Hackathon Project Phases Template

LogoCraft: Innovation Logo Generation with Diffusion

Team Name:

Team Zen

Team Members:

- Chandrika Tulluri
- Thollikonda Hema Srilakshmi
- T.Purna Nagasri

Phase-1: Brainstorming & Ideation

Objective:

To demonstrate the effective use of diffusion technology for generating unique and creative logos and develop a system that leverages diffusion technology to generate creative and unique logos from user input.

Key Points:

Problem Statement:

- Many startups lack the resources to hire professional logo designers, resulting in generic or unprofessional branding.

- Existing logo generation tools often provide limited customization options, failing to cater to the unique needs and preferences of individual users.

Proposed Solution:

- Develop a diffusion-based logo generation tool that allows startups to quickly and easily create unique and high-quality logos tailored to their specific industry and brand identity, even with limited design expertise.
- Create a platform that allows users to deeply customize their logo generation experience, incorporating specific style preferences, brand values, and even visual references to guide the diffusion model and achieve a truly personalized logo.

Target Users:

- Startups and Small Businesses: Often have limited budgets and need a quick, affordable way to create professional-looking logos for their branding.
- Designers (as a tool): Even professional designers can benefit from tools that spark creativity and speed up the initial design phase.

Expected Outcome:

- A logo generation tool using diffusion technology, can be multifaceted, encompassing both tangible deliverables and broader impacts.

Phase-2: Requirement Analysis

Objective:

Define technical and functional requirements for logocraft: logo generation using diffusion technology.

Key Points:

Technical Requirements:

Backend: Python (Flask), TensorFlow, Hugging Face Transformers,Diffusers.

Frontend: JavaScript (React), NPM

Cloud: AWS (or Google Cloud, Azure), Docker

Other: Git, VS Code.

Functional Requirements:

- ManyUsers provide text prompts (description) for their desired logo, and the system generates logo variations.
- Users choose from predefined logo style categories (eg., modern, vintage, geomentric).
- A simple and intuitive interface for prompt input, style/color selection, logo display, and download.

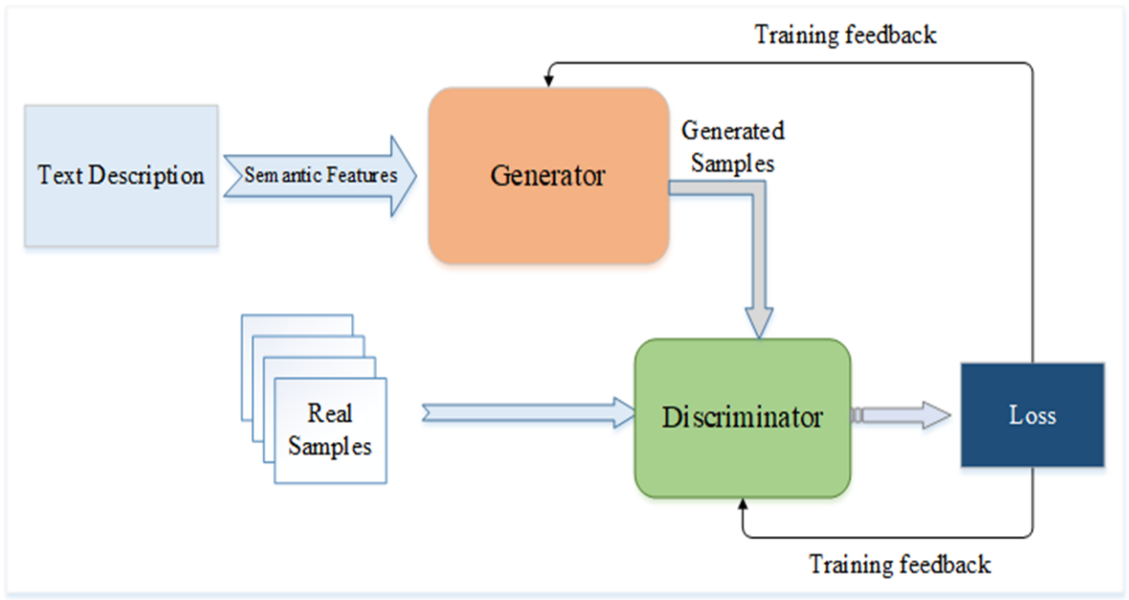
Constraints & Challenges:

- Diffusion models can sometimes struggle to generalize to very specific or logo requests. They might perform best with common logo styles.
- While you can guide the generation with prompts and styles, it can be challenging to achieve very precise control over the final logo design. Diffusion models can be somewhat unpredictable.

Phase-3: Project Design

Objective:

Develop the architecture and user flow of this application.



Key Points:

System Architecture Diagram:

- User interacts with the Frontend then Frontend sends requests to the Backend API.
- Backend API uses the Model and Dataset to generate logos.
- Logos are sent back to the Frontend and displayed to the user.

User Flow:

- User lands on Logo Generation Input.
- User enters a text prompt, selects a style, and (optionally) chooses a color palette.
- User clicks "Generate."
- User is taken to Logo Display and Selection.
- User browses the generated logos and selects one.
- User is taken to Logo Download.

UI/UX Considerations:

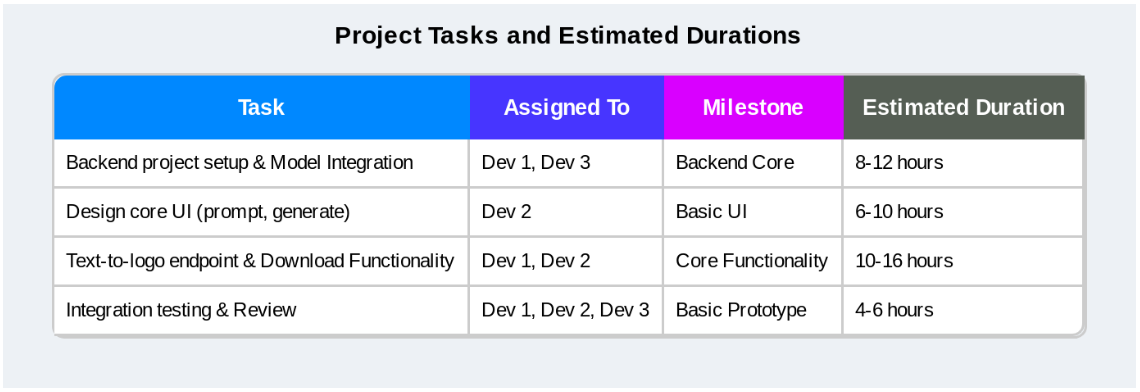
- Make it easy for users to navigate between the different screens (input, display, download). Use clear labels and intuitive icons.
- Keep the interface clean and uncluttered. Avoid overwhelming users with too many options or complex layouts. Focus on the essential elements.

Phase-4: Project Planning (Agile Methodologies)

Objective:

Break down development tasks for efficient completion.

Key Points:



Project Tasks and Estimated Durations

Task	Assigned To	Milestone	Estimated Duration
Backend project setup & Model Integration	Dev 1, Dev 3	Backend Core	8-12 hours
Design core UI (prompt, generate)	Dev 2	Basic UI	6-10 hours
Text-to-logo endpoint & Download Functionality	Dev 1, Dev 2	Core Functionality	10-16 hours
Integration testing & Review	Dev 1, Dev 2, Dev 3	Basic Prototype	4-6 hours

Sprint Planning:

- ManyThis sprint's goal is a functional Logocraft prototype. Dev 1 handles backend and model integration, Dev 2 the frontend UI and downloads, and Dev 3 contributes to backend setup and color features. Integration testing is a team effort . The sprint targets Backend Core, Basic UI, Core Functionality, and a Basic Prototype milestones. Estimated durations are provided.

Milestones:

- Backend Core: Backend setup and model integrated.
- Basic UI: Core UI elements designed.
- API & Download: Text-to-logo API and download working.
- Deployed Prototype: Deployed with color handling.
- Functional Prototype: Integrated, tested, and bug-fixed.

Phase-5: Project Development

Objective:

Implement core features of the logocraft.

Key Points:

Technology Stack Used:

- Backend: Python (Flask).
- Frontend: JavaScript (React), NPM.
- Cloud: AWS (or Google Cloud, Azure), Docker.

Development Process:

- 1. Setup:Install libraries (torch, diffusers, transformers, etc.).
- 2. Data: Load, preprocess, and prepare logo data.
- 3. Model:Train/fine-tune a diffusion model.
- 4. Generate:Implement logo generation function using the model.
- 5. UI: Build user interface (e.g., Flask/FastAPI) to interact with the model.
- 6. Deploy: Containerize (Docker) and deploy to cloud.

Challenges & Fixes:

- Challenge: Generated logos might be generic or uninspired.
  Fix: Varied prompts, refined model, diverse data.
- Challenge: Handling many users/requests.
  Fix: Cloud platform, optimized backened, load balancing.

Phase-6: Functional & Performance Testing

Objective:

Ensure the logocraft works as expected.

Key Points:

Test Cases Executed:

- Data: Empty/corrupted/duplicate images, varying sizes, large.
  Check: Handles gracefully, logs errors, resizes, efficient processing.

- Model: Insufficient/biased data, hyperparameter tuning.
  Check: Quality of logos, optimal parameters, loss decrease.

Bug Fixes & Improvements:

- Bug: System crashes when encountering a corrupted image file.
  Fix: Implement robust error handling to skip corrupted files and log the error.
  Improvement: Add data validation to check image formats and dimensions before processing.
- Bug: Dataset contains duplicate logos, leading to overfitting.
  Fix: Implement a deduplication step during data preprocessing.
  Improvement: Explore data augmentation techniques to increase dataset diversity without adding duplicates.

Final Submission

- **Project Report Based on the templates**
- **Demo Video (3-5 Minutes)**
- **GitHub/Code Repository Link**
- **Presentation**