DA24M016
DA24M017

Low-Level Design (LLD) – ALL Classifier

---

1 Source Tree

```
.
├── airflow/dags/
│   └── all_classification_pipeline.py
├── backend/
│   ├── main.py
│   ├── model_utils.py
│   └── Dockerfile
├── frontend_streamlit/
│   ├── app.py
│   └── Dockerfile
├── monitoring/
│   ├── prometheus.yml
│   └── grafana/provisioning/
└── docker-compose.*.yml
```

2 FastAPI Specification

Path	Method	Request	Response	HTTP Code
/predict	POST	multipart/form-data {file: image}	{ probabilities: float[4], predicted_class: int }	200
/feedback	POST	multipart/form-data { correct_label:str, predicted_label:str, file:image }	{ stored:bool, feedback_count:int }	200
/feedback/count	GET	–	{ feedback_count:int }	200
/metrics	GET	–	Prometheus exposition format	200

/ping	GET	–	{status:"ok"}	200

Exception Handling

```
@app.exception_handler(Exception)
async def fallback_ex(exc:Exception, req:Request):
    logger.exception("unhandled error")
    return JSONResponse({"detail":"server error"}, status_code=500)
```

3 SQLite Schema

```
CREATE TABLE feedback(
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  ts REAL,              -- unix timestamp
  correct TEXT,         -- ground-truth class
  predicted TEXT,       -- model prediction at time
  img BLOB              -- orig. JPEG/PNG bytes
);
```

DB file path: `${FEEDBACK_DIR:-/app/data}/feedback.db` (volume-mounted in prod stack for extraction).

4 Airflow DAG – `all_classification_pipeline`

Task-id	Operator	Callable / Cmd	Key output
`preprocess_images`	PythonOperator	`resize_images(...)`	resized images → `/data/processed`
`prepare_data`	PythonOperator	`make_manifest(...)`	CSV manifests
`train_classifier`	PythonOperator	`train_classifier(...)`	`run_id` (XCom)
`test_classifier`	PythonOperator	`test_classifier(..., run_id)`	metrics
`register_model`	PythonOperator	`register_model(..., run_id)`	MLflow model reg.

Retrain-on-Feedback Sensor

```
from airflow.sensors.python import PythonSensor


def enough_feedback(**_):
    conn = sqlite3.connect('/app/data/feedback.db')
```

```
        cnt  = conn.execute('SELECT COUNT(*) FROM feedback').fetchone()[0]
        return cnt >= 100

    wait_feedback = PythonSensor(
        task_id     = 'wait_feedback',
        python_callable = enough_feedback,
        poke_interval   = 3600,   # hourly
        timeout         = 86400,  # 1 day window
    )
    wait_feedback >> preprocess_images
```

(In dev DAG only; omitted from client-facing stack)

5 Monitoring Metrics

Metric	Type	Labels	Purpose
`http_requests_total`	Counter	method, handler, status	requests volume
`http_request_duration_seconds_bucket`	Histogram	handler	latency P50/P95
`process_cpu_seconds_total`	Counter	–	CPU usage
`node_memory_MemAvailable_bytes`	Gauge	–	free mem

Grafana-ready dashboard JSON (`monitoring/grafana/dashboards/all-api.json`) provisioned on container start.

6 Docker Images

Image	Base	Entrypoint
backend	python:3.9-slim	uvicorn main:app --host 0.0.0.0 --port 8000
streamlit	python:3.9-slim	streamlit run app.py --server.port 8501 --server.address 0.0.0.0
mlflow	python:3.9-slim	mlflow server --backend-store-uri sqlite:///mlruns.db ...
prometheus	prom/prometheus:latest	/bin/prometheus --config.file=/etc/prometheus/prometheus.yml

7 Environment Variables Summary

Variable	Default	Used By	Meaning
BACKEND_URL	http://backend:8000	Streamlit	API base for requests

`FEEDBACK_DIR`	`/app/data`	Backend	where to put `feedback.db`
`FB_THRESH`	`100`	Airflow sensor	min samples before retrain

(end of LLD)