**Software Requirements Specification**

**Version 1.0**

**March 26, 2024**

**TEXT TO IMAGE GENERATOR**

**Anagha Anoop**

**Arusha N S**

**Jeni M Rose**

**Revathy D R**

**Table Of Content**

**1. Introduction…………………………………………………………………………………………… 03**

**1.1 purpose………………………………………………………………………………………….. 03**

**1.2 Document Conventions………………………………………………………………….. 03**

**1.3 Intended Audience and Reading Suggestions…………………………………. 03**

**1.4 Project Scope…………………………………………………………………………………. 04**

**1.5 References……………………………………………………………………………………… 04**

**2. Overall Description………………………………………………………………………………… 05**

**2.1 Product perspective……………………………………………………………………….. 05**

**2.2 Product Features……………………………………………………………………………. 05**

**2.3 User classes and characteristics……………………………………………………… 06**

**2.4 Operating Environment………………………………………………………………….. 06**

**2.5 Design and implementation constraints…………………………………………. 06**

**2.6 Assumptions and dependencies……………………………………………………… 06**

**3. System Features…………………………………………………………………………………….. 07**

**3.1 Functional Requirements……………………………………………………………….. 07**

**4. External Interface Requirements……………………………………………………………. 09**

**4.1 User interface………………………………………………………………………………… 09**

**4.2 Hardware interface………………………………………………………………………… 09**

**4.3 Software interface…………………………………………………………………………. 09**

**4.4 Communication interface………………………………………………………………. 10**

**5. Non-functional Requirements………………………………………………………………… 11**

**5.1 performance Requirements……………………………………………………………. 11**

**5.2 safety Requirements………………………………………………………………………. 12**

**5.3 security Requirements……………………………………………………………………. 12**

**5.4 software Quality attributes……………………………………………………………. 13**

1. **Introduction**
   1. ***Purpose***

The purpose of this document is to define the requirements for the development of a text-to-image generator using Generative Adversarial Networks (GANs). It serves as a roadmap for the development team, guiding them through the design, implementation, and testing phases of the project. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli**.**

**1.2 *Document Conventions***

* Headings: Major sections are numbered and use a larger font size for easy navigation.
* Bullet Points: Used for listing requirements and specifications in an organized manner.
* Terminology: Technical terms and domain-specific terminology are defined and used consistently throughout the document.
  1. ***Intended Audience and Reading Suggestions:***

Audience: Developers, project managers, stakeholders, and quality assurance personnel involved in the project.

Reading Suggestions: It is recommended to read through the entire document to gain a thorough understanding of the project requirements and constraints***.***

* 1. ***Project Scope:***

The project aims to develop a text-to-image generator capable of producing high-quality images from textual descriptions. Designing and training a GAN model, developing a user interface, setting up a training pipeline, and deploying the system. The software will provide an intuitive user interface for inputting text and viewing generated images. It will leverage deep learning techniques, such as Generative Adversarial Networks (GANs), for image generation.

* 1. ***References:***

[1] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In ICLR, 2017.

[2] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. In ICLR, 2017.

[3] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li. Mode regularized generative adversarial networks. In ICLR, 2017.

[4] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In NIPS, 2016.

**2.Overall Description**

The text-to-image project aims to address the current limitations hindering the company's growth opportunities by developing a robust system capable of generating images from textual descriptions efficiently. The proposed system will feature a text input interface for users, deep learning algorithms for image generation, customization options, feedback mechanisms, and integration capabilities. Assumptions include the availability of suitable training datasets and dependencies on deep learning frameworks.

**2.1 *Product Perspective***

The text-to-image generator software will be a standalone application, allowing users to input textual descriptions and receive corresponding images. It will interface with users through a graphical user interface (GUI) and may utilize backend servers for computational tasks such as model inference.

* 1. ***Product Features***
* *Text Input:* Provide a text input interface for users to input descriptions of images they want to generate.
* *Image Generation:* Utilize deep learning models, such as GANs, to generate images based on the provided textual descriptions.
* *Image Display:* Display generated images to users in the software's graphical user interface.
* *Model Management:* Allow for the management and updating of the underlying deep learning model used for image generation.
  1. ***User Classes and Characteristics***

Developers: Responsible for designing, implementing, and maintaining the text-to-image generator software.

End Users: Individuals or organizations interested in generating synthetic images from textual descriptions for various purposes such as content creation and design.

* 1. ***Operating Environment***

The system will be developed using modern web technologies such as HTML, CSS, JavaScript, and backend frameworks like Flask.Compatibility with major web browsers and operating systems will be ensured for widespread accessibility.

* 1. ***Design and implementation constraints***

The software should be designed with usability and performance in mind to accommodate users with varying levels of technical expertise. Integration with deep learning frameworks and libraries for GAN model implementation and optimization.

* 1. ***Assumptions and dependencies***

*Assumptions:* Availability of suitable datasets for training and refining GAN models, access to deep learning frameworks and libraries.

*Dependencies:* Reliance on computational resources for model training and inference, cooperation from stakeholders for testing and feedback.

**3.System Features**

* 1. ***Functional Requirements***
     1. ***Text Input***

Users should be provided with an intuitive interface to input textual descriptions.

*Display:* Present a text input field or area on the web application interface.

*Validation:* Ensure the input field supports input of varying lengths and characters.

*Feedback:* Provide real-time feedback to users, such as character count or input validation messages.

* + 1. ***Image Generation (Generator)***

Utilize Generative Adversarial Networks (GANs) to generate images based on input text.

*GAN Architecture:* Implement a GAN model architecture optimized for text-to-image generation.

*Text-to-Image Conversion:* Translate textual descriptions into visually realistic images using the GAN model.

*Image Diversity:* Ensure generated images are diverse and reflect the semantic meaning of input text.

* + 1. ***Model Optimization***

Modify the GAN model architecture to improve stability and convergence during training.

*Architecture Modification:* Implement changes to the GAN model architecture to enhance stability and convergence.

*Training Stability:* Ensure the modified model exhibits stable training behavior and converges to optimal solutions.

*Hyperparameter Tuning:* Fine-tune model hyperparameters to improve performance and efficiency.

* + 1. ***Web Application Deployment***

Deploy the text-to-image generator as a web application accessible via standard web browsers.

*Web Server Setup:* Configure a web server environment capable of hosting the text-to-image generator application.

*User Authentication:* Implement user authentication mechanisms to control access to the application.

*Scalability:* Ensure the web application can handle multiple concurrent users and scale resources as needed.

**4. External Interface Requirements**

**4.1 *User Interface***

The user interface (UI) should provide a seamless interaction experience for users to input text descriptions and visualize generated images.

*Text Input Field:* Present a text input field or area where users can type or paste textual descriptions.

*Image Display Area:* Display the generated images in a designated area within the UI.

***4.2 Hardware interface***

The text-to-image generator web application should be compatible with standard hardware configurations commonly used by end users.

Compatibility: Ensure the application runs smoothly on a range of hardware devices, including desktops, laptops, tablets, and smartphones.

Resource Usage: Optimize resource usage to ensure efficient performance on devices with varying hardware specifications.

**4.3 *Software interface***

The text-to-image generator web application may interface with external software components or libraries for specific functionalities.

Deep Learning Framework: Interface with a deep learning framework, such as PyTorch or TensorFlow, for implementing the GAN model and text-to-image generation algorithms.

Image Processing Libraries: Utilize image processing libraries, such as OpenCV, for preprocessing and post-processing tasks related to image generation.

**4.4 *Communication Interface***

The text-to-image generator web application may require communication interfaces for handling user requests, server-side processing, and data exchange.

HTTP Protocol: Implement HTTP-based communication protocols for client-server interactions, including request and response handling.

RESTful API: Design and expose a RESTful API for accessing application functionalities programmatically, if applicable.

**5. Non-functional Requirements**

### ***Performance Requirements***

### **Response Time**

The goal is to provide a seamless user experience by ensuring the system processes text inputs and generates the corresponding images rapidly. To achieve this, the backend should be optimized for quick data processing, leveraging efficient algorithms and perhaps GPU acceleration for image generation tasks. Monitoring tools can be employed to continuously assess response times, with automated scaling solutions in place to handle peaks in demand.

#### **Scalability**

Scalability is crucial for accommodating a growing user base and increasing demand. The system should employ scalable cloud services, containerization (e.g., Docker), and orchestration tools (e.g., Kubernetes) to dynamically adjust resources. Load testing can help identify bottlenecks and thresholds for scaling, ensuring the system remains responsive under varying loads.

#### **Efficiency**

*Efficiency* pertains to the system’s ability to deliver high-quality outputs without wasteful resource utilization. Implementing algorithmic optimizations, selecting efficient data storage and retrieval systems, and using energy-efficient computing resources are key strategies. Regular profiling and optimization cycles will help maintain efficiency as the system evolves.

### **5.2 *Safety Requirements***

#### **Data Protection**

User inputs must be treated with the highest security standards to prevent exposure to risks. Implementing stringent data handling and storage protocols, along with employing encryption for data in transit and at rest, are fundamental. Additionally, anonymizing data where possible can further protect user privacy.

#### **Error Handling**

The system should be designed to handle errors smoothly, providing users with clear, constructive feedback in case of issues. This involves implementing comprehensive logging, automated error detection, and user-friendly notification systems to inform users about what went wrong and possibly how to correct their input.

### **5.3 *Security Requirements***

#### **Authentication and Authorization**

For platforms managing user accounts or sensitive data, robust authentication mechanisms (e.g., OAuth, Multi-Factor Authentication) must be in place. Authorization frameworks (e.g., RBAC, ACLs) ensure users can only access data and actions appropriate to their roles.

#### **Input Validation**

Protecting the system from common web vulnerabilities starts with stringent input validation to ensure only appropriately formatted data is processed. This includes checks against SQL injection, XSS attacks, and ensuring uploaded content is scanned for malware.

#### **Data Encryption**

All data transmitted between the client and server must be encrypted using HTTPS to prevent eavesdropping and man-in-the-middle attacks. Sensitive data stored by the system should also be encrypted, employing strong encryption standards.

### **5.4 *Software Quality Attributes***

#### **Reliability**

The system must consistently produce accurate and high-quality images that match the users’ text descriptions. This involves thorough testing, including unit, integration, and end-to-end tests, to ensure the system’s components work reliably together.

#### **Usability**

The application interface should be designed with the user in mind, offering a clean, intuitive experience. This includes clear navigation, accessible design, and providing immediate, understandable feedback to user actions. User testing can offer valuable insights into improving the interface.

#### **Maintainability**

The codebase should be well-documented, adhering to coding standards, and structured for easy maintenance and updates. Employing continuous integration/continuous deployment (CI/CD) practices ensures updates can be rolled out smoothly and efficiently.

#### **Portability**

The application should function across different devices, operating systems, and browsers, ensuring a broad user base can access the service. Responsive design principles and cross-platform compatibility testing are key strategies.