# LLMs Notes

## 1.1 Project Template

When developing end-to-end machine learning (ML) and artificial intelligence (AI) projects, particularly those involving large language models (LLMs), it is essential to follow a structured and modular approach. This ensures maintainability, scalability, and ease of collaboration. Here is an overview of a typical project template and the modular coding approach for such projects.

A screen shot of a computer

Description automatically generated

A screenshot of a diagram

Description automatically generated

### 1.2 Modular Coding Approach

1. **Separation of Concerns**: Each module should handle a specific part of the project (data handling, model architecture, training, etc.). This makes the code more readable and maintainable.
2. **Reusable Functions and Classes**: Write functions and classes that can be reused across different parts of the project. This avoids redundancy and keeps the code DRY (Don't Repeat Yourself).
3. **Configuration Management**: Use a central configuration file (config.py) to manage all the settings and parameters. This helps in keeping the code clean and makes it easy to change parameters without modifying the code.
4. **Logging**: Implement logging (logger.py) to keep track of events that happen when the software runs. This is crucial for debugging and monitoring the performance of the project.
5. **Testing**: Write unit tests and integration tests to ensure the correctness of the code. This is important for maintaining code quality and reliability.
6. **Documentation**: Properly document the code and provide a comprehensive README. This helps new developers understand the project quickly and makes it easier to maintain in the long run.

### Technologies Used

1. **Langchain**
2. **FastAPI**
3. **FAISS**
4. **ChromaDB**
5. **Pinecone**
6. **OpenAI/OpenSource**
7. **Embedding Model**

#### Langchain - Application

**Langchain** is a library designed to simplify the creation of language model applications. It provides utilities for chaining together multiple language model calls, which can be useful for building complex applications such as chatbots, question-answering systems, and automated summarization tools.

* **Use Case**: Creating complex language model workflows.
* **Features**: Supports chaining, conditional execution, and easy integration with various LLMs.

#### FastAPI - UI

**FastAPI** is a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints. It is designed for building APIs quickly and efficiently, with automatic interactive documentation.

* **Use Case**: Building the backend API for your application.
* **Features**: High performance, easy to use, automatic generation of OpenAPI documentation, and interactive API documentation.

#### FAISS - VectorDB

**FAISS** (Facebook AI Similarity Search) is a library for efficient similarity search and clustering of dense vectors. It's used for tasks such as nearest neighbor search and clustering.

* **Use Case**: Building and querying a vector database for similarity search.
* **Features**: High performance, supports large datasets, various indexing methods.

#### ChromaDB

**ChromaDB** is a database designed to store and query embeddings. It's optimized for high-performance operations on vector data, making it suitable for use cases involving embeddings from neural networks.

* **Use Case**: Efficient storage and querying of vector data.
* **Features**: High performance, optimized for embeddings.

#### Pinecone - VectorDB

**Pinecone** is a managed vector database service that allows you to store, index, and query vector embeddings at scale. It's designed to be highly scalable and integrates well with various ML and AI tools.

* **Use Case**: Storing and querying vector embeddings at scale.
* **Features**: Fully managed, scalable, integrates with various ML tools.

#### OpenAI/OpenSource - Issue

**OpenAI** provides various language models (e.g., GPT-3, GPT-4) that can be used for generating text, answering questions, and more. Open-source models like GPT-Neo, GPT-J, and others offer alternatives to proprietary models.

* **Use Case**: Natural language processing tasks.
* **Features**: High-quality text generation, large pre-trained models.

#### Embedding Model

**Embedding models** are used to convert text or other data into dense vector representations. These vectors capture the semantic meaning of the data and are used in various applications like similarity search, clustering, and more.

* **Use Case**: Converting text into dense vectors for downstream tasks.
* **Features**: Captures semantic meaning, used for various NLP tasks.