James Lu

Senior Software Engineer

james.lu9801@gmail.com | Austin, Texas, United States

PROFILE

Senior Full Stack Engineer with 7+ years of experience building scalable web applications and reliable backend services using TypeScript, React, Next.js, and Node.js. Strong background in designing API-driven architectures, optimizing data flows, and delivering responsive, performance-focused user interfaces for high-traffic platforms.
Experienced in working across cloud environments, integrating distributed services, and improving system reliability through clean code structure, testing discipline, and incremental refactoring. I focus on creating maintainable feature architecture, improving developer workflow clarity, and collaborating closely with product and design teams to release features that are stable, intuitive, and user-centered.

PROFESSIONAL EXPERIENCE

Google, *Senior Software Engineer* 11/2024 – 10/2025 | New York, United States

- Architected a cross-service data orchestration layer for Google Gemini's contextual reasoning engine, enabling dynamic retrieval of relevant notebook data with millisecond-level latency. Faced synchronization issues across distributed caches that caused outdated model prompts and inconsistent user responses. Resolved the issue by designing an event-driven data freshness system with Cloud Pub/Sub and Redis streams. This allowed model prompt context to stay updated in real time while cutting API response time by 27%. The solution became a foundational piece for Gemini's personalized query flow.
- Engineered a scalable semantic search pipeline for NotebookLM, integrating Elasticsearch with fine-tuned vector embeddings to surface semantically related notes. Initially faced degraded performance under high query load due to poorly optimized indexing shards. Refactored the shard allocation strategy and introduced asynchronous bulk indexing with batched commits. This improved indexing throughput by 2.3x while reducing CPU utilization across clusters. The new design significantly boosted NotebookLM's ability to retrieve insights across massive datasets.
- Directed the development of an in-document summarization service powered by Gemini's large context models. Encountered GPU memory fragmentation issues when handling long document inference sessions. Implemented adaptive context windowing and dynamic batching in TensorFlow Serving to balance load efficiently. This lowered GPU memory spikes by 41% and stabilized inference latency under concurrent workloads. The resulting system became a reusable framework for other Gemini-integrated summarization tools.
- Optimized a real-time collaboration backend for shared note editing in NotebookLM using TypeScript and Node.js. Conflict resolution between multiple active users often caused stale writes and data overwrites. Introduced operational transformation (OT) logic combined with CRDT-based reconciliation for conflict-free merges. Deployed change tracking to Firestore with atomic revision control to preserve user intent. This provided seamless, Google Docs–level real-time editing for multi-user NotebookLM sessions.
- Devised a data privacy enforcement module ensuring Gemini and NotebookLM models never access restricted workspace data. Discovered gaps where internal document embeddings were accidentally cached without correct permission filters. Developed a hierarchical ACL-based validator using Cloud IAM policies at pre-index and query time. The system dynamically revoked cached items that violated access scopes in under 100ms. This eliminated all shadow data leaks and met Google-wide security audit compliance.
- Built an observability dashboard for Gemini's document-processing workflow using Grafana and Prometheus. The original pipeline offered little visibility into queue delays, token utilization, or model inference durations. Integrated detailed instrumentation hooks and standardized tracing with OpenTelemetry. This exposed hidden performance bottlenecks, reducing average queue backlog by 35%. The dashboard became the primary tool for monitoring Gemini's distributed inference jobs.
- Designed an intelligent caching system for NotebookLM's context retrieval to reduce model re-query frequency. Initially, repeated retrieval calls caused redundant vector lookups and unnecessary GPU inference. Implemented an LRU-based hybrid cache combining Redis (for metadata) and Cloud Storage (for embeddings). This reduced redundant inference requests by 42% while preserving cache freshness with checksum validation. The caching logic now supports other Gemini extensions consuming shared notebook contexts.
- Contributed to a cross-product integration between Gemini and Google Workspace APIs, enabling data ingestion from Docs and Sheets into NotebookLM. During testing, faced format inconsistencies and schema drift across Workspace exports. Developed an adaptive data transformation layer using TypeScript streams and JSON schema validation. This unified data ingestion across APIs and normalized user content structures. The integration unlocked Gemini-assisted insights across all Workspace files with unified access control.

YouTube, *Software Engineer* 09/2021 – 11/2024 | San Francisco, United States

- Orchestrated the development of a real-time engagement analytics dashboard using TypeScript, Next.js, and Python-based data models. During scaling, the system struggled with query latency as concurrent video sessions exceeded 1 million. Integrated time-windowed BigQuery tables and server-side pre-aggregation to offload high-frequency computations. Optimized client-side rendering with incremental static regeneration in Next.js. This achieved a 63% performance boost, ensuring live dashboards updated within 2 seconds.

- Engineered a video retention insight tool embedded directly in the Creator Studio analytics suite. Faced data inconsistency issues between live streams and VODs due to asynchronous ingestion from multiple data sources. Introduced a Kafka-based data harmonization layer and synchronized timestamps with NTP-based offsets. This unified time-series metrics across ingest pipelines and eliminated retention curve mismatches. The system became the foundation for YouTube's audience drop-off analytics module.
- Refined a recommendation engine for Shorts content to improve creator discoverability across regions. Initial deployment caused severe cache churn and uneven exposure bias toward trending channels. Developed a weighted exposure balancer using Redis and dynamic TTLs for underrepresented creators. Combined metadata-driven embeddings with contextual filters for freshness and engagement parity. Resulted in a measurable 19% uplift in cross-category content discovery.
- Architected a content moderation service leveraging Python microservices and TensorFlow models for detecting borderline content. Faced model latency during concurrent inference, leading to backlog accumulation. Implemented asynchronous inference batching using Celery with RabbitMQ for distributed scheduling. This reduced inference time per batch by 40% and stabilized queue throughput under heavy moderation loads. The new pipeline supported 4x the previous processing rate without adding compute cost.
- Led the migration of YouTube's creator recommendation API to a GraphQL-based gateway using Apollo Server. Legacy REST endpoints caused redundant data fetches and inflated mobile payloads. Re-architected the schema to enable selective field fetching and persisted queries. Introduced response caching with Redis and CDN-level ETag validation to prevent over-fetching. This cut average mobile data consumption by 33% while improving perceived load time.
- Enhanced the frontend rendering system for YouTube Studio using Next.js edge rendering. Users experienced hydration mismatches and visible layout shifts on slower networks. Built a hybrid rendering approach combining ISR with React Suspense and streaming SSR. Deployed a granular CSS chunking system with automatic priority hints for above-the-fold content. The result was a 28% reduction in CLS and smoother client hydration across browsers.
- Optimized a metadata classification pipeline that tagged billions of videos using Python and BigQuery ML. Initial model drift led to poor labeling accuracy for emerging categories like "podcasts. "Added active learning loops with human feedback and retraining triggers every 72 hours. Implemented explainability metrics to track category overlap and precision per cluster. Accuracy improved from 84% to 95%, enabling better ad-targeting precision.
- Built a creator notification system for new engagement milestones using Node.js and Firebase Cloud Messaging. Encountered high error rates due to exponential backoff misconfiguration under mobile push loads. Redesigned the retry logic with circuit breakers and token refresh scheduling. Introduced distributed tracing with OpenTelemetry to track delivery performance per region. This achieved 99.7% reliability and real-time milestone notifications within 3 seconds.
- Deployed a real-time sentiment analysis service for live chats during major broadcasts. The model output experienced lag spikes due to unbounded message streams. Introduced message windowing with backpressure controls via Kafka Streams. Parallelized inference batches using asyncio in Python to maintain steady throughput. The system sustained over 250K messages/minute without exceeding 1.5s latency.
- Implemented a collaborative editing feature for YouTube caption editors using React and WebSockets. Developers faced frequent data collisions when multiple editors updated subtitles simultaneously. Added vector-clock synchronization and conflict resolution through CRDT-based merge trees. Deployed change reconciliation on server commit events, ensuring version integrity. This improved editing efficiency for large translation teams and prevented lost edits.
- Directed development of an A/B testing framework for UI components across YouTube's creator tools. Legacy experiments caused redundant deployments and inconsistent metrics capture. Standardized the testing layer using TypeScript decorators and unified analytics hooks. Integrated experiment results directly with BigQuery pipelines for centralized reporting. This reduced test turnaround time by 46% and improved rollout decision accuracy.
- Revamped the YouTube Studio permissions dashboard, enabling fine-grained access controls for multi-user channels. The old system relied on hardcoded role assignments, which made auditing difficult .Implemented an attribute-based access control (ABAC) model stored in Firestore with tokenized permissions. Built an admin UI in React for managing granular role configurations dynamically. This overhaul provided enterprise-level control for large media organizations managing shared accounts.

Amazon, *Software Development Engineer Intern* 06/2020 – 08/2020

Seattle, Washington, United States

- Engineered a real-time inventory visibility microservice to synchronize product availability across multiple regional warehouses. Faced stale stock data and overselling issues due to asynchronous updates between fulfillment centers. Introduced an event-driven architecture with Amazon Kinesis and DynamoDB Streams to propagate updates instantly. Optimized item-level write batching and introduced time-to-live markers for invalidation. Reduced stock discrepancy incidents by 91% during peak sales hours.
- Constructed an automated order notification pipeline using Python, AWS SNS, and Lambda functions. Initial prototype failed to guarantee message delivery under heavy order load. Added SQS-based buffering with idempotent message handling and exponential backoff retries. Instrumented detailed CloudWatch metrics to track delivery success rates per region. The new design achieved over 99.8% delivery reliability during Prime Day simulations.
- Developed a frontend dashboard for seller performance metrics using React, TypeScript, and GraphQL APIs. The dashboard suffered long load times due to multiple API calls for related data. Unified all queries through a federated GraphQL layer backed by Apollo Server. Added client-side caching via normalized entities in Redux Toolkit to minimize network round trips. The page now loads 62% faster and scales efficiently for large data visualizations.
- Enhanced an order recommendation microservice leveraging collaborative filtering on AWS Lambda. The model initially produced cold-start issues for new or low-activity users. Integrated a hybrid model combining item-based similarity with contextual embeddings generated via Python. Introduced Redis-based feature caching to avoid redundant model inference. This increased recommendation accuracy for new users by 22%.

- Refined an internal A/B testing tool used by Amazon Marketplace to validate pricing strategies. Legacy scripts were difficult to scale or integrate with production APIs. Rebuilt the workflow in Node.js using a plugin-based test harness with DynamoDB for persistent experiment tracking. Added CloudFormation templates for automated test deployment across environments. The system reduced setup time from hours to under 15 minutes per experiment.
- Collaborated on an API rate-limiting layer to prevent abuse of public product catalog endpoints. API throttling logic initially caused uneven limits across tenant IDs due to token bucket mismanagement. Redesigned the throttling system with Redis-based distributed counters and weighted fairness. Introduced Prometheus metrics to monitor throttle health and alert on anomalies. This stabilized API performance and reduced overload incidents by 38%.

University of Texas at Austin, *Reseach Assistant* 08/2019 – 05/2020 | Texas, United States

- Contributed to the design of a collaborative data annotation system for a behavioral ML study using React, TypeScript, and Node.js. Concurrency issues appeared when multiple annotators edited the same dataset simultaneously. Resolved this by applying optimistic locking, WebSocket synchronization, and fine-grained delta updates. Introduced automated conflict resolution and version tracking for data reliability. As a result, annotation throughput more than doubled and data loss incidents dropped to zero.
- Architected a high-throughput preprocessing framework to prepare engagement datasets for predictive modeling. The team encountered frequent memory overflows and unbounded joins on large event tables. Refined the ETL logic to chunk input batches, cache intermediate states, and enforce schema validation. Employed Python's multiprocessing for parallel operations and persistent checkpointing via SQLite. This overhaul cut processing time from nine hours to under three while stabilizing data consistency.
- Pioneered the creation of an interactive visualization dashboard for large-scale feature analytics using Flask and D3.js.Early versions suffered from slow rendering and excessive data transfer to browsers. Shifted heavy aggregation to the backend and applied virtualized rendering with canvas-level clipping. Added WebSocket-driven incremental updates to maintain interactivity on massive datasets. The final tool enabled near real-time exploration across millions of samples.
- Enhanced the lab's model evaluation suite to standardize metrics across multiple recommender experiments. Model outputs differed in structure and made comparison nearly impossible. Implemented a modular scoring interface supporting ranking-based and precision metrics interchangeably. Automated result ingestion and visualization through Matplotlib and JSON logging schemas. This harmonized analysis and reduced time spent debugging inconsistent results.
- Introduced a metadata-driven experiment tracking platform to solve reproducibility challenges in ML pipelines. Frequent re-runs produced inconsistent outputs due to missing configuration history. Engineered a PostgreSQL-backed registry with REST endpoints for logging and retrieval. Integrated automatic parameter capture directly from Python notebooks using decorator hooks. This established transparent experiment lineage and simplified reproducibility audits.
- Played a key role in developing a real-time performance monitoring dashboard for digital learning analytics. Latency and performance degraded sharply when live session data exceeded expected limits. Optimized the FastAPI backend through asynchronous processing and response chunking. Applied client-side data throttling and server batching for stable frame rendering. The improved architecture maintained sub-second updates even under heavy student activity.
- Advanced a custom text normalization module for multilingual datasets using spaCy and rule-based filters. Tokenization inconsistencies caused degraded model accuracy in cross-lingual experiments. Designed a layered NLP pipeline with domain-specific vocabulary injection and cached embeddings. Parallelized computation using Python's concurrent futures and disk-based persistence. Accuracy improved across test sets while processing throughput rose by nearly 50%.

JPMorgan Chase & Co., *Software Engineering Intern* 06/2019 – 08/2019 | New York, United States

- Assisted in developing a trade reconciliation dashboard for internal analysts using React and Node.js. The system initially lacked real-time data refresh, forcing manual page reloads to view updated positions. Integrated WebSocket streaming with server-side event emitters to deliver live updates efficiently. Deployed Redis Pub/Sub to broadcast incremental changes across distributed servers. This provided near-instant visibility into trade discrepancies for compliance teams.
- Built a data transformation service to normalize transaction records coming from multiple legacy systems. Parsing failures frequently occurred due to inconsistent CSV and XML formats. Created a robust schema mapping layer in Python using regex parsing and dynamic field validators. Added automated error-reporting hooks with structured logs sent to Splunk dashboards. Data normalization success rate improved from 72% to 98%, greatly reducing manual cleanup.
- Contributed to the modernization of an internal authentication microservice using OAuth 2.0 and JWT tokens. Legacy sessions often expired unexpectedly, disrupting analyst workflows. Implemented refresh-token logic with encrypted cookies and fine-grained access scopes. Set up integration testing in Postman and CI validation via Jenkins pipelines. This stabilized session management and reduced authentication-related incidents by over 60%.
- Created QA automation suite for internal financial APIs using PyTest and Docker; integrated with Jenkins and Slack for CI alerts.

SKILLS

Languages & Engineering Foundations
TypeScript, JavaScript, Go, Python, Django

Data & Storage Systems
MySQL, MongoDB, PostgreSQL, Redis

Backend & Distributed Systems
Microservices, REST APIs, PHP, Node.js, NestJS, GraphQL

Cloud & Infrastructure
Google Cloud, AWS, Docker, Kubernetes, CI/CD pipelines

Frontend & Application Frameworks
React, Next.js, Redux, Angular, Tailwind CSS, Material UI, React Native

EDUCATION

Bachelor Electrical and Computer Engineering, *University of Texas at Austin* 08/2017 – 05/2021 | Texas, United States