AI INTGATION WITH SPRING BOOT APPLICATION

INTRODUCTION:

In recent years, Artificial Intelligence (AI) has become a key component in building smart and responsive applications. Language models like OpenAI's GPT have made it possible to generate human-like responses, automate content creation, perform semantic analysis, and more — all through simple API calls.

Spring Boot, a powerful Java framework for building production-ready applications, provides a seamless way to integrate with AI services using HTTP clients or libraries like **Spring AI**. By connecting Spring Boot applications to AI APIs (e.g., OpenAI or Azure OpenAI), developers can enhance their software with capabilities such as natural language understanding, intelligent chat, summarization, and code generation.

This document explains how to connect a Spring Boot application with an AI model using standard tools and configurations, enabling intelligent interaction between backend services and advanced language models.

For Example:

we integrate our Spring boot application with two famous AI language models

1. OLLAMA

Now lets start with our first model OLLAMA

OLLAMA

Ollama is a **local AI runtime** that allows you to run and interact with large language models (LLMs) like **LLaMA**, **Mistral**, and **Gemma** directly on your machine — no cloud or API key required.

It simplifies the process of using open-source LLMs by handling model downloading, serving, and interaction through a unified and easy-to-use CLI or HTTP API.

Steps to integrate Spring boot application with Ollama

Step:1 Download and Install Ollama

- Visit Ollama Official website https://ollama.com/
- Download the Ollama according to your system configuration like windows, macOS,…..
- After downloaded Ollama, Just double click on that, by just clicking on next button
- After installation success click on Finish

Step:2 Start Ollama

- After installation successful open command prompt in your machine

  Commands to check ollama installed successfully or not:

  1. Verify Ollama is installed successfully or not

     Command: ollama

     Output:

```
C:\Users >ollama

Usage:

  ollama [flags]

  ollama [command]


Available Commands:

  serve      Start ollama

  create     Create a model from a Modelfile

  show       Show information for a model

  run        Run a model

  stop       Stop a running model

  pull       Pull a model from a registry

  push       Push a model to a registry

  list       List models

  ps         List running models

  cp         Copy a model

  rm         Remove a model

  help       Help about any command
```

```
Flags:
  -h, --help      help for ollama
  -v, --version   Show version information
```

➔ If the output is like above, you successfully installed it.

2. List out all the installed models
   Command: ollama list
   Output:

```
C:\Users>ollama list
NAME              ID              SIZE      MODIFIED
llama3.2:latest   a80c4f17acd5    2.0 GB    19 hours ago
```

➔ Observe the above output

It provides all the details about the AI Models

Here llama3.2:latest   -> Model name

a80c4f17acd5 -> Model ID

3. Run particular model

   Command: ollama run model_name

```
ollama run  llama3.2:latest
```

➔ If it is the first time you installed the model, it takes some time to install and
➔ If you already installed, it just start

4.

```
C:\Users>ollama run tinyllama:1.1b
>>> Send a message (/? for help)
```

Here you can give any prompt that will react.

Example: >>> Hello

Hello, How do you do…

Step:2 Integrate Ollama with Spring boot application

1. Create a simple Spring boot application in eclipse or IntelliJ or any other

   ➔ Add
   Spring web and Ollama dependencies in pom.xml

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-starter-model-ollama</artifactId>
</dependency>
```

   ➔ Add below property in pom.xml properties. It manages the AI versions.

```
<spring-ai.version>1.0.0</spring-ai.version>
```

   Add below BOM in dependency management

```
        <dependencyManagement>
                <dependencies>
                        <dependency>
                                <groupId>org.springframework.ai</groupId>
                                <artifactId>spring-ai-bom</artifactId>
                                <version>${spring-ai.version}</version>
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>
```

   ➔ Add this property in properties file. It loads by spring automatically.

```
spring.ai.ollama.chat.options.model=tinyllama:1.1b
```

- ➔ Controller

```
import org.springframework.ai.chat.client.ChatClient;
import org.springframework.ai.ollama.OllamaChatModel;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/ollama")
public class OllamaController {

	private ChatClient chatClient;

	public OllamaController(OllamaChatModel chatModel) {
			this.chatClient = ChatClient.create(chatModel);
	}

	@GetMapping("/chat")
	public String chat(@RequestParam String prompt) {
		return  chatClient.prompt(prompt).call().content();
	}
}
```

- ➔ After this just hit the request and get the output