SOMAIYA
VIDYAVIHAR UNIVERSITY

# K. J. Somaiya School of Engineering

**Department of Computer Engineering**

| | |
|---|---|
| **Batch: A4** | **Roll No.: 16010122139** |
| **Experiment No 3** | |
| **Group No:** | |

**Title: Prepare design document and Plan of project**

**Objective:** Chapter No.3 of Mini Project Report will include detailed design document and plan of implementation of the project

**Expected Outcome of Experiment:**

| | **At the end of successful completion of the course the student will be able to** |
|---|---|
| CO2 | Identify various hardware and software requirements for problem solution |
| CO5 | Prepare a technical report based on the Mini project. |

**Books/ Journals/ Websites referred:**

**1.**
**2.**
**3.**

**The students are expected to prepare chapter no 3 in the format given below**

# Chapter 3

## Design Document and Project plan

*A **design document** is crucial in a software project because it serves as a blueprint that outlines the architecture, components, data flow, and technical specifications of the system before implementation **Clear Vision & Planning** will improve collaboration with in the team members.*

## 1. Introduction

**Purpose:**
The purpose of this document is to provide a detailed description of the design and implementation strategy for the "AI Image Generator" mini-project. This document serves as a blueprint for developers, testers, and stakeholders involved.

**Expected Audience:**
This document is intended for software developers, testers, instructors, and evaluators involved in the project lifecycle.

**Scope:**
The AI Image Generator leverages pre-trained models to generate images from textual prompts using the Stable Diffusion model.

**Definitions, Acronyms, and Abbreviations:**

- **AI:** Artificial Intelligence
- **GPU:** Graphics Processing Unit
- **Stable Diffusion:** A deep learning-based generative model utilizing diffusion processes to gradually generate high-quality images from random noise guided by textual prompts.
- **UML:** Unified Modeling Language
- **CUDA:** Compute Unified Device Architecture, a parallel computing platform by NVIDIA.

**References:**

- Diffusers Documentation
- Stable Diffusion Documentation
- PyTorch Official Documentation
- CUDA Official Documentation

## 2. System Overview

### 2.1 System Architecture

The architecture consists of the following key components:

- **User Interface:** Captures textual input from the user.
- **Stable Diffusion Pipeline:** Generates images based on textual prompts by leveraging pre-trained Stable Diffusion models.
- **Scheduler (DPMSolverMultistepScheduler):** Controls and optimizes the steps involved in the diffusion process, improving efficiency and quality of image generation.
- **GPU/CUDA:** Provides necessary computational power through hardware acceleration to handle complex diffusion processes efficiently.

### 2.2 Design Goals

- **Scalability:** Capable of accommodating additional models and larger datasets.
- **Performance:** Fast and efficient image generation.
- **Maintainability:** Modular structure for ease of updates and troubleshooting.
- **Usability:** Intuitive and user-friendly interface.
- **Reliability:** Consistent and dependable image output quality.

## 3. Detailed Design

### 3.1 Module Description

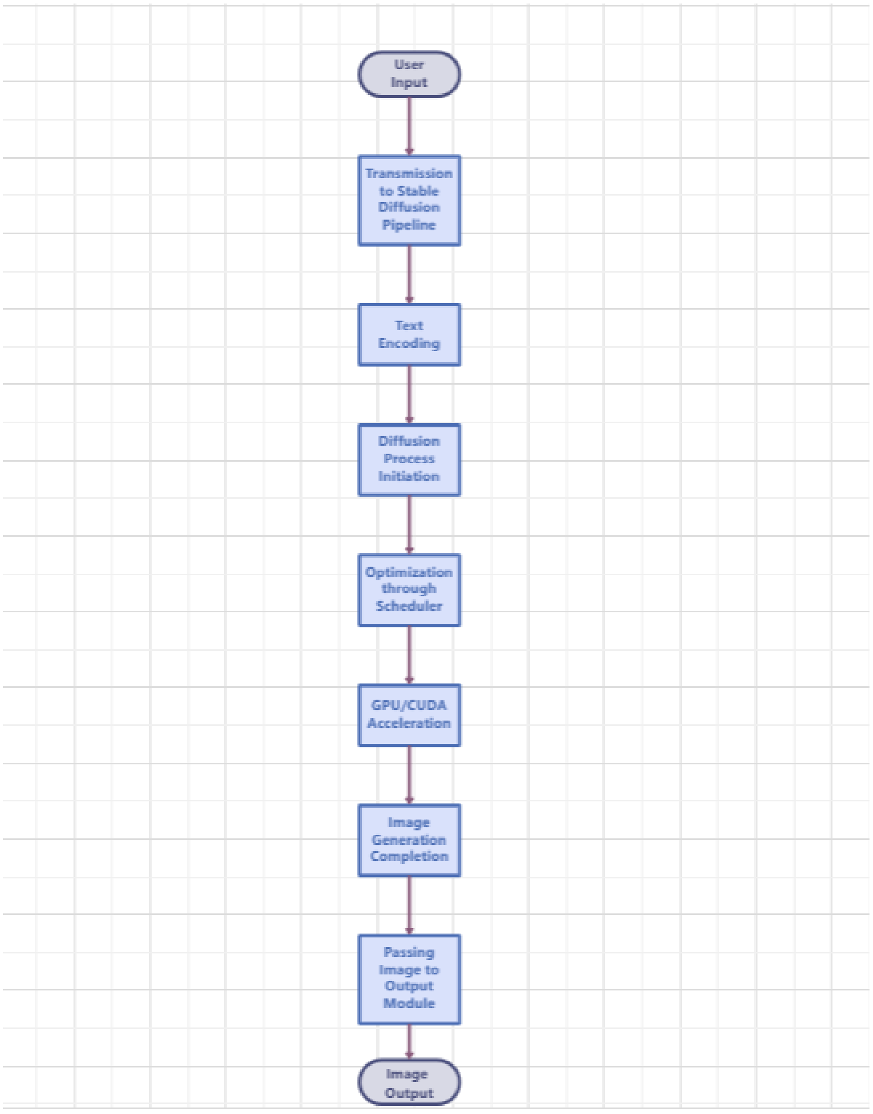
- **Input Module:** Responsible for capturing and preprocessing textual input from users.
- **Image Generation Module:** Core module utilizing Stable Diffusion to convert processed textual prompts into images.
- **Output Module:** Manages visualization and delivery of generated images to the user.

### 3.2 Data Flow & Components

**Data Flow:**

1. User provides a textual prompt.
2. Input Module preprocesses the text and forwards it to the Image Generation Module.
3. Image Generation Module employs Stable Diffusion models to generate images.
4. Generated images are passed to the Output Module for user display.

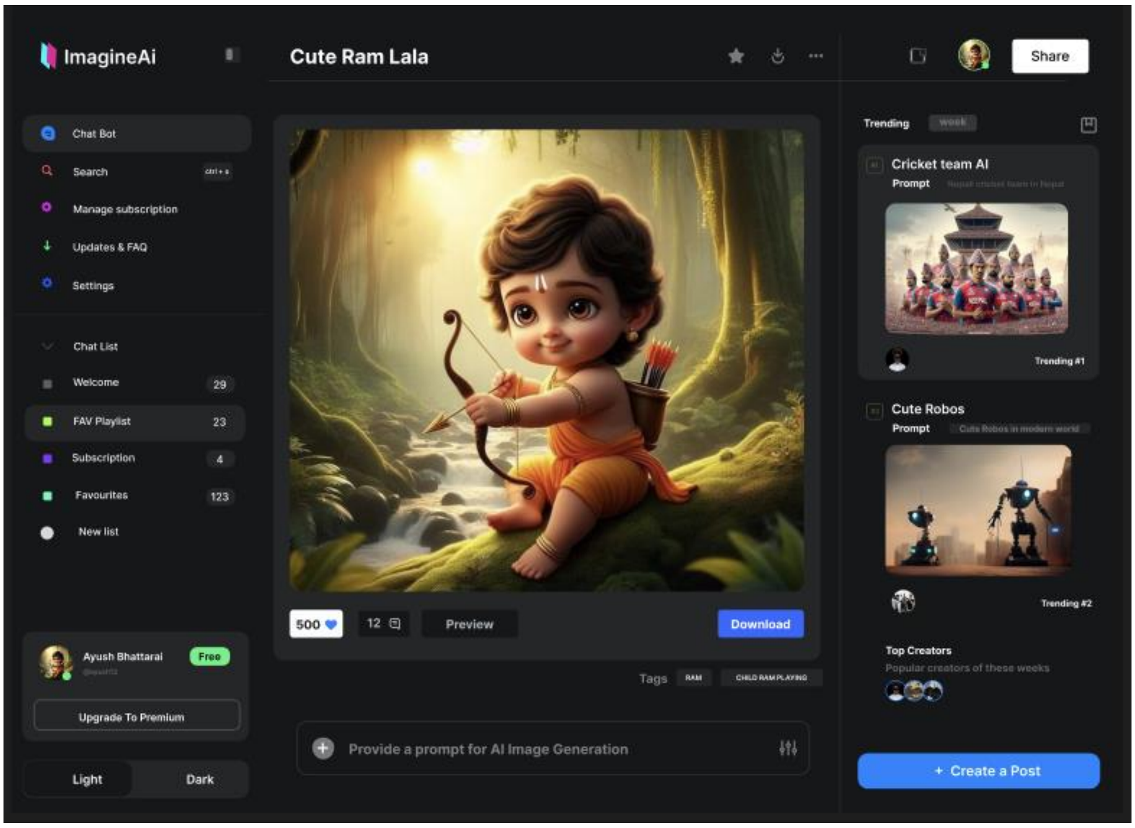
**UML/Flow chart Diagram Instructions:**


User Input
Transmission to Stable Diffusion Pipeline
Text Encoding
Diffusion Process Initiation
Optimization through Scheduler
GPU/CUDA Acceleration
Image Generation Completion
Passing Image to Output Module
Image Output

### 3.3 Database Design

No persistent database is required as the project primarily deals with real-time data processing and generation without storage.

### 3.4 User Interface Design



### 3.5 External Interfaces

- **Hugging Face Diffusers:** API integration for Stable Diffusion model access.
- **PyTorch:** Framework for deep learning and tensor computations.
- **CUDA:** GPU acceleration support for computational efficiency.

## 4. Project and Implementation Plan

### 4.1 Deliverables

- Complete and modular source code
- Detailed user and technical documentation
- Installation and deployment guides
- Presentation slides and demonstration

### 4.2 Team Roles and Responsibilities and Delivery Schedule

| Name of the Task | Developer | Tester | Approver | Date of Delivery |
|---|---|---|---|---|
| Setup Environment | Toyash | Smit | Khushi | January |
| Model Integration | Smit | Toyash | Khushi | March |

| | | | | |
|---|---|---|---|---|
| UI/UX Design | Khushi | Toyash | Smit | March |
| Testing and Debugging | Smit | Khushi | Toyash | April |
| Final Deployment | Toyash | Khushi | Smit | April |

### 4.3 Risk Management Plan

| Risk | Mitigation Strategy |
|---|---|
| Model compatibility issues | Early validation of model and dependency versions |
| GPU availability | Alternative arrangements via cloud platforms (e.g., Google Colab, AWS) |
| Performance issues | Regular profiling and incremental optimization |
| User Interface usability | Continuous feedback loops and iterative design improvements |

## 5. Testing & Deployment Plan

### 5.1 Testing Strategy

- **Unit Testing:** Verify individual module functionalities.
- **Integration Testing:** Validate interactions and data flow between modules.
- **System Testing:** Evaluate complete system performance and reliability.
- **User Acceptance Testing:** Ensure the final product meets user expectations.

### 5.2 Deployment Plan

- **Environment Setup:** Local development with Jupyter Notebooks and GPU.
- **Deployment Method:** Optionally containerize application using Docker for easy portability.
- **Rollback Strategy:** Maintain comprehensive version control with Git for efficient rollback capabilities.

*The next chapter, chapter no . 4 will explain test cases, test plan and test reports in detail*