# **Specification Document for Building hAIku Experience in Streamlit**

## **Project Overview**

The **hAIku Experience** application allows users to explore creativity, mindfulness, and art through haiku creation, visualization (haiga), and meditation. The project will be implemented in Streamlit, leveraging its interactive UI capabilities. The experience is organized into tabs, each representing a specific functionality.

## **Features and Tabs**

### **1. Introduction Tab**

* **Purpose**: Introduce the hAIku Experience and gather the user’s name for a personalized journey.
* **Components**:
  + Header and Introduction text.
  + Input box for the user’s name.
  + Button to proceed to the next tab.
* **Prompts**:
  + *Welcome to the hAIku Experience Project! Please enter your name to personalize the experience.*
* **Implementation Notes**:
  + Use st.text\_input() for the name input.
  + Store the user’s name using st.session\_state.
  + Use st.button() to navigate to the next tab.

### **2. Selection Tab**

* **Purpose**: Enable users to select keywords for haiku generation.
* **Components**:
  + Dropdowns or radio buttons for each keyword category:
    - Setting, Animal, Object, Emotion, Action, and Cut Word.
  + Display of selected keywords in JSON format.
* **Prompts**:
  + *Please select a word from each category to guide the haiku creation process.*
* **Implementation Notes**:
  + Use st.selectbox() or st.radio() for keyword selection.
  + Provide default selections in case the user skips input.
  + Store selections as a dictionary in st.session\_state.

### **3. Haiku Tab**

* **Purpose**: Generate a haiku based on the user’s selected keywords.
* **Components**:
  + Display the selected keywords.
  + Button to generate haiku.
  + Text display of the generated haiku.
* **Prompts**:
  + *Based on your selections, here is your haiku:*
* **Implementation Notes**:
  + Use OpenAI API (openai.ChatCompletion) for haiku generation.
  + Structure the prompt dynamically using the selected keywords.
  + Use st.text\_area() to display the haiku.

### **4. Haiga Tab**

* **Purpose**: Visualize the haiku as an artistic image.
* **Components**:
  + Display the generated haiku.
  + Button to generate a haiga image.
  + Display of the generated image.
* **Prompts**:
  + *This haiga represents your haiku:*
* **Implementation Notes**:
  + Use DALL-E API for image generation.
  + Include metadata for styling (e.g., postcard effect, red stamp, aspect ratio 16:9).
  + Use st.image() to display the generated image.

### **5. Meditation Tab**

* **Purpose**: Create a meditative experience based on the haiku.
* **Components**:
  + Display the haiku.
  + Button to start a guided meditation.
  + Text display of the meditation steps.
* **Prompts**:
  + *Let’s turn your haiku into a meditative journey:*
* **Implementation Notes**:
  + Dynamically generate meditation text based on haiku content.
  + Use st.text\_area() to display the meditation script.

### **6. How It Works Tab**

* **Purpose**: Explain the mechanics and methodology behind the application.
* **Components**:
  + Text or FAQ-style explanation.
* **Implementation Notes**:
  + Use static text sections with st.markdown() to explain:
    - The process of haiku creation.
    - AI technologies used (OpenAI, DALL-E).
    - Potential for expansion.

### **7. Contact Tab**

* **Purpose**: Provide contact details and resources.
* **Components**:
  + Display email address and website links.
* **Implementation Notes**:
  + Use st.markdown() to format contact details.

## **Libraries and Tools**

### **Backend**

* **OpenAI API**: For haiku and meditation text generation.
* **DALL-E API**: For haiga image creation.

### **Frontend (Streamlit)**

* **Streamlit Widgets**: For user inputs (st.text\_input, st.selectbox, etc.).
* **Session State**: To manage user data across tabs.
* **Markdown and Layout**: For styling (st.markdown, st.sidebar).

### **Additional Libraries**

* Pandas: If any tabular data processing is needed.
* Requests: For API calls.
* Pillow: For additional image processing, if required.

## **Application Flow**

1. **Start in Introduction Tab**:
   * User enters their name.
2. **Move to Selection Tab**:
   * User selects keywords.
3. **Generate Haiku in Haiku Tab**:
   * Display the generated haiku.
4. **Create Haiga in Haiga Tab**:
   * Generate and display a haiga image.
5. **Explore Meditation in Meditation Tab**:
   * Provide a guided meditation based on the haiku.
6. **Learn in How It Works Tab**:
   * Explain the project’s mechanics.
7. **Contact Details in Contact Tab**:
   * Provide email and website links.

## **Design Patterns**

* **Tab Navigation**: Use Streamlit's st.sidebar to organize tabs for seamless user navigation.
* **State Management**: Use st.session\_state to store user inputs and generated data.
* **Modular Prompts**: Dynamically construct prompts for API calls based on user inputs.
* **Error Handling**: Gracefully handle API errors and missing inputs using st.error().

## **Example Code Structure**

python

import streamlit as st

import openai

import requests

# Initialize Session State

if 'name' not in st.session\_state:

st.session\_state['name'] = ""

# Tab Navigation

st.sidebar.title("hAIku Experience")

tabs = ["Introduction", "Selection", "Haiku", "Haiga", "Meditation", "How It Works", "Contact"]

choice = st.sidebar.radio("Navigate", tabs)

# Introduction Tab

if choice == "Introduction":

st.title("Welcome to the hAIku Experience")

st.session\_state['name'] = st.text\_input("What is your name?", "")

if st.session\_state['name']:

st.write(f"Hello, {st.session\_state['name']}! Let's begin.")

# Repeat similar structure for other tabs

This document outlines the structure, functionalities, and technical requirements to implement the **hAIku Experience** in Streamlit. Let me know if you need further assistance or additional features!