Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the core architecture and implementation details for **TrendPilot - AI Market Sentiment Explorer**. TrendPilot is a platform leveraging generative AI to analyze, summarize, and visualize market sentiment from diverse sources, providing finance professionals and researchers with actionable insights.

---

1. System Components

Component	Technology	Key Responsibilities
Data Ingestion Service	Python, Docker	Fetch, parse, and normalize data from sources
AI Analysis Engine	Python, GenAI	Sentiment analysis, summarization, trend detection
Data Store	Elasticsearch	Store processed data, support search/aggregation
API Backend	Python (FastAPI)	Expose REST APIs, auth, business logic
Frontend Dashboard	React	Interactive UI, data visualization, user management
Auth Service	Python, JWT	User authentication and profile management

---

2. Class/Interface Overview

Class/Interface	Description	Key Methods/Attributes
`DataIngestor`	Handles data fetching/parsing	`fetch()`, `parse()`, `normalize()`
`SentimentAnalyzer`	AI-based sentiment & trend analysis	`analyze(text)`, `summarize(text)`
`TrendAggregator`	Aggregates sentiment over time	`aggregate(period)`, `get_trends()`
`UserProfile`	User data and preferences	`id`, `email`, `roles`, `settings`
`SentimentRecord`	Data model for sentiment entries	`source`, `timestamp`, `score`, `summary`
`APIController`	API endpoints for frontend	`getSentiment()`, `getTrends()`, `login()`

Relationships:

- `DataIngestor` → feeds data to `SentimentAnalyzer`
- `SentimentAnalyzer` → outputs to `TrendAggregator` & `SentimentRecord`
- `APIController` → interfaces with all backend services

---

3. Data Structure Overview

Model	Fields
`SentimentRecord`	`id`, `source`, `timestamp`, `raw_text`, `score`, `summary`, `tags`
`UserProfile`	`id`, `email`, `hashed_password`, `roles`, `settings`
`TrendData`	`period`, `average_score`, `top_keywords`, `change_percent`

---

4. Algorithms/Logic

Sentiment Analysis & Trend Aggregation (Pseudocode):

```
for item in incoming_data:
    sentiment = SentimentAnalyzer.analyze(item.text)
    summary = SentimentAnalyzer.summarize(item.text)
    record = SentimentRecord(
        source=item.source,
        timestamp=item.timestamp,
        score=sentiment.score,
        summary=summary
    )
    DataStore.save(record)

def aggregate_trends(period):
    records = DataStore.query(period=period)
    avg_score = mean([r.score for r in records])
    top_keywords = extract_keywords([r.summary for r in records])
    return TrendData(period, avg_score, top_keywords)
```

---

5. Error Handling

Scenario	Handling Approach
Data source unavailable	Retry with backoff, log, alert if persistent
AI model inference failure	Fallback to default summary, log error
Data store write/read error	Retry, log, escalate if repeated
Invalid user input/API request	Return 400/422 error, validation message
Auth failure/expired token	Return 401/403, prompt re-authentication

---

End of Document