****

**Superior University Lahore**

***Project Report***

**Name:** Muhammad Abdullah Kashif

**Roll No**: SU92-BSAIM-F23-070

**Instructor**: Sir Rasikh Ali

# Course: Programming for Artificial Intelligence (Lab)

***Project Title:***

**Image Caption Generator using Flask**

**1. Introduction**  
This project presents the development of a web-based application that generates descriptive textual captions for images uploaded by users. The system integrates a pre-trained vision-language deep learning model with a minimal Flask-based interface. The solution is designed to demonstrate the use of deep learning in natural language generation and computer vision, emphasizing usability and modular software design.

**2. Objective**  
The primary objective is to develop a lightweight and efficient web application that accepts an image input and returns a contextually relevant caption. This is achieved by utilizing the BLIP (Bootstrapped Language-Image Pretraining) model, which is capable of generating high-quality captions from raw image inputs. The system focuses on seamless user interaction, clean deployment structure, and accurate caption generation.

**3. System Architecture**  
The application architecture consists of two main components:

* **Frontend Interface (HTML templates):** Enables users to upload images and view results.
* **Backend Logic (Flask + PyTorch):** Handles file upload, image processing, model inference, and rendering of results.

The process flow is as follows:

1. The user uploads an image via the web interface.
2. The server saves the image to a predefined directory.
3. The image path is passed to a caption generation function.
4. The BLIP model processes the image and generates a caption.
5. The result is displayed back to the user along with the image.

**4. Tools and Technologies**

| **Component** | **Technology** |
| --- | --- |
| Web Framework | Flask 2.0.3 |
| Image Processing | Pillow (PIL) 9.3.0 |
| Deep Learning Model | BLIP (Salesforce/blip-image-captioning-base) |
| Model Framework | PyTorch |
| Templating Engine | Jinja2 (used within HTML) |
| Programming Language | Python 3.10 |
| Supporting Libraries | transformers (Hugging Face), os, werkzeug.utils |

**5. Detailed Implementation**

**5.1 Backend (app.py):**

* The Flask application is initialized and configured to store uploaded images in a static/uploads directory.
* A single route ('/') handles both GET and POST requests.
* When a POST request is received, the uploaded file is retrieved, sanitized using secure\_filename, and saved locally.
* The image path is passed to the generate\_caption function, and the output is rendered using the result.html template.

**5.2 Caption Generation Logic (caption\_generator.py):**

* The BLIP model and its processor are loaded using Hugging Face's transformers library.
* The generate\_caption() function opens the image, converts it to RGB, and processes it into the required tensor format.
* The model generates a tokenized caption, which is then decoded into human-readable text.

**5.3 Frontend (HTML Templates):**

* index.html contains a basic file upload form and submits the request to the server.
* result.html displays the uploaded image and the generated caption dynamically using Jinja2 variables.

**6. Directory Structure**

project/

├── app.py

├── caption\_generator.py

├── static/

│ └── uploads/

├── templates/

│ ├── index.html

│ └── result.html

**7. Model Overview**

The caption generation leverages the Salesforce BLIP image captioning base model. BLIP is a vision-language model designed for tasks like image-text retrieval, image captioning, and visual question answering. The model accepts raw images and uses a transformer-based encoder-decoder architecture to generate descriptive captions. The use of BlipProcessor ensures proper tokenization and preprocessing for the model.

**8. Results**

The application successfully generates meaningful captions for a variety of uploaded images. The integration of the BLIP model provides robust performance, producing captions that accurately reflect the content of the input image. The web interface remains lightweight and responsive, offering a smooth user experience.





