Australia Trip Scheduler - Technical Documentation

Table of Contents

Project Overview

Repository: https://github.com/petehep/interactive-Aus-map
Live URL: https://petehep.github.io/interactive-Aus-map/
Purpose: Interactive trip planning application for Australia, featuring route optimization, campsite discovery, and personal travel tracking.

Key Features

- Interactive map of Australia with OpenStreetMap data
- Route planning with OSRM routing engine
- Favorites and visited places tracking
- Campsite, fuel station, water point, and dump station discovery
- Firebase authentication for multi-user access
- Mobile-responsive design

Technology Stack

Frontend Framework

- **React 18.3.1** - UI framework
- **TypeScript 5.6.3** - Type safety and development experience
- **Vite 5.4.8** - Build tool and dev server

Mapping & GIS

- **Leaflet 1.9.4** - Interactive map library
- **React-Leaflet 4.2.1** - React bindings for Leaflet
- **React-Leaflet-Cluster 2.1.0** - Marker clustering for performance
- **Overpass API** - OpenStreetMap data queries
- **OSRM (Open Source Routing Machine)** - Route calculation

Backend Services

- **Firebase 10.14.0**
  - Authentication (Email/Password)
  - Firestore (database, prepared but not yet implemented)
- **Nominatim** - Geocoding service

Weather Integration

- **MeteoBlue** - 14-day weather forecasts

Hosting

- **GitHub Pages** - Static site hosting

Project Structure

```
aus-trip-scheduler/
├── public/                      # Static assets
├── src/
│   ├── components/             # React components
│   │   ├── Favorites.tsx       # Favorites list with state grouping
│   │   ├── Itinerary.tsx       # Trip itinerary display
│   │   ├── Login.tsx           # Authentication UI
│   │   ├── MapView.tsx         # Main map component (759 lines)
│   │   └── VisitedPlaces.tsx   # Visited places modal
│   ├── types/
│   │   └── react-leaflet-cluster.d.ts  # Type definitions
│   ├── App.tsx                 # Main application component (682 lines)
│   ├── firebase.ts             # Firebase configuration
│   ├── index.css               # Global styles and responsive design
│   ├── main.tsx               # React entry point
│   └── vite-env.d.ts          # Vite type definitions
├── index.html                   # HTML entry point
├── package.json                # Dependencies and scripts
├── tsconfig.json              # TypeScript configuration
├── vite.config.ts             # Vite build configuration
└── .github/workflows/         # GitHub Actions for deployment
```

Core Components

App.tsx (Main Application)

Location: `src/App.tsx`
Lines of Code: 682
Purpose: Root component managing application state and routing

Key State Variables

```
// Authentication
const [user, setUser] = useState<any>(null)
const [authLoading, setAuthLoading] = useState(true)

// Trip Planning
const [itinerary, setItinerary] = useState<ItineraryItem[]>([])
const [route, setRoute] = useState<RouteResult>(null)
const [startLocation, setStartLocation] = useState<{lat, lon, name}>()

// User Data (localStorage-backed)
const [favorites, setFavorites] = useState<Place[]>([])
const [visitedPlaces, setVisitedPlaces] = useState<Place[]>([])

// UI State
const [showFuelStations, setShowFuelStations] = useState(false)
const [showDumpPoints, setShowDumpPoints] = useState(false)
const [showWaterPoints, setShowWaterPoints] = useState(false)
const [showSmallTownsOnly, setShowSmallTownsOnly] = useState(false)
const [showCampsites, setShowCampsites] = useState(true)
const [showVisitedModal, setShowVisitedModal] = useState(false)
```

Key Functions

Route Calculation

```
const updateRoute = async () => {
  // Uses OSRM Trip API
  // Endpoint: https://router.project-osrm.org/trip/v1/driving/{coords}
  // Parameters: source=first, roundtrip=false, overview=full
}
```

Geocoding

```
const geocodeStart = async () => {
  // Uses Nominatim API
  // Endpoint: https://nominatim.openstreetmap.org/search
  // Converts place names to coordinates
}
```

Favorites Management

```
const toggleFavorite = (place: Place) => {
  // Adds/removes from favorites array
```

```
  // Syncs to localStorage: 'trip-favorites'
}

const toggleVisited = (id: string) => {
  // Marks place as visited
  // Updates both favorites and visitedPlaces
  // Stores timestamp
}

const unvisitPlace = (id: string) => {
  // Removes from visited list
  // Updates both storage locations
}
```

Data Persistence - localStorage keys: - `trip-favorites` - Favorites array - `trip-visited-places` - Visited places with timestamps

---

MapView.tsx (Map Component)

Location: `src/components/MapView.tsx`
Lines of Code: 937
Purpose: Interactive Leaflet map with marker clustering and data fetching

Type Definitions

```
export type Place = {
  id: string
  name: string
  type: 'city' | 'town' | 'village' | 'hamlet' | 'locality' | 'attraction'
  lat: number
  lon: number
  population?: number
  visited?: boolean
  visitedAt?: number
}
```

Map Configuration

- **Center:** [-25.2744, 133.7751] (Center of Australia)
- **Initial Zoom:** 4
- **Min Zoom:** 3
- **Max Zoom:** 15
- **Tile Layer:** OpenStreetMap standard tiles

Marker Icons (CSS Filters)

```
.purple-marker { filter: hue-rotate(90deg) saturate(1.8) brightness(1.1); }    /* Free
        camps */
.green-marker { filter: hue-rotate(220deg) saturate(2.5) brightness(0.95); }   /* Paid
        camps */
```

```
.orange-marker { filter: hue-rotate(330deg) saturate(2.0) brightness(1.1); }   /* Fuel
        */
.blue-marker { filter: hue-rotate(180deg) saturate(2.0) brightness(1.0); }     /*
        Dumps */
.cyan-marker { filter: hue-rotate(150deg) saturate(2.5) brightness(1.2); }     /*
        Water */
```

Progressive Data Loading

Zoom-based Place Loading:

```
if (zoom < 4) {
  // Show nothing
} else if (zoom < 6) {
  // Major cities only (population > 1M)
  qParts.push(`node["place"="city"]["population"~"[0-9]{6,}"]`)
} else if (zoom < 8) {
  // All cities and towns
  qParts.push(`node["place"="city"]`)
  qParts.push(`node["place"="town"]`)
} else {
  // Cities, towns, villages (+ hamlets/localities with Small Towns filter)
  qParts.push(`node["place"~"city|town|village"]`)
}
```

Campsite Loading: - Minimum zoom: 8 - Queries both `tourism=camp_site` and `tourism=caravan_site` - Separates into free and paid based on `fee` tag

Fuel/Dump/Water Points: - Only shown when explicitly toggled - Searches within 10km radius of itinerary stops - Queries: - Fuel: `amenity=fuel` - Dumps: `amenity=sanitary_dump_station` - Water: `amenity=drinking_water` OR `amenity=water_point`

Debouncing & Rate Limiting

- Map movement debounced by 1000ms
- Prevents excessive Overpass API calls
- Mitigates 429 (Too Many Requests) errors

Favorites.tsx

Location: `src/components/Favorites.tsx`
Purpose: Display favorites organized by Australian state

State Detection Algorithm

```
function getAustralianState(lat: number, lon: number): string {
  // Uses coordinate boundaries to determine state
  // NSW, VIC, QLD, SA, WA, TAS, NT, ACT
  // Rough boundaries based on lat/lon ranges
}
```

Features

- Groups favorites by state
- Alphabetical sorting within states
- Click name to center map
- Weather and Google Maps links
- Add to itinerary button
- Visited toggle
- Remove from favorites

VisitedPlaces.tsx

Location: `src/components/VisitedPlaces.tsx`
Purpose: Modal displaying all visited places

Features

- Modal overlay with backdrop
- State-organized display
- Visit timestamps with date formatting
- Unvisit functionality
- Click name to center map and close modal
- Persists independently from favorites

Login.tsx

Location: `src/components/Login.tsx`
Purpose: Firebase authentication UI

Features

- Email/password authentication
- Toggle between sign-up and sign-in
- Form validation (min 6 char password)
- Error display
- Loading states
- Full-screen modal design

Data Management

Current Implementation: localStorage

Storage Keys

trip-favorites

```
Place[] = [
  {
    id: "node/123456",
    name: "Sydney",
    type: "city",
    lat: -33.8688,
    lon: 151.2093,
    population: 5312000,
    visited: true,
    visitedAt: 1700000000000
  }
]
```

trip-visited-places

```
Place[] = [
  // Same structure as favorites
  // Independent list that persists even when favorites cleared
]
```

Data Flow

1. **User Action** (e.g., toggle favorite)
2. **State Update** (React setState)
3. **localStorage Sync** (JSON.stringify and setItem)
4. **Re-render** (React updates UI)

Limitations

- Data stored per-device only
- No cross-device sync
- No backup/recovery
- Limited storage (~5-10MB)

Future Implementation: Firestore

Prepared Structure

```
users/
  {userId}/
    favorites/
      {placeId}/
        - id
        - name
        - type
        - lat
        - lon
        - visited
        - visitedAt
```

```
    visited/
      {placeId}/
        - (same structure)

    itineraries/
      {itineraryId}/
        - name
        - created
        - places[]
        - route
```

Firebase Configuration

File: `src/firebase.ts`

```
import { getFirestore } from "firebase/firestore"
export const db = getFirestore(app)
```

Not yet implemented: - Firestore CRUD operations - Real-time listeners - Data migration from localStorage

---

Authentication System

Firebase Authentication

Configuration File: `src/firebase.ts`

```
const firebaseConfig = {
  apiKey: "AIzaSyDWdcEBNoobKZoNRQmWlooCJNCBfb_M8-U",
  authDomain: "lapmap-cc281.firebaseapp.com",
  projectId: "lapmap-cc281",
  storageBucket: "lapmap-cc281.firebasestorage.app",
  messagingSenderId: "102969245328",
  appId: "1:102969245328:web:1b3478a17782fc6f792215"
}
```

Authentication Flow

1. **App Load**

```
    useEffect(() => {
      const unsubscribe = onAuthStateChanged(auth, (currentUser) => {
        setUser(currentUser)
        setAuthLoading(false)
      })
      return () => unsubscribe()
    }, [])
```

2. **Not Authenticated** → Show `<Login />` component
3. **Sign Up**

```
await createUserWithEmailAndPassword(auth, email, password)
```

4. **Sign In**

```
await signInWithEmailAndPassword(auth, email, password)
```

5. **Authenticated** → Show main app with user.email in header
6. **Sign Out**

```
await signOut(auth)
```

Security

- Passwords hashed by Firebase
- No credentials stored in code
- Auth state managed by Firebase SDK
- Authorized domains configured in Firebase Console

Map Integration

Leaflet Setup

Imports:

```
import { MapContainer, TileLayer, Marker, Popup, Polyline, CircleMarker } from 'react-
        leaflet'
import MarkerClusterGroup from 'react-leaflet-cluster'
import L from 'leaflet'
```

Icon Path Fix for Vite:

```
import marker2x from 'leaflet/dist/images/marker-icon-2x.png'
import marker1x from 'leaflet/dist/images/marker-icon.png'
import markerShadow from 'leaflet/dist/images/marker-shadow.png'

L.Icon.Default.mergeOptions({
  iconRetinaUrl: marker2x,
  iconUrl: marker1x,
  shadowUrl: markerShadow,
})
```

Marker Clustering

Configuration:

```
<MarkerClusterGroup chunkedLoading>
  {places.map(p => <Marker key={p.id} position={[p.lat, p.lon]} />)}
</MarkerClusterGroup>
```

Important: MarkerClusterGroup does not support dynamic icon changes on existing markers. This is why visited status is shown in popup content rather than changing marker icons.

Route Display

```
{route && (
  <>
    <Polyline positions={route.coordinates} color="#2563eb" weight={4} />
    {route.waypoints.map((wp, i) => (
      <CircleMarker key={i} center={wp} radius={6} fillColor="#2563eb" />
    ))}
  </>
)}
```

API Integrations

Overpass API (OpenStreetMap Data)

Purpose: Query POIs, campsites, fuel stations, etc.

Endpoints: - `https://overpass-api.de/api/interpreter` - `https://lz4.overpass-api.de/api/interpreter` - `https://overpass.kumi.systems/api/interpreter`

Strategy: Try multiple endpoints in sequence for redundancy

Example Query:

```
[out:json][timeout:25];
(
  node["place"="city"](south,west,north,east);
  node["place"="town"](south,west,north,east);
);
out body;
```

Rate Limiting: - 1000ms debounce on map movement - Graceful fallback to alternative endpoints - Error handling for 429 (Too Many Requests)

OSRM (Routing)

Endpoint: `https://router.project-osrm.org/trip/v1/driving/{coords}`

Parameters: - `source=first` - Start from first coordinate - `roundtrip=false` - One-way trip - `overview=full` - Return full geometry - `geometries=geojson` - GeoJSON format

Input Format:

```
lon1,lat1;lon2,lat2;lon3,lat3
```

Response:

```
{
  "trips": [{
    "geometry": {
      "coordinates": [[lon, lat], ...]
    },
    "legs": [
      {"distance": 123456, "duration": 7890}
    ]
  }]
}
```

Nominatim (Geocoding)

Endpoint: `https://nominatim.openstreetmap.org/search`

Parameters: - `q` - Query string - `format=json` - `countrycodes=au` - Australia only - `limit=5` - Max results

Usage Policy: Rate limited, user-agent required

MeteoBlue (Weather)

URL Format:

```
https://www.meteoblue.com/en/weather/forecast/week/{lat}N{lon}E
```

Integration: External link, no API calls required

Google Maps

URL Format:

```
https://www.google.com/maps/search/?api=1&query={lat},{lon}
```

Integration: External link for viewing locations

Deployment

GitHub Pages

Configuration:

```
// vite.config.ts
export default defineConfig({
  base: '/interactive-Aus-map/',
  plugins: [react()],
})
```

Workflow: `.github/workflows/deploy.yml` - Triggers on push to `main` branch - Builds with `npm run build` - Deploys to `gh-pages` branch - Live in 2-3 minutes

Build Command: `npm run build` - TypeScript compilation - Vite production build - Output: `dist/` folder

Domain Configuration

Live URL: https://petehep.github.io/interactive-Aus-map/

Firebase Authorized Domains: - `localhost` (development) - `petehep.github.io` (production)

Development Workflow

Setup

```
# Clone repository
git clone https://github.com/petehep/interactive-Aus-map.git
cd interactive-Aus-map

# Install dependencies
npm install

# Start dev server
npm run dev
```

Development Server

Command: `npm run dev`
URL: http://localhost:5173/interactive-Aus-map/
Hot Module Replacement: Enabled
Port: 5173

Building for Production

```
npm run build
npm run preview  # Test production build locally
```

Git Workflow

```
# Make changes
git add .
git commit -m "Description of changes"
git push origin main

# Automatic deployment to GitHub Pages
```

Code Style

- **Language:** TypeScript strict mode
- **React:** Functional components with hooks
- **State Management:** useState, useCallback, useMemo, useEffect
- **Styling:** Inline styles and CSS classes
- **Component Structure:** Props interface, component function, export

---

Performance Optimizations

Map Performance

- **Marker Clustering** - Prevents rendering thousands of markers
- **Progressive Loading** - Loads appropriate detail for zoom level
- **Debouncing** - 1000ms delay on map movement

React Performance

- **useCallback** - Memoize functions to prevent re-renders
- **useMemo** - Memoize expensive computations (state grouping)
- **Key Props** - Proper keys on lists for efficient reconciliation

Network Optimization

- **Multiple API Endpoints** - Fallback for reliability
- **Conditional Fetching** - Only fetch when needed
- **Abort Controllers** - Cancel in-flight requests on new queries

---

Known Limitations

1. **Marker Icon Changes** - Cannot dynamically change marker icons in clusters
2. **OSM Data Coverage** - Remote areas have limited data
3. **Rate Limiting** - Overpass API has usage limits
4. **localStorage Size** - ~5-10MB limit per domain
5. **No Offline Mode** - Requires internet for tiles and data

6. **Single Device** - Data not synced across devices (yet)

Future Enhancements

Phase 1: Firestore Integration

- Migrate localStorage to Firestore
- User-specific data storage
- Cross-device synchronization
- Real-time updates

Phase 2: Advanced Features

- Offline map caching
- Route waypoint reordering (drag & drop)
- Distance/fuel cost calculations
- Custom map layers
- Export itinerary to PDF/GPX

Phase 3: Social Features

- Share itineraries with friends
- Public/private trip visibility
- Comments and ratings on places
- Photo uploads

Troubleshooting

Marker Icons Not Loading

Symptom: Placeholder boxes with “Mar…”
Cause: Vite asset path issues
Solution: Verify imports in MapView.tsx lines 8-10

Firebase Auth Not Working on GitHub Pages

Symptom: Login fails in production
Cause: Domain not authorized
Solution: Add `petehep.github.io` to Firebase Console → Authentication → Settings → Authorized domains

Overpass API 429 Errors

Symptom: Places not loading
Cause: Rate limiting
Solution: Increase debounce delay or reduce query frequency

TypeScript Errors

Symptom: Build fails
Cause: Type mismatches
Solution: Run `npm run build` to see errors, fix type definitions

Contact & Support

Repository: https://github.com/petehep/interactive-Aus-map
Issues: Report bugs via GitHub Issues
Documentation: This file and USER_MANUAL.md

Last Updated: November 24, 2025