<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8">

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

<title>AI 스토리보드 생성기</title>

<link href="https://fonts.googleapis.com/css2?family=Jua&display=swap" rel="stylesheet">

<style>

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

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

background: linear-gradient(135deg, #1a1a1a 0%, #2d2d2d 100%);

color: #f0f0f0;

min-height: 100vh;

display: flex;

overflow-x: hidden;

}

/\* 사이드바 스타일 \*/

.sidebar {

width: 450px;

background: rgba(30, 30, 30, 0.95);

padding: 30px;

border-right: 1px solid #444;

overflow-y: auto;

height: 100vh;

position: fixed;

left: 0;

top: 0;

z-index: 10;

animation: slideIn 0.5s ease;

}

@keyframes slideIn {

from {

transform: translateX(-100%);

}

to {

transform: translateX(0);

}

}

.sidebar h1 {

font-size: 28px;

margin-bottom: 30px;

background: linear-gradient(45deg, #ff6b6b, #feca57);

-webkit-background-clip: text;

-webkit-text-fill-color: transparent;

text-align: center;

}

/\* 입력 필드 스타일 \*/

.input-group {

margin-bottom: 25px;

animation: fadeIn 0.6s ease;

}

.input-group label {

display: block;

margin-bottom: 8px;

font-size: 14px;

color: #b0b0b0;

}

.input-group input,

.input-group textarea,

.input-group select {

width: 100%;

padding: 12px;

background: #2d2d2d;

border: 1px solid #444;

border-radius: 8px;

color: #f0f0f0;

font-size: 14px;

transition: all 0.3s ease;

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

}

.input-group input:focus,

.input-group textarea:focus,

.input-group select:focus {

outline: none;

background: #353535;

border-color: #ff6b6b;

box-shadow: 0 0 0 2px rgba(255, 107, 107, 0.1);

}

textarea {

resize: vertical;

min-height: 60px;

}

/\* 파일 업로드 스타일 \*/

.file-upload {

position: relative;

display: inline-block;

width: 100%;

}

.file-upload input[type="file"] {

position: absolute;

opacity: 0;

width: 100%;

height: 100%;

cursor: pointer;

}

.file-upload-label {

display: block;

padding: 12px;

background: #2d2d2d;

border: 2px dashed #444;

border-radius: 8px;

text-align: center;

cursor: pointer;

transition: all 0.3s ease;

}

.file-upload-label:hover {

background: #353535;

border-color: #ff6b6b;

}

.image-preview {

margin-top: 15px;

max-width: 100%;

border-radius: 8px;

overflow: hidden;

animation: fadeIn 0.5s ease;

}

.image-preview img {

width: 100%;

height: auto;

display: block;

}

.image-info {

margin-top: 10px;

padding: 10px;

background: rgba(255, 107, 107, 0.1);

border-radius: 6px;

font-size: 12px;

color: #a0a0a0;

}

/\* 시나리오 입력 섹션 \*/

.scenario-section {

margin-top: 30px;

padding-top: 20px;

border-top: 1px solid #444;

}

.scenario-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 15px;

}

.scenario-title {

font-size: 18px;

color: #feca57;

}

.add-scene-btn {

padding: 8px 16px;

background: #3d3d3d;

border: 1px solid #555;

border-radius: 6px;

color: #f0f0f0;

cursor: pointer;

transition: all 0.3s ease;

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

}

.add-scene-btn:hover {

background: #4d4d4d;

border-color: #feca57;

}

.scene-item {

background: #252525;

border-radius: 8px;

padding: 15px;

margin-bottom: 15px;

position: relative;

animation: fadeInUp 0.3s ease;

}

@keyframes fadeInUp {

from {

opacity: 0;

transform: translateY(10px);

}

to {

opacity: 1;

transform: translateY(0);

}

}

.scene-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 10px;

}

.scene-number {

background: linear-gradient(45deg, #ff6b6b, #feca57);

color: white;

padding: 4px 10px;

border-radius: 20px;

font-size: 12px;

}

.remove-scene-btn {

background: none;

border: none;

color: #ff6b6b;

cursor: pointer;

font-size: 20px;

transition: transform 0.3s ease;

}

.remove-scene-btn:hover {

transform: scale(1.2);

}

.scene-description {

width: 100%;

padding: 10px;

background: #2d2d2d;

border: 1px solid #3d3d3d;

border-radius: 6px;

color: #f0f0f0;

font-size: 13px;

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

resize: vertical;

min-height: 60px;

}

/\* 버튼 스타일 \*/

.btn {

padding: 14px 28px;

background: linear-gradient(45deg, #ff6b6b, #feca57);

border: none;

border-radius: 8px;

color: white;

font-size: 16px;

font-weight: bold;

cursor: pointer;

transition: all 0.3s ease;

position: relative;

overflow: hidden;

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

width: 100%;

margin-bottom: 10px;

}

.btn::before {

content: '';

position: absolute;

top: 0;

left: -100%;

width: 100%;

height: 100%;

background: linear-gradient(90deg, transparent, rgba(255,255,255,0.2), transparent);

transition: left 0.5s;

}

.btn:hover::before {

left: 100%;

}

.btn:hover {

transform: translateY(-2px);

box-shadow: 0 10px 20px rgba(255, 107, 107, 0.3);

}

.btn:disabled {

opacity: 0.5;

cursor: not-allowed;

transform: none;

}

.btn-secondary {

background: linear-gradient(45deg, #48dbfb, #0abde3);

}

/\* 메인 컨텐츠 영역 \*/

.main-content {

margin-left: 450px;

padding: 30px;

width: calc(100% - 450px);

min-height: 100vh;

}

/\* 스토리보드 그리드 \*/

.storyboard {

display: grid;

grid-template-columns: repeat(auto-fill, minmax(350px, 1fr));

gap: 25px;

animation: fadeIn 0.8s ease;

}

.story-card {

background: rgba(30, 30, 30, 0.9);

border-radius: 12px;

overflow: hidden;

transition: all 0.3s ease;

cursor: pointer;

animation: fadeInUp 0.6s ease;

position: relative;

}

.story-card:hover {

transform: scale(1.03);

box-shadow: 0 15px 30px rgba(255, 107, 107, 0.3);

}

.story-card img {

width: 100%;

height: 250px;

object-fit: cover;

background: #1a1a1a;

}

.story-card-info {

padding: 15px;

background: rgba(40, 40, 40, 0.9);

}

.story-card-number {

display: inline-block;

background: linear-gradient(45deg, #ff6b6b, #feca57);

color: white;

padding: 4px 12px;

border-radius: 20px;

font-size: 12px;

margin-bottom: 10px;

}

.story-card-description {

font-size: 14px;

color: #d0d0d0;

line-height: 1.5;

margin-bottom: 10px;

}

.story-card-time {

font-size: 12px;

color: #808080;

}

/\* 로딩 오버레이 \*/

.loading-overlay {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(0, 0, 0, 0.9);

display: none;

justify-content: center;

align-items: center;

z-index: 1000;

flex-direction: column;

}

.loading-overlay.active {

display: flex;

}

.spinner {

width: 60px;

height: 60px;

border: 4px solid rgba(255, 107, 107, 0.3);

border-top-color: #ff6b6b;

border-radius: 50%;

animation: spin 1s linear infinite;

}

@keyframes spin {

to { transform: rotate(360deg); }

}

.loading-text {

margin-top: 20px;

font-size: 18px;

color: #f0f0f0;

}

.progress-bar {

width: 300px;

height: 4px;

background: rgba(255, 255, 255, 0.1);

border-radius: 2px;

margin-top: 20px;

overflow: hidden;

}

.progress-fill {

height: 100%;

background: linear-gradient(45deg, #ff6b6b, #feca57);

border-radius: 2px;

transition: width 0.3s ease;

}

/\* 스토리보드 뷰어 \*/

.storyboard-viewer {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(0, 0, 0, 0.95);

display: none;

justify-content: center;

align-items: center;

z-index: 2000;

animation: fadeIn 0.3s ease;

}

.storyboard-viewer.active {

display: flex;

}

@keyframes fadeIn {

from { opacity: 0; }

to { opacity: 1; }

}

.viewer-content {

position: relative;

max-width: 90%;

max-height: 90%;

display: flex;

flex-direction: column;

align-items: center;

}

.viewer-image {

max-width: 100%;

max-height: 70vh;

object-fit: contain;

border-radius: 8px;

}

.viewer-description {

margin-top: 20px;

padding: 15px 25px;

background: rgba(40, 40, 40, 0.9);

border-radius: 8px;

max-width: 600px;

text-align: center;

}

.viewer-scene-number {

display: inline-block;

background: linear-gradient(45deg, #ff6b6b, #feca57);

color: white;

padding: 6px 16px;

border-radius: 20px;

font-size: 14px;

margin-bottom: 10px;

}

.viewer-scene-text {

color: #e0e0e0;

line-height: 1.6;

}

.viewer-close {

position: absolute;

top: -40px;

right: 0;

background: rgba(255, 255, 255, 0.1);

border: none;

color: white;

font-size: 24px;

padding: 10px 15px;

cursor: pointer;

border-radius: 4px;

transition: all 0.3s ease;

}

.viewer-close:hover {

background: rgba(255, 255, 255, 0.2);

}

.viewer-nav {

position: absolute;

top: 50%;

transform: translateY(-50%);

background: rgba(255, 255, 255, 0.1);

border: none;

color: white;

font-size: 30px;

padding: 20px;

cursor: pointer;

transition: all 0.3s ease;

border-radius: 4px;

}

.viewer-nav:hover {

background: rgba(255, 255, 255, 0.2);

}

.viewer-prev {

left: -60px;

}

.viewer-next {

right: -60px;

}

.viewer-controls {

position: absolute;

bottom: -60px;

left: 50%;

transform: translateX(-50%);

display: flex;

gap: 10px;

}

.viewer-btn {

padding: 8px 16px;

background: rgba(255, 255, 255, 0.1);

border: 1px solid rgba(255, 255, 255, 0.2);

color: white;

border-radius: 4px;

cursor: pointer;

transition: all 0.3s ease;

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

}

.viewer-btn:hover {

background: rgba(255, 255, 255, 0.2);

}

/\* 타임라인 뷰 \*/

.timeline-view {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: linear-gradient(135deg, #2d3436 0%, #000000 100%);

display: none;

z-index: 1500;

overflow: auto;

padding: 50px;

}

.timeline-view.active {

display: block;

}

.timeline-header {

text-align: center;

margin-bottom: 50px;

animation: fadeInDown 0.8s ease;

}

@keyframes fadeInDown {

from {

opacity: 0;

transform: translateY(-20px);

}

to {

opacity: 1;

transform: translateY(0);

}

}

.timeline-title {

font-size: 36px;

background: linear-gradient(45deg, #ff6b6b, #feca57);

-webkit-background-clip: text;

-webkit-text-fill-color: transparent;

}

.timeline-close {

position: fixed;

top: 20px;

right: 20px;

background: white;

border: none;

font-size: 24px;

padding: 10px 15px;

cursor: pointer;

border-radius: 50%;

box-shadow: 0 4px 8px rgba(0,0,0,0.2);

transition: all 0.3s ease;

z-index: 1600;

}

.timeline-close:hover {

transform: scale(1.1);

box-shadow: 0 6px 12px rgba(0,0,0,0.3);

}

.timeline-container {

position: relative;

max-width: 1200px;

margin: 0 auto;

}

.timeline-line {

position: absolute;

left: 50%;

top: 0;

bottom: 0;

width: 2px;

background: linear-gradient(180deg, #ff6b6b, #feca57);

transform: translateX(-50%);

}

.timeline-item {

display: flex;

margin-bottom: 50px;

position: relative;

animation: fadeInScale 0.6s ease;

}

@keyframes fadeInScale {

from {

opacity: 0;

transform: scale(0.9);

}

to {

opacity: 1;

transform: scale(1);

}

}

.timeline-item:nth-child(even) {

flex-direction: row-reverse;

}

.timeline-content {

width: 45%;

padding: 20px;

background: rgba(40, 40, 40, 0.9);

border-radius: 12px;

box-shadow: 0 4px 8px rgba(0,0,0,0.3);

}

.timeline-image {

width: 100%;

height: 200px;

object-fit: cover;

border-radius: 8px;

margin-bottom: 15px;

}

.timeline-dot {

position: absolute;

left: 50%;

top: 30px;

width: 20px;

height: 20px;

background: #ff6b6b;

border: 4px solid #2d3436;

border-radius: 50%;

transform: translateX(-50%);

z-index: 1;

}

.timeline-scene-number {

display: inline-block;

background: linear-gradient(45deg, #ff6b6b, #feca57);

color: white;

padding: 4px 12px;

border-radius: 20px;

font-size: 12px;

margin-bottom: 10px;

}

.timeline-description {

color: #d0d0d0;

line-height: 1.6;

}

/\* 알림 메시지 \*/

.alert {

position: fixed;

top: 20px;

left: 50%;

transform: translateX(-50%);

padding: 15px 25px;

background: rgba(255, 107, 107, 0.9);

color: white;

border-radius: 8px;

font-size: 14px;

z-index: 3000;

animation: slideDown 0.5s ease;

display: none;

}

.alert.show {

display: block;

}

.alert.success {

background: rgba(46, 213, 115, 0.9);

}

@keyframes slideDown {

from {

transform: translateX(-50%) translateY(-100%);

}

to {

transform: translateX(-50%) translateY(0);

}

}

/\* 빈 상태 메시지 \*/

.empty-state {

text-align: center;

padding: 60px 20px;

color: #808080;

}

.empty-state-icon {

font-size: 80px;

margin-bottom: 20px;

opacity: 0.5;

}

.empty-state-text {

font-size: 18px;

}

/\* 반응형 디자인 \*/

@media (max-width: 768px) {

.sidebar {

width: 100%;

position: relative;

height: auto;

}

.main-content {

margin-left: 0;

width: 100%;

}

body {

flex-direction: column;

}

.storyboard {

grid-template-columns: 1fr;

}

.viewer-prev {

left: 10px;

}

.viewer-next {

right: 10px;

}

.timeline-item,

.timeline-item:nth-child(even) {

flex-direction: column;

}

.timeline-content {

width: 100%;

}

.timeline-line {

left: 20px;

}

.timeline-dot {

left: 20px;

}

}

</style>

</head>

<body>

<!-- 사이드바 -->

<div class="sidebar">

<h1>🎬 AI 스토리보드 생성기</h1>

<div class="input-group">

<label for="apiKey">🔑 Gemini API 키</label>

<input type="password" id="apiKey" placeholder="API 키를 입력하세요">

</div>

<div class="input-group">

<label for="imageUpload">🎨 원본 이미지 (스타일 참조용)</label>

<div class="file-upload">

<input type="file" id="imageUpload" accept="image/\*">

<label for="imageUpload" class="file-upload-label">

클릭하여 이미지를 선택하세요

</label>

</div>

<div id="imagePreview" class="image-preview" style="display: none;">

<img id="previewImg" src="" alt="미리보기">

<div id="imageInfo" class="image-info"></div>

</div>

</div>

<div class="input-group">

<label for="mainCharacter">👤 주인공/주제 설명</label>

<textarea id="mainCharacter" placeholder="예: 빨간 모자를 쓴 소녀, 숲속의 작은 집">빨간 모자를 쓴 소녀</textarea>

</div>

<div class="input-group">

<label for="artStyle">🎭 아트 스타일</label>

<select id="artStyle">

<option value="maintain">원본 이미지 스타일 유지</option>

<option value="cinematic">시네마틱</option>

<option value="animation">애니메이션</option>

<option value="realistic">사실적</option>

<option value="fantasy">판타지</option>

<option value="noir">느와르</option>

</select>

</div>

<div class="scenario-section">

<div class="scenario-header">

<span class="scenario-title">📝 시나리오 컷</span>

<button class="add-scene-btn" onclick="addScene()">+ 씬 추가</button>

</div>

<div id="scenarioList">

<!-- 기본 씬 3개 -->

<div class="scene-item" data-scene-id="1">

<div class="scene-header">

<span class="scene-number">씬 1</span>

<button class="remove-scene-btn" onclick="removeScene(this)">×</button>

</div>

<textarea class="scene-description" placeholder="이 씬의 상황을 설명하세요...">소녀가 숲길을 걷고 있다. 햇살이 나뭇잎 사이로 비치고 있다.</textarea>

</div>

<div class="scene-item" data-scene-id="2">

<div class="scene-header">

<span class="scene-number">씬 2</span>

<button class="remove-scene-btn" onclick="removeScene(this)">×</button>

</div>

<textarea class="scene-description" placeholder="이 씬의 상황을 설명하세요...">갑자기 늑대가 나타나 소녀를 바라본다.</textarea>

</div>

<div class="scene-item" data-scene-id="3">

<div class="scene-header">

<span class="scene-number">씬 3</span>

<button class="remove-scene-btn" onclick="removeScene(this)">×</button>

</div>

<textarea class="scene-description" placeholder="이 씬의 상황을 설명하세요...">소녀가 할머니 집에 도착하여 문을 두드린다.</textarea>

</div>

</div>

</div>

<button id="generateBtn" class="btn">

✨ 스토리보드 생성 시작

</button>

<button id="timelineBtn" class="btn btn-secondary">

📚 타임라인 보기

</button>

</div>

<!-- 메인 컨텐츠 -->

<div class="main-content">

<div id="storyboard" class="storyboard">

<div class="empty-state">

<div class="empty-state-icon">🎬</div>

<div class="empty-state-text">

아직 생성된 스토리보드가 없습니다<br>

왼쪽 패널에서 시나리오를 작성하고 생성을 시작해보세요!

</div>

</div>

</div>

</div>

<!-- 로딩 오버레이 -->

<div id="loadingOverlay" class="loading-overlay">

<div class="spinner"></div>

<div class="loading-text">스토리보드를 생성 중입니다...</div>

<div class="progress-bar">

<div id="progressFill" class="progress-fill" style="width: 0%"></div>

</div>

<div id="progressText" style="margin-top: 10px; color: #a0a0a0; font-size: 14px;"></div>

</div>

<!-- 스토리보드 뷰어 -->

<div id="storyboardViewer" class="storyboard-viewer">

<div class="viewer-content">

<button class="viewer-close" onclick="closeViewer()">✕</button>

<button class="viewer-nav viewer-prev" onclick="navigateViewer(-1)">‹</button>

<button class="viewer-nav viewer-next" onclick="navigateViewer(1)">›</button>

<img id="viewerImage" class="viewer-image" src="" alt="">

<div class="viewer-description">

<div id="viewerSceneNumber" class="viewer-scene-number"></div>

<div id="viewerSceneText" class="viewer-scene-text"></div>

</div>

<div class="viewer-controls">

<button class="viewer-btn" onclick="downloadImage()">💾 다운로드</button>

<button class="viewer-btn" onclick="downloadAll()">📦 전체 다운로드</button>

<button class="viewer-btn" onclick="exportStoryboard()">📄 PDF 내보내기</button>

</div>

</div>

</div>

<!-- 타임라인 뷰 -->

<div id="timelineView" class="timeline-view">

<button class="timeline-close" onclick="closeTimeline()">✕</button>

<div class="timeline-header">

<h2 class="timeline-title">🎬 스토리 타임라인</h2>

</div>

<div class="timeline-container">

<div class="timeline-line"></div>

<div id="timelineItems"></div>

</div>

</div>

<!-- 알림 메시지 -->

<div id="alert" class="alert"></div>

<script>

// 전역 변수

let uploadedImage = null;

let generatedStoryboard = [];

let currentViewerIndex = 0;

let sceneCounter = 3;

// API 키 암호화/복호화

function encryptKey(key) {

return btoa(key);

}

function decryptKey(encryptedKey) {

try {

return atob(encryptedKey);

} catch {

return null;

}

}

// 이미지 업로드 처리

document.getElementById('imageUpload').addEventListener('change', function(e) {

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

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

const reader = new FileReader();

reader.onload = function(e) {

uploadedImage = e.target.result;

displayImagePreview(uploadedImage, file);

};

reader.readAsDataURL(file);

}

});

// 이미지 미리보기 표시

function displayImagePreview(imageSrc, file) {

const preview = document.getElementById('imagePreview');

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

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

previewImg.src = imageSrc;

preview.style.display = 'block';

const img = new Image();

img.onload = function() {

const ratio = (img.width / img.height).toFixed(2);

imageInfo.innerHTML = `

크기: ${img.width} x ${img.height}px<br>

비율: ${ratio}:1<br>

파일 크기: ${(file.size / 1024).toFixed(2)}KB

`;

};

img.src = imageSrc;

}

// 씬 추가

function addScene() {

sceneCounter++;

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

const sceneItem = document.createElement('div');

sceneItem.className = 'scene-item';

sceneItem.setAttribute('data-scene-id', sceneCounter);

sceneItem.innerHTML = `

<div class="scene-header">

<span class="scene-number">씬 ${sceneCounter}</span>

<button class="remove-scene-btn" onclick="removeScene(this)">×</button>

</div>

<textarea class="scene-description" placeholder="이 씬의 상황을 설명하세요..."></textarea>

`;

scenarioList.appendChild(sceneItem);

updateSceneNumbers();

}

// 씬 제거

function removeScene(button) {

const sceneItems = document.querySelectorAll('.scene-item');

if (sceneItems.length <= 1) {

showAlert('최소 1개의 씬은 필요합니다!');

return;

}

button.closest('.scene-item').remove();

updateSceneNumbers();

}

// 씬 번호 업데이트

function updateSceneNumbers() {

const sceneItems = document.querySelectorAll('.scene-item');

sceneItems.forEach((item, index) => {

item.querySelector('.scene-number').textContent = `씬 ${index + 1}`;

});

}

// 스토리보드 생성 시작

document.getElementById('generateBtn').addEventListener('click', async function() {

const apiKey = document.getElementById('apiKey').value;

const mainCharacter = document.getElementById('mainCharacter').value;

const artStyle = document.getElementById('artStyle').value;

// 유효성 검사

if (!apiKey) {

showAlert('API 키를 입력해주세요!');

return;

}

if (!uploadedImage) {

showAlert('원본 이미지를 업로드해주세요!');

return;

}

if (!mainCharacter) {

showAlert('주인공/주제를 입력해주세요!');

return;

}

// 시나리오 수집

const scenarios = [];

document.querySelectorAll('.scene-item').forEach((item, index) => {

const description = item.querySelector('.scene-description').value;

if (description.trim()) {

scenarios.push({

sceneNumber: index + 1,

description: description.trim()

});

}

});

if (scenarios.length === 0) {

showAlert('최소 1개의 시나리오를 작성해주세요!');

return;

}

// API 키 저장

sessionStorage.setItem('gemini\_api\_key', encryptKey(apiKey));

// 버튼 비활성화

this.disabled = true;

// 로딩 표시

showLoading(true);

// 스토리보드 초기화

generatedStoryboard = [];

updateStoryboard();

// 스토리보드 생성

await generateStoryboard(apiKey, mainCharacter, artStyle, scenarios);

// 버튼 활성화

this.disabled = false;

showLoading(false);

});

// 스토리보드 생성 함수

async function generateStoryboard(apiKey, mainCharacter, artStyle, scenarios) {

let previousContext = "";

let successCount = 0;

for (let i = 0; i < scenarios.length; i++) {

const scene = scenarios[i];

updateProgress(i + 1, scenarios.length, scene.sceneNumber);

// 스타일 매핑

const styleMap = {

'maintain': 'maintaining the exact visual style and art direction of the reference image',

'cinematic': 'cinematic movie style with dramatic lighting',

'animation': 'animated cartoon style',

'realistic': 'photorealistic style',

'fantasy': 'fantasy art style with magical elements',

'noir': 'film noir style with high contrast'

};

// 프롬프트 구성 (이전 컨텍스트 포함)

let contextPrompt = "";

if (previousContext) {

contextPrompt = `Previous story context: ${previousContext}. `;

}

const prompt = `${contextPrompt}

Create an image for scene ${scene.sceneNumber}: ${scene.description}

Main character/subject: ${mainCharacter}

Visual style: ${styleMap[artStyle]}

Important: Maintain visual consistency with the reference image's style, color palette, and artistic approach.

Keep the same character appearance and characteristics throughout.

The scene should naturally flow from the previous context.

High quality, detailed rendering with consistent art direction.`;

let retryCount = 0;

const maxRetries = 3;

let success = false;

while (retryCount < maxRetries && !success) {

try {

const imageData = await callGeminiAPI(apiKey, prompt, uploadedImage);

if (imageData) {

const storyItem = {

id: Date.now() + i,

src: imageData,

sceneNumber: scene.sceneNumber,

description: scene.description,

timestamp: new Date().toLocaleString('ko-KR')

};

generatedStoryboard.push(storyItem);

addStoryToBoard(storyItem);

successCount++;

success = true;

// 현재 씬을 이전 컨텍스트에 추가

previousContext += ` Scene ${scene.sceneNumber}: ${scene.description}`;

}

} catch (error) {

console.error(`생성 실패 (시도 ${retryCount + 1}/${maxRetries}):`, error);

retryCount++;

if (retryCount < maxRetries) {

await sleep(2000);

}

}

}

// API 호출 간격

if (i < scenarios.length - 1) {

await sleep(1500);

}

}

if (successCount > 0) {

showAlert(`✅ ${successCount}개의 스토리보드 생성 완료!`, 'success');

} else {

showAlert('❌ 스토리보드 생성에 실패했습니다. API 키를 확인해주세요.');

}

}

// Gemini API 호출

async function callGeminiAPI(apiKey, prompt, imageBase64) {

const API\_URL = `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash-image-preview:generateContent?key=${apiKey}`;

try {

const response = await fetch(API\_URL, {

method: 'POST',

headers: {

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

},

body: JSON.stringify({

contents: [{

parts: [

{

text: prompt

},

{

inline\_data: {

mime\_type: "image/jpeg",

data: imageBase64.split(',')[1]

}

}

]

}],

generationConfig: {

temperature: 0.9,

topK: 32,

topP: 1,

maxOutputTokens: 4096,

}

})

});

if (!response.ok) {

throw new Error(`API 오류: ${response.status}`);

}

const data = await response.json();

// 생성된 이미지 추출 (실제 API 응답 구조에 맞게 수정 필요)

if (data.candidates && data.candidates[0] && data.candidates[0].content) {

const parts = data.candidates[0].content.parts;

for (const part of parts) {

if (part.inlineData) {

return `data:image/png;base64,${part.inlineData.data}`;

}

}

}

// 데모용 더미 이미지 반환

return createDummyImage(prompt);

} catch (error) {

console.error('API 호출 오류:', error);

// 데모용 더미 이미지 반환

return createDummyImage(prompt);

}

}

// 데모용 더미 이미지 생성

function createDummyImage(text) {

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

canvas.width = 512;

canvas.height = 512;

const ctx = canvas.getContext('2d');

// 그라데이션 배경

const gradient = ctx.createLinearGradient(0, 0, 512, 512);

gradient.addColorStop(0, `hsl(${Math.random() \* 360}, 70%, 50%)`);

gradient.addColorStop(1, `hsl(${Math.random() \* 360}, 70%, 30%)`);

ctx.fillStyle = gradient;

ctx.fillRect(0, 0, 512, 512);

// 장식 요소

ctx.globalAlpha = 0.3;

for (let i = 0; i < 5; i++) {

ctx.beginPath();

ctx.arc(

Math.random() \* 512,

Math.random() \* 512,

Math.random() \* 100 + 50,

0,

Math.PI \* 2

);

ctx.fillStyle = `hsl(${Math.random() \* 360}, 70%, 60%)`;

ctx.fill();

}

// 텍스트

ctx.globalAlpha = 1;

ctx.fillStyle = 'white';

ctx.font = 'bold 24px Jua';

ctx.textAlign = 'center';

ctx.textBaseline = 'middle';

ctx.shadowColor = 'rgba(0,0,0,0.5)';

ctx.shadowBlur = 10;

ctx.fillText('스토리보드 이미지', 256, 256);

ctx.font = '16px Jua';

ctx.fillText(text.substring(0, 40) + '...', 256, 290);

return canvas.toDataURL();

}

// 스토리보드에 추가

function addStoryToBoard(story) {

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

// 빈 상태 메시지 제거

const emptyState = storyboard.querySelector('.empty-state');

if (emptyState) {

emptyState.remove();

}

const card = document.createElement('div');

card.className = 'story-card';

card.onclick = () => openViewer(generatedStoryboard.indexOf(story));

card.innerHTML = `

<img src="${story.src}" alt="씬 ${story.sceneNumber}">

<div class="story-card-info">

<span class="story-card-number">씬 ${story.sceneNumber}</span>

<div class="story-card-description">${story.description}</div>

<div class="story-card-time">🕒 ${story.timestamp}</div>

</div>

`;

storyboard.appendChild(card);

}

// 스토리보드 업데이트

function updateStoryboard() {

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

storyboard.innerHTML = '';

if (generatedStoryboard.length === 0) {

storyboard.innerHTML = `

<div class="empty-state">

<div class="empty-state-icon">🎬</div>

<div class="empty-state-text">

아직 생성된 스토리보드가 없습니다<br>

왼쪽 패널에서 시나리오를 작성하고 생성을 시작해보세요!

</div>

</div>

`;

} else {

generatedStoryboard.forEach(story => addStoryToBoard(story));

}

}

// 진행 상황 업데이트

function updateProgress(current, total, sceneNumber) {

const percent = (current / total) \* 100;

document.getElementById('progressFill').style.width = `${percent}%`;

document.getElementById('progressText').textContent = `씬 ${sceneNumber} 생성 중... (${current}/${total})`;

}

// 로딩 표시/숨기기

function showLoading(show) {

const overlay = document.getElementById('loadingOverlay');

overlay.classList.toggle('active', show);

if (!show) {

document.getElementById('progressFill').style.width = '0%';

document.getElementById('progressText').textContent = '';

}

}

// 뷰어 열기

function openViewer(index) {

if (generatedStoryboard.length === 0) return;

currentViewerIndex = index;

const viewer = document.getElementById('storyboardViewer');

const story = generatedStoryboard[index];

document.getElementById('viewerImage').src = story.src;

document.getElementById('viewerSceneNumber').textContent = `씬 ${story.sceneNumber}`;

document.getElementById('viewerSceneText').textContent = story.description;

viewer.classList.add('active');

}

// 뷰어 닫기

function closeViewer() {

document.getElementById('storyboardViewer').classList.remove('active');

}

// 뷰어 네비게이션

function navigateViewer(direction) {

currentViewerIndex += direction;

if (currentViewerIndex < 0) {

currentViewerIndex = generatedStoryboard.length - 1;

} else if (currentViewerIndex >= generatedStoryboard.length) {

currentViewerIndex = 0;

}

const story = generatedStoryboard[currentViewerIndex];

document.getElementById('viewerImage').src = story.src;

document.getElementById('viewerSceneNumber').textContent = `씬 ${story.sceneNumber}`;

document.getElementById('viewerSceneText').textContent = story.description;

}

// 이미지 다운로드

function downloadImage() {

const story = generatedStoryboard[currentViewerIndex];

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

link.href = story.src;

link.download = `storyboard\_scene\_${story.sceneNumber}\_${Date.now()}.png`;

link.click();

}

// 전체 다운로드

async function downloadAll() {

for (let i = 0; i < generatedStoryboard.length; i++) {

const story = generatedStoryboard[i];

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

link.href = story.src;

link.download = `storyboard\_scene\_${story.sceneNumber}.png`;

link.click();

await sleep(500); // 다운로드 간격

}

showAlert('모든 이미지 다운로드 완료!', 'success');

}

// PDF 내보내기 (간단한 구현)

function exportStoryboard() {

// 실제 구현에서는 jsPDF 라이브러리 사용

showAlert('PDF 내보내기 기능은 준비 중입니다.');

}

// 타임라인 보기

document.getElementById('timelineBtn').addEventListener('click', function() {

if (generatedStoryboard.length === 0) {

showAlert('먼저 스토리보드를 생성해주세요!');

return;

}

openTimeline();

});

// 타임라인 열기

function openTimeline() {

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

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

timelineItems.innerHTML = '';

generatedStoryboard.forEach((story, index) => {

const item = document.createElement('div');

item.className = 'timeline-item';

item.innerHTML = `

<div class="timeline-content">

<img src="${story.src}" alt="씬 ${story.sceneNumber}" class="timeline-image">

<span class="timeline-scene-number">씬 ${story.sceneNumber}</span>

<div class="timeline-description">${story.description}</div>

</div>

<div class="timeline-dot"></div>

`;

timelineItems.appendChild(item);

});

timelineView.classList.add('active');

}

// 타임라인 닫기

function closeTimeline() {

document.getElementById('timelineView').classList.remove('active');

}

// 알림 표시

function showAlert(message, type = '') {

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

alert.textContent = message;

alert.className = 'alert show';

if (type) {

alert.classList.add(type);

}

setTimeout(() => {

alert.classList.remove('show');

}, 3000);

}

// 유틸리티 함수

function sleep(ms) {

return new Promise(resolve => setTimeout(resolve, ms));

}

// 키보드 이벤트 처리

document.addEventListener('keydown', function(e) {

const viewer = document.getElementById('storyboardViewer');

const timeline = document.getElementById('timelineView');

if (viewer.classList.contains('active')) {

if (e.key === 'ArrowLeft') {

navigateViewer(-1);

} else if (e.key === 'ArrowRight') {

navigateViewer(1);

} else if (e.key === 'Escape') {

closeViewer();

}

} else if (timeline.classList.contains('active')) {

if (e.key === 'Escape') {

closeTimeline();

}

}

});

// 뷰어 배경 클릭으로 닫기

document.getElementById('storyboardViewer').addEventListener('click', function(e) {

if (e.target === this) {

closeViewer();

}

});

// 페이지 로드 시 API 키 복원

window.addEventListener('load', function() {

const savedKey = sessionStorage.getItem('gemini\_api\_key');

if (savedKey) {

const decryptedKey = decryptKey(savedKey);

if (decryptedKey) {

document.getElementById('apiKey').value = decryptedKey;

}

}

});

</script>

</body>

</html>