🖼️ Sticker Studio – User Manual

Authors: Nikshay Jain | MM21B044 & Sahil Kokare | MM21B036

📌 Introduction

Sticker Studio is an end-to-end pipeline for generating creative Ghibli-style or normal stickers using advanced AI techniques. It combines scraping, model training (YOLOv8 and Detectron2), text-to-image generation (StabilityAI), image classification, segmentation, captioning, and monitoring via Prometheus & Grafana.

This user manual outlines the system architecture, setup instructions, and usage guidance, integrating details from your README and the project diagram.

🧠 System Overview

The system comprises the following integrated modules:

1. Scraper & Dataset Management

- Uses Pinterest image scraping (100–150 images/day) via a cron job.
- DVC tracks folders to ensure dataset versioning.

2. Segmentor Model Trainer

- Two segmentation backbones supported:
  - **YOLOv8n** (finetuned for speed, low compute)
  - **Detectron2** (high accuracy, high compute)
- Human feedback loop to improve segmentation performance.
- Training and MLflow used to track hyperparameters and metrics.

3. Classifier

- A MobileNet-based classifier detects whether an image is "Ghibli" or "Normal".
- Used to decide which segmentation model (normal or finetuned) should be called.

4. Frontend & Backend Application

- **Streamlit UI** allows:

    - Text-to-image generation via StabilityAI or ModelsLab fallback
    - Uploading your own image
    - Live caption editing and font selection
- Backend logic handles:
    - Classification
    - Segmentation
    - Caption placement
    - File saving and return

5. Monitoring Stack

- **Prometheus** tracks metrics like:
    - Inference latency
    - Number of calls
    - Failure/success ratios
- **Grafana** visualizes metrics in near real time
- Python's `prometheus_client` exposes metrics on port `18000`

📂 Directory Structure

```
Sticker-Studio/
├── Backend-Model-Development/   # Cronjobs, training scripts, orchestration
├── Frontend-Tester/             # App logic, UI, models, and monitoring
├── image.png                    # Architecture diagram
└── Readme.md                    # Project documentation
```

⚙️ Setup Instructions

1. Clone the Repository

```
git clone https://github.com/Nikshay-Jain/Sticker-Studio.git
cd Sticker-Studio
```

2. Setup Python Environment

Recommended OS: WSL (Ubuntu) or macOS
Python version: **3.9**

Install dependencies:

```
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu
pip install -r requirements.txt
```

3. Install Detectron2

```
python3 Frontend-Tester/src/install_detectron2.py
```

🚀 Running the App

Launch the Frontend (Streamlit)

```
cd Frontend-Tester
streamlit run src/app.py
```

📈 Monitoring

Start Exporter (runs in background)

```
python3 Frontend-Tester/src/main.py
```

Run Prometheus & Grafana (Docker)

Ensure your Prometheus config is correct and run:

```
docker-compose up -d
```

View Dashboards

- 🔗 Prometheus: http://localhost:9090
- 📊 Grafana: http://localhost:3000

🧪 How It Works (from UI)

1. **User Inputs**: Upload image or enter text
2. **Image Generation**: Calls Stability AI / ModelsLab API
3. **Classification**: MobileNet classifies image as "Normal" or "Ghibli"
4. **Segmentation**:
   - Normal image → Normal YOLOv8
   - Ghibli image → Finetuned YOLOv8
5. **Captioning & Styling**
6. **Downloadable Sticker** served to user

🐞 Logs & Debugging

All logs are saved in:

```
Frontend-Tester/logs/
```

Errors in image generation, model failures, or API issues are logged here.

📌 Notes

- The project supports DVC-based reproducibility
- Cron jobs automate scraping & dataset updates
- All metrics can be visualized live via Grafana