TekSystems Takehome

GCP Services Used and Plan for Productionization

1. **Google BigQuery**: This is the primary data source. The application queries the public `bigquery-public-data.imdb.reviews` dataset to fetch movie review data. The interaction with BigQuery is handled by the `google-cloud-bigquery` client library, with data access logic in `reviews/data.py`.
2. **Google Cloud Authentication**: current application authenticates to GCP services using a service account with an env variable. This is best practice, for local testing. However, for a production deployment within a GCP environment like Vertex AI, this method would be replaced with a dedicated IAM (Identity and Access Management) service account. The Google Cloud client libraries currently used are designed to automatically use the credentials of the service account attached to the runtime environment.
3. *__Vertex AI (Potential Future Use)__: While the current implementation uses an in-memory session service (`InMemorySessionService`), the project notes and the choice of the `google-adk` make it easy to integrate **Vertex AI's** memory and session service.

Discussion of Results, Challenges, and Trade-offs

Challenges

- **Entity Resolution**
  - Common GenAI challenge: mapping user queries to structured records. This can be improved by improving our lookup tools with a stronger search tool.
  - In our case: `lookup_by_title` is brittle and fails on slight mismatches.
  - Needs improvement using fuzzy matching, embedding-based retrieval, or hybrid approaches.
- **Structured Output & Response Contracts**
  - Current tagging tool lacks strict format enforcement.
  - Pydantic can enforce schemas for consistent output.
  - LiteLLM integrates this seamlessly; unclear how ADK handles tight schema enforcement — needs investigation.

Google Agent Development Kit (ADK)

Overview

ADK is an open-source framework introduced at Google Cloud Next 2025. It enables modular, multi-agent orchestration with built-in support for GCP services and production-grade deployment on Vertex AI.

Pros

- Strong integration with GCP — ideal for teams already on Vertex AI.
- Modular agent design with `Sequential`, `Parallel`, `Loop`, and `LlmAgent` orchestration options.
- Rich tool ecosystem including OpenAPI, external functions, and managed services.
- Gemini CLI and Agent Engine UI provide a strong dev experience.

Cons

- New framework with limited community support and minimal third-party examples.
- Fewer prebuilt wrappers compared to LangChain or LiteLLM.
- Limited abstraction; requires more boilerplate and custom implementation for advanced use cases.
- Needs further validation around schema enforcement (pydantic) and tool compatibility like LiteLLM.

Productionization Plan

1. Deployment & CI/CD

- Deploy FastAPI on **Cloud Run** or **GKE Autopilot** for autoscaling and managed infrastructure.
- Use **Secret Manager** in prod, `.env` locally, and GitHub secrets in CI/CD.
- CI/CD (e.g., GitHub Actions) should run: Linting (`ruff`, `mypy`, `black`), Mocked tests for agents/tools, Deployment gates…
- Would probably deploy using a dockerized version of this fastapi app. Build and push Docker images to **Artifact Registry**.
  Deployment Options:
    - **Cloud Run (preferred)**: Simple, fast serverless deployment with autoscaling to zero, native logging, and IAM integration.
    - **GKE Autopilot + Helm**: For more control, use Helm to manage deployments, configure networking, and orchestrate multiple services.

2. GCP-Native Agentic Stack

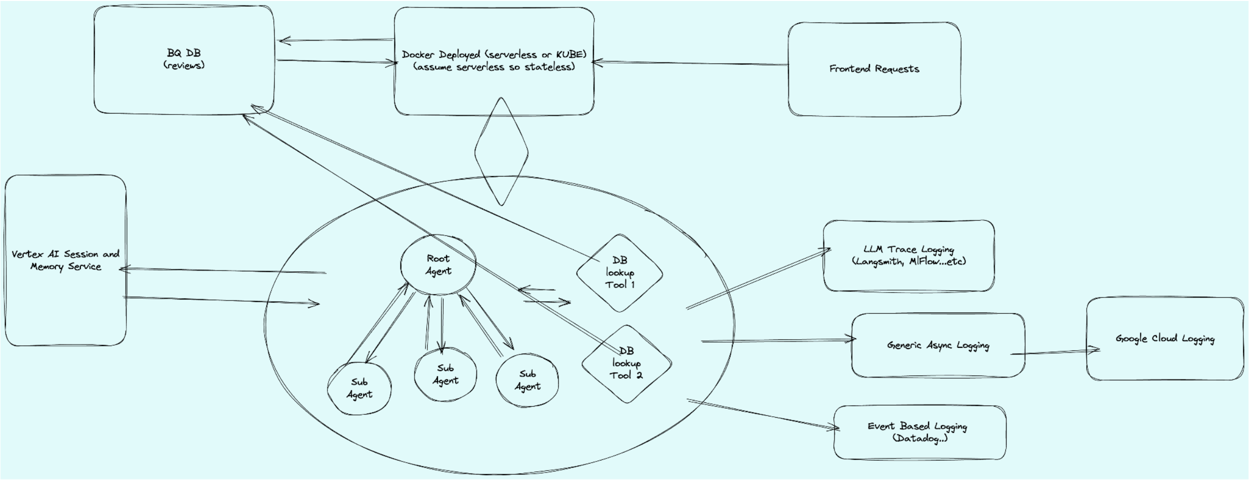
- Swap `InMemorySessionService` for **VertexAISessionService** for persistent memory.
- Use **Vertex AI Express Mode** for lightweight memory + session tracking.

3. Observability & Logging

- Add structured logging using `ContextVars`, stream to **Cloud Logging**. That would allow for user_ids, sessions_ids, and possible trace_ids to be sent to google native Cloud Logging.

- Wrap key functions with `@traceable` decorators using **LangSmith** or similar services to track agent flows, tool calls, and model outputs

Diagram



EXAMPLE RESULTS:

These can all be run using **main.py** with instructions in the readme.

🎬 Query: Tell me about *The Kite Runner*

Agent's Response

Summary:
"The Kite Runner," based on Khaled Hosseini's best-selling novel, explores friendship, betrayal, and redemption against the backdrop of Afghanistan's turbulent history. The narrative follows Amir, a privileged Pashtun boy, and Hassan, his loyal Hazara friend. A traumatic childhood event drives a wedge between them, sending Amir on a lifelong journey of guilt and redemption. Now living in the U.S., Amir is called back to Afghanistan to face his past.

While the film captures the innocence of youth and the emotional intensity of its themes, critics cite issues with pacing and narrative cohesion. Still, it is recognized for its emotional depth and evocative portrayal of Afghan culture.

Tags & Themes:

- Friendship
- Betrayal
- Redemption
- Guilt
- Afghan Culture
- Childhood Innocence
- Trauma
- War and Conflict
- Social Class Divide
- Family Dynamics

🎬 Query: What's the movie *A Funny Thing Happened on the Way to the Moon?*

Agent's Response

Summary:
"A Funny Thing Happened on the Way to the Moon" has been widely criticized for promoting conspiracy theories about the Apollo moon landings. Reviewers point out its lack of credible evidence, reliance on misinformation, and use of debunked claims. Though the technical quality of the documentary is passable, its content is viewed as misleading and propagandistic.

Tags & Themes:

- Conspiracy Theories
- Fake Moon Landing Claims
- NASA
- Apollo Missions
- Misinformation
- Documentary Criticism
- Scientific Accuracy
- Space Exploration
- Viewer Skepticism
- Historical Interpretation

🎬 Query: Summarize reviews for *Return of the Jedi*

Agent's Response

Summary:
"Star Wars: Episode VI – Return of the Jedi" receives mixed reviews overall. Many fans celebrate its emotional depth, thrilling action, and strong character arcs—especially the resolution of Darth Vader and Luke Skywalker's journeys. The film is praised for its impressive special effects and iconic score by John Williams. However, several critiques target the use of the Ewoks, viewed by some as tonally jarring and overly commercial. While some regard it as a satisfying conclusion to the original trilogy, others find it weaker than its predecessors. Based on 26 reviews, the response reflects both nostalgia-driven appreciation and measured skepticism.

Supplemental (Data Exploration .txt with BQ query)

```
-- Stats on review length, label distribution, and reviewer ratings
WITH movies AS (
  SELECT DISTINCT movie_id
  FROM `bigquery-public-data.imdb.reviews`
  WHERE RAND() < 0.2
  LIMIT 100
),
all_sampled_info AS (
  SELECT *
  FROM `bigquery-public-data.imdb.reviews`
  JOIN movies USING(movie_id)
  WHERE label IN ('Positive', 'Negative')
),
review_stats AS (
  SELECT
    MIN(LENGTH(review)) AS min_length,
    MAX(LENGTH(review)) AS max_length,
    AVG(LENGTH(review)) AS avg_length,
    APPROX_QUANTILES(LENGTH(review), 100)[OFFSET(50)] AS median_length
  FROM all_sampled_info
),
label_distribution AS (
  SELECT
    label,
    COUNT(*) AS count,
    COUNT(*) * 100.0 / SUM(COUNT(*)) OVER() AS percent
  FROM all_sampled_info
  GROUP BY label
),
rating_distribution AS (
  SELECT
    CAST(reviewer_rating AS INT64) AS rating,
    COUNT(*) AS count,
    COUNT(*) * 100.0 / SUM(COUNT(*)) OVER() AS percent
  FROM all_sampled_info
  GROUP BY rating
),
review_lengths AS (
  SELECT
    movie_id,
    SUM(LENGTH(review)) AS total_review_length
  FROM `bigquery-public-data.imdb.reviews`
  GROUP BY movie_id
),
label_distribution_by_movie as
(
  SELECT
  MIN(total_review_length) AS min_total_length,
  MAX(total_review_length) AS max_total_length,
  AVG(total_review_length) AS avg_total_length,
  APPROX_QUANTILES(total_review_length, 100)[OFFSET(50)] AS median_total_length,
  MIN(review_number) as min_review_number ,
  MAX(review_number) as max_review_number,
  AVG(review_number) as avg_review_number
)

-- Final SELECT to show all together (can also run individually)
--SELECT * FROM review_stats;

--SELECT * FROM label_distribution;

--SELECT * FROM rating_distribution ORDER BY rating;

SELECT * from label_distribution_by_movie;
```

Results

Stats by single review

```
Row     min_length    max_length   avg_length     median_length
1       166      5983     1250.8614097968946940
```

that means that average review is around 200-250

Stats by total reviews for each movie

```
min_total_length      max_total_length      avg_total_length       median_total_length
min_review_number max_review_number           avg_review_number
52        81735  9168.62469030933614739    1          30        7.0786437318609812
```

This means that max chars for a review is around 81k which is at most 81k/3 < 30k tokens which means this is lower than any models context window
we are safe to go to just throw em all in there for summarization

we also have an average of 7 reviews per movie which would be a nice good summary use case

Review Sentiments

1	**Negative**	**766**	**86%**
2	**Positive**	**124**	**14%**

rating	count	percent
1	334	39.018691588785046
2	120	14.018691588785046
3	140	16.355140186915889
4	104	12.149532710280374
7	40	4.6728971962616823
8	35	4.0887850467289724
9	31	3.6214953271028039
10	52	6.0747663551401869

Skewed to negative reviews which is interesting to see, would mean a classifier would have to be evaluated and trained accordingly
This affects our evaluator agent.
The review would number distribution confirms this: very left tailed skew of 40% 1/10 rating.