

PointMarker 機能仕様書

1. プロジェクト概要

1.1 プロジェクト名

PointMarker

1.2 目的

ハイキングマップ画像上にポイント、スポット、ルートをマーキングし、構造化されたJSONデータとして出力するWebアプリケーション

1.3 対象ユーザー

- ・ハイキング・登山愛好者
- ・地理情報管理者
- ・マップデータ作成者

1.4 技術仕様

- ・**言語:** バニラJavaScript (ES6モジュール)
- ・**UI:** HTML5、CSS3、Canvas API
- ・**ファイル処理:** File System Access API (フォールバック: 従来のinput要素)
- ・**データ形式:** JSON
- ・**レスポンシブ対応:** CSS Flexbox、CSS Grid
- ・**ブラウザ要件:** ES6モジュール対応ブラウザ、ローカルサーバー必須 (CORS制限回避)

2. アーキテクチャ

2.1 フォルダ構造

```
PointMarker/
├── index.html          # メインHTMLファイル
├── styles.css           # スタイルシート
├── CLAUDE.md            # プロジェクト指針
├── README.md             # プロジェクト概要
└── docs/
    ├── funcspec.md       # 機能仕様書
    ├── UsersGuide-202508.md # ユーザーガイド (旧版)
    └── UsersGuide-202509.md # ユーザーガイド (最新版)
└── js/
    ├── app.js             # メインアプリケーション
    └── core/
        └── Canvas.js        # キャンバス描画管理
    └── data/
        ├── FileHandler.js   # ファイル操作
        ├── PointManager.js   # ポイント管理
        └── RouteManager.js    # ルート管理
```

```

|   └── SpotManager.js      # スポット管理
|   └── ui/
|       ├── InputManager.js    # 動的入力管理
|       └── LayoutManager.js    # レイアウト管理
|   └── utils/
|       ├── Coordinates.js     # 座標変換
|       ├── Validators.js       # バリデーション
|       ├── DragDropHandler.js  # ドラッグ&ドロップ処理
|       └── ResizeHandler.js    # リサイズ処理

```

2.2 設計パターン

- **モジュール分離**: ES6モジュールによる機能別分離
- **MVC風アーキテクチャ**: データ・UI・ビジネスロジック分離
- **コールバック駆動**: 疎結合なコンポーネント間通信
- **責任単一原則**: 各クラスが特定の責任のみを持つ設計
- **レスポンシブデザイン**: CSS変数とFlexboxによる柔軟なレイアウト

2.3 主要クラス構成

メインクラス

- **PointMarkerApp**: アプリケーション統合管理・イベント処理統合
- **CanvasRenderer**: キャンバス描画管理・レンダリング処理

データ管理層

- **PointManager**: ポイントデータ管理・永続化・検証
- **RouteManager**: ルートデータ管理・検証・開始終了ポイント管理
- **SpotManager**: スポットデータ管理・星形マーカー描画・名前管理
- **FileHandler**: ファイル操作・File System Access API統合・JSON処理

UI管理層

- **InputManager**: 動的入力フィールド管理・ポップアップ表示・フォーカス制御
- **LayoutManager**: レイアウト・モード状態管理・表示切り替え

ユーティリティ層

- **CoordinateUtils**: 座標系変換処理・マウス座標変換
- **Validators**: データ検証・フォーマット処理・ID補正
- **DragDropHandler**: ドラッグ&ドロップ機能の統合管理
- **ResizeHandler**: ウィンドウリサイズ処理の統合管理

3. 機能仕様

3.1 画像管理機能

3.1.1 画像読み込み

- **対応形式:** PNG形式のみ
- **読み込み方法:** File System Access API（フォールバック：従来のinput要素）
- **表示:** HTML5 Canvasによるレスポンシブ表示
- **座標系管理:** 画像座標とキャンバス座標の自動変換
- **自動スケーリング:** 表示領域に合わせた自動リサイズ
- **Faviconサポート:** ⚡ ピンアイコンによる視覚的アイデンティティ

3.1.2 画像表示制御

- **アスペクト比維持:** 元画像の比率保持
- **レスポンシブ対応:** ウィンドウサイズ変更時の自動調整・座標スケーリング
- **最大サイズ制限:** レイアウトモードに応じた適切なサイズ調整
- **デバウンス処理:** リサイズイベントの最適化処理

実装メソッド

- `PointMarkerApp.handleImageSelection()`: File System Access API画像選択
- `PointMarkerApp.processLoadedImage()`: 画像読み込み完了処理
- `CanvasRenderer.setImage()`: 描画対象画像設定
- `CanvasRenderer.setupCanvas()`: キャンバスサイズ調整
- `ResizeHandler.handleResize()`: 統合リサイズ処理

3.2 レイアウト管理機能

3.2.1 レイアウトモード

- **サイドバー mode:** 横並びレイアウト（デフォルト、推奨）
 - 画面右側にコントロールパネル固定表示
 - 画面左側にキャンバス表示
 - デスクトップ環境で最適な作業効率
- **オーバーレイ mode:** 地図上オーバーレイ表示
 - キャンバスがフルスクリーン表示
 - コントロールパネルが半透明オーバーレイ
 - モバイル環境や大画面表示時に有効
- **動的切り替え:** リアルタイムレイアウト変更・状態保持

3.2.2 編集モード

- **ポイント編集モード:** ポイント追加・編集・削除・移動
- **ルート編集モード:** ルート中間点追加・開始終了ポイント設定
- **スポット編集モード:** スポット追加・編集・削除・移動（四角形マーカー）

3.2.3 編集モード切り替え時のUI制御

- **ポイントIDポップアップ表示制御:**
 - ポイント編集モード: 表示
 - ルート編集モード: 表示（背景灰色）
 - スポット編集モード: 非表示

- **スポット入力ボックス表示制御:**
 - スポット編集モードでのみ表示
 - 他モードでは非表示

実装メソッド

- `LayoutManager.setLayout()`: レイアウト変更・UI再配置
- `LayoutManager.setEditMode()`: 編集モード変更・パネル表示制御
- `LayoutManager.updateEditModeDisplay()`: モード切り替え・パネル表示制御
- `InputManager.setEditMode()`: 入力フィールド表示制御

3.3 ポイント編集機能

3.3.1 ポイント操作

- **追加:** キャンバスクリック→動的ポップアップ入力ボックス生成
 - 新規ポイント作成時、自動的に入力フィールドにフォーカス
 - カーソル位置が末尾に設定される (insertion point最適化)
- **削除:** Escapeキー、または空入力でblur
- **編集:**
 - 既存ポイントクリック→対応する入力フィールドにフォーカス
 - リアルタイム入力表示 (文字入力中の即座反映)
 - 入力中は自動補正なし、blur時に補正実行
- **移動:** ドラッグ&ドロップによるポイント位置変更
 - **統合オブジェクト検出:** ポイント・スポットの統一検出システム
 - **ホバー表示:** ポイント上にマウスオーバー時、crosshairカーソル維持
 - **ドラッグ処理:** DragDropHandlerクラスによる統合ドラッグ管理
 - **制限事項:** ポイント編集モードでのみ有効

3.3.2 ポイントID管理

- **フォーマット:** X-nn形式 (英大文字1桁-数字2桁) 例 : A-01, Z-99
- **自動補正:** 全角→半角変換、小文字→大文字変換、0埋め処理
- **入力制御:**
 - 入力中 (input event) : 補正処理なし、リアルタイム表示
 - フォーカス離脱時 (blur event) : 自動補正実行
- **バリデーション:**
 - Validatorsクラスによる統一バリデーション
 - 不正形式時の視覚的エラーフィードバック
 - **重複チェック:** JSON出力時に同一ID存在をエラー検出

3.3.3 動的ポップアップUI

- **ポップアップコンテナ:** 入力フィールド+コンテナの統合デザイン
- **編集時スタイル:** より濃い青色枠線+シャドウ効果で編集状態強調
- **位置最適化:** 画面端を考慮した自動位置調整・重なり回避
- **一貫性:** 統一されたスタイルング関数による管理

3.3.4 ルート編集・スポット編集モード時のUI制御

- **ルート編集モード:**
 - 入力フィールド無効化 (disabled状態)
 - 背景色統制: 統一されたスタイル関数による管理
- **スポット編集モード:**
 - ポイントIDポップアップを完全に非表示 (display: none)

3.3.5 一括操作

- **全ポイントクリア:** 確認なし即座削除・UI状態リセット
- **ポイントID名補正:** 全ポイント一括フォーマット+空ポイント削除
- **JSON出力・読み込み:** FileHandlerクラスによる統合ファイル操作

実装メソッド

- `PointManager.addPoint()`: ポイント追加・座標設定
- `PointManager.updatePointId()`: ポイントID更新・検証
- `PointManager.formatAllPointIds()`: 全ポイント一括補正
- `InputManager.createInputBox()`: 動的ポップアップ生成・イベント設定
- `InputManager.updateInputsState()`: 編集モード別UI制御
- `PointMarkerApp.findObjectAtMouse()`: 統合オブジェクト検出
- `DragDropHandler.startDrag()`: ドラッグ開始処理

3.4 スポット編集機能

3.4.1 スポット操作

- **追加:** キャンバスクリック→スポット作成・動的名前入力ボックス生成
 - 四角形マーカー (■) での視覚的表示 (青色)
 - 新規スポット作成時、自動的に入力フィールドにフォーカス
- **削除:** Escapeキー、または空入力でblur
- **編集:**
 - 既存スポットクリック→対応する入力フィールドにフォーカス
 - リアルタイム名前表示 (文字入力中の即座反映)
- **移動:** DragDropHandlerクラスによる統合ドラッグ処理
 - **制限事項:** スポット編集モードでのみ有効

3.4.2 スポット名前管理

- **入力制御:**
 - 最大10文字制限
 - リアルタイム表示・即座反映
 - trim処理による空白文字除去
- **バリデーション:** 基本的な文字列検証のみ

3.4.3 スポット視覚表示

- **マーカー形状:** 四角形 (■)
- **サイズ:** ポイントより大きめ (12px、視認性重視)
- **色:** 青色系 (#0066ff、ポイントの赤色と区別)
- **描画処理:** CanvasRenderer.drawSquare()による統一描画

3.4.4 スポット編集モード時のUI制御

- **ポイントIDpopupアップ:** 完全に非表示
- **スポット入力ボックス:** 表示・編集可能

3.4.5 一括操作

- **全スポットクリア:** 確認なし即座削除・UI状態リセット
- **JSON出力・読み込み:** FileHandlerクラスによる統合処理

実装メソッド

- `SpotManager.addSpot()`: スポット追加・座標設定
- `SpotManager.updateSpotName()`: スポット名前更新
- `SpotManager.findSpotAt()`: マウス座標でのスポット検出
- `InputManager.createSpotInputBox()`: 動的スポット名前入力ボックス生成
- `InputManager.updateSpotInputsState()`: スポット入力表示制御
- `CanvasRenderer.drawSpots()`: スポット描画処理

3.5 ルート編集機能

3.5.1 ルート構成要素

- **開始ポイント:** 既存ポイントIDから選択 (X-nn形式自動補正)
- **終了ポイント:** 既存ポイントIDから選択 (X-nn形式自動補正)
- **中間点:** キャンバスクリックルートポイント追加 (順次蓄積)

3.5.2 ルート編集制限・UI制御

- **ポイント編集禁止:** 既存ポイントの移動・削除・ID編集を完全制限
- **スポット編集禁止:** 既存スポットの編集を完全制限
- **入力フィールド制御:**
 - ポイント入力フィールドをdisabled状態に設定
 - 統一されたスタイル関数による視覚的フィードバック

3.5.3 開始・終了ポイント入力制御とバリデーション

- **input時処理:** フォーマット処理スキップ・リアルタイム表示
- **blur時処理:** X-nn形式自動補正・存在確認・重複チェック・視覚フィードバック更新
- **統合バリデーション:**
 - Validatorsクラスによる形式チェック
 - 統一されたスタイル関数によるエラー表示
 - 重複チェック機能の強化

3.5.4 ルート操作

- 中間点追加: 順次クリック→ルートポイント蓄積・カウンター更新
- 中間点クリア: 全中間点一括削除・UI状態リセット
- JSON出力・読み込み: 統合検証機能付きファイル操作

実装メソッド

- RouteManager.addRoutePoint(): 中間点追加・座標記録
- RouteManager.setStartPoint() / setEndPoint(): 開始終了ポイント設定・フォーマット
- RouteManager.validateStartEndPoints(): ポイント存在検証・重複検証
- PointMarkerApp.updateBothRoutePointsValidation(): 統合バリデーション
- PointMarkerApp.clearInputElementStyles(): 統一スタイル管理
- PointMarkerApp.setInputElementError(): 統一エラー表示

3.6 データ検証機能

3.6.1 統合バリデーション

- Validatorsクラス: 全バリデーション処理の統一管理
- ID形式検証: X-nn形式の厳密チェック (正規表現: /^[A-Z]-\d{2}\$/)
- 重複ID検証: JSON出力前の同一ID存在チェック・エラー表示
- 空ポイント検出: ID未入力ポイントの自動検出・削除

3.6.2 ルート検証

- 開始ポイント存在確認: 指定IDが登録済みポイントとして存在するか検証
- 終了ポイント存在確認: 指定IDが登録済みポイントとして存在するか検証
- 重複チェック: 開始・終了ポイントが同一IDでないか検証
- 中間点数確認: 最低1つ以上の中間点存在チェック
- 統合検証: 全条件クリア確認・詳細エラーメッセージ生成

3.6.3 スポット検証

- 名前検証: 基本的な文字列検証・trim処理
- 重複チェック: 現在は実装なし (必要に応じて拡張可能)

実装メソッド

- Validators.isValidPointIdFormat(): 統一ポイントID形式検証
- Validators.formatPointId(): 統一フォーマット処理
- PointMarkerApp.updateBothRoutePointsValidation(): ルートポイント総合検証
- PointMarkerApp.checkDuplicatePointIds(): 重複検証・分析

3.7 ファイル操作機能

3.7.1 画像ファイル処理

- PNG読み込み: File System Access API優先・フォールバック対応

- **ファイル形式検証**: MIME typeによるPNG形式確認
- **エラーハンドリング**: 不正ファイル・キャンセル時の適切な処理

3.7.2 JSON処理

- **統合ファイル操作**: FileHandlerクラスによる一元管理
- **ポイントJSON**: ポイント専用データ構造での入出力
- **ルートJSON**: ルート専用データ構造での入出力
- **スポットJSON**: スポット専用データ構造での入出力
- **ファイル名自動生成**:
 - ポイント: {画像名}_points.json
 - ルート: {画像名}_route_{開始ポイント}_to_{終了ポイント}.json
 - スポット: {画像名}_spots.json
- **高度保存機能**: File System Access API対応ブラウザでの利用

実装メソッド

- `FileHandler.selectImage()`: File System Access API画像選択・フォールバック
- `FileHandler.saveJSONwithUserChoice()`: JSON保存・ファイル名指定
- `FileHandler.loadJsonFile()`: JSONファイル読み込み・パース処理
- `FileHandler.exportPointData()`: ポイントJSON生成・座標変換
- `FileHandler.exportRouteData()`: ルートJSON生成・座標変換
- `FileHandler.exportSpotData()`: スポットJSON生成・座標変換

4. データ構造

4.1 ポイントJSON形式

```
{  
    "totalPoints": 3,  
    "imageReference": "sample.png",  
    "imageInfo": {  
        "width": 1920,  
        "height": 1080  
    },  
    "points": [  
        {  
            "index": 1,  
            "id": "A-01",  
            "imageX": 245,  
            "imageY": 387,  
            "isMarker": false  
        }  
    ],  
    "exportedAt": "2025-09-14T10:30:00.000Z"  
}
```

4.2 ルートJSON形式

```
{
  "routeInfo": {
    "startPoint": "A-01",
    "endPoint": "B-03",
    "waypointCount": 5
  },
  "imageReference": "sample.png",
  "imageInfo": {
    "width": 1920,
    "height": 1080
  },
  "points": [
    {
      "type": "waypoint",
      "index": 1,
      "imageX": 320,
      "imageY": 450
    }
  ],
  "exportedAt": "2025-09-14T10:45:00.000Z"
}
```

4.3 スポットJSON形式

```
{
  "totalSpots": 2,
  "imageReference": "sample.png",
  "imageInfo": {
    "width": 1920,
    "height": 1080
  },
  "spots": [
    {
      "index": 1,
      "name": "展望台",
      "imageX": 580,
      "imageY": 320
    }
  ],
  "exportedAt": "2025-09-14T11:00:00.000Z"
}
```

4.4 座標系管理

- **画像座標系**: 元PNG画像の実際のピクセル座標（永続化用・JSON出力用）
- **キャンバス座標系**: 表示用にスケールされた座標（描画用・UI配置用）
- **スクリーン座標系**: ブラウザ内の絶対位置座標（ポップアップ配置用）
- **マウス座標系**: ブラウザイベントから得られる座標（入力処理用）

- **統合座標変換:** CoordinateUtilsクラスによる一元管理

5. UI/UX仕様

5.1 レスポンシブデザイン

- **ブレークポイント:** 768px (モバイル対応境界)
- **レイアウト:** Flexboxベースの柔軟な配置・自動調整
- **フォント:** システムフォント優先 (Segoe UI, Tahoma, Geneva, Verdana) ・日本語対応
- **リサイズ処理:** ResizeHandlerクラスによるデバウンス最適化

5.2 アクセシビリティ

- **キーボード操作:** Tab移動・Escape削除対応
- **ARIA属性:** スクリーンリーダー対応・適切なラベリング
- **カラーコントラスト:** WCAG準拠の配色設計
- **フォーカス管理:** 視覚的フォーカスインジケーター・論理的Tab順序

5.3 ビジュアル仕様

- **カラーパレット:** CSS変数による統一配色・テーマ一貫性
- **Favicon:** ⚡ ピンアイコンによる視覚的アイデンティティ
- **ボタンデザイン:** 機能別カラーコーディング・統一レイアウト
 - クリア系: 赤色 (危険操作)
 - 出力系: 緑色 (成功・完了)
 - 通常操作: 青色系
- **フィードバック:** ホバー効果・状態変化アニメーション・transition効果
- **統一エラー表示:**
 - 形式エラー: 薄いピンク背景 + 赤枠
 - 存在エラー: 赤枠のみ
 - 重複エラー: 両フィールド赤枠

5.4 動的UI要素

- **ポップアップ入力ボックス:** ポイント・スポット位置に動的配置
- **位置最適化:** 画面端を考慮した自動位置調整・重なり回避
- **状態管理:** モード切り替えに応じた表示制御・UI一貫性
- **マウスカーソル制御:**
 - ポイント・スポット上: crosshairカーソル維持
 - 通常時: crosshairカーソル
- **統合オブジェクト検出:** findObjectAtMouse()による効率的検出

6. パフォーマンス仕様

6.1 描画パフォーマンス

- **Canvas最適化:** 必要時のみ再描画・不要な描画処理回避
- **座標キャッシュ:** スケール計算結果の再利用・計算負荷軽減
- **イベント効率化:** デバウンス処理によるリサイズ最適化

- **統合描画処理:** CanvasRendererクラスによる効率的描画

6.2 メモリ管理

- **オブジェクト再利用:** 不要なオブジェクト生成回避・GC負荷軽減
- **イベントリスナー管理:** 適切な削除とメモリリーク防止
- **DOM要素管理:** 動的要素の適切な作成・削除サイクル
- **コード分割:** 機能別クラス分離による最適化

6.3 コード効率化（リファクタリング成果）

- **機能統合:** 類似処理の統一化によるコード重複削除
- **責任分離:** 専用ハンドラークラスによる処理の最適化
- **統一インターフェース:** 一貫したAPI設計による保守性向上

7. ブラウザ対応

7.1 対応ブラウザ

- **Chrome:** 86+ (File System Access API対応・推奨環境)
- **Firefox:** 最新版（フォールバック動作・基本機能利用可能）
- **Safari:** 14+（フォールバック動作・基本機能利用可能）
- **Edge:** 86+ (Chromiumベース・File System Access API対応)

7.2 必要な機能

- **ES6モジュール:** import/export構文・必須要件
- **Canvas API:** 2D描画機能・画像表示・ポイント/スポット描画
- **File API:** ファイル読み込み・基本的なファイル操作
- **File System Access API:** 高度なファイル操作（オプション・Chrome系ブラウザ）
- **ローカルサーバー:** CORS制限回避のため必須（python -m http.server等）

8. セキュリティ仕様

8.1 ファイルアクセス

- **同一オリジン制限:** ES6モジュール使用によるCORS制限・ローカルサーバー必須
- **ファイル形式検証:** MIME typeによる安全性確認・PNG/JSON形式限定
- **サニタイゼーション:** 入力値の適切な処理・XSS対策

8.2 データ保護

- **ローカル処理:** 完全クライアントサイド処理・サーバー送信なし・プライバシー保護
- **XSS対策:** 動的コンテンツの適切なエスケープ・DOM操作安全性
- **入力検証:** 厳密なバリデーション・不正データ拒否
- **統一検証:** Validatorsクラスによる一貫したセキュリティ対策

9. 拡張性

9.1 モジュール拡張

- **プラグイン機構:** コールバックベースの拡張ポイント・疎結合設計
- **カスタムバリデーター:** 独自検証ルール追加可能
- **出力形式拡張:** JSON以外のフォーマット対応可能性
- **ハンドラー拡張:** 新しい機能ハンドラーの追加が容易

9.2 機能拡張候補

- **GPS連携:** 位置情報との統合・実世界座標変換
- **マルチレイヤー:** 複数画像の重ね合わせ・レイヤー管理
- **テンプレート機能:** ポイント・スポット配置パターンの保存・再利用
- **データベース連携:** 大量データ管理・検索機能

10. 運用・保守仕様

10.1 開発環境

- **ローカルサーバー:** `python -m http.server 8000` または `npx serve .`
- **デバッグツール:** ブラウザ開発者ツール・コンソール出力
- **バージョン管理:** Git・変更履歴管理

10.2 トラブルシューティング

- **CORS エラー:** ローカルサーバー起動確認・直接ファイル開放禁止
- **File System Access API:** Chrome系ブラウザ推奨・フォールバック動作確認
- **メモリリーク:** 大量ポイント・スポット作成時の動作確認・ブラウザ再起動推奨
- **Favicon 404エラー:** SVG + ICO形式の統合対応により解決済み

最終更新: 2025年9月14日 バージョン: 5.0 作成者: Claude Code Analysis 更新内容:

- リファクタリング後のアーキテクチャ更新 (DragDropHandler、ResizeHandler追加)
- 統合オブジェクト検出システムの実装 (findObjectAtMouse)
- 統一バリデーション・スタイル管理機能の実装
- スポットマーカーを星形から四角形に変更 (青色#0066ff)
- コード効率化成果の反映 (重複削除、責任分離、統一インターフェース)
- Favicon対応 (♀ ピンアイコン、404エラー解消)
- ファイル構造の最新化 (utils配下の新クラス追加)
- パフォーマンス仕様にコード効率化成果を追加