

SG Hangout Guide: Application of Generative Search Engine and Chatbot-As-A-Service Example

SG HANGOUT GUIDE

Find the perfect hangout spot in Singapore

for a cafe with friends

for shopping with family

to visit a museum indoors

to relax in an outdoor park

Found 5 hangout spots matching your criteria.

**ORCHARD ROAD**
Block 2, ORCHARD TURN, ORCHARD ROAD, Singapore 238801
Postal Code: 238801
Coordinates: 1.30409785787899, 103.832251902468

**ION ORCHARD**
Block 2, ORCHARD TURN, ION ORCHARD, Singapore 238801

Zhao Qixian U2321752L

Information

Source Code is available at:

https://github.com/ZhaoQixian/Cloud_Computing/tree/main/SG_hangout_guide

Deployed website is at: delay is expected as a free version is used

<https://singapore-hangout-guide.onrender.com/>

Introduction

The SG Hangout Guide is an interactive web platform that transforms casual, free-form queries—such as “find a cozy café near Chinatown for two”—into precise, location-based recommendations across Singapore. Leveraging advanced natural language processing (NLP) powered by OpenAI’s GPT-4.1-mini, the system offers real-time grammar corrections and AI-aided suggestions, prompting users to supply missing details (e.g., group size, region, or indoor/outdoor preference) without interrupting the conversational flow. Geospatial data is integrated seamlessly via Singapore’s OneMap API, delivering up-to-date venue information, addresses, and coordinates. Underpinning this functionality is a cloud-native architecture that combines containerized microservices (Express.js backend, React.js frontend) with serverless components and managed cloud databases. This design ensures automatic scaling under variable loads, low-latency responses, and high availability, enabling a responsive user experience.

Problem Statement

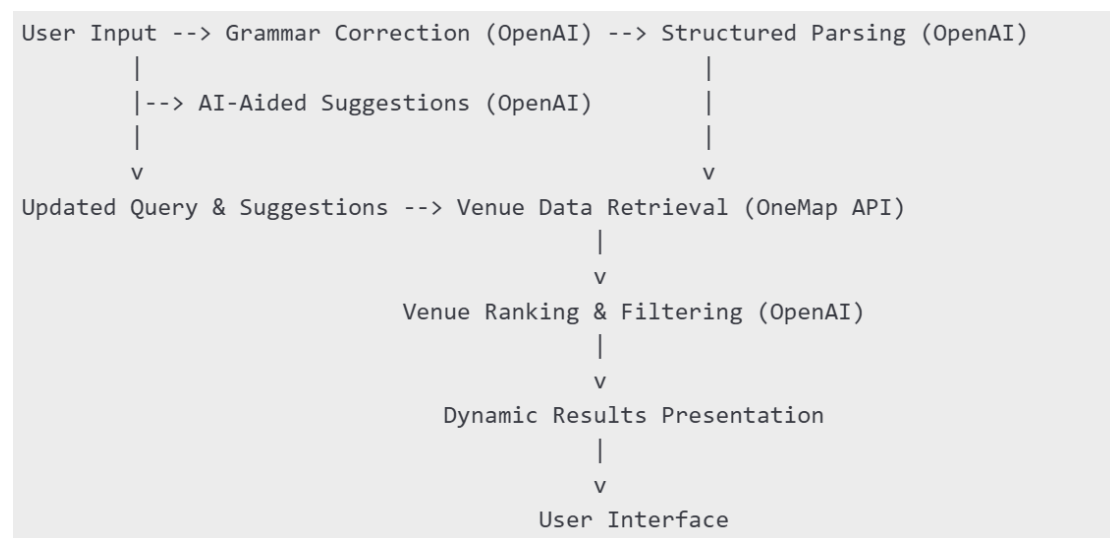
Conventional venue-search interfaces force users into rigid keyword fields and multi-level menus, creating friction when expressing needs in natural language. Interpreting conversational queries—such as “romantic outdoor spot for a family of four in East Coast”—requires extracting structured search parameters (activity type, region, group size, indoor/outdoor preference) in a robust, scalable manner. Integrating and normalizing geospatial data from the OneMap API introduces additional challenges around rate limits, CORS restrictions, and inconsistent response formats, which must be managed through intelligent caching and fault-tolerant data flows. Moreover, providing real-time AI-driven prompts to elicit missing criteria and delivering personalized recommendations at scale demands a dynamic cloud infrastructure that can auto-scale, optimize resource utilization, and maintain secure, low-latency interactions.

Solution Design

The SG Hangout Guide is built on a cloud-native architecture that seamlessly integrates containerized microservices with serverless AI functions and managed data services. Frontend components (React.js) and backend APIs (Express.js) run in Docker

containers orchestrated by Kubernetes, enabling rolling updates, service discovery, and horizontal scaling under variable demand. Stateless NLP tasks—such as grammar correction, structured query parsing, and suggestion generation—are executed via serverless functions (AWS Lambda/Azure Functions), which scale automatically to handle bursty workloads while minimizing idle costs. Persistent state and API cache entries are stored in a managed NoSQL database (DynamoDB/Cosmos DB) and an in-memory cache (Redis), reducing latency and mitigating OneMap API rate limits. Continuous integration and deployment pipelines ensure rapid, reliable releases, and monitoring tools (CloudWatch/Application Insights) track key performance metrics and trigger autoscaling. Fault tolerance is achieved through circuit breakers and exponential backoff patterns that gracefully handle transient API failures, ensuring consistent, low-latency responses and a robust user experience.

The data flow and user experience within the SG Hangout Guide can be described as follows:



1. **User Input:** Users enter conversational queries via a React-based user interface.

I want to go shoppinggggggggggg

2. **Grammar Correction:** Backend service checks and corrects query grammar through OpenAI's GPT-4.1-mini model.

I want to go shopping in Orchard

3. **Structured Query Parsing:** The corrected query is parsed into structured components (activity type, region, group size, indoor/outdoor preferences) using OpenAI. This is for backend query thus not shown in frontend

```
Parsed Criteria: {
  activityType: 'shopping',
  region: 'Orchard',
  groupSize: null,
  isIndoor: null
}
```

4. **AI-Aided Suggestion Generation:** The system proactively generates targeted suggestions to help users refine queries, addressing missing criteria or prompting for additional clarifications.

A user interface for refining a query. At the top, a text input field contains the query "I want to go shopping in Orchard". Below this, there are five orange, pill-shaped buttons arranged horizontally. Each button contains a suggestion: "with friends", "for family", "indoors", "on weekends", and "for couples".

5. **Venue Retrieval:** Structured queries are used to fetch relevant venue data from the OneMap API, ensuring accurate geographical and venue-specific information.

A screenshot of a search results interface. At the top, it says "Found 5 hangout spots matching your criteria." Below this is a yellow card for "ORCHARD SHOPPING CENTRE". The card includes the address "Block 321, ORCHARD ROAD, ORCHARD SHOPPING CENTRE, Singapore 238866", the "Postal Code: 238866", and "Coordinates: 1.30149992510852, 103.837888257002".

6. **Venue Ranking:** Results from OneMap are ranked by relevance to user queries using OpenAI's model.

```
Raw ranking response from OpenAI: [
  {"index": 1, "rank": 1},
  {"index": 2, "rank": 2},
  {"index": 5, "rank": 3},
  {"index": 3, "rank": 4},
  {"index": 4, "rank": 5}
]
```

7. **Final Presentation:** Ranked results, along with suggestions and grammar corrections, are dynamically presented back to the user via the frontend.

SG HANGOUT GUIDE

Find the perfect hangout spot in Singapore

indoors

outdoors

in a mall

at a boutique

at a street market

Found 5 hangout spots matching your criteria.

**ORCHARD SHOPPING CENTRE**
Block 321, ORCHARD ROAD, ORCHARD SHOPPING CENTRE, Singapore 238866
Postal Code: 238866
Coordinates: 1.30149992510852, 103.837888257002

**CITYLINK MALL**
Block 1, RAFFLES LINK, CITYLINK MALL, Singapore 039393
Postal Code: 039393
Coordinates: 1.2927777312893, 103.854173501417

Illustrative Examples

Consider a user entering the query: "Find a romantic café near Orchard."

Initially, the backend corrects any grammatical errors, ensuring optimal clarity. Next, it parses the query into structured search criteria, identifying that the user seeks:

- Activity Type: Café
- Region: Orchard
- Group Size: 2 (inferred from "romantic")
- Indoor Preference: Not specified

Subsequently, AI-aided suggestions prompt the user to refine the query further, such as "with outdoor seating" or "open late," enhancing search specificity and results relevance. Users may click those suggestions and add to their original query, or keep their own queries and view the queried results.

with indoor seating


with outdoor seating

with cozy indoor ambiance

with garden or patio seating

with scenic outdoor views

Found 5 hangout spots matching your criteria.

**THE ATRIUM @ ORCHARD**
Block 60A, ORCHARD ROAD, THE ATRIUM @ ORCHARD, Singapore 238890
Postal Code: 238890
Coordinates: 1.29947530530804, 103.845466275418

The refined criteria fetch suitable café listings via OneMap, followed by ranking these venues based on their relevance to the user's preferences. The results are dynamically presented, including venue names, addresses, and coordinates, ensuring a comprehensive and satisfying user experience.

Advantage

- **Minimal Knowledge Barrier:** The frontend uses standard React hooks and familiar libraries like axios and lodash, while the backend relies on Express.js and simple route handlers—skills commonly taught in web development courses.
- **Single and CHEAP Unified Query:** A single API call via `findHangoutSpots(query)` handles grammar checking, NLP parsing, OneMap search, and ranking, reducing complexity compared to multiple endpoints. Total cost of sending OpenAI gpt-4.1-mini cost during my development is below 10 US cents.
- **Debounce & State Management:** Built-in debouncing (lodash/debounce) and React state hooks ensure efficient query handling without overloading the backend, making it easy to maintain and extend.
- **Extensibility:** The modular design—separating grammar, suggestions, and search logic—allows easy addition of new features (e.g., voice input, alternative map APIs) without large refactors.

Limitations

The system's reliance on the OneMap API exposes it to rate limits, occasional downtime, and inconsistencies in geospatial data formats, which can degrade search accuracy or availability. Serverless AI functions may incur cold-start latency, leading to brief periods of slower response for rarely invoked endpoints. Additionally, continuous use of NLP and generative models can result in unpredictable cloud costs if not closely monitored and optimized.

Future Applications

The capabilities demonstrated by SG Hangout Guide represent just one facet of a broader shift toward generative search engines and conversational data access. In future implementations, similar architectures can be extended to natural language database querying: users could pose questions in plain English—such as “Show me last quarter’s top-selling products in Southeast Asia”—and the system would parse intent, generate optimized SQL queries, execute them against relational or data warehouse systems, and return results enriched with AI-generated summaries and visualizations. Beyond reporting use cases, generative search engines could streamline business intelligence workflows by converting informal business questions into OLAP queries, enable on-the-fly data exploration for analysts, and support self-service analytics. By combining domain-specific knowledge graphs, real-time data pipelines, and advanced NLP models, organizations across finance, supply chain, and healthcare can unlock

immediate insights from complex data stores using conversational interfaces.

By combining cloud-native microservices, serverless AI functions, and managed data services, SG Hangout Guide delivers a seamless, scalable conversational search experience. The principles of fault-tolerant API integration, AI-driven query refinement, and auto-scaling infrastructure can be adapted to diverse domains—healthcare triage, retail search, and AI tutoring—highlighting the broad applicability of this generative search engine model.

Conclusions

The SG Hangout Guide exemplifies how cloud computing and advanced NLP technologies can significantly enhance the user experience by providing intuitive, responsive, and personalized search solutions. The use of cloud services ensures scalability, robust performance, and real-time interactions, while the integration of OpenAI's models enables nuanced understanding and interpretation of user intent. Future iterations may explore voice-based interactions, more sophisticated multi-turn dialogues, and further optimization of API interactions to enhance system efficiency and reduce operational costs.

References

- OpenAI Documentation: <https://platform.openai.com/docs>
- OneMap API Documentation: <https://www.onemap.gov.sg/docs>
- React Documentation: <https://reactjs.org/docs/getting-started.html>
- Express.js Documentation: <https://expressjs.com/>