Technical Project Report

Sydney Event Aggregation & AI Assistant

Prathmesh Upadhyay
Domain: Web Intelligence, LLMs, & Data Engineering
Version: 1.0 (MVP)

1. Executive Summary

The 'Sydney Event Recommendation Assistant' is an autonomous ecosystem designed to solve the problem of information fragmentation in Sydney's leisure industry. The system integrates an automated data scraping pipeline, a dual-database storage strategy (SQL/NoSQL + Vector), and a Generative AI interface. By leveraging Retrieval-Augmented Generation (RAG), the project transitions from traditional keyword-based search to context-aware semantic recommendations.

2. System Architecture

Our architecture is decoupled to ensure high availability and modularity.

A. The Data Engine (Scrapers & Pipeline)

Technologies: Node.js, Puppeteer/Cheerio, Cron-Jobs.
Mechanism: Hourly triggers scrape major Sydney event hubs.
Data Cleaning: An automated normalization layer removes HTML tags, fixes date formats, and detects duplicates using title-venue-date hashing.

B. The Storage Layer (Hybrid Strategy)

Relational/Structured (MongoDB): Stores core event metadata (Venue, Price, Date, URL) for high-speed dashboard rendering.
Semantic Layer (ChromaDB): Text descriptions are converted into 384-dimensional vectors using the all-MiniLM-L6-v2 transformer model. This enables the system to understand that a user asking for 'soulful evening' should be shown 'Jazz' or 'Acoustic' events.

C. The AI Orchestration (Telegram + LangChain)

Interface: Telegram Bot API for low-latency, mobile-first interaction.
Logic: LangChain acts as the controller, managing memory (user preferences) and directing queries to the Vector DB.
Model: Open-source LLM (Llama 3/Mistral) hosted via Ollama to ensure data privacy and zero API costs.

3. Key Technical Features & Innovation

Feature	Technical Implementation	User Benefit
Semantic Matching	Vector Cosine Similarity (ChromaDB)	Finds events based on 'vibe' rather than just tags.
Proactive Alerts	Background Worker + User Preference	Users get a ping when a 'matching' event is scraped.
Admin Control Room	React-based Dashboard	High-quality, curated data through manual override options.
Auto-Expiration	Time-TTL Logic	No 'zombie' events; DB remains clean and performant.

4. Challenges Overcome

Dependency Hell

Resolved version conflicts between langchain, huggingface_hub, and httpcore by implementing an isolated Virtual Environment and strict version pinning.

Duplicate Detection

Implemented a fuzzy-matching algorithm to identify the same event posted on multiple platforms with slightly different names.

Cold Start Problem

Pre-seeded the Vector DB with 100+ Sydney events to ensure the AI had immediate 'knowledge' upon deployment.

5. Development Roadmap (Scalability)

Phase 2: Integration of Multi-modal AI (OCR for flyers) - Users can upload a photo of a street poster to add it to the DB.
Phase 3: Social Integration - Group chat bots that help friends vote on an event directly within Telegram.
Phase 4: Personalization Engine - Using Collaborative Filtering to suggest events based on what similar users liked.

Tools & Technologies Used

- React (Frontend)
- Node.js (Backend)
- Python (AI/Bot)
- MongoDB (Database)
- ChromaDB (Vector Store)
- Ollama (Local LLM)
- Telegram API

6. Conclusion

This project demonstrates the transition from a passive data-aggregator to an active AI assistant. By combining robust web-scraping with cutting-edge Vector Search, we have created a scalable solution that provides genuine utility to Sydney residents, making event discovery as simple as a text message.