Low Level Design Document

This Low Level Design (LLD) document details the implementation plan for **MediaPulse - Trending Content Analyzer**. MediaPulse is a Python-based tool enabling users to analyze and visualize trending media topics via keyword/hashtag input, with a focus on beginner-friendly analytics and a futuristic UI.

1. System Components

Component	Description	Key Responsibilities
UI Layer	User interface (web/local app)	Input, display charts/analytics, theme
Data Fetcher	Data retrieval (mocked/public API)	Fetch/generate time-series trend data
Data Processor	Data cleaning and aggregation (pandas)	Prepare data for visualization/analytics
Visualization Engine	Chart rendering (matplotlib/plotly)	Generate interactive trend charts
Analytics Module	Basic statistics computation	Calculate peak, average, trend direction

2. Class/Interface Overview

Class/Module	Key Methods/Attributes	Relationships
`MediaPulseApp`	`run()` , `render_ui()`	Main entry/UI controller
`DataFetcher`	`fetch_data(keyword: str) -> DataFrame`	Used by DataProcessor
`DataProcessor`	`clean(df)` , `aggregate(df)` , `get_stats(df)`	Uses pandas, feeds Analytics
`ChartRenderer`	`plot_trend(df) -> Figure`	Consumes processed data
`AnalyticsSummary`	`compute_peak(df)` , `compute_avg(df)` , `compute_trend(df)`	Used by UI

Example Method Signatures:

```python
class DataFetcher:
    def fetch_data(self, keyword: str) -> pd.DataFrame

class DataProcessor:
    def clean(self, df: pd.DataFrame) -> pd.DataFrame
    def aggregate(self, df: pd.DataFrame) -> pd.DataFrame

class AnalyticsSummary:
    def compute_peak(self, df: pd.DataFrame) -> int
    def compute_avg(self, df: pd.DataFrame) -> float
    def compute_trend(self, df: pd.DataFrame) -> str
```

---

3. Data Structure Overview

Field	Type	Description
date	string	Date in YYYY-MM-DD format
keyword	string	User-input keyword/hashtag
count	int	Popularity count for the date

Sample Data:

```
[
  {"date": "2024-06-01", "keyword": "AI", "count": 120},
  {"date": "2024-06-02", "keyword": "AI", "count": 150}
]
```

---

4. Algorithms / Logic

Main Flow:

```
def analyze_keyword(keyword):
    data = DataFetcher().fetch_data(keyword)
    clean_data = DataProcessor().clean(data)
    agg_data = DataProcessor().aggregate(clean_data)
    chart = ChartRenderer().plot_trend(agg_data)
    stats = AnalyticsSummary().compute_all(agg_data)
    MediaPulseApp().render_ui(chart, stats)
```

Trend Direction Logic:

- If last count > first count: ↑ (upward)
- If last count < first count: ↓ (downward)
- Else: → (stable)

---

5. Error Handling

Scenario	Handling Approach
Invalid/empty keyword input	Show UI error message, prompt re-entry
Data fetch failure	Display “No data available” message
Data processing error	Log error, show generic error in UI

Chart rendering failure	Fallback to static chart or error message
UI rendering issues	Show minimal error banner, log details

End of Document