**Hackathon Project Phases Template** for the **Transforming voice prompts into visual creations using Transformers** project.

---

**Hackathon Project Phases Template**

**Project Title:** Transforming voice prompts to visual creation using transformer

**Auto Sage App Using Gemini Flash**

**Team Name:**

Team GIRLIES

**Team Members:**

- P. Siri
- S. Sri Nagavalli
- S. Sathvika Shetty
- V. Lohitha

---

**Phase-1: Brainstorming & Ideation**

**Objective:**

The objective of transforming voice prompts into visual creations using transformers is to enable seamless, intuitive interaction between speech and visual content. This process allows users to generate images or visuals from voice descriptions, enhancing accessibility, creativity, and efficiency in content creation, while leveraging advanced language and vision models to ensure contextually accurate and relevant results**.**

**Key Points:**

1. **Problem Statement:**
   Audio2Art : Transforming Voice Prompts into Visual Creations using Transformers
2. **Proposed Solution:**
   **Voice Input Processing**: Convert voice prompts to text using speech recognition models.

   **Text Understanding**: Use NLP transformers (e.g., GPT, BERT) to interpret the context and intent of the transcribed text.

   **Text-to-Image Generation**: Employ a text-to-image model (e.g., DALL·E, Stable Diffusion) to create visuals based on the interpreted text.

**User Feedback**: Allow users to refine their input to enhance the generated visuals.

This process bridges speech and vision, enabling users to generate images from voice commands.

3. **Target Users:**

**General Consumers**: People looking for an easy and creative way to generate visuals for personal projects, social media, or entertainment without needing graphic design skills.

**Content Creators and Influencers**: Individuals in need of quick visual content creation for blogs, videos, or social media posts, leveraging voice commands to speed up the creative process.

**Designers and Artists**: Professionals who can use voice commands to streamline their creative workflows, helping with ideation and concept generation.

**Educators and Students**: For educational purposes, enabling users to visualize complex concepts or create custom illustrations for learning materials.

4. **Expected Outcome:**

**Seamless User Experience**: Users can generate high-quality visuals quickly and effortlessly by simply speaking, making the process more intuitive and accessible.

**Increased Creativity and Productivity**: By lowering the barriers to creating visuals, users—whether casual creators or professionals—can more easily bring their ideas to life, fostering greater creativity and efficiency.

**Improved Accessibility**: Voice-driven visual generation provides a valuable tool for people with disabilities, enabling them to interact with technology and create content more easily.

**Enhanced Personalization**: The system can generate highly relevant and customized visuals based on the user's voice descriptions, allowing for more personalized content creation.

**Faster Content Creation**: For businesses, educators, marketers, and content creators, this system speeds up the design process, helping them produce visuals more quickly for various purposes like advertising, teaching materials, or social media posts.

**Higher Engagement**: As voice and visual content merge, users may have more engaging and dynamic experiences, which can drive higher user satisfaction and interaction with the system.

**Phase-2: Requirement Analysis**

**Objective:**

1. Technical Requirements

a. Preprocessing and Data Handling

- Voice-to-Text Conversion:
    - Automatic Speech Recognition (ASR): You need an ASR system (like Google Speech-to-Text or Whisper) to convert spoken voice prompts into text. This is the first step before feeding text to transformer models.
    - Preprocessing: The transcribed text may need to be cleaned (e.g., removing unnecessary fillers like "um," correcting spelling errors, etc.).

b. Model Architecture

- Text Encoder (Transformer Models):
    - Transformer-based models like GPT or BERT for understanding and processing the voice prompt once converted to text.
    - These models would help understand the intent, key themes, and details within the voice prompt.
- Vision Transformer (ViT) or Generative Models:
    - DALL·E or other image generation models based on transformers (like CLIP and VQGAN+CLIP) can generate images based on the processed text prompt.
    - The model should be able to combine the contextual information from the text input with visual representations, ensuring the generated image matches the description accurately.
- Cross-Modal Interaction (Text-to-Image Mapping):
    - Text-to-Image Mapping: The integration of NLP and image generation is crucial. The text encoder extracts semantic features, and the image generation model (like DALL·E) needs to translate these features into a visual form.
    - Attention Mechanisms: Transformers should have attention mechanisms that allow them to focus on key parts of the input to guide image creation.

c. Training and Fine-Tuning

- Pre-training Data: Models like DALL·E are pre-trained on large multimodal datasets (images + text descriptions). The training needs to cover a broad range of contexts to ensure diverse image generation capabilities.
- Fine-Tuning: Depending on the domain, fine-tuning may be required to improve the accuracy of the generated images based on voice prompts that are specific (e.g., for a particular industry or artistic style).

d. Computational Resources

- Cloud Infrastructure / GPUs: Transformer models, especially generative ones, require substantial computational resources. Leveraging GPU clusters or cloud-based systems (e.g., AWS, Google Cloud) is crucial for both real-time and batch processing.

e. Latency and Real-Time Constraints

- Low Latency: Given that the system may need to generate images in real-time based on voice prompts, optimizing for low-latency processing is essential.
- Batch vs. Real-Time Processing: The ability to handle both single requests (real-time voice prompts) and batch processing (processing large volumes of prompts) is an important technical consideration.

---

2. Functional Requirements

a. User Interface and Interaction

- Voice Input Interface: A functional system for receiving and transcribing voice prompts. This can be achieved using voice assistants or standalone ASR models.
- Visual Output Interface: Once the image is generated, it should be presented to the user via a visual interface (web-based, mobile app, etc.).

b. Voice-to-Text Accuracy

- The system must accurately transcribe diverse accents, languages, and tones.
- Real-time transcriptions should be optimized to prevent errors that could affect the generated visual content.

c. Text Interpretation and Image Generation

- Contextual Understanding: The model should be able to generate images that are consistent with the voice prompt’s context and intent. For example, a voice prompt requesting "A futuristic city at sunset" should result in an image of a city in a sunset setting with futuristic architecture.

- Semantic Analysis: The model needs to perform detailed semantic analysis to understand not just the object in the prompt (e.g., "dog"), but also properties like color, shape, and environment

**Key Points:**

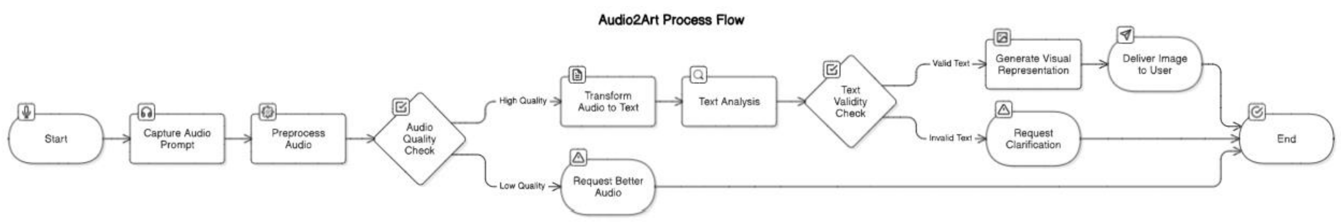
1. **Technical Requirements:**
    - Programming Language: **Python**
    - Backend: **Gemini Flash API**
    - Frontend: **Google Collab**
    - Database: **Not required initially**
2. **Functional Requirements:**
    - Ability to **fetch voice prompts and visuals** using Gemini Flash API.
    - The system must accept voice input from a microphone or uploaded audio file.
    - The system must transcribe speech into text with high accuracy, supporting multiple languages and accents.
    - The system should support accessibility features like voice commands and text-to-speech feedback.
    - The system must log and monitor API usage for security and performance tracking.
3. **Constraints & Challenges:**
    - Ensuring real-time updates from **Gemini API**.
    - High GPU demand & latency issues.
    - Speech-to-text errors & prompt misinterpretation.
    - UI complexity & real-time customization limits.
    - Content moderation, bias, & privacy concerns

---

**Phase-3: Project Design**

**Objective:**

Develop the architecture and user flow of the application.



**Key Points:**

**1.System Architecture:**

- **Voice Input** → Speech-to-Text (SpeechRecognition, Wav2Vec).
- **Text Processing** → NLP refinement (GPT, T5).
- **Image Generation** → Diffusion models (Stable Diffusion, DALLE).
- **Backend** → Fast API, Gemini platform
- **Frontend** → Web/App UI (vsc, Google Collab).
- **Deployment** → Cloud GPUs (AWS, GCP) or Edge (ONNX).

**2.User Flow:**

- **User Inputs Voice** → Speaks into the app or uploads an audio file.
- **Speech-to-Text (STT) Processing** → Converts speech into text.
- **Text Refinement & Optimization** → NLP model enhances the prompt.
- **Image Generation** → AI model (Stable Diffusion, DALLE) creates an image.
- **User Reviews & Edits** → Options to refine text or regenerate images.
- **Download/Share** → User saves or shares the final image. **UI/ UX**

**3.Considerations:**

**Accuracy** → Improve STT and prompt refinement.

**Performance** → Optimize latency with GPUs.

**UX** → Simple UI, customization options.

**Security** → Content moderation, privacy compliance.

**Scalability** → Cloud vs. edge deployment

**Phase-4: Project Planning (Agile Methodologies)**

**Objective:**

Break down development tasks for efficient completion.

| Sprint | Task | Priority | Duration | Deadline | Assigned To | Dependencies | Expected Outcome |
|---|---|---|---|---|---|---|---|
| Sprint 1 | Environment Setup & API Integration | ● High | 6 hours (Day 1) | End of Day 1 | P. Siri | Google API Key, Python, Google collab setup | API connection established & working |
| Sprint 1 | Frontend UI Development | □ Medium | 2 hours (Day 1) | End of Day 1 | S. Sri Nagavalli | API response format finalized | Basic UI with input fields |
| Sprint 2 | Functionality of the Project | ● High | 3 hours (Day 2) | Mid-Day 2 | S. Sathvika Shetty | API response, UI elements ready | Search functionality with filters |
| Sprint 2 | Error Handling & Debugging | ● High | 1.5 hours (Day 2) | Mid-Day 2 | P. Siri S. Sri Nagavalli | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI Enhancements | □ Medium | 1.5 hours (Day 2) | Mid-Day 2 | V. Lohitha | API response, UI layout completed | Responsive UI, better user experience |
| Sprint 3 | Final Presentation & Deployment | □ Low | 1 hour (Day 2) | End of Day 2 | Entire Team | Working prototype | Demo-ready project |

**Sprint Planning with Priorities**

**Sprint 1 – Setup & Integration (Day 1)**

(● **High Priority)** Set up the **environment** & install dependencies.
(● **High Priority)** Integrate **Google Gemini API**.
(□ **Medium Priority)** Build a **basic UI with input fields**.

**Sprint 2 – Core Features & Debugging (Day 2)**

(● **High Priority)** Implement **search & comparison functionalities**.
(● **High Priority)** Debug API issues & handle **errors in queries**.

**Sprint 3 – Testing, Enhancements & Submission (Day 2)**

(▯ **Medium Priority)** Test API responses, refine UI, & fix UI bugs.
(▯ **Low Priority)** Final **demo preparation & deployment**.

---

**Phase-5: Project Development**

**Objective:**

Transforming Voice prompts into Visual creations using Transformers

**Key Points:**

1. **Technology Stack Used:**
   - **Frontend:** Google Collab
   - **Backend:** Google Gemini Flash API
   - **Programming Language:** Python
2. **Development Process:**
   - Implement **API key authentication** and **Gemini API integration**.
   - Develop **the source code of audio-to-text and text-to-image.**
   - Optimize **search queries for performance and relevance**.
3. **Challenges & Fixes:**
   - **Challenge:** Delayed API response times.
     **Fix:** Implement **caching** to store frequently queried results.
   - **Challenge:** Limited API calls per minute.
     **Fix:** Optimize queries to fetch **only necessary data**.

---

**Phase-6: Functional & Performance Testing**

**Objective:**

Ensure that the source code works as expected.

| Test Case ID | Category | Test Scenario | Expected Outcome | Status | Tester |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| TC-001 | Functional Testing | Query "Best budget cars under ₹10 lakh" | Relevant budget cars should be displayed. | ✅ Passed | P. Siri |
| TC-002 | Functional Testing | Query "Motorcycle maintenance tips for winter" | Seasonal tips should be provided. | ✅ Passed | S.Sri Nagavalli |
| TC-003 | Performance Testing | API response time under 500ms | API should return results quickly. | ⚠ Needs Optimization | S. Sathvika Shetty |
| TC-004 | Bug Fixes & Improvements | Fixed incorrect API responses. | Data accuracy should be improved. | ✅ Fixed | P. Siri |
| TC-005 | Final Validation | Ensure UI is responsive across devices. | UI should work on mobile & desktop. | ❌ Failed - UI broken o n mobile | V. Lohitha |
| TC-006 | Deployment Testing | Host the app using Streamlit Sharing | App should be accessible online. | 🚀 Deployed | Devops |

---

**Final Submission**

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