Test SDK Page デザイン仕様書 (/test-sdk)

概要

Firebase SDK getBlob() 機能のテストに特化したビジュアルテストページです。メインの詩表示ページと同様のグラスモーフィズムデザインを採用し、Firebase Storage CORS回避技術の動作を実際に確認できます。

目的・役割

🔥 **Firebase SDK CORS回避テスト**

- getBlob() によるバイナリデータ取得確認
- Object URL 生成とクリーンアップテスト
- CORS制限を回避した画像読み込み検証

🎨 **ビジュアル統合テスト**

- BackgroundImage コンポーネントのSDK方式動作確認
- FloatingParticles アニメーションとの統合
- グラスモーフィズムUIの実際の動作

処理フロー

Firebase SDK CORS回避テスト処理フロー

```
flowchart TD
    A[ページ読み込み完了] --> B[FloatingParticles アニメーション開始]
    B --> C[BackgroundImage コンポーネント初期化]
    C --> D[currentTestId設定（デフォルト: 1752308749714）]
    D --> E[Firebase SDK getBlob() 方式強制実行]
    E --> F{Firebase接続成功？}
    F -->|失敗| G[フォールバック背景表示]
    F -->|成功| H[images/テストID.png パス生成]
    H --> I[getBlob() 実行]
    I --> J{getBlob() 成功？}
    J -->|失敗| K[Download URL方式フォールバック]
    J -->|成功| L[Blob データ取得]
    L --> M[Object URL 生成]
    M --> N[背景画像設定]
    N --> O[パフォーマンス情報記録]
    O --> P[左下に成功情報表示]
    K --> Q[従来方式で背景画像設定]
    G --> R[エラー情報表示]
    Q --> S[フォールバック完了]
    P --> T[ユーザーインタラクション待機]
    S --> T
    R --> T
```

画像ID切り替え処理フロー

```
flowchart TD
    A[ユーザーがID切り替えボタンクリック] --> B[選択されたIDを currentT
    B --> C[BackgroundImage コンポーネント再レンダリング]
    C --> D[新しいpoemId でuseEffect実行]
    D --> E[前回のObject URL クリーンアップ]
    E --> F[新しい画像パス生成: images/新ID.png]
    F --> G[Firebase SDK getBlob() 再実行]
    G --> H{新しい画像取得成功？}
    H -->|成功| I[新しいObject URL生成]
    H -->|失敗| J[エラー表示・フォールバック背景]
    I --> K[背景画像スムーズ切り替え]
    K --> L[パフォーマンス測定・表示更新]
```

```
    J --> M[エラー詳細コンソール出力]
    L --> N[切り替え完了]
    M --> N
```

統合テスト期待動作フロー

```
flowchart TD
    A[テストページ表示] --> B[期待される動作確認]
    B --> C[コンソールログ: Firebase SDK getBlob() 方式で画像読み込み
    C --> D[コンソールログ: getBlob() 成功 または適切なフォールバック]
    D --> E[左下に緑色パフォーマンス情報表示]
    E --> F[背景画像の正常表示確認]
    F --> G[FloatingParticles アニメーション動作確認]
    G --> H[グラスモーフィズム効果確認]
    H --> I[ID切り替えボタン群動作確認]
    I --> J[リロード・デバッグ・ホームボタン動作確認]
    J --> K{全ての動作正常？}
    K -->|正常| L[SDK CORS回避テスト成功]
    K -->|異常| M[デバッグページで詳細調査]
    L --> N[テスト完了]
    M --> O[問題の特定と対処]
```

デザインシステム

この SDKテストページは、統一されたデザインシステムに準拠しています。

詳細なカラーパレット、タイポグラフィ、グラスモーフィズム効果については、デザインシステム を参照してください。

UIコンポーネント仕様

1. テストバッジ

```
.sdk-test-badge {
  display: inline-block;
  background: rgba(255, 255, 255, 0.2);
  backdrop-filter: blur(8px);
  border: 1px solid rgba(255, 255, 255, 0.3);
  color: white;
  padding: 0.75rem 1.5rem;
  border-radius: 9999px;
  font-size: 0.875rem;
  font-weight: 500;
  box-shadow: 0 10px 25px -5px rgba(0, 0, 0, 0.1);
  margin-bottom: 2rem;
}

.sdk-test-badge::before {
  content: "🔥 ";
}
```

2. メインコンテンツエリア（グラスモーフィズム）

```
.sdk-main-content {
  position: relative;
  background: rgba(255, 255, 255, 0.1);
  backdrop-filter: blur(24px);
  border: 1px solid rgba(255, 255, 255, 0.2);
  border-radius: 1.5rem;
  padding: 2rem;
  margin-bottom: 2rem;
  box-shadow: 0 25px 50px -12px rgba(0, 0, 0, 0.25);
}

/* 内側グロー効果 */
.sdk-content-glow::before {
  content: "";
  position: absolute;
  inset: 0;
  background: linear-gradient(135deg, rgba(255, 255, 255, 0.05)
  border-radius: 1.5rem;
  pointer-events: none;
```

```
}

/* 装飾的光の効果 */
.sdk-light-decoration-1 {
  position: absolute;
  top: -0.25rem;
  left: -0.25rem;
  width: 4rem;
  height: 4rem;
  background: rgba(255, 255, 255, 0.1);
  border-radius: 50%;
  filter: blur(24px);
}

.sdk-light-decoration-2 {
  position: absolute;
  bottom: -0.25rem;
  right: -0.25rem;
  width: 3rem;
  height: 3rem;
  background: rgba(251, 191, 36, 0.2);  /* pink-300/20 */
  border-radius: 50%;
  filter: blur(16px);
}
```

3. テスト情報表示

```
.sdk-test-info {
  position: relative;
  color: white;
  font-weight: 500;
  text-align: center;
}

.sdk-test-title {
  font-size: 1.5rem;        /* text-xl md:text-2xl */
  line-height: 1.6;
  margin-bottom: 1rem;
}
```

```
.sdk-test-id {
  font-size: 1.125rem;      /* text-lg */
  margin-bottom: 0.5rem;
}

.sdk-test-note {
  font-size: 0.875rem;      /* text-sm */
  opacity: 0.75;
  margin-top: 0.5rem;
}

/* レスポンシブ対応 */
@media (min-width: 768px) {
  .sdk-test-title {
    font-size: 1.5rem;      /* md:text-2xl */
  }
}
```

4. 画像ID切り替えセクション

```
.sdk-id-switcher {
  margin-bottom: 2rem;
}

.sdk-switcher-label {
  color: white;
  font-size: 0.875rem;
  margin-bottom: 1rem;
  opacity: 0.75;
}

.sdk-id-grid {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  gap: 0.5rem;
}

@media (min-width: 768px) {
  .sdk-id-grid {
    grid-template-columns: repeat(4, 1fr);
```

```
  }
}

/* ID切り替えボタン */
.sdk-id-button {
  padding: 0.5rem 0.75rem;
  border-radius: 0.25rem;
  font-size: 0.75rem;
  font-family: monospace;
  border: 1px solid;
  cursor: pointer;
  transition: all 0.2s;
}

.sdk-id-button-active {
  background: rgba(59, 130, 246, 0.4);
  border-color: rgba(96, 165, 250, 0.5);
  color: white;
}

.sdk-id-button-inactive {
  background: rgba(255, 255, 255, 0.1);
  border-color: rgba(255, 255, 255, 0.2);
  color: rgba(255, 255, 255, 0.8);
}

.sdk-id-button-inactive:hover {
  background: rgba(255, 255, 255, 0.2);
}
```

5. アクションボタン

```
.sdk-action-buttons {
  display: flex;
  flex-direction: column;
  gap: 1rem;
  justify-content: center;
}

@media (min-width: 640px) {
```

```
  .sdk-action-buttons {
    flex-direction: row;
  }
}

/* ベースボタンスタイル */
.sdk-action-btn {
  backdrop-filter: blur(8px);
  border: 1px solid;
  color: white;
  padding: 1rem 2rem;
  border-radius: 9999px;
  font-weight: 500;
  transition: all 0.2s;
  transform: scale(1);
  box-shadow: 0 10px 25px -5px rgba(0, 0, 0, 0.1);
  display: flex;
  align-items: center;
  justify-content: center;
  text-decoration: none;
  cursor: pointer;
}

.sdk-action-btn:hover {
  transform: scale(1.05);
  box-shadow: 0 20px 25px -5px rgba(0, 0, 0, 0.15);
}

/* リロードボタン */
.sdk-reload-btn {
  background: rgba(34, 197, 94, 0.2);
  border-color: rgba(74, 222, 128, 0.3);
}

.sdk-reload-btn:hover {
  background: rgba(34, 197, 94, 0.3);
}

.sdk-reload-btn::before {
  content: "🔄 ";
  margin-right: 0.5rem;
}
```

```
/* デバッグボタン */
.sdk-debug-btn {
  background: rgba(168, 85, 247, 0.2);
  border-color: rgba(196, 181, 253, 0.3);
}

.sdk-debug-btn:hover {
  background: rgba(168, 85, 247, 0.3);
}

.sdk-debug-btn::before {
  content: "🔍 ";
  margin-right: 0.5rem;
}

/* ホームボタン */
.sdk-home-btn {
  background: rgba(236, 72, 153, 0.2);
  border-color: rgba(244, 114, 182, 0.3);
}

.sdk-home-btn:hover {
  background: rgba(236, 72, 153, 0.3);
}

.sdk-home-btn::before {
  content: "✨ ";
  margin-right: 0.5rem;
}
```

6. 期待結果表示

```
.sdk-expected-results {
  margin-top: 2rem;
  color: rgba(255, 255, 255, 0.7);
  font-size: 0.875rem;
}

.sdk-expected-title {
  font-weight: 500;
```

```
  margin-bottom: 0.5rem;
}

.sdk-expected-list {
  text-align: left;
  space-y: 0.25rem;
}

.sdk-expected-item {
  margin-bottom: 0.25rem;
}

.sdk-expected-item::before {
  content: "• ";
  margin-right: 0.5rem;
}
```

機能仕様

Firebase SDK テスト機能

1. **背景画像コンポーネント統合**

```
<BackgroundImage
  imageUrl={null}           // URL強制null
  poemId={currentTestId}    // テスト対象ID
/>
```

2. **実在画像ID管理**

```
const existingImageIds = [
  '1752308749714',   // Firebase Consoleで確認済み
  '1752304956761',
  '1752305274447',
  // 他の実在ID...
];
```

3. **動的ID切り替え**
    - ボタンクリックでtestIdを変更
    - BackgroundImageコンポーネントの自動再読み込み
    - リアルタイムでのgetBlob()テスト実行

パフォーマンス測定

1. **コンソールログ出力**
    - getBlob()開始時刻の記録
    - Object URL生成完了時刻
    - 総読み込み時間の計算
2. **パフォーマンス情報表示**
    - 左下に緑色の成功インジケーター
    - 読み込み時間とファイルサイズ
    - 使用されたメソッド（SDK/Legacy）

エラーハンドリング

1. **段階的フォールバック**
    - getBlob()失敗時のDownload URL方式
    - Firebase接続失敗時のフォールバック背景
    - タイムアウト制御
2. **詳細エラー表示**
    - 赤色のエラーボックス
    - 具体的な失敗理由
    - 推奨対処法の提示

アニメーション・エフェクト

パーティクルアニメーション

```
/* FloatingParticlesコンポーネントとの統合 */
.sdk-particles-layer {
  position: fixed;
  inset: 0;
  pointer-events: none;
  background: transparent;
  z-index: 5;
}
```

トランジション効果

```
.sdk-smooth-transition {
  transition: all 0.2s ease-in-out;
}

.sdk-hover-scale:hover {
  transform: scale(1.05);
}

.sdk-glass-shimmer {
  background: linear-gradient(
    45deg,
    rgba(255, 255, 255, 0.1),
    rgba(255, 255, 255, 0.05),
    rgba(255, 255, 255, 0.1)
  );
  animation: shimmer 3s ease-in-out infinite;
}

@keyframes shimmer {
  0%, 100% { opacity: 0.5; }
  50% { opacity: 1; }
}
```

開発者向け機能

デバッグ情報

1. **コンソール出力パターン**

```
// 期待されるコンソールログ
console.log('🔥 Firebase SDK getBlob() 方式で画像読み込み開始')
console.log('✅ getBlob() 成功');
console.log('📊 パフォーマンス:', performance);
```

2. **パフォーマンス測定**

```
const performance = {
  loadTime: 245,           // ミリ秒
  size: 1048576,          // バイト
  method: 'Firebase SDK'  // 使用方式
};
```

テストシナリオ

1. **正常系テスト**
   - 複数の実在画像IDでの動作確認
   - Object URL生成とクリーンアップ
   - パフォーマンス情報の正確性
2. **異常系テスト**
   - 存在しない画像IDでの動作
   - ネットワーク接続断での挙動
   - Firebase制限時のフォールバック

レスポンシブデザイン

モバイル（~640px）

- ボタンの縦並び配置
- IDグリッドの2カラム維持
- フォントサイズの調整

タブレット（640px~）

- ボタンの横並び配置
- IDグリッドの4カラム表示

デスクトップ（768px~）

- 最適なスペース活用
- ホバー効果の完全活用
- 大画面での情報密度向上

アクセシビリティ

セマンティック構造

```
<main role="main" aria-label="Firebase SDK テストページ">
  <section aria-labelledby="test-content">
    <h1 id="test-content">Firebase SDK CORS回避テスト</h1>

    <div role="group" aria-label="画像ID選択">
      <!-- ID切り替えボタン群 -->
    </div>

    <nav aria-label="テスト操作" role="navigation">
      <!-- アクションボタン群 -->
    </nav>
```

```
  </section>
</main>
```

キーボードナビゲーション

- Tab順序の論理的な配置
- ボタンのフォーカス状態明示
- Escapeキーでのクイックナビゲーション

統合テスト価値

Firebase連携確認

- Storage権限設定の妥当性
- getBlob()機能の実用性
- CORS回避の実効性

UI/UXテスト

- グラスモーフィズムの視覚効果
- アニメーションのパフォーマンス
- レスポンシブ動作の確認

パフォーマンス評価

- 読み込み速度の測定
- メモリ使用量の監視
- ユーザー体験の評価

更新履歴

- 2025-07-13: 初版作成

- Firebase SDK getBlob() CORS回避テスト機能の仕様文書化