**Hackathon Project Phases Template** for the **LogocraftApp** project.

Hackathon Project Phases Template

# Project Title:

**LogocraftApp Using Gemini Flash**

# Team Name:

SSSR

# Team Members:

* D.SANDEEP
* G.SRAVANI
* SAHARSHITHA
* REVATHI

# Phase-1: Brainstorming &Ideation

## Objective:

The **objective of developing a logo designing application** is to create a tool that allows users—whether they are professionals or beginners—to design logos quickly and efficiently, with a range of customizable options. This application should provide an intuitive and user-friendly interface that enables creative expression while ensuring that users can produce high-quality logos for their personal or business needs.

## Key Points:

### 1. ****Design Tools:****

* **Typography Support**: Offer a variety of fonts and typography settings such as font size, letter spacing, and alignment. Allow users to upload custom fonts for more design flexibility.
* **Icon/Image Upload**: Enable users to upload images (e.g., icons, vector art) that they can use in their logos, along with basic editing features like resizing and cropping.
* **Alignment and Spacing**: Include tools for aligning objects, snapping to grid, and adjusting spacing between elements for a clean design.

### ****2. High-Quality Output:****

* **Export Formats**: Allow users to export their logos in high-quality formats like PNG, SVG (vector for scalability), JPEG, and PDF for printing. Offer multiple resolution options based on user needs (e.g., high resolution for print and lower resolution for web).
* **Transparent Background**: Ensure that users can export logos with transparent backgrounds, especially useful for overlaying logos on other media.
* **File Size Optimization**: Ensure exported logos are optimized for various purposes (web, print) without compromising quality.

### ****3. Ease of Use for Non-Designers:****

* **Beginner-Friendly**: The application should be accessible to users who don't have a design background, with easy-to-understand features and minimal learning curve.
* **Guided Steps**: Provide helpful onboarding tutorials, tooltips, or even pre-designed logo templates to assist beginners in getting started.
* **Error Prevention**: Offer warnings if users make incompatible design choices (e.g., low contrast, hard-to-read text) and suggest improvements.

### 4. ****Advanced Design Features (For Professionals):****

* **Vector Editing**: Provide vector editing tools to professionals who need precise control over their logos.
* **Gradient Support**: Allow for the use of gradient fills and effects on both text and shapes.
* **Export to Vector Files**: Offer the ability to export logos as vector files (SVG, EPS) for professional-grade printing and scalability.
* **Advanced Image Effects**: Allow users to apply image effects like shadow, glow, blur, and opacity.

### 5. ****Performance and Scalability:****

* **Fast Rendering**: Ensure the app can handle complex designs without lag or slow rendering, especially with larger files.
* **Scalable Infrastructure**: If cloud-based, make sure that the backend is capable of handling many concurrent users, with a scalable database and storage solution.

### ****6. Advanced Design Features (For Professionals):****

* **Vector Editing**: Provide vector editing tools to professionals who need precise control over their logos.
* **Gradient Support**: Allow for the use of gradient fills and effects on both text and shapes.
* **Export to Vector Files**: Offer the ability to export logos as vector files (SVG, EPS) for professional-grade printing and scalability.
* **Advanced Image Effects**: Allow users to apply image effects like shadow, glow, blur, and opacity.

# Phase-2: Requirement Analysis

## Objective:

Define the technical and functional requirements for the LogosartApp.

## Key Points:

### ****Technical Requirements:****

These requirements outline how the logo designing app should be built and what technologies should be used to meet the functional needs.

#### 1. ****Platform Compatibility****

* **Web-Based**: The application should be accessible on major browsers (Chrome, Firefox, Safari, Edge).
* **Mobile Compatibility**: Ensure that the design tool is mobile-responsive and works on tablets and smartphones, or develop a native app for iOS/Android.
* **Cross-Browser Compatibility**: The app should function consistently across all modern web browsers.

#### 2. ****Frontend Technologies****

* **HTML5/CSS3**: For page structure and styling.
* **JavaScript**: For interactive functionality (shapes, text manipulation, etc.).
* **Canvas API**: Native HTML5 Canvas to render the design elements like shapes and text. Alternatively, use libraries like **Fabric.js** or **Konva.js** to simplify the drawing and manipulation of objects.
* **React.js**: Optionally, React can be used to build a modern, component-based UI.
* **Redux**: Manage the application's state, especially when dealing with complex UI components (e.g., managing selected objects, canvas state).
* **CSS Framework**: Use CSS frameworks like Bootstrap or Tailwind CSS to speed up responsive design.

#### 3. ****Backend Technologies (Optional)****

* **Node.js**: A backend environment to handle user authentication, design storage, and export functionality (e.g., converting designs into various file formats).
* **Express.js**: A web framework for Node.js to handle requests and manage APIs.
* **MongoDB/Firebase**: For cloud-based user data storage and design persistence (storing designs).
* **Cloud Storage**: Integration with cloud storage services (e.g., AWS S3, Google Cloud Storage) for file storage and management.
* **Authentication**: Use services like **JWT** (JSON Web Tokens) for secure authentication and user session management.

#### 4. ****Performance and Scalability****

* **Asynchronous Loading**: For a smooth user experience, use asynchronous loading for large files and images.
* **Real-Time Rendering**: Ensure that the application can handle real-time changes on the canvas without lag.
* **Caching**: Implement caching to minimize server load when users load and modify designs.
* **Scalability**: The app should scale to handle multiple concurrent users, especially if hosting the application for a large audience.

**2. Functional Requirements:**

#### 1. ****User Account Management****

* **Registration/Login**: Users must be able to register, log in, and manage accounts to store and retrieve their designs.
* **Password Recovery**: Ability to recover forgotten passwords.
* **Profile Management**: Users can update profile information, including username, email, and password.

#### 2. ****Design Workspace (Canvas)****

* **Canvas Area**: A central workspace for logo creation (with adjustable size, zoom-in/out options).
* **Grid and Guidelines**: Option to toggle a grid or guides for precise positioning.
* **Background Color**: Users can set the background color or make it transparent for logos.

#### 3. ****Shape and Icon Library****

* **Predefined Shapes**: Circle, square, triangle, polygons, etc., with the ability to resize, rotate, and manipulate shapes.
* **Icons Library**: Users can choose from a collection of vector-based icons for logo design (e.g., symbols, abstract images).
* **Custom Shape Drawing**: Allow users to draw custom shapes if needed.

#### 4. ****Text Editor****

* **Add Text**: Users can insert and customize text.
* **Typography Options**: Support for different fonts, text sizes, colors, spacing, and alignment.
* **Text Effects**: Ability to apply effects like shadow, outline, or gradient on text.

#### 5. ****Color Picker****

* **Basic Colors**: Allow selection from predefined color palettes.
* **Advanced Picker**: RGB/HEX input or a color wheel for more precision.
* **Gradient**: Support for linear and radial gradient fills.

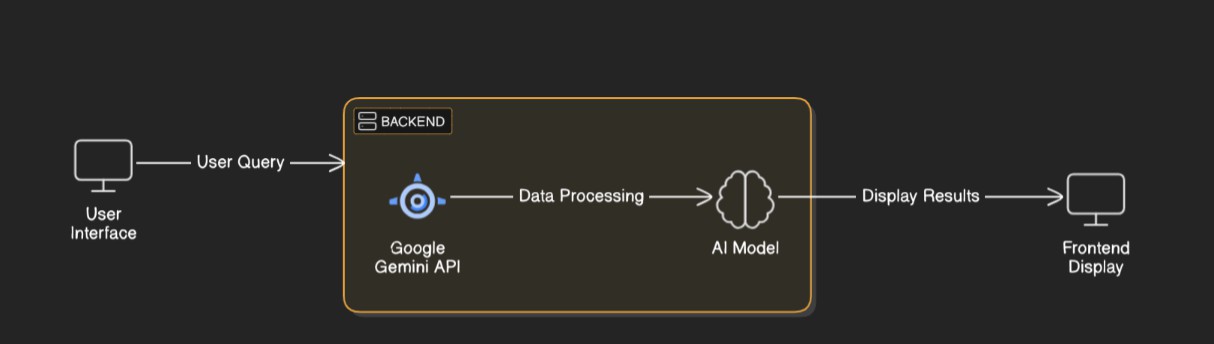
**3. Constraints &Challenges:**

* + Design and User Experience (UX) Challenges
  + Technical Challenges
  + Performance and Optimization Challenges

# Phase-3: Project Design

## Objective:

Develop the architecture and user flow of the application.



## Key Points:

1. **System Architecture:**
   * User enters logo related query via UI.
   * Query is processed using **GoogleGeminiAPI**.
   * AI model fetches and processes the data.
   * The front end displays **various logos with different designs.**
2. **User Flow:**
   * Step1:User enters a query(e.g.,"Bestmotorcyclesunder₹1lakh").
   * Step2:The back end **calls the Gemini Flash API** to generate logo.
   * Step3:The app processes the data and **displays results** in designed logo format.
3. **UI/UX Considerations:**
   * **Minimalist, user-friendly interface** for seamless designing.
   * **Filters for colors, fonts, and backgrounds**.
   * **Unique designs for** better choices of selecting.

# 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 | Sandeep | Google API Key, Python ,Streamlit setup | API connection established &working |
| Sprint 1 | Frontend UI Development | Medium | 2 hours  (Day 1) | End of Day 1 | Sravani | API response format finalized | Basic UI with input fields |
| Sprint 2 | Logo generating  using user’s input | High | 3 hours  (Day 2) | Mid-Day 2 | Sandeep and sravani | API response, UI elements ready | Design with filters |
| Sprint 2 | Error Handling & Debugging | High | 1.5 hours  (Day 2) | Mid-Day 2 | Sandeep and revathi | API logs,UI inputs | Improved API stability |
| Sprint 3 | Testing & UI Enhancements | Medium | 1.5 hours  (Day 2) | Mid-Day 2 | Sravani and saharshitha | 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:

**Sprint1– Setup & Integration (Day1):**

**(High Priority)** Setup the **environment** & install dependencies.

**(High Priority)** Integrate **Google Gemini API**.

**(Medium Priority)**Build a **basic UI with input fields**.

## Sprint2–CoreFeatures & Debugging (Day2):

**(High Priority)** Implement **search & comparison functionalities**.

**(High Priority)** Debug API issues & handle **errors in queries**.

## Sprint3–Testing, Enhancements & Submission (Day2):

**(Medium Priority)**Test API responses, refine UI, & fix UI bugs.

**(Low Priority)** Final **demo preparation & deployment**.

# Phase-5: Project Development

## Objective:

Implement core features of the Logcraft App.

## Key Points:

1. **Technology Stack Used:**
   * **Frontend:** Streamlit
   * **Backend:** Google Gemini Flash API
   * **Programming Language:** Python
2. **Development Process:**
   * Implement **API key authentication** and **Gemini API integration**.
   * Develop **different types of logos with unique designs**.
   * Optimize **search queries for performance and quality logos**.
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**.

# Code for developing a logo designing app:

# Install necessary libraries

!pip install pillow matplotlib ipywidgets

# Import required libraries

from PIL import Image, ImageDraw, ImageFont

import matplotlib.pyplot as plt

import ipywidgets as widgets

from ipywidgets import interactive

# Function to create a logo

def create\_logo(text, shape, text\_color, shape\_color, font\_size, save=False):

    # Create a blank image with white background

    width, height = 500, 500

    image = Image.new('RGB', (width, height), color='white')

    draw = ImageDraw.Draw(image)

    # Load font

    font = ImageFont.load\_default()

    # Draw shape (circle or rectangle)

    if shape == 'Circle':

        draw.ellipse([(width/4, height/4), (width\*3/4, height\*3/4)], fill=shape\_color)

    elif shape == 'Rectangle':

        draw.rectangle([(width/4, height/4), (width\*3/4, height\*3/4)], fill=shape\_color)

    # Add text in the center

    text\_width, text\_height = draw.textsize(text, font=font)

    position = ((width - text\_width) // 2, (height - text\_height) // 2)

    draw.text(position, text, fill=text\_color, font=font)

    # Display the image using matplotlib

    plt.imshow(image)

    plt.axis('off')

    plt.show()

    # Optionally save the logo to a file

    if save:

        image.save('logo.png')

        print("Logo saved as 'logo.png'")

# Create interactive widgets

text\_widget = widgets.Text(value='My Logo', description='Text:')

shape\_widget = widgets.Dropdown(

    options=['Circle', 'Rectangle'],

    value='Circle',

    description='Shape:'

)

text\_color\_widget = widgets.ColorPicker(value='black', description='Text Color:')

shape\_color\_widget = widgets.ColorPicker(value='blue', description='Shape Color:')

font\_size\_widget = widgets.IntSlider(value=30, min=10, max=100, step=5, description='Font Size:')

save\_widget = widgets.Checkbox(value=False, description="Save Logo")

# Create interactive plot

interactive\_plot = interactive(create\_logo,

                               text=text\_widget,

                               shape=shape\_widget,

                               text\_color=text\_color\_widget,

                               shape\_color=shape\_color\_widget,

                               font\_size=font\_size\_widget,

                               save=save\_widget)

# Display the widgets

interactive\_plot

# Output:

# C:\Users\Admin\Pictures\Screenshot (1).png

# Block diagram for better understanding:

# Free Block Diagram Maker - Create Block Diagram | Canva

# Phase-6: Functional & Performance Testing

## Objective:

Ensure that the Auto Logocraft app works as expected.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** | **Tester** |
| TC-001 | Functional Testing | Query" best food business logo" | Relevant logos should be displayed | Passed | Tester1 |
| TC-002 | Functional Testing | Query "change the font of then logo" | Font should be changed to new font | Passed | Tester2 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TC-003 | Performance Testing | API response time under 500ms | API should return results quickly. | Needs Optimization | Tester3 |
| TC-004 | Bug Fixes &Improvements | Fixed incorrect API responses. | Data accuracy should be improved. | Fixed | Developer |
| TC-005 | Final Validation | Ensure UI is responsive across devices. | UI should work on mobile & desktop. | Failed - UI broken on mobile | Tester2 |
| 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(3-5Minutes)**
3. **GitHub/Code Repository Link**
4. **Presentation**

### ****Conclusion:****

Developing a **logo designing app** presents an exciting opportunity to empower users to create unique, professional logos with ease, leveraging cutting-edge tools and technologies such as AI and machine learning. Throughout the development process, it is essential to balance **user-friendly interfaces**, **advanced design tools**, and **performance** to create an engaging, seamless experience for both novices and experienced designers.

Key challenges, such as **user personalization**, **AI integration**, and **real-time image processing**, must be addressed effectively to ensure the app remains functional and intuitive. Security, scalability, and **data privacy** also play crucial roles, particularly as users generate unique logos that may hold intellectual property value.

Ultimately, the success of the application hinges on:

* Building a strong foundation for **AI-powered design generation** that offers smart suggestions while allowing users full creative control.
* Ensuring the app is **accessible** and **responsive**, providing an experience that works across various platforms and devices.
* Ensuring **continuous improvement** with regular updates, user feedback, and the integration of new trends in logo design.

In a competitive market, **differentiating the app** with features such as AI-driven suggestions, advanced customization options, and robust user support can drive adoption. Moreover, focusing on **clear monetization strategies** and maintaining a strong focus on user needs and trends will ensure the app stays relevant and valuable in the long term.

By combining the power of **artificial intelligence**, **cloud infrastructure**, and **design thinking**, a well-executed logo designing app can unlock the creative potential of businesses, individuals, and organizations, helping them establish their brand identities with minimal effort and maximum impact.