# Natural Language Processing

## Text2Image AI Agent – Project Report


Project Team

Fatima Basit 21I-0711

Muhammad Yahya 21I-2592

Shahmeer Ali Akhtar 21I-0466

Course Instructor

Dr. Omer Beg


**Department of Computer Science**

**National University of Computer and Emerging Sciences**
**Islamabad, Pakistan**

# Contents

**Abstract**

The Text2Image AI Agent is a containerized microservice developed for a course project to generate images from text descriptions using Stable Diffu- sion, an open-weights model. It features a gRPC API for backend processing, a Gradio front-end for user interaction, and a CI/CD pipeline using GitHub Ac- tions. The front-end is deployed to Hugging Face Spaces, and the backend is deployable to remote servers or cloud platforms. Optimized for CPU-only en- vironments, the solution is self-contained, avoids online API calls, and meets all project requirements.

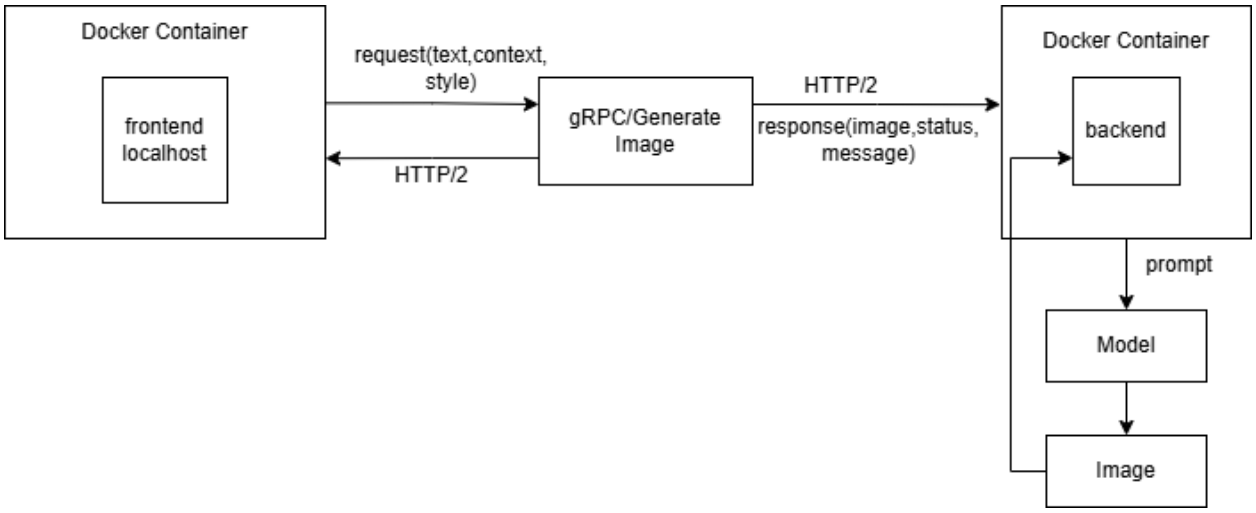
## 1 Introduction

The Text2Image AI Agent aims to generate high-quality images from text descrip- tions, context, and style inputs, leveraging modern AI and DevOps practices. Built as a course project, it integrates Stable Diffusion, gRPC, Docker, Gradio, GitHub Actions, and Hugging Face Spaces, adhering to requirements for an off- the-shelf open-weights model, automated CI/CD, and deployment to Spaces.

## 2 Objectives

- Develop a microservice for text-to-image generation.
- Use an open-weights model (Stable Diffusion).
- Implement a gRPC API and Gradio front-end.
- Automate CI/CD with GitHub Actions.
- Deploy the front-end to Hugging Face Spaces and the backend scalably.
- Ensure self-containment, no online API calls, and CPU compatibility.

## 3 Implementation




### 3.1 Model Selection

The project uses runwayml/stable-diffusion-v1-5, an open-weights model from Hugging Face, embedded in the Docker image ( 6–8 GB). It is configured for CPU inference with torch.float32, attention_slicing, and 256x256 resolution.

### 1.1 Backend (gRPC Microservice)

A gRPC server (src/server.py) handles asynchronous requests via a Text2Image service (text2image.proto). Stable Diffusion is integrated with local_files_only=True, loading weights from /app/models. The backend is Dockerized, with weights included and unnecessary files excluded via .dockerignore.

### 1.2 Front-End (Gradio)

A Gradio interface (frontend/frontend.py) calls the gRPC server, accepting text, context, and style inputs. It is packaged with dependencies (frontend/requirements.t for deployment to Hugging Face Spaces.

### 1.1 CI/CD Pipeline

GitHub Actions (.github/workflows/ci-cd.yml) automates:

- Linting with pylint.
- Unit testing with pytest.
- Building and pushing the Docker image to Docker Hub.

The pipeline runs on push and pull_request to the main branch.

### 1.2 Deployment

The backend is deployable to remote servers or cloud services (e.g., AWS ECS)

us- ing the Docker image. The Gradio front-end is deployed to Hugging Face Spaces, configured to call the backend's public IP.

### 1.3 Testing

Unit tests (tests/unit/test_server.py) validate the gRPC server. Test cases (tests/test_cases.md) cover functionality, edge cases, deployment, and front- end interaction. Testing is performed with src/client.py, Postman, and Gra- dio.

## 1 Outcomes

### 1.1 Functionality

The microservice generates images from text inputs (e.g., "A sunset over a lake") via gRPC and Gradio, with context and style customization.

### 1.1 Compliance

The project uses runwayml/stable-diffusion-v1-5, automates CI/CD with GitHub Actions, deploys the front-end to Hugging Face Spaces, is self-contained, avoids online API calls, and is CPU-compatible.

### 1.2 Performance

Images (256x256) are generated in 1–5 minutes on CPU, with potential for GPU optimization.

### 1.3 Scalability

The Dockerized backend supports various platforms, and the Gradio front-end is globally accessible via Spaces.

## 2 Limitations

- **Inference Speed**: CPU inference is slow (1–5 minutes). GPU recommended for production.
- **Image Size**: Docker image is large ( 6–8 GB), slowing builds.
- **Resources**: Requires 8 GB disk and 8–10 GB RAM.
- **Testing**: Limited automated tests for Gradio.

## 3 Future Improvements

- Use a lighter model for faster inference.
- Expand front-end tests.
- Automate Spaces deployment with GitHub Actions.
- Add an architecture diagram.
- Explore GPU deployment.

## 4 Conclusion

The Text2Image AI Agent meets all project requirements, delivering a functional, scalable, and automated solution for text-to-image generation. Comprehensive documentation ensures accessibility for users and evaluators.