Backend Speed Optimization :

Let's modify the main.py file to implement lazy loading and optimize imports:

main.py

Key optimizations made:

1. Added lazy loading using @lru\_cache decorator
2. Implemented async startup event to load models in background
3. Moved model loading to separate functions
4. Added local caching for embeddings
5. Load components only when needed
6. Used asyncio to parallelize model loading

Additional steps to improve startup time:

1. Create a new file called .env in your backend directory:
2. Create a new directory for model caching:

mkdir e:\MyApp\backend\model\_cache

These changes will:

* Cache models locally to avoid redownloading
* Enable parallel tokenization
* Optimize TensorFlow logging
* Use lazy loading for components
* Load models asynchronously in the background

The startup time should be significantly reduced because:

1. Models are loaded only when needed
2. Background loading doesn't block the server startup
3. Local caching prevents redownloading of models
4. Parallel processing is enabled where possible

Furthure optimization:

Yes, the optimizations are working well! The startup time has significantly improved. Let's analyze what's happening:

1. Initial FastAPI startup is quick
2. Model and Scaler load quickly (cached)
3. The only slight delay is with the transformers/FAISS loading

To further optimize and clean up the warnings, let's make two small changes:

[**.env**](vscode-file://vscode-app/e:/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

Add environment variables to suppress warnings and optimize model loading:

.env

[**main.py**](vscode-file://vscode-app/e:/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

Update the embedding model initialization:

The startup sequence now looks good with:

1. Fast initial loading
2. Cached model loading
3. Parallel processing where possible
4. Proper error handling

The warning about TRANSFORMERS\_CACHE is suppressed and the startup time is optimized. No further changes are needed unless you want to add GPU support in the future.

**Clear Summary :**

Here's a clear summary of all the optimizations we've made to the backend:

1. **Lazy Loading Implementation**

* Added @lru\_cache() decorators for model loading
* Models are only loaded when first needed
* Results are cached for subsequent uses

1. **Asynchronous Loading**

* Implemented background loading using asyncio
* Added startup event handler for FastAPI
* Models load in parallel using asyncio.gather

1. **Environment Optimization**

[**.env**](vscode-file://vscode-app/e:/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

.env

1. **Model Loading Optimization**

* Added local model caching
* Set explicit CPU device usage
* Optimized batch size for embeddings
* Added error handling and logging

1. **Code Structure Changes**

Main.py

1. **Directory Structure**

e:\MyApp\backend\

├── model\_cache/ # Added for local model caching

├── vectorstore/ # FAISS database location

├── main.py # Optimized main file

├── .env # Environment configuration

├── deeplearning.keras # ML model

└── scaler.pkl # Scaler model

**Key Benefits Achieved:**

1. Faster startup time
2. Models load only when needed
3. Parallel loading of components
4. Local caching prevents redownloads
5. Better memory management
6. Cleaner error handling
7. Optimized TensorFlow and PyTorch settings
8. Suppressed unnecessary warnings
9. Better logging for debugging

Current startup sequence:

1. FastAPI initializes quickly
2. Models load in background
3. Cached components load instantly on subsequent runs
4. Transformers use local cache
5. FAISS loads with CPU optimization

This has significantly reduced the startup time from the original implementation while maintaining all functionality.