CppStandards_Part2.pdf — *Design Style*

🔑 Key Themes:

- **Correctness, simplicity, and clarity** > performance hacks
- **Avoid premature optimization** but be mindful of **scalability**
- **One responsibility per entity** (function, class, etc.)
- **Information hiding**: reduce dependencies and expose abstractions
- **Minimize shared/global state**; prefer communication (message passing)
- **RAII & smart pointers**: automate resource management
- **Concurrency awareness**: know when and how to handle multithreaded access

🧠 Relevance to *ClinicTrendsAI*:

Concept	ClinicTrendsAI Application
Simplicity first	Your choice to refactor views and pipeline modules improves clarity and long-term maintainability.
One responsibility per function	Keep `train_tfidf_model` and views decoupled — avoid bloated logic inside single functions.
Avoid premature optimization	Focus on clean logic and business insights first; only optimize when you identify bottlenecks (e.g. BERTopic runtime).
Hide internal complexity	Keep user-facing functions and Streamlit views clean; do the heavy lifting in backend logic.
Plan for scalability	With growing datasets (customer feedback), choose linear or logarithmic algorithms for vectorization and classification.
Thread safety & concurrency	If future versions use multiprocessing (e.g. for model training), avoid shared state and ensure thread-safe design.