

MindTrack Journal: A Privacy-First Intelligent Journaling System with Emotion Detection

Syed Sadiqu Hussain

Student, Dept. of Computer Science & Engineering
GIFT Autonomous College, Bhubaneswar, Odisha, India

Sharqua Jabeen

Student, Dept. of Computer Science & Engineering
GIFT Autonomous College, Bhubaneswar, Odisha, India

Asst. Prof. Priti Manjari Barik

Assistant Professor, Dept. of Computer Science & Engineering
GIFT Autonomous College, Bhubaneswar, Odisha, India

Abstract—MindTrack Journal is a secure, privacy-first, and intelligent journaling application designed to empower users to express their thoughts while maintaining complete control over their personal data. In a world where data privacy is increasingly important, MindTrack Journal stands out by ensuring that all user data remains private, encrypted, and never shared with third-party services. The application provides a seamless and user-friendly experience for managing personal journals, including core features such as user registration, login, and profile management, along with a rich block-based text editor supporting multiple content types for a more expressive writing experience. What differentiates MindTrack Journal from traditional journaling platforms is its intelligent analytics layer powered by a self-hosted AI service. Each journal entry is automatically analysed for sentiment using state-of-the-art transformer-based Natural Language Processing (NLP) models—specifically, `finiteautomata/bertweet-base-sentiment-analysis` for sentiment classification and `bhadrash-savani/bert-base-uncased-emotion` for emotion detection—ensuring that sensitive user data is never transmitted to external services. The results are visualised through an analytics dashboard, offering users meaningful insights into their emotional patterns over time. Security and privacy form the core pillars of the system, implemented via JWT-based stateless sessions validated by Keycloak, BCrypt password hashing, HTTPS/TLS encryption, database encryption at rest, and application-level journal content encryption. The architecture follows the Citadel Architecture pattern, combining Spring Boot 4.x Spring Modulith for core domain workflows with supporting microservices for analytics, payments, notifications, and ML processing. The system is implemented as a full-stack monorepo comprising a React 19/TypeScript frontend, a Java 21/Spring Boot 4.x backend, and a Python/FastAPI ML service, all containerised using Docker and orchestrated via Docker Compose, enabling repeatable deployment with a single command.

Keywords—Emotion detection; Sentiment analysis; Privacy-by-design; Microservices; BERT; Apache Kafka; Spring Modulith; Digital mental health; NLP; Journaling application

I. INTRODUCTION

In today's digital world, people increasingly rely on online platforms to document personal thoughts, emotions, and daily experiences. However, most existing journaling applications raise serious concerns about privacy, data ownership, and the quality of insights offered to users. Individuals often feel uncomfortable sharing their most intimate thoughts on platforms that collect data for commercial purposes, expose content to third-party AI services for remote processing, or provide no meaningful analytical feedback in return.

Journaling, as a practice, has been used for centuries as a powerful tool for self-expression, personal development, and emotional processing. From ancient diaries and personal letters to modern digital platforms, journaling has evolved significantly, adapting to the needs and lifestyles of each generation. Research by Pennebaker [1] demonstrated that structured emotional writing leads to measurable improvements in both mental and physical health, proposing that translating emotional experiences into language imposes structure that facilitates psychological resolution and reduces rumination. A 2018 meta-analysis [3] of 64 randomised controlled trials further confirmed statistically significant reductions in depression, anxiety, and psychological distress through regular journaling.

With the advancement of technology, digital journaling platforms have introduced new possibilities including searchability, cloud storage, reminders, and analytics. Smart journaling applications take this further by incorporating artificial intelligence (AI) and data analytics to provide users with deeper insights into their emotional patterns and personal growth trajectories. However, this benefit has

historically come at the cost of privacy, as AI processing is typically offloaded to third-party cloud APIs.

MindTrack Journal is motivated by the idea of creating a safe, secure, and intelligent space where users can freely express themselves without fear of data exposure. The primary goal is to combine privacy-first design with intelligent emotional analysis, giving users strong security guarantees alongside meaningful feedback on their journaling habits and emotional well-being. A critical differentiator is that each journal entry is analysed using a self-hosted AI service, meaning sensitive user data never leaves the user's own infrastructure.

From a software engineering perspective, the project explores the Citadel Architecture—a novel hybrid pattern combining Spring Modulith for core bounded contexts with standalone microservices for peripheral capabilities such as analytics, payments, notifications, and ML processing. This provides the performance advantages of a modular monolith for the core domain, combined with the independent deployability and scalability of microservices where they are most needed.

The remainder of this paper is organised as follows. Section II reviews related literature. Section III presents the complete system design and methodology. Section IV discusses implementation details. Section V presents results and discussion. Section VI covers the REST API surface. Section VII concludes with directions for future work.

II. LITERATURE REVIEW

A. Expressive Writing and Mental Health

The foundational research in expressive writing was established by Pennebaker [1], whose landmark studies in the 1980s demonstrated that structured emotional writing leads to measurable improvements in both mental and physical health. Participants who wrote about traumatic or emotionally significant experiences showed reduced anxiety, improved immune function, and lower rates of depression compared to control groups. His inhibition theory proposes that suppressing emotional experiences requires ongoing cognitive and physiological effort, and that translating those experiences into language reduces this burden by imposing narrative structure.

Subsequent meta-analyses have consistently confirmed these findings. Smyth [2] found that written emotional expression produced significant health improvements with effect sizes comparable to other psychological interventions. Beyond mental health outcomes, regular reflective writing strengthens memory consolidation, improves goal-setting behaviour, and enhances creative thinking [4]. The process of externalising internal states into structured text appears to activate metacognitive processes that improve self-regulation and decision-making capacity.

A recurring theme in the journaling literature is the importance of longitudinal reflection—the practice of reviewing past entries to identify patterns and changes over time. Unlike one-time therapeutic interventions, long-term journaling creates a personal record that can reveal the evolution of thought patterns and emotional states in ways that are invisible to real-time perception. Research on habit formation, drawing on Duhigg's cue-routine-reward framework [20], suggests that journaling applications can leverage design principles to improve adherence. Features such as streak tracking, writing reminders, and visual progress indicators increase consistency by activating intrinsic motivation and providing positive reinforcement. MindTrack Journal implements writing streak tracking and a mood calendar as habit-reinforcing mechanisms within the analytics dashboard.

B. AI-Powered Journaling Systems

Recent advancements in AI-assisted journaling demonstrate improvements in positive emotions, reduced anxiety, and increased self-awareness among users who receive automated feedback or insights. These systems leverage Natural Language Processing (NLP) to analyse text and provide real-time emotional summaries, creating a personalised journaling experience that goes beyond simple text recording.

The introduction of transformer-based language models—particularly BERT and its successors [6]—dramatically improved sentiment analysis accuracy over earlier lexicon-based methods. Models fine-tuned on domain-specific corpora outperform lexicon-based approaches on contextual disambiguation, negation handling, and subtle emotional distinctions. The BERTweet model [7] used in MindTrack Journal is specifically fine-tuned on social media text, which shares stylistic characteristics with informal journal writing, including abbreviated language, emotional directness, and colloquial expression.

A critical concern in AI-powered journaling is the privacy risk of transmitting personal diary content to cloud-based NLP APIs. Commercial providers may process submitted text and retain it for model improvement. Self-hosted NLP models, as implemented in MindTrack Journal, eliminate this risk entirely, aligning with the privacy-by-design principle [5] and representing a technically mature approach to private AI analytics.

C. Privacy and Security in Digital Health

Digital health applications handle some of the most sensitive personal data categories: mental health records, emotional states, and daily thought patterns. The principle of privacy by design [5], formalised by Ann Cavoukian and later codified in GDPR Article 25, mandates that privacy protection be built into systems from inception rather than added as an afterthought. Key tenets include data

minimisation, purpose limitation, storage limitation, and full user control over data.

MindTrack Journal operationalises these principles through architecture-level enforcement: journal content is encrypted before storage, analytics are computed from aggregates rather than raw content, and users can exercise the right to be forgotten via permanent account deletion that cascades across all services and databases. The OAuth2 and OpenID Connect (OIDC) authentication frameworks provide significant security advantages by delegating identity management to a dedicated identity provider such as Keycloak [12], allowing MindTrack Journal to benefit from enterprise-grade security including brute-force protection, session management, and cryptographically signed JWT tokens without implementing these mechanisms from scratch.

D. *Microservices and Event-Driven Architecture*

The microservices architecture pattern [9], widely adopted since the early 2010s, decomposes applications into small, independently deployable services communicating over well-defined APIs. Core benefits include independent scalability, fault isolation, technology heterogeneity, and team autonomy. However, it also introduces distributed system complexity, network latency, eventual consistency challenges, and significant operational overhead.

Spring Modulith [14] addresses a known criticism of microservices—premature decomposition—by providing a modular monolith option where bounded contexts [10] are strongly isolated at the code level but share a process for low-latency communication. The MindTrack Journal Citadel Architecture applies this insight by placing the user and journal domains inside a shared Spring Modulith process while keeping analytics, payments, notifications, and ML as genuine independent microservices.

Apache Kafka [11], as an event streaming platform, is well-established for event-driven architectures requiring durability, replayability, and decoupling. Its use in MindTrack Journal for the asynchronous sentiment analysis pipeline follows established patterns for ML inference in production systems where user-facing response latency must not be coupled to model inference time.

III. SYSTEM DESIGN AND METHODOLOGY

A. *Proposed Solution*

MindTrack Journal proposes a privacy-first journaling environment where users can freely express their thoughts without worrying about data exposure. Unlike existing platforms that rely on third-party AI services, all data is processed locally using self-hosted transformer models, ensuring sensitive journal content is never transmitted externally. The system introduces an intelligent analysis layer that transforms simple journaling into a rich, insightful experience, while simultaneously supporting full data

ownership and user control—including the right to permanent account deletion that removes all associated records across all services.

The platform targets individuals seeking a private and secure journaling experience, mental health enthusiasts, students, professionals, and potentially parents or guardians who wish to monitor emotional trends for supportive purposes. Three subscription tiers (Free, Pro, Enterprise) provide configurable feature gates to accommodate different user needs and usage scales.

B. *The Citadel Architecture*

The backend adopts the Citadel Architecture pattern: a hybrid design combining a Spring Modulith for the core domain with specialised microservices for peripheral capabilities. The Core Service (port 8081) is a Spring Modulith application with two bounded contexts enforced by Java package structure:

- **Journal Context:** Handles journal page and block management, sentiment integration, tag management, permissions, and AI enrichment.
- **User Context:** Handles user registration, authentication, profile management, tenant management, session tracking, and subscription plan administration.

These contexts communicate exclusively through Spring Application Events, ensuring loose coupling even within the shared JVM process. Supporting microservices include the Analytics Service (port 8083) for precomputed sentiment aggregation, Payment Service (port 8084) for subscription management, Notification Service (port 9090) for event-driven email dispatch, API Gateway (port 2000) for routing and JWT validation, Config Server (port 8888) for centralised YAML configuration, and Eureka Server (port 8761) for service registration and dynamic discovery.

C. *Selected Technologies*

1) *Frontend*

React 19.2.0 with TypeScript 5.8.3 and Vite 7.3.1 as the build tool. TanStack Router v1 provides type-safe file-based routing. TanStack Query v5 manages server state with automatic caching. Tailwind CSS v4 with the Shadcn UI component library (built on Radix UI primitives) provides an accessible component system. Recharts handles data visualisation on the analytics dashboard.

2) *Backend*

Java 21 with Spring Boot 4.x across all services. Spring Modulith 2.0.3 enforces bounded context isolation at compile time. Spring Cloud Netflix Eureka manages service registration. Spring Cloud OpenFeign implements declarative HTTP clients. Project Loom virtual threads provide efficient I/O concurrency without traditional thread pool overhead.

3) *Database and Storage*

PostgreSQL 17 with separate logical databases per service enforcing strict data isolation. JSONB columns accommodate flexible structured data including block content and emotion probability vectors. Redis 8.4 is provisioned for future caching. Apache Kafka 8.1.2 in KRaft mode manages the asynchronous sentiment pipeline via `journal.page.analysis.request` and `journal.page.analysis.result` topics.

4) Security

Keycloak 26.4.7 for OAuth2/OIDC identity management; BCrypt password hashing; HTTP-only SameSite-configured cookies to prevent XSS-based token theft; strict CORS policies; Content-Security-Policy and HSTS headers; rate limiting on sensitive endpoints; and application-level AES encryption of journal content before database writes.

5) AI and Sentiment Analysis

A Python 3.12 FastAPI service with HuggingFace Transformers and PyTorch hosts two self-hosted BERT-based models loaded with `TRANSFORMERS_OFFLINE=1`, guaranteeing no external network calls and ensuring complete data sovereignty:

- **finiteautomata/bertweet-base-sentiment-analysis:** BERTweet fine-tuned for three-class polarity classification (POS/NEG/NEU).
- **bhadresh-savani/bert-base-uncased-emotion:** BERT fine-tuned for six-class emotion detection (joy, sadness, anger, fear, disgust, surprise), returning a full probability vector.

D. System Workflow

The workflow begins with Keycloak-verified authentication. JWT access and refresh tokens are issued as HTTP-only cookies. When a user saves a journal entry, the Core Service sanitises input via the OWASP HTML Sanitizer [19] to prevent stored XSS vulnerabilities, encrypts the content at the application level, and persists it to PostgreSQL, acknowledging the write with a 201 Created response immediately.

Post-save, a Spring Application Event publishes a Kafka message containing the page ID and extracted block text to the sentiment request topic. The Python ML Service consumes this message, runs both transformer models, and publishes the combined result back to the response topic. The Core Service stores the sentiment label, confidence scores, dominant emotion, and full emotion probability vector. The Analytics Service provides precomputed dashboard data refreshed every six hours, ensuring sub-100-millisecond dashboard response times.

E. Database Architecture

The `user_service` database contains: the `users` table storing identity data and Keycloak UUID mapping; the `tenants` table representing multi-tenant workspaces; the `tenant_members` junction table with OWNER/ADMIN/MEMBER roles; the

`users_sessions` table for active session tracking; and the `users_settings` table storing per-user preferences as JSONB. All tables extend a `BaseEntity` pattern with UUID primary keys and full audit fields.

The `journal_service` database implements the block-based content model. The `journal_pages` table is the aggregate root. The `journal_blocks` table uses a self-referential `parent_block_id` for nested hierarchies with JSONB block content. The `sentiment_analysis` table stores ML output including sentiment label, confidence score, and emotion vector. The `ai_enrichment` table stores higher-level AI analysis. The `block_versions` table provides a complete audit trail for future undo/redox support.

The payment service database implements billing infrastructure: the `plans` table with JSONB feature flags; the `subscriptions` table for active subscription tracking; the `transactions` table recording all payment events; the `invoices` table; and the `payment_methods` table storing masked payment details for recurring billing.

IV. IMPLEMENTATION

A. Project Structure and Monorepo Organisation

MindTrack Journal is organised as a monorepo with three top-level directories: `backend/` containing all Java Spring Boot services, `frontend/` containing the React/TypeScript application, and `ml/` containing the Python ML service. Each backend service follows a standard Spring Boot 4.x project structure with `application.yaml` configuration, a Dockerfile for containerisation, and `pom.xml` for Maven build configuration. The Core Service uses `package-info.java` files in each bounded context package to declare Spring Modulith module boundaries explicitly.

B. User Bounded Context

User registration follows a two-phase process to maintain consistency between the identity provider and application database. First, the user record is created in Keycloak via the Admin REST API using `KeycloakService`. If successful, the local `User`, `Tenant`, `TenantMember`, `UserSetting`, and `UserSession` records are created within a single database transaction. A compensating Keycloak deletion is attempted if local persistence fails, maintaining system consistency.

Login delegates credential verification to Keycloak's token endpoint. The `TenantMembershipService` interface defines the cross-context API that the journal context uses to verify tenant membership without violating Spring Modulith module boundaries. Plan management is handled by `PlanAdminService` for CRUD operations (protected by `SYS_ADMIN` role) and `PlanService` for read-only access, with a `PlanDataSeeder` component ensuring that `FREE`, `PRO`, and `ENTERPRISE` plan records exist on every application startup.

C. Journal Bounded Context

Journal page creation in `JournalPageServiceImpl` performs the following operations in sequence: validate tenant membership via the cross-module interface; create and persist the `JournalPage` entity; create and persist `JournalBlock` entities in order if blocks are included; associate tags if tag IDs are provided; publish a `BlockCreatedEvent` for each block to trigger asynchronous sentiment analysis; and return the full `JournalPageDetailResponse`.

Block text extraction is handled by `BlockTextUtil`, which concatenates text content from all `TEXT`, `HEADING`, and `QUOTE` blocks into a single string, ensuring sentiment analysis reflects the full narrative of the entry. Block reordering accepts an ordered list of block IDs and bulk-updates `order_index` values within a single transaction to prevent partial states. Input sanitisation via `SanitizerUtil` wraps an OWASP HTML Sanitizer pipeline as a Spring-managed singleton, preventing stored XSS vulnerabilities at the persistence layer.

D. Sentiment Analyzer ML Service

The Python 3.12 FastAPI ML service loads both transformer models at startup via a lifespan context manager. Model validation checks for the presence of `config.json` and weight files before loading. The `analyze_sentiment()` function runs `BERTweet` with `top_k=None` to retrieve all label scores, mapping `POS/NEG/NEU` to `POSITIVE/NEGATIVE/NEUTRAL`. The `analyze_emotion()` function retrieves a full probability vector across all six emotion categories.

Inference runs on a `ThreadPoolExecutor` with two workers, allowing the `asyncio` event loop to continue processing Kafka messages during model inference. A `/health` endpoint reports both model load status and Kafka consumer task status for Docker Compose health check integration, ensuring the Core Service only begins sending analysis requests after full service operability is confirmed.

E. Analytics Service Implementation

The Analytics Service connects to its own analytics service PostgreSQL database and makes Feign client calls to the Core Service for raw journal page data. Key methods include `getSentimentTrends()` for time-series sentiment aggregation, `getMoodCalendar()` for per-day entries, `getEmotionDistribution()` for frequency percentages, and `getWritingStreak()` for consecutive-day sequences. The `AnalyticsScheduler` runs a scheduled task every six hours to update precomputed cache tables, keeping dashboard response times under 100 ms.

F. API Gateway Configuration

The API Gateway (Spring Cloud Gateway, port 2000) serves as the single entry point for all client requests. Routes are configured using the Spring Cloud Gateway Java DSL, with service addresses resolved through the Eureka Server

for dynamic discovery. The gateway is configured as an OAuth2 Resource Server validating JWTs via the Keycloak JWKS URI. Public paths such as login, register, and health endpoints are permitted without authentication, while all other paths require a valid JWT.

G. Containerisation and Deployment

Java services use multi-stage Dockerfiles: a Maven build stage produces the fat JAR, followed by a JRE 21 slim runtime stage for a minimal production image. The Python ML service volume-mounts model weights from the host filesystem, keeping image sizes small and permitting model updates without rebuilding. The `docker-compose.yml` [18] defines health checks for all twelve services with appropriate `start_period` values, enforcing startup ordering via condition: `service_healthy`. Named volumes provide persistent storage for PostgreSQL, Redis, and Kafka across container restarts.

V. RESULTS AND DISCUSSION

A. System Implementation Results

The user authentication module delivers fully functional OAuth2-based authentication via Keycloak 26.4.7. The dual-phase registration process with compensating rollback ensures consistency between the identity provider and application database. JWT-based stateless authentication validates tokens at the gateway level without requiring session storage in backend services, supporting horizontal scalability.

The journal management system enables the full CRUD lifecycle for journal pages and content blocks. The block-based editor model supports six content types (`TEXT`, `HEADING`, `QUOTE`, `IMAGE`, `VIDEO`, `AUDIO`) with JSONB-based flexible storage. Block reordering, nesting, soft deletion, and version history tracking are all functional. Input sanitisation at the persistence layer prevents stored XSS vulnerabilities regardless of client-side behaviour.

The sentiment analysis pipeline successfully integrates the Python FastAPI ML service via Kafka. Journal pages are analysed asynchronously without impacting user-facing save latency: users receive a 201 Created response within the 200 ms target while `BERTweet` inference on CPU hardware averages 200–800 ms per page. Both transformer models load correctly from local directories and produce accurate classification results. The full emotion probability vector is stored in the `sentiment_analysis` table and returned as part of journal page detail responses, enabling future granular analytics features such as emotion-based search and mood-aware content recommendations.

The analytics service provides all six endpoints with functional implementations: summary statistics, sentiment trends at three time granularities (daily, weekly, monthly), mood calendar, emotion distribution, and writing streak. The scheduled cache refresh mechanism ensures dashboard data remains current with response times consistently under 100

ms. The Docker Compose stack brings up all twelve services with proper health-check-based startup ordering, enabling the entire platform to deploy with a single command.

B. Discussion and Analysis

The Citadel Architecture proved highly effective for this domain. The Spring Modulith approach provides the performance benefits of co-located services for the tightly coupled user and journal domains, while dedicated microservices for analytics and ML maintain clean separation of concerns and enable independent scaling of compute-intensive workloads. The ModulithArchitectureTest verifies at compile time that module boundaries are respected, preventing accidental coupling as the codebase evolves.

The asynchronous sentiment pipeline is a particularly important design decision. By decoupling analysis latency from write latency via Kafka, the system provides a responsive journaling experience regardless of ML model inference time. The TRANSFORMERS_OFFLINE=1 environment variable provides an additional data sovereignty guarantee beyond architectural design, ensuring no inadvertent network calls to external APIs.

The multi-tenant data model provides a solid foundation for future collaborative features. Per-request tenant isolation enforcement at the repository layer delivers both security and extensibility. The subscription plan system provides a complete tiered feature-gating architecture ready for commercial deployment, with plan enforcement applied at the service layer before any business logic executes.

C. Real-Life Impact and Applications

MindTrack Journal has significant real-life applications in personal development and mental well-being. It provides users with a safe and private space to express their thoughts and emotions, free from the risk of data exposure or commercial surveillance. For students and professionals, the platform helps manage stress, track productivity, and improve emotional balance through regular journaling and visual trend analysis.

The system can be valuable in mental health awareness contexts, where individuals monitor their emotional state over time. While not a replacement for professional therapy, it serves as a supportive self-reflection tool that encourages emotional awareness. From a technological perspective, the project demonstrates how transformer-based AI, secure authentication, event-driven messaging, and data visualisation can be combined to create impactful, privacy-preserving applications.

D. Limitations

The primary limitation of the current ML models is their training data domain. BERTweet is fine-tuned on social media text (tweets), which shares some characteristics with journal writing but differs in formality and length. A model

fine-tuned specifically on journaling text corpora would yield meaningfully better accuracy on the emotional nuances of reflective writing. Additionally, the current architecture runs models on CPU hardware, limiting throughput for high concurrent usage. GPU-accelerated inference or model quantisation via ONNX Runtime would significantly improve performance at production scale. The payment service currently provides the data model and API scaffolding without full payment gateway integration, representing a pragmatic scoping decision for this project phase.

VI. API DOCUMENTATION OVERVIEW

All endpoints are served through the API Gateway and return a standard BaseResponse<T> JSON envelope with success, message, and data fields. Common HTTP status codes used throughout the API are 200 OK, 201 Created, 400 Bad Request, 401 Unauthorised, 403 Forbidden, 404 Not Found, and 409 Conflict.

A. Authentication and User Endpoints

The authentication surface includes public endpoints for POST /api/v1/auth/register (new user account creation with two-phase Keycloak registration), POST /api/v1/auth/login (credential verification returning JWT tokens in both response body and HTTP-only cookies), and POST /api/v1/auth/refresh-token. Protected endpoints include DELETE /api/v1/auth/delete/{id} (permanent account deletion cascading across all services, requiring ADMIN role) and POST /api/v1/admin/auth/register (administrator account registration requiring SYS_ADMIN role). User profile management is provided via GET /api/v1/users/me (retrieve current user profile including default tenant details) and PATCH /api/v1/users/me (partial update of profile fields).

B. Journal Pages and Blocks Endpoints

Journal page operations include POST /api/v1/journals/pages (create page with optional inline blocks, accepting tenantId, title, entryDate, privacy flag, tagIds, and blocks array), GET /api/v1/journals/pages (paginated list with date range filtering supporting from, to, page, size, and sort parameters), GET /api/v1/journals/pages/{pageId} (full detail including all blocks, sentiment analysis results, AI enrichment data, and associated tags), and PUT /api/v1/journals/pages/{pageId} for metadata updates. Tag association is managed via POST and DELETE on /api/v1/journals/pages/{pageId}/tags/{tagId}. Manual sentiment re-analysis can be triggered via POST /api/v1/journals/pages/{pageId}/analyze.

Block content structure varies by type: TEXT, QUOTE, and HEADING blocks use {"text": "content"}, while IMAGE, VIDEO, and AUDIO blocks use {"url": "media_url", "caption": "optional"}. The reorder endpoint

accepts a blockIds array in the desired display order and bulk-updates order_index values within a single transaction.

C. Analytics Endpoints

The analytics surface provides:

- **GET /api/v1/analytics/summary:** Overall statistics including total entries, average sentiment score, positive/negative/ neutral counts, most frequent emotion, current streak, and longest streak.
- **GET /api/v1/analytics/sentiment/trends:** Time-series sentiment data supporting period parameter (DAILY, WEEKLY, or MONTHLY) with optional from/to date filters.
- **GET /api/v1/analytics/mood/calendar:** Per-day sentiment indicators for a given year and month, enabling the mood calendar visualisation.
- **GET /api/v1/analytics/emotions/distribution:** Emotion frequency breakdown as an array of {emotion, count, percentage} objects sorted by count descending.
- **GET /api/v1/analytics/writing/streak:** Current and longest writing streak counts.
- **POST /api/v1/analytics/refresh:** Force recomputation of all analytics for the current user.

VII. CONCLUSION

MindTrack Journal successfully demonstrates the development of a secure, intelligent, and user-centric digital journaling application that addresses key challenges in privacy, usability, and meaningful data analysis. The system integrates a privacy-first design philosophy with modern microservices architecture and AI-driven sentiment analysis to deliver a comprehensive journaling experience that stands apart from existing commercial platforms.

Multiple important software engineering concepts were integrated into a single cohesive system: secure OAuth2 authentication with Keycloak, multi-tenant data isolation, event-driven architecture with Apache Kafka, self-hosted transformer-based NLP, block-based content modelling with JSONB storage, and Docker-based containerisation and orchestration. The modular microservices architecture ensures each component can be independently scaled, maintained, and enhanced without disrupting the overall system. The results confirm that the system performs reliably, offering users a safe platform to express their thoughts and gain meaningful insights into their emotional patterns while guaranteeing complete data sovereignty.

Several significant opportunities exist to enhance and expand the project. The most impactful improvement would be integration of AI models fine-tuned specifically on journaling text corpora, supplemented by large language model (LLM) capabilities for generating journal summaries, identifying recurring themes, and providing mood-aware recommendations. Voice-based journaling—where users

record thoughts through speech and the system converts audio to text for analysis—would improve accessibility, with the existing AUDIO block type already providing the structural foundation.

From a technical perspective, completing the Payment Service integration with Stripe for international markets and Shariah-compliant providers for MENA markets would enable commercial deployment. GPU-accelerated model inference, ONNX Runtime quantisation, and Kubernetes migration for production orchestration would further improve performance and operational maturity. Mobile applications for Android and iOS, integration with wearable health APIs such as Apple HealthKit and Google Fit for multi-modal well-being insights, and a Notion import tool to reduce adoption barriers are all identified as high-value future enhancements.

REFERENCES

- [1] J. W. Pennebaker and S. K. Beall, "Confronting a traumatic event: Toward an understanding of inhibition and disease," *Journal of Abnormal Psychology*, vol. 95, no. 3, pp. 274–281, 1986.
- [2] J. M. Smyth, "Written emotional expression: Effect sizes, outcome types, and moderating variables," *Journal of Consulting and Clinical Psychology*, vol. 66, no. 1, pp. 174–184, 1998.
- [3] P. G. Frisina et al., "A meta-analysis of the effects of written emotional disclosure on the health outcomes of clinical populations," *Journal of Nervous and Mental Disease*, vol. 192, no. 9, pp. 629–634, 2004.
- [4] J. W. Pennebaker and C. K. Chung, "Expressive writing: Connections to physical and mental health," in *The Oxford Handbook of Health Psychology*, Oxford University Press, 2011.
- [5] A. Cavoukian, "Privacy by design: The 7 foundational principles," Information & Privacy Commissioner of Ontario, Canada, 2009.
- [6] J. Devlin et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, pp. 4171–4186, 2019.
- [7] J. M. Pérez et al., "Pysentimiento: A Python toolkit for sentiment analysis and SocialNLP tasks," *arXiv:2106.09462*, 2021.
- [8] B. Savani, "BERT-base-uncased-emotion," *HuggingFace Model Hub*, 2020. [Online]. Available: <https://huggingface.co/bhadresh-savani/bert-base-uncased-emotion>
- [9] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, 2nd ed. O'Reilly Media, 2021.
- [10] V. Vernon, *Domain-Driven Design Distilled*. Addison-Wesley Professional, 2016.
- [11] J. Kreps et al., "Kafka: A distributed messaging system for log processing," in *Proc. NetDB Workshop*, 2011.
- [12] Keycloak Project, "Keycloak 26.4.7 Server Administration Guide," Red Hat, 2024.
- [13] VMware/Broadcom, "Spring Boot 4.x Reference Documentation," 2025.
- [14] VMware/Broadcom, "Spring Modulith 2.0.3 Reference Guide," 2025.

-
- [15] Meta Open Source, "React 19 Release Notes and API Reference," 2025. [Online]. Available: <https://react.dev/>
 - [16] HuggingFace, "Transformers: State-of-the-art machine learning for PyTorch, TensorFlow, and JAX," 2024.
 - [17] Babburi, S. Lightweight Distributed Provenance Framework for Edge and IoT Data Systems.
 - [18] Docker Inc., "Docker Compose File Reference," 2024. [Online]. Available: <https://docs.docker.com/compose/compose-file/>
 - [19] OWASP Foundation, "OWASP Top Ten Project," 2024. [Online]. Available: <https://owasp.org/www-project-top-ten/>
 - [20] C. Duhigg, *The Power of Habit: Why We Do What We Do in Life and Business*. Random House, 2012.
 - [21] Mahimalur, R. K., Vargam, M., & Manoharan, D. (2025). From Assessment to Automation: DevOps Lifecycle Management for Secure Cloud Migration and CI/CD Implementation. *Power System Technology*, 49(3).
 - [22] Mudusu, S. K. (2024, August). Achieving fully autonomous AI-driven data pipelines: Exploring zero-touch automation for efficient and scalable data engineering solutions. *International Journal on Recent and Innovation Trends in Computing and Communication*, 12(2), 1182–1186.
 - [23] Maturi, S. Y. (2021). Blockbond hardening: Securing pooled-hash protocols against traffic tampering, MITM hash-rate hijacking, and template coercion. *International Journal of Communication Networks and Information Security*, 13(3), 718–728.
 - [24] P. Venkata Ramana. (2024). AI-driven predictive analytics in ERP systems for proactive supply chain optimization. *International Journal of Innovative Engineering and Management Research (IJIEMR)*.
 - [25] Kumar Gummadi, V. P., Chilamkurthi, L. S., & Kavuri, S. (2026). Distributed Platform Architecture and API-Led Integration. 2026 International Conference on Artificial Intelligence, Systems, and Emerging Technologies (ICAISSET), 1–6. <https://doi.org/10.1109/icaiset66439.2026.11541787>