**🗺️ Journey 1: Route Planning (Mostly Complete)**

**Flow:** greet → menu\_selection → route\_info → station\_selection → action\_selection**What's Implemented:**

* ✅ User greets and selects route planning
* ✅ Bot asks for start/destination locations
* ✅ User provides route info (e.g., "from Richmond to Dandenong")
* ✅ Bot finds charging stations along the route
* ✅ User selects a station
* ✅ Bot offers actions (directions, compare, check availability)

**🚨 Journey 2: Emergency Charging (Mostly Complete)**

**Flow:** greet → emergency\_charging → emergency\_location\_input → station\_selection → action\_selection**What's Implemented:**

* ✅ User indicates emergency/low battery
* ✅ Bot asks for current location and battery level
* ✅ User provides location + battery (e.g., "Frankston, 6% battery")
* ✅ Bot finds nearest emergency charging stations
* ✅ User selects a station
* ✅ Bot offers actions (directions, compare, check availability)

**⚡ Journey 3: Charging Preferences (Partially Complete)**

**Flow:** greet → preference\_charging → station\_selection → action\_selection**What's Implemented:**

* ✅ User selects preference-based charging
* ✅ Bot asks for preference type (cheapest, fastest, closest, premium)
* ❌ **Missing:** Location input step
* ❌ **Missing:** Preference-based station filtering
* ✅ User selects a station
* ✅ Bot offers actions (directions, compare, check availability)

**Current Status:** ~60% complete - missing the preference filtering logic

\*\*🎯 Choose ONE station for emergency charging:\*\*

• \*\*'1'\*\* - Get directions to first station

• \*\*'2'\*\* - Get directions to second station

• \*\*'3'\*\* - Get directions to third station

Or say 'get directions to station 1'

Your input -> 2

📊 \*\*Comparing charging options...\*\*

🔍 \*\*Analysis includes:\*\*

• Power output (kW)

• Cost per kWh

• Distance from your route

• Available charging points

• User ratings & reviews

This will show a detailed comparison of all nearby stations along your route.

What would you like to do next?

• Select a different station

• Get directions to any station

• Plan a new route

• Go back to main menu

**🎯 Summary:**

* **Station data**: 90% real (from CSV), 10% hardcoded fallbacks
* **Location coordinates**: 50% real (from CSV), 50% hardcoded (user location)
* **Distance calculations**: 100% real (calculated from CSV coordinates)
* **Fallback responses**: 100% hardcoded (when data service fails)

The system is working correctly - it's using real data when it can find it, and only falling back to hardcoded data when there are errors or missing data.

**To do next:**

1. **Implement Real-Time Location Detection (find out more) (alex)**
2. **Add Real-Time Station Availability (doable)**

**Chargefox (station availability) (harshit)**

1. **Implement Context Memory (slots – Doing this)**

* **Entity recognition** for station locations, battery levels, preferences
* **Fuzzy matching** for location names

1. **Implement Progressive Disclosure**

**User: "I need charging"**

**Bot: "What's your current location?"**

**User: "Richmond"**

**Bot: "What's your battery level?"**

**User: "15%"**

**Bot: "I found 3 stations. Would you like to see the closest, cheapest, or fastest?"**

**Set up API keys for:**

**TomTom (traffic & routing)**

1. **Replace Hardcoded Location (coordinates\_config.py)**
2. *# Instead of hardcoded coordinates*
3. def get\_user\_location():
4. *# Try GPS first for website deployment*
5. *# Fallback to IP geolocation*
6. *# Cache for session*
7. *pass*

**2. Add Real-Time Station Status (data\_service.py)**

def get\_real\_time\_availability(*self*, *station\_id*):

*# Call Chargefox API*

*# Return current status*

*# Handle API failures gracefully*

*pass*

**3. Implement Progressive Conversation (actions.py)**

def \_ask\_follow\_up\_question(*self*, *context*):

*# Based on conversation state*

*# Ask one question at a time*

*# Build context progressively*

*pass*

**To do next:**

**Implement Real-Time Location Detection (find out more) (alex)**

**Add Real-Time Station Availability (doable)**

**Chargefox (station availability) (harshit)**

**Implement Context Memory (slots – Doing this)**

**Entity recognition for station locations, battery levels, preferences**

**Fuzzy matching for location names**