AutoCognitix: Fejlett AI-alapú Járműdiagnosztikai és Döntéstámogató Rendszer – Architektúrális Terv és Megvalósíthatósági Tanulmány

1. Vezetői Összefoglaló

Az autóipar technológiai robbanása, a szoftvervezérelt járművek (Software Defined Vehicles - SDV) térnyerése és a diagnosztikai adatok exponenciális növekedése paradigmaváltást kényszerít ki a járműjavítási szektorban. A modern autószerelő műhelyekben a hagyományos tudásbázisok és a tapasztalati úton szerzett ismeretek már nem elegendőek a komplex, egymással összefüggő hibajelenségek gyors és pontos feltárásához. Jelen jelentés egy olyan táblagép-alapú, mesterséges intelligenciával (AI) támogatott diagnosztikai szoftverplatform – munkanéven **AutoCognitix** – részletes műszaki tervét és megvalósíthatósági elemzését tartalmazza, amely kifejezetten a professzionális autószerelők igényeire lett szabva.

A projekt célja egy olyan rendszer létrehozása, amely áthidalja a determinisztikus gépi adatok (OBD-II hibakódok) és a probabilisztikus emberi megfigyelések (ügyfélpanaszok, hanghatások, tesztvezetés) közötti szakadékot. A javasolt megoldás egy **Hibrid Retrieval-Augmented Generation (RAG)** architektúrára épül, amely egyesíti a Nagy Nyelvi Modellek (LLM) következtetési képességét a strukturált Tudásgráfok (Knowledge Graph) precizitásával.[1] A rendszer képes kezelni a strukturált hibakódokat és a strukturálatlan tünetleírásokat, integrálja a gyártói javítási útmutatókat (OEM Manuals) és a Technikai Szervizközleményeket (TSB), miközben egy ipari környezetre optimalizált felhasználói élményt (UX) biztosít.

A jelentés részletesen tárgyalja a backend és frontend architektúrát, a multi-ágens rendszerek (Multi-Agent Systems) alkalmazását a diagnosztikai logika vezérlésére, valamint azokat a kritikus "edge case"-eket és hibaágakat, amelyek kezelése elengedhetetlen a műhelykörnyezetben való biztonságos működéshez.

---

2. Piaci Helyzetkép és Felhasználói Követelmények Elemzése

2.1 A Diagnosztikai Rés (The Diagnostic Gap)

A jelenlegi piaci megoldások többsége, mint a Mitchell1 ProDemand vagy a HaynesPro,

hatalmas digitális könyvtárként funkcionál.[3] Bár tartalmazzák a szükséges információkat, a szerelőtől várják el, hogy tudja, mit és hol keressen. Ezzel szemben a "fogyasztói" OBD-olvasók (pl. Carly, BlueDriver) csupán a hibakód definícióját adják meg, kontextus és ok-okozati összefüggések nélkül.[5]

A felhasználói igény egyértelműen egy olyan "Szakértő Másodpilóta" (Expert Co-Pilot) irányába mutat, amely képes szintetizálni a különböző forrásból származó adatokat:

- **Adatforrás A (OBD):** P0171 hibakód (Rendszer túl szegény, 1. hengersor).
- **Adatforrás B (Szerelői megfigyelés):** Sziszegő hang a szívósor környékéről hidegindításkor.
- **Adatforrás C (Ügyfélpanasz):** "Rángat az autó piros lámpánál."
- **Adatforrás D (TSB):** Gyártói közlemény a PCV szelep membránjának repedéséről adott évjáratnál.

Egy hagyományos keresőmotor négy különálló dokumentumot adna vissza. Az AutoCognitix célja, hogy ezeket korrelálja, és egyetlen, valószínűsített diagnózist adjon: "Vákuumszivárgás a PCV szelepnél, amelyet a P0171 kód és a sziszegő hang igazol, összhangban a #12-34-56 sz. TSB-vel."

2.2 Funkcionális Követelmények (Functional Requirements - FR)

Az alábbi táblázat összegzi a felhasználói kérésből levezetett kritikus rendszerkövetelményeket:

Azonosító	Követelmény	Technikai Implikáció
FR-01	**Hibrid Bemenet Kezelése**	A rendszernek egyaránt fel kell dolgoznia a strukturált OBD-II adatfolyamot és a természetes nyelvű (szöveg/hang) bemenetet.
FR-02	**Járműazonosítás (VIN Decoding)**	Pontos Évjárat/Gyártmány/Típus/Motorizáció (YMME) feloldása alvázszám alapján a releváns javítási adatok szűréséhez.[7]
FR-03	**Szemantikus Értelmezés**	Az ügyfél "furcsa hang" leírását össze kell vetni a

		szerelő "kopogás terhelésen" megfigyelésével és a technikai adatbázissal.
FR-04	**Ipari UX/UI**	A táblagépes felületnek kezelhetőnek kell lennie olajos kézzel, kesztyűben, zajos és rossz megvilágítású környezetben.[8]
FR-05	**Állandó Kapcsolat Kihasználása**	A felhőalapú erőforrások (LLM inferencia) maximalizálása, de a hálózati ingadozások (packet loss) tolerálása a felületen.
FR-06	**Döntéstámogatás, nem Döntéshozatal**	A rendszernek javaslatokat (hipotéziseket) kell adnia valószínűségi mutatókkal, nem "fekete doboz" döntéseket.

2.3 Felhasználói Persona: A Modern Autószerelő

A szoftver tervezésekor a célcsoport (Persona) tulajdonságai határozzák meg a UX döntéseket:

- **Időnyomás:** Minden perc, amit a számítógép előtt töltenek adminisztrációval vagy kereséssel, kiesett bevétel (non-billable hours).
- **Környezet:** A műhely fizikai valósága magában foglalja a szennyeződéseket (olaj, zsír), a zajt (légkulcsok, motorzaj) és a változó fényviszonyokat (akna, motortér).
- **Szakértelem:** A felhasználók skálája a tanulótól (akinek lépésről lépésre útmutató kell) a mesterig terjed (akinek csak a kapcsolási rajzra és a nyomatékadatokra van szüksége). A rendszernek adaptívnak kell lennie.
- **Bizalom:** A szerelők szkeptikusak a "fekete doboz" technológiával szemben. Ha a szoftver azt mondja, "Cseréld ki az ECU-t", a szerelő nem fogja megtenni indoklás nélkül. A rendszernek transzparensnek kell lennie: "Azért javaslom az ECU cseréjét, mert a 3-as lábon mért feszültség 0V, miközben a kapcsolási rajz szerint 5V-nak kellene lennie."

---

3. Rendszerarchitektúra és a "Kognitív Motor"

A felhasználói kérdés egyik központi dilemmája: *"nem tudom hogy ez egy custom agent legyen egy api-val vagy integrálva legyen egy llm model."*

A szakértői javaslat: Semmiképpen nem egyetlen monolitikus LLM modellre (pl. csak egy ChatGPT promptra) kell építeni a rendszert, mivel az hajlamos a hallucinációra (valótlan tények generálása) és nem rendelkezik naprakész, specifikus tudással.[9]

A helyes megközelítés egy **Multi-Agent Orchestration (Több-Ágensű Vezérlés)** architektúra, amely egy **Custom Agent keretrendszert** (pl. LangGraph vagy CrewAI) használ, és integrálja a **Foundational LLM-eket** (mint gondolkodó motort) külső **API eszközökkel** (mint tudásbázis). Ezt hívjuk **Agentic RAG** megközelítésnek.[1]

3.1 Magas Szintű Topológia

A rendszer három logikai rétegre bontható:

1. **Edge Réteg (Tablet/Jármű):** Valós idejű adatgyűjtés, felhasználói interakció, előfeldolgozás.
2. **Orchestrációs Réteg (Cloud API):** Munkamenet-kezelés, ágensek koordinációja, biztonsági szűrők.
3. **Kognitív Réteg (Adat és AI):** LLM inferencia, Tudásgráf, Vektorkönyvtárak.

Architektúrális Diagram Leírása:

- **Jármű** <--(Bluetooth/Wi-Fi)--> **Tablet App (Flutter)** <--(HTTPS/WSS)--> **API Gateway (FastAPI)**
- **API Gateway** --> **Ágens Vezérlő (LangGraph)**
- **Ágens Vezérlő** <--> **LLM (GPT-4o)** (Okos következtetés)
- **Ágens Vezérlő** <--> **Tudásgráf (Neo4j)** (Strukturált tudás: Alkatrész -> Hiba)
- **Ágens Vezérlő** <--> **Vektor Adatbázis (Pinecone)** (Szemantikus keresés: Tünetleírások)

3.2 A Multi-Ágens Rendszer Felépítése

Egyetlen AI ügynök helyett specializált ágensek dolgoznak együtt, mint egy virtuális konzílium:

1. **Triage Agent (Osztályozó):** Elemzi a bemenetet. Eldönti, hogy OBD-kód alapú, tünet alapú, vagy adminisztratív kérdésről van-e szó.
2. **Data Retrieval Agent (Kutató):** API hívásokat generál. "Keresd meg a P0300 kódot a 2018-as Ford Focus adatbázisban" és "Keresd meg a TSB-ket rángatásra".
3. **Diagnostic Agent (Diagnoszta):** Ez a "Sherlock Holmes". Nem keres adatot, hanem a Kutató által hozott információk alapján hipotéziseket állít fel. Összeköti a pontokat: "A P0300 (égéskimaradás) és a P0171 (szegény keverék) együtt valószínűleg vákuumszivárgást jelent, nem gyertyahibát."
4. **Safety & Validation Agent (Ellenőr):** Mielőtt a választ kiküldi a tabletnek, ellenőrzi a

biztonsági előírásokat (pl. "Figyelmeztetés: Magasfeszültségű rendszer, áramtalaníts!") és szűri a hallucinációkat.[11]

3.3 Hibrid RAG: Gráf és Vektor Szimbiózisa

Miért nem elég egy egyszerű vektoros keresés (Standard RAG)? A vektoros keresés a szemantikai hasonlóságot találja meg. Ha rákeresünk a "Motor túlmelegedés" kifejezésre, találhat dokumentumokat a "Klíma túlmelegedéséről" is. A járműdiagnosztikában azonban szigorú **kauzális (ok-okozati)** és **hierarchikus** kapcsolatok vannak, amelyeket egy **Tudásgráf (Knowledge Graph)** tud leképezni.[12]

- **Gráf Kapcsolat:** (Termosztát) ----> (Hűtőfolyadék Áramlás) ----> (Motor Hőmérséklet).
- Ha a szerelő "túlmelegedést" jelent, a Gráf alapú keresés (GraphRAG) képes végigmenni ezen az útvonalon, és javasolni a termosztát ellenőrzését, még akkor is, ha a "termosztát" szó nem szerepelt a bemenetben.
- **Hibrid Megközelítés:** A Vektor adatbázis (Pinecone) kezeli a "furcsa hangok" leírását, a Gráf (Neo4j) pedig a fizikai alkatrészek logikáját.

---

4. Backend Tervezés: Adatfeldolgozás és API

A rendszer "agya" a felhőben fut, kihasználva a skálázhatóságot és az állandó internetkapcsolatot.

4.1 Adatbetöltési Pipeline (Data Ingestion)

A legnagyobb kihívás a nyers, PDF formátumú szervizkönyvek és kapcsolási rajzok feldolgozása. A hagyományos szövegkinyerés (OCR) elveszíti a struktúrát (pl. táblázatok, ábrák feliratai).

Megoldás: Multimodális Dokumentumfeldolgozás A **LayoutLMv3** vagy az Adobe **DocLLM** modelljét kell alkalmazni.[14] Ezek a modellek nemcsak a szöveget "olvassák", hanem értik a dokumentum vizuális elrendezését is.

- **Folyamat:**
  1. **Szegmentálás:** A PDF oldalakat szekciókra bontja (pl. "Hibakeresési Táblázat", "Nyomatékadatok").
  2. **Entitás Kiemelés (NER):** Kinyeri a kulcsadatokat: Alkatrész Neve, Cikkszám, Meghúzási Nyomaték.
  3. **Tudásgráf Építés:** Az entitásokat betölti a gráf adatbázisba a **VSSo (Vehicle Signal Specification Ontology)** szabvány szerint.[16] Ez biztosítja, hogy a rendszer tudja: a "Coolant Temp Sensor" és az "ECT Sensor" ugyanaz az alkatrész.

4.2 API Struktúra (FastAPI)

Az API-nak állapotmentesnek (stateless) kell lennie a skálázhatóság miatt, de a diagnosztikai folyamat állapotát (session context) egy gyorsítótárban (Redis) kell tárolni.

Végpont Tervezet (Példa):

- POST /api/v1/session/init: Új diagnosztikai menet indítása (VIN, Szerelő ID).
- POST /api/v1/diagnosis/obd: Hibakódok beküldése.
  - *Payload:* { "dtc_list": ["P0300", "P0171"], "freeze_frame": { "rpm": 850, "ect": 92 } }
- POST /api/v1/diagnosis/symptom: Tünetleírás beküldése.
  - *Payload:* { "description": "Kattogó hang bal elöl kanyarodáskor", "conditions": ["turning_left", "low_speed"] }
- GET /api/v1/repair/procedure/{procedure_id}: Konkrét javítási lépések lekérése.

4.3 Integráció Külső Szolgáltatókkal

A "custom agent" nem tudhat mindent fejből. Szükséges a **Motor.com** vagy hasonló szolgáltatók (pl. HaynesPro) API-jának integrálása.[4]

- **TSB (Technical Service Bulletins):** Kritikus fontosságú. A gyártók gyakran adnak ki szoftverfrissítéseket vagy módosított alkatrészeket típushibákra. Az Ágensnek elsőbbséget kell adnia a TSB találatoknak a generikus diagnózissal szemben.

---

5. Frontend és Ipari UX Tervezés

A táblagépes alkalmazásnak (Tablet App) meg kell felelnie az ipari környezet követelményeinek.

5.1 Technológiák: Flutter vs. React Native

A jelentés a **Flutter** keretrendszert javasolja.[19]

- **Teljesítmény:** A Flutter Skia motorja közvetlenül a GPU-ra renderel, ami elengedhetetlen a sima görgetéshez nagy felbontású kapcsolási rajzok (SVG) vagy valós idejű élő adat (Live Data) grafikonok esetén.
- **Hardver elérés:** A Flutter FFI (Foreign Function Interface) kiválóan kezeli a Bluetooth Low Energy (BLE) kommunikációt az OBD dongle-lel.

5.2 Felhasználói Felület (UI) Elvek - "Grease Mode"

A műhelyben dolgozó szerelők keze gyakran olajos, vagy kesztyűt viselnek. A UX designnak ezt tükröznie kell [8]:

1. **"Fat Finger" Design:** Minden interaktív elemnek (gomb) legalább 72x72 dp méretűnek kell lennie (kb. 15mm fizikai méret), szemben a fogyasztói appok 48dp szabványával.
2. **Magas Kontrasztú Sötét Mód (Dark Mode):** Alapértelmezett. A sötét motortérben vagy

az autó alatt a vakító fehér képernyő zavaró, és gyorsabban meríti az OLED kijelzőt. A szövegnek tiszta fehérnek kell lennie sötétszürke háttéren.

3. **Hierarchikus Információközlés:** A képernyőn egyszerre csak a legfontosabb információ jelenjen meg.
    - *Rossz:* 50 soros szöveg a motor működéséről.
    - *Jó:* Egy kártya: "Ellenőrizd a 4-es henger gyújtótrafóját." -> "Hogyan?" gomb a részletekért.

5.3 Hangvezérlés (VoiceUI) - "Hands-Free" Működés

Mivel a szerelő keze foglalt, a hangvezérlés nem "extra", hanem alapkövetelmény.[21]

- **Megvalósítás:** "Wake Word" detektálás az eszközön (pl. "Hey AutoBot"), majd a parancs streamelése a felhőbe.
- **Kihívás:** Műhelyzaj (légkulcs, kompresszor).
- **Megoldás:** Zajszűrő mikrofon-array használata a tableten, és olyan ASR (Automatic Speech Recognition) modell finomhangolása (pl. OpenAI Whisper), amely ismeri a szakzsargont ("lambdaszonda", "szívósor", "kardánkereszt").

---

6. Diagnosztikai Folyamatok (User Flows)

Két fő felhasználói útvonalat (User Flow) kell megtervezni a kérésnek megfelelően.

6.1 Útvonal A: Determinisztikus Diagnosztika (OBD Kóddal)

Ez az "ideális" eset, amikor az autó ECU-ja jelzi a hibát.

1. **Csatlakozás:** A szerelő bedugja a Bluetooth OBD adaptert. A tablet automatikusan párosodik és kiolvassa az alvázszámot (VIN) és a tárolt hibakódokat (DTC).
2. **Adatgazdagítás:** A rendszer lekéri a VIN alapján a pontos felszereltséget.
3. **Ágens Elemzés:**
    - A bemenet: P0420 (Katalizátor hatásfok alacsony) + Toyota Prius 2012.
    - Az Ágens lekérdezi a Tudásgráfot.
    - *Logika:* Prius esetén a P0420 gyakran nem a katalizátor hibája, hanem a kipufogórendszer tömítetlensége a leömlőnél.
4. **Eredmény Megjelenítése:**
    - **Prioritás 1 (90%):** "Tömítetlenség a kipufogó leömlőnél." -> *Ellenőrzési lépés:* "Fújj szappanos vizet a csatlakozásra hideg motornál."
    - **Prioritás 2 (10%):** "Katalizátor elöregedése." -> *Ellenőrzési lépés:* "O2 szenzor jelalak vizsgálata."

6.2 Útvonal B: Probabilisztikus Diagnosztika (Tünet Alapján)

Ez a "nincs kód" eset, ahol a szerelő és az ügyfél megfigyelései dominálnak.

1. **Jármű Bevitel:** VIN beolvasása kamerával (OCR) vagy manuális kiválasztás (Év/Gyártmány/Modell).
2. **Tünetinterjú (Chat/Voice):**
    - *Szerelő:* "Az ügyfél szerint rángat az autó kigyorsításkor. Nincs hibakód. Tesztúton éreztem, hogy főleg alacsony fordulaton csinálja."
3. **Szemantikus Feldolgozás:**
    - Az Ágens kinyeri az entitásokat: {Symptom: "Hesitation/Jerking", Condition: "Acceleration", Range: "Low RPM", DTC: "None"}.
4. **Hipotézis Generálás:**
    - Az LLM a gráfban keresi az összefüggéseket: "Rángatás kód nélkül" -> Gyújtásrendszer (gyenge szikra, ami még nem éri el a misfire küszöböt) VAGY Üzemanyagnyomás ingadozás.
5. **Interaktív Pontosítás (Test Drive Input):**
    - *Ágens:* "Milyen a hangja a rángatáskor? Hallani csattanást vagy durrogást?"
    - *Szerelő:* "Nem, csendes, csak megtorpan."
6. **Javaslat:**
    - "Tisztítsd meg a légtömegmérőt (MAF) és ellenőrizd a gyertyahézagokat. A típusra jellemző a gyertyák idő előtti kopása."

6.3 Tesztvezetés és Megfigyelés Rögzítése

A kérés specifikus eleme a tesztvezetés eredményeinek rögzítése.

- **UI Implementáció:** Egy dedikált "Test Drive Mode" képernyő nagy gombokkal: "Rángat", "Zaj", "Fékrázás".
- **Adatrögzítés:** A tablet gyorsulásmérőjének (accelerometer) adatait is rögzíthetjük a tesztút alatt, hogy objektív adatot kapjunk a vibrációról, amit az AI elemezhet.

---

7. Speciális Modulok és "Edge Case" Kezelés

7.1 Akusztikus Diagnosztika (Audio Spectrogram Analysis)

A "milyen hangot" kérdésre a szöveges válasz szubjektív. ("Kattog", "Kelepel", "Koppan" - mindenki mást ért alatta).

- **Megvalósítás:** A szoftver tartalmaz egy "Hangfelvétel" funkciót. A rögzített 10 másodperces hangmintát a rendszer Mel-spektrogrammá alakítja, és egy konvolúciós neurális hálózat (CNN) osztályozza.[23]
- **Kimenet:** A modell nem azt mondja, hogy "Hajtókarcsapágyas", hanem: "95% valószínűséggel fémes ütemes kopogás (Engine Knock), frekvencia a fordulatszámmal arányos." Ezt az objektív adatot kapja meg a Diagnosztikai Ágens.

7.2 Hibaágak és "A szerelő tévedése"

Mi történik, ha a szerelő rosszul ítéli meg a helyzetet?

- **Edge Case:** A szerelő azt írja: "Generátor hiba", de valójában csak az ékszíj csúszik.
- **Kezelés:** Az ágensnek nem szabad vakon elfogadnia a szerelő diagnózisát tényként. A rendszernek *ellenőrző kérdéseket* kell feltennie: "Ellenőrizted az ékszíj feszességét a generátor cseréje előtt?"
- **Biztonsági Fék:** Ha a rendszer elektromos/hibrid autót érzékel, és a szerelő a nagyfeszültségű kábelek közelében lévő alkatrészt említ, a szoftvernek kötelezően meg kell jelenítenie a "High Voltage Disable" procedúrát ("Pop-up warning").

7.3 Offline Működés

Bár a követelmény "állandó internetkapcsolat", a valóságban a Wi-Fi megszakadhat.

- **Caching:** A munkamenet indításakor (VIN azonosítás) a rendszer töltse le a járműre vonatkozó *leggyakoribb* hibák adatbázisát és a kapcsolási rajzokat a gyorsítótárba.
- **Lokális Fallback:** Ha nincs net, az AI "gondolkodás" nem működik, de a szoftvernek vissza kell butulnia egy hagyományos OBD-olvasó és kézikönyv-nézegető szintre, hogy a munka ne álljon meg.

---

8. Adatstratégia és Jogi Megfontolások

8.1 Adatforrások és Licencelés

A rendszer nem működhet lopott PDF-ekkel. Professzionális felhasználásra hivatalos adatlicencekre van szükség.

- **OBD Adatok:** Az SAE J2012 szabvány [25] definiálja a generikus kódokat.
- **Gyártói Adatok (OEM):** Az Egyesült Államokban a "Right to Repair" törvények, Európában a csoportmentességi rendelet (BER) biztosítja a független fejlesztők hozzáférését a javítási adatokhoz, de ez fizetős (pl. Motor.com DaaS szolgáltatás).[18]
- **Közösségi Adat:** Fórumok (pl. Reddit MechanicAdvice, iATN) "bányászata" értékes, de jogilag aggályos és zajos adat. Ezt csak tisztítva, vektoros formában, "tippként" szabad felhasználni, nem hivatalos eljárásként.

8.2 Felelősségkizárás (Liability)

Ha az AI rossz tanácsot ad, és a fék nem fog, ki a felelős?

- **Jogi Keret:** A szoftvert "Döntéstámogató Eszközként" (Decision Support Tool) kell definiálni, nem "Autonóm Szerelőként".
- **Disclaimer:** Minden diagnózis végén szerepelnie kell: *"A fenti javaslat statisztikai valószínűségen alapul. A javítás megkezdése előtt fizikai ellenőrzés szükséges."*.[26]

---

9. Megvalósíthatóság és Ütemterv

9.1 Technológiai Stack Összefoglalása

Komponens	Technológia	Indoklás
Frontend	Flutter (Dart)	Nagy teljesítmény, multi-platform, erős Bluetooth támogatás.
Backend API	Python (FastAPI)	Gyors fejlesztés, kiváló AI könyvtárak (LangChain, PyTorch).
LLM Model	GPT-4o (Reasoning) + Llama-3 (Extraction)	A GPT-4o a "gondolkodó", a Llama-3 (olcsóbb) az adatfeldolgozó.
Adatbázis	Neo4j (Gráf) + Pinecone (Vektor) + PostgreSQL	Hibrid RAG támogatása.
OBD Hardver	ELM327 / STN1170	Szabványos, olcsó, széleskörűen támogatott.

9.2 Implementációs Roadmap

1. **Fázis 1 (MVP - 3 hónap):**
   - Csak OBD-olvasás (Flutter app).
   - Alapvető hibakód-keresés (Statikus adatbázis).
   - Nincs AI, csak kódolvasás.
2. **Fázis 2 (Data Pipeline - 6 hónap):**
   - OEM kézikönyvek feldolgozása (LayoutLMv3).
   - Tudásgráf építése egy kiválasztott márkára (pl. Volkswagen csoport).
3. **Fázis 3 (Agentic AI - 9 hónap):**
   - LangGraph ágensek implementálása.
   - Tünet alapú keresés bevezetése.
   - Béta teszt partnerszervizekkel.
4. **Fázis 4 (Audio & Voice - 12 hónap):**
   - Hangfelismerés és akusztikus diagnosztika integrálása.

10. Konklúzió

Az AutoCognitix projekt technológiailag megvalósítható, és valós piaci igényt elégít ki. A kulcs nem önmagában az AI modell (GPT), hanem annak integrációja a strukturált autóipari adatokkal (Tudásgráf) és a műhelykörnyezetre optimalizált felhasználói felülettel. A javasolt hibrid architektúra (Determinisztikus OBD + Probabilisztikus AI) biztosítja a legnagyobb pontosságot és bizalmat a professzionális felhasználók körében. A fejlesztés legnagyobb kockázata nem a szoftver, hanem a jogtiszta és minőségi szervizadatok beszerzése és strukturálása.

A Vezetői Összefoglaló vége. A részletes jelentés következik.

1. Rendszertervezés és Specifikáció: AutoCognitix Platform

1. Bevezetés és Projekt Hatókör

A modern járművek komplexitása meghaladta az emberi memória és a lineáris hibakeresési módszerek határait. Egy átlagos prémium kategóriás autó több mint 100 vezérlőegységet (ECU) tartalmaz, és a hibák gyakran nem ott jelentkeznek, ahol a kiváltó ok található (pl. egy szenzorhiba szoftveres leállást okozhat egy teljesen más alrendszerben). A projekt célja egy olyan szoftverökoszisztéma létrehozása, amely a szerelőt nem helyettesíti, hanem képességeit kiterjeszti (Augmented Intelligence).

1.1 A Feladat Definíciója

A felhasználói igény egy kétkomponensű diagnosztikai rendszerre vonatkozik:

1. **Hardver-szoftver interfész:** Kommunikáció a járművel OBD-II porton keresztül.
2. **Kognitív asszisztens:** Egy AI ágens, amely értelmezi a gépi és emberi bemeneteket, és megoldási javaslatokat ad.

1.2 Célközönség és Felhasználási Környezet

A szoftver végfelhasználói autószerelők. Ez a demográfia technológiailag változatos felkészültségű, pragmatikus, és eredményorientált.

- **Fizikai környezet:** A műhelyben a tabletet leejthetik, koszos kézzel nyomkodják, a háttérzaj magas. A UX-nek ezt "Industrial Grade" megoldásokkal kell kezelnie.
- **Pszichológiai környezet:** A szerelő a felelős az autóért. Ha a szoftver téved, a szerelő veszít presztízst (és pénzt). Ezért a rendszernek *magyarázhatónak* (Explainable AI - XAI) kell lennie. Nem elég annyit írni: "Cseréld ki az X alkatrészt". Oda kell írni: "Azért javaslom

az X cseréjét, mert a Y szenzor értéke a határértéken kívül esik."

---

2. Részletes Rendszerarchitektúra

A "Custom Agent vs. LLM integration" kérdésre a válasz: **Mindkettő**. Egy nyers LLM (pl. ChatGPT webes felület) nem integrálható biztonságosan ipari folyamatba. Szükség van egy keretrendszerre (Custom Agent), amely "pórázon tartja" az LLM-et.

2.1 Backend Architektúra (Felhő)

A backend mikroszolgáltatás (Microservices) alapú, konténerizált (Docker/Kubernetes) környezetben fut.

2.1.1 Az Ágens Vezérlő (Orchestrator) - LangGraph

A **LangGraph** keretrendszert javasoljuk a diagnosztikai folyamat vezérlésére.[27] A diagnosztika nem lineáris (A -> B -> C), hanem ciklikus (A -> Teszt -> Eredmény -> B -> Új Teszt -> C). A LangGraph lehetővé teszi ciklikus gráfok definiálását, ahol az ágens visszakérdezhet, ha nincs elég információja.

A Gráf Csomópontjai (Nodes):

1. **Input Analyzer:** Eldönti, hogy a bemenet strukturált (OBD kód) vagy strukturálatlan (szöveg).
2. **Context Builder:** Lekéri a jármű előéletét (szervizkönyv) és a típusra jellemző típushibákat a Tudásgráfból.
3. **Hypothesis Generator (LLM):** GPT-4o modell, amely a kontextus alapján felállít 3 lehetséges okot, valószínűségi sorrendben.
4. **Verification Engine:** Minden hipotézishez hozzárendel egy tesztelési eljárást (pl. "Mérd meg az ellenállást a két pont között").
5. **Critique & Safety:** Ellenőrzi, hogy a javasolt lépés biztonságos-e (pl. nem javasol üzemanyagrendszer bontást forró motornál).

2.1.2 Adatbázis Réteg - A Hibrid Megközelítés

A rendszer lelke az adat.

- **Gráf Adatbázis (Neo4j):** Tárolja a "fizikai világot".
  - Node: Vehicle, Part, DTC, Symptom.
  - Relationship: HAS_PART, MONITORS, CAUSES.
  - *Példa:* (EGR Szelep) --> (P0401) --> (Rángatás).
- **Vektor Adatbázis (Pinecone/Milvus):** Tárolja a "szemantikus világot".
  - Szervizkönyvek szöveges leírásai, fórum bejegyzések, tünetleírások beágyazásai (embeddings).
  - Ez teszi lehetővé, hogy a "furcsa nyiszogás" keresőkifejezés megtalálja a "serpentine

belt squeal" technikai leírást.
- **Relációs Adatbázis (PostgreSQL):** Felhasználói adatok, munkalapok, fizetési információk.

2.2 Frontend Architektúra (Tablet)

2.2.1 Keretrendszer: Flutter

A Flutter választása stratégiai döntés.[19]

- **Egységes Kód:** Android és iOS (iPad) támogatás egy kódbázisból.
- **OBD Kommunikáció:** A Flutter flutter_blue_plus csomagja és a natív C/C++ integráció lehetővé teszi a stabil, alacsony szintű kommunikációt az ELM327 chipekkel.
- **Renderelés:** A kapcsolási rajzok (Wiring Diagrams) interaktív megjelenítése (nagyítás, érintésre szálkiemelés) nagy grafikai teljesítményt igényel, amit a Flutter Skia motorja biztosít.

2.2.2 Hardver Követelmények

- **Tablet:** Android (Samsung Galaxy Tab Active sorozat - strapabíró, ipari) vagy iPad Pro tokban. Követelmény: jó minőségű mikrofon és kamera (vakuval) a sötét helyek fotózásához.
- **OBD Adapter:** A piac tele van olcsó kínai ELM327 klónokkal, amelyek instabilak. A szoftverhez ajánlott egy **saját márkás, vagy validált adapter** (pl. OBDLink MX+ vagy Vgate vLinker) használata, ami támogatja a gyorsabb adatátvitelt és a gyártó-specifikus protokollokat (MS-CAN, SW-CAN).

---

3. Adatfeldolgozás és "Tudásgyár" (Data Ingestion)

Ahhoz, hogy a rendszer okosabb legyen egy átlagos szerelőnél, fel kell dolgoznia a teljes szakirodalmat.

3.1 PDF-ből Tudásgráf (PDF-to-Graph Pipeline)

A gyártói javítási útmutatók (Manuals) PDF formátumban érhetők el. Ezek feldolgozása a legnagyobb technikai kihívás.

- **Layout Analysis:** A **LayoutLMv3** modell használata javasolt.[14] Ez a modell képes megkülönböztetni a szöveget a táblázatoktól és az ábráktól.
  - *Példa:* Ha egy táblázatban van a meghúzási nyomaték, a modell felismeri a sor-oszlop kapcsolatot, nem csak ömlesztett szövegként kezeli.
- **Hierarchikus Chunking:** A szöveget nem fix méretű darabokra vágjuk, hanem a dokumentum logikai szerkezete (Fejezet -> Alfejezet -> Eljárás) szerint. Így az AI mindig tudja, hogy a "Csavar eltávolítása" lépés a "Vízpumpa cseréje" folyamathoz tartozik.

3.2 TSB és Valós Idejű Adatok

A rendszernek folyamatosan figyelnie kell a **Technical Service Bulletin (TSB)** forrásokat (pl. NHTSA API, gyártói portálok).

- **Prioritási Logika:** Ha egy adott évjárathoz/hibához van TSB, az *mindig* felülírja a kézikönyv általános utasítását. Az Ágens logikájában a TSB node-ok magasabb súlyozást kapnak.

---

4. Felhasználói Élmény (UX) és Design Best Practices

A műhelyben a UX nem esztétikai, hanem biztonsági és hatékonysági kérdés.

4.1 "Grease Mode" UI Minta

A szoftvernek tartalmaznia kell egy dedikált üzemmódot szerelés közbeni használatra.

- **Gombok:** Minimum 15-20mm fizikai méret.
- **Interakció:** Swipe (húzás) gesztusok preferálása a pontos koppintás helyett (pl. "Következő lépés" -> Jobbra húzás a képernyőn bárhol).
- **Hang-visszajelzés (Text-to-Speech):** A rendszer olvassa fel az instrukciókat ("Lazítsd meg a 10-es csavart"), hogy a szerelőnek ne kelljen a képernyőre néznie munka közben.

4.2 A Hibakód (OBD) User Flow Részletesen

Lépés	Szerelő Tevékenysége	Rendszer Reakciója
1.	Csatlakoztatja az OBD dongle-t és elindítja az appot.	Automatikus Bluetooth párosítás, protokoll felismerés (ISO 15765 CAN).
2.	Megnyomja a "Scan" gombot.	Kiolvassa a Mode 03 (Jelenlegi) és Mode 07 (Függőben lévő) kódokat, valamint a Freeze Frame (pillanatkép) adatokat.
3.	Megtekinti a listát.	A kódok nem csak listázva vannak, hanem csoportosítva (Motor, Váltó, Fék). A rendszer letölti a

		kódok leírását.
4.	Kiválaszt egy kódot (pl. P0300).	Az Ágens elindítja a diagnosztikai folyamatot a háttérben.
5.	**Döntéstámogatás.**	A képernyőn megjelenik: **"Valószínű ok: Gyújtótrafó hiba (80%)"**. Alatta: "Ezt a P0300 kód és a Freeze Frame-ben látható magas motorterhelés támasztja alá."
6.	Interaktív Teszt.	Gomb: "Hengerlekapcsolás Teszt". A rendszer utasítja az ECU-t, hogy kapcsolja le az 1-es hengert. A szerelő figyeli a változást.

4.3 A Tünet Alapú (Symptom) User Flow és a "Hang" Probléma

A "milyen hangot" kérdés kezelése kritikus. A szóbeli leírás ("kattog") szubjektív.

- **Akusztikus Analízis Modul:** A szoftver képes hangmintát rögzíteni a tablet mikrofonjával.
  - **Technológia:** A hangmintát Mel-spektrogrammá alakítjuk, és egy CNN (Convolutional Neural Network) modell osztályozza.[23]
  - **Képzés:** Olyan adathalmazokon tanítva, mint az *OtoMobile* (motorhangok, csapágyhangok).
  - **Eredmény:** A rendszer objektíven megállapítja: "Fémes, ütemes kopogás, frekvenciája arányos a fordulatszámmal" (Rod Knock). Ezt az objektív adatot táplálja be az Ágensnek a "kattog" szó helyett.

A tesztvezetés rögzítése:

A szoftverben van egy "Test Drive" mód.

- **Szenzorok:** Rögzíti a tablet gyorsulásmérő (Accelerometer) adatait (vibráció detektálása) és a GPS sebességet.
- **Hangjegyzet:** A szerelő vezetés közben diktálhat: "Most csinálja, 3000-es fordulatnál." Az AI ezt később összefésüli az OBD élő adatokkal (Live Data), ha elérhető.

---

5. Kockázatok, Hibaágak és Edge Case-ek

5.1 Hibaág: A téves diagnózis felelőssége

Ha az AI azt mondja "Cseréld ki a féket", és baleset történik, ki a felelős?

- **Megoldás:** A rendszert jogilag "információs forrásként" kell pozicionálni. Minden javaslat mellett szerepelnie kell a forrásnak (pl. "Forrás: Ford Szervizkönyv 102. oldal").
- **UI:** Minden kritikus lépésnél egy "Confirm" gomb, amivel a szerelő igazolja, hogy elvégezte a saját ellenőrzést.

5.2 Edge Case: Az "Érzékelő hazudik"

Gyakori hiba, hogy egy szenzor rossz értéket ad, ami miatt az ECU más hibát dob (pl. rossz hőgomba miatt dúsít a motor -> lambda szonda hiba).

- **AI Logika:** Az Ágenst fel kell készíteni a "Racionalitás Vizsgálatra". Ha a hűtővíz hőmérséklet szenzor -40°C-ot mutat nyáron, az Ágensnek fel kell ismernie, hogy ez fizikai képtelenség (szakadás), és nem azt javasolni, hogy "takard le a hűtőt", hanem hogy "ellenőrizd a szenzor kábelezését".

5.3 Internetkapcsolat elvesztése

Bár a követelmény az állandó net, a valóságban lehetnek kimaradások.

- **Offline Mode:** A tabletnek tárolnia kell egy "Lite" adatbázist a leggyakoribb OBD kódokról és egy 4-bites kvantált "Small Language Model"-t (pl. Phi-3 vagy Llama-3-8B), ami képes alapvető tanácsokat adni internet nélkül is, még ha korlátozottan is.[29]

---

6. Összegzés

Az AutoCognitix platform egy ambiciózus, de technológiailag megvalósítható projekt. A siker kulcsa nem egy "mindenható" AI modell létrehozása, hanem a **szűkített, specializált ágensek** (Tudásgráf-alapú keresés, hangfelismerés, OBD protokoll kezelés) intelligens orkesztrációja. A rendszer legnagyobb értéke, hogy képes kontextusba helyezni a szerelő szubjektív tapasztalatait az objektív mérési adatokkal, így csökkentve a diagnosztikai időt és növelve a javítási pontosságot.

A megvalósításhoz egy modern technológiai stack (Flutter, Python/FastAPI, Neo4j, LangGraph) és egy fokozatos bevezetési stratégia (először csak OBD, majd tünet alapú, végül hang alapú diagnosztika) javasolt.

Works cited

1. A RAG-Based Automotive Sector AI Assistant for Enhanced Information Retrieval - DergiPark, accessed on February 1, 2026,

https://dergipark.org.tr/en/download/article-file/5001732
2. An Automotive Fault Diagnosis Framework Based on Knowledge Graphs and Large Language Models - MDPI, accessed on February 1, 2026, https://www.mdpi.com/2079-9292/14/21/4180
3. ProDemand®: Automotive Repair Software & Solutions, accessed on February 1, 2026, https://mitchell1.com/prodemand/
4. HaynesPro® WorkshopData™, accessed on February 1, 2026, https://www.atal.cz/storage/207/workshop-data-prospektus-en.pdf
5. Carly Car Scanner - The Truth You NEED To See - YouTube, accessed on February 1, 2026, https://www.youtube.com/watch?v=_Ktui3jo4u8
6. BlueDriver | Bluetooth Automotive Diagnostic Tool, accessed on February 1, 2026, https://us.bluedriver.com/
7. Vehicle API - NHTSA vPIC - Department of Transportation, accessed on February 1, 2026, https://vpic.nhtsa.dot.gov/api/
8. UX for the Industrial Environment, Part 2 - UXmatters, accessed on February 1, 2026, https://www.uxmatters.com/mt/archives/2017/09/ux-for-the-industrial-environment-part-2.php
9. Who is responsible when AI causes harm? AI and product liability | Insights - Torys LLP, accessed on February 1, 2026, https://www.torys.com/our-latest-thinking/resources/forging-your-ai-path/ai-and-product-liability
10. Optimizing RAG Techniques for Automotive Industry PDF Chatbots: A Case Study with Locally Deployed Ollama Models - arXiv, accessed on February 1, 2026, https://arxiv.org/pdf/2408.05933
11. Empowering Automotive Software Development with LLM-RAG Integration - Gupea, accessed on February 1, 2026, https://gupea.ub.gu.se/bitstreams/b760a365-3b62-498c-ba06-4dbdd5cd12a3/download
12. What is GraphRAG? - IBM, accessed on February 1, 2026, https://www.ibm.com/think/topics/graphrag
13. HybridRAG and Why Combine Vector Embeddings with Knowledge Graphs for RAG?, accessed on February 1, 2026, https://memgraph.com/blog/why-hybridrag
14. Problem Solved? Information Extraction Design Space for Layout-Rich Documents using LLMs - arXiv, accessed on February 1, 2026, https://arxiv.org/html/2502.18179v1
15. DocLLM: Revolutionizing Multimodal Document Understanding with Layout-Aware Language Models - Dataception, accessed on February 1, 2026, https://www.dataception.com/blog/docllm-revolutionizing-multimodal-document-understanding-with-layout-aware-language-models.html
16. VSSo: Vehicle Signal Specification Ontology Primer - W3C on GitHub, accessed on February 1, 2026, https://w3c.github.io/vsso/spec/vsso-primer.html
17. (PDF) VSSo: A Vehicle Signal and Attribute Ontology - ResearchGate, accessed on February 1, 2026, https://www.researchgate.net/publication/328631637_VSSo_A_Vehicle_Signal_and

_Attribute_Ontology
18. MOTOR Developer HUB, accessed on February 1, 2026, https://www.motor.com/developer-hub/
19. begaz/OBDII: Now everyone can connect safety to elm327 hugs with Begaz - GitHub, accessed on February 1, 2026, https://github.com/begaz/OBDII
20. Best Practices for HMI Design in Industrial and Safety-Critical Applications, accessed on February 1, 2026, https://www.mouser.com/blog/best-practices-hmi-design-industrial-safety-critical-applications
21. Automation in Cars: Voice Controlled Car Assistant System and Automatic Breaking System – A Review | Request PDF - ResearchGate, accessed on February 1, 2026, https://www.researchgate.net/publication/360405517_Automation_in_Cars_Voice_Controlled_Car_Assistant_System_and_Automatic_Breaking_System_-_A_Review
22. The need and potential for AI-enabled voice assistants in vehicles - Chalmers ODR, accessed on February 1, 2026, https://odr.chalmers.se/server/api/core/bitstreams/4c608c10-c073-4912-bb6b-3b4910c4cad3/content
23. OtoMobile Dataset - Zenodo, accessed on February 1, 2026, https://zenodo.org/records/3382945
24. OTOMECHANIC: AUDITORY AUTOMOBILE DIAGNOSTICS VIA QUERY-BY-EXAMPLE Max Morrison Northwestern University Computer Science Deptartm - Interactive Audio Lab, accessed on February 1, 2026, https://interactiveaudiolab.github.io/assets/papers/mm_bp_dcase_2019_cr.pdf
25. J2012_201612 : Diagnostic Trouble Code Definitions - SAE International, accessed on February 1, 2026, https://www.sae.org/standards/j2012_201612-diagnostic-trouble-code-definitions
26. AI Generated Content Disclaimer | Tony Menendez, DDS, MAGD, accessed on February 1, 2026, https://www.drtonymenendez.com/ai-generated-content-disclaimer/
27. The Basics of LangGraph: A Step-by-Step Guide to AI Workflows | by Susmit Panda, accessed on February 1, 2026, https://medium.com/@susmit.vssut/the-basics-of-langgraph-a-step-by-step-guide-to-ai-workflows-478852840f5d
28. Build a LangGraph Multi-Agent system in 20 Minutes with LaunchDarkly AI Configs, accessed on February 1, 2026, https://launchdarkly.com/docs/tutorials/agents-langgraph
29. MobileAIBench: Benchmarking LLMs and LMMs for On-Device Use Cases - arXiv, accessed on February 1, 2026, https://arxiv.org/html/2406.10290v1
30. Top 6 Local AI Models for Maximum Privacy and Offline Capabilities - Software Mansion, accessed on February 1, 2026, https://blog.swmansion.com/top-6-local-ai-models-for-maximum-privacy-and-offline-capabilities-888160243a94