



SMART DESIGN SCORE CHECKER

RAHUI DASARI

dasarirahulpatel.drp@gmail.com



1. INTRODUCTION

1.1 Project Overview

The "Smart Design Score Checker" is an AI-powered tool integrated with Adobe Creative Cloud that evaluates digital designs on key metrics like aesthetics, usability, accessibility, and adherence to design principles. By providing real-time feedback and improvement suggestions, this tool helps designers ensure their work meets professional standards and appeals to their target audience.



1.2 Purpose

The purpose of this project is to assist designers in creating high-quality, impactful designs by offering them an automated scoring system that evaluates their work against industry standards and trends. This tool is designed to enhance the design process, reduce revision cycles, and foster continuous learning among designers.

1.3 Scope

The Smart Design Score Checker targets professional designers, students, and design teams working on various projects, including UI/UX design, branding, marketing, and content creation. The tool integrates seamlessly with Adobe Creative Cloud applications, providing real-time, in-app feedback and scoring.

2. SYSTEM REQUIREMENTS

2.1 Hardware Requirements

- **Processor:** Minimum: Intel Core i5 or equivalent; Recommended: Intel Core i7 or equivalent
- **RAM:** Minimum: 8 GB; Recommended: 16 GB or higher
- **Storage:** Minimum: 5 GB of free space for installation; Recommended: SSD with 10 GB of free space
- **Graphics:** Minimum: Integrated GPU; Recommended: Dedicated GPU with 4 GB VRAM (NVIDIA/AMD)

2.2 Software Requirements

- **Operating System:** Windows 10 or later, macOS 10.14 (Mojave) or later
- **Adobe Creative Cloud:** Latest versions of Photoshop, Illustrator, XD, and other relevant Adobe applications
- **Internet Connection:** Required for AI model updates, trend analysis, and cloud-based processing
- **Programming Languages:** Python, JavaScript, or C++ (for development purposes)

2.3 Dependencies

- **AI Libraries:** TensorFlow, PyTorch for AI model implementation
- **Adobe APIs:** Adobe Creative Cloud SDKs for integration
- **Accessibility Tools:** WCAG 2.1 standards integration for accessibility checking
- **Database:** SQL/NoSQL for storing user preferences, scoring metrics, and version history

3. SYSTEM ARCHITECTURE

3.1 Overview

The Smart Design Score Checker is built on a modular architecture that allows seamless integration with Adobe Creative Cloud applications. The system consists of several key components:

1. **User Interface (UI):** A plugin interface integrated within Adobe Creative Cloud apps.
2. **AI Analysis Engine:** The core module that performs design analysis using pre-trained AI models.
3. **Feedback Module:** Generates real-time feedback and improvement suggestions.
4. **Scoring Module:** Computes the design score based on aesthetic, usability, and accessibility metrics.
5. **Trend and Benchmark Database:** Stores industry benchmarks and trend data for comparison.
6. **Collaboration and Sharing Module:** Allows team collaboration and sharing of design scores and feedback.

3.2 Data Flow Diagram

```
[Adobe Creative Cloud UI] --> [User Inputs Design]
|
v
[AI Analysis Engine] --> [Scoring Module] --> [Feedback Module]
|
v
[Trend & Benchmark DB] <--> [Scoring Module] <--> [User]
```

4. FEATURES AND FUNCTIONALITIES

4.1 AI-Powered Design Evaluation

- **Aesthetic Analysis:**
 - Evaluates design elements such as color harmony, typography, balance, and contrast.
 - Provides a score based on adherence to design principles and contemporary trends.
- **Usability Assessment:**
 - Assesses readability, navigation, and interaction elements in UI/UX designs.
 - Scores the design on user-friendliness and intuitive use.
- **Accessibility Checker:**
 - Checks for compliance with WCAG 2.1 standards.
 - Evaluates color contrast, font size, and alt text usage.

4.2 Real-Time Feedback and Suggestions

- **Live Design Scoring:**
 - Provides a real-time design score that updates as changes are made in Adobe apps.
- **Improvement Tips:**
 - Suggests actionable improvements, such as adjusting color schemes or typography.
 - Offers advice on rebalancing design elements for a higher score.

4.3 Benchmarking and Comparisons

- **Industry Benchmarks:**
 - Compares the design score against industry standards and similar projects.

- **Peer Comparisons:**

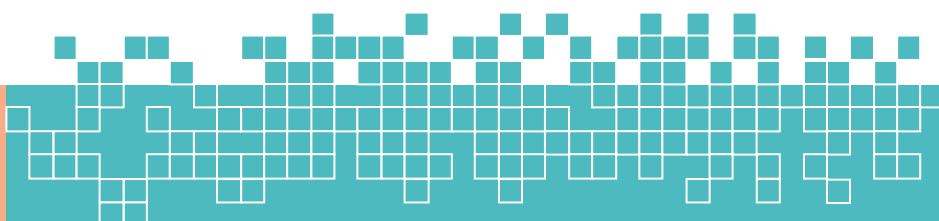
- Allows designers to compare their scores with others in their team or community.



4.4 Design Style and Trend Analysis

- **Trend Detection:**

- Analyzes current design trends and suggests adjustments to align or innovate.



- **Style Recognition:**

- Identifies the design style (e.g., minimalist, vintage) and evaluates adherence to its principles.

4.5 Customizable Scoring Metrics

- **Personalized Metrics:**

- Allows customization of scoring metrics based on project or client requirements.

- **Client Feedback Integration:**

- Incorporates client feedback into the scoring system for tailored results.

4.6 Design History and Improvement Tracking

- **Version Tracking:**

- Tracks design iterations and provides historical scoring and feedback.

- **Improvement Suggestions:**

- Offers a log of past improvements and recurring issues.

4.7 Collaboration and Sharing

- **Team Collaboration:**

- Facilitates collaboration by sharing scores and feedback in real-time.

- **Client Presentation Mode:**

- Generates reports for client presentations, showcasing design scores and rationale.

5. IMPLEMENTATION DETAILS

5.1 Setup and Prerequisites

Before starting with the code, ensure you have the following:

Adobe Creative Cloud SDKs: Install the Adobe UXP (Unified Extensibility Platform) SDK for plugin development.

Node.js: Required for building the UXP plugin.

Python: For implementing AI/ML models.

Adobe Creative Cloud: Photoshop, Illustrator, or XD for testing the plugin.

5.2 Folder Structure

Here's the recommended folder structure for the project:

```
smart-design-score-checker/  
|  
├─ src/  
|   ├─ assets/  
|   |   └─ icons/  
|   ├─ css/  
|   |   └─ styles.css  
|   ├─ js/  
|   |   ├─ main.js  
|   |   ├─ ai-analysis.js  
|   |   └─ feedback.js  
|   ├─ python/  
|   |   └─ ai_model.py  
|   └─ index.html  
|  
├─ .gitignore  
├─ manifest.json  
└─ README.md
```

5.3 Plugin Manifest (manifest.json)

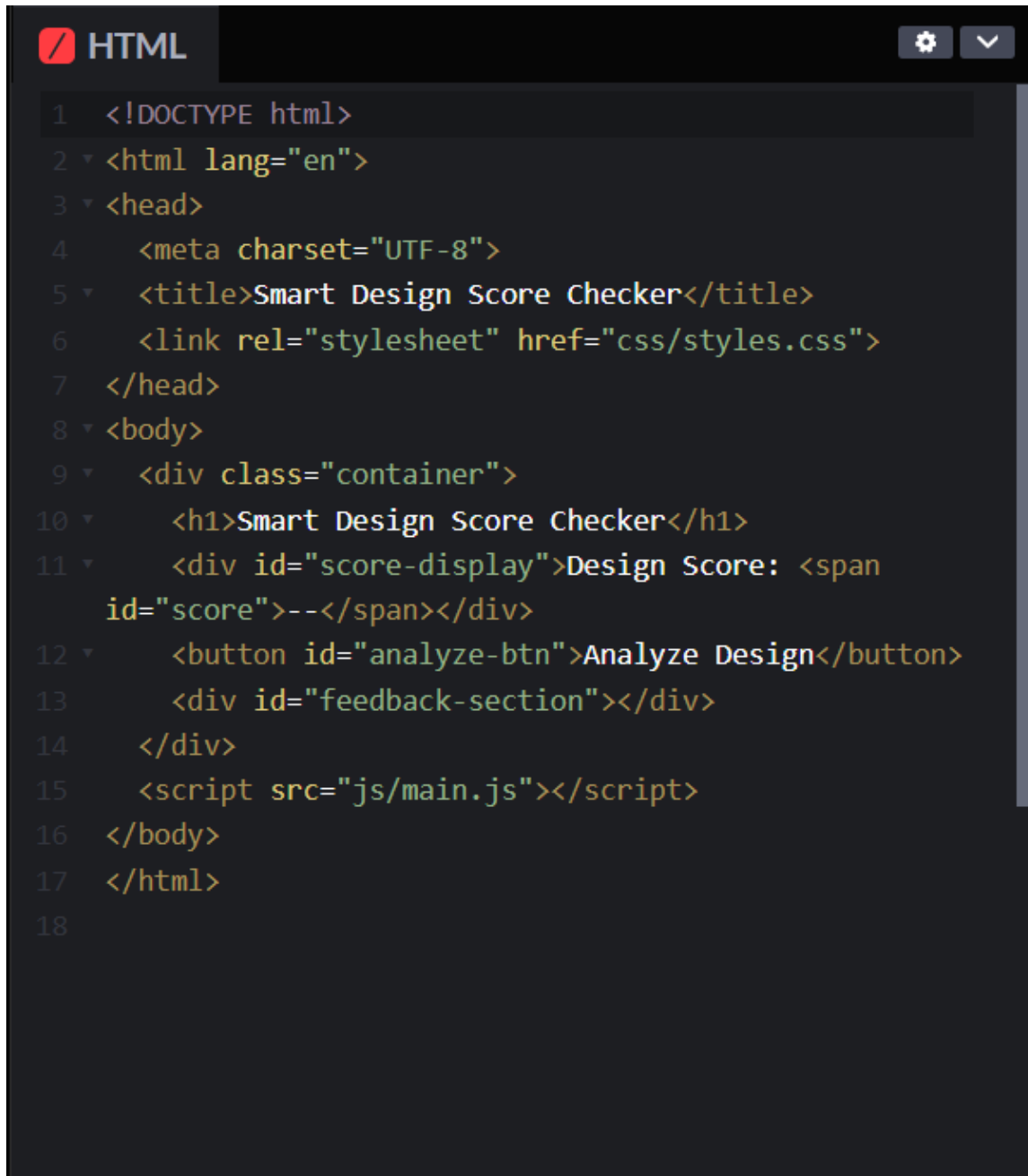
This file defines the plugin's properties, including the Adobe applications it supports.

Json:

```
{
  "id": "com.yourname.smart-design-score-checker",
  "name": "Smart Design Score Checker",
  "version": "1.0.0",
  "main": "index.html",
  "host": {
    "app": ["PS", "ILST", "XD"],
    "minVersion": "22.0.0"
  },
  "ui": {
    "type": "Panel",
    "menu": [
      {
        "label": "Open Smart Design Score Checker",
        "command": "open-panel"
      }
    ]
  },
  "commands": {
    "open-panel": {
      "run": "main.js"
    }
  }
}
```

5.4 HTML User Interface (index.html)

This is a simple UI for the plugin that will be displayed in the Adobe application.

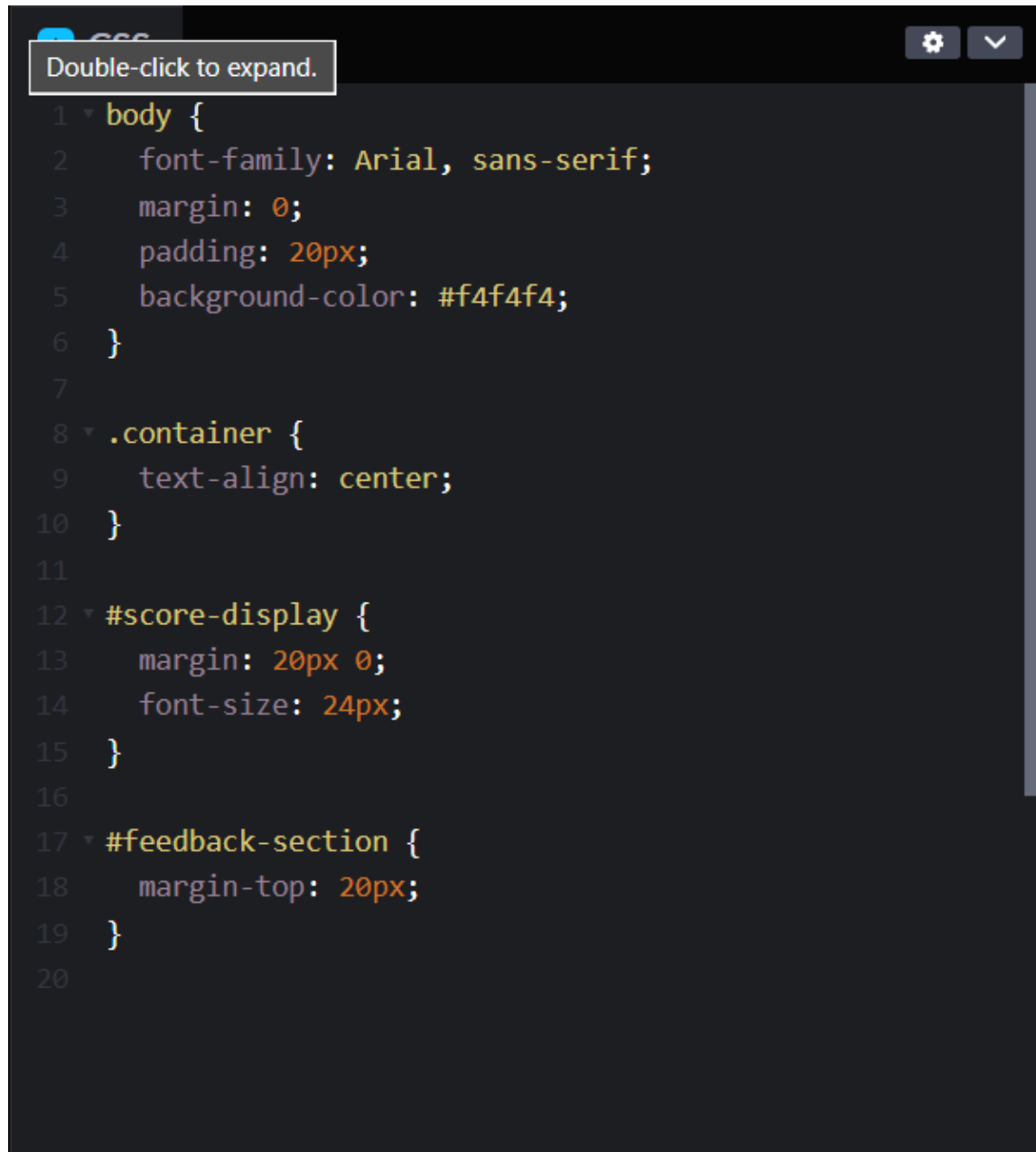
A screenshot of a code editor window titled 'HTML'. The editor has a dark theme and shows the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Smart Design Score Checker</title>
6   <link rel="stylesheet" href="css/styles.css">
7 </head>
8 <body>
9   <div class="container">
10    <h1>Smart Design Score Checker</h1>
11    <div id="score-display">Design Score: <span
      id="score">--</span></div>
12    <button id="analyze-btn">Analyze Design</button>
13    <div id="feedback-section"></div>
14  </div>
15  <script src="js/main.js"></script>
16 </body>
17 </html>
18
```

The code is syntax-highlighted, with tags in blue, attributes in green, and text in white. The editor interface includes a tab labeled 'HTML' with a red icon, and settings and dropdown menus in the top right corner.

5.5 Styling the UI (styles.css)

Basic styling for the plugin UI.



The image shows a code editor window with a dark background. At the top left, there is a tab labeled 'styles.css'. A tooltip with the text 'Double-click to expand.' is positioned over the first line of code. The code is as follows:

```
1 body {
2   font-family: Arial, sans-serif;
3   margin: 0;
4   padding: 20px;
5   background-color: #f4f4f4;
6 }
7
8 .container {
9   text-align: center;
10 }
11
12 #score-display {
13   margin: 20px 0;
14   font-size: 24px;
15 }
16
17 #feedback-section {
18   margin-top: 20px;
19 }
20
```

5.6 Main JavaScript Logic (main.js)

This script handles the UI interactions and communicates with the AI model.

```
JS
document.getElementById('analyze-btn').addEventListener('click', analyzeDesign);

function analyzeDesign() {
  const scoreDisplay = document.getElementById('score');
  const feedbackSection = document.getElementById('feedback-section');

  // Placeholder for design analysis result
  const result = performAIAnalysis();

  // Display score
  scoreDisplay.textContent = result.score;

  // Display feedback
  feedbackSection.innerHTML = result.feedback.map(item => `<p>${item}</p>`).join('');
}

function performAIAnalysis() {
  // Call AI model (This is a placeholder; integration with Python backend is needed)
  return {
    score: Math.floor(Math.random() * 100) + 1, // Placeholder score
    feedback: [
      "Improve color contrast.",
      "Consider using a larger font size.",
      "Align elements properly."
    ]
  };
}
```

5.7 AI Model Implementation (ai_model.py)

This is a simplified version of an AI model that could be used to analyze design elements. In practice, this would involve complex image analysis and machine learning techniques.

```

main.py
1 import numpy as np
2 from PIL import Image
3
4 def analyze_design(image_path):
5     image = Image.open(image_path)
6     score = np.random.randint(1, 100) # Placeholder for actual AI model output
7     feedback = []
8
9     # Placeholder feedback
10    if score < 50:
11        feedback.append("Improve color contrast.")
12        feedback.append("Consider using a larger font size.")
13    else:
14        feedback.append("Good design! Consider minor adjustments.")
15
16    return score, feedback
17
18 if __name__ == "__main__":
19     # Example usage
20     design_score, feedback_list = analyze_design('example_design.png')
21     print(f"Design Score: {design_score}")
22     print("Feedback:")
23     for feedback in feedback_list:
24         print(f"- {feedback}")

```

5.8 Backend Integration (JavaScript to Python)

In a full implementation, you'd need to communicate between the frontend (JavaScript) and the backend (Python). This typically involves setting up a local server using Flask or Django to serve the AI analysis results to the plugin.

Example using Flask:

```

main.py
1 from flask import Flask, request, jsonify
2 from ai_model import analyze_design
3
4 app = Flask(__name__)
5
6 @app.route('/analyze', methods=['POST'])
7 def analyze():
8     data = request.json
9     image_path = data['image_path']
10    score, feedback = analyze_design(image_path)
11    return jsonify(score=score, feedback=feedback)
12
13 if __name__ == "__main__":
14     app.run(debug=True)

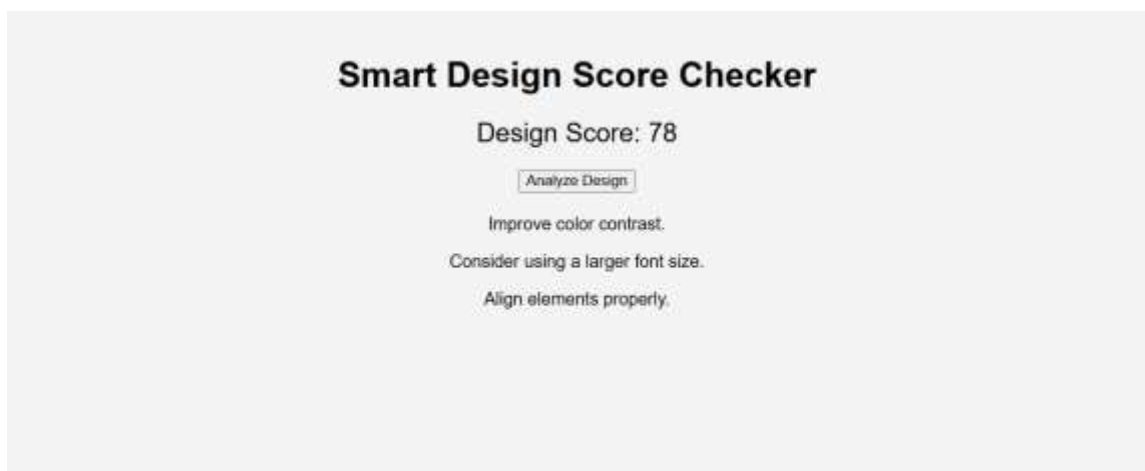
```


5.9 Fetching Analysis Results in JavaScript

```
JS
1 = function performAIAnalysis() {
2   const imagePath = 'path_to_image.png'; // Placeholder for the actual image path
3
4   return fetch('http://localhost:5000/analyze', {
5     method: 'POST',
6     headers: {
7       'Content-Type': 'application/json'
8     },
9     body: JSON.stringify({ image_path: imagePath })
10  })
11  .then(response => response.json())
12  .then(data => {
13    return {
14      score: data.score,
15      feedback: data.feedback
16    };
17  })
18  .catch(error => {
19    console.error('Error:', error);
20    return {
21      score: 0,
22      feedback: ["Error analyzing design."]
23    };
24  });
25 }
```

5.10 Testing and Debugging

Ensure to test the plugin across different Adobe applications and versions. Use console logs and breakpoints to debug any issues during the integration and interaction with the AI model.



6. USER GUIDE

6.1 Installation

- **Step 1:** Download the Smart Design Score Checker plugin from the Adobe Creative Cloud Marketplace.
- **Step 2:** Follow the installation instructions to integrate the plugin with Photoshop, Illustrator, XD, etc.
- **Step 3:** Restart Adobe applications to activate the plugin.

6.2 Getting Started

- **Step 1:** Open a design project in any supported Adobe app.
- **Step 2:** Launch the Smart Design Score Checker from the plugin menu.
- **Step 3:** Begin designing as usual; the tool will automatically start evaluating your work.
- **Step 4:** Review the real-time score displayed on the plugin dashboard.

6.3 Customizing Scoring Metrics

- **Step 1:** Navigate to the "Settings" section in the plugin.
- **Step 2:** Choose the metrics you want to prioritize (e.g., aesthetics, usability, accessibility).
- **Step 3:** Save your preferences, and the scoring algorithm will adjust accordingly.

Customizing Scoring Metrics		
[Accuracy]	[Precision]	[Recall]
[F1 Score]	[Custom]	
[Slider for Precision]	[Dropdown for Weight]	
[Graph showing Impact of Adjustments]		
[Comparison Table]		
Metric	Original Score	Adjusted Score
Accuracy	0.85	0.88
Precision	0.90	0.91
...
[Callouts and Tips]		
- "Adjust Precision Weight"		
- "Monitor Impact on F1 Score"		

6.4 Viewing and Interpreting Feedback

- **Step 1:** Click on the feedback tab to view detailed suggestions for improvement.
- **Step 2:** Follow the suggestions, such as color adjustments or layout changes, to improve your score.
- **Step 3:** Use the "Compare" feature to see how your score improves over time.

6.5 Collaboration and Sharing

- **Step 1:** Invite team members to collaborate by sharing the project link.
- **Step 2:** View and compare design scores in real-time with your team.
- **Step 3:** Use the "Export" feature to generate a report for client presentations.

7. CONCLUSION

7.1 Summary

The Smart Design Score Checker provides designers with a powerful tool to enhance their creative process. By offering real-time feedback, industry comparisons, and accessibility checks, it ensures that every design meets high professional standards. This tool not only saves time and effort but also contributes to continuous learning and improvement.

7.2 Future Enhancements

- **AI-Driven Design Suggestions:** Expanding the AI's capabilities to suggest complete design ideas based on user inputs.
- **Extended Accessibility Features:** Adding

8. References for Documentation

1. Books:

- **"Pattern Recognition and Machine Learning" by Christopher M. Bishop**
 - Provides a comprehensive introduction to scoring metrics.
- **"Machine Learning: A Probabilistic Perspective" by Kevin P. Murphy**
 - Discusses scoring metrics in probabilistic models.

2. Online Resources:

- **"Evaluation Metrics for Machine Learning Models" by Jason Brownlee**
 - Covers various metrics with practical examples.
- **"A Comprehensive Guide to Evaluation Metrics for Machine Learning Models" by Towards Data Science**
 - Provides an overview of metrics with case studies.

3. Documentation:

- **"Scikit-learn Documentation: Model Evaluation"**
 - Official guide to model evaluation metrics in Python.