<div>

<button id="btnBackup" onclick="backupSettings()">💾 حفظ نسخة احتياطية</button>

<p id="backupStatus" style="margin-top:8px; color:green;"></p>

</div>

<script>

function backupSettings() {

document.getElementById('btnBackup').disabled = true;

document.getElementById('backupStatus').innerText = 'جاري الحفظ...';

google.script.run

.withSuccessHandler(function(url) {

document.getElementById('backupStatus').innerHTML =

'✅ تم الحفظ: <a href="' + url + '" target="\_blank">عرض الملف</a>';

})

.withFailureHandler(function(err) {

document.getElementById('backupStatus').innerText =

'❌ فشل الحفظ: ' + err.message;

})

.backupEnvSettingsToDrive();

}

</script>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Enhanced Application</title>

<!-- Tailwind CSS CDN -->

<script src="https://cdn.tailwindcss.com"></script>

<!-- Inter Font -->

<link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700&display=swap" rel="stylesheet">

<style>

body {

font-family: 'Inter', sans-serif;

@apply bg-gray-100 flex justify-center items-center min-h-screen p-4;

}

.container {

@apply bg-white p-8 rounded-xl shadow-2xl max-w-lg w-full text-center border border-gray-200;

}

h1 {

@apply text-4xl font-extrabold text-gray-800 mb-4;

}

p {

@apply text-gray-600 text-lg mb-6 leading-relaxed;

}

.button {

@apply bg-blue-600 text-white py-3 px-8 rounded-xl font-bold text-lg shadow-lg hover:bg-blue-700 transition-all duration-300 ease-in-out transform hover:scale-105 inline-block;

}

.message-box {

@apply mt-6 p-4 rounded-lg font-medium text-sm;

}

.message-box.error {

@apply bg-red-100 text-red-700 border border-red-200;

}

.message-box.success {

@apply bg-green-100 text-green-700 border border-green-200;

}

.message-box.warning {

@apply bg-yellow-100 text-yellow-700 border border-yellow-200;

}

</style>

<!--

\*\*IMPORTANT:\*\* Replace 'YOUR\_API\_KEY' with your actual Google Maps API key.

This script needs to be loaded before any JavaScript code attempts to use the 'google' object.

The 'callback=initMap' ensures initMap is called once the API is ready.

-->

<script async defer src="https://maps.googleapis.com/maps/api/js?key=YOUR\_API\_KEY&callback=initMap"></script>

</head>

<body>

<div class="container">

<h1>Welcome to Your App</h1>

<p>

This is an enhanced application demonstrating how to include external scripts and handle common JavaScript errors.

The Google Maps API script has been added for potential map functionalities.

</p>

<button class="button" id="backendCallBtn">

Simulate Backend Call

</button>

<div id="messageBox" class="message-box hidden"></div>

<div id="userIdDisplay" class="mt-4 text-gray-500 text-sm"></div>

</div>

<script type="module">

// Placeholder for Firebase configuration and initialization

// These global variables are expected to be provided by the Canvas environment.

const appId = typeof \_\_app\_id !== 'undefined' ? \_\_app\_id : 'default-app-id';

const firebaseConfig = typeof \_\_firebase\_config !== 'undefined' ? JSON.parse(\_\_firebase\_config) : {};

const initialAuthToken = typeof \_\_initial\_auth\_token !== 'undefined' ? \_\_initial\_auth\_token : null;

// Import Firebase modules

import { initializeApp } from "https://www.gstatic.com/firebasejs/11.6.1/firebase-app.js";

import { getAuth, signInAnonymously, signInWithCustomToken, onAuthStateChanged } from "https://www.gstatic.com/firebasejs/11.6.1/firebase-auth.js";

import { getFirestore, doc, getDoc, setDoc, collection, query, where, addDoc, getDocs, onSnapshot } from "https://www.gstatic.com/firebasejs/11.6.1/firebase-firestore.js";

let app;

let db;

let auth;

let userId;

let isAuthReady = false; // Flag to indicate if auth state has been checked

const messageBox = document.getElementById('messageBox');

const userIdDisplay = document.getElementById('userIdDisplay');

/\*\*

\* Displays a message in the message box.

\* @param {string} message - The message to display.

\* @param {'success'|'error'|'warning'|'info'} type - The type of message.

\*/

function displayMessage(message, type) {

messageBox.textContent = message;

messageBox.className = 'message-box'; // Reset classes

messageBox.classList.add(type);

messageBox.classList.remove('hidden');

// Optional: Hide message after a few seconds

setTimeout(() => {

messageBox.classList.add('hidden');

}, 5000);

}

/\*\*

\* Initializes Firebase app and handles user authentication.

\*/

async function initializeFirebaseAndAuth() {

try {

app = initializeApp(firebaseConfig);

db = getFirestore(app);

auth = getAuth(app);

onAuthStateChanged(auth, async (user) => {

if (user) {

userId = user.uid;

console.log("User ID:", userId);

userIdDisplay.textContent = `Authenticated User ID: ${userId}`;

displayMessage(`Authenticated User ID: ${userId}`, 'success');

} else {

// Sign in anonymously if no custom token or user is not signed in

if (initialAuthToken) {

await signInWithCustomToken(auth, initialAuthToken);

console.log("Signed in with custom token.");

} else {

await signInAnonymously(auth);

console.log("Signed in anonymously.");

}

// The onAuthStateChanged listener will be called again with the new user

}

isAuthReady = true; // Auth state has been checked

});

} catch (error) {

console.error("Error initializing Firebase or authenticating:", error);

displayMessage(`Firebase initialization error: ${error.message}`, 'error');

}

}

/\*\*

\* Simulates a backend call and performs a Firestore operation.

\*/

async function callBackendFunction() {

if (!isAuthReady) {

displayMessage('Firebase authentication is not ready yet. Please wait.', 'warning');

return;

}

try {

displayMessage('Simulating backend function call...', 'info');

console.log("Backend function simulated successfully.");

// Example of a Firestore operation after authentication

if (db && userId) {

const docRef = doc(db, `artifacts/${appId}/users/${userId}/app\_data`, "exampleDoc");

await setDoc(docRef, { timestamp: new Date().toISOString(), message: "Data saved after backend call simulation" });

displayMessage('Example data saved to Firestore.', 'success');

console.log("Example data saved to Firestore.");

} else {

displayMessage('Firestore not ready or userId not available. Cannot save data.', 'warning');

console.warn("Firestore not ready or userId not available. Cannot save data.");

}

} catch (error) {

console.error("Error in callBackendFunction:", error);

displayMessage(`Error during backend call: ${error.message}`, 'error');

}

}

// Global callback for Google Maps API

window.initMap = function() {

console.log("Google Maps API loaded successfully.");

// You can initialize your map here if needed

// For example:

// const map = new google.maps.Map(document.getElementById('map'), {

// center: { lat: -34.397, lng: 150.644 },

// zoom: 8,

// });

displayMessage('Google Maps API loaded successfully!', 'info');

};

// Event listener for DOM content loaded

document.addEventListener('DOMContentLoaded', () => {

initializeFirebaseAndAuth(); // Initialize Firebase when DOM is ready

document.getElementById('backendCallBtn').addEventListener('click', callBackendFunction);

});

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<base target="\_top">

<meta charset="UTF-8" />

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

<title>Gemini Code Reviewer</title>

<style>

:root {

--bg-color: #111827;

--bg-secondary-color: #1f2937;

--bg-tertiary-color: #374151;

--border-color: #4b5563;

--text-color: #f3f4f6;

--text-secondary-color: #d1d5db;

--text-tertiary-color: #9ca3af;

--primary-color: #4f46e5;

--primary-hover-color: #4338ca;

--primary-disabled-color: #312e81;

--error-color: #be123c;

--error-bg-color: rgba(190, 18, 60, 0.2);

--error-border-color: rgba(190, 18, 60, 0.5);

--success-color: #16a34a;

--font-sans: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, sans-serif;

--font-mono: "Courier New", Courier, monospace;

}

\*, \*::before, \*::after { box-sizing: border-box; }

body, html { margin: 0; padding: 0; width: 100%; height: 100%; overflow-x: hidden; }

body {

background-color: var(--bg-color);

color: var(--text-color);

font-family: var(--font-sans);

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

#root { min-height: 100vh; }

.container { max-width: 800px; margin: 0 auto; padding: 1rem; }

button, input, select, textarea {

font-family: inherit;

font-size: 1rem;

border-radius: 0.5rem;

border: 1px solid var(--border-color);

background-color: var(--bg-secondary-color);

color: var(--text-color);

transition: all 0.2s ease-in-out;

}

button {

cursor: pointer;

padding: 0.75rem 1.25rem;

font-weight: 600;

background-color: var(--primary-color);

border-color: transparent;

}

button:hover { background-color: var(--primary-hover-color); }

button:disabled { background-color: var(--primary-disabled-color); color: var(--text-tertiary-color); cursor: not-allowed; }

textarea, input, select { padding: 0.75rem; width: 100%; }

textarea:focus, input:focus, select:focus {

outline: none;

border-color: var(--primary-color);

box-shadow: 0 0 0 2px rgba(79, 70, 229, 0.5);

}

.loading-spinner {

animation: spin 1s linear infinite;

width: 1.25rem;

height: 1.25rem;

border-radius: 50%;

border: 2px solid var(--border-color);

border-top-color: var(--text-color);

}

@keyframes spin {

to { transform: rotate(360deg); }

}

</style>

</head>

<body>

<div id="root"></div>

<script type="importmap">

{

"imports": {

"react": "https://esm.sh/react@19.1.0",

"react-dom/client": "https://esm.sh/react-dom@19.1.0/client",

"@google/genai": "https://esm.sh/@google/genai@1.9.0"

}

}

</script>

<script type="module">

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

import ReactDOM from 'react-dom/client';

import { GoogleGenAI, Type } from "@google/genai";

// --- START OF TYPES ---

const FeedbackType = {

BUG: 'BUG',

SUGGESTION: 'SUGGESTION',

PRAISE: 'PRAISE',

STYLE: 'STYLE'

};

// --- START OF CONSTANTS ---

const SUPPORTED\_LANGUAGES = [

'JavaScript', 'TypeScript', 'Python', 'Java', 'Go', 'Rust', 'C++', 'C#', 'HTML', 'CSS', 'SQL', 'Shell'

];

// --- START OF ICONS ---

const CodeIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("polyline", { points: "16 18 22 12 16 6" }), React.createElement("polyline", { points: "8 6 2 12 8 18" }));

const SparklesIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", viewBox: "0 0 20 20", fill: "currentColor", ...props }, React.createElement("path", { fillRule: "evenodd", d: "M10 2a.75.75 0 01.75.75v.263a3.5 3.5 0 014.288 4.288H15.25a.75.75 0 010 1.5h-.263a3.5 3.5 0 01-4.288 4.288v.263a.75.75 0 01-1.5 0v-.263a3.5 3.5 0 01-4.288-4.288H4.75a.75.75 0 010-1.5h.263A3.5 3.5 0 019.25 4.75v-.263A.75.75 0 0110 2zM8.341 8.341a.5.5 0 00-.707.707l.707-.707zm3.318 3.318a.5.5 0 00.707-.707l-.707.707zM5.505 11.293a.5.5 0 00.707.707l-.707-.707zM11.293 5.505a.5.5 0 00-.707-.707l.707.707z", clipRule: "evenodd" }));

const BugIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("path", { d: "M12 20h-4a2 2 0 0 1 -2 -2v-12a2 2 0 0 1 2 -2h12v4" }), React.createElement("path", { d: "M20 12h-4a2 2 0 0 0 -2 2v4a2 2 0 0 0 2 2h4v-8z" }), React.createElement("path", { d: "M16 9h4" }), React.createElement("path", { d: "M12 6h.01" }), React.createElement("path", { d: "M12 12h.01" }));

const LightBulbIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("path", { d: "M9 18h6" }), React.createElement("path", { d: "M10 22h4" }), React.createElement("path", { d: "M9 14a6.04 6.04 0 0 1 6 0" }), React.createElement("path", { d: "M12 2v4" }), React.createElement("path", { d: "M3.5 6.5l2 2" }), React.createElement("path", { d: "M18.5 6.5l-2 2" }));

const PaintBrushIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("path", { d: "M9.5 2.5a1 1 0 0 1 1 1v1a1 1 0 0 1-1 1h-3a1 1 0 0 1-1-1v-1a1 1 0 0 1 1-1h3z" }), React.createElement("path", { d: "M12.5 5.5a1 1 0 0 1 1 1v1a1 1 0 0 1-1 1h-1a1 1 0 0 1-1-1v-1a1 1 0 0 1 1-1h1z" }), React.createElement("path", { d: "M16 3v18" }), React.createElement("path", { d: "M10 3v18" }));

const WandIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("path", { d: "M15 4V2" }), React.createElement("path", { d: "M15 8V6" }), React.createElement("path", { d: "M12.5 6.5L14 5" }), React.createElement("path", { d: "M17 9.5L15.5 8" }), React.createElement("path", { d: "M20 9.5h-2" }), React.createElement("path", { d: "M4 9.5H2" }), React.createElement("path", { d: "m9 13.5 1 1" }), React.createElement("path", { d: "M4.5 18.5 3 20" }), React.createElement("path", { d: "M21 20l-1.5-1.5" }), React.createElement("path", { d: "M18.5 4.5 20 3" }), React.createElement("path", { d: "M9 4.5 7.5 3" }), React.createElement("path", { d: "M15 12.5v-1.5a6 6 0 0 0-6-6h-1.5a6 6 0 0 0-6 6v1.5a6 6 0 0 0 6 6h1.5a6 6 0 0 0 6-6Z" }));

const ChatBubbleIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("path", { d: "M21 15a2 2 0 0 1-2 2H7l-4 4V5a2 2 0 0 1 2-2h14a2 2 0 0 1 2 2z" }));

const DownloadIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("path", { d: "M21 15v4a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2v-4" }), React.createElement("polyline", { points: "7 10 12 15 17 10" }), React.createElement("line", { x1: "12", y1: "15", x2: "12", y2: "3" }));

const UploadIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("path", { d: "M21 15v4a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2v-4" }), React.createElement("polyline", { points: "17 8 12 3 7 8" }), React.createElement("line", { x1: "12", y1: "3", x2: "12", y2: "15" }));

// --- START OF geminiService ---

const ai = new GoogleGenAI({ apiKey: process.env.API\_KEY });

const refactorSchema = { type: Type.OBJECT, properties: { startLine: { type: Type.INTEGER, description: "The 1-based starting line number of the code block to be replaced. Inclusive." }, endLine: { type: Type.INTEGER, description: "The 1-based ending line number of the code block to be replaced. Inclusive. Should be >= startLine." }, code: { type: Type.STRING, description: "The new code to replace the specified lines. For a deletion, this must be an empty string." } }, required: ["startLine", "endLine", "code"] };

const reviewSchema = { type: Type.OBJECT, properties: { overallScore: { type: Type.INTEGER, description: "A score from 0 to 100 representing the overall quality of the code." }, summary: { type: Type.STRING, description: "A brief, high-level summary of the code review findings." }, feedbackItems: { type: Type.ARRAY, description: "A list of specific feedback points about the code.", items: { type: Type.OBJECT, properties: { type: { type: Type.STRING, enum: ['BUG', 'SUGGESTION', 'PRAISE', 'STYLE'], description: "The category of feedback: BUG, SUGGESTION, PRAISE, or STYLE." }, line: { type: Type.INTEGER, description: "The primary line number the feedback applies to. For multi-line changes, this should be the starting line. Null for general comments.", nullable: true }, description: { type: Type.STRING, description: "A clear and concise description of the issue or praise." }, suggestion: { type: Type.STRING, description: "A human-readable explanation of the praise or suggested change. Null if not applicable.", nullable: true }, refactor: { ...refactorSchema, description: "An actionable code refactoring suggestion. Provide this only for BUG, SUGGESTION, or STYLE feedback where a direct code change is applicable. Null otherwise.", nullable: true, } }, required: ["type", "description"] } } }, required: ["overallScore", "summary", "feedbackItems"] };

const reviewCode = async (code, language, customInstructions) => {

const customInstructionsSection = customInstructions.trim() ? `\nThe user has provided the following specific instructions for this review. Pay close attention to them:\n---\n${customInstructions.trim()}\n---\n` : '';

const prompt = `Please perform a thorough code review on the following ${language} code snippet.\nAnalyze it for bugs, style violations, performance issues, and areas for improvement.\nAlso, identify any good practices worth praising.\n${customInstructionsSection}\nFor each feedback item that involves a concrete code change (bugs, suggestions, style fixes), provide a 'refactor' object. This object must contain:\n- 'startLine': The 1-based starting line number of the code to replace.\n- 'endLine': The 1-based ending line number of the code to replace.\n- 'code': The new code that will replace the specified lines.\n\nExamples:\n- To REPLACE lines 5-7 with new code: \`{ "startLine": 5, "endLine": 7, "code": "new replacement code" }\`\n- To DELETE line 10: \`{ "startLine": 10, "endLine": 10, "code": "" }\`\n- To INSERT code BEFORE line 15: \`{ "startLine": 15, "endLine": 14, "code": "code to insert" }\` (Note: endLine is less than startLine for insertion)\n\nThe 'suggestion' field should always contain a human-readable explanation of the change, not the code itself.\n\nCode to review:\n\`\`\`${language.toLowerCase()}\n${code}\n\`\`\``;

try {

const response = await ai.models.generateContent({ model: "gemini-2.5-flash", contents: prompt, config: { systemInstruction: "You are an expert code reviewer. Your analysis must be insightful, accurate, and constructive. You must respond ONLY with a JSON object that adheres to the provided schema.", responseMimeType: "application/json", responseSchema: reviewSchema } });

const text = response.text.trim(); if (!text) { throw new Error("Received an empty response from the API."); }

const parsedResult = JSON.parse(text);

parsedResult.feedbackItems.forEach(item => { item.id = crypto.randomUUID(); });

return parsedResult;

} catch (error) { console.error("Error calling Gemini API:", error); if (error instanceof Error && error.message.includes('SAFETY')) { throw new Error("The code could not be reviewed due to safety restrictions. Please ensure the code doesn't violate safety policies."); } throw new Error(`Failed to get review from Gemini API. ${error instanceof Error ? error.message : ''}`); }

};

const continueDiscussion = async (code, language, feedbackItem, question) => {

const history = (feedbackItem.discussion || []).map(msg => `${msg.author === 'user' ? 'User' : 'Assistant'}: ${msg.message}`).join('\n');

const prompt = `You are an expert code reviewer in an ongoing discussion with a user about a specific piece of feedback you provided.\nYour tone should be helpful, clarifying, and concise.\nThis was the original code snippet under review (${language}):\n\`\`\`${language.toLowerCase()}\n${code}\n\`\`\`\nThis was your original feedback for the code around line ${feedbackItem.line}:\n- Category: ${feedbackItem.type}\n- Description: ${feedbackItem.description}\n- Your Suggestion: ${feedbackItem.suggestion || 'N/A'}\nSo far, the conversation has been:\n${history.length > 0 ? history : "No previous discussion."}\nThe user has a new question for you:\nUser: ${question}\nYour task is to provide a direct and helpful answer to the user's question.\nFocus only on providing your next response. Do not repeat the context I just gave you.`;

try {

const response = await ai.models.generateContent({ model: "gemini-2.5-flash", contents: prompt, });

const text = response.text.trim(); if (!text) { throw new Error("Received an empty follow-up response from the API."); } return text;

} catch (error) { console.error("Error calling Gemini API for discussion:", error); throw new Error(`Failed to get discussion response from Gemini API. ${error instanceof Error ? error.message : ''}`); }

};

// --- START OF COMPONENTS ---

const Header = () => React.createElement("header", { style: { padding: '1.5rem 0', borderBottom: '1px solid var(--border-color)', backgroundColor: 'rgba(17, 24, 39, 0.3)', backdropFilter: 'blur(4px)', position: 'sticky', top: 0, zIndex: 10 } }, React.createElement("div", { className: 'container', style: { display: 'flex', alignItems: 'center', gap: '1rem' } }, React.createElement("div", { style: { backgroundColor: 'var(--primary-color)', padding: '0.5rem', borderRadius: '0.5rem', boxShadow: '0 4px 6px -1px rgba(79, 70, 229, 0.3), 0 2px 4px -2px rgba(79, 70, 229, 0.3)' } }, React.createElement(CodeIcon, { style: { height: '1.5rem', width: '1.5rem', color: 'white' } })), React.createElement("h1", { style: { fontSize: '1.5rem', fontWeight: 'bold', color: 'var(--text-color)', letterSpacing: '-0.025em' } }, "Gemini Code Reviewer")));

const GoogleSheetIntegration = ({ onFetchCode, onUpdateCode, isSheetContext, isLoading }) => {

if (!isSheetContext) return null;

return React.createElement("div", { style: { marginBottom: '1.5rem', padding: '1rem', backgroundColor: 'rgba(16, 185, 129, 0.1)', border: '1px solid rgba(5, 150, 105, 0.3)', borderRadius: '0.5rem' } },

React.createElement("h3", { style: { fontSize: '1.125rem', fontWeight: '600', color: '#6ee7b7', marginBottom: '0.75rem' } }, "Google Sheets Integration"),

React.createElement("p", { style: { fontSize: '0.875rem', color: 'var(--text-secondary-color)', marginBottom: '1rem' } }, "Use these buttons to sync code with your active cell. Applying a fix will automatically update the cell."),

React.createElement("div", { style: { display: 'flex', gap: '1rem', flexDirection: 'row' } },

React.createElement("button", { onClick: onFetchCode, disabled: isLoading, style: { flex: 1, display: 'inline-flex', alignItems: 'center', justifyContent: 'center', gap: '0.5rem', backgroundColor: 'var(--bg-tertiary-color)' } }, React.createElement(DownloadIcon, { style: { height: '1rem', width: '1rem' } }), "Get Code from Cell"),

React.createElement("button", { onClick: onUpdateCode, disabled: isLoading, style: { flex: 1, display: 'inline-flex', alignItems: 'center', justifyContent: 'center', gap: '0.5rem', backgroundColor: 'var(--bg-tertiary-color)' } }, React.createElement(UploadIcon, { style: { height: '1rem', width: '1rem' } }), "Update Cell with Code")

)

);

};

const CodeInputForm = ({ code, setCode, onReview, isLoading }) => {

const [language, setLanguage] = useState(SUPPORTED\_LANGUAGES[0]);

const [customInstructions, setCustomInstructions] = useState('');

const handleSubmit = (e) => { e.preventDefault(); onReview(language, customInstructions); };

return React.createElement("form", { onSubmit: handleSubmit, style: { display: 'flex', flexDirection: 'column', gap: '1.5rem' } },

React.createElement("div", { style: { display: 'flex', gap: '1rem', flexDirection: 'column' } },

React.createElement("div", { style: { flexGrow: 1 } },

React.createElement("label", { htmlFor: "code-input", style: { display: 'block', fontSize: '0.875rem', fontWeight: '500', color: 'var(--text-secondary-color)', marginBottom: '0.5rem' } }, "Your Code"),

React.createElement("textarea", { id: "code-input", value: code, onChange: (e) => setCode(e.target.value), placeholder: `// Paste your ${language} code here...`, style: { width: '100%', height: '20rem', padding: '1rem', fontFamily: 'var(--font-mono)', fontSize: '0.875rem', backgroundColor: 'var(--bg-color)', resize: 'vertical' }, spellCheck: "false", disabled: isLoading })

),

React.createElement("div", { style: { width: '100%' } },

React.createElement("label", { htmlFor: "language-select", style: { display: 'block', fontSize: '0.875rem', fontWeight: '500', color: 'var(--text-secondary-color)', marginBottom: '0.5rem' } }, "Language"),

React.createElement("select", { id: "language-select", value: language, onChange: (e) => setLanguage(e.target.value), disabled: isLoading }, SUPPORTED\_LANGUAGES.map((lang) => React.createElement("option", { key: lang, value: lang }, lang)))

)

),

React.createElement("div", null,

React.createElement("label", { htmlFor: "custom-instructions", style: { display: 'block', fontSize: '0.875rem', fontWeight: '500', color: 'var(--text-secondary-color)', marginBottom: '0.5rem' } }, "Custom Instructions ", React.createElement("span", { style: { color: 'var(--text-tertiary-color)' } }, "(Optional)")),

React.createElement("textarea", { id: "custom-instructions", value: customInstructions, onChange: (e) => setCustomInstructions(e.target.value), placeholder: "e.g., Focus on security and performance...", style: { width: '100%', height: '6rem', padding: '1rem', fontFamily: 'var(--font-sans)', resize: 'vertical' }, disabled: isLoading })

),

React.createElement("div", null,

React.createElement("button", { type: "submit", disabled: isLoading || !code.trim(), style: { width: '100%', display: 'flex', alignItems: 'center', justifyContent: 'center', gap: '0.5rem' } },

isLoading ? React.createElement(React.Fragment, null, React.createElement("div", { className: 'loading-spinner' }), "Reviewing...") : React.createElement(React.Fragment, null, React.createElement(SparklesIcon, { style: { height: '1.25rem', width: '1.25rem' } }), "Review Code")

)

)

);

};

const DiscussionThread = ({ messages, isLoading, onSendMessage }) => {

const [input, setInput] = useState('');

const endOfMessagesRef = useRef(null);

const handleSubmit = (e) => { e.preventDefault(); if (input.trim() && !isLoading) { onSendMessage(input.trim()); setInput(''); } };

useEffect(() => { endOfMessagesRef.current?.scrollIntoView({ behavior: 'smooth' }); }, [messages, isLoading]);

const ChatBubble = ({ message }) => {

const isUser = message.author === 'user';

return React.createElement("div", { style: { display: 'flex', alignItems: 'flex-start', gap: '0.625rem', justifyContent: isUser ? 'flex-end' : 'flex-start' } },

!isUser && React.createElement("div", { style: { flexShrink: 0, width: '2rem', height: '2rem', borderRadius: '50%', background: 'linear-gradient(to bottom right, #6366f1, #a855f7)', display: 'flex', alignItems: 'center', justifyContent: 'center' } }, React.createElement(SparklesIcon, { style: { width: '1.25rem', height: '1.25rem', color: '#e0e7ff' } })),

React.createElement("div", { style: { display: 'flex', flexDirection: 'column', gap: '0.25rem', width: '100%', maxWidth: '320px', lineHeight: 1.5, padding: '0.75rem', borderRadius: '0.75rem', backgroundColor: isUser ? 'var(--primary-color)' : 'var(--bg-tertiary-color)', borderBottomRightRadius: isUser ? 0 : '0.75rem', borderBottomLeftRadius: isUser ? '0.75rem' : 0 } }, React.createElement("p", { style: { fontSize: '0.875rem', fontWeight: '400', color: 'white' } }, message.message)),

isUser && React.createElement("div", { style: { flexShrink: 0, width: '2rem', height: '2rem', borderRadius: '50%', backgroundColor: 'var(--bg-tertiary-color)', display: 'flex', alignItems: 'center', justifyContent: 'center' } }, React.createElement("svg", { style: { width: '1.25rem', height: '1.25rem', color: 'var(--text-secondary-color)' }, fill: "currentColor", viewBox: "0 0 20 20", xmlns: "http://www.w3.org/2000/svg" }, React.createElement("path", { fillRule: "evenodd", d: "M10 9a3 3 0 100-6 3 3 0 000 6zm-7 9a7 7 0 1114 0H3z", clipRule: "evenodd" })))

);

};

return React.createElement("div", { style: { display: 'flex', flexDirection: 'column', gap: '1rem' } },

React.createElement("div", { style: { display: 'flex', flexDirection: 'column', gap: '1rem', maxHeight: '16rem', overflowY: 'auto', paddingRight: '0.5rem' } },

messages.map((msg, index) => React.createElement(ChatBubble, { key: index, message: msg })),

isLoading && React.createElement("div", { style: { display: 'flex', alignItems: 'flex-start', gap: '0.625rem' } },

React.createElement("div", { style: { flexShrink: 0, width: '2rem', height: '2rem', borderRadius: '50%', background: 'linear-gradient(to bottom right, #6366f1, #a855f7)', display: 'flex', alignItems: 'center', justifyContent: 'center' } }, React.createElement(SparklesIcon, { style: { width: '1.25rem', height: '1.25rem', color: '#e0e7ff', animation: 'pulse 1.5s infinite' } })),

React.createElement("div", { style: { display: 'flex', flexDirection: 'column', gap: '0.25rem', width: '100%', maxWidth: '320px', lineHeight: 1.5, padding: '0.75rem', borderRadius: '0.75rem', backgroundColor: 'var(--bg-tertiary-color)', borderBottomLeftRadius: 0 } }, React.createElement("p", { style: { fontSize: '0.875rem', fontStyle: 'italic', color: 'var(--text-tertiary-color)' } }, "Gemini is thinking..."))

),

React.createElement("div", { ref: endOfMessagesRef })

),

React.createElement("form", { onSubmit: handleSubmit, style: { display: 'flex', alignItems: 'center', gap: '0.5rem' } },

React.createElement("input", { type: "text", value: input, onChange: (e) => setInput(e.target.value), placeholder: "Ask a follow-up question...", disabled: isLoading, style: { flexGrow: 1 } }),

React.createElement("button", { type: "submit", disabled: isLoading || !input.trim() }, "Send")

)

);

};

const ReviewOutput = ({ review, isLoading, onApplyFix, discussionState, onSendMessage }) => {

if (isLoading) return React.createElement("div", { style: { marginTop: '1.5rem', textAlign: 'center', color: 'var(--text-tertiary-color)' } },

React.createElement("p", null, "Analyzing your code..."),

React.createElement("div", { style: { marginTop: '1rem', backgroundColor: 'rgba(55, 65, 81, 0.5)', borderRadius: '0.5rem', padding: '1.5rem', display: 'flex', flexDirection: 'column', gap: '1rem', animation: 'pulse 2s cubic-bezier(0.4, 0, 0.6, 1) infinite' } },

React.createElement("div", { style: { height: '1rem', backgroundColor: 'var(--bg-tertiary-color)', borderRadius: '0.25rem', width: '25%' } }),

React.createElement("div", { style: { height: '2rem', backgroundColor: 'var(--bg-tertiary-color)', borderRadius: '0.25rem', width: '33%' } }),

React.createElement("div", { style: { display: 'flex', flexDirection: 'column', gap: '0.5rem' } },

React.createElement("div", { style: { height: '1rem', backgroundColor: 'var(--bg-tertiary-color)', borderRadius: '0.25rem', width: '100%' } }),

React.createElement("div", { style: { height: '1rem', backgroundColor: 'var(--bg-tertiary-color)', borderRadius: '0.25rem', width: '83.33%' } })

)

)

);

if (!review) return React.createElement("div", { style: { marginTop: '2rem', padding: '3rem 0', textAlign: 'center', border: '2px dashed var(--border-color)', borderRadius: '0.5rem' } }, React.createElement("p", { style: { color: 'var(--text-tertiary-color)' } }, "Your code review will appear here."));

const getFeedbackTypeStyles = (type) => {

switch (type) {

case FeedbackType.BUG: return { Icon: BugIcon, bgColor: 'rgba(225, 29, 72, 0.15)', borderColor: 'rgba(225, 29, 72, 0.4)', textColor: '#fecdd3', title: 'Bug Detected' };

case FeedbackType.SUGGESTION: return { Icon: LightBulbIcon, bgColor: 'rgba(217, 119, 6, 0.15)', borderColor: 'rgba(217, 119, 6, 0.4)', textColor: '#fde68a', title: 'Suggestion' };

case FeedbackType.STYLE: return { Icon: PaintBrushIcon, bgColor: 'rgba(59, 130, 246, 0.15)', borderColor: 'rgba(59, 130, 246, 0.4)', textColor: '#bfdbfe', title: 'Style Suggestion' };

case FeedbackType.PRAISE: return { Icon: SparklesIcon, bgColor: 'rgba(22, 163, 74, 0.15)', borderColor: 'rgba(22, 163, 74, 0.4)', textColor: '#bbf7d0', title: 'Good Practice' };

default: return { Icon: LightBulbIcon, bgColor: 'var(--bg-tertiary-color)', borderColor: 'var(--border-color)', textColor: 'var(--text-secondary-color)', title: 'Feedback' };

}

};

const FeedbackCard = ({ item }) => {

const { Icon, bgColor, borderColor, textColor, title } = getFeedbackTypeStyles(item.type);

const canApplyFix = item.refactor && (item.type === FeedbackType.BUG || item.type === FeedbackType.SUGGESTION || item.type === FeedbackType.STYLE);

const [isDiscussionOpen, setIsDiscussionOpen] = useState(false);

const isDiscussing = discussionState?.feedbackId === item.id;

return React.createElement("div", { style: { padding: '1rem', borderRadius: '0.5rem', border: `1px solid ${borderColor}`, backgroundColor: bgColor, boxShadow: '0 1px 2px 0 rgba(0, 0, 0, 0.05)', transition: 'all 0.3s' } },

React.createElement("div", { style: { display: 'flex', alignItems: 'center', gap: '0.75rem' } },

React.createElement(Icon, { style: { height: '1.25rem', width: '1.25rem', flexShrink: 0, color: textColor } }),

React.createElement("h3", { style: { fontSize: '1.125rem', fontWeight: '600', color: textColor } }, title),

item.line && React.createElement("span", { style: { marginLeft: 'auto', fontSize: '0.75rem', fontFamily: 'var(--font-mono)', backgroundColor: 'var(--bg-tertiary-color)', color: 'var(--text-secondary-color)', padding: '0.25rem 0.5rem', borderRadius: '0.25rem' } }, `Line: ${item.line}`)

),

React.createElement("p", { style: { marginTop: '0.75rem', color: 'var(--text-secondary-color)' } }, item.description),

item.suggestion && React.createElement("p", { style: { marginTop: '0.75rem', fontSize: '0.875rem', color: 'var(--text-tertiary-color)', fontStyle: 'italic' } }, `"${item.suggestion}"`),

item.refactor && React.createElement("div", { style: { marginTop: '0.75rem', paddingTop: '0.75rem', borderTop: '1px solid var(--border-color)' } },

React.createElement("p", { style: { fontSize: '0.875rem', fontWeight: '600', color: 'var(--text-tertiary-color)', marginBottom: '0.25rem' } }, "Suggested Code:"),

React.createElement("pre", { style: { padding: '0.75rem', backgroundColor: 'rgba(0,0,0,0.3)', borderRadius: '0.375rem', fontSize: '0.875rem', fontFamily: 'var(--font-mono)', color: 'var(--text-color)', overflowX: 'auto' } }, React.createElement("code", null, item.refactor.code))

),

React.createElement("div", { style: { marginTop: '1rem', display: 'flex', alignItems: 'center', gap: '0.75rem' } },

canApplyFix && React.createElement("button", { onClick: () => onApplyFix(item.refactor), style: { display: 'inline-flex', alignItems: 'center', gap: '0.5rem', padding: '0.5rem 1rem' } }, React.createElement(WandIcon, { style: { height: '1rem', width: '1rem' } }), "Apply Fix"),

React.createElement("button", { onClick: () => setIsDiscussionOpen(!isDiscussionOpen), style: { display: 'inline-flex', alignItems: 'center', gap: '0.5rem', padding: '0.5rem 1rem', backgroundColor: 'var(--bg-tertiary-color)' } }, React.createElement(ChatBubbleIcon, { style: { height: '1rem', width: '1rem' } }), isDiscussionOpen ? 'Close' : 'Discuss')

),

(isDiscussionOpen || (isDiscussing && item.discussion && item.discussion.length > 0)) && React.createElement("div", { style: { marginTop: '1rem', paddingTop: '1rem', borderTop: `1px solid var(--border-color)` } }, React.createElement(DiscussionThread, { messages: item.discussion || [], isLoading: isDiscussing && !!discussionState?.isLoading, onSendMessage: (message) => onSendMessage(item.id, message) }))

);

};

const ScoreDonut = ({ score }) => {

const radius = 50; const circumference = 2 \* Math.PI \* radius; const offset = circumference - (score / 100) \* circumference;

let color = score < 50 ? '#ef4444' : score < 80 ? '#f59e0b' : '#22c55e';

return React.createElement("div", { style: { position: 'relative', display: 'flex', alignItems: 'center', justifyContent: 'center', width: '8rem', height: '8rem' } },

React.createElement("svg", { style: { position: 'absolute', transform: 'rotate(-90deg)', width: '100%', height: '100%' }, viewBox: "0 0 120 120" },

React.createElement("circle", { strokeWidth: "10", stroke: "var(--bg-tertiary-color)", fill: "transparent", r: radius, cx: "60", cy: "60" }),

React.createElement("circle", { strokeWidth: "10", strokeDasharray: circumference, strokeDashoffset: offset, strokeLinecap: "round", stroke: color, fill: "transparent", r: radius, cx: "60", cy: "60", style: { transition: 'stroke-dashoffset 1s ease-out' } })

),

React.createElement("span", { style: { fontSize: '2rem', fontWeight: 'bold', color: color } }, score)

);

};

return React.createElement("div", { style: { marginTop: '2rem', display: 'flex', flexDirection: 'column', gap: '2rem' } },

React.createElement("div", { style: { padding: '1.5rem', backgroundColor: 'var(--bg-secondary-color)', borderRadius: '0.75rem', border: '1px solid var(--border-color)' } },

React.createElement("h2", { style: { fontSize: '1.5rem', fontWeight: 'bold', color: 'var(--text-color)', marginBottom: '1rem' } }, "Review Summary"),

React.createElement("div", { style: { display: 'flex', alignItems: 'center', gap: '1.5rem', flexDirection: 'column' } },

React.createElement(ScoreDonut, { score: review.overallScore }),

React.createElement("div", { style: { flex: 1 } }, React.createElement("p", { style: { color: 'var(--text-secondary-color)' } }, review.summary))

)

),

React.createElement("div", { style: { display: 'flex', flexDirection: 'column', gap: '1rem' } },

review.feedbackItems.length > 0

? review.feedbackItems.map((item) => React.createElement(FeedbackCard, { key: item.id, item: item }))

: React.createElement("div", { style: { textAlign: 'center', padding: '2rem 0', color: 'var(--text-tertiary-color)' } }, "No specific feedback items were generated. The code looks good!")

)

);

};

// --- START OF App ---

const App = () => {

const [code, setCode] = useState('');

const [language, setLanguage] = useState('');

const [review, setReview] = useState(null);

const [isLoading, setIsLoading] = useState(false);

const [error, setError] = useState(null);

const [discussionState, setDiscussionState] = useState(null);

const [isSheetContext, setIsSheetContext] = useState(false);

useEffect(() => {

try {

if (typeof google !== 'undefined' && google.script && google.script.run) {

setIsSheetContext(true);

}

} catch (e) {

setIsSheetContext(false);

}

}, []);

const handleReviewRequest = useCallback(async (lang, customInstructions) => {

if (!code.trim()) { setError("Please enter some code to review."); return; }

setIsLoading(true); setError(null); setReview(null); setLanguage(lang);

try {

const result = await reviewCode(code, lang, customInstructions);

setReview(result);

} catch (e) { console.error(e); setError(e instanceof Error ? `An error occurred: ${e.message}. Check the console for details.` : "An unknown error occurred."); }

finally { setIsLoading(false); }

}, [code]);

const handleApplyFix = useCallback((refactor) => {

const lines = code.split('\n');

const { startLine, endLine, code: newCode } = refactor;

const startIndex = startLine - 1;

let newLines;

if (endLine < startLine) { newLines = [...lines.slice(0, startIndex), ...newCode.split('\n'), ...lines.slice(startIndex)]; }

else { const count = endLine - startLine + 1; const newCodeLines = newCode ? newCode.split('\n') : []; newLines = [...lines.slice(0, startIndex), ...newCodeLines, ...lines.slice(startIndex + count)]; }

const newCodeString = newLines.join('\n');

setCode(newCodeString);

if (isSheetContext) { google.script.run.withFailureHandler((err) => { console.error(`Error auto-updating Sheet after applying fix: ${err.message}`); }).updateCodeInActiveCell(newCodeString); }

}, [code, isSheetContext]);

const handleSendMessage = useCallback(async (feedbackId, message) => {

if (!review || !language) return;

const itemToDiscuss = review.feedbackItems.find(item => item.id === feedbackId); if (!itemToDiscuss) return;

setDiscussionState({ feedbackId, isLoading: true });

const updatedReview = { ...review, feedbackItems: review.feedbackItems.map(item => item.id === feedbackId ? { ...item, discussion: [...(item.discussion || []), { author: 'user', message }] } : item) };

setReview(updatedReview);

try {

const geminiResponse = await continueDiscussion(code, language, itemToDiscuss, message);

setReview(prevReview => { if (!prevReview) return null; return { ...prevReview, feedbackItems: prevReview.feedbackItems.map(item => item.id === feedbackId ? { ...item, discussion: [...(item.discussion || []), { author: 'gemini', message: geminiResponse }] } : item) }; });

} catch (e) { console.error("Error during discussion:", e); setError("Sorry, I couldn't get a response. Please try again."); setReview(review); }

finally { setDiscussionState(null); }

}, [review, code, language]);

const handleFetchCodeFromSheet = useCallback(() => {

if (!isSheetContext) return;

setIsLoading(true); setError(null);

google.script.run.withSuccessHandler((fetchedCode) => { setCode(fetchedCode || ''); setIsLoading(false); }).withFailureHandler((err) => { setError(`Error fetching from Sheet: ${err.message}`); setIsLoading(false); }).getCodeFromActiveCell();

}, [isSheetContext]);

const handleUpdateSheetWithCode = useCallback(() => {

if (!isSheetContext) return;

setIsLoading(true); setError(null);

google.script.run.withSuccessHandler(() => { setIsLoading(false); }).withFailureHandler((err) => { setError(`Error updating Sheet: ${err.message}`); setIsLoading(false); }).updateCodeInActiveCell(code);

}, [isSheetContext, code]);

return React.createElement("div", { style: { minHeight: '100vh' } },

React.createElement(Header, null),

React.createElement("main", { className: 'container' },

React.createElement("div", { style: { backgroundColor: 'var(--bg-secondary-color)', borderRadius: '1rem', boxShadow: '0 10px 15px -3px rgba(0,0,0,0.1), 0 4px 6px -2px rgba(0,0,0,0.05)', padding: '1.5rem', border: '1px solid var(--border-color)' } },

React.createElement(GoogleSheetIntegration, { isSheetContext: isSheetContext, isLoading: isLoading, onFetchCode: handleFetchCodeFromSheet, onUpdateCode: handleUpdateSheetWithCode }),

React.createElement(CodeInputForm, { code: code, setCode: setCode, onReview: handleReviewRequest, isLoading: isLoading }),

error && React.createElement("div", { style: { marginTop: '1.5rem', padding: '1rem', backgroundColor: 'var(--error-bg-color)', border: `1px solid var(--error-border-color)`, color: 'var(--error-color)', borderRadius: '0.5rem' } },

React.createElement("p", { style: { fontWeight: '600' } }, "Error"),

React.createElement("p", null, error)

),

React.createElement(ReviewOutput, { review: review, isLoading: isLoading, onApplyFix: handleApplyFix, discussionState: discussionState, onSendMessage: handleSendMessage })

)

),

React.createElement("footer", { style: { textAlign: 'center', padding: '1rem', color: 'var(--text-tertiary-color)', fontSize: '0.875rem' } },

React.createElement("p", null, "Powered by Gemini.")

)

);

};

// --- FINAL RENDER ---

const rootElement = document.getElementById('root');

if (!rootElement) throw new Error("Could not find root element");

const root = ReactDOM.createRoot(rootElement);

root.render(React.createElement(React.StrictMode, null, React.createElement(App, null)));

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<base target="\_top">

<meta charset="UTF-8" />

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

<title>Gemini Code Reviewer</title>

<style>

:root {

--bg-color: #111827;

--bg-secondary-color: #1f2937;

--bg-tertiary-color: #374151;

--border-color: #4b5563;

--text-color: #f3f4f6;

--text-secondary-color: #d1d5db;

--text-tertiary-color: #9ca3af;

--primary-color: #4f46e5;

--primary-hover-color: #4338ca;

--primary-disabled-color: #312e81;

--error-color: #be123c;

--error-bg-color: rgba(190, 18, 60, 0.2);

--error-border-color: rgba(190, 18, 60, 0.5);

--success-color: #16a34a;

--font-sans: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, sans-serif;

--font-mono: "Courier New", Courier, monospace;

}

\*, \*::before, \*::after { box-sizing: border-box; }

body, html { margin: 0; padding: 0; width: 100%; height: 100%; overflow-x: hidden; }

body {

background-color: var(--bg-color);

color: var(--text-color);

font-family: var(--font-sans);

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

#root { min-height: 100vh; }

.container { max-width: 800px; margin: 0 auto; padding: 1rem; }

button, input, select, textarea {

font-family: inherit;

font-size: 1rem;

border-radius: 0.5rem;

border: 1px solid var(--border-color);

background-color: var(--bg-secondary-color);

color: var(--text-color);

transition: all 0.2s ease-in-out;

}

button {

cursor: pointer;

padding: 0.75rem 1.25rem;

font-weight: 600;

background-color: var(--primary-color);

border-color: transparent;

}

button:hover { background-color: var(--primary-hover-color); }

button:disabled { background-color: var(--primary-disabled-color); color: var(--text-tertiary-color); cursor: not-allowed; }

textarea, input, select { padding: 0.75rem; width: 100%; }

textarea:focus, input:focus, select:focus {

outline: none;

border-color: var(--primary-color);

box-shadow: 0 0 0 2px rgba(79, 70, 229, 0.5);

}

.loading-spinner {

animation: spin 1s linear infinite;

width: 1.25rem;

height: 1.25rem;

border-radius: 50%;

border: 2px solid var(--border-color);

border-top-color: var(--text-color);

}

@keyframes spin {

to { transform: rotate(360deg); }

}

</style>

</head>

<body>

<div id="root"></div>

<script type="importmap">

{

"imports": {

"react": "https://esm.sh/react@19.1.0",

"react-dom/client": "https://esm.sh/react-dom@19.1.0/client",

"@google/genai": "https://esm.sh/@google/genai@1.9.0"

}

}

</script>

<script type="module">

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

import ReactDOM from 'react-dom/client';

import { GoogleGenAI, Type } from "@google/genai";

// --- START OF TYPES ---

const FeedbackType = {

BUG: 'BUG',

SUGGESTION: 'SUGGESTION',

PRAISE: 'PRAISE',

STYLE: 'STYLE'

};

// --- START OF CONSTANTS ---

const SUPPORTED\_LANGUAGES = [

'JavaScript', 'TypeScript', 'Python', 'Java', 'Go', 'Rust', 'C++', 'C#', 'HTML', 'CSS', 'SQL', 'Shell'

];

// --- START OF ICONS ---

const CodeIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("polyline", { points: "16 18 22 12 16 6" }), React.createElement("polyline", { points: "8 6 2 12 8 18" }));

const SparklesIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", viewBox: "0 0 20 20", fill: "currentColor", ...props }, React.createElement("path", { fillRule: "evenodd", d: "M10 2a.75.75 0 01.75.75v.263a3.5 3.5 0 014.288 4.288H15.25a.75.75 0 010 1.5h-.263a3.5 3.5 0 01-4.288 4.288v.263a.75.75 0 01-1.5 0v-.263a3.5 3.5 0 01-4.288-4.288H4.75a.75.75 0 010-1.5h.263A3.5 3.5 0 019.25 4.75v-.263A.75.75 0 0110 2zM8.341 8.341a.5.5 0 00-.707.707l.707-.707zm3.318 3.318a.5.5 0 00.707-.707l-.707.707zM5.505 11.293a.5.5 0 00.707.707l-.707-.707zM11.293 5.505a.5.5 0 00-.707-.707l.707.707z", clipRule: "evenodd" }));

const BugIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("path", { d: "M12 20h-4a2 2 0 0 1 -2 -2v-12a2 2 0 0 1 2 -2h12v4" }), React.createElement("path", { d: "M20 12h-4a2 2 0 0 0 -2 2v4a2 2 0 0 0 2 2h4v-8z" }), React.createElement("path", { d: "M16 9h4" }), React.createElement("path", { d: "M12 6h.01" }), React.createElement("path", { d: "M12 12h.01" }));

const LightBulbIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("path", { d: "M9 18h6" }), React.createElement("path", { d: "M10 22h4" }), React.createElement("path", { d: "M9 14a6.04 6.04 0 0 1 6 0" }), React.createElement("path", { d: "M12 2v4" }), React.createElement("path", { d: "M3.5 6.5l2 2" }), React.createElement("path", { d: "M18.5 6.5l-2 2" }));

const PaintBrushIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("path", { d: "M9.5 2.5a1 1 0 0 1 1 1v1a1 1 0 0 1-1 1h-3a1 1 0 0 1-1-1v-1a1 1 0 0 1 1-1h3z" }), React.createElement("path", { d: "M12.5 5.5a1 1 0 0 1 1 1v1a1 1 0 0 1-1 1h-1a1 1 0 0 1-1-1v-1a1 1 0 0 1 1-1h1z" }), React.createElement("path", { d: "M16 3v18" }), React.createElement("path", { d: "M10 3v18" }));

const WandIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("path", { d: "M15 4V2" }), React.createElement("path", { d: "M15 8V6" }), React.createElement("path", { d: "M12.5 6.5L14 5" }), React.createElement("path", { d: "M17 9.5L15.5 8" }), React.createElement("path", { d: "M20 9.5h-2" }), React.createElement("path", { d: "M4 9.5H2" }), React.createElement("path", { d: "m9 13.5 1 1" }), React.createElement("path", { d: "M4.5 18.5 3 20" }), React.createElement("path", { d: "M21 20l-1.5-1.5" }), React.createElement("path", { d: "M18.5 4.5 20 3" }), React.createElement("path", { d: "M9 4.5 7.5 3" }), React.createElement("path", { d: "M15 12.5v-1.5a6 6 0 0 0-6-6h-1.5a6 6 0 0 0-6 6v1.5a6 6 0 0 0 6 6h1.5a6 6 0 0 0 6-6Z" }));

const ChatBubbleIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("path", { d: "M21 15a2 2 0 0 1-2 2H7l-4 4V5a2 2 0 0 1 2-2h14a2 2 0 0 1 2 2z" }));

const DownloadIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("path", { d: "M21 15v4a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2v-4" }), React.createElement("polyline", { points: "7 10 12 15 17 10" }), React.createElement("line", { x1: "12", y1: "15", x2: "12", y2: "3" }));

const UploadIcon = (props) => React.createElement("svg", { xmlns: "http://www.w3.org/2000/svg", width: "24", height: "24", viewBox: "0 0 24 24", fill: "none", stroke: "currentColor", strokeWidth: "2", strokeLinecap: "round", strokeLinejoin: "round", ...props }, React.createElement("path", { d: "M21 15v4a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2v-4" }), React.createElement("polyline", { points: "17 8 12 3 7 8" }), React.createElement("line", { x1: "12", y1: "3", x2: "12", y2: "15" }));

// --- START OF geminiService ---

const ai = new GoogleGenAI({ apiKey: process.env.API\_KEY });

const refactorSchema = { type: Type.OBJECT, properties: { startLine: { type: Type.INTEGER, description: "The 1-based starting line number of the code block to be replaced. Inclusive." }, endLine: { type: Type.INTEGER, description: "The 1-based ending line number of the code block to be replaced. Inclusive. Should be >= startLine." }, code: { type: Type.STRING, description: "The new code to replace the specified lines. For a deletion, this must be an empty string." } }, required: ["startLine", "endLine", "code"] };

const reviewSchema = { type: Type.OBJECT, properties: { overallScore: { type: Type.INTEGER, description: "A score from 0 to 100 representing the overall quality of the code." }, summary: { type: Type.STRING, description: "A brief, high-level summary of the code review findings." }, feedbackItems: { type: Type.ARRAY, description: "A list of specific feedback points about the code.", items: { type: Type.OBJECT, properties: { type: { type: Type.STRING, enum: ['BUG', 'SUGGESTION', 'PRAISE', 'STYLE'], description: "The category of feedback: BUG, SUGGESTION, PRAISE, or STYLE." }, line: { type: Type.INTEGER, description: "The primary line number the feedback applies to. For multi-line changes, this should be the starting line. Null for general comments.", nullable: true }, description: { type: Type.STRING, description: "A clear and concise description of the issue or praise." }, suggestion: { type: Type.STRING, description: "A human-readable explanation of the praise or suggested change. Null if not applicable.", nullable: true }, refactor: { ...refactorSchema, description: "An actionable code refactoring suggestion. Provide this only for BUG, SUGGESTION, or STYLE feedback where a direct code change is applicable. Null otherwise.", nullable: true, } }, required: ["type", "description"] } } }, required: ["overallScore", "summary", "feedbackItems"] };

const reviewCode = async (code, language, customInstructions) => {

const customInstructionsSection = customInstructions.trim() ? `\nThe user has provided the following specific instructions for this review. Pay close attention to them:\n---\n${customInstructions.trim()}\n---\n` : '';

const prompt = `Please perform a thorough code review on the following ${language} code snippet.\nAnalyze it for bugs, style violations, performance issues, and areas for improvement.\nAlso, identify any good practices worth praising.\n${customInstructionsSection}\nFor each feedback item that involves a concrete code change (bugs, suggestions, style fixes), provide a 'refactor' object. This object must contain:\n- 'startLine': The 1-based starting line number of the code to replace.\n- 'endLine': The 1-based ending line number of the code to replace.\n- 'code': The new code that will replace the specified lines.\n\nExamples:\n- To REPLACE lines 5-7 with new code: \`{ "startLine": 5, "endLine": 7, "code": "new replacement code" }\`\n- To DELETE line 10: \`{ "startLine": 10, "endLine": 10, "code": "" }\`\n- To INSERT code BEFORE line 15: \`{ "startLine": 15, "endLine": 14, "code": "code to insert" }\` (Note: endLine is less than startLine for insertion)\n\nThe 'suggestion' field should always contain a human-readable explanation of the change, not the code itself.\n\nCode to review:\n\`\`\`${language.toLowerCase()}\n${code}\n\`\`\``;

try {

const response = await ai.models.generateContent({ model: "gemini-2.5-flash", contents: prompt, config: { systemInstruction: "You are an expert code reviewer. Your analysis must be insightful, accurate, and constructive. You must respond ONLY with a JSON object that adheres to the provided schema.", responseMimeType: "application/json", responseSchema: reviewSchema } });

const text = response.text.trim(); if (!text) { throw new Error("Received an empty response from the API."); }

const parsedResult = JSON.parse(text);

parsedResult.feedbackItems.forEach(item => { item.id = crypto.randomUUID(); });

return parsedResult;

} catch (error) { console.error("Error calling Gemini API:", error); if (error instanceof Error && error.message.includes('SAFETY')) { throw new Error("The code could not be reviewed due to safety restrictions. Please ensure the code doesn't violate safety policies."); } throw new Error(`Failed to get review from Gemini API. ${error instanceof Error ? error.message : ''}`); }

};

const continueDiscussion = async (code, language, feedbackItem, question) => {

const history = (feedbackItem.discussion || []).map(msg => `${msg.author === 'user' ? 'User' : 'Assistant'}: ${msg.message}`).join('\n');

const prompt = `You are an expert code reviewer in an ongoing discussion with a user about a specific piece of feedback you provided.\nYour tone should be helpful, clarifying, and concise.\nThis was the original code snippet under review (${language}):\n\`\`\`${language.toLowerCase()}\n${code}\n\`\`\`\nThis was your original feedback for the code around line ${feedbackItem.line}:\n- Category: ${feedbackItem.type}\n- Description: ${feedbackItem.description}\n- Your Suggestion: ${feedbackItem.suggestion || 'N/A'}\nSo far, the conversation has been:\n${history.length > 0 ? history : "No previous discussion."}\nThe user has a new question for you:\nUser: ${question}\nYour task is to provide a direct and helpful answer to the user's question.\nFocus only on providing your next response. Do not repeat the context I just gave you.`;

try {

const response = await ai.models.generateContent({ model: "gemini-2.5-flash", contents: prompt, });

const text = response.text.trim(); if (!text) { throw new Error("Received an empty follow-up response from the API."); } return text;

} catch (error) { console.error("Error calling Gemini API for discussion:", error); throw new Error(`Failed to get discussion response from Gemini API. ${error instanceof Error ? error.message : ''}`); }

};

// --- START OF COMPONENTS ---

const Header = () => React.createElement("header", { style: { padding: '1.5rem 0', borderBottom: '1px solid var(--border-color)', backgroundColor: 'rgba(17, 24, 39, 0.3)', backdropFilter: 'blur(4px)', position: 'sticky', top: 0, zIndex: 10 } }, React.createElement("div", { className: 'container', style: { display: 'flex', alignItems: 'center', gap: '1rem' } }, React.createElement("div", { style: { backgroundColor: 'var(--primary-color)', padding: '0.5rem', borderRadius: '0.5rem', boxShadow: '0 4px 6px -1px rgba(79, 70, 229, 0.3), 0 2px 4px -2px rgba(79, 70, 229, 0.3)' } }, React.createElement(CodeIcon, { style: { height: '1.5rem', width: '1.5rem', color: 'white' } })), React.createElement("h1", { style: { fontSize: '1.5rem', fontWeight: 'bold', color: 'var(--text-color)', letterSpacing: '-0.025em' } }, "Gemini Code Reviewer")));

const GoogleSheetIntegration = ({ onFetchCode, onUpdateCode, isSheetContext, isLoading }) => {

if (!isSheetContext) return null;

return React.createElement("div", { style: { marginBottom: '1.5rem', padding: '1rem', backgroundColor: 'rgba(16, 185, 129, 0.1)', border: '1px solid rgba(5, 150, 105, 0.3)', borderRadius: '0.5rem' } },

React.createElement("h3", { style: { fontSize: '1.125rem', fontWeight: '600', color: '#6ee7b7', marginBottom: '0.75rem' } }, "Google Sheets Integration"),

React.createElement("p", { style: { fontSize: '0.875rem', color: 'var(--text-secondary-color)', marginBottom: '1rem' } }, "Use these buttons to sync code with your active cell. Applying a fix will automatically update the cell."),

React.createElement("div", { style: { display: 'flex', gap: '1rem', flexDirection: 'row' } },

React.createElement("button", { onClick: onFetchCode, disabled: isLoading, style: { flex: 1, display: 'inline-flex', alignItems: 'center', justifyContent: 'center', gap: '0.5rem', backgroundColor: 'var(--bg-tertiary-color)' } }, React.createElement(DownloadIcon, { style: { height: '1rem', width: '1rem' } }), "Get Code from Cell"),

React.createElement("button", { onClick: onUpdateCode, disabled: isLoading, style: { flex: 1, display: 'inline-flex', alignItems: 'center', justifyContent: 'center', gap: '0.5rem', backgroundColor: 'var(--bg-tertiary-color)' } }, React.createElement(UploadIcon, { style: { height: '1rem', width: '1rem' } }), "Update Cell with Code")

)

);

};

const CodeInputForm = ({ code, setCode, onReview, isLoading }) => {

const [language, setLanguage] = useState(SUPPORTED\_LANGUAGES[0]);

const [customInstructions, setCustomInstructions] = useState('');

const handleSubmit = (e) => { e.preventDefault(); onReview(language, customInstructions); };

return React.createElement("form", { onSubmit: handleSubmit, style: { display: 'flex', flexDirection: 'column', gap: '1.5rem' } },

React.createElement("div", { style: { display: 'flex', gap: '1rem', flexDirection: 'column' } },

React.createElement("div", { style: { flexGrow: 1 } },

React.createElement("label", { htmlFor: "code-input", style: { display: 'block', fontSize: '0.875rem', fontWeight: '500', color: 'var(--text-secondary-color)', marginBottom: '0.5rem' } }, "Your Code"),

React.createElement("textarea", { id: "code-input", value: code, onChange: (e) => setCode(e.target.value), placeholder: `// Paste your ${language} code here...`, style: { width: '100%', height: '20rem', padding: '1rem', fontFamily: 'var(--font-mono)', fontSize: '0.875rem', backgroundColor: 'var(--bg-color)', resize: 'vertical' }, spellCheck: "false", disabled: isLoading })

),

React.createElement("div", { style: { width: '100%' } },

React.createElement("label", { htmlFor: "language-select", style: { display: 'block', fontSize: '0.875rem', fontWeight: '500', color: 'var(--text-secondary-color)', marginBottom: '0.5rem' } }, "Language"),

React.createElement("select", { id: "language-select", value: language, onChange: (e) => setLanguage(e.target.value), disabled: isLoading }, SUPPORTED\_LANGUAGES.map((lang) => React.createElement("option", { key: lang, value: lang }, lang)))

)

),

React.createElement("div", null,

React.createElement("label", { htmlFor: "custom-instructions", style: { display: 'block', fontSize: '0.875rem', fontWeight: '500', color: 'var(--text-secondary-color)', marginBottom: '0.5rem' } }, "Custom Instructions ", React.createElement("span", { style: { color: 'var(--text-tertiary-color)' } }, "(Optional)")),

React.createElement("textarea", { id: "custom-instructions", value: customInstructions, onChange: (e) => setCustomInstructions(e.target.value), placeholder: "e.g., Focus on security and performance...", style: { width: '100%', height: '6rem', padding: '1rem', fontFamily: 'var(--font-sans)', resize: 'vertical' }, disabled: isLoading })

),

React.createElement("div", null,

React.createElement("button", { type: "submit", disabled: isLoading || !code.trim(), style: { width: '100%', display: 'flex', alignItems: 'center', justifyContent: 'center', gap: '0.5rem' } },

isLoading ? React.createElement(React.Fragment, null, React.createElement("div", { className: 'loading-spinner' }), "Reviewing...") : React.createElement(React.Fragment, null, React.createElement(SparklesIcon, { style: { height: '1.25rem', width: '1.25rem' } }), "Review Code")

)

)

);

};

const DiscussionThread = ({ messages, isLoading, onSendMessage }) => {

const [input, setInput] = useState('');

const endOfMessagesRef = useRef(null);

const handleSubmit = (e) => { e.preventDefault(); if (input.trim() && !isLoading) { onSendMessage(input.trim()); setInput(''); } };

useEffect(() => { endOfMessagesRef.current?.scrollIntoView({ behavior: 'smooth' }); }, [messages, isLoading]);

const ChatBubble = ({ message }) => {

const isUser = message.author === 'user';

return React.createElement("div", { style: { display: 'flex', alignItems: 'flex-start', gap: '0.625rem', justifyContent: isUser ? 'flex-end' : 'flex-start' } },

!isUser && React.createElement("div", { style: { flexShrink: 0, width: '2rem', height: '2rem', borderRadius: '50%', background: 'linear-gradient(to bottom right, #6366f1, #a855f7)', display: 'flex', alignItems: 'center', justifyContent: 'center' } }, React.createElement(SparklesIcon, { style: { width: '1.25rem', height: '1.25rem', color: '#e0e7ff' } })),

React.createElement("div", { style: { display: 'flex', flexDirection: 'column', gap: '0.25rem', width: '100%', maxWidth: '320px', lineHeight: 1.5, padding: '0.75rem', borderRadius: '0.75rem', backgroundColor: isUser ? 'var(--primary-color)' : 'var(--bg-tertiary-color)', borderBottomRightRadius: isUser ? 0 : '0.75rem', borderBottomLeftRadius: isUser ? '0.75rem' : 0 } }, React.createElement("p", { style: { fontSize: '0.875rem', fontWeight: '400', color: 'white' } }, message.message)),

isUser && React.createElement("div", { style: { flexShrink: 0, width: '2rem', height: '2rem', borderRadius: '50%', backgroundColor: 'var(--bg-tertiary-color)', display: 'flex', alignItems: 'center', justifyContent: 'center' } }, React.createElement("svg", { style: { width: '1.25rem', height: '1.25rem', color: 'var(--text-secondary-color)' }, fill: "currentColor", viewBox: "0 0 20 20", xmlns: "http://www.w3.org/2000/svg" }, React.createElement("path", { fillRule: "evenodd", d: "M10 9a3 3 0 100-6 3 3 0 000 6zm-7 9a7 7 0 1114 0H3z", clipRule: "evenodd" })))

);

};

return React.createElement("div", { style: { display: 'flex', flexDirection: 'column', gap: '1rem' } },

React.createElement("div", { style: { display: 'flex', flexDirection: 'column', gap: '1rem', maxHeight: '16rem', overflowY: 'auto', paddingRight: '0.5rem' } },

messages.map((msg, index) => React.createElement(ChatBubble, { key: index, message: msg })),

isLoading && React.createElement("div", { style: { display: 'flex', alignItems: 'flex-start', gap: '0.625rem' } },

React.createElement("div", { style: { flexShrink: 0, width: '2rem', height: '2rem', borderRadius: '50%', background: 'linear-gradient(to bottom right, #6366f1, #a855f7)', display: 'flex', alignItems: 'center', justifyContent: 'center' } }, React.createElement(SparklesIcon, { style: { width: '1.25rem', height: '1.25rem', color: '#e0e7ff', animation: 'pulse 1.5s infinite' } })),

React.createElement("div", { style: { display: 'flex', flexDirection: 'column', gap: '0.25rem', width: '100%', maxWidth: '320px', lineHeight: 1.5, padding: '0.75rem', borderRadius: '0.75rem', backgroundColor: 'var(--bg-tertiary-color)', borderBottomLeftRadius: 0 } }, React.createElement("p", { style: { fontSize: '0.875rem', fontStyle: 'italic', color: 'var(--text-tertiary-color)' } }, "Gemini is thinking..."))

),

React.createElement("div", { ref: endOfMessagesRef })

),

React.createElement("form", { onSubmit: handleSubmit, style: { display: 'flex', alignItems: 'center', gap: '0.5rem' } },

React.createElement("input", { type: "text", value: input, onChange: (e) => setInput(e.target.value), placeholder: "Ask a follow-up question...", disabled: isLoading, style: { flexGrow: 1 } }),

React.createElement("button", { type: "submit", disabled: isLoading || !input.trim() }, "Send")

)

);

};

const ReviewOutput = ({ review, isLoading, onApplyFix, discussionState, onSendMessage }) => {

if (isLoading) return React.createElement("div", { style: { marginTop: '1.5rem', textAlign: 'center', color: 'var(--text-tertiary-color)' } },

React.createElement("p", null, "Analyzing your code..."),

React.createElement("div", { style: { marginTop: '1rem', backgroundColor: 'rgba(55, 65, 81, 0.5)', borderRadius: '0.5rem', padding: '1.5rem', display: 'flex', flexDirection: 'column', gap: '1rem', animation: 'pulse 2s cubic-bezier(0.4, 0, 0.6, 1) infinite' } },

React.createElement("div", { style: { height: '1rem', backgroundColor: 'var(--bg-tertiary-color)', borderRadius: '0.25rem', width: '25%' } }),

React.createElement("div", { style: { height: '2rem', backgroundColor: 'var(--bg-tertiary-color)', borderRadius: '0.25rem', width: '33%' } }),

React.createElement("div", { style: { display: 'flex', flexDirection: 'column', gap: '0.5rem' } },

React.createElement("div", { style: { height: '1rem', backgroundColor: 'var(--bg-tertiary-color)', borderRadius: '0.25rem', width: '100%' } }),

React.createElement("div", { style: { height: '1rem', backgroundColor: 'var(--bg-tertiary-color)', borderRadius: '0.25rem', width: '83.33%' } })

)

)

);

if (!review) return React.createElement("div", { style: { marginTop: '2rem', padding: '3rem 0', textAlign: 'center', border: '2px dashed var(--border-color)', borderRadius: '0.5rem' } }, React.createElement("p", { style: { color: 'var(--text-tertiary-color)' } }, "Your code review will appear here."));

const getFeedbackTypeStyles = (type) => {

switch (type) {

case FeedbackType.BUG: return { Icon: BugIcon, bgColor: 'rgba(225, 29, 72, 0.15)', borderColor: 'rgba(225, 29, 72, 0.4)', textColor: '#fecdd3', title: 'Bug Detected' };

case FeedbackType.SUGGESTION: return { Icon: LightBulbIcon, bgColor: 'rgba(217, 119, 6, 0.15)', borderColor: 'rgba(217, 119, 6, 0.4)', textColor: '#fde68a', title: 'Suggestion' };

case FeedbackType.STYLE: return { Icon: PaintBrushIcon, bgColor: 'rgba(59, 130, 246, 0.15)', borderColor: 'rgba(59, 130, 246, 0.4)', textColor: '#bfdbfe', title: 'Style Suggestion' };

case FeedbackType.PRAISE: return { Icon: SparklesIcon, bgColor: 'rgba(22, 163, 74, 0.15)', borderColor: 'rgba(22, 163, 74, 0.4)', textColor: '#bbf7d0', title: 'Good Practice' };

default: return { Icon: LightBulbIcon, bgColor: 'var(--bg-tertiary-color)', borderColor: 'var(--border-color)', textColor: 'var(--text-secondary-color)', title: 'Feedback' };

}

};

const FeedbackCard = ({ item }) => {

const { Icon, bgColor, borderColor, textColor, title } = getFeedbackTypeStyles(item.type);

const canApplyFix = item.refactor && (item.type === FeedbackType.BUG || item.type === FeedbackType.SUGGESTION || item.type === FeedbackType.STYLE);

const [isDiscussionOpen, setIsDiscussionOpen] = useState(false);

const isDiscussing = discussionState?.feedbackId === item.id;

return React.createElement("div", { style: { padding: '1rem', borderRadius: '0.5rem', border: `1px solid ${borderColor}`, backgroundColor: bgColor, boxShadow: '0 1px 2px 0 rgba(0, 0, 0, 0.05)', transition: 'all 0.3s' } },

React.createElement("div", { style: { display: 'flex', alignItems: 'center', gap: '0.75rem' } },

React.createElement(Icon, { style: { height: '1.25rem', width: '1.25rem', flexShrink: 0, color: textColor } }),

React.createElement("h3", { style: { fontSize: '1.125rem', fontWeight: '600', color: textColor } }, title),

item.line && React.createElement("span", { style: { marginLeft: 'auto', fontSize: '0.75rem', fontFamily: 'var(--font-mono)', backgroundColor: 'var(--bg-tertiary-color)', color: 'var(--text-secondary-color)', padding: '0.25rem 0.5rem', borderRadius: '0.25rem' } }, `Line: ${item.line}`)

),

React.createElement("p", { style: { marginTop: '0.75rem', color: 'var(--text-secondary-color)' } }, item.description),

item.suggestion && React.createElement("p", { style: { marginTop: '0.75rem', fontSize: '0.875rem', color: 'var(--text-tertiary-color)', fontStyle: 'italic' } }, `"${item.suggestion}"`),

item.refactor && React.createElement("div", { style: { marginTop: '0.75rem', paddingTop: '0.75rem', borderTop: '1px solid var(--border-color)' } },

React.createElement("p", { style: { fontSize: '0.875rem', fontWeight: '600', color: 'var(--text-tertiary-color)', marginBottom: '0.25rem' } }, "Suggested Code:"),

React.createElement("pre", { style: { padding: '0.75rem', backgroundColor: 'rgba(0,0,0,0.3)', borderRadius: '0.375rem', fontSize: '0.875rem', fontFamily: 'var(--font-mono)', color: 'var(--text-color)', overflowX: 'auto' } }, React.createElement("code", null, item.refactor.code))

),

React.createElement("div", { style: { marginTop: '1rem', display: 'flex', alignItems: 'center', gap: '0.75rem' } },

canApplyFix && React.createElement("button", { onClick: () => onApplyFix(item.refactor), style: { display: 'inline-flex', alignItems: 'center', gap: '0.5rem', padding: '0.5rem 1rem' } }, React.createElement(WandIcon, { style: { height: '1rem', width: '1rem' } }), "Apply Fix"),

React.createElement("button", { onClick: () => setIsDiscussionOpen(!isDiscussionOpen), style: { display: 'inline-flex', alignItems: 'center', gap: '0.5rem', padding: '0.5rem 1rem', backgroundColor: 'var(--bg-tertiary-color)' } }, React.createElement(ChatBubbleIcon, { style: { height: '1rem', width: '1rem' } }), isDiscussionOpen ? 'Close' : 'Discuss')

),

(isDiscussionOpen || (isDiscussing && item.discussion && item.discussion.length > 0)) && React.createElement("div", { style: { marginTop: '1rem', paddingTop: '1rem', borderTop: `1px solid var(--border-color)` } }, React.createElement(DiscussionThread, { messages: item.discussion || [], isLoading: isDiscussing && !!discussionState?.isLoading, onSendMessage: (message) => onSendMessage(item.id, message) }))

);

};

const ScoreDonut = ({ score }) => {

const radius = 50; const circumference = 2 \* Math.PI \* radius; const offset = circumference - (score / 100) \* circumference;

let color = score < 50 ? '#ef4444' : score < 80 ? '#f59e0b' : '#22c55e';

return React.createElement("div", { style: { position: 'relative', display: 'flex', alignItems: 'center', justifyContent: 'center', width: '8rem', height: '8rem' } },

React.createElement("svg", { style: { position: 'absolute', transform: 'rotate(-90deg)', width: '100%', height: '100%' }, viewBox: "0 0 120 120" },

React.createElement("circle", { strokeWidth: "10", stroke: "var(--bg-tertiary-color)", fill: "transparent", r: radius, cx: "60", cy: "60" }),

React.createElement("circle", { strokeWidth: "10", strokeDasharray: circumference, strokeDashoffset: offset, strokeLinecap: "round", stroke: color, fill: "transparent", r: radius, cx: "60", cy: "60", style: { transition: 'stroke-dashoffset 1s ease-out' } })

),

React.createElement("span", { style: { fontSize: '2rem', fontWeight: 'bold', color: color } }, score)

);

};

return React.createElement("div", { style: { marginTop: '2rem', display: 'flex', flexDirection: 'column', gap: '2rem' } },

React.createElement("div", { style: { padding: '1.5rem', backgroundColor: 'var(--bg-secondary-color)', borderRadius: '0.75rem', border: '1px solid var(--border-color)' } },

React.createElement("h2", { style: { fontSize: '1.5rem', fontWeight: 'bold', color: 'var(--text-color)', marginBottom: '1rem' } }, "Review Summary"),

React.createElement("div", { style: { display: 'flex', alignItems: 'center', gap: '1.5rem', flexDirection: 'column' } },

React.createElement(ScoreDonut, { score: review.overallScore }),

React.createElement("div", { style: { flex: 1 } }, React.createElement("p", { style: { color: 'var(--text-secondary-color)' } }, review.summary))

)

),

React.createElement("div", { style: { display: 'flex', flexDirection: 'column', gap: '1rem' } },

review.feedbackItems.length > 0

? review.feedbackItems.map((item) => React.createElement(FeedbackCard, { key: item.id, item: item }))

: React.createElement("div", { style: { textAlign: 'center', padding: '2rem 0', color: 'var(--text-tertiary-color)' } }, "No specific feedback items were generated. The code looks good!")

)

);

};

// --- START OF App ---

const App = () => {

const [code, setCode] = useState('');

const [language, setLanguage] = useState('');

const [review, setReview] = useState(null);

const [isLoading, setIsLoading] = useState(false);

const [error, setError] = useState(null);

const [discussionState, setDiscussionState] = useState(null);

const [isSheetContext, setIsSheetContext] = useState(false);

useEffect(() => {

try {

if (typeof google !== 'undefined' && google.script && google.script.run) {

setIsSheetContext(true);

}

} catch (e) {

setIsSheetContext(false);

}

}, []);

const handleReviewRequest = useCallback(async (lang, customInstructions) => {

if (!code.trim()) { setError("Please enter some code to review."); return; }

setIsLoading(true); setError(null); setReview(null); setLanguage(lang);

try {

const result = await reviewCode(code, lang, customInstructions);

setReview(result);

} catch (e) { console.error(e); setError(e instanceof Error ? `An error occurred: ${e.message}. Check the console for details.` : "An unknown error occurred."); }

finally { setIsLoading(false); }

}, [code]);

const handleApplyFix = useCallback((refactor) => {

const lines = code.split('\n');

const { startLine, endLine, code: newCode } = refactor;

const startIndex = startLine - 1;

let newLines;

if (endLine < startLine) { newLines = [...lines.slice(0, startIndex), ...newCode.split('\n'), ...lines.slice(startIndex)]; }

else { const count = endLine - startLine + 1; const newCodeLines = newCode ? newCode.split('\n') : []; newLines = [...lines.slice(0, startIndex), ...newCodeLines, ...lines.slice(startIndex + count)]; }

const newCodeString = newLines.join('\n');

setCode(newCodeString);

if (isSheetContext) { google.script.run.withFailureHandler((err) => { console.error(`Error auto-updating Sheet after applying fix: ${err.message}`); }).updateCodeInActiveCell(newCodeString); }

}, [code, isSheetContext]);

const handleSendMessage = useCallback(async (feedbackId, message) => {

if (!review || !language) return;

const itemToDiscuss = review.feedbackItems.find(item => item.id === feedbackId); if (!itemToDiscuss) return;

setDiscussionState({ feedbackId, isLoading: true });

const updatedReview = { ...review, feedbackItems: review.feedbackItems.map(item => item.id === feedbackId ? { ...item, discussion: [...(item.discussion || []), { author: 'user', message }] } : item) };

setReview(updatedReview);

try {

const geminiResponse = await continueDiscussion(code, language, itemToDiscuss, message);

setReview(prevReview => { if (!prevReview) return null; return { ...prevReview, feedbackItems: prevReview.feedbackItems.map(item => item.id === feedbackId ? { ...item, discussion: [...(item.discussion || []), { author: 'gemini', message: geminiResponse }] } : item) }; });

} catch (e) { console.error("Error during discussion:", e); setError("Sorry, I couldn't get a response. Please try again."); setReview(review); }

finally { setDiscussionState(null); }

}, [review, code, language]);

const handleFetchCodeFromSheet = useCallback(() => {

if (!isSheetContext) return;

setIsLoading(true); setError(null);

google.script.run.withSuccessHandler((fetchedCode) => { setCode(fetchedCode || ''); setIsLoading(false); }).withFailureHandler((err) => { setError(`Error fetching from Sheet: ${err.message}`); setIsLoading(false); }).getCodeFromActiveCell();

}, [isSheetContext]);

const handleUpdateSheetWithCode = useCallback(() => {

if (!isSheetContext) return;

setIsLoading(true); setError(null);

google.script.run.withSuccessHandler(() => { setIsLoading(false); }).withFailureHandler((err) => { setError(`Error updating Sheet: ${err.message}`); setIsLoading(false); }).updateCodeInActiveCell(code);

}, [isSheetContext, code]);

return React.createElement("div", { style: { minHeight: '100vh' } },

React.createElement(Header, null),

React.createElement("main", { className: 'container' },

React.createElement("div", { style: { backgroundColor: 'var(--bg-secondary-color)', borderRadius: '1rem', boxShadow: '0 10px 15px -3px rgba(0,0,0,0.1), 0 4px 6px -2px rgba(0,0,0,0.05)', padding: '1.5rem', border: '1px solid var(--border-color)' } },

React.createElement(GoogleSheetIntegration, { isSheetContext: isSheetContext, isLoading: isLoading, onFetchCode: handleFetchCodeFromSheet, onUpdateCode: handleUpdateSheetWithCode }),

React.createElement(CodeInputForm, { code: code, setCode: setCode, onReview: handleReviewRequest, isLoading: isLoading }),

error && React.createElement("div", { style: { marginTop: '1.5rem', padding: '1rem', backgroundColor: 'var(--error-bg-color)', border: `1px solid var(--error-border-color)`, color: 'var(--error-color)', borderRadius: '0.5rem' } },

React.createElement("p", { style: { fontWeight: '600' } }, "Error"),

React.createElement("p", null, error)

),

React.createElement(ReviewOutput, { review: review, isLoading: isLoading, onApplyFix: handleApplyFix, discussionState: discussionState, onSendMessage: handleSendMessage })

)

),

React.createElement("footer", { style: { textAlign: 'center', padding: '1rem', color: 'var(--text-tertiary-color)', fontSize: '0.875rem' } },

React.createElement("p", null, "Powered by Gemini.")

)

);

};

// --- FINAL RENDER ---

const rootElement = document.getElementById('root');

if (!rootElement) throw new Error("Could not find root element");

const root = ReactDOM.createRoot(rootElement);

root.render(React.createElement(React.StrictMode, null, React.createElement(App, null)));

</script>

</body>

</html>