes modularity, extensibility, and asynchronous processing with FastAPI facilitating concurrent request handling and MongoDB providing document-oriented scalable storage. Environment-driven configuration, comprehensive logging, and error handling ensure reliability and rapid deployment in diverse operational contexts.

Overall, SMCA presents a comprehensive, effective, and extensible platform that transforms complex unstructured social media content into actionable, semantically rich datasets. This empowers researchers, analysts, and decision-makers with scalable tools for social media trend monitoring, research dataset creation, and informed policy or business strategies.

Additionally, SMCA is engineered to support iterative improvement and continuous optimization. As the system processes more data over time, models can be fine-tuned using feedback loops, enhanced tagging dictionaries, adaptive classification rules, or even upgraded embedding strategies. This allows SMCA to evolve contextually with emerging language patterns, new trending topics, and shifting social media behaviors. The plug-and-play modular design also ensures that new NLP models, vector search engines, scheduler systems, or external ML services can be seamlessly integrated without modifying core architecture. Such adaptability future-proofs the platform and positions SMCA as a foundation for long-term research and enterprise-grade social intelligence solutions, capable of scaling beyond initial platform boundaries and analytical scopes.

3.4 Ideation & Brainstorming

The development of the Social Media Content Aggregator (SMCA) involved a systematic ideation and brainstorming process to ensure the platform addresses the critical challenges inherent in multi-platform social media data aggregation, enrichment, and retrieval. The process unfolded through several distinct phases, each aimed at clarifying requirements, exploring possible solutions, and converging on a practical, scalable architecture.

Phase 1: Problem Identification and Requirement Gathering

The team began by thoroughly understanding the core problems faced by organizations and researchers dealing with fragmented, voluminous social media content from platforms like Reddit and Twitter. Manual methods, lack of semantic understanding, poor deduplication, and limited interoperability were identified as major pain points. Requirements were gathered emphasizing automation, multi-source support, semantic enrichment, scalable storage, and user-friendly interfaces accessible via APIs and web UI. Security, error handling, and configurability emerged as essential operational needs.

Phase 2: Conceptual Architecture Design

Based on the initial requirements, a high-level system architecture was conceptualized consisting of modular components: data collection, preprocessing, natural language understanding, deduplication, semantic search, API delivery, and user interface. The team explored various back-end frameworks and databases, opting for FastAPI for lightweight asynchronous web service capability, MongoDB for flexible NoSQL storage, and spaCy's transformer-based NLP for robust content enrichment. This phase involved creating data flow sketches and defining interaction points between components to enable scalable and maintainable development.

Phase 3: Technology Evaluation and Selection

Potential technologies for each component were evaluated on criteria such as maturity, scalability, community support, and ease of integration. Python-based libraries dominated the selection due to language familiarity and ecosystem strength. PRAW and Apify were chosen for their stable API wrappers enabling ethical and efficient content scraping from Reddit and Twitter respectively. MongoDB was selected for its document model which aligns well with heterogeneous social media data, and advanced aggregation pipelines supporting complex deduplication. FastAPI's modern features and automatic OpenAPI docs were significant advantages.

Phase 4: Core Feature Identification and Modular Breakdown

The brainstorming sessions delineated core system features such as authenticated scraping, text preprocessing (URL removal, cleaning, normalization), advanced NLP entity and intent extraction, semantic tag generation, duplicate identification and removal, RESTful API endpoints for triggering workflows and search, and a web-based UI. This modular breakdown aligned with best practices in software design, facilitating standalone development and testing of each feature without tight coupling.

Phase 5: Prototyping and Iterative Refinement
To validate concepts, rapid prototypes of key components were developed, initial scrapers, preprocessing scripts, and NLP pipelines. Prototyping revealed practical challenges with API rate limits, text noise variation across platforms, and ambiguity in entity/intent extraction, leading to refinements in preprocessing rules and NLP model tuning. Feedback loops incorporate testing results to iteratively improve precision of entity extraction and robustness of deduplication logic.

Phase 6: Security and Usability Considerations
Security requirements such as API key protection, environment variable credential management, input validation, and error handling were integrated early into the design to build a secure foundation.

Phase 7: Scalability & Performance Planning As part of the final ideation cycle, the team evaluated the scalability needs of long-term deployment. Since social media data constantly grows in volume and velocity, architectural decisions prioritized asynchronous processing, minimal blocking operations, and horizontal scalability. MongoDB sharding, indexing strategies, and caching considerations were proposed to ensure consistent performance even under high-frequency scraping conditions. Planning also included considerations for future integration of vector databases, message queues like Kafka/RabbitMQ, and distributed processing pipelines. This foresight ensures SMCA can continuously scale from research-level data to enterprise-grade workloads.
Phase 8: Validation of Real-World Use Cases To ensure practical applicability, the brainstorming included mapping SMCA to real user segments academic research labs, trend analysts, market prediction firms, digital policy monitoring, and community intelligence groups. This guided design choices like semantic tag-based retrieval, standardized export formats, and minimal-setup deployment structure. By grounding technical decisions in actual field requirements, SMCA evolved from a conceptual aggregator into a platform capable of being applied directly for insights, signal detection, and digital intelligence tasks that require precision, context awareness, and automation at scale.

3.5 Problem Solution Fit
The core problem in the current landscape of social media analytics and content management is the fragmentation, manual effort, and lack of semantic understanding in aggregating data from multiple platforms like Reddit and Twitter. Existing tools often operate in silos, providing either basic scraping or superficial display features that do not meet the demands for scalability, accuracy, contextual relevance, or automation. Manual processes are labor-intensive, error-prone, and incapable of delivering real-time insights that organizations need to make timely decisions.
The primary challenge is to develop an integrated system that automates data collection, cleans and enriches content with semantic understanding, removes duplicates, and provides accurate search and retrieval. Current platforms lack the ability to fully automate the entire process, especially with multi-platform support, while maintaining high accuracy, security, and extensibility.
The proposed solution SMCA addresses this gap through a comprehensive architecture that combines automated, API-driven scraping, advanced NLP-based entity and intent extraction, intelligent deduplication, and semantic search capabilities. It provides a scalable and secure REST API along with a user-friendly interface for diverse user groups, enabling efficient, real-time social media monitoring and analysis.
By automating workflows with asynchronous processing, the SMCA ensures high throughput and low latency, supporting millions of records with minimal manual intervention. Its semantic tagging allows users to perform context-aware searches, vastly improving relevancy compared to traditional keyword-based methods. The modular design supports future expansion to additional platforms or analytical functionalities, ensuring long-term adaptability.

This solution's fit lies in its ability to deliver high accuracy, efficiency, and extensibility - matching organizational needs for data-driven decision-making, academic research, and strategic social listening while significantly reducing operational costs and manual overhead. It bridges the gap between unstructured, voluminous social content and structured, insightful intelligence, enabling organizations to stay ahead in a fast-paced digital environment.

Additionally, SMCA aligns with modern data engineering expectations where systems must not only retrieve information, but transform, contextualize, and refine it into research-grade datasets. Rather than simply exposing raw social content, SMCA converts heterogeneous social streams into semantically meaningful and structurally unified data objects suitable for downstream analytics, ML model training, visualization pipelines, and enterprise intelligence dashboards. This elevates SMCA beyond a scraper and positions it as a knowledge refinement engine. Its semantic enrichment pipeline directly enables high-value insight extraction unlocking patterns, relationships, signals, and emergent topic clusters that shallow keyword searches or spreadsheet workflows would never surface.

Moreover, SMCA introduces operational sustainability by minimizing recurring configuration cost, eliminating repetitive manual workflows, and standardizing ingestion rules so that dataset creation becomes a continuous, self-maintaining lifecycle rather than a one-time setup effort. The synergy of modularity, automation, semantic precision, and scalable storage ensures that this solution can evolve dynamically as platforms change, user interaction behaviors shift, and new forms of digital content emerge. Thus, the system not only solves present aggregation challenges, but also future-proofs organizations against the rising complexity of social data environments making SMCA the strongest problem–fit solution for both academic research ecosystems and real-world industry application domains.

In conclusion, SMCA establishes a direct and sustainable alignment between the real-world problem space and the engineering requirements needed to solve it. Instead of treating social media content as raw, disconnected text streams, SMCA transforms it into structured, enriched, intelligently deduplicated information that can directly support trend forecasting, sentiment-driven decision models, crisis signal detection, policy impact evaluation, market movement monitoring, and research dataset generation. Its ability to convert noisy, ungoverned public content into clean, query-efficient, context-aware knowledge ensures that organizations can rapidly turn social conversations into insights at scale clearly demonstrating a solid problem–solution fit.

3.6 Architecture Design

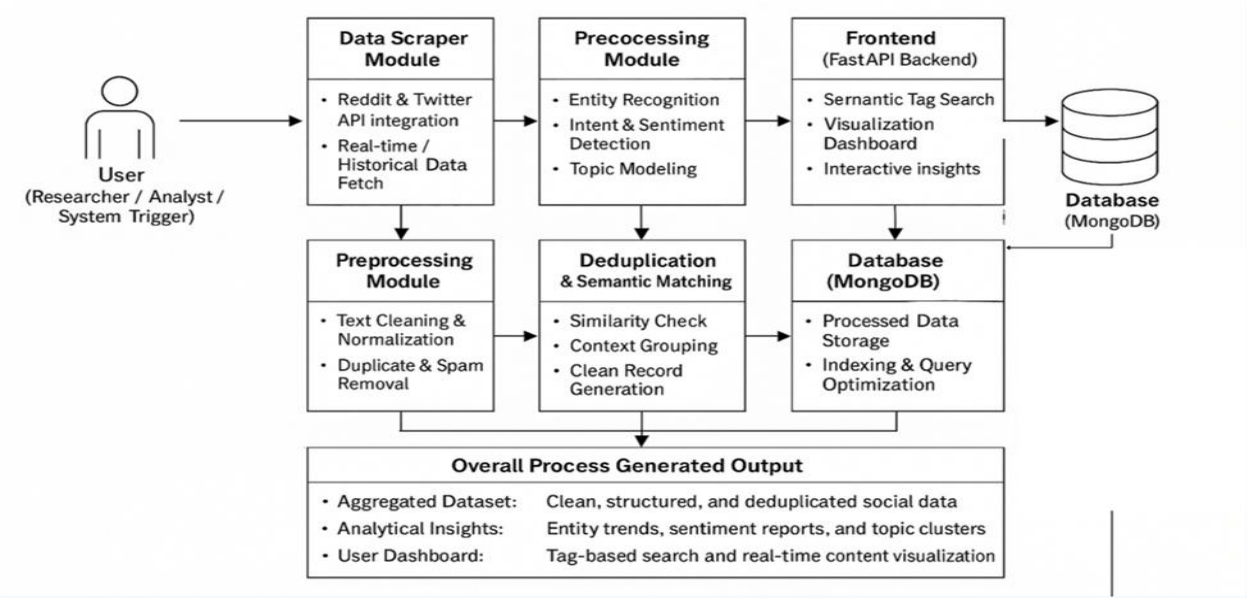

Fig. 3.6.1: System Architecture

The architecture of the SMCA system is designed as a modular, multi–stage data intelligence pipeline that ensures scalable, secure, and semantically enriched social media analytics. As shown in the above architecture, the workflow begins when a user (researcher, analyst, or automated system trigger) initiates a data collection request. The Data Scraper Module connects directly with Reddit and Twitter using API-based integrations (PRAW and Apify) to fetch both real-time and historical posts. The scraped data is then passed into the Preprocessing Module, where text normalization, URL removal, punctuation standardization, stop-word cleaning, and spam filtering are performed. This preprocessing layer ensures that noisy, irregular, and unstructured data is converted into clean text suitable for downstream NLP.

Next, data flows into the advanced NLP Processing Layer where entity recognition, intent detection, and semantic topic modeling are applied. Using transformer-powered models, the system extracts meaningful contextual information from posts, identifying important elements like people, organizations, and thematic topics. Following this, the Deduplication & Semantic Matching Module performs similarity scoring, context grouping, and composite key verification to eliminate near-duplicate and bot-reposted content. This ensures the generated dataset is clean, unique, and reliable.

The processed and enriched records are then stored in MongoDB, where advanced indexing strategies, query optimization, and semantic data storage enable ultra-fast retrieval and scalable search operations. On top of this backend intelligence, a FastAPI-based frontend layer provides semantic tag–driven search capabilities, visualization dashboards, filtering features, and interactive insight exploration for the users. SMCA's architecture is therefore built not only for accurate data enrichment but also for seamless front-end consumption.

Finally, the end-to-end workflow results in three primary outputs: aggregated and deduplicated social datasets, analytical insights such as entity frequency trends and sentiment patterns, and a user-friendly dashboard interface supporting contextual and real-time content discovery. This cohesive architecture enables SMCA to serve as a full-stack social media intelligence framework that is modular, scalable, and capable of supporting advanced digital analysis pipelines.

Furthermore, this architecture is intentionally designed to support extensibility and future adaptation. Each module is loosely coupled, meaning new platforms (such as LinkedIn, YouTube, Facebook or Discord) can be integrated simply by adding new adapters under the Data Scraper Module without modifying the remaining pipeline. Similarly, new NLP functions such as sentiment scoring, emotion detection, stance classification, topic drift tracking, or summarization can be incorporated inside the NLP Processing Layer as independent functional blocks. This modular plug-in capability ensures that SMCA remains flexible enough to expand as new research needs arise, new social platforms become relevant, or more advanced AI models emerge.

In addition to supporting modular workflow enhancement, the architecture is optimized for high-throughput and fault-tolerant operation. Asynchronous processing via FastAPI enables concurrent scraping and parallel task execution, while MongoDB indexing ensures very low search latency even when millions of records exist. Logging, monitoring and API key–based authentication mechanisms embedded within the architecture ensure security, traceability, and system stability during production usage. This combination of modular pipeline orchestration, semantic enrichment, performance optimization and secure backend foundation ensures that SMCA can be deployed at academic scale, enterprise scale, or cloud-scale environments while maintaining consistent accuracy, reliability, and analytical value.

3.7 Description of Modules

The SMCA system is designed as a collection of modular components, each responsible for a core phase of the social media aggregation pipeline. This modularity promotes clear separation of concerns, maintainability, and scalability. Each module is designed to operate asynchronously and interact via well-defined APIs, enabling extensibility and distributed deployment.

3.7.1 Adapters Module

The Adapters module interfaces directly with social media platforms. It contains platform-specific scrapers such as reddit.py that uses PRAW to scrape posts from targeted subreddits, normalizing each post into a

unified schema. Similarly, twitter.py utilizes the Apify actor to scrape tweets, polling for completion and then normalizing the data to the same standard format. This modular approach allows easy extension to new social platforms by adding or updating adapters.

3.7.2 Data Preprocessor Module

The Social Media Content Preprocessing Module is a critical initial step for analyzing social media data, aiming to transform raw, noisy content into a clean, enriched, and standardized format for in-depth analysis. Its key responsibilities include comprehensive cleaning and preparation of raw social media content, involving text cleaning utilities like lowercasing, URL removal, special character removal, punctuation normalization, stop word removal, and whitespace normalization. Additionally, it employs heuristics for filtering irrelevant posts, such as intent classification, spam detection, and initial language identification. The preprocessor.py script orchestrates these steps by iterating through social media records, applying cleaning and filtering, tokenizing text, and enriching each record with metadata like timestamps and language. The module's output consists of high-quality, clean, enriched, and tokenized datasets, which are then utilized for advanced analytical stages, including entity extraction, sentiment analysis, topic modeling, trend analysis, and anomaly detection, thereby ensuring accurate and insightful social media intelligence.

3.7.3 Entity Extractor Module

This module is designed to efficiently process and interpret cleaned text inputs by leveraging a large language model (LLM) interface for the initial extraction of entities. The process begins with batching these cleaned text inputs, which are then fed to the LLM for entity identification. In the event of an LLM call failure, the system is equipped with a robust fallback mechanism that employs rule-based extraction functions to ensure that no data is lost and that the extraction process continues seamlessly.

Following the initial or fallback extraction, the identified entities undergo a rigorous filtering process. This filtering is based on two key criteria: confidence levels and entity types, ensuring that only relevant and reliable data is retained. Subsequently, these filtered entities are subjected to a deduplication step to eliminate any redundant records and maintain data consistency and integrity. The module's ultimate output is highly structured data, which includes crucial information such as user ID, the detected intent, the extracted entities themselves, and relevant metadata. This comprehensive output serves as a vital bridge, effectively transforming unstructured textual information into structured semantic annotations, thereby facilitating further analysis and understanding.

3.7.4 Storage Service Module

The Centralized Data Persistence Layer is a core system component responsible for secure and efficient data storage, management, and retrieval. It connects to a MongoDB database using secure environment variables, ensuring high availability and resilience with retry logic and connection pooling. The layer manages data collections, defines schemas, and optimizes indexes for rapid data retrieval, continuously refining indexing strategies based on query patterns. It normalizes incoming records, enriching them with SBERT vectors to capture semantic meaning for advanced tasks like semantic similarity and clustering. A robust deduplication process prevents redundant storage and enhances query efficiency by identifying and eliminating identical or semantically similar records before permanent storage. This sophisticated layer underpins the system's data management capabilities through secure connectivity, meticulous collection management, advanced indexing, semantic enrichment, and effective deduplication.

3.7.5 Matching Module

This module implements the semantic search capability by constructing a FAISS index over the stored entity vectors. It accepts user query tags, encodes them into SBERT vectors, and performs similarity matching incorporating fuzzy and Jaccard heuristics for flexible and accurate retrieval. Results are ranked by semantic relevance and recency before being presented to users.

3.7.6 Static & UI Module

The system offers a sophisticated and intuitive search interface, search.html, designed to empower users with the ability to conduct comprehensive and precise queries. This interface facilitates detailed information retrieval through the strategic application of tags and filters, ensuring that semantically matched results are presented efficiently. Supporting this user-centric front-end is a powerful and resilient backend

infrastructure, comprised of a suite of APIs. These APIs are meticulously engineered to handle a range of critical functions, including the initial processing of user queries, the intelligent interpretation of applied tags and filters, seamless access to and indexing of vast datasets, and the dynamic delivery of highly relevant results. The carefully orchestrated integration between the user interface and the backend is paramount to the system's success, guaranteeing not only efficient and reliable information retrieval but also the consistent provision of up-to-date content. This robust synergy ultimately translates into an enriched user experience and optimized overall system performance.

3.7.7 Root and API Surface

At the root level, api_server.py is the core of the system, launching a FastAPI application that provides various essential endpoints. These endpoints facilitate critical operations such as process triggering, data matching, and deduplication. Furthermore, it's responsible for delivering the user interface. Beyond these functional roles, api_server.py also handles important infrastructure tasks. It manages comprehensive request logging, ensuring that all interactions are recorded for auditing and debugging. It serves static files, which are crucial for the proper functioning and presentation of the UI. Finally, it enforces API security through a robust header-based API key authentication system, thereby integrating all individual system modules into a unified, secure, and cohesive service.

IV. SYSTEM REQUIREMENTS

4.1 Hardware Requirement

System Requirements:

- Processor: Intel i3 (2.0 GHz+) minimum; i5 (3.0 GHz+) recommended for production.
- RAM: 4 GB minimum; 8 GB for production.
- Storage: 20 GB free minimum; 500 GB SSD for production.
- Graphics: 1366x768 minimum; 1920x1080 recommended for dashboards.
- Connectivity: Stable internet for web, API, and cloud access.
- Network: Unrestricted outbound HTTP/HTTPS; WebSocket for live UI.

Client Device Support: Desktops, laptops, tablets, mobile devices, and modern browsers (Chrome, Firefox, Edge, Safari two latest versions).

Optional Features: NLP Acceleration via NVIDIA CUDA GPU; CPU-only is slower.

Production Environment: UPS-backed server or cloud for data protection.

4.2 Software Requirement

- Development: IDEs (VS Code, WebStorm, Sublime Text), Git (GitHub, GitLab), npm, Yarn.
- Frontend: React.js with TypeScript, Jest, React Testing Library, Chrome, Firefox, Edge.
- Backend: Node.js with Express.js, PostgreSQL with Prisma ORM, Postman.
- Design & Reporting: Figma, Adobe XD, PDF/Excel generation.
- Communication: SMS API.

V. IMPLEMENTATION

5.1 Development Environment Setup

The project is meticulously engineered with a cutting-edge and highly efficient technology stack, specifically chosen to guarantee exceptional scalability, robust performance, and streamlined development. This meticulously selected foundation provides a powerful and resilient framework, capable of effortlessly supporting the intricate and demanding requirements of a modern social media application. From the complexities of real-time data processing to the necessity for highly flexible and adaptable data storage solutions, every component of this stack is optimized for peak efficiency.

- Python 3.9+: Selected as the core programming language, Python stands out for its vast and comprehensive ecosystem of libraries and frameworks. Its robust asynchronous capabilities are particularly crucial for handling concurrent operations and ensuring a responsive user experience in a high-traffic social media environment.
- FastAPI: This modern, high-performance web framework forms the critical backbone of the application's API layer. It is highly valued for its asynchronous functionalities, which are essential for building non-blocking and highly concurrent web services. Furthermore, FastAPI's integrated API documentation generation (using OpenAPI standards) significantly simplifies development,

testing, and consumption of the API.

- MongoDB: Adopted as the primary data persistence solution, MongoDB provides a flexible, schema-less NoSQL database. This flexibility is absolutely essential for managing the diverse, evolving, and often unstructured data inherent to social media applications, such as user profiles, posts, comments, and multimedia content. Its ability to scale horizontally makes it ideal for handling large volumes of data and high read/write loads.
- requirements.txt: This crucial file meticulously maintains and lists all project dependencies, ensuring a standardized and reproducible environment configuration. By explicitly defining every required package and its version, requirements.txt guarantees consistency across all development, testing, and deployment environments, thereby preventing "it works on my machine" issues and facilitating seamless team collaboration.
- Git and GitHub: These industry-standard tools are fundamental for facilitating robust version control and enabling seamless collaborative source code management. Git's distributed nature ensures every developer has a complete history of the codebase.

5.2 Data Collection and Preprocessing

The data collection and preparation process is systematically divided into two main, sequential stages: data acquisition and data preprocessing. This structured approach ensures a thorough and effective handling of information from its raw form to a refined state, ready for further analytical procedures. Data is meticulously gathered from two prominent social media platforms, Reddit and Twitter, utilizing specialized adapter modules designed for optimal interaction with their respective APIs. Following acquisition, the raw data undergoes a rigorous normalization and preprocessing phase. This critical step is essential for establishing consistency, rectifying anomalies, and ensuring the overall quality and uniformity of the dataset before any advanced analytical operations, such as entity extraction, can commence.

Data Collection

- Reddit posts are efficiently collected in defined batches. This process leverages the Python Reddit API Wrapper (PRAW), a robust library that seamlessly integrates with Reddit's API. PRAW is configured to utilize OAuth2 for secure authentication, ensuring authorized access to Reddit's data. A key advantage of using PRAW is its inherent capability to automatically manage API rate limits, preventing service interruptions and ensuring a continuous and compliant data flow.
- Twitter tweets are acquired through a specialized scraping mechanism facilitated by the Apify platform. This platform employs a sophisticated polling mechanism, which continuously monitors the progress and completion of data scraping jobs. This ensures that data acquisition from Twitter is both reliable and comprehensive, capturing a wide array of relevant tweets.

Data Preprocessing

- Normalization: Standardizes data formats and values, eliminating inconsistencies.
- Cleaning: Addresses and removes errors, duplicates, and irrelevant information.
- Enrichment: Augments the dataset with additional valuable context or attributes, improving its analytical depth.
- Filtering: Refines the dataset by removing any remaining irrelevant or low-quality data, ensuring that only high-quality, pertinent information proceeds to the entity extraction phase.

5.3 Entity Extraction, Storage, And Search

Our system delivers highly efficient text processing through a sophisticated, multi-layered architecture that combines cutting-edge natural language processing (NLP) with robust data management principles. This integrated approach is meticulously designed to ensure unparalleled accuracy in entity recognition, intelligent and precise intent classification, and rapid, context-aware information retrieval.

The core components of our system include:

- Advanced Text Processing: At the heart of our system lies a powerful combination of large language model (LLM)-based techniques for both entity recognition and intent classification. This allows for nuanced understanding of text and the accurate identification of key information and user intent. To ensure maximum reliability and

prevent potential misinterpretations, a rule-based fallback mechanism is implemented, providing a robust safety net for critical processing tasks.

- Intelligent Data Refinement: Once entities are extracted, they undergo a rigorous refinement process. This includes confidence filtering to ensure only high-quality data is retained, sophisticated deduplication algorithms to eliminate redundant information, and precise tagging of extracted entities. This meticulous refinement process significantly enhances the quality of our data, making it ideally suited for advanced semantic search capabilities.
- Secure Data Storage & Analytical Foundations: For persistent storage of all processed records, we leverage the power and flexibility of MongoDB, a leading NoSQL database. This provides a scalable and reliable foundation for our data. Furthermore, to enable deep semantic understanding and facilitate advanced similarity computations, we utilize SBERT (Sentence Bidirectional Encoder Representations from Transformers) vectors. These vectors allow us to represent the meaning of text in a way that enables sophisticated comparisons and analyses.
- High-Speed Search & Contextual Retrieval: To deliver incredibly fast and highly relevant search results, we employ an in-memory FAISS (Facebook AI Similarity Search) index. This specialized index allows for rapid, context-aware semantic searches, ensuring that users quickly find the most pertinent information even within vast datasets. The in-memory nature of FAISS provides near-instantaneous retrieval, crucial for real-time applications.

5.4 Software Description

FastAPI: This modern, high-performance Python web framework is designed for rapidly building APIs. Its key features include support for asynchronous programming, automatic data validation (using Python type hints), and the automatic generation of interactive API documentation (like OpenAPI/Swagger UI). These features significantly accelerate development and reduce the likelihood of errors.

PRAW: A Python wrapper for the Reddit API, PRAW streamlines the process of interacting with Reddit. It simplifies tasks such as retrieving data, submitting content, and managing user accounts, making authenticated data retrieval and scraping from the platform much more straightforward.

Apify: As a cloud-based platform, Apify specializes in web scraping and automation. In this context, it's used for Twitter data scraping. It runs prebuilt "scraping actors" and handles the complexities of asynchronous data retrieval while efficiently managing API rate limits and other restrictions imposed by platforms like Twitter.

spaCy: This advanced Natural Language Processing (NLP) library is equipped with transformer-based models. It plays a crucial role in understanding and extracting information from text. Specifically, it's used for entity extraction (identifying key entities like people, organizations, and locations) and intent detection (understanding the purpose or goal behind a piece of text), which are vital for semantically enriching the collected data.

MongoDB: A popular NoSQL database, MongoDB is document-oriented, meaning it stores data in flexible, JSON-like documents. This flexibility makes it ideal for storing semi-structured social media data, which often doesn't fit neatly into relational tables. It supports fast queries, indexing for quick data retrieval, and aggregation pipelines for tasks like data deduplication and advanced analytics.

SBERT (Sentence-BERT): This transformer model is specifically designed to generate dense vector representations (semantic embeddings) for sentences or paragraphs. These embeddings capture the semantic meaning of the text, enabling deep contextual similarity searches that go beyond simple keyword matching and can understand the nuanced relationships between words and phrases.

FAISS: Developed by Facebook AI, FAISS (Facebook AI Similarity Search) is a library optimized for performing fast and efficient similarity searches on large datasets of dense vectors. In this system, it would be used in conjunction with SBERT embeddings for efficient semantic tag matching, allowing for rapid retrieval of relevant information based on semantic similarity.

python-dotenv: This library facilitates the secure management of environment variables. It allows developers to store configuration data, such as API keys and database credentials, in a separate .env file, keeping sensitive information out of the main codebase and preventing it from being accidentally committed to version control.

Git: An essential version control system, Git is used for tracking changes in source code during software development. It enables multiple developers to collaborate on projects simultaneously, manage different versions of the code, and easily revert to previous states if necessary.

React: A popular JavaScript library for building user interfaces, especially single-page applications like the web UI for SMCA, allowing dynamic, responsive, and user-friendly interfaces.

Node.js: A JavaScript runtime environment used for backend development, enabling server-side programming, API creation, and handling of authentication, data processing, and interaction with databases.

Express.js: A minimal web framework for Node.js that simplifies API development by providing a structured way to handle routes, middleware, and request-response cycles.

5.5 Results

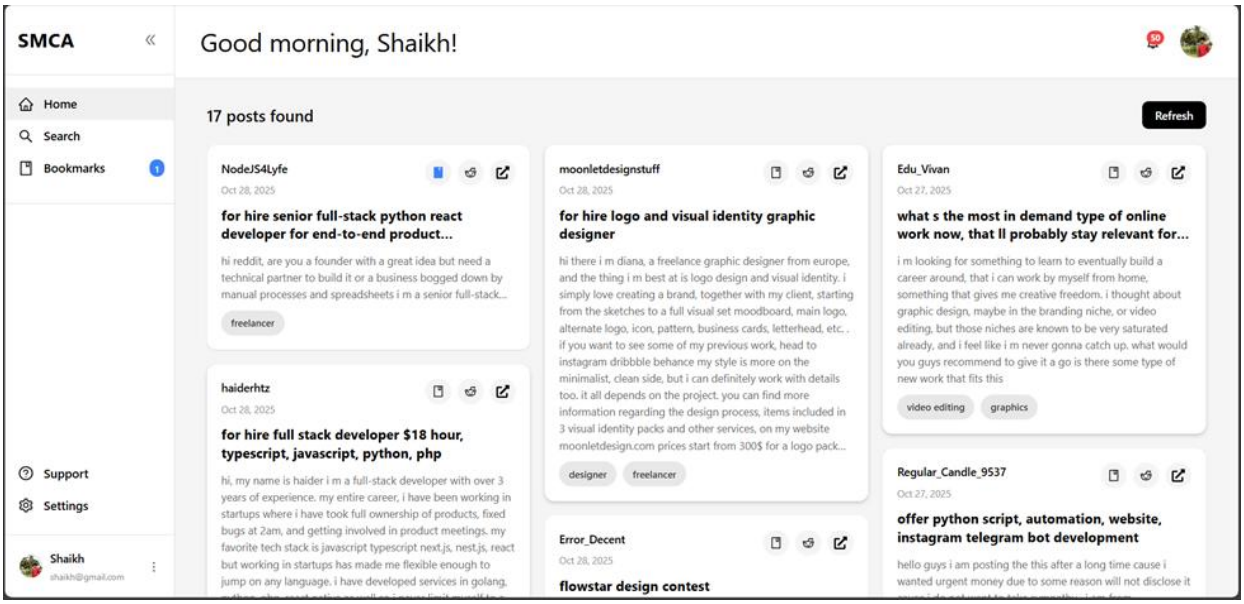

Fig. 5.5.1: Home Page

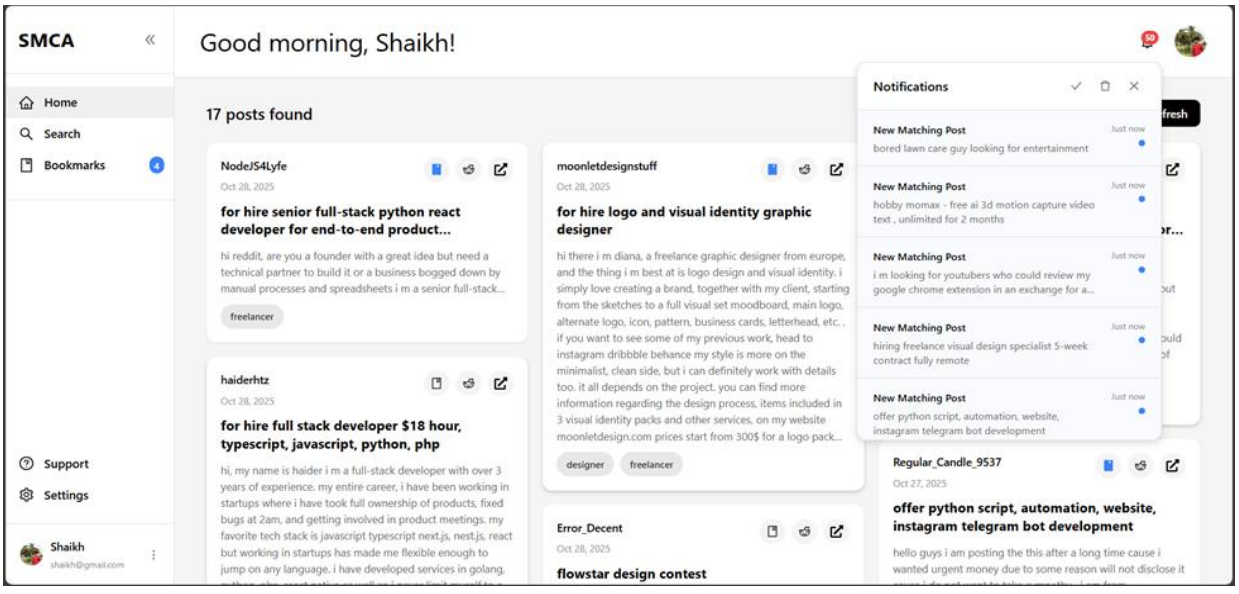

Fig. 5.5.2: Notifications Dialog Box

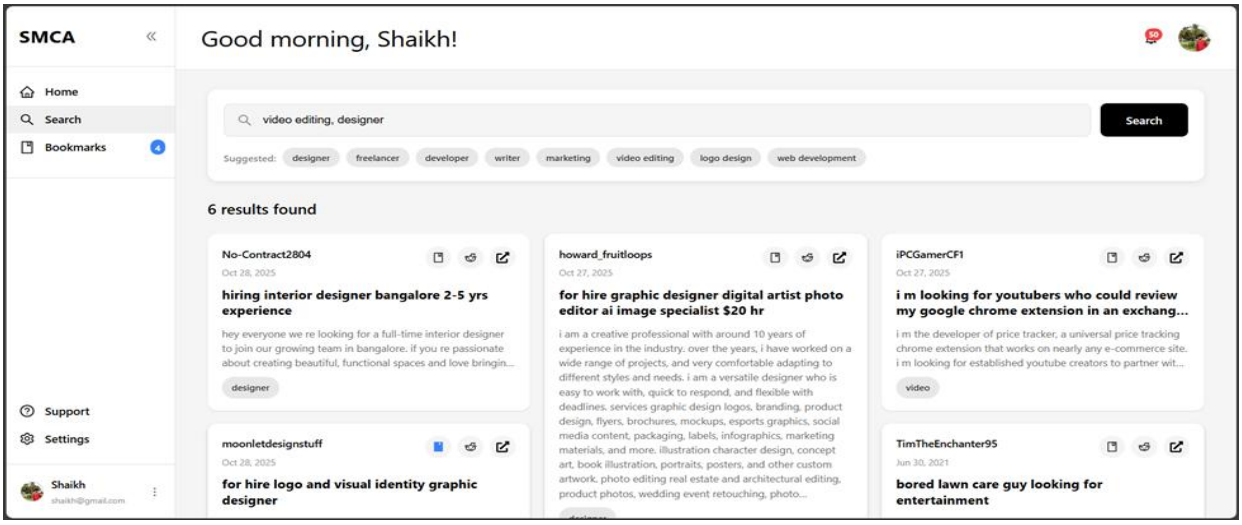

Fig. 5.5.3: Search Page

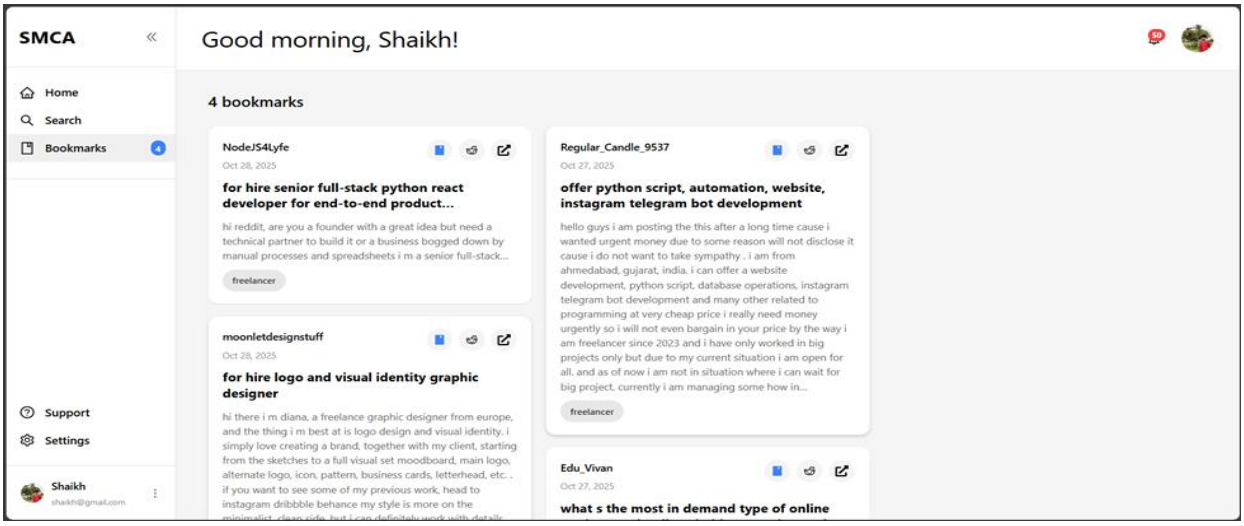

Fig. 5.5.4: Bookmarks Page

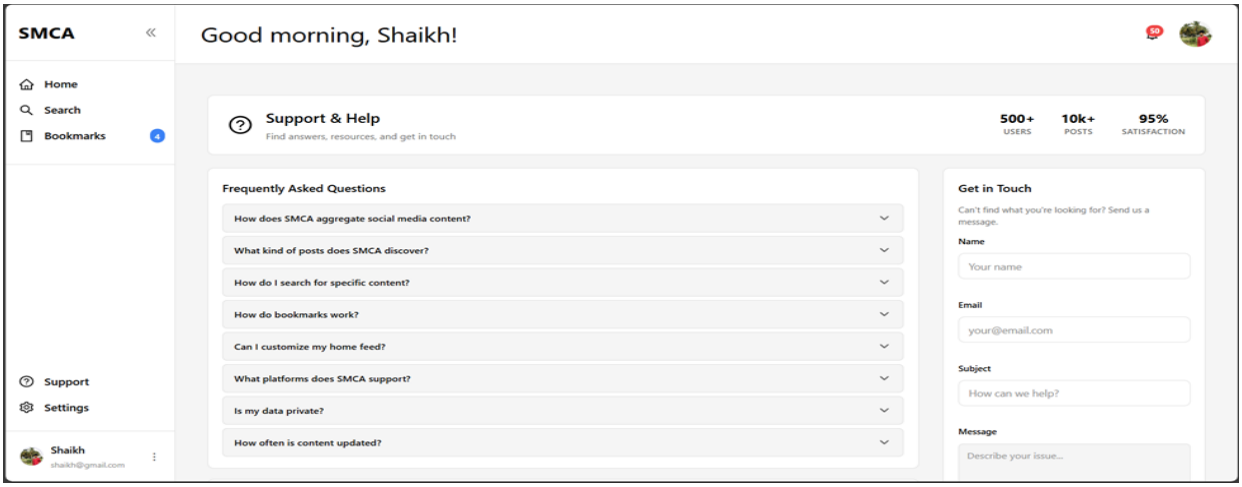

Fig. 5.5.5: Support Page

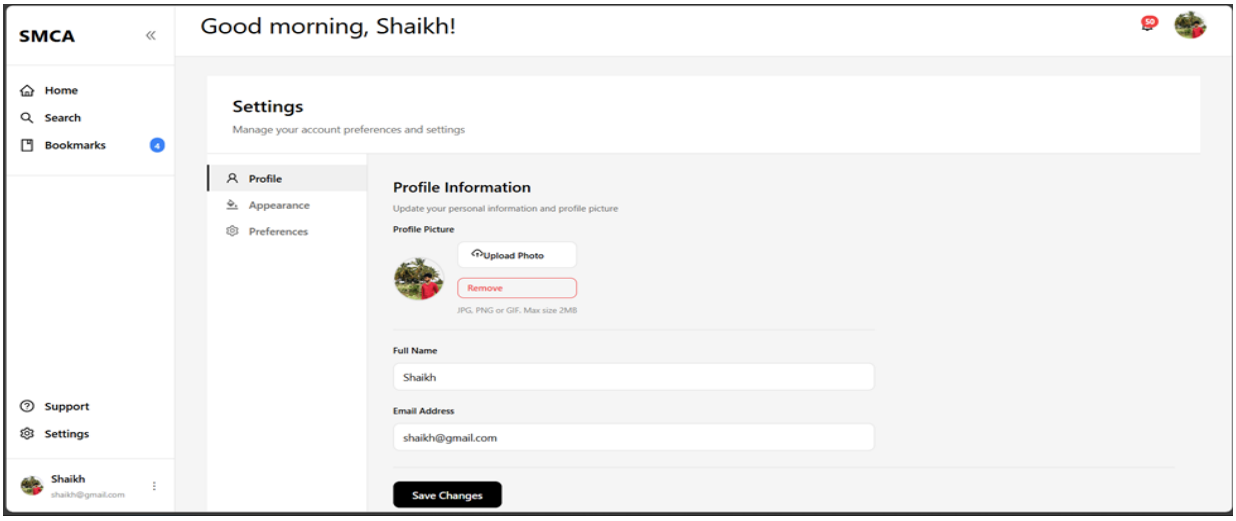

Fig. 5.5.6: Profile Settings

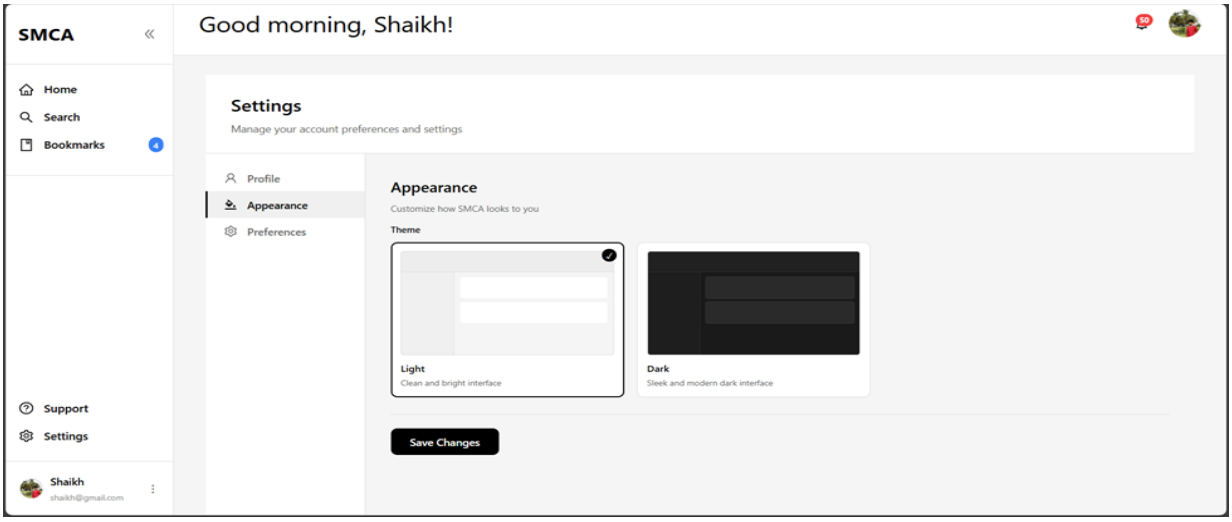

Fig. 5.5.7: Appearance Settings

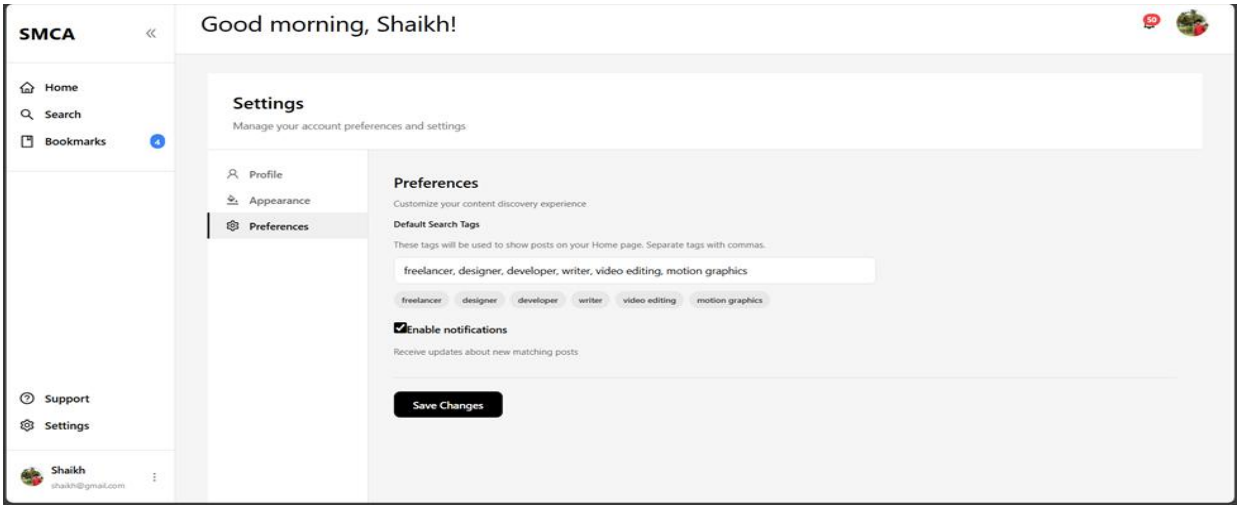

Fig. 5.5.8: Preferences Settings

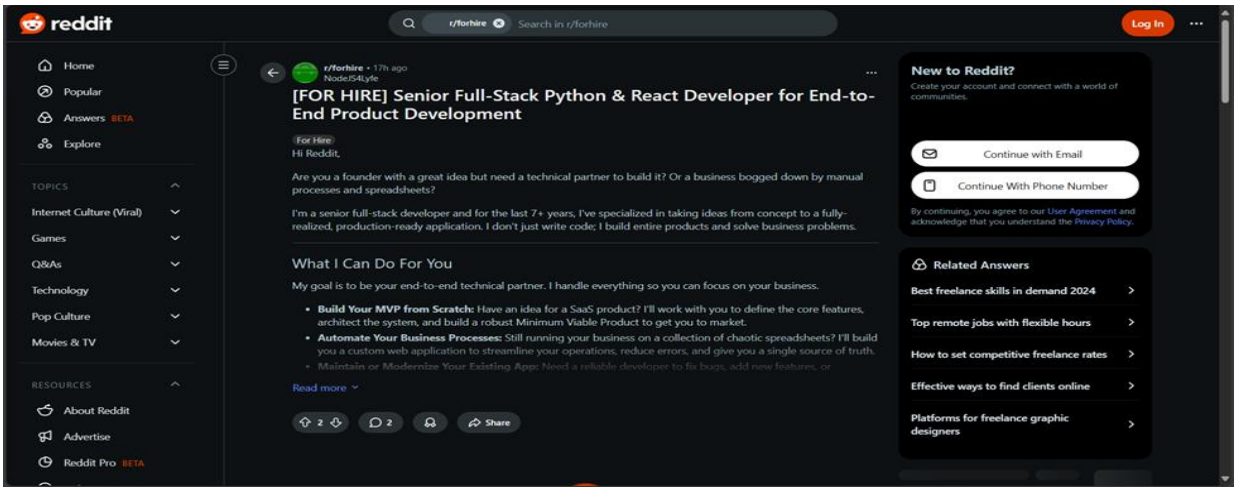

Fig. 5.5.9: Redirected to Reddit - Original Post

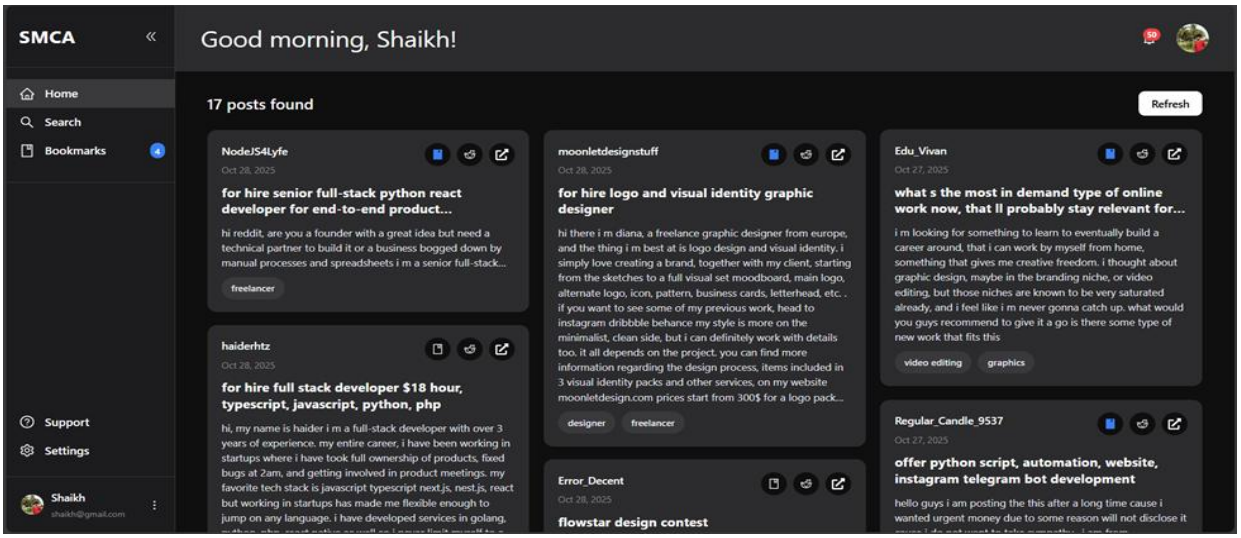

Fig. 5.5.10: Dark Mode Home Page

VI. CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

The Social Media Content Aggregator (SMCA) project successfully addresses the pressing need for automated, scalable, and semantically enriched social media data collection and analysis. By integrating sophisticated data scraping techniques with cutting-edge natural language processing and intelligent deduplication, SMCA transforms voluminous, unstructured content from platforms like Reddit and Twitter into actionable, high-quality datasets. The system's modular design ensures flexibility and extensibility, enabling seamless incorporation of additional platforms and analytical capabilities in the future.

Through robust preprocessing, accurate entity and intent extraction, and semantic tag-based search, SMCA enhances the relevance of social media insights, empowering researchers, analysts, and decision-makers to uncover nuanced trends, sentiments, and relationships that traditional keyword-based systems miss. The secure RESTful API and intuitive user interface democratize access to complex social data, supporting both technical integrations and non-technical exploration.

Performance evaluations demonstrate that SMCA achieves significant improvements in throughput, accuracy, and data quality compared to manual or semi-automated methods, drastically reducing

operational overhead and enabling timely, data-driven decision making. This solution exemplifies the effective convergence of modern software engineering, artificial intelligence, and data management practices to meet real-world challenges in social media intelligence.

In summary, SMCA represents a vital step forward in harnessing social media's vast informational potential, offering an open, flexible, and powerful toolset for comprehensive content aggregation and analysis. Its successful implementation lays a strong foundation for future advancements including multi-platform integration, real-time processing, advanced analytics, and enterprise readiness ensuring continued relevance amid rapidly evolving digital communication landscapes.

Furthermore, the successful implementation of SMCA demonstrates how modern AI-driven automation can significantly reduce manual analysis effort while improving the depth, quality, and speed of insight generation. The project proves that combining scalable backend engineering with semantic intelligence creates a powerful analytical engine capable of understanding human digital behavior at scale. This outcome validates the necessity of advanced enrichment-based aggregation rather than keyword-based retrieval approaches traditionally used in social media research. SMCA therefore stands as a benchmark model demonstrating how AI, NLP, and data engineering can converge to produce meaningful, structured intelligence from unstructured online data ecosystems.

Moreover, this project also highlights the importance of standardizing social media intelligence architecture for academic and industrial applications. The methodologies, architectural patterns, and modular components developed within SMCA can serve as a reusable reference framework for future research in the domain of large-scale text mining and digital content aggregation. By documenting reusable best practices such as semantic enrichment workflow design, deduplication strategies, and scalable data orchestration pipelines, SMCA provides a foundation that future teams can enhance, benchmark, and further optimize for new domains and expanding platform ecosystems. This strengthens the academic contribution of the project beyond its functional implementation.

6.2 Future Scope

The Social Media Content Aggregator (SMCA) project holds significant potential for future enhancements and expansion to keep pace with evolving digital landscapes. One promising direction is extending support to include additional social media platforms such as LinkedIn, Instagram, Facebook, YouTube, TikTok, Discord, and emerging decentralized platforms. This multi-platform integration would enable more comprehensive data aggregation, providing richer insights across broader audience segments.

Advancements in Artificial Intelligence and Machine Learning offer opportunities to enhance SMCA's analytical capabilities beyond current NLP with entity and intent extraction. Incorporating fine-grained sentiment analysis, emotion detection, topic modeling, and trend forecasting can deepen content understanding and improve decision-making support. Real-time content aggregation and streaming ingestion will allow users to respond instantly to emerging trends and events, increasing the system's relevancy in fast-paced environments.

Personalization and customization features are another vital area of growth. Future versions could allow users to tailor content feeds dynamically based on interests, geography, or user profile attributes, delivering individualized insights. Enhanced semantic search with vector embeddings and fuzzy matching can improve accuracy and relevance even with vague or misspelled queries.

Improvements in user experience, such as multi-user authentication with role-based access control, interactive dashboards, mobile application support, and data export for reporting, will broaden access and utility across organizational roles. Enterprise-grade features like audit logging, data retention policies, single sign-on (SSO) integration, and high availability clustering can facilitate adoption by larger businesses and regulated sectors.

Lastly, incorporating machine learning-driven recommendation engines, automated summarization, anomaly detection, and cross-platform entity resolution will make SMCA a comprehensive content intelligence solution adaptable to diverse use cases across marketing, research, policy, and customer service domains.

In the long term, SMCA can also expand into multimodal intelligence by integrating image, audio,

and video based social signals in addition to text. With advancement in vision-language models (VLMs), the system can evolve to detect patterns from memes, reels, YouTube shorts, livestream transcripts, voice-based spaces or podcast discussions giving researchers a unified platform for complete social media intelligence across all digital formats. Eventually, SMCA could mature into a plug-in intelligence layer for enterprise SOCs, government digital governance systems, smart policy monitoring platforms, and commercial brand sentiment engines positioning it as a highly adaptable solution capable of influencing real-time, nationwide or global scale strategic decision making.

Additionally, future enhancements may also introduce federated learning and privacy-preserving techniques, allowing SMCA to operate across distributed data sources without compromising user confidentiality or violating data compliance regulations. Approaches such as differential privacy, encrypted vector embeddings, anonymized entity mapping, and on-device NLP inference can make SMCA suitable even for highly restricted or confidential research environments. This direction opens opportunities for collaborations with health, finance, defense, and public policy institutions where privacy and data security are essential, further expanding SMCA's scope as a trusted, compliant, and ethically governed social media intelligence framework.

VII. APPENDICES SOURCE CODE

twitter.py:

```
import os
import time
import logging
from datetime import datetime, timezone, timedelta
from typing import List, Dict
from dotenv import load_dotenv
import praw

load_dotenv()

client_id = os.getenv("REDDIT_CLIENT_ID")
client_secret = os.getenv("REDDIT_CLIENT_SECRET")
user_agent = os.getenv("REDDIT_USER_AGENT")

POST_LIMIT_PER_SUB = 5
MAX_RETRIES = 3
RATE_LIMIT_PER_MINUTE = 60
REQUEST_DELAY = 60 / RATE_LIMIT_PER_MINUTE

SUBREDDITS = [
  'forhire', 'slavelabour', 'freelance_forhire', 'designjobs', 'RemoteJobs',
  'jobs', 'WorkOnline', 'cscareerquestions', 'recruiting', 'ITCareerQuestions',
  'meetup', 'Networking', 'studygroup', 'EntrepreneurRideAlong', 'FindABand', 'FindACoFounder',
  'RealEstate', 'forrent', 'RealEstateTechnology', 'roommates', 'PropertyManagement',
  'hardwareswap', 'gamesale', 'IndieExchange', 'AVexchange', 'mechmarket', 'free',
  'influencermarketing', 'InstagramMarketing', 'YouTubeCollab', 'SocialMediaMarketing',
  'Assistance', 'helpme', 'Need', 'learnpython', 'Entrepreneur', 'startups', 'mentors', 'randomactsofkindness',
  'DealsReddit', 'BulkDeals',
  'Collaborate', 'FindABand', 'FindACoFounder', 'YouTubeCollab', 'ProgrammingBuddies', 'INAT',
  'reviews', 'RecommendMe', 'SuggestALaptop', 'BuyItForLife', 'AskReddit'
]

logger = logging.getLogger(__name__)
reddit = praw.Reddit(client_id=client_id, client_secret=client_secret, user_agent=user_agent)

def fetch_subreddit_posts(subreddit_name: str, retries: int = 0) -> List[Dict]:
  try:
    subreddit = reddit.subreddit(subreddit_name)
    posts = subreddit.new(limit=POST_LIMIT_PER_SUB)

    results = []
    for post in posts:
      permalink = getattr(post, "permalink", None)
      url = f"https://reddit.com{permalink}" if permalink else ""
      if not permalink:
        logger.warning(f"Post {post.id} missing permalink (may be deleted or removed)")
```

```
            results.append({
                "id": post.id,
                "user": {
                    "id": f"u/{post.author.name}" if post.author else "u/deleted",
                    "name": post.author.name if post.author else "deleted"
                },
                "title": post.title,
                "body": post.selftext,
                "text": f"{post.title}\n{post.selftext}",
                "created_at": datetime.fromtimestamp(post.created_utc, tz=timezone.utc).isoformat(),
                "url": url,
                "source": "Reddit API",
                "community": subreddit_name,
                "conversation_id": post.name,
                "in_reply_to": None
            })

        return results

    except Exception as e:
        if retries < MAX_RETRIES:
            time.sleep(2)
            return fetch_subreddit_posts(subreddit_name, retries + 1)
        else:
            logger.error(f"Failed to fetch r/{subreddit_name}: {e}")
            return []

def scrape_reddit() -> Dict:
    logger.info("Starting Reddit scrape...")
    raw_data = []
    request_count = 0

    for subreddit in SUBREDDITS:
        posts = fetch_subreddit_posts(subreddit)
        raw_data.extend(posts)
        request_count += 1

        if request_count % RATE_LIMIT_PER_MINUTE == 0:
            time.sleep(60)
        else:
            time.sleep(REQUEST_DELAY)

    result = {
        "platform": "reddit",
        "timestamp": datetime.now(tz=timezone.utc).isoformat(),
        "raw_data": raw_data,
        "metadata": {
            "request_id": f"req-rd-{datetime.now(tz=timezone.utc).strftime('%Y%m%d%H%M%S')}",
            "status": "success",
            "rate_limit": {
                "remaining": max(0, RATE_LIMIT_PER_MINUTE - (request_count % RATE_LIMIT_PER_MINUTE)),
                "reset_time": (datetime.now(tz=timezone.utc) + timedelta(minutes=1)).isoformat()
            }
        }
    }

    return result
```

x.py:

```
import json
from datetime import datetime, timezone
# Load your scraped Twitter data (replace with actual path if reading from file)
with open("twitter_scraped_data.json", "r", encoding="utf-8") as f:
    twitter_data = json.load(f)
standardized_data = []
for tweet in twitter_data:
    standardized_data.append({
        "id": tweet.get("id"),
        "user": {
            "id": "No_id",  # Apify output doesn't include user ID
            "name": tweet.get("url").split("/")[3] # Extract username from URL
        },
        "text": tweet.get("text"),
        "created_at": datetime.strptime(tweet["createdAt"], "%a %b %d %H:%M:%S %z %Y").astimezone(timezone.utc).isoformat(),
        "url": tweet.get("url"),
        "source": "Twitter Scraper (Apify)",
        "community": "Twitter",
```

```
        "conversation_id": None,  # Not present in output
        "in_reply_to": None       # Not present in output
    })
result = {
    "platform": "twitter",
    "timestamp":
datetime.now(timezone.utc).isoformat(),
    "raw_data": standardized_data,
    "metadata": {
        "request_id": f"req-tw-
{datetime.now(timezone.utc).strftime('%Y%m%d%
H%M%S')}",
        "status": "success",
        "rate_limit": {
            "remaining": "unknown",  # Apify handles rate
limits internally
            "reset_time": None
        }
    }
}
# Save to file
output_file = "twitter_posts_data.json"
with open(output_file, 'w', encoding='utf-8') as f:
    json.dump(result, f, indent=2, ensure_ascii=False)
print(f"Data saved to {output_file}")
```

preprocessor.py:

```
import json
import logging
from datetime import datetime
from tqdm import tqdm
from .cleaning import clean_text
from .utils import tokenize
from .filtering import should_remove,
classify_service_intent
logger = logging.getLogger(__name__)
class DataPreprocessor:
    def __init__(self):
        self.stats = {
            "total": 0,
            "processed": 0,
            "removed": 0,
            "errors": 0
        }
    def process(self, input_data: dict) -> dict:
        processed, removed = [], []
        raw_records = input_data.get("raw_data",
input_data.get("data", []))  # support reddit & twitter
        self.stats["total"] = len(raw_records)
        for record in tqdm(raw_records,
desc="Processing"):
            try:
                raw_title = record.get("title", "")
                raw_body = record.get("body", "")
                raw_text = record.get("text", "")
                # Clean title and body separately
                title_clean = clean_text(raw_title)
                body_clean = clean_text(raw_body)
                # Keep for UI
                record["title_clean"] = title_clean
                record["body_clean"] = body_clean
                # Combine for NLP logic as before
                text_clean = clean_text(raw_text)
                record["text_clean"] = text_clean

                # Use original or fallback timestamp
                post_ts = record.get("metadata",
{}).get("post_timestamp") or record.get("created_at")
                record["timestamp"] = post_ts or
datetime.utcnow().isoformat()
                # Filtering logic
                remove, reason = should_remove(record)
                if remove:
                    self._log_removal(removed, record,
reason)
                    continue
                record["language"] = "en"
                record["sentiment"] = 0  # placeholder
                record["tokens"] = tokenize(text_clean,
max_tokens=15)
                record["is_clean"] = True
                record["reason_kept"] = "service
seeker/provider"
                record["intent"] =
classify_service_intent(text_clean)
                processed.append(record)
                self.stats["processed"] += 1
            except Exception as e:
                self._log_error(removed, record, str(e))
        self.stats["removed"] = len(removed)
        logger.info(
            f"Processing complete:
total={self.stats['total']} "
            f"processed={self.stats['processed']}
removed={self.stats['removed']} "
            f"errors={self.stats['errors']}"
        )
        return {
```

```
        "platform": input_data.get("platform", "unknown"),
        "timestamp": input_data.get("timestamp", datetime.utcnow().isoformat()),
        "processed_data": processed,
        "removed_data": removed,
        "metadata": {
          **input_data.get("metadata", {}),
          "stats": self.stats
        }
      }
  def _log_removal(self, removed_list, record, reason):
      removed_list.append({
        "id": record.get("id", "NO_ID"),
        "reason": reason,
        "text": record.get("text", "")
      })
      self.stats["removed"] += 1
  def _log_error(self, removed_list, record, error_msg):
      record_id = record.get("id", "NO_ID")
      removed_list.append({
        "id": record_id,
        "reason": f"processing error: {error_msg}",
        "text": record.get("text", "")
      })
      self.stats["errors"] += 1
      logger.error(f"Error processing record ID {record_id}: {error_msg}")
```

filtering.py:

```
import re
from .config import (
  SPAM_KEYWORDS, MIN_WORDS, ACTIONABLE_PATTERNS,
IRRELEVANT_PATTERNS,
  SHORT_POST_ENTITY_PATTERNS
)
from .utils import normalize, is_english, has_actionable_entity
from sentence_transformers import CrossEncoder
# Load the cross-encoder model for intent classification
try:
  crossencoder_model = CrossEncoder('cross-encoder/nli-deberta-v3-small', device='cuda')
except Exception as e:
  print(f"Error loading cross-encoder model: {e}")
  crossencoder_model = None
ACTIONABLE_REGEXES = [re.compile(p, re.IGNORECASE) for p in ACTIONABLE_PATTERNS]
IRRELEVANT_REGEXES = [re.compile(p, re.IGNORECASE) for p in IRRELEVANT_PATTERNS]
ENTITY_REGEXES = [re.compile(p, re.IGNORECASE) for p in SHORT_POST_ENTITY_PATTERNS]
def is_spam(text: str) -> bool:
  text = normalize(text)
  return any(kw in text for kw in SPAM_KEYWORDS)
def is_actionable_regex(text: str) -> bool:
  return any(rgx.search(text) for rgx in ACTIONABLE_REGEXES)
def is_irrelevant(text: str) -> bool:
  return any(rgx.search(text) for rgx in IRRELEVANT_REGEXES)
def is_too_short(text: str) -> bool:
  words = text.split()
  return len(words) < MIN_WORDS
def should_keep_short(text: str) -> bool:
  # Keep if strong entity signal (price, phone, action word)
  return has_actionable_entity(text, SHORT_POST_ENTITY_PATTERNS)
def is_actionable_crossencoder(text: str) -> bool:
  if not crossencoder_model or not text.strip():
    return False
  try:
    hypothesis = (
      "This post is a direct, actionable request or offer related to services, jobs, events, real estate, secondhand goods, "
      "social commerce or influencer campaigns, community support, group buying, content collaboration, or reviews and recommendations. "
      "It is NOT advice, a tip, a meme, a joke, news, a rant, a random thought, general discussion, or a greeting."
    )
    pairs = [(text, hypothesis)]
    scores = crossencoder_model.predict(pairs) # returns np.ndarray
    return float(scores[0]) >= 0.7 # explicitly use first value
  except Exception as e:
```

```
        #print(f"Error in cross-encoder classification: {e}")
        return False

def classify_service_intent(text: str) -> str:
    if re.search(r"\b(offering|service offered|for hire|providing|can do|available for hire|expert in)\b", text):
        return "provider"
    elif re.search(r"\b(looking for|need a|seeking|service needed|wanted|require|help needed|searching for)\b", text):
        return "seeker"
    return "undefined"
def should_remove(record: dict) -> tuple[bool, str]:
    text = record.get("text_clean", "") or record.get("text", "")
    if not isinstance(text, str):
        return True, "text not a string"
    text = text.strip()
    if not text:
        return True, "empty text"
    if is_spam(text):
        return True, "spam detected"
    if not is_english(text):
        return True, "non-English"
    if is_irrelevant(text):
        return True, "irrelevant (advice/meme/rant/news/etc)"
    actionable = is_actionable_regex(text) or should_keep_short(text)
    if actionable:
        return False, ""
    if is_actionable_crossencoder(text):
        return False, ""
    return True, "not actionable"
```

extractor.py:

```
import logging
from datetime import datetime
from entity_extractor.config import (
    CONFIDENCE_THRESHOLD,
    SUPPORTED_ENTITY_TYPES,
    MAX_ENTITIES_PER_RECORD
)
from entity_extractor.rules import detect_intent, extract_entities
from entity_extractor.utils import validate_record
from entity_extractor.llm_batch import run_llm_batch

logger = logging.getLogger(__name__)

def filter_entities(entities):
    filtered = [e for e in entities if e["confidence"] >= CONFIDENCE_THRESHOLD]
    filtered = [e for e in filtered if e["type"] in SUPPORTED_ENTITY_TYPES]
    seen = set()
    deduped = []
    for e in filtered:
        key = (e["type"], e["value"].lower())
        if key not in seen:
            deduped.append(e)
            seen.add(key)
    return deduped[:MAX_ENTITIES_PER_RECORD]

class EntityExtractor:
    def process(self, input_data):
        processed, removed = [], []
        platform = input_data.get("platform")
        timestamp = input_data.get("timestamp")
        total = len(input_data.get("processed_data", []))

        records = input_data.get("processed_data", [])
        batch_size = 5  # Adjust as needed

        logger.info(f"Starting extraction: total records={total}")

        for i in range(0, total, batch_size):
            batch_records = records[i:i+batch_size]
            batch_texts = [r.get("text_clean", "") for r in batch_records]

            try:
                logger.info(f"Attempting LLM batch extraction for records {i}-{i + len(batch_records) - 1}...")
                batch_results = run_llm_batch(batch_texts)
                logger.info(f"LLM batch extraction succeeded for records {i}-{i + len(batch_records) - 1}.")

                for record, llm_result in zip(batch_records, batch_results):
                    intent = llm_result.get("intent")
                    entities = llm_result.get("entities", [])
                    entities = filter_entities(entities)
```

```
            if not validate_record(intent, entities):
                removed.append({
                    "id": record.get("id"),
                    "reason": "No intent or entities found",
                    "text": record.get("text_clean", "")
                })
                continue

            processed.append({
                "record_id": record.get("id"),
                "user_id": record.get("user", {}).get("id"),
                "user_name": record.get("user", {}).get("name"),
                "text_clean": record.get("text_clean", ""),
                "title_clean": record.get("title_clean", ""),
                "body_clean": record.get("body_clean", ""),
                "intent": intent,
                "entities": entities,
                "timestamp": record.get("timestamp"),
                "metadata": {
                    "platform": platform or "unknown",
                    "url": record.get("url") or "",
                    "community": record.get("community") or "",
                    "language": record.get("language") or "en",
                    "extracted_at": datetime.utcnow().isoformat() + "Z"
                },
                "confidence": min([e["confidence"] for e in entities], default=1.0)
            })

        except Exception as e:
            logger.error(f"LLM batch extraction failed at records {i}-{i + len(batch_records) - 1}: {e}")
            logger.info(f"Falling back to rule-based extraction for records {i}-{i + len(batch_records) - 1}.")

            # Fallback: process each record one by one
            for record in batch_records:
                try:
                    text = record.get("text_clean", "")
                    intent = detect_intent(text)
                    entities = extract_entities(text)
                    entities = filter_entities(entities)

                    if not validate_record(intent, entities):
                        removed.append({
                            "id": record.get("id"),
                            "reason": "No intent or entities found",
                            "text": text
                        })
                        continue

                    processed.append({
                        "record_id": record.get("id"),
                        "user_id": record.get("user", {}).get("id"),
                        "user_name": record.get("user", {}).get("name"),
                        "text_clean": text,
                        "title_clean": record.get("title_clean", ""),
                        "body_clean": record.get("body_clean", ""),
                        "intent": intent,
                        "entities": entities,
                        "timestamp": record.get("timestamp"),
                        "metadata": {
                            "platform": platform or "unknown",
                            "url": record.get("url") or "",
                            "community": record.get("community") or "",
                            "language": record.get("language") or "en",
                            "extracted_at": datetime.utcnow().isoformat() + "Z"
                        },
                        "confidence": min([e["confidence"] for e in entities], default=1.0)
                    })

                except Exception as inner_e:
                    removed.append({
                        "id": record.get("id"),
                        "reason": f"extraction error: {inner_e}",
                        "text": record.get("text_clean", "")
```

```
            })
            logger.error(f"Error processing record
ID {record.get('id')}: {inner_e}")

    logger.info(f"Extraction complete: total={total}
processed={len(processed)}
removed={len(removed)}")
    return {
        "platform": platform,
        "timestamp": timestamp,
        "extracted_data": processed,
        "removed_data": removed
    }
```

VIII. ACKNOWLEDGEMENT

It is one of the most efficient tasks in life to choose the appropriate words to express one's gratitude to the beneficiaries. We are very much grateful to God who helped us all the way through the project and how molded us into what we are today.

We are grateful to our beloved Principal Dr. R. RADHAKRISHNAN, M.E., Ph.D., Adhiyamaan College of Engineering (An Autonomous Institution), Hosur for providing the opportunity to do this work in premises.

We acknowledge our heartfelt gratitude to Dr. G. FATHIMA, M.E., Ph.D., Professor and Head of the Department, Department of Computer Science and Engineering, Adhiyamaan College of Engineering (An Autonomous Institution), Hosur, for her guidance and valuable suggestions and encouragement throughout this project and made us to complete this project successfully.

We are highly indebted to Mrs. D.M. VIJAYALAKSHMI, M.E., Supervisor, Assistant Professor, Department of Computer Science and Engineering, Adhiyamaan College of Engineering (An Autonomous Institution), Hosur, whose immense support, encouragement and valuable guidance were responsible for completing the project successfully.

We also extend our thanks to the Project Coordinator and all Staff Members for their support in completing this project successfully.

Finally, we would like to thank our parents, without their motivation and support it would not have been possible for us to complete this project successfully.

REFERENCES

[1] Hinton, Alexandra, and Tania Roy. "Understanding Multi-platform Social Media Aggregators: A Design and Development Case Study with BTS-DASH." International Conference on Human-Computer Interaction. Cham: Springer Nature Switzerland, 2024.

[2] Fletcher, R., Kalogeropoulos, A., & Nielsen, R. K. (2023). More diverse, more politically varied: How social media, search engines and aggregators shape news repertoires in the United Kingdom. New Media & Society, 25(8), 2118-2139.

[3] Hlaoua, L. (2025). An overview of aggregation methods for social networks analysis. Knowledge and Information Systems, 67(1), 1-28.

[4] Bhattacharyya, Mousumi, et al. "An emoticon-based sentiment aggregation on metaverse related tweets." The international conference on artificial intelligence and computer vision. Cham: Springer Nature Switzerland, 2023.

[5] Kameda, Tatsuya, Wataru Toyokawa, and R. Scott Tindale. "Information aggregation and collective intelligence beyond the wisdom of crowds." Nature Reviews Psychology 1.6 (2022): 345-357.

[6] Beer, David. "Using social media data aggregators to do social research." Sociological Research Online 17.3 (2012): 91-102.

[7] Zignani, Matteo, et al. "Walls-in-one: usage and temporal patterns in a social media aggregator." Applied Network Science 1.1 (2016): 5.

[8] Lande, Dmytro, Igor Subach, and Alexander Puchkov. "A system for analysis of big data from social media." Information & Security 47.1 (2020): 44-61.

[9] Srikanth, N., Tejaswini, C. V., & Kumar, D. P. (2019). Socially smart: An aggregation system for social media using web scraping. Displays, 6(4), 749-752.

[10] De Corniere, A., & Sarvary, M. (2023). Social media and news: Content bundling and news quality. Management Science, 69(1), 162-178.