Backend Integration Guide – Melbourne Car Park Solution

1. Overview

The **frontend** (my part) is already done. It:

- Displays a map of Melbourne
- Lets the user search for a destination
- Shows **available parking spots**
- Displays **environment-friendly suggestions**
- Shows **insight charts** (average occupancy, busiest hours)
- Currently uses **mock data** — we need your backend to replace these with **real data**.

Your job:

- Build APIs the frontend can call.
- Make sure the API returns **JSON** in the format described below.
- Enable **CORS** so the frontend can access the API from a different port.

---

2. Required API Endpoints

1) Search for parking spots

Endpoint:

bash
CopyEdit
```
GET /api/v1/parking?dest=Flinders%20Street
```

Response JSON:

json
CopyEdit

```
[
  {
    "id": "PARK001",
    "name": "Flinders St Car Park",
    "lat": -37.8183,
    "lng": 144.9671,
    "capacity": 200,
    "available": 35
  },
  {
    "id": "PARK002",
    "name": "Fed Square Parking",
    "lat": -37.8179,
    "lng": 144.9691,
    "capacity": 150,
    "available": 50
  }
]
```

---

2) Environment-friendly travel suggestions

Endpoint:

bash
CopyEdit
```
GET /api/v1/environment?dest=Flinders%20Street
```

Response JSON:

json
CopyEdit
```
{
  "publicTransport": "Take Tram 70 from Swanston St, 5 min walk to destination",
  "co2SavedKg": 3.5
}
```

---

3) Parking insights (for charts)

Endpoint:

bash
CopyEdit
```
GET /api/v1/stats/parking
```

Response JSON:

json
CopyEdit
```
{
  "averageOccupancy": [
    { "carPark": "Flinders St", "percentage": 60 },
    { "carPark": "Fed Square", "percentage": 45 }
  ],
  "busiestHours": [
    { "hour": "08:00", "count": 50 },
    { "hour": "09:00", "count": 80 },
    { "hour": "10:00", "count": 120 }
  ]
}
```

---

3. CORS Setup (VERY IMPORTANT for local development)

If backend is running on `localhost:4000` and frontend is on `localhost:5500`:

- Enable CORS in Express:

javascript
CopyEdit
```
const cors = require('cors');
app.use(cors());
```

This allows the frontend to call your API without being blocked by the browser.

---

4. Folder & File Structure (Recommended)

markdown
CopyEdit
```
backend/
  server.js
  routes/
```

```
    parking.js
    environment.js
    stats.js
```

---

5. Example Minimal Backend (Node + Express)

javascript
CopyEdit

```
const express = require('express');
const cors = require('cors');
const app = express();
const PORT = 4000;

app.use(cors());

// Search parking
app.get('/api/v1/parking', (req, res) => {
  res.json([
    { id: 'PARK001', name: 'Flinders St Car Park', lat: -37.8183,
lng: 144.9671, capacity: 200, available: 35 },
    { id: 'PARK002', name: 'Fed Square Parking', lat: -37.8179, lng:
144.9691, capacity: 150, available: 50 }
  ]);
});

// Environment suggestions
app.get('/api/v1/environment', (req, res) => {
  res.json({
    publicTransport: 'Take Tram 70 from Swanston St, 5 min walk to
destination',
    co2SavedKg: 3.5
  });
});

// Stats
app.get('/api/v1/stats/parking', (req, res) => {
  res.json({
    averageOccupancy: [
      { carPark: 'Flinders St', percentage: 60 },
      { carPark: 'Fed Square', percentage: 45 }
    ],
    busiestHours: [
      { hour: '08:00', count: 50 },
      { hour: '09:00', count: 80 },
      { hour: '10:00', count: 120 }
    ]
  });
});

app.listen(PORT, () => console.log(`Backend running on
http://localhost:${PORT}`));
```

---

6. How Backend & Frontend Work Together

1. **Frontend**: User searches for a destination.
2. **Frontend**: Sends `GET /api/v1/parking?dest=...` to your backend.
3. **Backend**: Looks up parking spots in DB or live API and returns JSON.
4. **Frontend**: Displays spots on map.
5. **Frontend**: Calls `/api/v1/environment` for eco-friendly options.
6. **Frontend**: Calls `/api/v1/stats/parking` for chart data.
7. **Frontend**: Updates charts and insights.