PickPoints 機能仕様書

1. プロジェクト概要

1.1 アプリケーション名

Pick Points and Routes - ハイキングマップポイント・ルート選択ツール

1.2 目的

PNG画像のハイキングマップから視覚的にポイント・ルートを選択し、座標データをJSONファイルとして出力するWebアプリケーション。ハイキング計画やマップ作成をサポートします。

1.3 技術構成

- **フロントエンド**: HTML5, JavaScript (ES6+ Class構文), CSS3
- **描画**: HTML5 Canvas API
- **ファイル処理**: File System Access API, File API, Blob API
- **レスポンシブ対応**: CSS Flexbox, Grid
- **ブラウザ対応**: モダンブラウザ（Chrome 86+, Firefox 88+, Edge 90+, Safari 14+）

2. 主要機能

2.1 画像読み込み機能

- **対応形式**: PNG画像のみ
- **読み込み方法**: File System Access API（対応ブラウザ）またはファイル選択ダイアログ
- **表示**: Canvas要素でのリアルタイム描画
- **リサイズ**: 画面サイズに応じた自動リサイズ（アスペクト比維持）
- **ファイルハンドル保存**: JSON出力時の同一フォルダ保存用

2.2 編集モード切り替え

2.2.1 ポイント編集モード

- **機能**: 個別ポイントの配置と管理
- **ID入力**: 必須（4文字まで、自動大文字変換）
- **ID形式**: 推奨「X-nn」形式（英大文字1桁-数字2桁）、例：A-01, B-15
- **フォーマット補助**: 全角文字の半角自動変換、数字の0埋め処理
- **マーカー表示**: 赤色円形マーカー（半径4px、白枠1.5px）
- **特殊マーカー**: 開始/終了ポイント指定時は青色表示
- **削除条件**: ID名がブランクの場合は自動削除（blur時）
- **自動フォーカス**: 新規ポイント追加時にID入力欄に自動フォーカス
- **未入力ポイント管理**: 直前の未入力ポイントは新規追加時に自動削除
- **入力ボックス配置**: ポイント近傍の最適位置（画面端回避）

2.2.2 ルート編集モード

- **機能**: ルート中間点（waypoint）の配置と管理
- **ID入力**: 不要
- **マーカー表示**: 青色円形マーカー（半径3px、白枠1px）
- **開始/終了ポイント**: 既存ポイントのIDで指定（必須）
- **入力フィールド**: 開始・終了ポイントID（幅60px、4文字制限）
- **自動大文字変換**: 入力時のリアルタイム変換
- **バリデーション**:
  - 開始・終了ポイントの存在確認
  - 両方のポイントが必須
  - ポイント編集モードで事前登録が必要
- **自動ファイル名**: `{画像名}_route_{開始}-{終了}.json`

2.3 レイアウト切り替え

2.3.1 サイドバーレイアウト

- **構成**: 左側に地図、右側にコントロールパネル（幅240px）
- **適用場面**: デスクトップ環境での作業
- **モバイル対応**: 768px以下で縦積みレイアウトに自動変更

2.3.2 オーバーレイレイアウト

- **構成**: 全画面地図、右上にフローティングコントロールパネル
- **適用場面**: 大きな地図表示が必要な場合
- **背景**: 半透明白（95%透明度）+ ブラーエフェクト

2.4 座標管理機能

2.4.1 座標システム

- **入力座標**: Canvas相対座標（画面表示位置）
- **出力座標**: 画像絶対座標（元画像サイズ基準）
- **変換処理**: スケール変換による座標正規化

2.4.2 ポイント管理

- **最大文字数**: ID名4文字まで
- **入力制限**: 英数字のみ、自動大文字変換
- **位置調整**: 入力ボックスの自動配置（画面端での位置調整）
- **自動削除**: 未入力ポイントの自動管理機能

2.5 データ出力機能

2.5.1 ポイントJSON出力

- **保存場所**: PNG画像と同じフォルダ（File System Access API対応時）
- **ファイル名**: `{画像名}_points.json`
- **出力条件**: 1個以上のポイントが存在すること
- **キャンセル対応**: ユーザーキャンセル時は出力停止

- **フォールバック**: 従来のダウンロード方式に自動切り替え
- **画像参照**: 元ファイル名を含む

```
{
  "totalPoints": 10,
  "imageReference": "hakone_map.png",
  "imageInfo": {
    "width": 1920,
    "height": 1080
  },
  "points": [
    {
      "index": 1,
      "id": "A-01",
      "x": 640,
      "y": 480,
      "isMarker": false
    }
  ],
  "exportedAt": "2025-01-08T12:34:56.789Z"
}
```

2.5.2 ルートJSON出力

- **保存場所**: PNG画像と同じフォルダ（File System Access API対応時）
- **ファイル名**: `{画像名}_route_{開始ポイント}-{終了ポイント}.json`
- **例**: `hakone_map_route_A-01-B-05.json`
- **出力条件**:
  - 1個以上の中間点が存在すること
  - 開始・終了ポイントが両方設定されていること
  - 開始・終了ポイントが事前にポイント登録されていること
- **事前チェック**: 開始・終了ポイントの存在確認
- **キャンセル対応**: ユーザーキャンセル時は出力停止
- **フォールバック**: 従来のダウンロード方式に自動切り替え

```
{
  "routeInfo": {
    "startPoint": "A-01",
    "endPoint": "B-05",
    "waypointCount": 5
  },
  "imageReference": "hakone_map.png",
  "imageInfo": {
    "width": 1920,
    "height": 1080
  },
  "points": [
    {
```

```
      "type": "waypoint",
      "index": 1,
      "x": 320,
      "y": 240
    }
  ],
  "exportedAt": "2025-01-08T12:34:56.789Z"
}
```

2.6 データ読み込み機能

2.6.1 ポイントJSON読み込み

- **ファイル選択**: JSON形式ファイルのみ受付
- **前提条件**: 画像が事前に読み込まれていること
- **データ復元**: 既存ポイントデータの完全復元
- **座標変換**: 元画像座標からキャンバス座標への自動変換
- **マーカー復元**: `isMarker`プロパティに基づく表示切り替え
- **入力ボックス生成**: 通常ポイント用の入力フィールド自動生成
- **エラーハンドリング**: 形式不正・データ不備時のアラート表示

2.6.2 ルートJSON読み込み

- **ファイル選択**: JSON形式ファイルのみ受付
- **前提条件**: 画像が事前に読み込まれていること
- **データ復元**: ルート中間点と開始/終了ポイントIDの復元
- **座標変換**: 元画像座標からキャンバス座標への自動変換
- **UI更新**: 開始・終了ポイント入力フィールドへの自動反映
- **カウント更新**: 中間点数の表示更新
- **エラーハンドリング**: 形式不正時のアラート表示

3. ユーザーインターフェース

3.1 レスポンシブデザイン

- **ブレークポイント**: 768px
- **フォント**: システムフォント（Segoe UI等）、サイズ13px
- **カラーパレット**:
  - プライマリ: #3498db（青）
  - プライマリダーク: #2980b9（濃い青）
  - セカンダリ: #2c3e50（ダークグレー）
  - 成功: #27ae60（緑）
  - 成功ダーク: #229954（濃い緑）
  - 警告: #f39c12（オレンジ）
  - 警告ダーク: #d68910（濃いオレンジ）
  - 危険: #e74c3c（赤）
  - 危険ダーク: #c0392b（濃い赤）
  - 情報: #17a2b8（水色）

- 情報ダーク: #138496（濃い水色）

3.2 操作フロー

3.2.1 基本操作フロー

1. **画像読み込み**: PNG画像を選択（File System Access API優先）
2. **レイアウト選択**: サイドバー or オーバーレイ（右上切り替え）
3. **編集モード選択**: ポイント編集 or ルート編集
4. **ポイント配置**: 地図上クリックで配置
5. **ID入力**: ポイント編集時のみ必須（自動フォーカス）
6. **データ出力**: JSON形式でダウンロード

3.2.2 ポイント編集フロー

1. PNG画像読み込み → ポイント編集モード自動選択
2. 地図上クリック → 赤色マーカー表示 → ID入力ボックス自動フォーカス
3. ID入力（X-nn形式推奨） → Enter/blur で確定
4. 必要に応じてポイント追加/削除
5. 「ポイントをJSON出力」でダウンロード

3.2.3 ルート編集フロー

1. 事前にポイント編集モードで開始・終了ポイントを登録
2. ルート編集モードに切り替え
3. 開始・終了ポイントIDを入力フィールドに入力
4. 地図上クリックでルート中間点を配置（青色マーカー）
5. 「ルートをJSON出力」でダウンロード

3.3 視覚的フィードバック

- **ホバー効果**: ボタンと入力フィールドでの色変化
- **フォーカス効果**: 入力ボックスの拡大表示（1.05倍）
- **状態表示**: ポイント数/中間点数のリアルタイム更新
- **エラー表示**: アラートダイアログでの通知
- **自動削除**: 未入力ポイントの視覚的フィードバック
- **ID形式エラー**: 不正形式時の薄いピンク背景表示
- **マーカー色分け**:
  - 通常ポイント: 赤色
  - ルート中間点: 青色
  - 開始/終了指定ポイント: 青色
  - JSON読み込みマーカー: 小さい青色
- **キャンバスホバー**: 地図境界線のハイライト表示

4. 技術仕様

4.1 ファイル構成

```
PickPoints/
├── index.html          # メインHTMLファイル
├── app.js              # メインJavaScriptロジック
├── styles.css          # スタイルシート
├── docs/
│   ├── funcspec.md     # 本仕様書
│   └── UsersGuide-202508.md  # ユーザー手引
└── CLAUDE.md           # 開発ガイドライン
```

4.2 主要クラス・メソッド

4.2.1 PickPointsクラス

- **constructor()**: アプリケーション初期化、プロパティ設定
- **initializeEventListeners()**: 全DOM要素へのイベントリスナー設定
- **initializeLayoutManager()**: レイアウト管理とリサイズ対応初期化
- **handleImageSelection()**: 新しい画像選択処理（File System Access API優先）
- **handleImageLoad()**: 従来の画像読み込み処理（フォールバック用）
- **loadImageFromFile()**: ファイルオブジェクトから画像読み込み（共通処理）
- **setupCanvas()**: 画像サイズに応じたキャンバス設定
- **drawImage()**: 画像と全ポイントの描画
- **drawAllPoints()**: 全ポイント一括描画（色・サイズ制御）
- **addPoint()**: ポイント追加（ポイント編集モード）
- **addRoutePoint()**: 中間点追加（ルート編集モード）
- **removeTrailingEmptyUserPoints()**: 未入力ポイントの自動削除
- **createInputBox()**: 入力ボックス生成（フォーカス制御対応）
- **positionInputBox()**: 入力ボックス最適位置計算
- **formatPointId()**: ID形式の自動修正（X-nn形式）
- **convertFullWidthToHalfWidth()**: 全角文字の半角変換
- **isValidPointIdFormat()**: ID形式バリデーション
- **validateStartEndPoints()**: 開始・終了ポイントの存在チェック
- **generateRouteFilename()**: ルート用ファイル名生成
- **exportJSON()**: ポイントデータJSON出力
- **exportRouteJSON()**: ルートデータJSON出力
- **downloadJSONWithUserChoice()**: File System Access APIによる保存
- **loadPointsFromJSON()**: ポイントデータ読み込み
- **loadRouteFromJSON()**: ルートデータ読み込み
- **setLayout()**: レイアウト変更処理
- **setEditingMode()**: 編集モード変更処理

4.3 制約事項

- **PWA非対応**: デスクトップブラウザ推奨
- **オフライン動作**: 不可
- **ファイルサイズ**: 大容量PNG画像は動作に影響の可能性
- **ブラウザ互換性**: IE11以前は非対応
- **File System Access API**: Chrome 86+, Edge 86+のみ対応（その他ブラウザは従来方式）

5. データ仕様

5.1 座標系

- **Canvas座標**: 0,0が左上角の相対座標
- **出力座標**: 元画像サイズ基準の絶対座標
- **精度**: 整数値（Math.round処理）

5.2 ポイントデータ構造

フィールド	型	必須	説明
index	number	○	連番（1から開始）
id	string	○	識別子（最大4文字）
x	number	○	X座標（画像絶対座標）
y	number	○	Y座標（画像絶対座標）
isMarker	boolean	○	マーカーフラグ

5.3 ルートデータ構造

フィールド	型	必須	説明
type	string	○	固定値 "waypoint"
index	number	○	中間点番号
x	number	○	X座標
y	number	○	Y座標

6. 動作環境

6.1 推奨環境

- **OS**: Windows 10/11, macOS 10.15+, Ubuntu 18.04+
- **ブラウザ**: Chrome 90+, Firefox 88+, Edge 90+, Safari 14+
- **画面解像度**: 1024×768以上
- **メモリ**: 4GB以上推奨

6.2 開発環境

- **サーバー**: 静的ファイルサーバー（python -m http.server等）
- **デバッグ**: ブラウザ開発者ツール
- **バージョン管理**: Git推奨

7. 今後の拡張可能性

7.1 機能拡張案

- GPXファイル出力対応
- 複数画像の重ね合わせ表示
- ポイント間距離計算
- ルート描画機能
- 印刷機能
- ポイント検索・フィルタ機能
- バッチ処理機能

7.2 技術改善案

- TypeScript化
- PWA対応
- WebAssembly活用（大容量画像処理）
- オフライン対応
- データベース連携
- クラウドストレージ対応

作成日: 2025年8月11日
バージョン: 2.0
更新内容: 実装内容に基づく全面見直し、ID形式バリデーション、全角変換処理、レイアウト管理、エラーハンドリング詳細、操作フロー追加
作成者: Claude Code