FINAL YEAR PROJECT REPORT

Cloud-Based Text-to-Image Generation System using Google Vertex AI

Submitted By: Yaseen Raza

College: SSGS College

Academic Year: 2025-2026

Abstract

This project presents a cloud-based generative AI system that converts textual prompts into high-quality images using
The system leverages diffusion-based models deployed on scalable cloud infrastructure.
Users can enter prompts, generate AI-based images, store them securely, and download outputs.

1. Introduction

Generative AI is a rapidly growing field of Artificial Intelligence that focuses on generating new content such as image
Text-to-image models are based on diffusion and transformer architectures.
This project implements a scalable web-based AI image generation system using Google Cloud services.

2. Literature Survey

Existing research in GANs (Generative Adversarial Networks) and Diffusion Models has significantly advanced image
Stable Diffusion, DALL·E, and Imagen are examples of modern text-to-image systems.
Cloud-based AI platforms provide scalable infrastructure for model deployment.

3. System Architecture

The system consists of:
1. Frontend (HTML/CSS/JS)
2. Backend API (Python Flask)
3. Vertex AI Image Generation Model
4. Cloud Storage
5. Cloud Run Deployment

Workflow:
User → Backend → Vertex AI → Cloud Storage → User Download

4. Technologies Used

Cloud Platform: Google Cloud Platform
AI Service: Vertex AI
Backend: Python (Flask)
Frontend: HTML/CSS/JavaScript
Storage: Google Cloud Storage
Deployment: Cloud Run

5. Implementation Details

Step 1: Create Google Cloud Project
Step 2: Enable Vertex AI API
Step 3: Create Storage Bucket
Step 4: Develop Backend API
Step 5: Deploy using Cloud Run

6. Backend Code Example (Flask API)

```
from flask import Flask, request, jsonify
from vertexai.preview.vision_models import ImageGenerationModel

app = Flask(__name__)

@app.route('/generate', methods=['POST'])
def generate_image():
    prompt = request.json['prompt']
    model = ImageGenerationModel.from_pretrained("imagegeneration@006")
    response = model.generate_images(prompt=prompt, number_of_images=1)
    response[0].save("output.png")
    return jsonify({"message": "Image Generated Successfully"})

if __name__ == '__main__':
    app.run(debug=True)
```

7. Cloud Storage Upload Code

```
from google.cloud import storage

def upload_to_bucket(file_name, bucket_name):
    client = storage.Client()
    bucket = client.bucket(bucket_name)
    blob = bucket.blob(file_name)
    blob.upload_from_filename(file_name)
    print("File uploaded successfully")
```

8. Frontend HTML Example

```
<!DOCTYPE html>
<html>
<head><title>AI Image Generator</title></head>
<body>
<h2>Enter Prompt</h2>
<input type="text" id="prompt">
<button onclick="generate()">Generate</button>

<script>
function generate(){
 fetch('/generate', {
   method:'POST',
   headers:{'Content-Type':'application/json'},
   body:JSON.stringify({prompt:document.getElementById('prompt').value})
 });
}
</script>
</body>
</html>
```

9. Example Prompts

1. A futuristic cyberpunk city at night
2. A lion wearing sunglasses in 8K
3. Realistic astronaut riding a horse
4. Anime-style warrior princess

10. Testing

Unit Testing: API endpoint testing
Integration Testing: Backend + Vertex AI
Performance Testing: Load testing using multiple prompts

11. Results

The system successfully generates high-resolution AI images.
Cloud deployment ensures scalability and low latency performance.

12. Advantages

• Scalable cloud infrastructure
• No local GPU required
• Secure storage
• Real-time generation

13. Limitations

- API cost dependency
- Prompt sensitivity
- Internet dependency

14. Future Enhancements

• User authentication
• Image history tracking
• Fine-tuning custom datasets
• Mobile app integration

15. Conclusion

This project demonstrates how Generative AI and Cloud Computing can be integrated to build scalable and efficient AI-powered applications.

16. References

1. Google Cloud Documentation
2. Research Papers on Diffusion Models
3. Generative AI Whitepapers

Abstract

This project presents a cloud-based generative AI system that converts textual prompts into high-quality images using
The system leverages diffusion-based models deployed on scalable cloud infrastructure.
Users can enter prompts, generate AI-based images, store them securely, and download outputs.

1. Introduction

Generative AI is a rapidly growing field of Artificial Intelligence that focuses on generating new content such as image
Text-to-image models are based on diffusion and transformer architectures.
This project implements a scalable web-based AI image generation system using Google Cloud services.

2. Literature Survey

Existing research in GANs (Generative Adversarial Networks) and Diffusion Models has significantly advanced image
Stable Diffusion, DALL·E, and Imagen are examples of modern text-to-image systems.
Cloud-based AI platforms provide scalable infrastructure for model deployment.

3. System Architecture

The system consists of:
1. Frontend (HTML/CSS/JS)
2. Backend API (Python Flask)
3. Vertex AI Image Generation Model
4. Cloud Storage
5. Cloud Run Deployment

Workflow:
User → Backend → Vertex AI → Cloud Storage → User Download

4. Technologies Used

Cloud Platform: Google Cloud Platform
AI Service: Vertex AI
Backend: Python (Flask)
Frontend: HTML/CSS/JavaScript
Storage: Google Cloud Storage
Deployment: Cloud Run

5. Implementation Details

Step 1: Create Google Cloud Project
Step 2: Enable Vertex AI API
Step 3: Create Storage Bucket
Step 4: Develop Backend API
Step 5: Deploy using Cloud Run

6. Backend Code Example (Flask API)

```
from flask import Flask, request, jsonify
from vertexai.preview.vision_models import ImageGenerationModel

app = Flask(__name__)

@app.route('/generate', methods=['POST'])
def generate_image():
    prompt = request.json['prompt']
    model = ImageGenerationModel.from_pretrained("imagegeneration@006")
    response = model.generate_images(prompt=prompt, number_of_images=1)
    response[0].save("output.png")
    return jsonify({"message": "Image Generated Successfully"})

if __name__ == '__main__':
    app.run(debug=True)
```

7. Cloud Storage Upload Code

```
from google.cloud import storage

def upload_to_bucket(file_name, bucket_name):
    client = storage.Client()
    bucket = client.bucket(bucket_name)
    blob = bucket.blob(file_name)
    blob.upload_from_filename(file_name)
    print("File uploaded successfully")
```

8. Frontend HTML Example

```
<!DOCTYPE html>
<html>
<head><title>AI Image Generator</title></head>
<body>
<h2>Enter Prompt</h2>
<input type="text" id="prompt">
<button onclick="generate()">Generate</button>

<script>
function generate(){
 fetch('/generate', {
   method:'POST',
   headers:{'Content-Type':'application/json'},
   body:JSON.stringify({prompt:document.getElementById('prompt').value})
 });
}
</script>
</body>
</html>
```

9. Example Prompts

1. A futuristic cyberpunk city at night
2. A lion wearing sunglasses in 8K
3. Realistic astronaut riding a horse
4. Anime-style warrior princess

10. Testing

Unit Testing: API endpoint testing
Integration Testing: Backend + Vertex AI
Performance Testing: Load testing using multiple prompts

11. Results

The system successfully generates high-resolution AI images.
Cloud deployment ensures scalability and low latency performance.

12. Advantages

• Scalable cloud infrastructure
• No local GPU required
• Secure storage
• Real-time generation

13. Limitations

- API cost dependency
- Prompt sensitivity
- Internet dependency

14. Future Enhancements

• User authentication
• Image history tracking
• Fine-tuning custom datasets
• Mobile app integration

15. Conclusion

This project demonstrates how Generative AI and Cloud Computing can be integrated to build scalable and efficient AI-powered applications.

16. References

1. Google Cloud Documentation
2. Research Papers on Diffusion Models
3. Generative AI Whitepapers