**Sprint Plan for LLM-Driven AI Analytics System**

**Sprint 1: Data Ingestion, Text Processing & LLM Integration**

**Goal**: Set up the data ingestion pipeline, integrate an open-source LLM for text analysis, and provide APIs for basic analytics.

**Tasks:**

1. **Data Ingestion**:
   * Create a pipeline to ingest unstructured data (e.g., reports, reviews, articles) into the system.
   * Use **FastAPI** for building REST APIs that allow uploading and storing text data.
   * Store text documents in a **NoSQL database** like **MongoDB** (free-tier, open-source).
2. **Preprocessing & Text Cleaning**:
   * Clean and preprocess the text data (e.g., remove special characters, lowercase, stopword removal, tokenization).
   * Use **SpaCy** or **NLTK** for tokenization and text preprocessing.
3. **LLM Integration**:
   * Integrate **Hugging Face Transformers** (open-source models such as GPT-3.5, GPT-Neo, or BERT) for advanced text analytics.
   * Implement APIs to perform tasks such as sentiment analysis, named entity recognition (NER), topic modeling, and summarization using the transformer models.
4. **Basic Analytics & Insights**:
   * Perform sentiment analysis on the ingested data using pre-trained models (e.g., **DistilBERT** for sentiment).
   * Use **TextBlob** (open-source Python library) for polarity and subjectivity analysis.
   * Generate summaries of long documents using the LLM's text generation capabilities.
   * Implement FastAPI endpoints for querying the processed text and obtaining basic analytics.

**Deliverables (Sprint 1):**

* REST API for ingesting, storing, and preprocessing text data.
* Integrated LLM for performing basic analytics (e.g., sentiment analysis, summarization).
* API endpoints for querying processed text analytics.

**Tools & Stack (Sprint 1):**

* **Backend**: FastAPI, MongoDB (NoSQL).
* **LLM**: Hugging Face Transformers (GPT, BERT).
* **Preprocessing**: SpaCy, NLTK, TextBlob.
* **Deployment**: Local or cloud-based (Heroku Free Tier).

**Sprint 2: Advanced Text Analytics & Visualization**

**Goal**: Develop advanced analytics capabilities and visualizations for the LLM-generated insights.

**Tasks:**

1. **Advanced Analytics**:
   * Implement topic modeling using **LDA (Latent Dirichlet Allocation)** or transformer-based topic clustering.
   * Extract named entities (persons, organizations, locations) using NER models (e.g., **spaCy** or **Hugging Face’s BERT NER**).
   * Perform more in-depth sentiment analysis with polarity and subjectivity scores for specific sections of the text.
2. **LLM-Generated Insights**:
   * Enable **question-answering** over text data using models like **T5** or **GPT-Neo** (open-source).
   * Implement text generation features for summarizing key insights or responding to specific queries based on the data.
3. **Visualization Dashboard**:
   * Build a **data visualization dashboard** using **Dash** (open-source Python web app framework) or **Streamlit**.
   * Visualize key insights like:
     + Sentiment analysis trends over time.
     + Word clouds of common topics.
     + Named entity distributions (e.g., organizations mentioned frequently).
   * Use **Plotly** or **Matplotlib** (open-source libraries) for data visualization.
4. **Embedding Analytics**:
   * Visualize document similarity using embeddings created by transformer models.
   * Use **UMAP** or **t-SNE** for dimensionality reduction and clustering of document embeddings.
   * Integrate **FAISS** for fast retrieval of similar documents based on embedding distances.

**Deliverables (Sprint 2):**

* Advanced text analytics features (topic modeling, NER, question-answering).
* Interactive visualization dashboard showing insights (sentiment, entities, topics).
* Embedding-based analytics for document similarity.

**Tools & Stack (Sprint 2):**

* **Backend**: FastAPI, Hugging Face Transformers, FAISS.
* **Visualization**: Dash/Streamlit, Plotly/Matplotlib.
* **Topic Modeling**: LDA, UMAP/t-SNE.
* **Deployment**: Heroku, GitHub Pages.

**Sprint 3: User Interaction, Export, and Reporting**

**Goal**: Build a user-friendly frontend for interacting with the analytics system and implement export functionality for generating reports.

**Tasks:**

1. **Frontend Interface**:
   * Build a simple web-based interface using **HTML5**, **CSS3**, and **JavaScript**.
   * Allow users to upload text files or input queries for analysis.
   * Provide a search bar to query insights or ask questions about the data.
2. **Dashboard & Interaction**:
   * Display results from LLM-driven analysis (e.g., sentiment, topics, entities) on the frontend.
   * Allow users to select specific documents or sections for further analysis (highlighting named entities or key phrases).
3. **Export Analytics Reports**:
   * Implement functionality to export analytics results as **PDF** or **DOCX** reports using **pdf-lib** or **docx.js** (open-source libraries).
   * Allow export of raw results (e.g., sentiment scores, NER data) as **JSON** or **CSV** files.
4. **Real-time Updates**:
   * Use **WebSockets** in FastAPI for real-time updates on long-running analysis tasks.
   * Display progress bars for document ingestion, processing, and analysis in the UI.
5. **Authentication**:
   * Implement **JWT (JSON Web Tokens)** or **OAuth2.0** for user authentication and authorization.
   * Allow users to create accounts and save their analysis history or results.
6. **Testing**:
   * Write unit tests for backend and frontend using **Pytest** for Python and **Jest** for JavaScript.
7. **Deployment to Cloud**:
   * Deploy the backend to **Heroku** or **Railway** (free-tier platforms).
   * Host the frontend using **GitHub Pages** or **Netlify** for static hosting.

**Deliverables (Sprint 3):**

* Fully interactive frontend for AI analytics system.
* Exportable reports in PDF/DOCX format.
* Real-time notifications for analysis progress.
* User authentication and history saving.
* Fully deployed system on free-tier cloud platforms.

**Tools & Stack (Sprint 3):**

* **Frontend**: HTML5, CSS3, JavaScript, WebSockets.
* **Export**: pdf-lib, docx.js.
* **Authentication**: OAuth2.0/JWT via Authlib.
* **Testing**: Pytest, Jest.
* **Deployment**: Heroku, GitHub Pages, Netlify.