Semantic Book Recommender System

1**. Project Description**

The Semantic Book Recommender System is an AI-powered application that recommends books based on a user's input description, optionally filtered by category and emotional tone. It leverages text embeddings and vector similarity search to understand user intent at a semantic level rather than relying on keyword matches. The recommender uses either a cloud-based embedding API (Gorq) or falls back to a local HuggingFace model (all-MiniLM-L6-v2) for vector generation.

The application provides an intuitive and visually appealing Gradio-based web interface that allows users to:

- Enter a brief description of the kind of book they want.
- Select a genre/category.
- Select a preferred emotional tone.
- View recommended books with thumbnails and summaries.

2. Functional Requirements

2.1 User Input

Text input box for the user to describe a book.

Dropdown to choose a book category (genre).

Dropdown to choose an emotional tone (e.g., Joy, Sadness, Fear).

2.2 Recommendation Engine

Parse and clean tagged_description.txt containing ISBN-tagged descriptions.

Generate vector embeddings for all book descriptions.

Perform similarity search using a vector database (ChromaDB).

Apply optional filters on category and emotional tone.

Return top k book recommendations (default: 16).

2.3 Book Metadata Processing

Load and validate metadata from books_with_emotions.csv.

Map ISBNs from semantic search results to book metadata.

Format author names and handle missing thumbnails with a default image.

2.4 UI Output

Display results in a gallery with book cover, title, author, and a 30-word summary.

Return fallback message if no books are found or input is too short.

3. Non-Functional Requirements

1. Fallback Embeddings: If the cloud embedding API is unavailable, switch to a local HuggingFace model automatically.
2. Performance: Load embeddings and build vector database on startup for fast querying.
3. Resilience: Handle missing or malformed data files gracefully with logging and exception handling.
4. Scalability: Can be extended to use larger models, more metadata fields, or plug into an online book API.
5. Note: API keys are hard-coded(Future improvements will include API keys stored in .env and not hard-coded).

4. Technical Stack

Component	Technology
UI	Gradio (gr.Blocks)
Embeddings	HuggingFace (local) / Gorq API (cloud)
Vector Search	ChromaDB
Data	CSV (books_with_emotions.csv), TXT (tagged_description.txt)
Environment	Python 3.x, dotenv, pandas, socket, requests
Logging	Python logging module

5. Directory & File Structure

```
project/
├── dashboard.py             # Main application and Gradio interface
├── tagged_description.txt      # ISBN-tagged semantic descriptions
├── books_with_emotions.csv      # Book metadata with emotion scores
├── cover-not-found.jpg        # Placeholder cover image
├── .env                       # Environment variables (GORQ_API_KEY etc.)
```

6. Future Enhancements

- Add user login and rating functionality.
- Improve emotion tagging with a dedicated sentiment model.
- Support for multilingual queries and metadata.
- Integrate with an online bookstore API to fetch real-time availability and pricing.
- API keys stored in .env and not hard-coded.