Introduction to Software Engineering (Global Edition)

Summary:

- Provides a foundational understanding of what software engineering is: the **systematic application of engineering principles** to software development.
- Covers:
    - **Software process models** (e.g., Waterfall, Agile, Incremental).
    - **Software project management** basics: planning, risk, and configuration management.
    - **Professional responsibilities** like ethics, software safety, and legal issues.
- Emphasizes the need for **engineering discipline** in software, balancing user needs, cost, and reliability.
- Discusses **evolving nature** of software systems and the challenges of **maintenance** and **scalability**.
- Introduces **key terms** like requirements, design, implementation, validation, and evolution.

✅ A great high-level primer for students or professionals new to structured software development.

Focus: Foundation of software engineering, process models, and professional responsibilities.

🔍 Key Concepts and Their ClinicTrends AI Relevance:

Concept	Explanation	ClinicTrends AI Application
Software Process Models	Waterfall, Agile, etc. define how development flows.	You are working iteratively: adding sentiment, now topics, next insights — that's **Agile + Incremental**.
Requirements Engineering	Capturing what the system should do.	You identified clear functional needs: "upload CSV", "detect negative scores", "generate reports".

Software evolution	Software must adapt to changing needs.	You already evolved the system: from sentiment-only to BERTopic integration, and soon auto-insight generation.
Validation & Verification	Testing and confirming the software meets goals.	Your real-time feedback on UI output (word clouds, alerts) is a live form of verification.
Professional Ethics & Quality	Delivering systems that are reliable, non-biased, and useful.	You're building something **used to make business decisions**. Keeping models interpretable (e.g. via SHAP or topic labels) supports ethical transparency.

Final Thought

You're already intuitively applying many of these LLD and SE principles — just without the formal wrapper. If you ever want to **present this project professionally** (e.g., in a portfolio or master's submission), you can **reverse-map your components** to these principles and show how you implemented:

- **Component-level encapsulation**
- **Low coupling / high cohesion**
- **Iterative agile model**
- **Ethical, testable, and user-centric software design**