# **WebDesign AI Generator - Simplified Project Documentation**

## **Project Overview**

WebDesign AI Generator is a web application that allows users to upload existing website designs, analyze them using AI (Gemini), and generate multiple new design variations based on the original design using Claude AI models. The application provides a straightforward interface for interacting with designs, expanding on them, and saving favorites.

## **Core Features**

1. **Design Upload**: Upload website design images
2. **AI Analysis**: Analyze designs using Gemini 2.0 Flash
3. **Design Generation**: Generate new designs based on analysis and user prompts using Claude
4. **AI Selection**: Choose different AI models for both analysis and generation phases
5. **Custom API Keys**: Use personal API keys for AI services
6. **Design Management**: View designs in detail, expand on specific designs, and save favorites

## **Tech Stack**

### **Frontend**

* **Framework**: React.js
* **Language**: JavaScript (TypeScript optional)
* **Styling**: Tailwind CSS
* **State Management**: React Context API + localStorage

## **Project Structure**

src/

├── components/

│ ├── common/

│ │ ├── Button.js

│ │ ├── LoadingSpinner.js

│ │ └── ErrorMessage.js

│ ├── layout/

│ │ ├── Header.js

│ │ └── Container.js

│ ├── upload/

│ │ └── ImageUpload.js

│ ├── prompt/

│ │ ├── PromptInput.js

│ │ └── AISelector.js

│ └── designs/

│ ├── DesignGallery.js

│ ├── DesignCard.js

│ └── DesignDetail.js

├── contexts/

│ ├── SettingsContext.js

│ ├── DesignContext.js

│ └── FavoritesContext.js

├── pages/

│ ├── Home.js

│ ├── Settings.js

│ ├── Favorites.js

│ ├── DesignDetail.js

│ └── ExpandDesign.js

├── services/

│ ├── geminiService.js

│ ├── claudeService.js

│ ├── favoriteService.js

│ └── settingsService.js

├── utils/

│ ├── localStorage.js

│ └── errorHandling.js

├── App.js

└── index.js

## **Implementation Guide**

### **1. Setup Project**

Create a new React project and install dependencies:

npx create-react-app webdesign-ai-generator

cd webdesign-ai-generator

npm install tailwindcss react-router-dom axios

Configure Tailwind CSS:

npx tailwindcss init

### **2. Core Components**

#### **2.1 ImageUpload Component**

A simple component that allows users to upload image files:

// components/upload/ImageUpload.js

import React, { useState } from 'react';

function ImageUpload({ onImageUpload }) {

const [previewUrl, setPreviewUrl] = useState(null);

const handleFileChange = (e) => {

const file = e.target.files[0];

if (file && file.type.startsWith('image/')) {

const reader = new FileReader();

reader.onload = () => {

setPreviewUrl(reader.result);

onImageUpload(file);

};

reader.readAsDataURL(file);

}

};

const handleDrop = (e) => {

e.preventDefault();

const file = e.dataTransfer.files[0];

if (file && file.type.startsWith('image/')) {

const reader = new FileReader();

reader.onload = () => {

setPreviewUrl(reader.result);

onImageUpload(file);

};

reader.readAsDataURL(file);

}

};

return (

<div

className="border-2 border-dashed rounded-lg p-6 text-center"

onDrop={handleDrop}

onDragOver={(e) => e.preventDefault()}

>

{!previewUrl ? (

<>

<input

type="file"

id="image-upload"

className="hidden"

accept="image/\*"

onChange={handleFileChange}

/>

<label

htmlFor="image-upload"

className="cursor-pointer text-blue-500 hover:text-blue-700"

>

Click to upload image or drag and drop

</label>

<p className="text-sm text-gray-500 mt-1">

Supported formats: PNG, JPG, JPEG, GIF, WEBP

</p>

</>

) : (

<div className="mt-4">

<img

src={previewUrl}

alt="Preview"

className="max-h-64 mx-auto rounded"

/>

<button

onClick={() => {

setPreviewUrl(null);

onImageUpload(null);

}}

className="mt-2 text-red-500 hover:text-red-700"

>

Remove Image

</button>

</div>

)}

</div>

);

}

export default ImageUpload;

#### **2.2 PromptInput Component**

A simple text input for entering design prompts:

// components/prompt/PromptInput.js

import React from 'react';

function PromptInput({ value, onChange, placeholder }) {

return (

<div className="w-full">

<textarea

value={value}

onChange={(e) => onChange(e.target.value)}

placeholder={placeholder || "Enter additional instructions..."}

className="w-full p-3 border border-gray-300 rounded-md"

rows={4}

/>

</div>

);

}

export default PromptInput;

#### **2.3 AISelector Component**

A dropdown to select AI models:

// components/prompt/AISelector.js

import React from 'react';

function AISelector({ models, selectedModel, onSelect }) {

return (

<div className="mb-4">

<label className="block text-sm font-medium text-gray-700 mb-1">

Select AI Model

</label>

<select

value={selectedModel}

onChange={(e) => onSelect(e.target.value)}

className="mt-1 block w-full py-2 px-3 border border-gray-300 rounded-md shadow-sm focus:outline-none focus:ring-blue-500 focus:border-blue-500"

>

{models.map((model) => (

<option key={model.id} value={model.id}>

{model.name}

</option>

))}

</select>

</div>

);

}

export default AISelector;

#### **2.4 DesignCard Component**

Displays a single design with action buttons:

// components/designs/DesignCard.js

import React from 'react';

import { useNavigate } from 'react-router-dom';

function DesignCard({ design, onSave, isSaved }) {

const navigate = useNavigate();

return (

<div className="border rounded-lg overflow-hidden shadow-sm">

<div

className="h-48 bg-gray-100 cursor-pointer"

onClick={() => navigate(`/design/${design.id}`)}

>

<iframe

title={`Design ${design.id}`}

srcDoc={`

<html>

<head>

<style>${design.cssContent}</style>

</head>

<body>${design.htmlContent}</body>

</html>

`}

className="w-full h-full border-0"

/>

</div>

<div className="p-3 flex justify-between">

<button

onClick={() => navigate(`/expand/${design.id}`)}

className="px-3 py-1 bg-blue-500 text-white rounded hover:bg-blue-600"

>

Expand

</button>

<button

onClick={() => onSave(design)}

className={`px-3 py-1 rounded ${

isSaved

? 'bg-gray-200 text-gray-800'

: 'bg-gray-800 text-white hover:bg-gray-700'

}`}

>

{isSaved ? 'Saved' : 'Save'}

</button>

</div>

</div>

);

}

export default DesignCard;

### **3. Context Setup**

#### **3.1 Settings Context**

Manages user preferences and API keys:

// contexts/SettingsContext.js

import React, { createContext, useState, useEffect } from 'react';

const SettingsContext = createContext();

const defaultSettings = {

apiKeys: {

claude: '',

gemini: ''

},

defaultModels: {

analysis: 'gemini-2.0-flash',

generation: 'claude-3-7-sonnet-20250219'

}

};

function SettingsProvider({ children }) {

const [settings, setSettings] = useState(defaultSettings);

// Load settings from localStorage on initial render

useEffect(() => {

const storedSettings = localStorage.getItem('settings');

if (storedSettings) {

try {

setSettings(JSON.parse(storedSettings));

} catch (error) {

console.error('Error loading settings:', error);

}

}

}, []);

// Update localStorage when settings change

useEffect(() => {

localStorage.setItem('settings', JSON.stringify(settings));

}, [settings]);

const updateApiKey = (service, key) => {

setSettings(prev => ({

...prev,

apiKeys: {

...prev.apiKeys,

[service]: key

}

}));

};

const updateDefaultModel = (phase, model) => {

setSettings(prev => ({

...prev,

defaultModels: {

...prev.defaultModels,

[phase]: model

}

}));

};

return (

<SettingsContext.Provider value={{

settings,

updateApiKey,

updateDefaultModel

}}>

{children}

</SettingsContext.Provider>

);

}

function useSettings() {

const context = React.useContext(SettingsContext);

if (!context) {

throw new Error('useSettings must be used within a SettingsProvider');

}

return context;

}

export { SettingsProvider, useSettings };

#### **3.2 Design Context**

Manages the current design session:

// contexts/DesignContext.js

import React, { createContext, useState, useContext } from 'react';

import { analyzeDesign } from '../services/geminiService';

import { generateDesigns } from '../services/claudeService';

import { useSettings } from './SettingsContext';

import { v4 as uuidv4 } from 'uuid';

const DesignContext = createContext();

function DesignProvider({ children }) {

const { settings } = useSettings();

const [originalImage, setOriginalImage] = useState(null);

const [analysisResult, setAnalysisResult] = useState(null);

const [generatedDesigns, setGeneratedDesigns] = useState([]);

const [currentPrompt, setCurrentPrompt] = useState('');

const [isAnalyzing, setIsAnalyzing] = useState(false);

const [isGenerating, setIsGenerating] = useState(false);

const [selectedModels, setSelectedModels] = useState({

analysis: settings.defaultModels.analysis,

generation: settings.defaultModels.generation

});

const analyzeImage = async () => {

if (!originalImage) return;

setIsAnalyzing(true);

try {

const reader = new FileReader();

reader.readAsDataURL(originalImage);

reader.onload = async () => {

const imageData = reader.result.split(',')[1]; // Remove data URL prefix

const result = await analyzeDesign(

imageData,

selectedModels.analysis,

settings.apiKeys.gemini

);

setAnalysisResult(result);

};

} catch (error) {

console.error('Error analyzing image:', error);

alert('Failed to analyze image. Please try again.');

} finally {

setIsAnalyzing(false);

}

};

const generateDesigns = async () => {

if (!analysisResult) return;

setIsGenerating(true);

try {

const designs = await generateDesigns(

analysisResult,

currentPrompt,

6, // Number of designs to generate

selectedModels.generation,

settings.apiKeys.claude

);

setGeneratedDesigns(designs);

} catch (error) {

console.error('Error generating designs:', error);

alert('Failed to generate designs. Please try again.');

} finally {

setIsGenerating(false);

}

};

const expandDesign = async (designId, additionalPrompt) => {

setIsGenerating(true);

try {

const design = generatedDesigns.find(d => d.id === designId);

if (!design) throw new Error('Design not found');

// Create a combined prompt

const combinedPrompt = `${design.prompt || ''} ${additionalPrompt}`.trim();

// Generate a new design based on the selected design

const newDesigns = await generateDesigns(

analysisResult,

combinedPrompt,

1, // Just generate one design

selectedModels.generation,

settings.apiKeys.claude

);

if (newDesigns && newDesigns.length > 0) {

const expandedDesign = {

...newDesigns[0],

parentDesignId: designId

};

setGeneratedDesigns([

expandedDesign,

...generatedDesigns

]);

return expandedDesign;

}

} catch (error) {

console.error('Error expanding design:', error);

alert('Failed to expand design. Please try again.');

} finally {

setIsGenerating(false);

}

};

const selectModel = (phase, modelId) => {

setSelectedModels(prev => ({

...prev,

[phase]: modelId

}));

};

const resetSession = () => {

setOriginalImage(null);

setAnalysisResult(null);

setGeneratedDesigns([]);

setCurrentPrompt('');

};

return (

<DesignContext.Provider value={{

originalImage,

analysisResult,

generatedDesigns,

currentPrompt,

isAnalyzing,

isGenerating,

selectedModels,

setOriginalImage,

setCurrentPrompt,

analyzeImage,

generateDesigns,

expandDesign,

resetSession,

selectModel

}}>

{children}

</DesignContext.Provider>

);

}

function useDesign() {

const context = React.useContext(DesignContext);

if (!context) {

throw new Error('useDesign must be used within a DesignProvider');

}

return context;

}

export { DesignProvider, useDesign };

#### **3.3 Favorites Context**

Manages saved designs:

// contexts/FavoritesContext.js

import React, { createContext, useState, useEffect } from 'react';

const FavoritesContext = createContext();

function FavoritesProvider({ children }) {

const [favorites, setFavorites] = useState([]);

// Load favorites from localStorage on initial render

useEffect(() => {

const storedFavorites = localStorage.getItem('favorites');

if (storedFavorites) {

try {

setFavorites(JSON.parse(storedFavorites));

} catch (error) {

console.error('Error loading favorites:', error);

}

}

}, []);

// Update localStorage when favorites change

useEffect(() => {

localStorage.setItem('favorites', JSON.stringify(favorites));

}, [favorites]);

const addFavorite = (design) => {

setFavorites(prev => {

// Don't add if already in favorites

if (prev.some(fav => fav.id === design.id)) {

return prev;

}

return [...prev, design];

});

};

const removeFavorite = (designId) => {

setFavorites(prev => prev.filter(design => design.id !== designId));

};

const isFavorite = (designId) => {

return favorites.some(design => design.id === designId);

};

return (

<FavoritesContext.Provider value={{

favorites,

addFavorite,

removeFavorite,

isFavorite

}}>

{children}

</FavoritesContext.Provider>

);

}

function useFavorites() {

const context = React.useContext(FavoritesContext);

if (!context) {

throw new Error('useFavorites must be used within a FavoritesProvider');

}

return context;

}

export { FavoritesProvider, useFavorites };

### **4. Services**

#### **4.1 Gemini Service**

Handles communication with the Gemini API:

// services/geminiService.js

import axios from 'axios';

export const analyzeDesign = async (imageData, model = 'gemini-2.0-flash', apiKey = '') => {

try {

const headers = {

'Content-Type': 'application/json'

};

if (apiKey) {

headers['x-api-key'] = apiKey;

}

const response = await axios.post(

'https://api.google.com/v1/models/gemini-pro-vision:generateContent',

{

model,

contents: [

{

parts: [

{

text: "Analyze this website design. Identify layout structure, color palette, typography, and key design elements. Return the analysis in a simple format."

},

{

inline\_data: {

mime\_type: "image/jpeg",

data: imageData

}

}

]

}

]

},

{ headers }

);

// Extract the analysis text from the response

const analysisText = response.data.candidates[0].content.parts[0].text;

// Return a structured analysis result

return {

id: Date.now().toString(),

originalImageData: imageData,

rawAnalysis: analysisText,

model,

createdAt: Date.now()

};

} catch (error) {

console.error('Error analyzing design:', error);

throw new Error('Failed to analyze design');

}

};

export const geminiModels = [

{ id: 'gemini-2.0-flash', name: 'Gemini 2.0 Flash', description: 'Fast analysis with high accuracy' },

{ id: 'gemini-2.0-pro', name: 'Gemini 2.0 Pro', description: 'Best for detailed analysis' }

];

#### **4.2 Claude Service**

Handles communication with the Claude API:

// services/claudeService.js

import axios from 'axios';

import { v4 as uuidv4 } from 'uuid';

export const generateDesigns = async (

analysisResult,

userPrompt = '',

count = 6,

model = 'claude-3-7-sonnet-20250219',

apiKey = ''

) => {

try {

const headers = {

'Content-Type': 'application/json',

'anthropic-version': '2023-06-01'

};

if (apiKey) {

headers['x-api-key'] = apiKey;

}

const response = await axios.post(

'https://api.anthropic.com/v1/messages',

{

model,

max\_tokens: 4000,

messages: [

{

role: "user",

content: `Based on this website design analysis: ${analysisResult.rawAnalysis}

${userPrompt ? `And considering this additional request: ${userPrompt}` : ''}

Generate ${count} variations of HTML/CSS code for new website designs that maintain the core aesthetic but offer creative alternatives.

For each design, provide:

1. The complete HTML code

2. The complete CSS code

3. A brief description of the design

Format each design with clear separation between the HTML and CSS.`

}

]

},

{ headers }

);

// Extract the design data from the response

const responseText = response.data.content[0].text;

// Parse the designs from the response text

// This is a simplified approach - in a real app, you'd want more robust parsing

const designs = extractDesignsFromText(responseText);

// Add metadata to each design

return designs.map(design => ({

...design,

id: uuidv4(),

createdAt: Date.now(),

prompt: userPrompt,

analysisModel: analysisResult.model,

generationModel: model

}));

} catch (error) {

console.error('Error generating designs:', error);

throw new Error('Failed to generate designs');

}

};

// Helper function to extract designs from Claude's response

const extractDesignsFromText = (text) => {

const designs = [];

// Split the text by common section dividers

const sections = text.split(/Design \d+:|Variation \d+:|Example \d+:/);

// Skip the first section if it's just introduction text

const designSections = sections.slice(1);

designSections.forEach(section => {

let htmlContent = '';

let cssContent = '';

// Look for HTML blocks

const htmlMatch = section.match(/<html[\s\S]\*?<\/html>/) ||

section.match(/<body[\s\S]\*?<\/body>/) ||

section.match(/<div[\s\S]\*?<\/div>/);

if (htmlMatch) {

htmlContent = htmlMatch[0];

}

// Look for CSS blocks

const cssMatch = section.match(/<style[\s\S]\*?<\/style>/);

if (cssMatch) {

cssContent = cssMatch[0].replace(/<\/?style>/g, '');

} else {

// Try to find CSS in a code block

const cssBlockMatch = section.match(/```css\s\*([\s\S]\*?)\s\*```/);

if (cssBlockMatch) {

cssContent = cssBlockMatch[1];

}

}

// Only add if we found both HTML and CSS

if (htmlContent && cssContent) {

designs.push({

htmlContent,

cssContent

});

}

});

return designs;

};

export const claudeModels = [

{ id: 'claude-3-7-sonnet-20250219', name: 'Claude 3.7 Sonnet', description: 'Best for creative design generation' },

{ id: 'claude-3-5-sonnet-20240620', name: 'Claude 3.5 Sonnet', description: 'Good balance of creativity and speed' }

];

### **5. Main Pages**

#### **5.1 Home Page**

// pages/Home.js

import React from 'react';

import ImageUpload from '../components/upload/ImageUpload';

import PromptInput from '../components/prompt/PromptInput';

import AISelector from '../components/prompt/AISelector';

import DesignGallery from '../components/designs/DesignGallery';

import { useDesign } from '../contexts/DesignContext';

import { geminiModels } from '../services/geminiService';

import { claudeModels } from '../services/claudeService';

function Home() {

const {

originalImage,

setOriginalImage,

analysisResult,

currentPrompt,

setCurrentPrompt,

isAnalyzing,

isGenerating,

selectedModels,

selectModel,

analyzeImage,

generateDesigns,

generatedDesigns

} = useDesign();

return (

<div className="max-w-4xl mx-auto">

<section className="mb-8">

<h2 className="text-xl font-semibold mb-4">Upload Website Design</h2>

<ImageUpload onImageUpload={setOriginalImage} />

</section>

{originalImage && (

<section className="mb-8">

<div className="flex justify-between items-center mb-4">

<h2 className="text-xl font-semibold">Analyze Design</h2>

<AISelector

models={geminiModels}

selectedModel={selectedModels.analysis}

onSelect={(model) => selectModel('analysis', model)}

/>

</div>

<button

onClick={analyzeImage}

disabled={isAnalyzing}

className="w-full py-2 px-4 bg-blue-500 text-white rounded disabled:bg-blue-300"

>

{isAnalyzing ? 'Analyzing...' : 'Analyze Design'}

</button>

</section>

)}

{analysisResult && (

<section className="mb-8">

<div className="flex justify-between items-center mb-4">

<h2 className="text-xl font-semibold">Generate Variations</h2>

<AISelector

models={claudeModels}

selectedModel={selectedModels.generation}

onSelect={(model) => selectModel('generation', model)}

/>

</div>

<div className="mb-4">

<PromptInput

value={currentPrompt}

onChange={setCurrentPrompt}

placeholder="Describe how you want to modify the design (e.g., 'Make it more minimal' or 'Use warmer colors')..."

/>

</div>

<button

onClick={generateDesigns}

disabled={isGenerating}

className="w-full py-2 px-4 bg-blue-500 text-white rounded disabled:bg-blue-300"

>

{isGenerating ? 'Generating...' : 'Generate Design Variations'}

</button>

</section>

)}

{generatedDesigns.length > 0 && (

<section>

<h2 className="text-xl font-semibold mb-4">Generated Designs</h2>

<DesignGallery />

</section>

)}

</div>

);

}

export default Home;

#### **5.2 Design Detail Page**

// pages/DesignDetail.js

import React, { useState, useEffect } from 'react';

import { useParams, useNavigate } from 'react-router-dom';

import { useDesign } from '../contexts/DesignContext';

import { useFavorites } from '../contexts/FavoritesContext';

function DesignDetail() {

const { designId } = useParams();

const navigate = useNavigate();

const { generatedDesigns } = useDesign();

const { favorites, addFavorite, removeFavorite, isFavorite } = useFavorites();

const [design, setDesign] = useState(null);

const [activeTab, setActiveTab] = useState('preview');

useEffect(() => {

// Find design in generated designs or favorites

let foundDesign = generatedDesigns.find(d => d.id === designId);

if (!foundDesign) {

foundDesign = favorites.find(d => d.id === designId);

}

if (foundDesign) {

setDesign(foundDesign);

} else {

navigate('/');

}

}, [designId, generatedDesigns, favorites, navigate]);

const handleToggleFavorite = () => {

if (isFavorite(design.id)) {

removeFavorite(design.id);

} else {

addFavorite(design);

}

};

const handleDownload = () => {

// Create HTML file content

const htmlContent = `

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Design ${design.id}</title>

<style>

${design.cssContent}

</style>

</head>

<body>

${design.htmlContent}

</body>

</html>

`;

// Create download link

const blob = new Blob([htmlContent], { type: 'text/html' });

const url = URL.createObjectURL(blob);

const a = document.createElement('a');

a.href = url;

a.download = `design-${design.id}.html`;

document.body.appendChild(a);

a.click();

document.body.removeChild(a);

URL.revokeObjectURL(url);

};

if (!design) {

return <div className="text-center py-10">Loading...</div>;

}

return (

<div className="max-w-4xl mx-auto p-4">

<div className="flex justify-between items-center mb-4">

<h1 className="text-2xl font-semibold">Design Detail</h1>

<div className="flex space-x-2">

<button

onClick={() => navigate(`/expand/${design.id}`)}

className="px-4 py-2 bg-blue-500 text-white rounded"

>

Expand on Design

</button>

<button

onClick={handleToggleFavorite}

className={`px-4 py-2 rounded ${

isFavorite(design.id)

? 'bg-gray-200 text-gray-800'

: 'bg-gray-800 text-white'

}`}

>

{isFavorite(design.id) ? 'Saved' : 'Save'}

</button>

<button

onClick={handleDownload}

className="px-4 py-2 border border-gray-300 rounded"

>

Download

</button>

</div>

</div>

<div className="mb-4">

<div className="flex border-b">

<button

className={`py-2 px-4 ${activeTab === 'preview' ? 'border-b-2 border-blue-500 font-medium' : 'text-gray-500'}`}

onClick={() => setActiveTab('preview')}

>

Preview

</button>

<button

className={`py-2 px-4 ${activeTab === 'html' ? 'border-b-2 border-blue-500 font-medium' : 'text-gray-500'}`}

onClick={() => setActiveTab('html')}

>

HTML

</button>

<button

className={`py-2 px-4 ${activeTab === 'css' ? 'border-b-2 border-blue-500 font-medium' : 'text-gray-500'}`}

onClick={() => setActiveTab('css')}

>

CSS

</button>

</div>

</div>

<div className="mb-6">

{activeTab === 'preview' && (

<div className="border rounded-lg overflow-hidden h-96">

<iframe

title={`Design ${design.id}`}

srcDoc={`

<html>

<head>

<style>${design.cssContent}</style>

</head>

<body>${design.htmlContent}</body>

</html>

`}

className="w-full h-full border-0"

/>

</div>

)}

{activeTab === 'html' && (

<pre className="bg-gray-100 p-4 rounded-lg overflow-auto h-96">

<code>{design.htmlContent}</code>

</pre>

)}

{activeTab === 'css' && (

<pre className="bg-gray-100 p-4 rounded-lg overflow-auto h-96">

<code>{design.cssContent}</code>

</pre>

)}

</div>

<div>

<h2 className="text-lg font-medium mb-2">Design Details</h2>

<div className="bg-gray-100 p-4 rounded-lg">

<p><strong>Prompt:</strong> {design.prompt || 'No prompt provided'}</p>

<p><strong>Analysis Model:</strong> {design.analysisModel}</p>

<p><strong>Generation Model:</strong> {design.generationModel}</p>

<p><strong>Created:</strong> {new Date(design.createdAt).toLocaleString()}</p>

</div>

</div>

</div>

);

}

export default DesignDetail;

#### **5.3 Expand Design Page**

// pages/ExpandDesign.js

import React, { useState, useEffect } from 'react';

import { useParams, useNavigate } from 'react-router-dom';

import { useDesign } from '../contexts/DesignContext';

import PromptInput from '../components/prompt/PromptInput';

function ExpandDesign() {

const { designId } = useParams();

const navigate = useNavigate();

const [design, setDesign] = useState(null);

const [additionalPrompt, setAdditionalPrompt] = useState('');

const { generatedDesigns, expandDesign, isGenerating } = useDesign();

useEffect(() => {

const foundDesign = generatedDesigns.find(d => d.id === designId);

if (foundDesign) {

setDesign(foundDesign);

} else {

navigate('/');

}

}, [designId, generatedDesigns, navigate]);

const handleSubmit = async (e) => {

e.preventDefault();

if (!additionalPrompt.trim()) return;

try {

await expandDesign(designId, additionalPrompt);

navigate('/');

} catch (error) {

console.error('Error expanding design:', error);

alert('Failed to expand design');

}

};

if (!design) {

return <div className="text-center py-10">Loading...</div>;

}

return (

<div className="max-w-4xl mx-auto p-4">

<h1 className="text-2xl font-semibold mb-4">Expand on Design</h1>

<div className="mb-6 border rounded-lg overflow-hidden">

<iframe

title={`Design ${design.id}`}

srcDoc={`

<html>

<head>

<style>${design.cssContent}</style>

</head>

<body>${design.htmlContent}</body>

</html>

`}

className="w-full h-64 border-0"

/>

</div>

<div className="mb-6">

<h2 className="text-lg font-medium mb-2">Original Prompt</h2>

<p className="bg-gray-100 p-3 rounded">{design.prompt || 'No prompt provided'}</p>

</div>

<form onSubmit={handleSubmit}>

<div className="mb-4">

<h2 className="text-lg font-medium mb-2">Additional Directions</h2>

<PromptInput

value={additionalPrompt}

onChange={setAdditionalPrompt}

placeholder="Add more specific directions for how to expand this design..."

/>

</div>

<div className="flex justify-end space-x-2">

<button

type="button"

onClick={() => navigate('/')}

className="px-4 py-2 border border-gray-300 rounded"

>

Cancel

</button>

<button

type="submit"

disabled={isGenerating || !additionalPrompt.trim()}

className="px-4 py-2 bg-blue-500 text-white rounded disabled:bg-blue-300"

>

{isGenerating ? 'Generating...' : 'Expand Design'}

</button>

</div>

</form>

</div>

);

}

export default ExpandDesign;

#### **5.4 Settings Page**

// pages/Settings.js

import React from 'react';

import { useSettings } from '../contexts/SettingsContext';

import { geminiModels } from '../services/geminiService';

import { claudeModels } from '../services/claudeService';

function Settings() {

const { settings, updateApiKey, updateDefaultModel } = useSettings();

return (

<div className="max-w-2xl mx-auto p-4">

<h1 className="text-2xl font-semibold mb-6">Settings</h1>

<div className="mb-8">

<h2 className="text-xl font-medium mb-4">API Keys</h2>

<p className="mb-4 text-gray-600">

Enter your own API keys to use with the services. Your keys are stored locally in your browser.

</p>

<div className="mb-4">

<label className="block text-sm font-medium text-gray-700 mb-1" htmlFor="claude-key">

Claude API Key

</label>

<input

id="claude-key"

type="password"

value={settings.apiKeys.claude}

onChange={(e) => updateApiKey('claude', e.target.value)}

className="w-full p-2 border border-gray-300 rounded"

placeholder="Enter your Claude API key"

/>

</div>

<div>

<label className="block text-sm font-medium text-gray-700 mb-1" htmlFor="gemini-key">

Gemini API Key

</label>

<input

id="gemini-key"

type="password"

value={settings.apiKeys.gemini}

onChange={(e) => updateApiKey('gemini', e.target.value)}

className="w-full p-2 border border-gray-300 rounded"

placeholder="Enter your Gemini API key"

/>

</div>

</div>

<div>

<h2 className="text-xl font-medium mb-4">Default Models</h2>

<div className="mb-4">

<label className="block text-sm font-medium text-gray-700 mb-1">

Default Analysis Model

</label>

<select

value={settings.defaultModels.analysis}

onChange={(e) => updateDefaultModel('analysis', e.target.value)}

className="w-full p-2 border border-gray-300 rounded"

>

{geminiModels.map((model) => (

<option key={model.id} value={model.id}>

{model.name} - {model.description}

</option>

))}

</select>

</div>

<div>

<label className="block text-sm font-medium text-gray-700 mb-1">

Default Generation Model

</label>

<select

value={settings.defaultModels.generation}

onChange={(e) => updateDefaultModel('generation', e.target.value)}

className="w-full p-2 border border-gray-300 rounded"

>

{claudeModels.map((model) => (

<option key={model.id} value={model.id}>

{model.name} - {model.description}

</option>

))}

</select>

</div>

</div>

</div>

);

}

export default Settings;

#### **5.5 Favorites Page**

// pages/Favorites.js

import React from 'react';

import { useFavorites } from '../contexts/FavoritesContext';

import DesignCard from '../components/designs/DesignCard';

function Favorites() {

const { favorites, addFavorite, removeFavorite, isFavorite } = useFavorites();

if (favorites.length === 0) {

return (

<div className="max-w-4xl mx-auto p-4">

<h1 className="text-2xl font-semibold mb-6">Favorites</h1>

<div className="text-center py-10 bg-gray-50 rounded-lg">

<p className="text-gray-500">You haven't saved any designs yet.</p>

</div>

</div>

);

}

return (

<div className="max-w-4xl mx-auto p-4">

<h1 className="text-2xl font-semibold mb-6">Favorites</h1>

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">

{favorites.map(design => (

<DesignCard

key={design.id}

design={design}

onSave={() => removeFavorite(design.id)}

isSaved={true}

/>

))}

</div>

</div>

);

}

export default Favorites;

### **6. App Component**

Set up routing and providers:

// App.js

import React from 'react';

import { BrowserRouter, Routes, Route } from 'react-router-dom';

import Header from './components/layout/Header';

import Home from './pages/Home';

import Settings from './pages/Settings';

import Favorites from './pages/Favorites';

import DesignDetail from './pages/DesignDetail';

import ExpandDesign from './pages/ExpandDesign';

import { SettingsProvider } from './contexts/SettingsContext';

import { DesignProvider } from './contexts/DesignContext';

import { FavoritesProvider } from './contexts/FavoritesContext';

function App() {

return (

<BrowserRouter>

<SettingsProvider>

<DesignProvider>

<FavoritesProvider>

<div className="min-h-screen bg-gray-50">

<Header />

<main className="container mx-auto py-6 px-4">

<Routes>

<Route path="/" element={<Home />} />

<Route path="/settings" element={<Settings />} />

<Route path="/favorites" element={<Favorites />} />

<Route path="/design/:designId" element={<DesignDetail />} />

<Route path="/expand/:designId" element={<ExpandDesign />} />

</Routes>

</main>

</div>

</FavoritesProvider>

</DesignProvider>

</SettingsProvider>

</BrowserRouter>

);

}

export default App;

### **7. Layout Components**

#### **7.1 Header Component**

// components/layout/Header.js

import React from 'react';

import { Link, useLocation } from 'react-router-dom';

function Header() {

const location = useLocation();

const isActive = (path) => {

return location.pathname === path ? 'text-blue-600 border-blue-600' : 'text-gray-500 border-transparent hover:text-gray-700';

};

return (

<header className="bg-white shadow-sm">

<div className="container mx-auto px-4">

<div className="flex justify-between items-center h-16">

<Link to="/" className="flex-shrink-0">

<h1 className="text-xl font-bold text-gray-800">WebDesign AI Generator</h1>

</Link>

<nav className="flex space-x-8">

<Link

to="/"

className={`inline-flex items-center px-1 pt-1 border-b-2 text-sm font-medium ${isActive('/')}`}

>

Home

</Link>

<Link

to="/favorites"

className={`inline-flex items-center px-1 pt-1 border-b-2 text-sm font-medium ${isActive('/favorites')}`}

>

Favorites

</Link>

<Link

to="/settings"

className={`inline-flex items-center px-1 pt-1 border-b-2 text-sm font-medium ${isActive('/settings')}`}

>

Settings

</Link>

</nav>

</div>

</div>

</header>

);

}

export default Header;

### **8. Common Components**

#### **8.1 LoadingSpinner Component**

// components/common/LoadingSpinner.js

import React from 'react';

function LoadingSpinner() {

return (

<div className="flex justify-center items-center">

<div className="animate-spin rounded-full h-8 w-8 border-b-2 border-blue-500"></div>

</div>

);

}

export default LoadingSpinner;

#### **8.2 ErrorMessage Component**

// components/common/ErrorMessage.js

import React from 'react';

function ErrorMessage({ message }) {

return (

<div className="bg-red-50 border border-red-200 text-red-700 px-4 py-3 rounded relative">

<strong className="font-bold">Error:</strong>

<span className="block sm:inline"> {message || 'Something went wrong'}</span>

</div>

);

}

export default ErrorMessage;

## **Deployment**

### **1. Build the App**

npm run build

### **2. Deployment Options**

1. **Vercel** - Easy one-click deployment
2. **Netlify** - Simple deployment with drag-and-drop or Git integration
3. **GitHub Pages** - Free for public repositories

### **3. Environment Variables**

Set up environment variables for your production deployment:

* REACT\_APP\_DEFAULT\_GEMINI\_API\_ENDPOINT
* REACT\_APP\_DEFAULT\_CLAUDE\_API\_ENDPOINT

## **Error Handling Strategy**

1. Use try/catch blocks for API calls
2. Display user-friendly error messages
3. Log detailed errors to console for debugging
4. Implement loading states for all async operations

## **Testing**

Keep testing simple:

1. Manual testing of each component
2. Test in different browsers (Chrome, Firefox, Safari)
3. Test on different devices (desktop, tablet, mobile)

## **Conclusion**

This simplified approach to the WebDesign AI Generator focuses on:

1. **Core functionality** without overengineering
2. **Simple state management** using React Context
3. **Direct localStorage** for persistence without complex data layers
4. **Minimal dependencies** to reduce potential errors
5. **Straightforward component structure** with clear responsibilities

By following this documentation, you and Cursor AI can build a functional, maintainable application that meets all the requirements without unnecessary complexity. The simplified approach makes the code more approachable and reduces potential points of failure, while still delivering all the core features needed.