**Software Requirements Specification (SRS)**

**Project Name:** AI Text-to-Image Generator  
**Platform:** Expo React Native (Frontend) + FastAPI (Backend) + MySQL (Database)  
**Backend Architecture**: Custom Python backend using PyTorch and a custom Diffusion **Version:** 1.0 **Author:** Jatin Nath **Date:** July 1, 2025

**1. Introduction**

**1.1 Purpose**

This document defines the software requirements for a mobile-based AI-powered Text-to-Image application. The app allows users to generate images using a custom Stable Diffusion v1.5 model via natural language prompts, including image-to-image transformation, generation queuing, and user management.

**1.2 Intended Audience**

* Developers working on frontend/backend
* Project managers
* Testers
* End users (Admin and General Users)

**1.3 Scope**

The application enables:

* Secure user authentication
* Text-to-image and image-to-image generation
* Negative prompt support
* Generation queuing and cancellation
* Prompt/image history
* Admin dashboard for monitoring usage
* Real-time progress tracking via backend
* Cross-platform accessibility via Expo Go

**2. Overall Description**

**2.1 Product Perspective**

The system is composed of:

* Frontend: React Native (Expo), communicates via Axios with the backend
* Backend: FastAPI REST API with async queueing and Torch-based Stable Diffusion model
* Database: MySQL for storing users, prompts, queue items, and images

**2.2 Product Functions**

* User Login/Register
* Prompt Submission (Text/Image)
* Image Generation via Stable Diffusion
* Queue Management (add, cancel, track)
* View History and Queue Status
* Admin Control Panel

**2.3 User Classes and Characteristics**

| **User Type** | **Permissions** |
| --- | --- |
| **General** | **Register, Login, Generate, Queue, History** |
| **Admin** | **View all users, all images, monitor usage**  **Edit users** |

**3. Specific Requirements**

**3.1 Functional Requirements**

**3.1.1 User Management**

* Users can register/login with username & password
* User sessions are persisted
* On logout, clear generation queue

**3.1.2 Prompt Submission**

* Text-to-Image input with:
  + Prompt
  + Negative Prompt
* Image-to-Image input with:
  + Initial Image (base64)
  + Prompt and negative prompt

**3.1.3 Generation Queue**

* Async generation queue handled by FastAPI with asyncio.Lock
* Only one generation is processed at a time
* Each queue item includes:
  + User ID
  + Prompt
  + Type (txt2img or img2img)
  + Optional image
* Cancel queued prompt by user
* Cancel active generation mid-way
* Track queue progress (via tqdm)

**3.1.4 History & Display**

* On completion, generated image stored with prompt & timestamp
* Display all completed prompts on "MyQueue" screen
* Store in MySQL: images, prompts, users, queue

**3.1.5 Admin Panel**

* View all registered users
* On selecting a user, view:
  + All their prompts
  + Generated images

**3.2 Non-Functional Requirements**

**3.2.1 Performance**

* CPU and GPU inference support
* **Prompt response time ranges from ~4 to 6 minutes based on GPU compatibility and hardware performance**
* Queue must maintain sequential processing even with concurrency

**3.2.2 Scalability**

* Backend designed to handle 100+ concurrent users
* Future support for distributed inference (optional)

**3.2.3 Usability**

* Mobile UI optimized using Expo
* Scrollable history
* Input validation (empty prompts, large images)

**3.2.4 Reliability**

* Handle queue crashes gracefully
* Prevent duplicate prompts
* Ensure cancelling generation doesn’t leave locks hanging

**3.2.5 Security**

* Input sanitation
* Image size/type restrictions

**3.2.6 Maintainability**

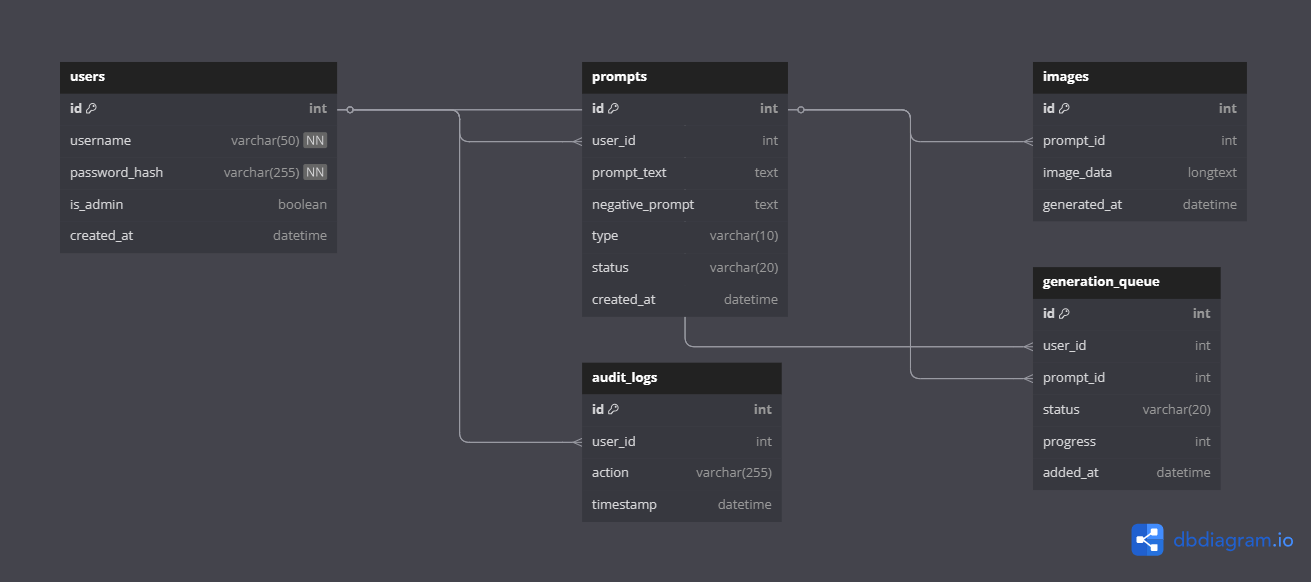
* Modular codebase with separation of concerns (Frontend, Backend, Model)
* Dockerized backend for consistent deployment
* Use of logging and exception tracking for debugging and uptime

**3.2.7 Portability**

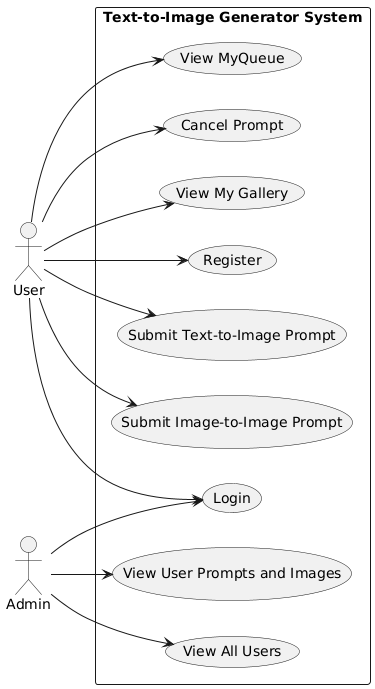
* Frontend runs on Android, iOS, and Web via Expo
* Backend deployable on any machine supporting Python and CUDA
* Model weights and configs externalized for replacement or upgrades

**4. System Models**

**4.1 ER Diagram**



**4.2 Use Case Diagram**



**5. Appendices**

**5.1 Glossary**

| **Term** | **Definition** |
| --- | --- |
| Prompt | The input text used to generate an image |
| Negative Prompt | Specifies what the image should **not** contain |
| Img2Img | Image-to-image generation using initial base image |
| Queue | System managing multiple prompt submissions |
| CLIP | Contrastive Language–Image Pretraining model |

**5.2 Future Enhancements**

* Support for multiple concurrent generations using GPU batching
* Feature to share generated images via social media
* Style presets or templates (anime, realistic, abstract)
* User profiles with favourites and private/public image collections
* Support for multilingual prompts and translation
* Server-side caching for repeated prompts