

Master IT   
University: Mohamed 5

Faculty of sciences Rabat

Rapport about

**Text-to-Image Generation using Stable Diffusion 3 and LoRA: A Gradio Powered Application**

Prepared by: EL BACHIRI ABDESSALED et ASSIA BELGUEDDAR

Professor: ABDELHAK MAHMOUDI

1. **Introduction**

In recent years, generative models have revolutionized artificial intelligence, especially in the field of image synthesis. One of the most prominent techniques is text-to-image generation, where a model transforms a textual description into a realistic image. This report details a practical implementation of such a system using the Stable Diffusion 3 (SD3) model enhanced with Low-Rank Adaptation (LoRA) and deployed using the Gradio interface for user-friendly interaction.

1. **Objectives of the Project**

* Explore and apply a pre-trained model: Stable Diffusion 3 (SD3).
* Improve the model’s performance using a LoRA fine-tuned adapter.
* Build an intuitive front-end using Gradio.
* Enable users to generate images from custom textual prompts.
* Understand the interplay between deep learning, diffusion models, and UI integration.

1. **Technical Stack Overview**

**3.1 Python and PyTorch**

Python is the primary programming language. PyTorch is used as the deep learning backend due to its dynamic computation graph and GPU acceleration support.

**3**.**2 Diffusers Library**

The diffusers library from Hugging Face allows easy access to state-of-the-art diffusion models like SD3. It handles model loading, inference, and LoRA integration.

**3.3 Gradio**

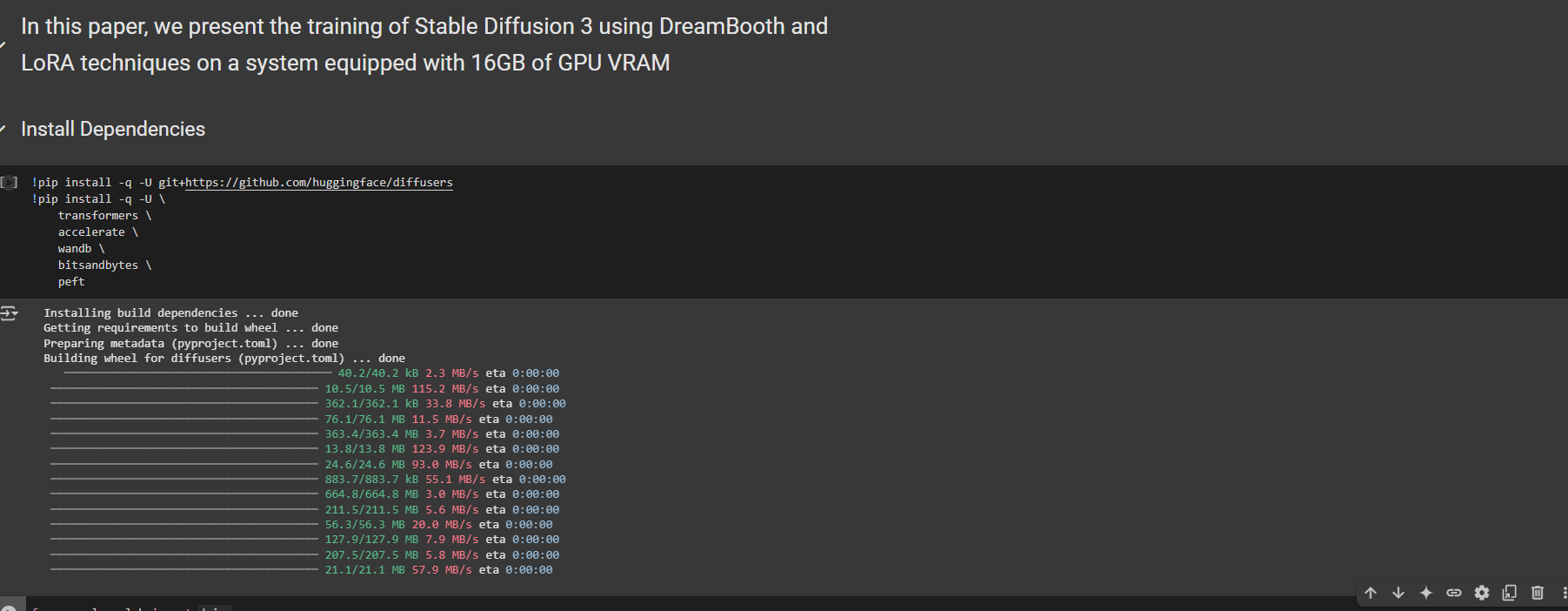
Gradio is used to build a web interface that allows users to interact with the model by inputting prompts and receiving generated images.

**Part 1: Training Model**

In this section, we use Google Colab to train our model. Below is a step-by-step guide to training a custom model

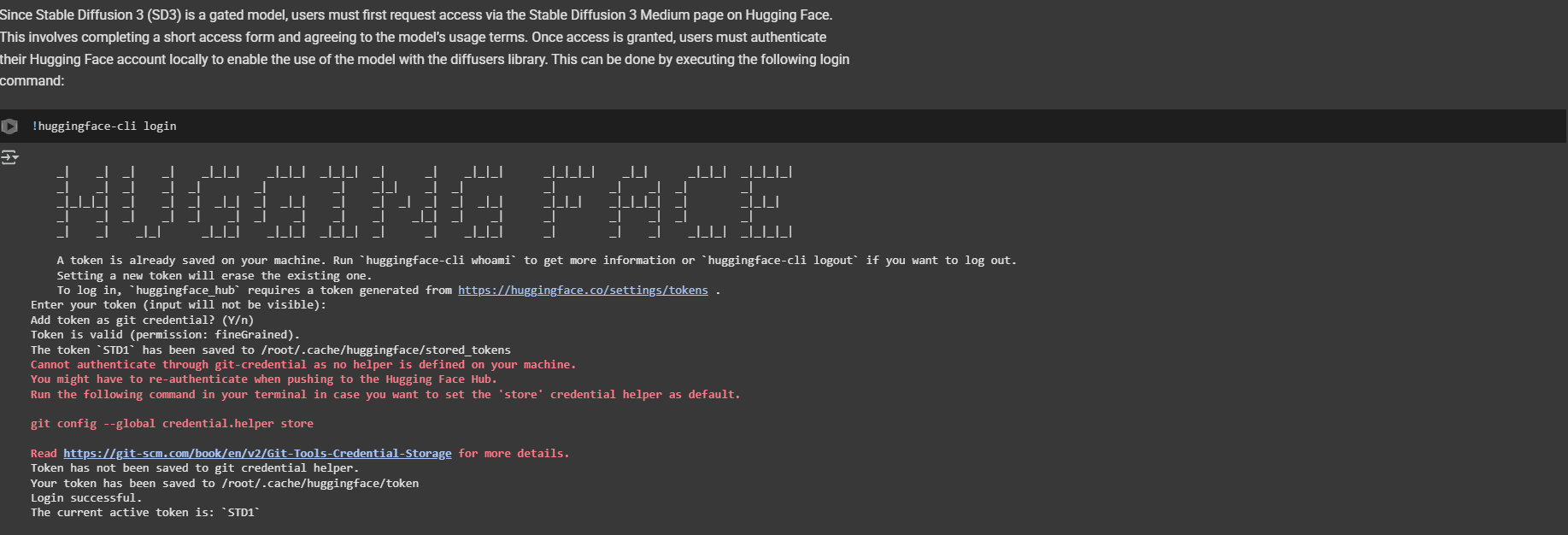
**Step 1: Environment Setup**

In this step, we prepare the training environment by installing essential libraries. These include diffusers for working with diffusion models, transformers for text processing, accelerate for efficient training, wandb for tracking, bitsandbytes for memory optimization, and peft for applying LoRA fine-tuning. This ensures the system is ready to train Stable Diffusion 3 using DreamBooth and LoRA on a 16GB GPU.



**Step 2: Hugging Face Authentication**

To access the Stable Diffusion 3 model, which is gated, users must first accept its terms on Hugging Face. After access is granted, the huggingface-cli login command is used to authenticate the session using a personal access token. This step enables secure programmatic access to the model through the diffusers library.



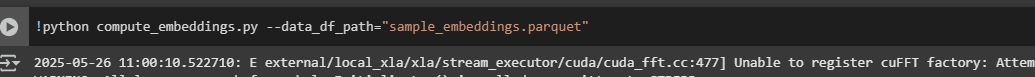
**Step 3: Prepare Custom Training Dataset**

In this step, we use a custom dataset of images stored locally in a folder (e.g. ./cat). A caption.json file provides text prompts associated with each image. The script reads the JSON file, checks that each image exists, and builds a structured table linking each image path to its corresponding prompt. This data is then saved as a .parquet file (sample\_embeddings.parquet), which will later be used by the compute\_embeddings.py script to generate embeddings. This format is required for DreamBooth training in the text-to-image con



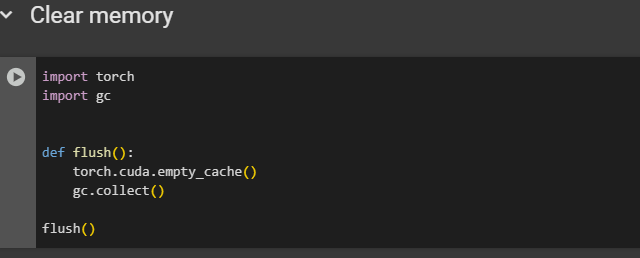
**Step 4: Compute Image Embeddings**

We run the compute\_embeddings.py script to generate embeddings that link the cat images with the prompt “a photo of cat.” This step is essential for DreamBooth + LoRA training, as it helps the model associate visual features with the target text. During execution, the script downloads base model weights in .safetensors format to perform this embedding process efficiently and securely.



**Step 5: Clear GPU and CPU Memory**

To avoid memory issues before training, we define a flush() function that frees up GPU cache using torch.cuda.empty\_cache() and releases unused CPU memory with gc.collect(). Running this step ensures that maximum memory is available for loading and training the large SD3 model, especially important on a 16GB VRAM GPU.

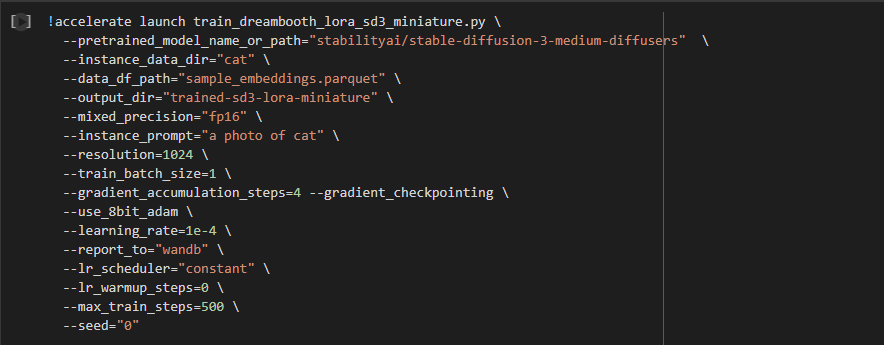


**Step 6: Train the Model**

We launch the training process using the accelerate command, which allows efficient fine-tuning of Stable Diffusion 3 with DreamBooth and LoRA. The script uses the cat images and their embeddings to teach the model how to associate the prompt “a photo of cat” with the visual features. Key parameters include:

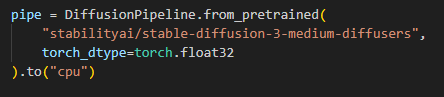
* --mixed\_precision="fp16" and --use\_8bit\_adam to save memory,
* --train\_batch\_size=1 with --gradient\_accumulation\_steps=4 to simulate a larger batch,
* --resolution=1024 for high-quality output,
* --output\_dir="trained-sd3-lora-miniature" to store the fine-tuned model.

This setup enables training on a GPU with just 16GB of VRAM while maintaining stability and efficiency.



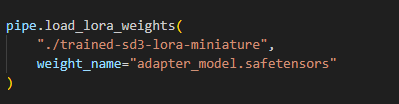
**Part 2: using fine-tuned model**

**Loading the Diffusion Model**



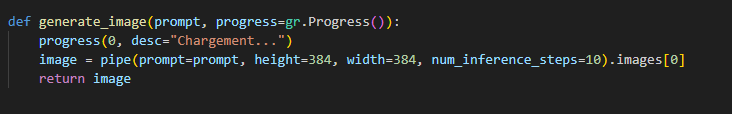
* Loads the medium variant of SD3.
* Uses float32 precision for CPU compatibility.
* .to("cpu") is used due to hardware limitations.

**Loading LoRA Adapter**



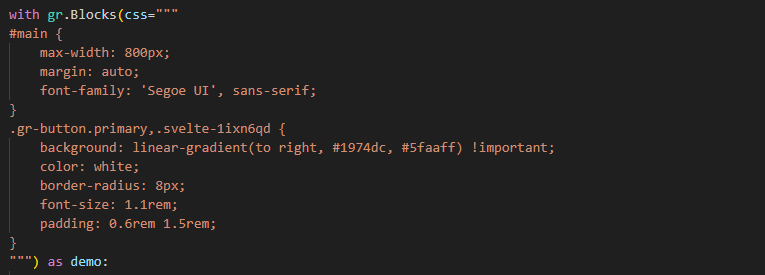
* Loads a LoRA-trained adapter for improved domain-specific generation.
* safetensors ensures fast and secure model weight loading.

**Image Generation Function**

Receives a prompt and generates an image.

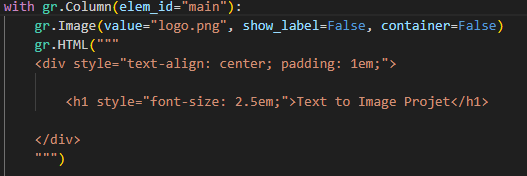
* Uses 10 denoising steps for faster yet decent-quality outputs.

**Gradio Interface Design with Custom Styling**



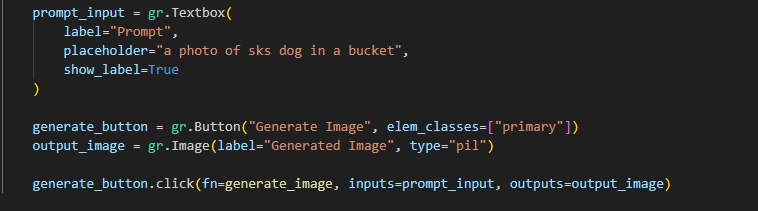
* Adds a modern blue button gradient.
* Improves visual user interaction.

**Layout Elements**

****

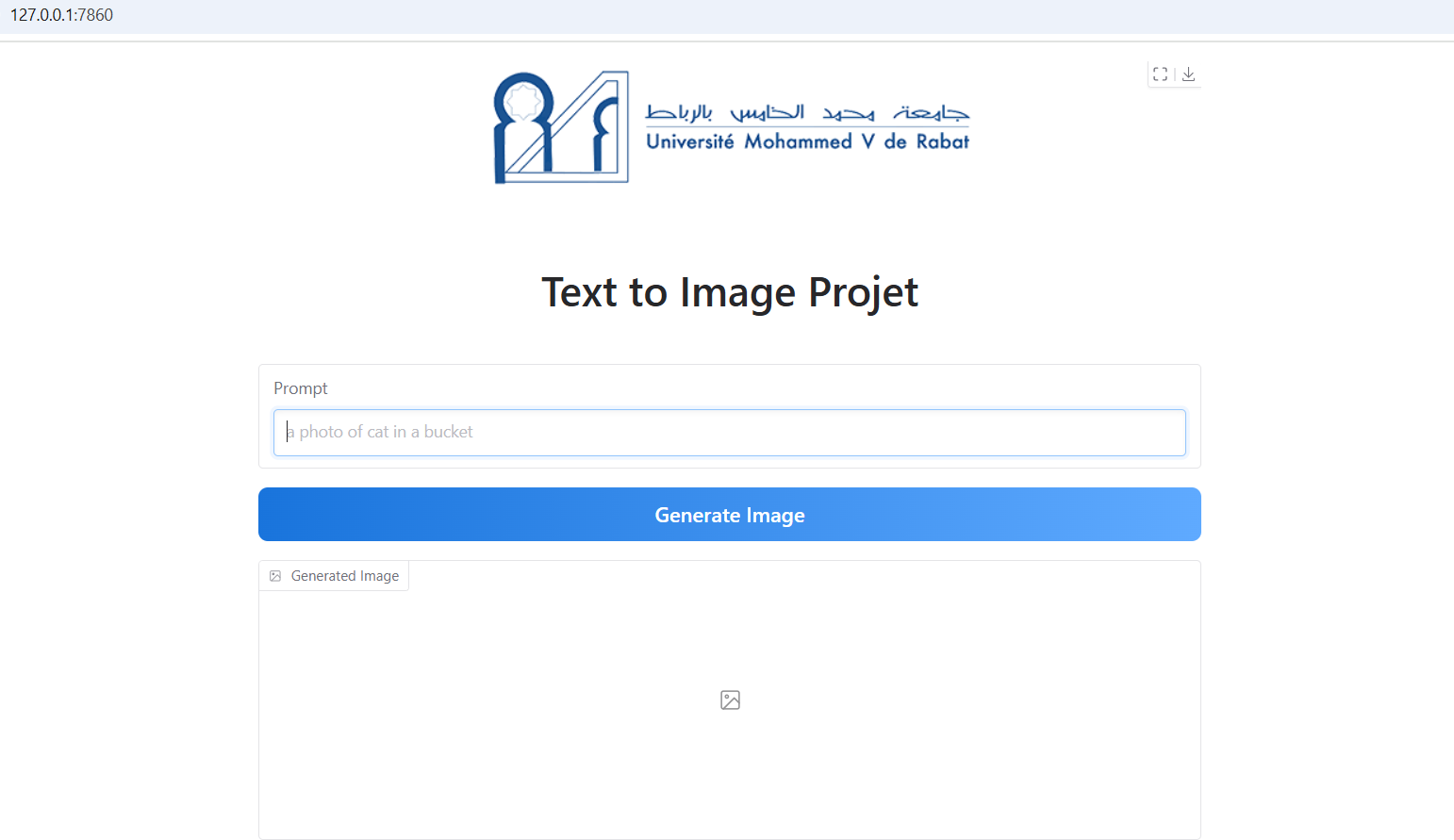
* Shows a custom logo and project title.
* Provides a clean and branded look.

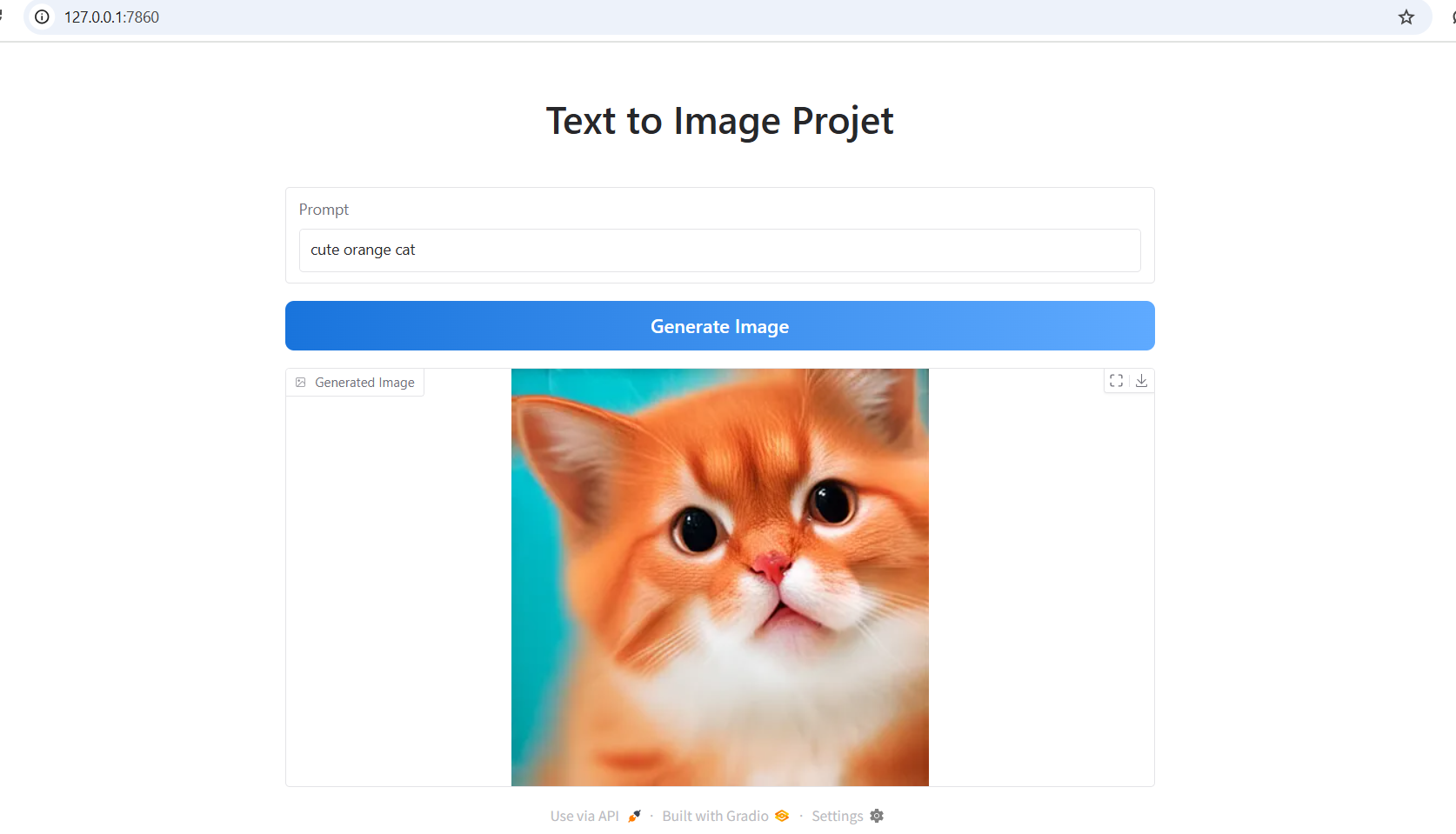
**User Interaction**

****

* Textbox to input prompt.
* Button triggers generation.
* Resulting image is displayed below.

**User interface**





**Results and Observations**

* The model is able to generate reasonable images from short textual prompts.
* LoRA adapter noticeably improves coherence and detail in niche categories (if trained well).
* Interface is intuitive and mobile-friendly.
* The image size and inference speed are well-balanced for CPU usage.

**Challenges Faced**

* **Limited Resources**: Running on CPU leads to slower inference and limits image resolution.
* **Model Size**: SD3 is heavy; managing memory and loading times is crucial.
* **Training LoRA**: While this report uses a pre-trained LoRA, fine-tuning it on a new dataset would require GPU resources and time.

**Conclusion**

This project demonstrates the power of diffusion models combined with LoRA efficiency and Gradio simplicity. It provides a foundation for exploring generative AI applications in creative, educational, and industrial fields. Through this report, we have detailed the model's configuration, usage, and interface design — offering a reproducible and extendable project for future exploration.

**References**

Hugging Face – <https://huggingface.co>

Stability AI – <https://stability.ai>

Diffusers Library – <https://github.com/huggingface/diffusers>

Gradio – <https://www.gradio.app>

Low-Rank Adaptation (LoRA) Paper – <https://arxiv.org/abs/2106.09685>