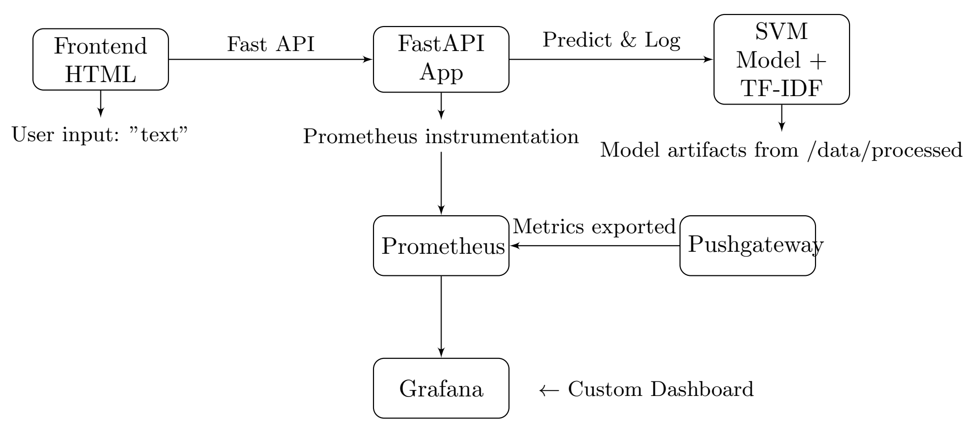
Twitter Sentiment Analysis System

Complete Technical Documentation

Contents

1 Architecture Diagram and Explanation



1.1 Component Explanations

Frontend HTML Web interface that collects user text input and sends it to the backend via REST API calls.

FastAPI App Python-based web framework that handles incoming requests, processes data, and coordinates with the ML model. Also instruments code for monitoring metrics.

SVM Model + TF-IDF Machine learning classification system that uses Support Vector Machine with TF-IDF text vectorization to process and classify text inputs.

Prometheus Time-series database and monitoring tool that collects and stores metrics from the application.

Pushgateway The purpose of using Prometheus Pushgateway is to enable pushing custom metrics (like data drift) from short-lived or background jobs into Prometheus.

Grafana Data visualization platform that creates dashboards and alerts based on metrics from Prometheus.

1.2 Data Flow

1. User submits text through the frontend interface
2. FastAPI receives the request and forwards it to the SVM model
3. Model processes text using TF-IDF vectorization and makes predictions
4. Prediction results are returned to the user and logged
5. System performance metrics are collected by Prometheus instrumentation
6. Metrics are exported through Pushgateway to Prometheus
7. Grafana visualizes system metrics through custom dashboards

2 High-Level Design Document (HLD)

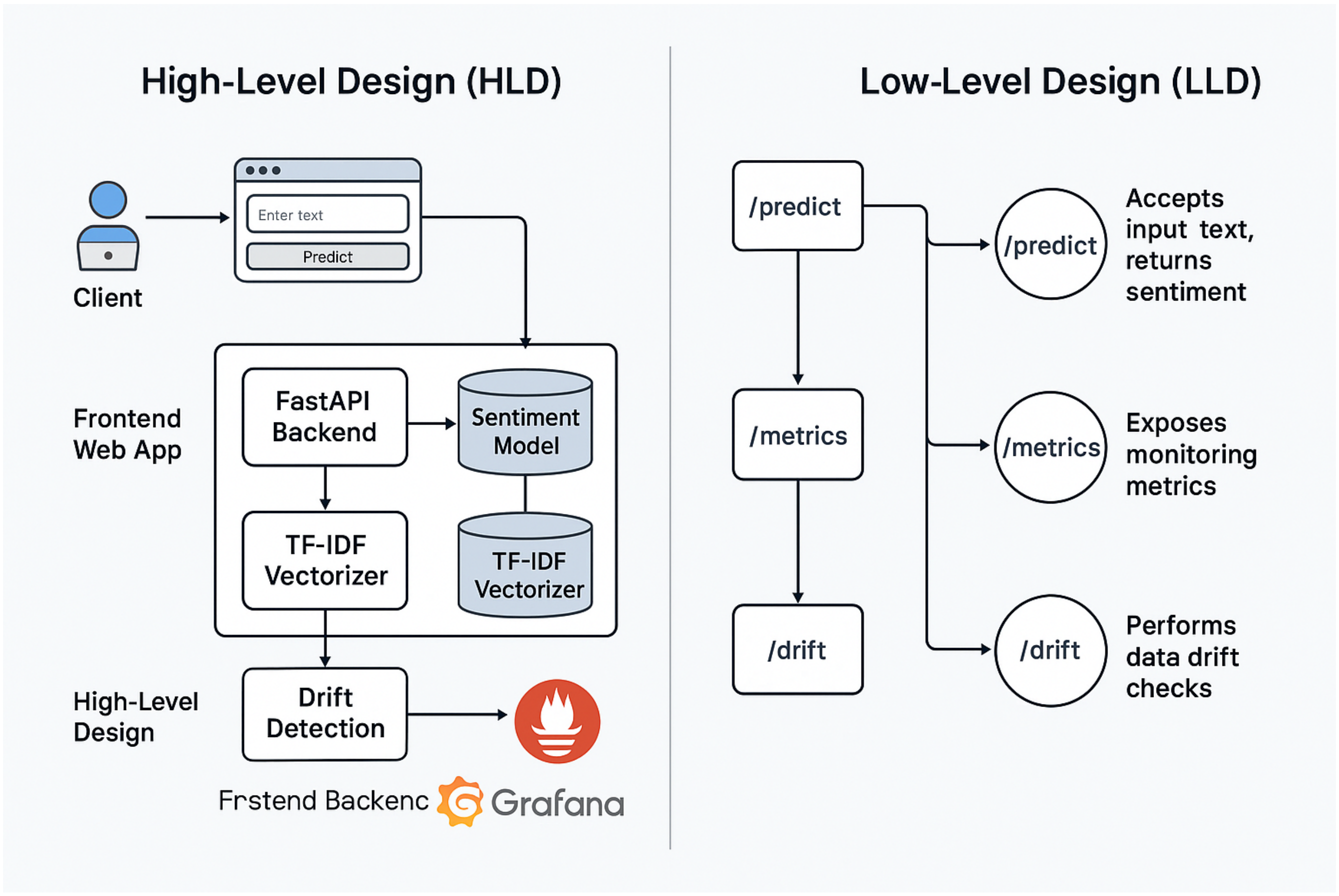


Figure 1: High-Level Design (HLD) and Low Level Design (LLD) Diagram

2.1 Key Design Decisions

- **Model**: Lightweight SVM + TF-IDF chosen for high accuracy and fast deployment.
- **Framework**: FastAPI for RESTful serving.
- **Monitoring**: Prometheus & Grafana used for API metrics + drift monitoring.
- **ML Tracking**: MLflow tracks experiments, metrics, and artifacts.
- **Data Engineering**: Preprocessing and pipeline tracked via DVC.
- **Drift Detection**: Cosine similarity between recent and training text distributions.
- **Deployment**: Docker & Docker Compose to containerize the app and monitoring.

3 Low-Level Design (LLD)

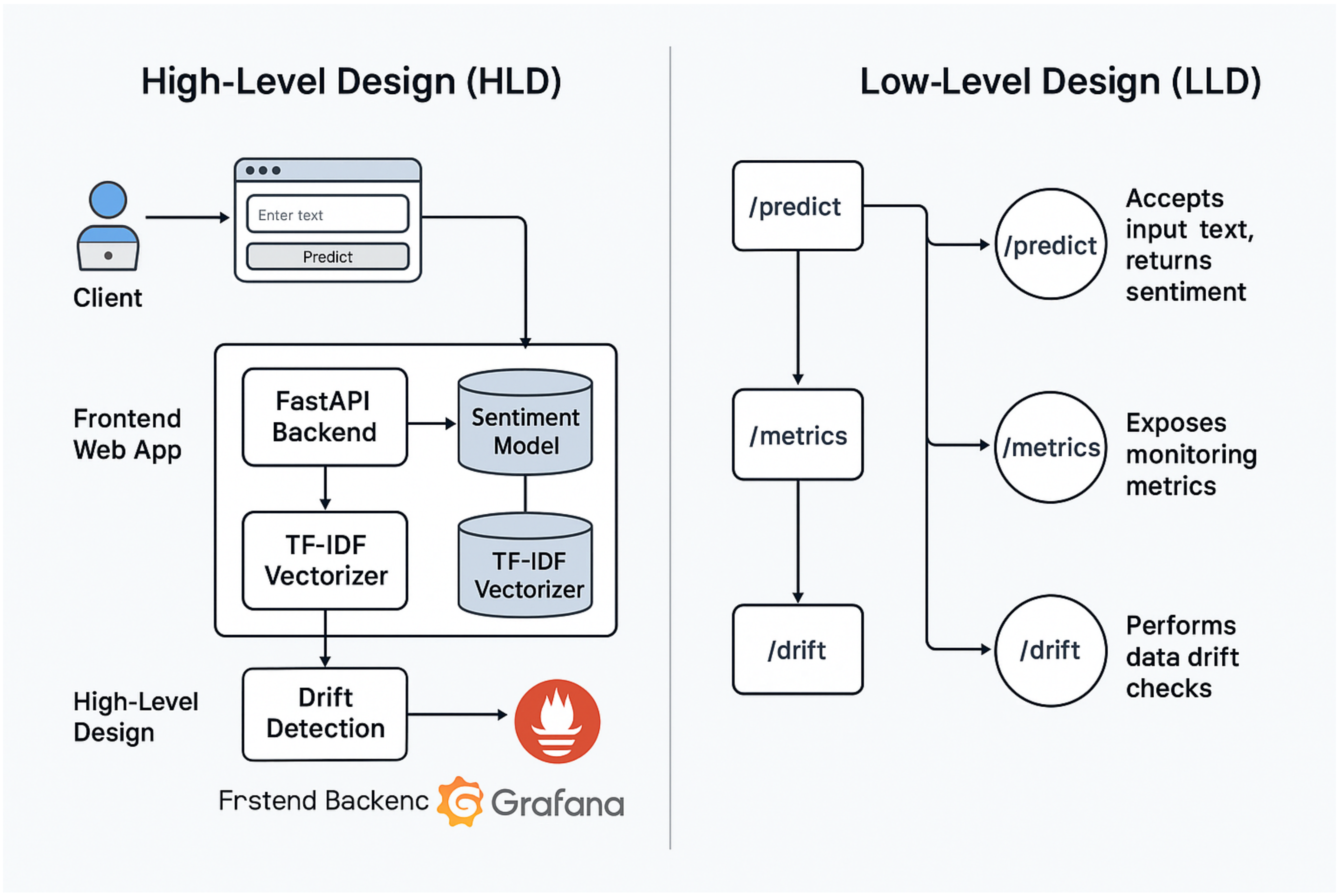


Figure 2: High Level Design (HLD) and Low-Level Design (LLD) Diagram

3.1 API Endpoints

Endpoint	Method	Input	Output
/	GET	—	HTML page
`/predict`	POST	`text` (Form)	HTML Response
`/metrics`	GET	—	Prometheus

Table 1: API Endpoint Specifications

/ Renders prediction form.

/predict Predicts sentiment, logs result.

/metrics Exposes metrics for scraping.

3.2 File Paths

Component	Path
Model	`data/processed/svm_model.pkl`
Vectorizer	`data/processed/tfidf_vectorizer.pkl`
Preprocessing Script	`src/data/preprocessing.py`
Inference API	`src/deployment/api.py`
Drift Detection	`src/drift/drift_detection.py`

Table 2: File Path Specifications

4 Test Plan & Test Cases

4.1 Scope

- Data cleaning
- Model prediction
- Integration

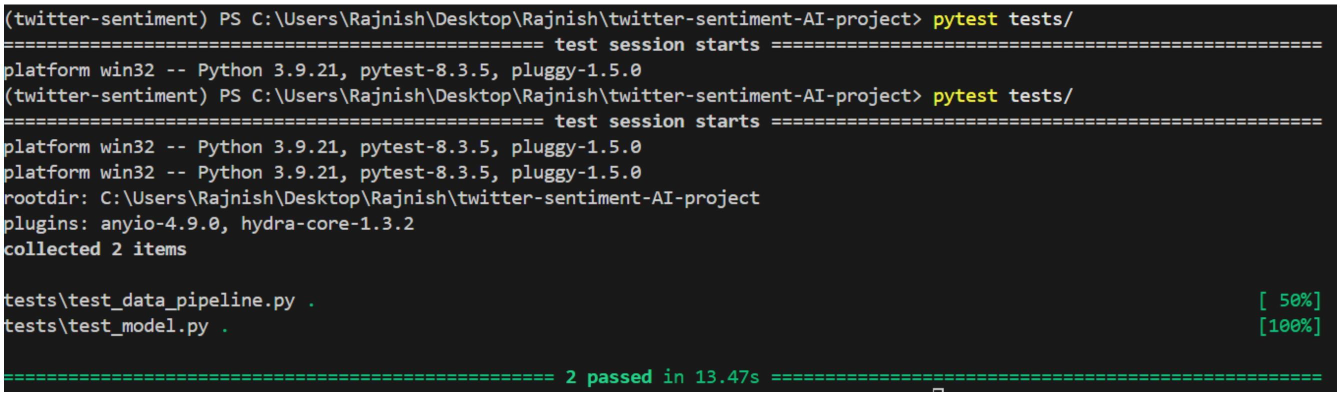
4.2 Test Cases

Test	Status	Location
Clean tweet text		`tests/test_data_pipeline.py`
Model loads and predicts		`tests/test_model.py`
Drift detection computation		`drift_detection.py` test run
FastAPI response (manual)		via browser/Postman

Table 3: Test Case Status

4.3 Run Command

```
pytest tests/
```



```
(twitter-sentiment) PS C:\Users\Rajnish\Desktop\Rajnish\twitter-sentiment-AI-project> pytest tests/
================================================ test session starts ================================================
platform win32 -- Python 3.9.21, pytest-8.3.5, pluggy-1.5.0
(twitter-sentiment) PS C:\Users\Rajnish\Desktop\Rajnish\twitter-sentiment-AI-project> pytest tests/
================================================ test session starts ================================================
platform win32 -- Python 3.9.21, pytest-8.3.5, pluggy-1.5.0
platform win32 -- Python 3.9.21, pytest-8.3.5, pluggy-1.5.0
rootdir: C:\Users\Rajnish\Desktop\Rajnish\twitter-sentiment-AI-project
plugins: anyio-4.9.0, hydra-core-1.3.2
collected 2 items

tests\test_data_pipeline.py .                                                                                  [ 50%]
tests\test_model.py .                                                                                          [100%]

================================================= 2 passed in 13.47s ================================================
```

Figure 3: Unit Testing

5 User Manual (Non-Technical)

5.1 Step-by-Step Usage

1. **Access App**
   Open browser and go to: `http://localhost:8000`

2. **Enter Tweet**
   - Type or paste a tweet.
   - Example: `"I love this product!"`

3. **Click Predict**
   - Wait a second for the result.
   - You'll see: `Sentiment: Positive` or `Negative`

4. **Monitoring**
   - Open Prometheus: `http://localhost:9090`
   - Open Grafana: `http://localhost:3000` → View dashboards.

5. **Retrain**
   - Drift auto-detected every 20s.
   - Triggers `dvc repro` if score is greater than threshold.