

OBDscribe – Evidence Pack

Context:

- App: OBDscribe (service writer copilot demo)
- Screen: /app/new-report
- Goal: Replace free-text vehicle fields with structured dropdowns driven by a JSON dataset.

What was implemented:

1) JSON-backed make/model dataset

- File: src/data/vehicle-makes-models.json
- Shape: { "brands": { "<Make>": ["Model1", "Model2", ...], ... } }

2) Vehicle options helper module

- File: src/data/vehicleOptions.ts
- Responsibilities:
 - Import JSON dataset and expose:
 - YEARS: number[] (1940–2026, descending)
 - MAKES: string[] (sorted list of makes)
 - VEHICLE_MAKE_MODELS: Record<string, string[]> (make → models)

3) New Report form wiring

- File: src/app/app/new-report/page.tsx
- Changes:
 - Import YEARS, MAKES, VEHICLE_MAKE_MODELS from vehicleOptions.
 - Add availableModels = useMemo(() => VEHICLE_MAKE_MODELS[form.make] ?? [], [form.make]).
 - Replace free-text year, make, model with native <select> elements:
 - Year: dropdown with YEARS; stores numeric year in form.year.
 - Make: dropdown with MAKES; when make changes, model is reset to "".
 - Model: dropdown bound to availableModels, disabled until a make is selected.

Behavior (manual test):

- 1) Navigate to /app/new-report while dev server is running.
- 2) Year field:
 - Click Year dropdown; you should see a list from 2026 down to 1940.
 - Selecting a year updates form.year and persists through submit.
- 3) Make field:
 - Click Make dropdown; you should see makes sourced from vehicle-makes-models.json.
 - Example: Toyota, Honda, BMW, etc.
 - Selecting Toyota updates form.make to 'Toyota' and clears form.model.
- 4) Model field:
 - Before selecting a make: the field is disabled and shows 'Select a make first'.
 - After selecting Toyota: the field becomes active and lists Toyota-specific models from VEHICLE_MAKE_MODELS['Toyota'] (e.g., Corolla, Camry, RAV4).
 - Selecting a model updates form.model.
- 5) Generate Report flow:
 - Fill Year, Make, Model, Mileage, OBD codes, Complaint, and optional notes.
 - Click 'Generate Report'.

- The request body includes structured vehicle info (year, make, model, etc.) plus complaint and codes.
- On success, Tech View, Customer View, and Maintenance Suggestions are populated from the AI backend.

Why this is valuable:

- Reduces typos and inconsistent vehicle naming in reports.
- Makes the demo feel like real shop software (Carfax-style UX).
- Sets up a clean path to later enrich behavior (VIN decoding, trim, engine type, etc.).

Files touched in this evidence:

- src/data/vehicle-makes-models.json
- src/data/vehicleOptions.ts
- src/app/app/new-report/page.tsx

Status:

- Manual testing confirms the dropdowns work end-to-end and the AI report still generates successfully with the structured vehicle payload.