🌊 ゆるVibe Pages ページ遷移フロー図

> *ユーザー体験の流れと開発者向けテストフローの包括的可視化*

フロー概要

このページ遷移図は、ゆるVibe Pagesの実装済み画面遷移を正確に表現しています。メインユーザーフローと、開発・テスト用フローの両方を網羅し、Next.js App Routerの動的ルーティングも含めた全体像を示しています。

メインユーザーフロー

```
journey
    title ゆるVibe Pages ユーザージャーニー
    section テーマ入力
      ホームページにアクセス          : 9: ユーザー
      感情テーマを入力                : 8: ユーザー
      詩を生成するボタンクリック      : 7: ユーザー
    section AI生成処理
      GPT-4o詩生成                    : 5: システム
      DALL-E画像生成                  : 5: システム
      Firebase Storage保存            : 4: システム
      Firestore詩データ保存           : 4: システム
    section 詩表示
      詩ページに自動遷移              : 9: ユーザー
      美しい詩と背景画像を確認        : 10: ユーザー
```

```
    SNS共有ボタンクリック          : 8: ユーザー
    新しい詩作成リンククリック     : 7: ユーザー
```

詳細ページ遷移図

```
graph TD
    A[/ - ホームページ] --> B{テーマ入力}
    B -->|空文字| C[エラー表示]
    B -->|有効| D[ローディング開始]
    C --> B

    D --> E[POST /api/generate-storage]
    E --> F{API成功？}

    F -->|失敗| G[エラーメッセージ表示]
    F -->|成功| H[/view/[id] - 詩表示ページ]
    G --> B

    H --> I[Firestore詩データ取得]
    I --> J{データ存在？}
    J -->|なし| K[404エラーページ]
    J -->|あり| L[詩内容表示]

    L --> M[Firebase画像読み込み]
    M --> N{画像読み込み成功？}
    N -->|成功| O[背景画像表示]
    N -->|失敗| P[フォールバック背景]

    O --> Q[SNS共有ボタン]
    P --> Q
    Q --> R{ユーザーアクション}

    R -->|X共有| S[Twitter共有画面]
    R -->|新しい詩| A
    R -->|ページ共有| T[URL コピー]

    S --> U[外部Twitter]
    T --> V[共有完了通知]
    V --> H
```

```
    %% スタイル定義
    classDef mainPage fill:#e3f2fd
    classDef apiCall fill:#fff3e0
    classDef dataOp fill:#e8f5e8
    classDef userAction fill:#fce4ec
    classDef external fill:#f3e5f5

    class A,H mainPage
    class E,I,M apiCall
    class J,N dataOp
    class B,R,Q userAction
    class S,U external
```

テスト・デバッグページフロー

```
graph TD
    A[/ - ホームページ] --> B{開発者モード？}
    B -->|No| C[通常ユーザーフロー]
    B -->|Yes| D[テストページ群アクセス]

    D --> E[/test - 基本APIテスト]
    D --> F[/test-simple - Safe版テスト]
    D --> G[/test-dummy - ダミー版テスト]
    D --> H[/test-sdk - SDK CORS テスト]
    D --> I[/debug - Storage デバッグ]

    E --> J[POST /api/generate]
    F --> K[POST /api/generate-safe]
    G --> L[POST /api/generate-dummy]

    J --> M{API レスポンス}
    K --> M
    L --> M

    M -->|成功| N[結果表示 + 詩ページリンク]
    M -->|失敗| O[エラー詳細表示]

    N --> P[/view/[id] へ遷移可能]
```

```
    O --> Q[リトライボタン]
    Q --> E

    H --> R[Firebase SDK getBlob() テスト]
    R --> S[既存画像ID選択]
    S --> T{画像読み込み}
    T -->|成功| U[パフォーマンス情報表示]
    T -->|失敗| V[CORS エラー詳細]

    I --> W[Storage詳細調査]
    W --> X[詩ID入力]
    X --> Y{Firestore取得}
    Y -->|成功| Z[画像読み込みテスト実行]
    Y -->|失敗| AA[Firestore エラー表示]

    Z --> BB{画像アクセス}
    BB -->|成功| CC[img + CSS background 両方テスト]
    BB -->|失敗| DD[CORS/権限エラー記録]

    %% スタイル定義
    classDef testPage fill:#e8f5e8
    classDef debugPage fill:#fff3e0
    classDef apiEndpoint fill:#fce4ec
    classDef result fill:#f3e5f5

    class E,F,G,H,I testPage
    class W,X,Y,Z debugPage
    class J,K,L,R apiEndpoint
    class M,N,O,U,V result
```

Next.js App Router 構造

```
graph LR
    A[src/app/] --> B[layout.js]
    A --> C[page.js /]
    A --> D[view/]
    A --> E[test/]
    A --> F[test-simple/]
    A --> G[test-dummy/]
```

```
    A --> H[test-sdk/]
    A --> I[debug/]
    A --> J[api/]

    D --> K[[id]/page.js]

    E --> L[page.js]
    F --> M[page.js]
    G --> N[page.js]
    H --> O[page.js]
    I --> P[page.js]

    J --> Q[generate/route.js]
    J --> R[generate-safe/route.js]
    J --> S[generate-storage/route.js]
    J --> T[generate-dummy/route.js]
    J --> U[generate-simple/route.js]
    J --> V[generate-direct/route.js]

    %% ルート種別スタイル
    classDef staticRoute fill:#e3f2fd
    classDef dynamicRoute fill:#fff3e0
    classDef apiRoute fill:#e8f5e8
    classDef testRoute fill:#fce4ec

    class C,L,M,N,O,P staticRoute
    class K dynamicRoute
    class Q,R,S,T,U,V apiRoute
    class E,F,G,H,I testRoute
```

ユーザー体験マップ

初回訪問ユーザー

```
stateDiagram-v2
    [*] --> ホームページ到達
    ホームページ到達 --> テーマ考案中
    テーマ考案中 --> テーマ入力完了
```

```
    テーマ入力完了 --> 生成ボタンクリック
    生成ボタンクリック --> 生成中待機
    生成中待機 --> 詩ページ表示
    詩ページ表示 --> 詩内容確認
    詩内容確認 --> 背景画像読み込み
    背景画像読み込み --> 完全表示
    完全表示 --> SNS共有検討
    SNS共有検討 --> 共有実行
    SNS共有検討 --> 新詩作成
    共有実行 --> Twitter投稿
    新詩作成 --> ホームページ到達
    Twitter投稿 --> [*]
```

リピーターユーザー

```
stateDiagram-v2
    [*] --> 直接ホームアクセス
    直接ホームアクセス --> 迅速テーマ入力
    迅速テーマ入力 --> 即座生成実行
    即座生成実行 --> 生成中待機
    生成中待機 --> 詩ページ確認
    詩ページ確認 --> 品質評価
    品質評価 --> 満足時共有
    品質評価 --> 不満足時再生成
    満足時共有 --> SNS投稿
    不満足時再生成 --> 直接ホームアクセス
    SNS投稿 --> [*]
```

開発者フロー詳細

API テスト戦略

```
graph TD
    A[開発者] --> B{テスト目的}
```

```
    B -->|基本機能確認| C[/test]
    B -->|安全性確認| D[/test-simple]
    B -->|オフライン開発| E[/test-dummy]
    B -->|CORS問題調査| F[/test-sdk]
    B -->|詳細デバッグ| G[/debug]

    C --> H[/api/generate 呼び出し]
    D --> I[/api/generate-safe 呼び出し]
    E --> J[/api/generate-dummy 呼び出し]

    F --> K[Firebase getBlob() テスト]
    G --> L[Storage + Firestore 詳細調査]

    H --> M{結果確認}
    I --> M
    J --> M
    K --> N{CORS対応確認}
    L --> O{問題特定}

    M -->|OK| P[本番デプロイ準備]
    M -->|NG| Q[エラー調査]
    N -->|OK| R[SDK実装成功]
    N -->|NG| S[CORS設定確認]
    O -->|特定| T[修正実行]
    O -->|不明| U[さらなる調査]

    %% 開発フェーズスタイル
    classDef devAction fill:#e8f5e8
    classDef testResult fill:#fff3e0
    classDef decision fill:#fce4ec

    class A,C,D,E,F,G devAction
    class M,N,O testResult
    class B,P,Q,R,S,T,U decision
```

エラーハンドリングフロー

```
graph TD
    A[ユーザーアクション] --> B{入力バリデーション}
```

```
    B -->|OK| C[API呼び出し]
    B -->|NG| D[クライアント側エラー表示]

    C --> E{API成功？}
    E -->|成功| F[詩ページ遷移]
    E -->|失敗| G[サーバー側エラー]

    G --> H{エラー種別}
    H -->|OpenAI API制限| I[制限エラー表示]
    H -->|Firebase エラー| J[Firebase エラー表示]
    H -->|ネットワークエラー| K[接続エラー表示]
    H -->|その他| L[一般エラー表示]

    F --> M[Firestore データ取得]
    M --> N{データ存在？}
    N -->|存在| O[詩表示]
    N -->|なし| P[404エラーページ]

    O --> Q[画像読み込み]
    Q --> R{画像読み込み}
    R -->|成功| S[完全表示]
    R -->|失敗| T[フォールバック背景]

    D --> U[入力フィールドフォーカス]
    I --> V[リトライボタン表示]
    J --> V
    K --> V
    L --> V
    P --> W[ホームページリンク]

    %% エラー種別スタイル
    classDef success fill:#e8f5e8
    classDef warning fill:#fff3e0
    classDef error fill:#ffebee
    classDef fallback fill:#f3e5f5

    class C,F,O,S success
    class T,V warning
    class D,G,I,J,K,L,P error
    class U,W fallback
```

パフォーマンス最適化ポイント

画像読み込み最適化フロー

```
sequenceDiagram
    participant U as ユーザー
    participant P as 詩ページ
    participant B as BackgroundImage
    participant F as Firebase SDK
    participant S as Storage

    U->>P: 詩ページアクセス
    P->>B: 画像URL渡し

    alt Firebase Storage URL
        B->>F: getBlob() 実行
        F->>S: Blob リクエスト
        S-->>F: Blob データ
        F-->>B: Object URL 作成
        Note over B: CORS 回避成功
    else getBlob() 失敗
        B->>F: getDownloadURL()
        F->>S: URL リクエスト
        S-->>F: Download URL
        F-->>B: 直接URL
        Note over B: CORS 依存
    else 完全失敗
        B->>B: EmergencyBackground
        Note over B: フォールバック背景
    end

    B-->>P: 画像表示完了
    P-->>U: 詩ページ完全表示
```

> *「ページの流れは詩の調べのように。ユーザーの心に寄り添う優雅な遷移を、にゃ〜」* ✨