Name – Puspak Chakraborty, Roll No. – DA24S002

Details of some blocks of architecture diagram

1. User Interface (UI)

Purpose:

Provides the main interaction point for users to draw, submit, and label doodles.

Responsibilities:

- Offers a drawing canvas for users to create doodles.
- Encodes drawn images (28x28 grayscale, flattens it first to 784 dimensional vector) as base64 and sends them to the backend for prediction or labeling.
- Displays classification results to the user.
- Allows users to correct/label doodles if the prediction is incorrect, sending labeled data to the backend for retraining.

Key Interactions:

- Sends POST requests to the FastAPI backend (/invocations/ for prediction, /save-image/ for labeling).
- Receives prediction results and confirmation messages.
- Runs on port 9080 in its own container.

2. FastAPI Backend

Purpose:

Acts as the central API server, handling all communication between the UI, model server, and storage.

Responsibilities:

- Exposes REST API endpoints for image classification (/invocations/) and image saving (/save-image/).
- Preprocesses and validates incoming image data.
- Forwards prediction requests to the MLflow model server and returns results to the UI.
- Saves labeled images to persistent storage (organized by category).
- Exposes a /metrics endpoint for Prometheus monitoring.
- Logs all actions and errors for audit and debugging.

Key Interactions:

- Receives requests from the UI.
- Forwards prediction requests to the MLflow Model Server.
- Writes images to disk for retraining.
- Exposes metrics to Prometheus.
- Runs on port 8000 in its own container.

3. MLflow Model Server

Purpose:

Hosts the latest trained doodle classifier model and serves predictions.

Responsibilities:

- Loads and serves the most recent model registered via the retraining pipeline.

- Accepts POST requests at /invocations/ with image data and returns predicted class probabilities.
- Can be updated/restarted as new models are registered.
- Runs on port 5002 (can be containerized or run on host).

Key Interactions:

- Receives prediction requests from the FastAPI backend.
- Returns prediction results as JSON.

4. Persistent Storage (Image Data)

Purpose:

Stores user-labeled doodle images for future retraining.

Responsibilities:

- Organizes images by category in a directory structure.
- Ensures data is available for the retraining pipeline.
- Supports both new data (awaiting retraining) and merged datasets.

Key Interactions:

- Written to by the FastAPI backend when users save labeled doodles.
- Read by the retraining pipeline during data ingestion and preprocessing.

5. Retraining Pipeline

Purpose:

Automates the process of updating the model as new labeled data accumulates.

Responsibilities:

- Orchestrated by pipeline_orchestrator.py, which runs each stage as a subprocess.
- Stages include: scanning for new data, converting images, merging datasets, versioning with DVC/Git, retraining the model, registering the new model with MLflow, and cleaning up empty folders.
- Tracks pipeline run and stage status in a PostgreSQL database for monitoring and reproducibility.
- Logs outputs and errors for each stage.
- Ensures robust error handling and graceful halting on failure or insufficient data.

Key Interactions:

- Reads new and existing data from persistent storage.
- Registers new models with the MLflow Model Server.
- Updates pipeline run status in the database.

6. Pipeline Viewer (Dashboard)

Purpose:

Provides visibility into the status and progress of the retraining pipeline.

Responsibilities:

- Backend (FastAPI) exposes /pipeline_runs endpoint, fetching run/stage status from PostgreSQL.
- Frontend (HTML/JS) displays a table of pipeline runs and their stage statuses, updating periodically.

- Allows users and operators to monitor retraining activity, spot failures, and audit model updates.

Key Interactions:

- Fetches data from the pipeline run database.
- Displays data in the browser for human monitoring.

7. Monitoring Stack (Prometheus & Grafana)

Purpose:

Ensures observability and operational health of the system.

Responsibilities:

- **Prometheus:** Scrapes metrics from the FastAPI backend (/metrics) and Windows Exporter (system metrics).
- **Grafana:** Visualizes API and system metrics (requests, latency, api errors, CPU, memory, disk, network).
- **Windows Exporter:** Runs on host, exposes system-level metrics for Prometheus.

Key Interactions:

- Prometheus scrapes metrics endpoints on FastAPI and Windows Exporter.
- Grafana queries Prometheus to build dashboards.
- Operators use Grafana dashboards for real-time monitoring and alerting.

8. Database (PostgreSQL)

Purpose:

Tracks the status and history of retraining pipeline runs.

Responsibilities:

- Stores metadata for each pipeline run: run ID, start time, per-stage status and timestamps.
- Used by both the pipeline orchestrator (to update status) and the pipeline viewer (to display status).
- Schema is managed by db_util.py.

Key Interactions:

- Written to by the retraining pipeline orchestrator.
- Read by the pipeline viewer backend.

9. Containerization and Orchestration (Docker Compose)

Purpose:

Manages deployment, scaling, and networking of all major services.

Responsibilities:

- Defines services for UI, backend, Prometheus, Grafana, and possibly the pipeline viewer.
- Exposes necessary ports and mounts volumes for persistent data.
- Sets environment variables for configuration.
- Ensures all services can communicate over a Docker network.