**# 🧠 Demo AI – Image Question Answering Web App (Spring Boot + Flask + Hugging Face)**

This is a full-stack AI application that allows users to upload an image, ask a question about it, and get an intelligent answer. The system uses a \*\*Spring Boot\*\* frontend with \*\*Thymeleaf\*\*, and a \*\*Flask\*\* microservice running \*\*BLIP-2 (from Hugging Face)\*\* for visual question answering (VQA).

---

**## 🛠️ Technologies Used**

| Layer | Technology |

|--------------|----------------------------------|

| Frontend | HTML + CSS (Thymeleaf) |

| Backend | Java + Spring Boot + JPA |

| Database | MySQL |

| AI Model | BLIP-2 (`Salesforce/blip2-opt-2.7b`) |

| Image QA | Flask + HuggingFace Transformers |

| HTTP Client | RestTemplate |

| Tests | JUnit + Spring Boot Test |

---

**## 📁 Project Structure**

demoai/

├── src/

│ ├── main/

│ │ ├── java/com/demoai/

│ │ │ ├── controller/PromptController.java

│ │ │ ├── entity/Prompt.java

│ │ │ ├── repository/PromptRepository.java

│ │ │ └── service/MultipartInputStreamFileResource.java

│ │ └── resources/templates/index.html

│ └── test/java/... (JUnit Tests)

├── blip2\_service.py ← Flask-based AI microservice

├── README.md

└── pom.xml

---

**## 🚀 Features**

- ✅ Add, update, and view prompt templates (name + prompt text).

- ✅ Upload an image and ask a question about it.

- ✅ Calls HuggingFace BLIP-2 model via a local Python Flask microservice.

- ✅ Response displayed below the image upload form.

- ✅ JUnit test coverage for major features.

---

**## ⚙️ Prerequisites**

### Backend (Spring Boot)

- Java 17+

- Maven

- IDE (like IntelliJ or Eclipse)

### Python Microservice

- Python 3.9+ (tested on 3.11)

- pip packages:

```bash

pip install flask torch transformers accelerate pillow

**🧪 How to Run This Project**

1. Clone the Repository

git clone https://github.com/your-username/demoai.git

cd demoai

2. Run the Python Flask Microservice (AI Model)

Ensure blip2\_service.py is in the root project folder.

# Run this in a separate terminal

python blip2\_service.py

This will:

Load the BLIP-2 model (~15GB download on first run)

Start the server on: http://localhost:5000/predict

⚠️ First run will take time to download model weights.

3. Run the Spring Boot Application

# From root directory

mvn spring-boot:run

Open your browser at: http://localhost:8080

4. Using the App

Add a new prompt (prompt name + text)

Upload an image and select prompt

Ask your question

The app calls the Flask API, gets AI-generated answer, and displays it

🧪 Testing

JUnit tests are located in src/test/java/.

To run tests:

mvn test

**🐍 blip2\_service.py – Flask AI Service**

**Note:** Add this file to desktop and run desktop command prompt **python blip2\_service.py** first before running springboot project.

You should see:

Running on http://0.0.0.0:5000

python

from flask import Flask, request, jsonify

from PIL import Image

from transformers import Blip2Processor, Blip2ForConditionalGeneration

import torch

app = Flask(\_\_name\_\_)

processor = Blip2Processor.from\_pretrained("Salesforce/blip2-opt-2.7b")

model = Blip2ForConditionalGeneration.from\_pretrained(

"Salesforce/blip2-opt-2.7b",

device\_map="auto",

torch\_dtype=torch.float16 if torch.cuda.is\_available() else torch.float32

)

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

model.to(device)

@app.route('/predict', methods=['POST'])

def predict():

if "image" not in request.files or "question" not in request.form:

return jsonify({"error": "Missing image or question"}), 400

image = Image.open(request.files["image"]).convert("RGB")

question = request.form["question"]

inputs = processor(image, question, return\_tensors="pt").to(device)

generated\_ids = model.generate(\*\*inputs)

answer = processor.batch\_decode(generated\_ids, skip\_special\_tokens=True)[0].strip()

return jsonify({"answer": answer})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=5000)

📸 Sample Demo Screenshot

Include a screenshot of your app showing upload + response

**If using MySQL, change to:**

properties

spring.datasource.url=jdbc:mysql://localhost:3306/your\_db

spring.datasource.username=root

spring.datasource.password=your\_password

spring.jpa.hibernate.ddl-auto=update

**📌 Change the Flask API URL in PromptController.java:**

java

String flaskUrl = "http://localhost:5000/predict";

**🔚 Future Improvements**

Integrate with GPT or other multimodal models

Allow uploading Excel questions and batch processing

Store user queries and answers in database

Add image preview & progress indicator in frontend

**👨‍💻 Author**

Sameer Sheik