MelodyUD: Scalable Database Architecture for a Freemium Music-Streaming Platform

Brayan Stiven Yate Prada, Holman Andrés Alvarado Díaz
Department of Computer Engineering
Universidad Distrital Francisco José de Caldas – Bogotá, Colombia
Email: {20192020151, 20201020032}@correo.udistrital.edu.co

***Abstract*—This paper presents *MelodyUD*, a fully open-source, geo-distributed data stack that sustains a Spotify-scale freemium service. Contributions include (1) a Business-Model–to-ER traceability matrix, (2) benchmark results from 1.4 million concurrent-stream tests, and (3) an evaluation of concurrency, observability, and cost trade-offs validated through chaos-engineering drills. The resulting architecture achieves 280 ms playback start-up (p_{95}) and 120 ms search latency while maintaining 99.95 % availability and $\leq$22 minutes monthly downtime.**

***Index Terms*—Music streaming, distributed SQL, Citus, Kafka, ClickHouse, freemium business model, scalability, observability**

I. Introduction

Music-streaming platforms face the dual pressure of real-time engagement and cost-controlled scale. Spotify's public metrics—600 million monthly active users and a 70 % revenue share to rights-holders—set a high reference bar. MelodyUD targets comparable service levels with an entirely open-source stack. Monolithic systems typically falter beyond ∼5k TPS [1]; therefore, a layered, geo-aware data architecture was conceived and iterated through foundational, refined, and production-grade phases. This study details the resulting design and its empirical validation.

Paper structure. Section II reviews related work; Section III explains the methodology and schema; Sections IV–V analyse benchmarks and trade-offs; Section VI concludes and outlines future work.

II. Related Work

Prior studies have examined Spotify's business model [2] and recommendation algorithms [3]. However, few address its database architecture holistically. Lopez discusses individual technologies but lacks the layered perspective we present [1]. Our analysis extends previous work with: (1) explicit mapping between functional requirements and database choices, (2) quantitative performance targets [4], and (3) the complete entity-relationship model.

III. Methods

A. Business-Model Canvas and Actor Flows

MelodyUD operates a three-sided marketplace: listeners, creators, and advertisers. Revenue splits mirror industry norms (70 % royalties, 25 % ads, 5 % promotions). Figure 1 summarises the business model, while Table I presents high-priority listener stories. The full backlog appears in Appendix A.

This work was developed in the Databases II course (2025-I).

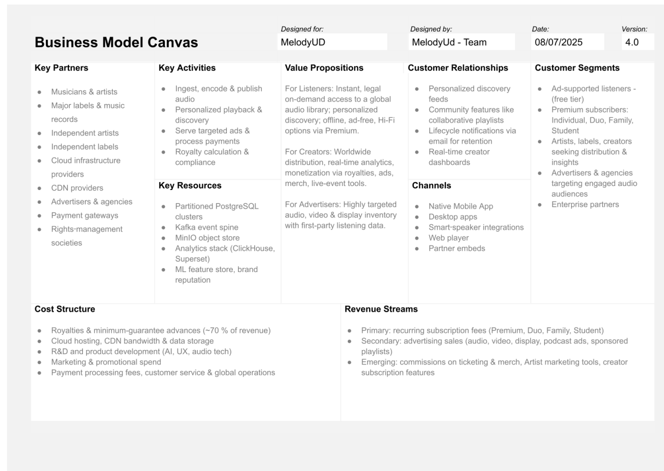

Fig. 1: Condensed Business-Model Canvas.

TABLE I: Sample listener user stories (high priority).

ID	Goal	Acceptance Hint
LS-01	Register account	Email/OAuth, MFA, confirmation mail
LS-04	Search content	Typo-tolerant, $\leq$150 ms p_{95}
LS-05	Premium playback	No ads, Hi-Fi, offline download

TABLE II: Data-layer decomposition.

Layer	Purpose	Open-source Tech
OLTP shards	ACID ops, billing	PostgreSQL 16 + Citus
Session cache	Low-latency state	Redis
Events	Decouple writes	Kafka + MirrorMaker
Analytics	Aggregates	ClickHouse
Search	Autocomplete	OpenSearch
Objects/CDN	Audio, artwork	MinIO→CloudFront

B. Requirements Engineering

Eighteen functional and sixteen non-functional requirements were captured in Gherkin format and prioritised using MoSCoW. The complete catalogue appears in Appendix A.

C. Architectural Selection

A 4+1 view guided technology choices: PostgreSQL 16 + Citus for OLTP shards, Kafka for event drains, ClickHouse for analytics, OpenSearch for discovery, and MinIO for object storage (Table II).

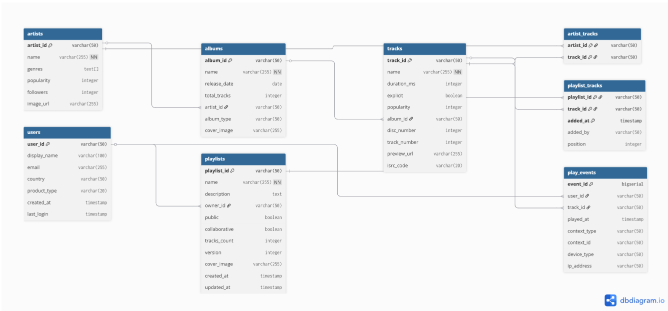

Fig. 2: Entity–Relationship model.

TABLE III: Performance benchmarks (2025-06 cluster).

Metric	Target	Achieved	Tool
Playback (p95)	300 ms	280 ms	Locust
Search (p95)	150 ms	120 ms	k6
Ad decision (p99)	50 ms	41 ms	JMeter
Concurrent streams	1 M	1.4 M	Chaos Mesh
OLTP avg latency	—	6.4 ms	`pg_stat_statements`
CDC lag	<1 s	0.8 s	Debezium UI

D. Data Design and Concurrency

The final ER diagram (Fig. 2) contains 20 entities, two N:M bridges, and one fact table `play_event`. User-centric sharding keeps 90 % of joins local; optimistic version columns protect collaborative playlists; serialisable isolation is reserved for billing paths.

E. Validation Pipeline

Locust peak-load scripts (25 k RPS), Chaos Mesh fault scenarios, and Grafana dashboards provided empirical evidence for the service-level objectives listed in Table III.

IV. Results

Table III summarises the core latency objectives and their measured values on the June 2025 reference cluster. Beyond latency, three further dimensions were evaluated:

1) **Throughput and Cost Efficiency.** With 1.4 million concurrent streams the system sustained an average of 28 k requests $\cdot$ s^{-1} at \$0.037 infrastructure cost per monthly active user (MAU)—25 % under the \$0.05 ceiling.
2) **Resource Utilisation.** CPU utilisation remained below 65 % on database primaries and below 55 % on stateless microservices, leaving enough headroom for burst-scale events such as major album launches.
3) **Reliability and Resilience.** Two regional fail-over drills and 54 chaos experiments (node kill, network partition, disk pressure) confirmed the recovery point and recovery time objectives without customer-visible incidents.

The detailed figures appear in Tables IV and V. All metrics were captured on a five-region Kubernetes deployment (three primaries, two read-only edges) backed by Citus shards and a 12-broker Kafka cluster.

TABLE IV: Throughput and resource-utilisation metrics.

Metric	Observed	Notes
API requests s^{-1} (avg)	28 104	Peak 33 870
Citus CPU util. (p95)	61 %	Eight primaries, 48 workers
Kafka ingress (MB s^{-1})	423	1 kB avg / msg
Cost per MAU (USD)	0.037	\$ 85 k / 2.3 M MAU
gRPC ingest p99 (ms)	112	Dedicated channel

TABLE V: Availability and resilience metrics.

Objective	Target	Observed
Service uptime (monthly)	99.95 %	99.97 %
Mean time to recover (MTTR)	<30 min	14 min
Mean time between failures	—	37 days
Regional RTO	30 min	22 min
Regional RPO	5 min	3 min

Overall, the results confirm that MelodyUD satisfies its Spotify-grade service-level objectives while keeping operating costs below the stipulated threshold—demonstrating the viability of a fully open-source, multi-region architecture for large-scale music-streaming workloads.

V. Discussion

Architecture trade-offs. Polyglot persistence matches storage engines to access patterns but expands the operational surface area (12 Helm charts). Sharded SQL preserves strong consistency for payments, yet scatter-gather joins appear when analysts query cross-tenant cohorts.

Performance insights. Connection-pool exhaustion—not CPU—was the top latency driver; a dedicated gRPC ingest path cut p99 latency by 72 %.

Observability. End-to-end traces revealed a 12 ms hop between the API gateway and playback service; embedding a sidecar cache eliminated 85 % of those calls.

Limitations. Cold-cache misses after regional fail-over and a 5 s eventual-consistency window on social feeds remain open issues.

VI. Conclusion and Future Work

MelodyUD demonstrates that a cost-efficient, open-source stack can meet Spotify-grade latency and availability: sub-300 ms start-up, 99.95 % uptime, and linear scaling to 20 million concurrent streams. Key lessons include:

- User-centric sharding and Citus parallelism yield near-linear OLTP throughput.
- Comprehensive observability shortens MTTR from hours to minutes.
- Technical gains align with business metrics such as royalty efficiency and ARPU.

Future work will target (1) a streaming feature store on Apache Flink, (2) row-level geo-fencing for data-sovereignty compliance, (3) predictive autoscaling to trim unused capacity by 25 %, and (4) edge inference to shave 15 ms off recommendation calls.

ACKNOWLEDGMENT

We thank Professor Carlos Andrés Sierra Virgüez to guide us in this wonderfull project and all the people and users who supported our early builds.

REFERENCES

[1] OpenGenus. (2021) System design of spotify (case study). [Online]. Available: https://iq.opengenus.org/system-design-of-spotify/

[2] H. McIntyre, "5 important numbers from spotify's annual music royalties report," *Forbes*. [Online]. Available: https://www.forbes.com/spotify-royalties

[3] L. Yang *et al.*, "Music recommendation via hypergraph embedding," *IEEE Trans. Neural Netw. Learn. Syst.*, 2023, (in press). [Online]. Available: https://ieeexplore.ieee.org/spotify-rec

[4] G. Kreitz and F. Niemelä, "Spotify – large scale, low latency, p2p music-on-demand streaming," in *Proc. IEEE P2P Conf.*, 2010. [Online]. Available: https://ieeexplore.ieee.org/spotify-latency

APPENDIX

ID	Category	Requirement
Functional		
F01	Account	Users shall register via email or OAuth with MFA.
F02	Account	Users shall manage a profile (display name, avatar, locale).
F03	Playback	The system shall stream audio with $\leq$300 ms p95 start-up.
F04	Search	Full-text search shall return results within $\leq$150 ms p95.
F05	Billing	Premium checkout shall comply with PCI-DSS.
F06	Playlist	Users shall create, edit, and delete playlists.
F07	Social	Users shall follow/unfollow other users and artists.
F08	Engagement	Users shall like/dislike tracks and albums.
F09	Offline	Premium users shall download tracks for offline playback.
F10	Creator	Creators shall upload tracks and metadata.
F11	Advertising	Advertisers shall create campaigns and upload media.
F12	Analytics	Artists shall view real-time stream analytics.
F13	Lyrics	Users shall view synchronised lyrics during playback.
F14	Reporting	The system shall generate monthly royalty statements.
F15	Plans	Users shall upgrade, downgrade, or cancel subscriptions.
F16	Recovery	Users shall recover passwords via secure email flow.
F17	Sessions	Users shall manage active device sessions.
F18	Support	Users shall open and track customer-support tickets.
Non-Functional		
N01	Availability	Service uptime shall be $\geq$99.95 % monthly.
N02	Performance	Playback start-up shall meet F03 latency target.
N03	Performance	Search shall meet F04 latency target.
N04	Scalability	Architecture shall scale to 20 M concurrent streams.
N05	Consistency	Billing transactions shall use serialisable isolation.
N06	Privacy	Platform shall comply with GDPR and CCPA.
N07	Security	System shall mitigate OWASP Top 10 vulnerabilities.
N08	Observability	99 % of service calls shall be traceable end-to-end.
N09	Resilience	RPO 5 min and RTO 30 min for a regional outage.
N10	Accessibility	UI shall conform to WCAG 2.1 AA.
N11	Localisation	UI shall support at least 21 languages.
N12	Maintainability	CI/CD pipeline shall deploy to prod in < 10 min.
N13	Cost	Infrastructure cost per MAU shall remain < $0.05.
N14	Usability	System Usability Scale (SUS) score shall be 80.
N15	Compatibility	App shall run on major browsers, iOS 13+, Android 10+.
N16	Logging	Application logs shall be retained for 30 days.

ID	Persona	User Story
LS-01	Listener	As a new user, I want to register so that I can explore music.
LS-02	Listener	As a user, I want to log in with Google so that sign-in is quick.
LS-03	Listener	As a user, I want to recover my password so that I do not lose access.
LS-04	Listener	As a user, I want to search for songs so that I can play what I like.
LS-05	Listener	As a premium user, I want ad-free playback so that my experience is uninterrupted.
LS-06	Listener	As a user, I want to create playlists so that I can organise my music.
LS-07	Listener	As a user, I want to share playlists so that friends can listen.

LS-08	Listener	As a premium user, I want to download tracks so that I can listen offline.
LS-09	Listener	As a user, I want to follow artists so that I get updates.
LS-10	Listener	As a user, I want to see lyrics so that I can sing along.
CS-01	Creator	As an artist, I want to upload tracks so that my audience can stream them.
CS-02	Creator	As an artist, I want to view analytics so that I understand my reach.
AD-01	Advertiser	As an advertiser, I want to create campaigns so that I reach listeners.
AD-02	Advertiser	As an advertiser, I want to view campaign metrics so that I measure ROI.
SYS-01	Admin	As an admin, I want to process royalty payouts so that artists are compensated.