Workshop 2: Data System Architecture and Information Retrieval
Information Requirements for AgroClima Prediction System

César Andrés Torres Bernal (20191020147)
Juan David Duarte Ruiz (20191020159)

Universidad Distrital Francisco José de Caldas

High-level Architecture

The AgroClimaIQ platform is designed to operate as a scalable, data-driven system capable of ingesting, processing, storing, and delivering actionable climate intelligence to agricultural stakeholders. Given the system's reliance on large volumes of external data and personalized recommendations, a robust and modular data system architecture is essential.

This architecture follows a layered design pattern that supports continuous data ingestion from external APIs and simulated sources, distributed storage of structured and unstructured data, real-time and batch data processing, and secure delivery of insights through APIs and user interfaces. The architecture also integrates a Business Intelligence (BI) layer to enable data visualization and strategic decision-making. Each component in the system plays a specialized role—from collecting and validating weather data, to executing climate risk models, to generating field-specific recommendations. The architecture ensures high availability, scalability, and modularity to support a wide range of user roles, from farmers in the field to analysts and system administrators.

The following section shows the high-level architecture diagram and describes each architectural layer, its key components, the technologies employed, and how data flows between them to enable real-time decision support and long-term agricultural planning.

High-level Architecture Diagram

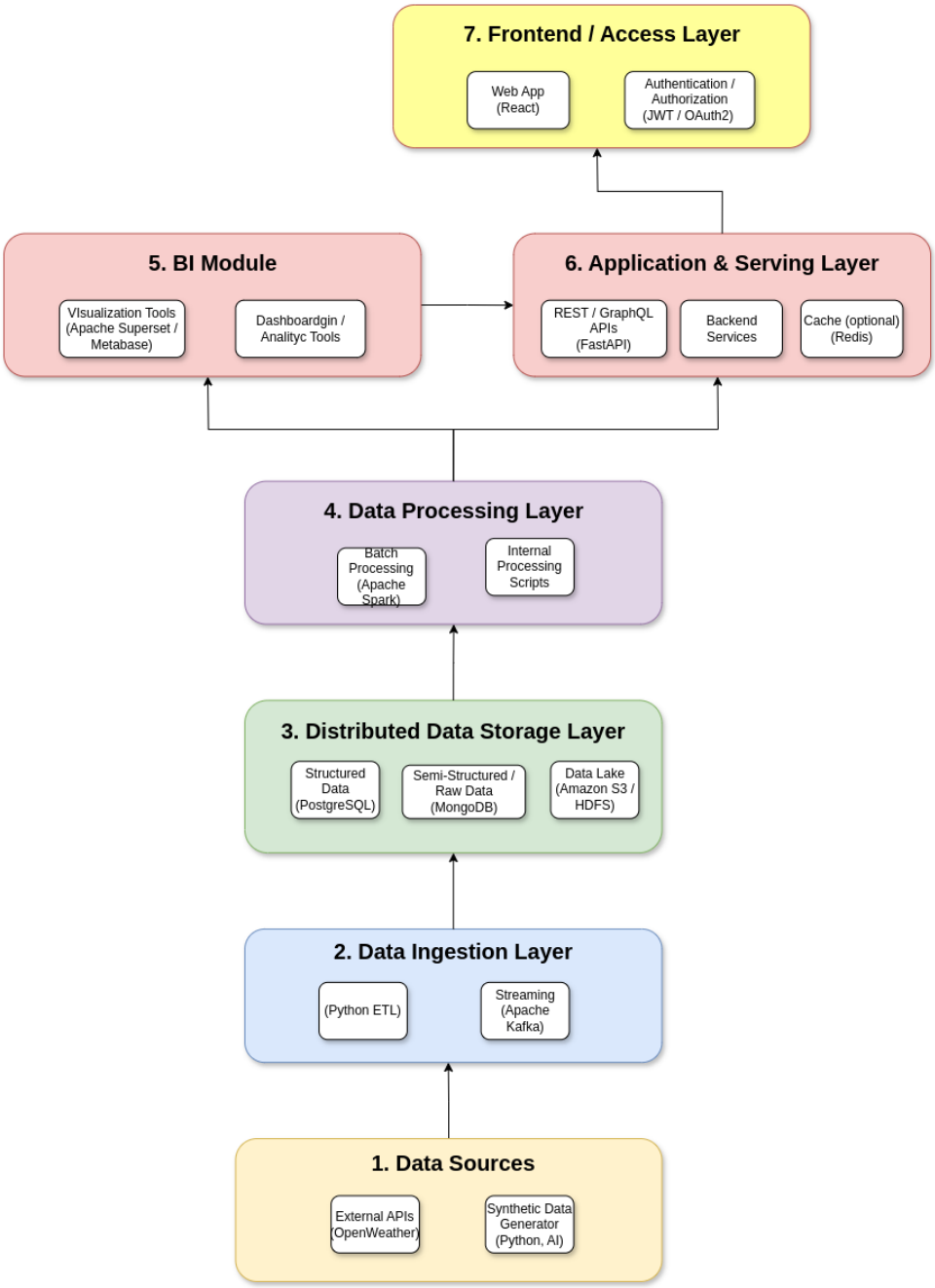

Figure 1: High-level Architecture Diagram

1. Data Sources Layer

Role: This layer serves as the entry point for all raw data entering the system. It is responsible for retrieving weather, climate, and agricultural data, both from external sources and synthetic generators.

Components & Technologies:

- **External APIs:** `OpenWeather`, `OpenMeteo`, `NOAA` – provide real-time and historical weather and climate data.
- **Synthetic Data Generator:** Python scripts using libraries such as `Faker`, `NumPy`, or `Pandas` – simulate agricultural data in academic or testing scenarios.

Interaction: These sources feed data directly into the ingestion layer through scheduled or real-time calls.

2. Data Ingestion Layer

Role: Responsible for acquiring, normalizing, validating, and routing incoming data from all sources to the system's internal data stores.

Components & Technologies:

- **Apache Kafka** (optional): For high-throughput, real-time stream ingestion where data events are published and consumed asynchronously.
- **Python ETL Scripts / Apache Airflow:** Used for scheduled jobs that run data transformation and enrichment pipelines.

Interaction: Data flows from external APIs or synthetic data generators → is collected and processed via custom Python ETL scripts → and is then stored directly in PostgreSQL (for structured data) and MongoDB (for unstructured or semi-structured data). Kafka (if used) serves to decouple real-time data streams from downstream processing and storage components.

3. Data Storage Layer

Role: This layer stores both raw and refined data, allowing persistence and long-term access. It includes both relational and `NoSQL` solutions.

Components & Technologies:

- **PostgreSQL:** Stores structured relational data: users, crops, alerts, predictions, sessions. Chosen for its `ACID` compliance, query performance, and relational integrity.
- **MongoDB:** Stores semi-structured data: `API` responses, recommendation documents, user feedback, and activity logs. Chosen for its schema flexibility and fast querying of document-based data.
- **Data Lake** (Optional): `Amazon S3` or local `HDFS` bucket for storing raw bulk data (e.g., climate archives) if needed.

Interaction: Data from ingestion flows into `PostgreSQL` for relational entities and into `MongoDB` for flexible or nested documents. These stores are later accessed by the processing and application layers.

4. Data Processing & Intelligence Layer

Role: Transforms raw data into useful insights. Executes `ML` models, risk analysis, climate forecasting, and generates recommendations.

Components & Technologies:

- **Apache Spark:** Performs batch processing on climate data, joins datasets from multiple sources, applies complex transformations.

– **Internal Processing Scripts:** Handle logic for alert generation, anomaly detection, and cleaning pipelines.

Interaction: Reads input data from `PostgreSQL` and `MongoDB` → runs models and transformations → writes back results to `PostgreSQL` (structured outputs like predictions) or `MongoDB` (rich recommendations and alerts).

5. Business Intelligence (BI) Module

Role: Provides data visualization and decision-making tools for analysts and administrators.

Components & Technologies:

– **Metabase / Apache Superset** Dashboarding and analytics tools connected to `PostgreSQL` and `MongoDB` for generating graphs, `KPIs`, reports.

Interaction: `BI` tools query `PostgreSQL` for analytics tables or materialized views and query `MongoDB` for custom visualizations or log-based dashboards.

6. Application & Serving Layer

Role: Serves processed data to external applications and end-users through interfaces and `APIs`. Implements business logic and manages user access.

Components & Technologies:

– **FastAPI** (Python): `REST/GraphQL APIs` expose data from both `PostgreSQL` and `MongoDB` securely.

– **Backend Services:** Handle user session management, request routing, access control, and processing `API` calls.

– **Redis** (optional): Used for caching frequently accessed predictions or alerts.

Interaction: `APIs` call `PostgreSQL` (for structured data like user profiles and prediction logs) and `MongoDB` (for dynamic recommendation responses). `Redis` may cache common or urgent responses for performance.

7. Frontend / Access Layer

Role: This is the interface through which users interact with the system – whether via web dashboards or mobile devices.

Components & Technologies:

– **React** Responsive `UIs` for displaying predictions, recommendations, alerts, and reports.

– **JWT / OAuth2:** Authentication and authorization systems ensure secure access per role (e.g., farmer, analyst, admin).

Interaction: The frontend communicates with the `API` layer → receives processed data from `PostgreSQL` or `MongoDB` → renders information tailored to the user's location, role, and preferences.