MVP Definíció & Gyakorlati Megvalósítás

MVP Scope - Mit építünk első körben?

```
// MVP Feature Set (8-10 hetes fejlesztés)

MVP_FEATURES = {
  "essential": [
    "OBD-II hibakód bevitel (manuális)",
    "Autó típus kiválasztása (make/model/year)",
    "AI-alapú diagnózis generálás",
    "Eredmény megjelenítés (strukturált formátum)",
    "Session history (utolsó 20 diagnózis)"
  ],

  "not_in_mvp": [
    "Tünet-alapú diagnózis (Phase 2)",
    "Barcode scanner (Phase 2)",
    "Exportálás PDF-be (Phase 2)",
    "Többfelhasználós rendszer (Phase 2)",
    "Offline mode (Phase 3)",
    "RAG knowledge base (Phase 2-3)",
    "Mechanic feedback loop (Phase 2)"
  ]
}
```

Pontos Lépések - Hogyan Csináld Meg?

Héét 1-2: Setup & Adatbázis Feltöltés

Backend Setup

```
# 1. Projekt inicializálás
mkdir auto-diagnostic-mvp
cd auto-diagnostic-mvp

# Backend
mkdir backend && cd backend
npm init -y
npm install express typescript @types/node @types/express
npm install pg redis openai zod helmet cors express-rate-limit
npm install -D tsx nodemon
```

```
# tsconfig.json setup
npx tsc --init
```

Backend Struktúra:

```
backend/
├── src/
│   ├── server.ts                # Express app entry
│   ├── config/
│   │   └── database.ts          # PostgreSQL connection
│   ├── routes/
│   │   ├── diagnostic.routes.ts  # POST /diagnose
│   │   └── vehicle.routes.ts     # GET /vehicles/search
│   ├── services/
│   │   ├── llm.service.ts        # OpenAI integration
│   │   └── diagnostic.service.ts # Core logic
│   ├── models/
│   │   └── types.ts              # TypeScript interfaces
│   └── utils/
│       └── validators.ts        # Zod schemas
├── prisma/
│   └── schema.prisma            # Database schema
└── package.json
```

PostgreSQL Setup (Railway)

```
# Railway CLI telepítés
npm install -g @railway/cli

# Login & projekt létrehozás
railway login
railway init

# PostgreSQL hozzáadás
railway add postgresql

# Connection string elérése
railway variables
# Kimásolja a DATABASE_URL-t
```

Prisma Schema (backend/prisma/schema.prisma):

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model Vehicle {
  id          String   @id @default(uuid())
  make        String
  model       String
  yearFrom    Int
  yearTo      Int?
  engineTypes Json?    // [{"code": "1.6 TDI", "hp": 105}]
  createdAt   DateTime @default(now())

  @@index([make, model])
  @@index([yearFrom, yearTo])
}

model OBDCode {
  id               String   @id @default(uuid())
  code             String   @unique  // P0300
  category         String?  // Powertrain
  descriptionShort String
  descriptionLong  String?
  commonCauses     Json?    // ["cause1", "cause2"]
  criticality      String?  // Critical, High, Medium, Low
  generic          Boolean  @default(true)
  createdAt        DateTime @default(now())

  @@index([code])
  @@index([category])
}

model DiagnosticSession {
  id                 String   @id @default(uuid())
  sessionType        String   // 'obd_code'
  obdCodes           String[]
  vehicleManualEntry Json     // {make, model, year}
  llmProvider        String
  llmModel           String
  diagnosis          String
  recommendedActions Json?
  confidenceScore    Float?
  createdAt          DateTime @default(now())
  processingTimeMs   Int?

  @@index([createdAt(sort: Desc)])
}
```

```
# Schema alkalmazása
npx prisma migrate dev --name init
npx prisma generate
```

Adatok Forrásai & Feltöltés

1️⃣ OBD-II Hibakód Database

Legjobb Ingyenes Források:

A. Klavkarr Database (11,000+ kód, INGYENES)

```
# Letöltés
wget https://www.klavkarr.com/data-trouble-code-obd2.php

# Vagy használd a publikus listákat:
# - Generic codes: P0xxx, P2xxx, P34xx (SAE standardizált)
# - Manufacturer codes: P1xxx, P30-33xx (gyártó-specifikus)
```

Példa adatstruktúra (CSV formátum):

```
code,category,description_short,description_long,common_causes,criticality
P0300,Powertrain,Random/Multiple Cylinder Misfire Detected,"Multiple cylinders misfiring
P0171,Powertrain,System Too Lean (Bank 1),"Fuel mixture too lean on bank 1","Vacuum leak|
P0420,Powertrain,Catalyst System Efficiency Below Threshold,"Catalytic converter not work
P0455,Powertrain,EVAP System Leak Detected (Large Leak),"Large leak in evaporative emissi
```

Python Script - OBD Adatok Feldolgozása:

```
# backend/scripts/populate_obd_codes.py
import csv
import json
from prisma import Prisma

async def populate_obd_codes():
    db = Prisma()
    await db.connect()

    # Alapvető OBD-II kódok (manuális adatforrás)
    obd_codes = [
        {
            "code": "P0300",
            "category": "Powertrain",
            "descriptionShort": "Random/Multiple Cylinder Misfire Detected",
            "descriptionLong": "Multiple cylinders are misfiring randomly, which can be c
            "commonCauses": json.dumps([
                "Faulty spark plugs or ignition coils",
                "Clogged fuel injectors",
                "Vacuum leak in intake manifold",
                "Low fuel pressure",
                "Carbon buildup in cylinders",
                "Worn piston rings or valve seals"
            ]),
            "criticality": "High",
            "generic": True
        },
        {
            "code": "P0171",
            "category": "Powertrain",
            "descriptionShort": "System Too Lean (Bank 1)",
            "descriptionLong": "The fuel mixture is too lean on bank 1, meaning there's t
            "commonCauses": json.dumps([
                "Vacuum leak in intake system",
                "MAF (Mass Air Flow) sensor malfunction",
                "Weak fuel pump or clogged fuel filter",
                "Faulty oxygen sensor",
                "Exhaust leak before oxygen sensor",
                "Dirty or faulty fuel injectors"
            ]),
            "criticality": "Medium",
            "generic": True
        },
        {
            "code": "P0420",
            "category": "Powertrain",
            "descriptionShort": "Catalyst System Efficiency Below Threshold (Bank 1)",
            "descriptionLong": "The catalytic converter is not functioning efficiently en
            "commonCauses": json.dumps([
                "Worn or damaged catalytic converter",
                "Faulty downstream oxygen sensor",
                "Exhaust leak before catalytic converter",
                "Engine running too rich or lean",
                "Oil or coolant contamination in exhaust"
            ]),
            "criticality": "Medium",
            "generic": True
        },
        {
            "code": "P0455",
            "category": "Powertrain",
            "descriptionShort": "EVAP System Leak Detected (Large Leak)",
            "descriptionLong": "A large leak has been detected in the evaporative emissio
            "commonCauses": json.dumps([
                "Loose or damaged fuel cap",
                "Cracked or disconnected EVAP hoses",
                "Faulty purge valve",
                "Leaking fuel tank or filler neck",
                "Damaged charcoal canister"
            ]),
            "criticality": "Low",
            "generic": True
        },
        # Add top 50-100 most common codes manually
    ]

    for code in obd_codes:
        await db.obdcode.create(data=code)

    print(f"Inserted {len(obd_codes)} OBD codes")
    await db.disconnect()

# Futtatás
import asyncio
asyncio.run(populate_obd_codes())
```

MVP Stratégia OBD Kódokhoz:

- **Kezdéshez:** Top 100 leggyakoribb kód manuálisan (P0300, P0171, P0420, etc.)
- **Adatforrások:**
    - SAE J2012 standard (publikus specifikáció)
    - NHTSA TSB database (ingyenes, lásd alább)
    - Klavkarr publikus adatok
    - Webscraping: obd-codes.com, engine-codes.com

B. NHTSA TSB & Recall Database (INGYENES, HIVATALOS)

API Endpoint:

```
# NHTSA Vehicle Safety & Defects Database
BASE_URL="https://api.nhtsa.gov/complaints"

# TSB lekérés
curl "https://api.nhtsa.gov/vehicles/bymanufacturer/vw?format=json"

# Hibakód keresés
curl "https://api.nhtsa.gov/safetyRatings/modelyear/2015/make/volkswagen/model/golf"
```

Python Script - NHTSA TSB Scraping:

```
# backend/scripts/fetch_nhtsa_data.py
import requests
import json

def fetch_tsbs(make, model, year):
    url = f"https://api.nhtsa.gov/products/vehicle/makes/{make}/models/{model}/years/{yea
    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()
        # Process and save to database
        return data
    else:
        print(f"Error fetching data: {response.status_code}")
        return None

# Example usage
tsbs = fetch_tsbs("VOLKSWAGEN", "GOLF", 2015)
```

2️⃣ Járműadatok Database

Források:

A. VehicleDatabases.com API (Paid, de van free tier)

```
# Free tier: 100 requests/day
curl -X GET "https://www.vehicledatabases.com/api/v1/vehicle/lookup?vin=WVWZZZ1KZBW123456
  -H "Authorization: Bearer YOUR_API_KEY"
```

B. CarQuery API (INGYENES)

```
# Publikus API, no key szükséges
curl "https://www.carqueryapi.com/api/0.3/?cmd=getTrims&make=volkswagen&model=golf&year=2
```

Response példa:

```
{
  "Trims": [
    {
      "model_id": "12345",
      "model_make_id": "volkswagen",
      "model_name": "Golf VII",
      "model_trim": "1.6 TDI",
      "model_year": "2015",
      "model_body": "Hatchback",
      "model_engine_position": "Front",
      "model_engine_cc": "1598",
      "model_engine_type": "Inline 4",
      "model_engine_fuel": "Diesel",
      "model_engine_power_ps": "105",
      "model_engine_power_rpm": "4400",
      "model_engine_torque_nm": "250",
      "model_engine_torque_rpm": "1500"
    }
  ]
}
```

C. Manual Entry + Wikipedia (MVP Megoldás)

Egyszerűbb MVP megközelítés:

```
# backend/scripts/populate_vehicles.py
from prisma import Prisma
import json

async def populate_common_vehicles():
    db = Prisma()
    await db.connect()
```

```python
# Top 50 leggyakoribb autó Európában/Magyarországon
vehicles = [
    {
        "make": "Volkswagen",
        "model": "Golf",
        "yearFrom": 2012,
        "yearTo": 2019,
        "engineTypes": json.dumps([
            {"code": "1.2 TSI", "hp": 85, "fuel": "petrol"},
            {"code": "1.4 TSI", "hp": 125, "fuel": "petrol"},
            {"code": "1.6 TDI", "hp": 105, "fuel": "diesel"},
            {"code": "2.0 TDI", "hp": 150, "fuel": "diesel"}
        ])
    },
    {
        "make": "Volkswagen",
        "model": "Passat",
        "yearFrom": 2010,
        "yearTo": 2019,
        "engineTypes": json.dumps([
            {"code": "1.6 TDI", "hp": 105, "fuel": "diesel"},
            {"code": "2.0 TDI", "hp": 140, "fuel": "diesel"},
            {"code": "2.0 TDI", "hp": 170, "fuel": "diesel"}
        ])
    },
    {
        "make": "Ford",
        "model": "Focus",
        "yearFrom": 2011,
        "yearTo": 2018,
        "engineTypes": json.dumps([
            {"code": "1.0 EcoBoost", "hp": 125, "fuel": "petrol"},
            {"code": "1.5 TDCi", "hp": 95, "fuel": "diesel"},
            {"code": "1.6 TDCi", "hp": 115, "fuel": "diesel"}
        ])
    },
    {
        "make": "Opel",
        "model": "Astra",
        "yearFrom": 2009,
        "yearTo": 2019,
        "engineTypes": json.dumps([
            {"code": "1.4 Turbo", "hp": 140, "fuel": "petrol"},
            {"code": "1.6 CDTI", "hp": 110, "fuel": "diesel"},
            {"code": "1.7 CDTI", "hp": 125, "fuel": "diesel"}
        ])
    },
    # ... top 50 autó
]

for vehicle in vehicles:
    await db.vehicle.create(data=vehicle)

print(f"Inserted {len(vehicles)} vehicles")
await db.disconnect()
```

```
import asyncio
asyncio.run(populate_common_vehicles())
```

MVP Járműadatok Stratégia:

1. **Fókusz:** Top 50 leggyakoribb autó Magyarországon (VW, Skoda, Ford, Opel, BMW, Audi)
2. **Adatgyűjtés:**
   - Wikipedia infoboxes (automatizálható)
   - Gyártói weboldalak specifikációi
   - AutoData.net (ha van budget: €50-100/hó)
3. **Bővítés:** Mechanikai feedback alapján, ahogy új autókkal találkoznak

Héét 3-4: Core Backend Implementation

LLM Service (OpenAI Integration):

```
// backend/src/services/llm.service.ts
import OpenAI from 'openai';
import { PrismaClient } from '@prisma/client';

const openai = new OpenAI({
  apiKey: process.env.OPENAI_API_KEY,
});

const prisma = new PrismaClient();

export interface DiagnosticRequest {
  obdCodes: string[];
  vehicle: {
    make: string;
    model: string;
    year: number;
    engine?: string;
    mileage?: number;
  };
  additionalContext?: string;
}

export async function generateDiagnosis(request: DiagnosticRequest) {
  // 1. Fetch code definitions from database
  const codeData = await prisma.oBDCode.findMany({
    where: {
      code: { in: request.obdCodes }
    }
  });

  // 2. Build prompt
  const prompt = buildDiagnosticPrompt(request, codeData);

  // 3. Call OpenAI
  const startTime = Date.now();
  const response = await openai.chat.completions.create({
    model: 'gpt-4o',
    messages: [
      { role: 'system', content: DIAGNOSTIC_SYSTEM_PROMPT },
      { role: 'user', content: prompt }
    ],
    response_format: { type: 'json_object' },
    temperature: 0.3
  });

  const processingTime = Date.now() - startTime;
  const diagnosis = JSON.parse(response.choices[^0].message.content || '{}');

  // 4. Save session
  await prisma.diagnosticSession.create({
    data: {
      sessionType: 'obd_code',
      obdCodes: request.obdCodes,
      vehicleManualEntry: request.vehicle,
      llmProvider: 'openai',
      llmModel: response.model,
      diagnosis: JSON.stringify(diagnosis),
      recommendedActions: diagnosis.recommended_actions,
      confidenceScore: diagnosis.confidence,
      processingTimeMs: processingTime
    }
  });

  return {
    diagnosis,
    metadata: {
      model: response.model,
      processingTimeMs: processingTime,
      tokensUsed: response.usage?.total_tokens
    }
  };
}

function buildDiagnosticPrompt(request: DiagnosticRequest, codeData: any[]) {
  let prompt = `JÁRMŰINFORMÁCIÓ:
- Gyártmány: ${request.vehicle.make}
- Modell: ${request.vehicle.model}
- Évjárat: ${request.vehicle.year}
${request.vehicle.engine ? `- Motor: ${request.vehicle.engine}` : ''}
${request.vehicle.mileage ? `- Futásteljesítmény: ${request.vehicle.mileage} km` : ''}

HIBAKÓDOK:
`;

  codeData.forEach(code => {
    const causes = JSON.parse(code.commonCauses || '[]');
    prompt += `
Kód: ${code.code}
Kategória: ${code.category}
Leírás: ${code.descriptionLong}
Gyakori okok: ${causes.join(', ')}
Kritikusság: ${code.criticality}
`;
  });

  if (request.additionalContext) {
    prompt += `\nTOVÁBBI MEGJEGYZÉSEK:\n${request.additionalContext}\n`;
  }

  prompt += `
KÉRT KIMENET (JSON formátum):
{
  "primary_issue": "Főprobléma rövid összefoglalója magyarul",
  "confidence": 0.0-1.0,
  "codes_analyzed": [
    {
      "code": "P0XXX",
      "severity": "critical|high|medium|low",
      "potential_causes": [
        {
          "cause": "Konkrét alkatrész vagy rendszer magyarul",
          "probability": "high|medium|low",
          "reasoning": "Miért valószínű ez a kód és jármű kontextusában"
        }
      ]
    }
  ],
  "recommended_actions": [
    {
      "step": 1,
      "action": "Konkrét diagnosztikai vagy javítási lépés magyarul",
      "estimated_time": "X perc",
      "tools_needed": ["Eszköz 1", "Eszköz 2"],
      "priority": "high|medium|low"
    }
  ],
  "estimated_total_repair": {
    "time_hours": "X-Y",
    "difficulty": "simple|moderate|complex",
    "cost_range_huf": "X-Y"
  },
  "warnings": ["Kritikus biztonsági vagy jármű károsodási kockázatok magyarul"]
}

Fókuszálj gyakorlati, végrehajtható tanácsokra. Priorizáld a gyakori problémákat és logik

  return prompt;
}

const DIAGNOSTIC_SYSTEM_PROMPT = `Te egy szakértő autószerelői diagnosztikai asszisztens

1. OBD-II hibakódok elemzése a konkrét jármű gyártmány, modell és évjárat kontextusában
2. Gyakori meghibásodási minták és gyártó-specifikus problémák figyelembevétele
3. Logikus, lépésről-lépésre diagnosztikai eljárások ajánlása
4. A leghatékonyabb útvonal ajánlása a kiváltó ok azonosításához
5. Javítási idő, nehézség és költségek reális becslése
```

```
6. Figyelmeztetés potenciális biztonsági problémákról vagy kárláncról

Kimeneted legyen:
- Praktikus és végrehajtható standard műhelyi felszereléssel
- Valószínűség és költséghatékonyság szerint priorizálva
- Világosan strukturálva, gyors referenciaként tableten használható
- Őszinte a bizonyossági szintekről és diagnosztikai bizonytalanságról

Mindig vedd figyelembe:
- Jármű életkor és futásteljesítmény (kopási minták)
- Gyártó-specifikus meghibásodási módok
- Logikus diagnosztikai sorrend (legolcsóbb/legegyszerűbb tesztek először)
- Több kód kölcsönhatásai és kiváltó ok kapcsolatok

Minden választ MAGYARUL adj!`;
```

API Route:

```
// backend/src/routes/diagnostic.routes.ts
import express from 'express';
import { z } from 'zod';
import { generateDiagnosis } from '../services/llm.service';

const router = express.Router();

const DiagnosticRequestSchema = z.object({
  obd_codes: z.array(z.string().regex(/^[PCBU][0-9]{4}$/)).min(1).max(10),
  vehicle: z.object({
    make: z.string().min(2),
    model: z.string().min(1),
    year: z.number().int().min(1996).max(new Date().getFullYear() + 1),
    engine: z.string().optional(),
    mileage: z.number().int().min(0).optional()
  }),
  additional_context: z.string().max(1000).optional()
});

router.post('/diagnose', async (req, res) => {
  try {
    // Validate input
    const validated = DiagnosticRequestSchema.parse(req.body);

    // Generate diagnosis
    const result = await generateDiagnosis({
      obdCodes: validated.obd_codes,
      vehicle: validated.vehicle,
      additionalContext: validated.additional_context
    });

    res.json({
      success: true,
      data: result
    });

  } catch (error) {
    if (error instanceof z.ZodError) {
      res.status(400).json({
        success: false,
        error: 'Validation error',
        details: error.errors
      });
    } else {
      console.error('Diagnosis error:', error);
      res.status(500).json({
        success: false,
        error: 'Internal server error'
      });
    }
  }
});

export default router;
```

Hét 5-6: Frontend (React + TypeScript)

```
# Frontend setup
npm create vite@latest frontend -- --template react-ts
cd frontend
npm install
npm install @tanstack/react-query axios zustand react-hook-form zod
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

Tablet-optimalizált Component:

```
// frontend/src/components/OBDDiagnosis.tsx
import React, { useState } from 'react';
import { useForm } from 'react-hook-form';
import { useMutation } from '@tanstack/react-query';
import axios from 'axios';

interface DiagnosisForm {
  codes: string[];
  make: string;
  model: string;
  year: number;
  engine?: string;
  context?: string;
}

export function OBDDiagnosis() {
  const [codes, setCodes] = useState<string[]>(['']);
  const { register, handleSubmit, formState: { errors } } = useForm<DiagnosisForm>();

  const diagnosisMutation = useMutation({
    mutationFn: async (data: DiagnosisForm) => {
      const response = await axios.post('/api/v1/diagnose', {
        obd_codes: data.codes.filter(c => c),
        vehicle: {
          make: data.make,
          model: data.model,
          year: data.year,
          engine: data.engine
        },
        additional_context: data.context
      });
      return response.data;
    }
  });

  const addCode = () => setCodes([...codes, '']);

  return (
    <div className="min-h-screen bg-gray-900 text-white p-8">
      <div className="max-w-6xl mx-auto">
        <h1 className="text-5xl font-bold mb-12">Autó Diagnosztika</h1>

        <div className="grid grid-cols-2 gap-8">
          {/* Input Section */}
          <div className="bg-gray-800 p-8 rounded-lg">
            <h2 className="text-3xl font-bold mb-6">Hibakódok</h2>

            {codes.map((code, idx) => (
              <div key={idx} className="mb-4">
                <label className="block text-xl mb-2">Kód {idx + 1}</label>
                <input
                  type="text"
                  value={code}
                  onChange={(e) => {
                    const newCodes = [...codes];
                    newCodes[idx] = e.target.value.toUpperCase();
                    setCodes(newCodes);
                  }}
                  className="w-full text-2xl px-6 py-4 bg-gray-700 border-2 border-gray-6
                  placeholder="P0300"
                  maxLength={5}
                />
              </div>
            ))}

            <button
              onClick={addCode}
              className="w-full py-4 text-xl bg-gray-700 hover:bg-gray-600 rounded-lg mt-
            >
              + További kód
            </button>

            <div className="mt-8">
              <h3 className="text-2xl font-bold mb-4">Jármű Adatok</h3>

              <input
                {...register('make', { required: true })}
                placeholder="Gyártmány (pl. Volkswagen)"
                className="w-full text-xl px-6 py-4 bg-gray-700 rounded-lg mb-4"
              />

              <input
                {...register('model', { required: true })}
                placeholder="Modell (pl. Golf)"
                className="w-full text-xl px-6 py-4 bg-gray-700 rounded-lg mb-4"
              />

              <input
                {...register('year', { required: true, valueAsNumber: true })}
                type="number"
                placeholder="Évjárat (pl. 2015)"
                className="w-full text-xl px-6 py-4 bg-gray-700 rounded-lg mb-4"
              />

              <textarea
                {...register('context')}
                placeholder="További információk (opcionális)"
                className="w-full text-lg px-6 py-4 bg-gray-700 rounded-lg h-32"
              />
            </div>

            <button
              onClick={handleSubmit((data) => diagnosisMutation.mutate({ ...data, codes }
              disabled={diagnosisMutation.isPending}
              className="w-full py-6 text-2xl font-bold bg-blue-600 hover:bg-blue-700 rou
            >
              {diagnosisMutation.isPending ? 'Elemzés...' : 'Diagnosztika Indítása'}
            </button>
          </div>

          {/* Results Section */}
          <div className="bg-gray-800 p-8 rounded-lg overflow-y-auto max-h-screen">
            {diagnosisMutation.isPending && (
              <div className="text-center py-20">
                <div className="animate-spin rounded-full h-32 w-32 border-b-2 border-blu
                <p className="text-2xl mt-8">Diagnosztika folyamatban...</p>
              </div>
            )}

            {diagnosisMutation.data && (
              <DiagnosisResults data={diagnosisMutation.data.data.diagnosis} />
            )}
          </div>
        </div>
      </div>
    </div>
  );
}

function DiagnosisResults({ data }: { data: any }) {
  return (
    <div>
      <div className="mb-6">
        <div className="flex items-center justify-between mb-2">
          <h2 className="text-3xl font-bold">Diagnosztika Eredmény</h2>
```

```
            <div className="text-xl">
              Bizonyosság: <span className="font-bold text-blue-400">{(data.confidence * 10
            </div>
          </div>
          <p className="text-xl text-gray-300">{data.primary_issue}</p>
        </div>

        <div className="mb-8">
          <h3 className="text-2xl font-bold mb-4">Lehetséges Okok</h3>
          {data.codes_analyzed?.map((code: any, idx: number) => (
            <div key={idx} className="bg-gray-700 p-6 rounded-lg mb-4">
              <div className="flex items-center justify-between mb-3">
                <span className="text-2xl font-mono font-bold text-yellow-400">{code.code}<
                <span className={`text-lg px-4 py-1 rounded ${
                  code.severity === 'critical' || code.severity === 'high' ? 'bg-red-600' :
                  code.severity === 'medium' ? 'bg-yellow-600' : 'bg-green-600'
                }`}>
                  {code.severity}
                </span>
              </div>

              {code.potential_causes?.map((cause: any, cidx: number) => (
                <div key={cidx} className="mb-3 pl-4 border-l-4 border-blue-500">
                  <div className="flex items-center justify-between">
                    <p className="text-lg font-semibold">{cause.cause}</p>
                    <span className="text-sm bg-gray-600 px-3 py-1 rounded">{cause.probabil
                  </div>
                  <p className="text-gray-400 mt-1">{cause.reasoning}</p>
                </div>
              ))}
            </div>
          ))}
        </div>

        <div>
          <h3 className="text-2xl font-bold mb-4">Ajánlott Lépések</h3>
          {data.recommended_actions?.map((action: any, idx: number) => (
            <div key={idx} className="bg-gray-700 p-6 rounded-lg mb-4">
              <div className="flex items-center justify-between mb-2">
                <h4 className="text-xl font-bold">Lépés {action.step}</h4>
                <span className={`px-4 py-1 rounded ${
                  action.priority === 'high' ? 'bg-red-600' :
                  action.priority === 'medium' ? 'bg-yellow-600' : 'bg-green-600'
                }`}>
                  {action.priority}
                </span>
              </div>
              <p className="text-lg mb-2">{action.action}</p>
              <div className="flex gap-4 text-sm text-gray-400">
                <span>□ {action.estimated_time}</span>
                <span>□ {action.tools_needed?.join(', ')}</span>
              </div>
            </div>
          ))}
        </div>
```

```
      {data.warnings && data.warnings.length > 0 && (
        <div className="mt-6 bg-red-900 border-2 border-red-600 p-6 rounded-lg">
          <h3 className="text-xl font-bold mb-2">⚠ Figyelmeztetések</h3>
          <ul className="list-disc list-inside">
            {data.warnings.map((warning: string, idx: number) => (
              <li key={idx} className="text-lg">{warning}</li>
            ))}
          </ul>
        </div>
      )}
    </div>
  );
}
```

Deployment (Railway)

```
# Backend deployment
cd backend
railway up

# Environment variables beállítása Railway dashboardon:
# DATABASE_URL (auto-set)
# OPENAI_API_KEY
# NODE_ENV=production

# Frontend deployment (Vercel)
cd frontend
npm install -g vercel
vercel --prod
```

Költségbecslés MVP-re

```
Fejlesztés (Te csinálod):
- Időbefektetés: 8-10 hét @ 20-30 óra/hét
- Költség: €0 (saját munka)

Infrastruktúra (havonta):
- Railway PostgreSQL: $5/hó (Hobby tier)
- Railway Backend hosting: $5/hó
- Vercel Frontend: $0 (Hobby tier, elég MVP-hez)
- OpenAI API: ~$50-100/hó (100-200 diagnosis)
- Domain: €10/év
TOTAL: ~€60-110/hó MVP futtatására

Adatforrások:
- OBD kódok: €0 (publikus adatok)
- Járműadatok: €0-50/hó (kezdetben manuális, később API)
- NHTSA TSB: €0 (ingyenes)
```

Adatminőség Biztosítása

1. Validáció & Testing

```
// Test cases OBD kódokhoz
const testCases = [
  {
    code: "P0300",
    vehicle: { make: "Volkswagen", model: "Golf", year: 2015 },
    expectedCategories: ["ignition", "fuel", "mechanical"]
  },
  // ... több
];

// Minden új kód manuális ellenőrzése 3 forrásból
```

2. Feedback Loop (Phase 2)

```
// Szerelői visszajelzés gyűjtése
interface MechanicFeedback {
  sessionId: string;
  wasAccurate: boolean;
  actualProblem: string;
  timesSaved: string;
}
```

3. Continuous Improvement

- **Havonta:** Top 10 leggyakoribb kód elemzése
- **Negyedévente:** Új autómodellek hozzáadása
- **Feedback alapján:** Pontosság javítása

Következő Lépések

1. ✅ **Most azonnal:** PostgreSQL + Prisma setup Railway-en
2. ✅ **Hét 1:** Top 100 OBD kód manuális feltöltése
3. ✅ **Hét 1-2:** Top 50 jármű adatbázis (VW, Skoda, Ford, Opel, BMW)
4. ✅ **Hét 3:** OpenAI integráció + prompt engineering
5. ✅ **Hét 4:** Backend API befejezése
6. ✅ **Hét 5-6:** React frontend tablet-optimalizálással
7. ✅ **Hét 7:** Testing valódi mechanikusokkal
8. ✅ **Hét 8:** Beta launch 2-3 műhellyel

Kezdjük a Database setup-pal? Vagy inkább az LLM prompt engineering részletezését szeretnéd?

⁂

1. https://www.microbiologyresearch.org/content/journal/mgen/10.1099/mgen.0.001287
2. https://academic.oup.com/gpb/advance-article-pdf/doi/10.1093/gpbjnl/qzae047/58319729/qzae047.pdf
3. https://pmc.ncbi.nlm.nih.gov/articles/PMC7015990/
4. http://arxiv.org/pdf/2409.04824.pdf
5. https://arxiv.org/pdf/2206.12653.pdf
6. https://academic.oup.com/nar/article/53/D1/D495/7848843
7. https://pmc.ncbi.nlm.nih.gov/articles/PMC1669712/
8. https://academic.oup.com/nar/advance-article-pdf/doi/10.1093/nar/gkad1019/53691867/gkad1019.pdf
9. https://www.klavkarr.com/data-trouble-code-obd2.php
10. https://smartcar.com/blog/car-database-api
11. https://www.m1repair.com/mitchell1-prodemand-how-to-access-technical-service-bulletins-tsbs-campaigns-and-recalls
12. https://www.nature.com/articles/cddis2015380
13. https://obdii-trouble-codes.soft112.com
14. https://www.afteriize.com/api/
15. https://www.motor.com/products-services/data-products/technical-service-bulletins/
16. http://www.auto-consulting.it/dwn/dtc_database.pdf
17. https://api.auto-data.net
18. https://www.youtube.com/watch?v=rcArQGlIBH4
19. https://obd-ll-codes.en.aptoide.com/app
20. https://www.openmart.com/data/car-repair
21. https://www.reddit.com/r/cars/comments/1ihy4x/where_to_find_tsbs/
22. https://apps.microsoft.com/detail/9wzdncrdfx27?hl=hu-HU
23. https://www.nature.com/articles/srep30624
24. https://www.reddit.com/r/webdev/comments/1qeqqtj/building_a_car_app_here_are_the_10_best/
25. https://www.diagnosedan.com/automotive-technical-service-bulletins-tsbs-oem-vs-aftermarket/
26. https://www.preprints.org/manuscript/202108.0259/v1
27. https://www.semanticscholar.org/paper/e8fd09f69739d0e5c068bd12ffbc6139a6ae865b
28. https://www.semanticscholar.org/paper/ebd47d69ab0a97eae28ddf7329efa5f5f7f2e193
29. https://www.atlantis-press.com/article/9981
30. https://www.jstor.org/stable/343101?origin=crossref
31. https://pmc.ncbi.nlm.nih.gov/articles/PMC10767886/