Trade-offs and Design Decisions

1. Global Cache (Redis)

Trade-offs:

- **Pros:** Fast data retrieval, reduces database load, improves response time.
- **Cons:** Potential data inconsistency, additional infrastructure to manage.

Decision:

- Redis is used as a global cache to store frequently accessed weather data and reduce latency.

2. Separate Service Manager

Trade-offs:

- **Pros:** Centralized orchestration, easier service communication.
- **Cons:** Potential single point of failure, additional complexity.

Decision:

- Implementing a separate service manager simplifies service coordination while ensuring scalability.

3. Load Balancer Strategy

Trade-offs:

- **Round-robin:** Ensures equal load distribution but doesn't account for server health.
- **Least connections:** Distributes traffic based on server load but may cause uneven distribution in sudden spikes.
- **IP Hash:** Ensures session persistence but can lead to uneven load.

Decision:

- A combination of **Least Connections** and **Round-robin** is used for optimal traffic distribution.

4. Rate Limiter Implementation

Trade-offs:

- **Fixed window:** Simple but may cause bursts at window edges.
- **Sliding window:** More even distribution but higher computation overhead.
- **Token bucket:** Smooth flow but requires fine-tuned configuration.

Decision:

- **Token bucket** is used to allow smooth request handling while preventing abuse.

5. SQL vs NoSQL Databases

Trade-offs:

- **SQL:** Strong consistency but may struggle with high scalability demands.
- **NoSQL:** High availability and scalability but eventual consistency issues.

Decision:

- SQL is used for **Profile and Notification services** due to structured data needs.
- NoSQL is used for **weather-related services** due to high scalability requirements.

6. Kafka for Event-Driven Architecture

Trade-offs:

- **Pros:** High-throughput messaging, decoupled architecture.
- **Cons:** Operational complexity, additional resource consumption.

Decision:

- Kafka is used to efficiently handle real-time event processing and ensure scalability.

7. Centralized Service Manager

Trade-offs:

- **Pros:** Simplifies orchestration and service communication.
- **Cons:** Single point of failure risk, increased infrastructure complexity.

Decision:

- A Service Manager is implemented to coordinate service interactions efficiently.

8. Monitoring with Prometheus & Grafana

Trade-offs:

- **Pros:** Real-time monitoring, alerting, and insights into system health.
- **Cons:** Additional overhead in setup and maintenance.

Decision:

- Prometheus and Grafana are used to ensure observability of the system.

9. REST API Over GraphQL or gRPC

Trade-offs:

- **REST:** Simpler implementation, widely supported but less efficient for complex queries.
- **GraphQL:** Flexible queries but requires additional optimization.
- **gRPC:** Faster communication but requires strict contract definitions.

Decision:

- REST API is chosen for its simplicity and ease of implementation.

10. CDN for Frontend Optimization

Trade-offs:

- **Pros:** Faster content delivery, reduced backend load.
- **Cons:** Additional caching invalidation complexity.

Decision:

- CDN is used to serve static frontend assets for performance improvements.