# Concepts to know for the Project

### Introduction

A project that leverages cutting-edge AI technology to revolutionize the way we create and manage software tests.

Our project uses GPT-4, a state-of-the-art language model, to generate macro tests from functional analysis documentation.

We then chunk this documentation and store it in a FAISS vector database for efficient retrieval.

This allows us to extract relevant chunks for each macro test and use them to generate detailed micro tests.

Let's dive into the technologies that make this possible.

### 1. GPT-4

- **What is GPT-4?**
  - GPT-4, or Generative Pre-trained Transformer 4, is an advanced AI language model developed by OpenAI.
  - It's capable of understanding and generating human-like text, making it a powerful tool for a variety of natural language processing tasks.
- **How are we using GPT-4?**
  - We use GPT-4 to analyze functional analysis documentation and generate macro tests that cover the full scope of software changes or updates.
  - Later, GPT-4 helps us create micro tests from the macro tests and related content extracted from our vector database.

### 2. Token

- **What is a Token?**
  - In the context of language models like GPT-4, a token is a piece of text that the model recognizes as a single unit.
  - This could be a word, part of a word, or punctuation.
- **Why are Tokens important?**
  - Tokens are the building blocks of the text that GPT-4 processes. The model's understanding and generation of text are based on these tokens.

### 3. Token Limitation

- **What is Token Limitation?**

    - GPT-4 can only process a certain number of tokens at a time. This limit affects how much text we can send to the model in one go.

- **How does this affect our project?**

    - We need to be mindful of the token limit when designing our system to ensure that the documentation is split appropriately for analysis and test generation.

## 4. GPT-4 Pricing

- **How is GPT-4 Priced?**

    - OpenAI charges for GPT-4 usage based on the number of tokens processed.

- **Project Costs**

    - We need to consider the cost implications of using GPT-4, especially since we're processing large documents and generating numerous tests.

## 5. Embeddings

- **What are Embeddings?**

    - Embeddings are numerical representations of text that capture semantic meaning. They allow us to compare and retrieve similar pieces of text efficiently.

- **Role in our project**

    - We convert chunks of our functional analysis documentation into embeddings so that we can store and search them in our vector database.

## 6. Vector Database (FAISS)

- **What is a Vector Database?**

    - A vector database, like FAISS (Facebook AI Similarity Search), is designed to store and search through large collections of embeddings.

- **How do we use it?**

    - We store the embeddings of our documentation chunks in a FAISS database, which allows us to perform similarity searches and retrieve relevant content for each macro test.

## 7. Additional Topics

- **Chunking**
    - The process of breaking down the functional analysis documentation into manageable pieces, each associated with a heading until the next one.
- **Similarity Search**
    - A technique used to find content in our vector database that is semantically similar to a given macro test.
- **Micro Test Generation**

- The creation of detailed test cases for each macro test using the related content extracted from the vector database.

### Conclusion

By combining GPT-4's language processing capabilities with the efficient storage and retrieval power of a vector database, we've created a system that can significantly streamline the test generation process. This not only saves time but also ensures that our tests are comprehensive and aligned with the latest functional changes in our software.

Thank you for your attention. I'm happy to answer any questions you may have.

# Project Documentation: Functional Analysis Summarization and Test Case Generation

To optimize the step-by-step explanation of the developed project, we can focus on making it more concise while retaining clarity and readability. Here's the optimized version:

1. The project, located in `RAG_implementation.ipynb`, is currently in the Proof of Concept (POC) stage.

2. In the initial stage, an API key is initialized using environment variables. Ensure your OpenAI API key is added to the `.env` file.

3. The user is prompted to input a "File Name" that should be analysed from the "Data" directories. This file contains Functional Analysis data used for generating test cases.

4. The primary objective is to summarize Functional Analysis documents into macro tests using the GPT4 model. The output structure includes:

   - Title
   - Description
   - Objective
   - Expected Output (if mentioned)
   - Expected Input (if mentioned)
   - Keywords

5. Simultaneously, the Functional Analysis document is chunked into sections consisting of headers and body text. Each chunk is embedded using the 'text-embedding-ada-002' model and stored in a FAISS vector database to create a knowledge base.

6. The next step involves parsing the GPT4 output to extract individual macro tests, which are stored in a list.

7. The list of macro tests is then iterated over to generate more detailed micro tests. This involves embedding a query for similarity search, extracting and sorting relevant chunks from the vector database, and formatting prompts for micro test generation.

8. Micro test templates include fields such as Title, Description, Objective, Expected Input/Output, and Procedure. The goal is to create as many micro tests as possible based on the provided context.

9. The generated micro tests are saved in the `micro_tests_results` folder.

By condensing the explanation and focusing on essential details, the optimized version provides a clearer overview of the project's workflow.