公共交通オープンデータチャレンジ2025 ― 次審査ヒアリング資料

発表概要

- **作品名**: STAYLINE
- **発表時間**: 3分間のプレゼンテーション + 5分間の質疑応答

3分間プレゼンテーション台本

【0:00〜0:30】導入・問題提起

本日は貴重なお時間をいただきありがとうございます。

「その格安ホテル、交通費と移動時間を含めても本当にお得ですか？」

私が開発した「STAYLINE」は、この問いに答えるサービスです。

【0:30〜1:00】開発背景

開発のきっかけは、私自身がライブやコンサートに頻繁に参加する中での実体験です。

東京ドームや武道館周辺のホテルは、人気イベント時にはすぐに満室になるか、価格が高騰します。「少し離れても電車一本で行ける場所」を探そうとしても、既存の予約サイトは「エリア単位」での検索が主流で、「沿線軸」での横断検索が非常に困難でした。

不動産サイトでは当たり前の「沿線・駅で探す」機能を、宿泊検索にも導入すれば、選択肢が何倍にも広がるはずだ――これが本サービスの原点です。

【1:00〜1:45】デモ・機能説明

（画面を表示しながら）

実際の画面をご覧ください。

1. **目的地を選択**：例えば「後楽園駅（東京ドーム）」を選びます
2. **日付と人数を設定**：タップ操作のみで完結します
3. **検索実行**

すると、東京メトロ全路線の駅周辺ホテルが一括検索され、以下の情報が表示されます：

- **実質価格**：宿泊費＋往復交通費の合計
- **移動時間**：電車所要時間＋徒歩時間
- **コスパ指数**：1分移動時間を延ばすといくら節約できるか
- **終電・始発時刻**：当日の終電と翌日の始発

特徴的なのは、徒歩時間の計算です。直線距離ではなく、OSRMを使用して実際の道路ネットワークに基づいた経路を計算しています。「直線では近いが線路を渡れず遠回りが必要」といったミスマッチを防いでいます。

【1:45〜2:30】技術的特徴・公共交通データの活用

ODPTデータの活用として、以下を組み合わせています：

1. **運賃データ（odpt:RailwayFare）**：IC運賃・切符運賃を取得
2. **時刻表データ（odpt:TrainTimetable）**：駅間の正確な所要時間を計算
3. **駅時刻表（odpt:StationTimetable）**：終電・始発の自動表示

これらを楽天トラベルAPIのホテル情報、OSRMの徒歩ルート計算と動的に結合し、**「移動コスト」という複合指標でのリアルタイムソート**を実現しました。

【2:30〜3:00】社会的意義・まとめ

本サービスは、単なる便利ツールではありません。

公共交通の価値再発見を促し、観光ガイドに載らない「穴場駅」の魅力を可視化することで、**オーバーツーリズムの緩和**と**沿線地域の活性化**に寄与します。

また、遠くの安いホテルを選ぶことで浮いた宿泊費の一部が交通費として鉄道事業者に還元される——**利用者と事業者のWin-Winな経済循環**を生み出します。

以上です。ご清聴ありがとうございました。

想定質問と回答

技術的質問

Q1. 経路探索のアルゴリズムについて詳しく教えてください

回答:

ダイクストラ法に近い優先度付きキューを用いたBFS（幅優先探索）を実装しています。

具体的には：

1. 目的地駅を起点として、全駅への到達時間を逆算
2. 時刻表データから隣接駅間の実際の所要時間を取得しエッジの重みとして使用
3. 乗り換えは同一駅名の他路線駅への遷移として扱い、乗り換え時間5分を加算
4. 最大乗り換え回数（デフォルト1回）を超える経路は探索打ち切り

```
// 疑似コード
queue.push({ id: destinationId, time: 0, transfers: 0 });
while (queue.length > 0) {
    queue.sort((a, b) => a.time - b.time); // 優先度キュー
    const current = queue.shift();
    // 隣接駅への移動、乗り換え駅への移動を探索
}
```

計算量を抑えるため、時刻表データは平日昼間の代表的な列車20本のみを解析し、駅間時間を事前キャッシュしています。

Q2. ODPT APIのレート制限にはどう対処していますか？

回答:

以下の対策を講じています：

1. **プリビルドによる静的データ生成**
   - ビルド時にODPT APIから取得したデータをJSONファイルとして保存
   - 実行時はまず静的ファイルを読み込み、APIコールを削減
2. **リクエスト間隔の制御**
   - API呼び出し間に200ms〜1000msのディレイを挿入
   - ステータス429（レート制限）を検知したら該当路線をスキップ
3. **積極的なキャッシュ戦略**
   - 運賃データ、時刻表データ、徒歩ルートをMapでキャッシュ
   - 同一セッション内での重複リクエストを排除

Q3. 徒歩時間の計算にOSRMを使用した理由は？

回答:

直線距離での計算には限界があります。例えば：

- 線路を渡れず大きく迂回が必要なケース
- 川や高速道路で分断されているケース
- 地形的な障壁があるケース

OSRMはOpenStreetMapのデータを基に、実際に歩ける道路ネットワークに沿った経路を計算します。これにより「徒歩5分と思ったら実際は15分かかった」というユーザーの不満を防止できます。

実装詳細:

```
const url = `${OSRM_FOOT_URL}/${lng},${lat};${destLng},${destLat}?overview=false`;
const response = await fetch(url);
const route = response.routes[0];
return Math.round(route.duration / 60); // 秒→分に変換
```

OSRMの公開サーバー（routing.openstreetmap.de）を使用しており、追加コストなしで高精度な徒歩時間を取得できます。

Q4. 「コスパ指数」の計算ロジックを説明してください

回答:

コスパ指数は「1分移動時間を延ばすことで何円節約できるか」を表す独自指標です。

```
コスパ指数 = 節約額 ÷ 追加移動時間（分）
```

具体的な計算：

1. **ベースライン設定**: 目的地駅（または最寄り駅）の最安ホテル
2. **節約額**: ベースラインの宿泊費 - (当該ホテルの宿泊費 + 往復運賃)
3. **追加移動時間**: 当該ホテルへの総移動時間 - ベースラインへの総移動時間

例：

- ベースライン: 後楽園駅、宿泊費10,000円、移動時間0分
- 候補A: 茗荷谷駅、宿泊費7,000円、運賃340円、移動時間15分
- 節約額 = 10,000 - (7,000 + 340) = 2,660円
- コスパ指数 = 2,660 ÷ 15 = 約177円/分

「15分多く移動すると2,660円節約でき、1分あたり177円の価値がある」と解釈できます。

Q5. リアルタイムのソート処理でパフォーマンスは問題ありませんか？

回答:

以下の最適化を行っています：

1. **インクリメンタル更新**
   - 検索中も結果を順次表示（Progressive Rendering）
   - 全結果を待たずにユーザーは確認可能
2. **Refを使用したリアルタイムソート**
   - useStateの更新遅延を回避するため、ソートモードをuseRefでも保持
   - 非同期ループ内でも常に最新のソートモードを参照
3. **重複排除の効率化**
   - 同一ホテルは「最短時間」と「最安コスト」の2パターンのみ保持
   - MapとSetを活用してO(n)で処理

```
const shortestTimeMap = new Map<string, Result>();
const lowestCostMap = new Map<string, Result>();
// 1回のループで両方を計算
```

Q6. 楽天トラベルAPIとODPTデータの結合はどのように行っていますか？

回答:

結合のキーは「位置情報（緯度・経度）」です。

処理フロー:

1. ODPTから各駅の座標を取得
2. 目的地駅への経路計算を実行し、各駅の所要時間・運賃を算出
3. 各駅の座標を中心に楽天トラベルAPIでホテル検索（半径1km程度）
4. ホテルごとに最寄り駅との徒歩時間をOSRMで計算
5. 全データを結合して ExtendedResult オブジェクトを生成

```
interface ExtendedResult {
    name: string;           // 駅名（ODPT）
    hotel: HotelInfo;       // ホテル情報（楽天）
    icFare: number;         // 運賃（ODPT）
    trainTime: number;      // 電車時間（ODPT時刻表）
    walkTime: number;       // 徒歩時間（OSRM）
    totalCost: number;      // 計算値
}
```

事業・社会的質問

Q7. なぜ東京メトロに限定したのですか？JRや私鉄は？

回答:

東京メトロを選定した理由は3つあります：

1. **イベント会場の網羅性**: 東京ドーム（後楽園）、日本武道館（九段下）、代々木第一体育館（代々木公園）など主要会場が沿線にある
2. **ちょうどいい距離感**: 都心を網の目のように結ぶため、数駅離れるだけで静かでリーズナブルなエリアに到達できる
3. **ODPTデータの充実度**: 運賃・時刻表・駅情報が体系的に提供されており、実装しやすかった

今後の展開として、JR・私鉄への拡張は技術的には可能です。ODPTに他社のデータも含まれているため、同じアーキテクチャで対応できます。

Q8. 収益モデルは考えていますか？

回答:

現時点では収益化は考えておらず、公共交通の利用促進という社会貢献が主目的です。

仮に収益化を検討する場合は：

- 楽天トラベルのアフィリエイトプログラム
- ホテル予約への送客手数料

ただし、本サービスの価値は「公共交通データを活用した新しい検索体験」にあるため、広告過多でUXを損なうことは避けたいと考えています。

Q9. 類似サービスとの差別化ポイントは？

回答:

観点	既存サービス	本サービス
検索軸	エリア・地図	沿線・駅
価格表示	宿泊費のみ	交通費込みの総額
移動時間	直線距離ベース	時刻表＋道路ネットワーク
終電情報	なし	自動表示

不動産サイトでは「沿線検索」は当たり前ですが、旅行予約サイトでは実装されていません。この発想の転換が最大の差別化です。

Q10. オーバーツーリズム緩和にどう貢献しますか？

回答:

本サービスは以下のメカニズムで貢献します：

1. **需要の分散化**
   - 「渋谷駅周辺」に集中していた検索を「渋谷まで10分以内」に拡大
   - 結果として中野、荻窪、三軒茶屋などへの宿泊が促進
2. **穴場の可視化**
   - 観光ガイドに載らない駅でも「実はアクセス抜群」を数字で証明
   - 例：「北参道駅は原宿まで1駅2分」
3. **経済効果の分散**
   - 宿泊者が滞在する地域の飲食店・商店街への波及効果

デモ・UXに関する質問

Q11. キーボード入力を極力排除した理由は？

回答:

主なユースケースは「イベント会場で空きホテルを探す」場面です。

- 混雑した会場で立ったまま操作
- 片手でスマートフォンを持ちながら検索
- 急いでいる状況

このため、日付選択・人数選択・駅選択をすべてタップ操作で完結できるUI設計としました。駅名検索もオートコンプリート形式で、最小限の入力で候補が表示されます。

Q12. Neumorphism Designを採用した意図は？

回答:

Neumorphismは「触りたくなる」質感を生み出すデザイン手法です。

ホテル検索は「たくさんの情報を比較する」行為であり、視覚的な疲労が課題です。柔らかい陰影とシンプルな配色により、長時間の閲覧でも目が疲れにくいUIを目指しました。

技術的にはTailwind CSS v4とカスタムCSSを組み合わせて実装しています。

備考

- デモ環境は事前にローカルで起動しておく（ `npm run dev` ）
- ネットワーク接続を確認（ODPT API、楽天API、OSRMへのアクセス必須）
- バックアップとして静的データ（networkData.json）も用意済み