



**BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM**  
**Gazdaság- és Társadalomtudományi Kar**  
Menedzsment és Vállalkozásgazdaságtan Tanszék

# **Mesterséges intelligencia-alapú automatizálás a menedzseri feladatdelegációban és tevékenységkövetésben**

Készítette: Kozma Szabolcs András  
mérnökinformatikus szakos hallgató

Konzulens: Dr. Benedek Petra  
egyetemi adjunktus, GTK, MVT  
Konzulens: Dr. Szűcs Gábor  
egyetemi docens, VIK, TMIT  
Konzulens: Ficzere Dániel  
tanársegéd, VIK, TMIT

**Tudományos Diákköri Konferencia  
Budapest, 2025**

# TARTALOMJEGYZÉK

<b>1. BEVEZETÉS.....</b>	<b>1</b>
1.1. CÉLOK .....	1
1.2. ÁTTEKINTÉS.....	2
1.2.1. A munkahelyi folyamatok automatizálásának jelenlegi helyzete .....	2
1.2.2. Automatizáló platformok .....	3
1.2.3. Az n8n platform .....	4
1.2.4. Nagy nyelvi modellek jelenlegi helyzete .....	5
<b>2. TERVEZÉS, MEGVALÓSÍTÁS MÓDJAI.....</b>	<b>8</b>
2.1. FELHASZNÁLT ARCHITEKTÚRÁLIS ELEMEK.....	8
2.2. ELSŐ MEGKÖZELÍTÉS .....	10
2.3. MÁSODIK MEGKÖZELÍTÉS .....	11
2.4. HASZNÁLT LLM-EK.....	11
2.5. ERROR WORKFLOW .....	13
2.6. KIMUTATÁSOK.....	14
2.7. MEGJELENÍTÉS.....	14
<b>3. SKÁLÁZHATÓSÁG, BŐVÍTHETŐSÉG .....</b>	<b>17</b>
3.1. SKÁLÁZHATÓSÁG .....	17
3.2. BŐVÍTHETŐSÉG .....	17
<b>4. IMPLEMENTÁCIÓ LEÍRÁSA .....</b>	<b>19</b>
4.1. GMAIL KÖRNYEZET KIALAKÍTÁSA .....	19
4.2. ARCHITEKTÚRA MEGVALÓSÍTÁSA.....	20
4.2.1. Chat Input Node.....	21
4.2.2. AI Agent Node.....	21
4.2.3. LLM Node .....	22
4.2.4. Simple Memory Node.....	23
4.2.5. Google Calendar Node .....	23
4.2.6. Google Sheet Node .....	25
4.3. ERROR WORKFLOW .....	27
4.4. FELÜLETEK ELKÉSZÍTÉSE.....	27
4.4.1. Felület típusok .....	28
4.4.2. Azonosítás.....	29
4.4.3. Használt eszközök, nyelvek .....	31
<b>5. TESZTELÉS .....</b>	<b>32</b>
5.1. TESZTELÉS MÓDJA .....	32
5.2. TESZTELÉSI METRIKÁK .....	32
5.3. TESZT ESETEK .....	32

5.4. EREDMÉNYEK .....	33
<b>6. KONKLÚZIÓ .....</b>	<b>36</b>

## **1. BEVEZETÉS**

A 21. században számos technológia született, amelyek jelentős javulást eredményeztek a munkafolyamatok hatékonysága terén. Az automatizálás új korszakába léptünk, ahol a gépek már nem csupán előre meghatározott utasításokat hajtanak végre, hanem képesek tanulni, döntéseket hozni és önállóan reagálni a környezetük változásaira. Ennek mozgatórugója a mesterséges intelligencia, ami egyértelműen a legmeghatározóbb technológiai újítás napjainkban. A 2020-as évekre érte el a “generatív” szintet, vagyis önálló szövegek generálására is képessé váltak. 2022 végén- 2023 elején jelent meg a “tool use” koncepció, ami a külső eszközök hívására fókusztált, mint például kalkulátor, adatbázisok, keresőmotorok és sok már API-n (Application Programming Interface) keresztül elérhető szolgáltatás. 2024-ben jelent meg az ügynök szintű automatizálás, ahol nagy nyelvi modellek (Large Language Model, LLM) komplex eszköz láncokat képesek vezérelni, ezáltal személyre szabott feladatokat ellátni.

Napjainkban ez a technológia jelentősen átformálja az ipar szinte minden területét és az alkalmazása jelentős versenyelőnyt jelenthet azok számára, akik kiaknázzák a benne rejlő lehetőségeket. Az olyan menedzseri feladatok, mint a feladat delegálás és a tevékenység követés, nagy potenciállal rendelkeznek ebből a szempontból, számos tevékenység váltható ki automatizált mesterséges intelligencia (Artificial Intelligence, AI) megoldásokkal és ezáltal jelentős mennyiséggű idő szabadítható fel mind a vezetők, mind az alkalmazottak számára, amit a monoton feladatok végzése helyett a stratégiai döntéshozatalra, illetve a precízebb munkavégzésre fordíthatnak, ezáltal a létrehozott végtermék vagy szolgáltatás minősége is javulhat.

### **1.1. Célok**

A dolgozat célja megvizsgálni és értékelni a szervezeten belüli menedzsment tevékenységek (feladatdelegálás, értekezlet összehívás, tevékenységkövetés) AI megoldásokkal való automatizálhatóságát. Egy prototípus készítésén keresztül bemutatásra kerülnek a különböző építőelemek, amik a rendszert alkotják, valamint egy ilyen mesterséges intelligenciát használó alkalmazás felépítése. A most következő években, évtizedekben az egyik, ha nem a legmeghatározóbb piac, a mesterséges intelligencia különböző területeken való alkalmazása lesz. A technológia fejlődése elérte azt a pontot, hogy már könnyen elérhető és integrálható számos AI szolgáltatás, amiket alkalmazva rendkívül erős eszközöket lehet létrehozni. Ilyen eszköz a projekt során fejlesztett rendszer is. Az alkalmazás gyorsabbá és hatékonyabbá hivatott tenni a feladatok delegálását, értekezletek összehívását, illetve az

információ lekérdezést. A tesztelés és költség-haszon elemzés célja meghatározni az elkezelésben rejlő piaci potenciált mind hatékonyság, mind megtérülés szempontjából.

## 1.2. Áttekintés

Az említett cél eléréséhez szükséges áttekinteni a terület jelenlegi helyzetét, az ott használt megoldásokat, az egyes automatizáló platformokat, azon belül részletesen a projekt során használt platformot, illetve a nagy nyelvi modellek jelenlegi helyzetét és történetét.

### 1.2.1. A munkahelyi folyamatok automatizálásának jelenlegi helyzete

A szervezetek manapság egyre inkább nyomás alatt állnak. A technológia egyre gyorsuló fejlődésével folyamatosan nyomon kell követniük a trendeket, hogy képesek legyenek tartani a lépést versenytársakkal hatékonyság, gyorsaság és költségek terén. Emiatt az automatizált folyamatok száma és az automatizálás szerepe egyre növekszik, mivel a jól strukturált automatizálás számos területen jelentősen csökkentheti az emberi beavatkozás szükségességét és a költségeket, valamint kiküszöbölni az emberi hibafaktort.

Napjainkra a vállalatok, illetve vállalkozások több mint kétharmada old meg legalább egy üzleti folyamatot automatizált megoldással. (Bernzweig, 2024) Ezek nagyrészt egyszerű, szabályalapú feladatok, mint adatrögzítés vagy számlák kezelése, azonban egyre több kutatás téma olyan automatizáció, ahol az ember és a gép kooperációja van a fókuszban.

HR területen is nagyszerű eredményeket értek már el, ami bátorítja a vállalkozásokat hasonló megoldások alkalmazására. Megfelelő körülmények között, akár nyolc-tízszeres gyorsulás és 100%-os hibamentesség érhető el a hagyományos módszerekhez képest. (Mohamed et al., 2022) Ezáltal jelentős mennyiségű emberi erőforrás (idő és energia) szabadul fel, amit az ismétlődő munka helyett kreatívabb feladatokra, kivételkezelésre, illetve azok pontosabb végrehajtására tudnak fordítani. Ezáltal pedig jelentős hatékonyság növekedés érhető el.

Az említett "megfelelő körülmények" azonban elengedhetetlenek. Ha a folyamat nem jól dokumentált, tesztelt, akkor könnyen kialakulhat hibás gyakorlat. Rendkívül nagy odafigyeléssel szabad csak bevezetni automatizált folyamatokat, egy-egy elhamarkodott döntés nagy veszteséggel járhat.

A téma technikai oldalán túl emberi, illetve kulturális korlátokkal is számolni kell. Az adaptáció sikere nagymértékben függ a munkatársak nyitottságától, szabálykövetésétől és technológiai kompetenciájától. Az utóbbi tekintetben nem megfelelő szinten lévő alkalmazottakban kételkedést, illetve szorongást kelthet az új technológiák bevezetése, ami a

munkamorál romlását eredményezheti. Ebből kifolyólag azon szervezetek alkalmazzák könnyebben az AI, illetve automatizációs megoldásokat, ahol a munkavállalók rendelkeznek a megfelelő alap tudással, vagy nyitottak a tanulásra. (Ekuma, 2024) A menedzsment példamutatása, támogatása és szervezeti kultúra formáló képessége szintén fontos tényező a sikeres bevezetésben.

Megfelelő körülmények között az automatizáció viszont nagyban javíthatja az alkalmazotti elégedettséget. A hatékony és igazságos HR folyamatoknak óriási szerepe van a dolgozók lojalitásában, illetve motivációjában, amelyek kulcs faktorok, ha a szervezet hatékonyságának növeléséről beszélünk. (Firawi, 2024)

### **1.2.2. Automatizáló platformok**

Számos módja van az automatizálásnak, mint például az üzleti folyamatok automatizálása (Business Process Automation, BPA), a robotizált folyamatautomatizálás (Robotic Process Automation, RPA), az intelligens folyamatautomatizálás (Intelligent Process Automation, IPA), IT- és infrastruktúra, vagy a fizikai és ipari automatizálás. Természetesen minden területen más módjai és eszközei vannak a gépesítésnek. A dolgozatban tárgyalta projekt a BPA alcsoporthoz, vagyis a folyamat alapú automatizálás alá tartozik.

Számos szoftver/ platform felmerülhet típusról, szervezeti mérettől, adatvédelemről és személyre szabhatóságtól függően. Első szűrési szempont a szervezeti méret volt. Mivel kisvállalkozások számára használható prototípus fejlesztése volt a cél, a külön erre a célra készített eszközöket kutattam fel. Szóba jött az n8n, Zapier, Pipefy és a Make.com. (Hiter, 2024) (Shakudo, 2025)

Az 1. táblázatban bemutatom a vizsgált platformokat, a legfontosabb szempontok szerint: adatvédelem, személyre szabhatóság, AI integráció.

1. táblázat: Workflow automatizáló platformok összehasonlítása  
Forrás: az egyes platformok dokumentációja és felhasználási javaslata

	<b>n8n</b>	<b>Zapier</b>	<b>Pipefy</b>	<b>Make.com</b>
Adatvédelem	kiemelkedő (self hosting lehetőség)	közepes (felhő alapú, nincs self hosting)	jó (felhő alapú, ISO 27001 tanúsítvány, nincs self hosting)	jó (felhő alapú, GDPR kompatibilis, nincs self hosting)
Személyre szabhatóság	nagyon magas	korlátozott (prémium funkcióval)	közepes	magas
AI integrációk	erős	közepes	fejlődő	erős

Egyértelműen az elsődleges szempont az adatvédelem volt, mivel ez befolyásolja a valós felhasználhatóságot. Az n8n self hosting lehetősége a legbiztonságosabb, mivel így az adatok nem mennek ki idegen szerverekre és helyben lehet tárolni. Ez a platform open-source (nyílt forráskódú) voltából következik, ami magával hozza azt a nem mellékes előnyt is, hogy teljesen ingyenesen elérhető benne minden funkció.

Mivel a projekt első lépése a platform kiválasztása volt, fontos volt az egyes elemek személyre szabhatósága, hogy a lehető legjobban lehessen megoldani a problémákat, minél kevésbé kelljen kompromisszumokat kötni. Ezen a téren az n8n platform tünt ki, mivel lehetőséget biztosít egyedi node-ok (gráf pont) és webhook-ok (esemény vezérelt értesítés) definiálására.

Nem utolsó sorban fontos volt még az AI eszközök integrációja. Sok automatizációs platform rengeteg integrációval rendelkezik, azonban kevesebb még az olyan, ahol hatékonyan és egyszerűen lehet LLM modelleket felhasználni, viszont ezen a területen évről évre sokat fejlődnek az egyes szoftverek. Itt az n8n és a Make.com fej-fej mellett teljesítenek, azonban figyelembe véve a többi szempontot nem volt kérdés, hogy az n8n a legalkalmasabb az alkalmazás megvalósítására.

### 1.2.3. Az n8n platform

Miután ismertettem a platform előnyeit, fontos annak architekturális építőelemeit is bemutatni, amelyekből összeáll egy munkafolyamat. Három fő elem típus különíthető el: Node (gráf pont), Connection (kapcsolat) és Sticky Note (ragadós cetli).

A legfontosabbak a Node-ok. Ezek jelképezik az egyes állomásokat a munkafolyamatban. Ezek lehetnek AI agentek (magyarul mesterséges intelligencia ügynök, egy olyan speciális AI megoldás, ami egy bizonyos feladatra specializáltan képes önállóan feladatokat végrehajtani), csatlakoztatott API-n keresztül elérhető szolgáltatások vagy logikai elemek, mint például fork, if, stb. Node-on keresztül lehet elindítani egy munkafolyamatot (ez a trigger node), adatot gyűjteni és tovább küldeni, vagy adatot feldolgozni és módosítani. Számos beépített, előre elkészített node-ot használhatunk, valamint létrehozhatunk saját, személyre szabott elemet is. (n8n Docs: Nodes, n.d.)

A node-okat a connection-ök kötik össze. Egy node bemenetére be lehet kötni több különböző másik node-ot is, ahogy egy gráf pont kimenetét is be lehet kötni számos másik elembe. A kapcsolatok jelölik az adat áramlását és annak irányát, az összekötött node-ok között van adatmegosztás. (n8n Docs: Connections, n.d.) A gráfelemek és kapcsolatok létrehozása az 1. ábrán látható.



1. ábra: Connection létrehozása

*Forrás: n8n dokumentáció*

A sticky note-ok segítenek annotációkkal és kommentekkel ellátni a munkafolyamatot. Erősen javasolt ezeknek a használata az átláthatóság és az érthetőség érdekében. A cetlik szövegét ízlés szerint formázhatjuk markdown (könnyen használható dokumentum annotációs nyelv) formátumban, illetve különböző színeket állíthatunk be nekik. (n8n Docs: Sticky Notes, n.d.)

#### 1.2.4. Nagy nyelvi modellek jelenlegi helyzete

Az alkalmazás kulcs gondolata a menedzseri feladat delegáció mesterséges intelligenciával való megtámogatása. Ez LLM, vagyis nagy nyelvi modellek segítségével valósítható meg. Ez egy nagyon gyorsan és dinamikusan fejlődő terület, évről évre jönnek ki újítások, amelyek teljesen felülírják az éppen aktuális helyzetet. A Fortune Business Insights már 2024-ben 233 milliárd dollárra becsülte az AI piac méretét, és szerintük 2032-re elérheti az 1771 milliárd dolláros értéket is. (Fortune Business Insights, 2025)

A mai legjobban teljesítő LLM-ek elvi alapjait egy 2017-es cikkben publikálták. A Google Research publikációjában megjelent “Attention is all you need” című műben fektetik le a Transformer-architektúra elméleti alapjait, amire később nagyban építettek. (Vaswani et al., 2017)

2018-ban bemutatták az első valódi LLM-et, az OpenAI GPT-1-es modelljét, ami 117 millió paraméterrel rendelkezett. 2020-ban szabályosan betörtek a köztudatba, ekkor a legnépszerűbb modell, az OpenAI GPT-3-ma több mint 100 milliárd paraméterrel rendelkezett. Ekkor a modellek általános nyelvi képességekkel rendelkeztek, alkalmasak voltak szövegalkotásra, azonban külső eszközök használatára még nem voltak képesek. Ezt a szintet körülbelül 2022-re érték el, ekkortól nem csak kutatási, hanem ipari szinten is sokrétűen használhatóvá váltak. (Boris, 2024)

Az LLM-ek fejlődését gyakran a modellek paraméterszámával, illetve a tanításhoz felhasznált adat mennyiségével szokták jellemezni. A paraméterszám a neurális hálóban megjelenő súlyok (weights) és torzítások (biases) számának összege. (Goodfellow et al., 2016) Ez a szám az elmúlt években rohamosan megnőtt, azonban ez a mutató magában nem jelenti a

hatékonyság növekedést is, viszont a piac jellemzésére alkalmas. Egyes portálok szerint az Alibaba által fejlesztett Qwen-3-Max modell több mint egy billió, vagyis ezer milliárd, paraméterrel rendelkezik. Ez jól mutatja milyen ütemben fejlődnek a nagy nyelvi modellek, azonban ez a folyamatos növelhetőség, ami a paraméterszámra igaz, nem mondható el a modell betanításához használt adatmennyiségről. Az optimális tanításhoz ezt a két értéket összhangban kell növelni, azonban ez egyes kutatások szerint nem mindenkor lesz lehetséges. (Hoffmann et al., 2022) Az LLM-ek tanításához felhasznált írásos adatok mennyisége 2026 és 2032 között elérheti a 100%-ot, vagyis minden rendelkezésre álló adatot felhasználhatnak, ami fontos kihívás elé állítja a gyártókat. (O'Brien, 2024) A tanításhoz felhasznált adatmennyiség növekedő tendenciáját a 2. ábra jól szemlélteti.



2. ábra: Modellekhez felhasznált adatmennyiség növekedése  
Forrás: Epoch AI, n.d.

A nagy nyelvi modellek alkalmazási területe nagyon széles. Leggyakrabban az ügyfélszolgálatban, tartalomgyártásban, programozásban, egészségügyben alkalmazzák, de egyre szívesebben használják az oktatásban is és az üzleti döntéshozatalban is. Mára képesek a természetes nyelvi megértésre és generálásra, a multimodalitásra (vagyis szövegen kívül képet, kódöt és hangot is képesek feldolgozni és generálni) valamint a külső eszközhasználatra, mint bongészés, komplex számítások elvégzése, adatbázis lekérdezések vagy API hívások (amikor egy kliens-alkalmazás üzenetet küld egy másik szoftveralkalmazásnak vagy szolgáltatásnak azzal a céllal, hogy adatot lekérjen, adatot küldjön, vagy műveletet végezzen) (GeeksforGeeks,

2025). Ezen utóbbi képességek teszik lehetővé az úgynövezett ügynöki működést, vagyis az autonóm feladatvégzést több lépésben. (Gemini Team, 2024) (OpenAI, 2025)

## **2. TERVEZÉS, MEGVALÓSÍTÁS MÓDJAI**

A fejezet célja egy absztrakciós szinttel tovább menni a témaban. A szükséges háttérírimeretek és célok tárgyalása után a megvalósítás egyes részeit, lépéseit ismertetem. Ebben a fejezetben szó van az egyes architekturális elemekről, amelyeket az implementáció során felhasználtam, a fő megközelítésekről, illetve, hogy mi okozta az azok közötti váltást. Továbbá a fejlesztés során felhasznált LLM modellekéről, a felhasznált memória típusról, az esetleges használat közben fellépő hibák elhárításának módjáról, valamint arról, hogy milyen hasznos kimutatásokat szolgáltat az alkalmazás az azt használó vezetőnek. Ezen utolsó elemhez kapcsolódóan elengedhetetlen megemlíteni a megjelenítési módokat, az egyes különböző nézeteket, amelyek segítségével minden felhasználó számára egyszerű és praktikus az alkalmazás használata. Jelen fejezetben a kontextust kibővíve az elképzelés egyértelmű leírásán van a hangsúly, magának az implementációnak a leírása egy későbbi fejezet témája.

### **2.1. Felhasznált architekturális elemek**

A korábban az n8n platform bemutatásánál nevesített architekturális elemek 6 csoportba sorolhatóak a struktúrában betöltött szerepük szerint: bemeneti elem, AI ügynök, LLM-ek, Google Calendar elemek, Google Sheet elemek és memóriáért felelős elemek. Ez a szakasz az adat áramlásának folyamata szerint ismerteti az egyes architekturális elemeket, az egyes csoportokon haladva végig.

A felhasználóval való interakcióért a bemeneti elem a felelős. Ez az úgynevezett „Chat Input Node”, vagyis a beszélgetés bemeneteként szolgáló gráfpont. Ebben a pontban tudja a felhasználó megadni a természetes nyelvű kívánságait, instrukcióit az alkalmazás felé, valamint az alkalmazáslogika (az AI ügynök) ezen az elemen keresztül tud válaszolni a felhasználónak. A válasz lehet üdvözölés, információ közlés, információ bekérése, vagy egyéb közlés az elvégzett/ elvégzendő feladatokkal kapcsolatban. A felhasználó mindenkor minden elemet látja a működésből. Természetes nyelvű üzenetekre természetes nyelvű válaszokat kap, ezzel biztosított az egyszerű használat mindenki számára.

A folyamat következő eleme az AI ügynök. Ez az elem megkapja teljes egészében a felhasználó által megadott üzenetet, amelyet a Chat Input Node továbbít felé. Ennek az elemnek a feladata összefogni sok másik elemet és információt, amelyek felhasználásával képes lesz értelmezni a kapott üzeneteket és arra válaszolni, valamint végrehajtani a kívánt feladatokat. A jelenlegi megvalósításban az AI ügynök kapcsolatban áll minden más architekturális elemmel. Amint azt a bevezetőben említettem, ezek között vannak elem típusok, amelyekkel

mindenképpen kapcsolatban kell állnia. Ilyen a Chat Model elem, ami a felhasznált LLM modellt tartalmazza, illetve egy Memory elem, ami pedig a felhasznált memóriát. Ezeket túl az AI ügynök kapcsolatban áll még a Google Calendar elemek, illetve a Google Sheet elemek csoportjával. Ezekben a csoportokban öt, illetve három különböző elem található, ezeket hamarosan bővebben ismertetem.

Miután az ügynök megkapta a beérkező üzenetet, aktiválja a hozzá csatlakoztatott Chat Model elemet. Jelen esetben ez az OpenAI Chat Model elem. Ebben az elemben vannak eltárolva a hitelesítő adatok, amelyek segítségével azonosítani tudjuk magunkat a szolgáltatónál, illetve a használni kívánt modell neve. A hitelesítést követően elérhetővé válik az AI ügynök számára a megnevezett modell, és annak segítségével feldolgozhatja a beérkezett üzenetet. A hitelesítés módjáról és a felhasznált modellről az Implementáció leírása fejezetben további részletek olvashatók.

A Chat Model elem mellett, a másik kötelezően az AI ügynökhöz csatlakoztatott elem a Memory elem. Ez felelős az ügynök memoriájával kapcsolatos beállítások, illetve adatok tárolásáért. Ez az elem elengedhetetlen a jó felhasználói élményért. Lehetővé teszi, hogy az ügynök úgymond vissza tudjon emlékezni a korábbi üzenetekre. Így el tud emelkedni az interakció az egyszerű parancsriadás szintjéről a beszélgetés szintje felé. Természetesen nem optimális, ha minden korábbi üzenetre emlékszik az ügynök, mivel az esetek nagy részében csak overhead (felesleges ismétlés) keletkezik. (Minden ügynök esetében külön meg kell határozni a memoriában tárolt korábbi üzenetek számát, amelyek alkalmazási területektől függően változhat. Amennyiben csak arra kell a memória, hogy „redo” műveletet lehessen végezni, elegendő akár 2-3 üzenet tárolása, azonban, ha élethű beszélgetés imitálása a cél ez a szám elérheti az 50-100-as nagyságrendet is.) Amint korábban említettem, akár 50-100 üzenetre is emlékezhet az ügynök, de az alkalmazás nem igényel ekkora memóriát, így ezt az értéket 10-re állítottam. Ez elegendő ahhoz, hogy referálni lehessen az AI ügynöknek korábbi adatokra, illetve problémamentes legyen az interakció akkor is, ha egy feladat kiadása több üzeneten keresztül terjed, viszont nem keletkezik felesleges ismétlés.

A további csatlakoztatott elemek a tool-ok (olyan funkció, melyet az agent használhat a környezetével való interakcióra vagy feladatok végrehajtására). Ahogyan a neve is jelzi, ezeket eszközként tudja használni az agent, hogy olyan feladatokat is el tudjon látni, amelyekre egyébként nem lenne képes. Többnyire API-n keresztül csatlakoztatott alkalmazások egyes funkcióiról van szó. Ebben az esetben is ez a helyzet: a Google Calendar illetve a Google Sheets különböző funkciót csatlakoztattam, hogy az agent képes legyen létrehozni és észlelni

értekezleteket, eseményeket a csatlakoztatott Google Calendarban, valamint táblázatokat olvasni és kitölteni a hozzá társított Google Sheets dokumentumokban. A Calendar-nak négy (esemény felvétele, törlése, módosítása, olvasása), a Sheets-nek három funkcióját (sorok olvasása, sor felvétele, sor módosítása) vettem fel, mint elérhető eszközök. Ezek öt, illetve három tool-t tesznek ki, mivel az esemény felvételét kétszer kellett felvenni, hogy külön lehessen használni meeting, illetve új feladat felvételekor. Az alkalmazás fő tevékenységét ezekkel az eszközökkel végzi az ügynök, így annak a gerincét képezik.

## 2.2. Első megközelítés

A tervezés során folyamatosan cserélődtek a használt elemek, mivel az egyes felmerülő problémák új megközelítést igényeltek. A sok kisebb változtatás mellett két nagyobb eltérő megközelítés különíthető el. A tervezési folyamat kezdetén csak Google Sheets tool-ok voltak csatlakoztatva az AI agent-hez és ezek látták el a kiosztott feladatok és összehívott meetingek adatainak tárolását is. Az elképzelenben három különböző munkalapon tároltam az adatokat. Egy munkalap felelt a feladatok és meetingek időpontjainak tárolásáért a következő formában: Az első oszlopban az egyes cellák a munkaidő fél órás egységeit jelölték, a további oszlopok pedig sorrendben a munkanapot. Egymás alatt minden dolgozó számára létrehoztam egy ilyen táblázatot, ahol az elfoglaltságaiat tárolni lehet. Egy feladat kiosztásakor vagy meeting összehívásakor az agent a tevékenység időintervallumának megfelelő cellákat módosította: beleírta a tevékenység nevét és beszínezte pirosra. Az elérhetőség vizsgálatakor is a kitöltött cellákat figyelte és ez alapján adta vissza az egyes alkalmazottakra vonatkozó leterheltségi adatokat. Egy másik munkalap tárolta az agent által végrehajtott feladatok adatait: feladatot vagy meetinget vett fel/ törölt, mikor stb. A harmadik munkalap pedig az alkalmazottak és a vezető adatait tárolta: email cím és teljes név. Azonban ezzel a megoldással kapcsolatban több olyan probléma is fellépett, ami indokolttá tette a váltást. Egy tevékenység (feladat vagy meeting) bevitele túl sok műveletből állt, mivel egy 4 órás feladat felviteléhez 8 sort kellett módosítani, illetve az alkalmazottak elérhetőségéhez az összes sort be kellene olvasnia, ami közel sem volt hatékony. Rengeteg tokent (a szövegegenség, amit az LLM használ) igényelt az adatok ilyen tárolása és olvasása, ami kritikus hiányosságnak tűnt. Ezen túl a törléssel is probléma volt. Egy feladat vagy meeting törléséhez ugyanis valójában módosítani kellett a cellákat, azonban a megoldás tesztelése során rendre ragadtak be módosítatlan cellák, így olyan adat maradt az adatbázisban, ami inkonzisztenssé tette. Mindezek következményeként jutottam arra a következtetésre, hogy más megközelítést kell alkalmazni a megfelelő működéshez.

### **2.3. Második megközelítés**

A korábbi megközelítés annyiban módosult, hogy a delegált feladatokat, illetve összehívott értekezleteket nem táblázatos adatbázisban tároltam, hanem a Google Calendarban. Ez kézenfekvő volt, mivel a Google Calendart éppen ilyen célra fejlesztették ki és ezáltal sokkal egyszerűbbé és hatékonyabbá vált a folyamat. Ebben a megközelítésben több külön naptárt, illetve Google Sheet-et kezel a rendszer. minden embernek a szervezetben külön naptára van, illetve van egy közös csapat naptár is, ahol az olyan meetingeket lehet elérni, ahol mindenki részt vesz, vagy fontos mindenki számára tudni róla. A Google Sheets táblázatokban pedig a korábbi megoldáshoz hasonlóan a tranzakciós és alkalmazotti adatok tárolódnak. A Google Calendar tehát az korábbiakban említett első munkalapot helyettesíti. A beépített “Create event”, “Edit event” és “Get events” metódusok révén megvalósítható mind az új esemény felvétele, az esemény módosítása, valamint az események lekérése adott időszakban. Ezen túl egy új esemény felvételekor a résztvevők értesítést kapnak automatikusan.

### **2.4. Használt LLM-ek**

A bevezető már áttekintette az LLM-ek, vagyis nagy nyelvi modellek fejlődését és jelenlegi helyzetét, most pedig ezen a területen is mélyebbre ás a dolgozat. Számos szolgáltató modelljét megvizsgáltam, amikor kerestem a megfelelő modellt a projekt számára.

Összetett szempontrendszer szerint vizsgáltam az egyes modelleket: lokális vagy API-n keresztül elérhető, árazás, erősség, sebesség, bemeneti kontextus mérete, illetve az ajánlott felhasználási terület. Vizsgáltam a Gemini (Gemini 2.5 család), OpenAI (GPT 4.1 és 4o család), Meta (Llama 2 modellek) és az Anthropic (Claude 4.5 Haiku) szolgáltatókat. (Google Cloud Vertex AI, n.d.) (OpenAI, 2024) (Anthropic, n.d.)

Az első szempont, ami alapján döntöttem, a lokalitás volt. A legbiztonságosabb és legolcsóbb egy lokális, open source modell használata, mint a Llama2, azonban ehhez szükséges egy megbízható és erős szerver, hogy olyan hatékonyságot érjünk el, mint amit a legtöbb nem nyílt forrású modell kínál. Mivel ez nem állt rendelkezésemre, az API-n keresztül elérhető modellek mellett kellett döntenem. Itt áttekintettem és elemeztem a különböző Gemini, OpenAI és Anthropic modelleket.

Mivel az egyes szolgáltatók modelljeinek használatához egyenleget kell feltölteni a szolgáltató oldalán, amiből a használati költségeket folyamatosan vonják le és igyekeztem minél kevesebb pénzt elkölteni API használati díjakra, választanom kellett egyet az előbb felsorolt szolgáltatók közül. A választásom az OpenAI-ra esett, mivel ez kínálta a legnagyobb választékot a projekt szempontjából releváns modellek terén.

Négy modell volt, ami rövid kutatás után igéretesnek bizonyult, így ezeket közelebbről, részletesebben is megvizsgáltam. A GPT-4o, GPT-4o mini, GPT-4.1 és GPT 4.1 mini, ahogyan a nevük is mutatja két külön családba tartozó párosok. A mini modellek és nagyobb testvéreik között elsősorban teljesítményben és árban van különbség.

A GPT-4.1 egy viszonylag új modellcsalád. Egymillió tokenes bemeneti kontextus támogatást nyújt, megbízható instrukció-követést és “tool calling” -ot biztosít, ami az alkalmazás számára szükséges is. Az árazása alakulását a 2. táblázat mutatja (egymillió tokenenként).

2. táblázat: GPT-4.1 és GPT-4.1 mini modellek árazása  
*Forrás: Az OpenAI hivatalos oldala*

	<b>Input</b>	<b>Cached Input</b>	<b>Output</b>
GPT-4.1	\$2.00	\$0.50	\$8.00
GPT-4.1 mini	\$0.40	\$0.10	\$1.60

A GPT-4o eggyel korábbi modellcsalád, így a képességei sem érik el a GPT-4.1 szintjét, de a projekt nem is feltétlen igényli azt. 128 ezer tokenes bemeneti kontextust biztosít, ami jelen esetben elegendő: Egy esemény felvétele az alkalmazásban körülbelül 10 ezer tokent igényel összesen (input, belső működés és output egyben), ha ehhez hozzávesszük a korábban említett 10 üzenetre visszaterjedő memóriát, az is csak 110 ezer token, így a kontextus szám megfelelő. Az OpenAI szerint rendkívül jó szövegfeldolgozásban és a válaszideje is elég gyors. (OpenAI, 2024) Az árazása változó az előző családhoz képest, a GPT-4o modell drágább azonban a mini verzió verhetetlen. Ennek adatait a 3. táblázat mutatja.

3. táblázat: GPT-4o és GPT-4o mini modellek árazása  
*Forrás: Az OpenAI hivatalos oldala*

	<b>Input</b>	<b>Cached Input</b>	<b>Output</b>
GPT-4o	\$2.50	\$1.25	\$10.00
GPT-4o mini	\$0.15	\$0.075	\$0.60

A modellek teljes összehasonlítását a 4. táblázat foglalja össze.

4. táblázat: GPT-4o és GPT-4o mini modellek árazása

*Forrás: Az OpenAI hivatalos oldala*

	<b>GPT-4o</b>	<b>GPT-4o mini</b>	<b>GPT-4.1</b>
Kontextus méret	~128 000	~128 000	1 000 000
Teljesítmény	erős, multimodális rendszer	egyszerűbb, de praktikus	fejlettebb instrukció követés, jobb kód, mélyebb long-context
Késleltetés	gyors, de nem optimalizált a legkisebb késleltetésre	cél a gyors válasz, kisebb modell	optimalizáltabb, alacsonyabb késleltetés
Ár / költség	magasabb	alacsonyabb ár / költséghatékony	versenyképes ár
Tudásbázis frissesség	korábbi	korábbi	újabb (pl. 2024 közepé)
Legjobb felhasználási területek	multimodális input/output (képek, hang), komplex feladatok, agentek	olcsó multimodális próba, kisebb appok	komplex agent rendszer, nagy dokumentum környezet, hosszú folyamatok

Figyelembe véve, hogy nem szükséges az egymillió tokenból álló kontextus ablak, gyors válaszadásra van szükség, fontos az alacsony ár, nem annyira fontos a betanított tudásbázis (mivel előre definiált feladatokat kell a modellnek elvégeznie) és a rendszer komplexitása sem nagy, a GPT-4o mini modell tünt a legalkalmasabbnak. Amennyiben ez nem vált volna be, a sorban a következő lett volna a GPT-4.1 mini, azonban a 4o mini is gond nélkül, pontosan végezte a feladatokat. Kísérletezési és tesztelési célra tökéletesen megfelelő, így ezzel a modellel folytattam a munkát.

## 2.5. Error Workflow

Mint bármilyen folyamatban, itt is léphetnek fel nem várt hibák, amelyek működésképtelenné teszik a rendszert. Ilyen esetekre szokás az alap workflow, vagyis munkafolyamat mellé egy úgynévezett error workflow-t is definiálni, aminek a feladata kezelni valamilyen módon a problémát. Ezt a workflow-t össze lehet kapcsolni a fő munkafolyamattal és amikor az nem várt működést produkál, automatikusan lefut. Sok esetben az error workflow szintén módosításokat végez a használt adatbázisban, egyből orvosolva a hibát. Tartalmazhat ez is AI agenteket, amelyek meghatározzák a hibát és javítják azt. Azonban, mivel az volt a

tapasztalat a projekt során, hogy a legtöbb nem várt hiba jogosultság vagy azonosítási probléma volt, így ez a workflow nem módosít semmit, csupán riasztást küld. Ezek a jogosultsági, illetve azonosítási gondok a használt API-knál léphetnek fel. Lehetsz a chat model, vagy akár Google Calendar, Google Sheet API. Ilyen esetben nélkülözhetetlen a manuális beavatkozás, mint új API kulcs generálása és betöltése, vagy az egyenleg feltöltése. Az error workflow a következő elemekből áll: a hibaüzenet, egy AI agent, és egy csatlakoztatott Gmail tool. A hibaüzenetet az AI agent feldolgozza és jól strukturált módon beépíti a küldendő e-mailbe. Ezt aztán a Gmail tool segítségével elküldi azokra az e-mail címekre, amelyeket megadunk neki. Jelen projektben ez csupán az én személyes e-mail címem, azonban éles környezetben lehet ez bármilyen üzemeltető vagy döntésképes személy.

## 2.6. Kimutatások

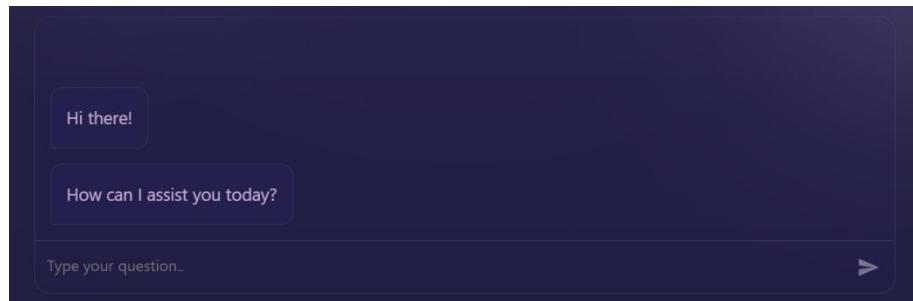
Azon túl, hogy jelentősen lerövidítse a feladatdelegálásra, illetve értekezlet összehívásra szentelt időt, az alkalmazásnak más fontos célja is van. Ez pedig strukturált, könnyen érthető és hasznos kimutatások szolgáltatása a vezetők számára. Mindazon adatok megjelenítése, ami az alkalmazáson keresztül megy: alkalmazottak leterheltsége, adott heti teendők, hamarosan bekövetkező határidők stb. Számos hasznos adatot ki lehet nyerni az alkalmazásból, ami segít egy vezetőt a gyorsabb, jobb és igazságosabb döntések meghozatalában. Ezeket a kimutatásokat könnyen személyre lehet szabni, azonban fontos arra is figyelni, hogy felesleges adatokat ne jelenítsünk meg a felületen, mivel azzal veszít az alkalmazás a lényegi előnyéből, ami a gyorsaság és a precizitás. Az elkészített prototípusban az alkalmazottak terhelési adatait és heti teendőit jelenítettem meg a vezető felületén. A leterheltségi adatokat kördiagram formájában jeleníttem meg minden egyes alkalmazothoz külön diagramon. Egy nagyon egyszerű mutató áll mögötte: (adott alkalmazott feladatainak teljes hossza az adott héten) / (az adott héten munkaidője, vagyis 40 óra). Az adott heti teendőket pedig a Google Calendar-ból emelem át minden egyes alkalmazothoz és ugyanolyan táblázatos formában jeleníttem meg.

## 2.7. Megjelenítés

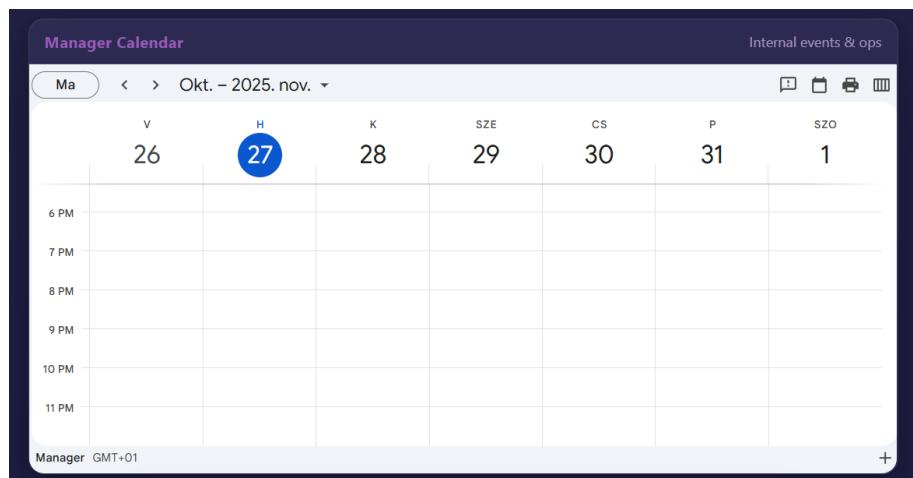
Fontos specifikálni a működésen túl azt is, hogy a szervezetben ki mit tud csinálni, kinek mi jelenik meg a felületen. Kettő fő nézetet lehet megkülönböztetni: vezetői nézet, alkalmazotti nézet. Amint azt az imént már említettem a vezető számára láthatóak kimutatások az alkalmazottak munkaidjét illetően. Ezzel szemben az alkalmazotti nézetben minden alkalmazott csak a saját adatait látja.

A menedzseri nézet négy fő elemet tartalmaz: chat input mező, menedzseri naptár, alkalmazotti naptár carousel (lapozható megjelenítés), aktuális idő. Ezek a 3-tól 6-ig sorszámozott ábrákon tekinthetők meg, illetve később a 8. és 9. ábrán is láthatóak. A chat inputra igazak a korábban specifikáltak, ezen keresztül érhető el a munkafolyamat és az AI agent. Az egyes bemeneti mezők továbbítják az agent felé azt is, hogy melyik felhasználó küldte a kérést így tudja ellenőrizni a jogosultságokat. A chat input mező alatt látható egy beágyazott Google Calendar ablak, amiben a menedzser saját naptára látszik, a saját eseményeivel. Alatta pedig az alkalmazotti naptárak, amelyek egy lapozható elemben jelennek meg. Itt nyilakkal válthat az egyes naptárak között. Ez azért fontos, hogy a menedzser akár saját maga is át tudja nézni az egyes alkalmazottak heti beosztását. Legalul pedig egy óra látható, ami megjeleníti a pontos időt.

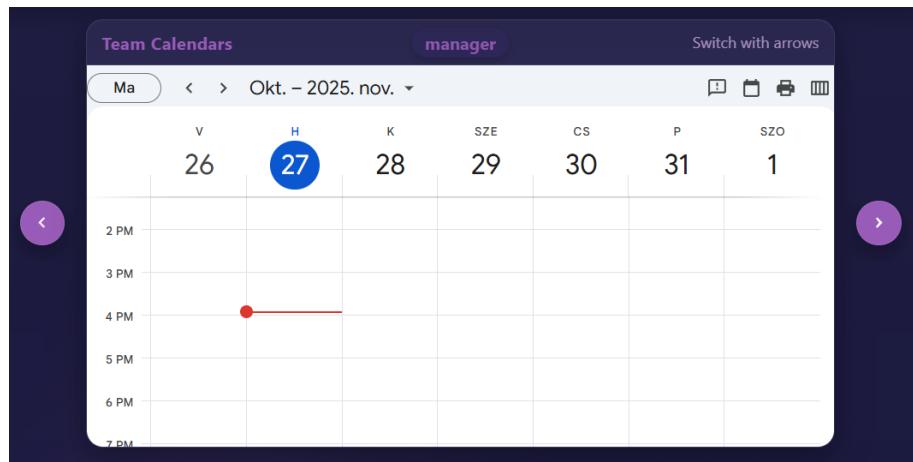
Az alkalmazotti nézetben 3 fő elem található, ugyanazok, mint a menedzseri nézetben, leszámítva természetesen a lapozható naptár elemet. A bemeneti elem itt is továbbítja a felhasználó nevét, így az alkalmazotti nézetből nem hozható létre meeting, illetve nem delegálható feladat. Azonban lekérdezhető információ és megadhatók az aktivitással kapcsolatos adatok.



3. ábra: Chat input mező



4. ábra: Felhasználó naptára



5. ábra: Naptár carousel

### **3. SKÁLÁZHATÓSÁG, BŐVÍTHETŐSÉG**

A megjelenítésen, illetve a működésen túl fontos aspektus a tervezésnél a skálázhatóság és a bővíthetőség is. A skálázhatóság ebben az esetben a felhasználó szám növelhetőségét jelenti, illetve annak korlátait, a bővíthetőség pedig az új funkciókkal való kiegészítés módját, egyszerűségét.

#### **3.1. Skálázhatóság**

Skálázhatóság terén két dolgot érdemes megvizsgálni: új felhasználó felvételéhez szükséges idő, illetve a maximális elérhető felhasználó szám. A skálázhatóságot legfőképpen a Google Calendar felhasználói megkötései határozzák meg. A felhasználói szám növelése jelen esetben az új alkalmazott felvételét, a rendszerbe való, integrálását jelenti. Az architektúra Google Calendart érintő felépítéséből következik, hogy egy új alkalmazott felvételéhez a felhasználón belül egy új “alnaptárat” kell létrehozni, fel kell venni az alkalmazott adatait az arra kijelölt táblázatba, illetve a felület kódjában fel kell venni az alkalmazott saját oldalát és hozzá csatolni a naptárát. Ezek mellett az n8n-ben is fel kell venni az új naptárat kezelő toolokat illetve a hozzájuk tartozó sorokat a rendszer prompt-ba. Ez rendkívül gyorsan megtehető, körülbelül 5-10 percet vesz igénybe. Tehát, az új felhasználó felvételének ideje kellően rövid, ebből a szempontból az alkalmazás jól skálázható. Azonban a felvehető naptárak számát illetően merülnek fel korlátozások. Igaz, nincsen a Google Calendar dokumentációjában maximalizálva, hogy egy fiókkal hány naptárat lehet felvenni, (Google Developers, n.d.) ajánlást adnak erre az értékre. A Google segítő fórumain olvasható, hogy nincsen maximalizálva a szám, de 60 naptár felvétele után read-only módba vált a felület, így nem vehető fel új esemény. (Google Support, n.d.) Ez feltehetőleg az erőforrás használat korlátozása érdekében történik így, viszont a projekt szempontjából ez rendkívül nagy problémát jelentene. Ebből kifolyólag 60 alkalmazottig gond nélkül skálázható az alkalmazás, utána pedig egy újabb fiók létrehozásával ismét 60 alkalmazottat fel tudunk venni és ez az eset sem tart sokkal több ideig (egy új email cím létrehozása szükséges). Ez a módszer természetesen nem kézenfekvő több száz vagy ezer felhasználónál, azonban az alkalmazás nem is több száz embert foglalkoztató szervezetek számára készült, hanem kisvállalkozások számára, melyek nagy része belefér a 60-as létszám korlátba.

#### **3.2. Bővíthetőség**

Bővíthetőség alatt azt értjük, hogy milyen módon egészíthető ki az alkalmazás új funkciókkal. A jól bővíthető alkalmazásoknál, ez könnyen megtehető, jellemzően a rendszer

kis részén kell változtatásokat elvégezni, míg a rosszul bővíthető alkalmazásoknál a rendszer nagyobb részét kell módosítani. Jelenleg az alkalmazás a meeting összehívást, feladatdelegálást és az aktivitáskötést támogatja. Ezen túl hasznos funkció lehet a menedzseri oldalon az emailek automatizálása, vagy egy mind menedzseri, mind alkalmazotti oldalon elérhető tudásbázis csatolása. Ezekben túl egyéb funkcióval is könnyedén kiegészíthető az alkalmazás. A felületeken nem kell változtatni, mivel az egyes funkciók minden chat bemeneten keresztül érhetőek el. Miután létrehoztuk az új funkció elemeit vagy egy külön munkafolyamatot az AI-ügynökhöz, tool formájában csatlakoztathatjuk. Ezután bele kell foglalni a "rendszerpromptba", azaz az ügynök működését meghatározó rendszerüzenetbe.

## **4. IMPLEMENTÁCIÓ LEÍRÁSA**

A megvalósítás módja fejezet ismertette a fő elképzést és működést, hogy milyen elemekből épül fel az architektúra és melyik elemnek milyen feladata, illetve felelőssége van. Jelen fejezet célja, hogy egy szinttel továbblépjön, és bemutassa a konkrét megvalósítást, az egyes elemek beállításait, a mögöttük álló logikát, valamint áttekintést adjon a felhasználói felület felépítéséről. A teljes architektúra képét a harmadik függelék tartalmazza.

### **4.1. Gmail környezet kialakítása**

Amint az a korábbi fejezetekben már kiderült, az alkalmazás alapja a Google Calendar. Itt tárolódnak az egyes létrehozott feladatok és értekezletek. minden további implementációs lépés előtt létre kellett hozni az egyes dolgozók naptárait, amelyekbe később beilleszthetők lesznek a tevékenységek.

Ahogyan a munkafolyamat architektúránál, itt is több megvalósítási lehetőség merült fel. Első elképzélés egy szervezeti gmail környezet kialakítása volt, ahol minden dolgozó rendelkezik saját gmail fiókkal, ezáltal email címmel és Google Calendarral. Ez számos előnyt jelentene. Például rendelkezne minden dolgozó ingyenes tárhellyel, amit a munkához fel tudna használni, illetve el tudnák különíteni a levelezéstüket a privát beszélgetésektől. Bizonyos vállalatoknál ez bevett szokás, így kézenfekvő lenne a használata. Azonban a projekt során mégsem bizonyult megvalósíthatónak a koncepció, mivel az egy telefonszámmal létrehozható gmail fiókok száma korlátozott, és ez akadályt jelentett.

Ebből adódóan egy már meglévő gmail fiókot kellett felhasználni a környezet kialakításához, ami a prototípusban a “céges email” szerepét tölti be. Ez nem korlátozza az alkalmazást semmilyen funkcionálisában, illetve a felületi megjelenést sem befolyásolja. A Calendar felhasználóban, amint azt már említettem, lehetőség van felvenni több különböző naptárt, így létre lehetett hozni minden dolgozónak egy-egy külön példányt a My Calendars fül alatt. Itt megjelenik egy “Birthdays” és egy “Tasks” naptár is, amelyek alapértelmezetten létrejönnek a fiókkal, ezekkel nem kell foglalkozni. Ezzel párhuzamosan a minta dolgozókat fel kellett venni a Google Sheet adatbázis azon munkalapjára, ami a dolgozók nevét és elérhetőségét tárolja. A 6. ábrán tekinthetők meg az említett adatok.

The screenshot shows a Google Sheets interface. On the left, there is a sidebar titled "My calendars" with checkboxes for various users and categories. The main area contains a table with three columns: "A", "B", and "C". The data is as follows:

A	B	C	
1	Name	Email	Role
2	Kozma Szabolcs	kozmaszabolcs@gmail.com	Editor
3	Bányász Emma	banyaszemma@email.com	Project manager
4	Kovács István	kovacsistvan@email.com	Editor
5	Szabó József	szabojozsef@email.com	Editor
6	Horváth Miklós	horvathmiklos@email.com	Project manager
7	Magyar Károly	magyarkaroly@email.com	Project manager

6. ábra: Minta felhasználók adatai és naptáraiak

Annak érdekében, hogy a naptárak hozzáférhetőek legyenek a felület kódjából, mindegyiknél végre kellett hajtani néhány módosítást a beállításokban. Elsősorban nyilvánossá kellett tenni a naptárat, hogy annak adatait a Google Calendar alkalmazáson kívül is le lehessen kérdezni. Ezt egyszerűen a megfelelő rubrika megjelölésével meg lehetett tenni, illetve ki lehetett választani a megosztás módját: minden adat mutatása, vagy csak foglalt/szabad státusz mutatása. Itt a minden adat mutatása lehetőség a megfelelő, mivel a dolgozóknak saját felületükön látniuk kell a pontos részleteket, illetve a chat felületen is így lesznek megfelelően lekérdezhetők. A nyilvánossá tétele a 7. ábrán látható.

#### Access permissions for events



7. ábra: Naptár nyilvánossá tétele

Ezen túl a Google Calendar minden naptárhoz megadja a beágyazáshoz szükséges kódot is, az "Embed code" szekcióban. A kód egy iframe elemet tartalmaz, a naptár azonosítójával, illetve néhány változóval együtt, amivel a megjelenő naptár méretét lehet állítani.

## 4.2. Architektúra megvalósítása

Miután elkészültek a dolgozók naptárai, következő lépés volt az n8n munkafolyamat megvalósítása. Ez magában foglalja többek között az egyes node-ok létrehozását, az AI agent konfigurálását, az LLM modell hozzá kapcsolását, illetve a Google Calendar és Google Sheet eszközök csatlakoztatását.

#### **4.2.1. Chat Input Node**

Az elsőként felvett elem a chat input node, itt fogadja a folyamat a beírt üzenetet. Az elem tartalmaz egy chat URL-t, aminek a felhasználásával később lehetőség lesz a felhasználói felületről elérni azt és azon keresztül a teljes workflow-t.

A legfontosabb beállítások, amelyeket módosítani kellett, a chat nyilvánossága, a felhasználás módja, illetve az autentikáció. Természetesen nyilvánossá kellett tenni a beszélgetést, hogy el lehessen érni azt az URL-en keresztül. A felhasználás módjánál két lehetőség közül lehet választani: hosted chat és embed chat. A hosted chat lehetőségnél maga az n8n generál egy, annak saját domain-jén keresztül elérhető, weboldalt, ahol megjelenik a chat bemenet és kommunikálni lehet a felépített munkafolyamattal, azonban itt nem lehet személyre szabni. A másik, “embed chat” opciónál ez viszont megtehető, így ez a megfelelő választás. Az n8n dokumentációjában pontos leírás található a beágyazás módjáról, minta kódot is szolgáltatnak, amit a weboldal kódjába beillesztve használhatóvá válik a workflow. Erről részletesebben a felületek elkészítése szekció nyújt ismertetőt. Nem utolsó sorban be kellett állítani az autentikáció módját is. Lehetőség van az n8n-re bízni az autentikációt, azonban ehhez előfizető felhasználónak kell lenni és a szolgáltató szervereit kell igénybe venni. Ebből adódóan az autentikációt statikusan a weboldal kezeli, ami éles rendszernél nem lenne biztonságos, de tesztelési célra alkalmas és nem kerül extra költségekbe. A “Basic auth” (az n8n felhasználó jelszava használandó mindenhol), “n8n User Auth” (lehetőség van különböző userek felvételére) és a “none” (nem foglalkozik az n8n autentikációval) lehetőségek közül a legutóbbi került beállításra.

#### **4.2.2. AI Agent Node**

A chat bemeneti elem létrehozása után létre kellett hozni azt az elemet, amibe az becsatlakozik, akinek átadja a felhasználók által begépelt üzenetet. Ez az AI Agent Node, ami a gerincét képezni a munkafolyamatnak. Ebbe fognak csatlakozni a további elemek és ez szolgál a workflow “agyaként”, a csatlakoztatott LLM modellre támaszkodva.

Az AI agent elem beállításaiiban be kellett állítani, hogy honnan vegye a bemenetét, illetve, hogy milyen logika, illetve megkötések alapján működjön. A bemenetét a “Prompt (User message)” lehetőségnél lehetett hozzá csatolni JSON formátumban, ahol a `{} $json.chatInput {}` sort megadva a korábban létrehozott chat bemeneti elem került bekötésre. A működés leírását pedig az úgynevezett “System prompt” -ban vagy “System message” -ben lehet megadni. Ezt markdown formátumban érdemes megadni, hogy tudatni lehessen a modellel a fontosabb részeket.

A system prompt egy hosszú iterációs folyamat eredményeképpen jött létre, az alap funkcionálások és tool-ok felvételétől indulva az egyre specifikusabb részek rögzítése felé. Végül 8 egység együtteseként épült fel. A rendszer prompt pontos szövege az 1. függelékben található.

Első és legalapvetőbb része a rendszer üzenetnek, annak a szerepe. Ezt angolul “Role”-ként adtam meg az agent-nek. Ebben a részben kell specifikálni, hogy az adott agent milyen szerepet fog betölteni, a rendszerben, illetve nagy vonalakban milyen fajta viselkedés elvárt tőle.

Ezt követően meg kell adni a feladatainak a pontosabb leírását, jelen esetben, hogy a meetingeket a csatlakoztatott Google Calendarba kell majd felvennie, illetve onnan tud információkat kinyerni a csatlakoztatott tool-okkal.

Az előző pontban említett eszközöket is meg kell adni a pontos nevükkel, ahogyan a munkafolyamatban is szerepelnek, illetve egy leírást adni, hogy milyen esetben használja az adott tool-t és mit tud vele csinálni. Ebben a projektben ezek azok az eszközök, amelyekkel az alkalmazottak email címeit tudja lekérni a táblázatból, az egyes személyek naptáraiba tud felvenni eseményeket, azokat törleni, lekérdezni vagy módosítani.

Ezeken túl lehetőség van egyéb kikötéseket megadni, amit a modell követ a működés során. Ezek kiterjednek a létrehozandó események információira, hogy miként határozza meg az időablakot, milyen típusú feladatról van szó (feladatdelegálás vagy értekezlet), illetve az esemény leírása. Emellett bizonyos szabályok is itt kapnak helyet az értekezlet létrehozással, dátummal, munkaidővel, válasz formátummal, valamint azzal kapcsolatban, hogy mennyire dolgozhat “saját kútföből”.

#### **4.2.3. LLM Node**

LLM-ként, ahogyan egy korábbi fejezet is kifejtette, az OpenAI GPT-4o mini modellt használja az alkalmazás. Emiatt hozzá kellett adni egy kapcsolódó LLM Node-ot, az „OpenAI Chat Model”-t. Lehetőség lett volna más szolgáltatókat szimbolizáló nagy nyelvi modell gráf pontot felvenni, mint például Anthropic, Ollama vagy Gemini.

Ez az elem első látásra az egyik legegyszerűbb elemnek tűnhet, mivel elég kevés mezővel rendelkezik, mindössze kettővel: “Credential to connect with” és “Model”, vagyis az LLM szolgáltatónál használható érvényes hitelesítő adatok és a használandó modell. Az utóbbit egy legördülő listából kell csupán kiválasztani. Természetesen a döntés mögötti kutatómunka igényelhet több erőfeszítést.

A hitelesítő adatok megadása bonyolultabb. Itt van lehetőség az API kulcs megadására, amin keresztül elérhetővé válik a modell. Az API kulcsot a szolgáltató oldalán lehet igényelni. Először be kell regisztrálni, majd egyenleget feltölteni a fiókhoz. Ebből az egyenlegből fog levonásra kerülni a modell használati díja, ami egymillió bemeneti, kimeneti, illetve kontextuális tokenenként van megadva. A projekt során 5\$ került felhasználásra egyenlegként, ami elegendő is volt a tervezés, illetve a tesztelés folyamatához. Az egyenleg feltöltése után létre kell hozni az API kulcsot, amit az “Organization” szakasz “API keys” fülön lehet megtenni. Létrehozás után a kulcs “secret key” attribútumát el kell menteni egy biztonságos helyre, mivel később nem lesz megtéríthető egészében. A kimásolt titkos kulcsot az n8n platformon be kell másolni a “Credential to connect with” lehetőséghez, aztán elmenteni, így az AI agent már majdnem használható lesz.

#### **4.2.4. Simple Memory Node**

Ahhoz, hogy az AI ügynök használható legyen, vagyis képes legyen az LLM modellt felhasználva válaszolt alkotni bemenetekre, szükséges egy memóriát csatolni hozzá. Több memória típus közül lehet választani. Mindegyik ugyanazt a célt szolgálja, azonban a token felhasználása, a pontosság és a háttér működése különböző. Az n8n-ben ez a fajta memória eszköze három adatbázis szolgáltatónál érhető el: MongoDB, Redis és Postgres. Ezek mellett lehetőség van az n8n saját beépített memória eszközét használni, ami a “Simple Memory Node” névre hallgat.

A projekt szempontjából ez a legutóbbi volt a legalkalmasabb, mivel a többit komplex alkalmazásokhoz ajánlják, melyek ennyi adat tárolásához nem lettek volna hatékonyak. Be kell állítani egy értéket, hogy hány üzenetre terjedjen ki az agent kontextusa. Ennyi üzenetre fog az ügynök “visszaemlékezni”. Egy ekkora alkalmazásnál 10 üzenet kontextus elegendő, főleg tesztelésnél célszerű kisebb “Context window” -t állítani, mivel így kevesebb tokent használ a modell, ugyanis minden új üzenetnél újraolvassa az egész kontextust.

#### **4.2.5. Google Calendar Node**

A munkafolyamat legtöbb példányban előforduló elem típusa a Google Calendar Node. Összesen négyszer hétfő, azaz 28 példányban. minden dolgozóhoz, beleértve a vezetőt is, tartozik négy különböző Calendar Node: lekérdező, létrehozó, törlő és módosító. Mivel minden embernek külön naptára van, így külön elem is szükséges az egyes műveletekhez.

A naptár node-ok létrehozást követően eleinte megegyeznek. Mindegyiknél be kell állítani a hitelesítő adatokat, amin keresztül elérhetőek a naptárak, a “Resource” -t, az “Operation” -t és a “Calendar” -t.

A Resource a művelet forrását jelöli, vagyis, hogy egy eseményen (Event), naptáron (Calendar) kell műveletet végrehajtani, vagy esetleg egyedi API hívás (Custom API call) kerül megadásra. Itt minden esetben az Event lehetőség került beállításra, mivel az egyes naptárakba csak a bejegyzéseket kell módosítani, magukat a naptárakat nem, illetve új naptárakat sem kell felvenni. Az Event Resource-nál az Operation lehetséges beállításai: Create, Delete, Get, Get Many, Update, Custom API Call. Ezek ahogyan a neveik is mutatják az események létrehozását, törlését, lekérdezését, csoportos lekérdezését és módosítását támogatják. Az előző ponthoz hasonlatosan, itt is lehetőség van egyedi API hívást megadni, azonban a projekt során erre nem került sor. Azt, hogy melyik naptáron menjen végbe a művelet, a Calendar pontban lehet beállítani.

Minden művelet típus adott formát követ az elnevezésben. A lekérdező elemek a get\_[felhasználónév] karakteristikát követik. Így az AI agent is tudja egyértelműen, melyik eszközt hívja meg, ha az egyik dolgozónak az eseményeit kell lekérdezni. Hasonló formát követ a többi Calendar tool típus is. Az esemény létrehozó eszközök a create\_[felhasználónév] nevezéktant követik, míg a törlő és módosító elemek a delete\_[felhasználónév] és update\_[felhasználónév] terminológiát.

A lekérdező “get\_” tool-ok, mint az már kiderült, a lekérdezés támogatására vannak. Ezzel tudja a modell megnézni, hogy egy adott dolgozónak milyen tevékenységei vannak rögzítve a naptárában egy adott időablakban. A lekérdezés a Get Many operációt használja, mivel események csoportját kell visszaadni, amiből meg tudja határozni majd, hogy mikor szabad az alkalmazott. Három fontos beállítás van, amit említeni kell: “Limit”, “After”, “Before”. Az After és a Before jelölik az időablak elejét és végét, amiben vizsgálni kell a naptárat. Ezt kézzel is meg lehetne adni, fix értékkal, vagy egy változóhoz kötni, azonban ez nem lenne célszerű, sokkal jobb választás a “Defined automatically by the model”, vagyis az LLM model határozza meg az intervallumot a bemenet alapján. A Limit lehetőségnél pedig a visszaadott elemek számára lehet felső korlátot beállítani, ami fölött nem ad vissza többet. Az 50 azért megfelelő érték ide, mivel lehet, hogy egyszerre több nap tevékenységeit is le kell kérdezni, így 5-10 nem lenne elegendő, de 50 tevékenység átlagosan 2-3 héttel gyűlik össze.

A “create\_” elemek egy nagyon fontos dologban térnek el a lekérdező tool-uktól. Az operáció, amit elvégez az nem Get Many, hanem értelemszerűen Create. Ugyanúgy vannak

időablakot meghatározó attribútumok, viszont itt Start és End névre hallgatnak. Ezek határozzák meg a rögzítendő esemény kezdési és befejezési idejét. Amint az előző pontban, itt is a “Defined automatically by the model” a legkézenfekvőbb választás, így tudjuk leginkább kiaknázni az LLM előnyeit. Létrehozásnál lehetőség van felvenni további, opcionális attribútumokat, amelyekkel ebben az esetben élni is kell. Ilyen tulajdonságok az Attendees, Color ID, Description, Guests Can See Other Guests, Location, Show Me As, Summary és a Visibility. Az elnevezések is viszonylag jól leírják a szerepüket, de érdemes végig menni rajtuk. Az Attendees attribútumban tárolódnak az esemény résztvevői, innen látják a dolgozók, hogy kikkel fognak részt venni egy meetingen, vagy ha közös feladat van, akkor kivel kell együtt dolgozni. A Description tartalmazza a rövid leírását az eseménynek, amiből a dolgozók informálódhatnak a feladatot vagy értekezletet illetően. A Location természetesen a helyszínt jelöli, ezzel kapcsolatban a System prompt-ban lehet megadni utasításokat, a kezelési módról. A Summary a Description-höz hasonló szerepet tölt be, ez lesz az esemény címe a naptárban. Az eddig említett attribútumokat, mind a model határozza meg önállóan, a többi négy beállításnál pedig fix érték szerepel. Az esemény színe minden dolgozónál más, minden igaz, hogy a meghívottak látják egymást (Guests Can See Other Guests), a Show Me As, vagyis, hogy milyen státuszba teszi a dolgozót az esemény minden a “Busy”, azaz Elfoglalt, (másik lehetőség az Available, vagyis elérhető), illetve minden esemény láthatósága (Visibility) publikus.

Az összetett, sok beállítással bíró Create művelet után a Delete egy sokkal kisebb feladat. A közös tulajdonságokon kívül, minden attribútum van, amit meg kell határozni, ez pedig a törlésre kijelölt esemény azonosítója, Event ID-ja. Ennek meghatározása is a modellre van bízva, először végrehajt egy get\_ műveletet, onnan ki nyeri az azonosítót, majd törli az eseményt. Ez a viselkedés és az eszközök használatának láncolata a System Prompt-ban van rögzítve.

Végül, minden naptárhoz tartozik egy Update metódus, mint csatolt eszköz, az események frissítéséhez. Itt az operáció Update, illetve a létrehozásnál és törlésnél említett attribútumok közül kerül újra elő néhány: Event ID, Attendees, Description, Location, Start, End és Summary. Ezek szerepe megegyezik a korábbiakkal.

#### 4.2.6. Google Sheet Node

A Google Calendar mellett a másik Google szolgáltatás, amit használ az alkalmazás az a Google Sheets. Ebben tárolódnak az alkalmazottak nevei, email címei, tranzakciós adatok és egyéb kimutatások.

Három munkalapon találhatóak meg az említett információk. Az első munkalap tartalmazza a dolgozók teljes nevét és email címét. Ezt reprezentálja az “attendee\_emails” elnevezésű Google Sheet Node. Itt is, mint az előbb, meg kell adni Resource-t, és Operation-t, miután létrehoztuk a hitelesítő adatokat és bemásoltuk azokat a megfelelő helyre. Az előző pontban a Resource az Event és a Calendar volt, vagyis mondhatni egy gyermek-szülő viszony állt fenn közöttük. Ez most sincs másként, a választható lehetőségek a Document és a Sheet Within Document, azaz maga a dokumentum, vagy egy azon belüli munkalap.

Természetesen egész dokumentum nem kerül létrehozásra vagy törlésre, így a a második lehetőség a megfelelő. Műveletet pedig egy nyolc elemű listából lehet választani: Append or Update Row (sor felvétele vagy módosítása), Append Row (sor felvétele), Clear (a munkalap egészének vagy egy részének törlése), Create (munkalap létrehozása), Delete (munkalap törlése), Delete Rows or Columns (sorok vagy oszlopok törlése), Get Rows (sorok lekérdezése) és Update Rows (sorok frissítése). Az alkalmazottak adatainak eléréséhez a Get Rows operáció szükséges, így ez került beállításra, valamint a megfelelő munkalap.

A Google Sheet ezen kívül két másik fontos funkciót támogat: tranzakciós adatok és kimutatások tárolása. A tranzakciós adatok tárolásánál egy egyedi adatbázis sémát kellett definiálni, ami a következő attribútumokkal rendelkezik: alkalmazott teljes neve, feladat hossza, dátum, leírás, státusz és azonosító. Ide a modell minden egyes esemény felvétel után felvesz egy sort, kitöltve az oszlopokat a megfelelő adatokkal. Ez az adatbázis szolgál majd a kimutatások alapjául, mivel ilyen táblázatos formából sokkal hatékonyabban lehet kinyerni adatokat, mint a Google Calendar eseményeiből. Ez az adatbázis a második munkalapon foglal helyet, a munkafolyamatban pedig az “append\_data” és a “delete\_data” tool-ok reprezentálják. Az append\_data felelős az új sor felvételéért, amit az Append Row művelettel tesz meg, az URL-lel megadott adatbázis kiválasztott munkalapjára. A delete\_data eszköznél, pedig az Update Row operáció van használatban. Logikusnak tűnhet a Delete használata, azonban az csak a legfelső sort képes törölni, így nem elvárt működést eredményez. Ezért került bele a státusz attribútum, ami alapjáraton mindig “Active” értékkel rendelkezik, kivéve a törölt eseményknél, azoknál “Deleted” látható, ezt a módosítást pedig az Update Rows operáció végzi el a megfelelő sorban. Azt, hogy melyik sort kell módosítani, ismét az LLM modell dönti el, lekérdezésekkel végignézi az adatbázist és az egyes attribútumokat megvizsgálva kiválasztja a megfelelőt.

### **4.3. Error Workflow**

Ahogyan az a Megvalósítás módja fejezetben is szerepel, a működéshez nem elegendő csupán az architekturális elemek jó sorrendben való összecsatolása és beállítása. Felléphetnek olyan esetek, amikor nem megfelelően működik a rendszer. Például elfogy az egyenleg az LLM szolgáltatói fiókból, lejár egy API kulcs, elérhetetlenné válik a Google Calendar vagy a Google Sheet, vagy egyéb nem várt működés. Ezeket az eseteket a rendszer nem tudja magától javítani, ilyenkor szükséges a külső beavatkozás. Amit a rendszer tehet, hogy értesítést küld a karbantartónak, vagyis a rendszer készítőjének.

Ezt a feladatot látja el az úgynevezett Error Workflow, vagyis a hibákat kezelő munkafolyamat. Amint az már korábban szerepelt a dolgozatban, lehetőség van az error workflow-ban komplex hibakezelést is tervezni, ezáltal felkészíteni az alkalmazást kisebb hibák automatikus elhárítására, azonban ebben a projektben a lehetséges problémák zöme a felsorolt külső kapcsolatok megszűnéséből adódhat, amit nem lehet belülről kijavítani.

Ebből kifolyólag egy e-mail automatizáció kap helyet ebben a mellék munkafolyamatban. A fő workflow-hoz hasonlóan egy AI agent végzi a munkát, ami ugyanazokkal az hitelesítő adatokkal, ugyanahhoz az OpenAI fiókhoz van csatolva. Memóriának, 1 üzenetnyi kontextus van megadva, mivel nincsen szüksége a korábbi üzenetekre. Eszközöként, mindenben egy Gmail tool van csatlakoztatva, ami elvégzi az email elküldését, amit az ügynök megfogalmaz.

A Gmail eszközben is a korábbiakhoz hasonlóan csatlakoztatni kell a Google fiókot a hitelesítő adatokon keresztül, meg kell adni a Resource-t, ami itt a Message, illetve az Operation-t, ami a Send. A címzett email címnek be van égetve egy email cím, minden error üzenetet erre fog küldeni. A címet, tartalmat és csatolmányokat az AI agent határozza meg a hibaüzenet és a System prompt alapján.

A System promptot itt is személyre kell szabni, meg kell határozni az ügynök szerepét és feladatát. A fő munkafolyamatban működő agent rendszerüzenetéhez képest ez rendkívül rövid.

### **4.4. Felületek elkészítése**

Az alkalmazás weboldalon keresztül érhető el. Itt érhető el a chat felület, ami a munkafolyamat bemeneteként szolgál. A felhasználók be tudják gépelni az üzenetet, majd amint elküldik azt, aktiválódik a workflow és végbemegy a folyamat. A válasz ugyanazon a chat felületen látható.

Az egyes dolgozók nem mind ugyanazokkal a jogokkal rendelkeznek, a menedzser és az alkalmazottak között van különbség. A menedzsernek lehetősége van feladatokat delegálni és

értekezleteket összehívni, azonban ezt a többi dolgozó nem teheti meg, hogy szabályozott munkavégzés legyen a szervezetben. Ebből kifolyólag külön kell választani a menedzseri és alkalmazotti felületet.

#### 4.4.1. Felület típusok

Kétféle felület típus került definiálásra. Menedzseri és alkalmazotti felület. Menedzseri oldalból egy példány készült, alkalmazotti felületből, azonban minden alkalmazottnak külön. A kettő stílusa és kinézete azonos, azonban a tartalmuk különbözik. A menedzseri felület az alkalmazotti felület kibővített változata.

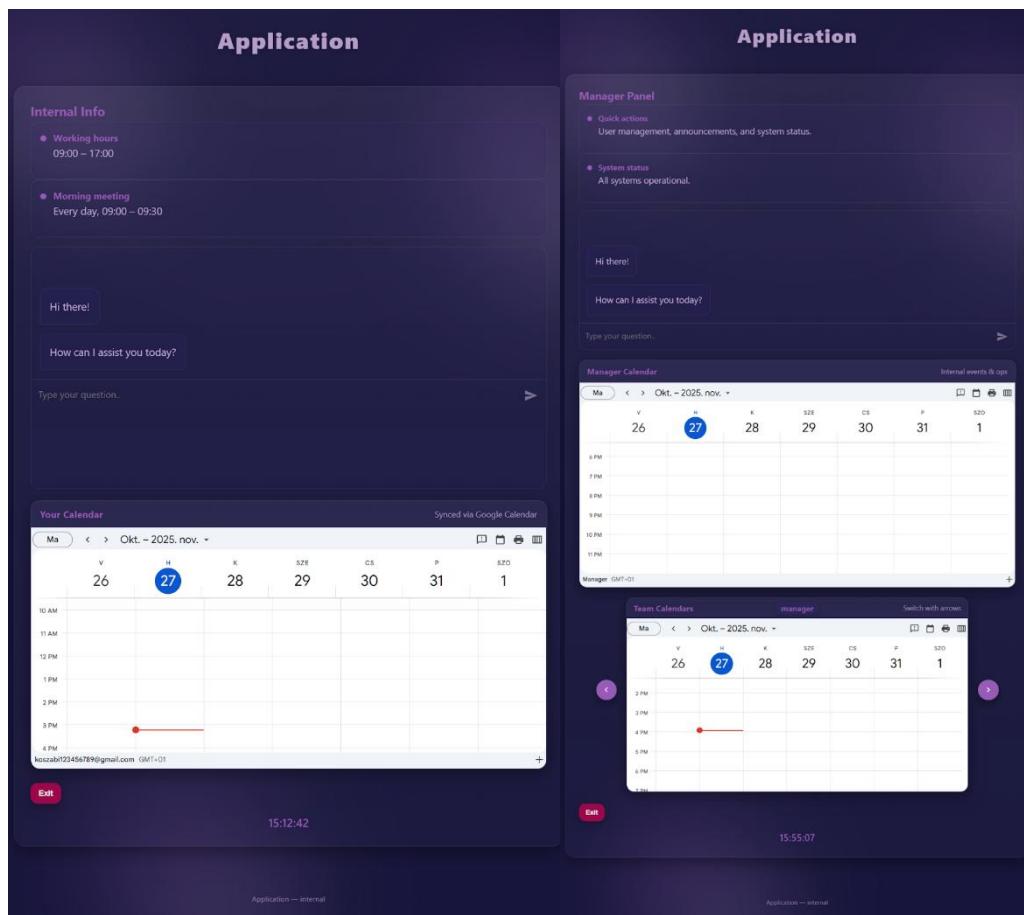
Mindkét oldal típuson legfelül található egy nagy fejléc, ami jelenleg az “Application” szöveget tartalmazza, ide lehet írni a szervezet nevét. Alatta, a fő oldalelem tetején található aktuális információkat tartalmazó elemek. Az alkalmazotti oldalon ilyen lehet például a munkaórák feltüntetése, vagy egy fix, minden nap ismétlődő meeting időpontja. A funkcionálitásban ennek nincsen szerepe. Menedzseri oldalon ugyanezen a részen egy “Quick actions” és egy “System status” mező látható, melyek statikus szövegek, nincsen hozzájuk semmilyen script kötve.

Ezek alatt helyezkedik el a chat felület. Ez kinézetre azonos, mind a menedzseri, mind az alkalmazotti oldalon. Működés szempontjából lesznek különbségek, ezt a következő alfejezet tárgyalja. Az oldal betöltésekor két előre definiált üzenet fogadja a felhasználót. “Hi there! How can I assist you?”. Alattuk látható a szövegdoboz, ahova be lehet gépelni a kívánt üzenetet, majd Enter-t nyomva, vagy a papírrepülő ikonra kattintva, elküldi az üzenetet és megjelenik az üdvözlő üzenetek alatt, eltérő színnel a jobb oldalon. A chat felület beágyazásához az n8n platform help fóruma szolgáltat minta kódot, amit személyre szabva az oldal többi eleméhez és stílusához lehet igazítani. (n8n Chat, n.d.)

A chat felület alatt minden oldalon megtalálható egy beágyazott Google Calendar elem. A fő funkcionálitást a chat felület szolgáltatja. Az alkalmazottak is le tudnak kérdezni információkat más dolgozók eseményeiről, illetve amennyiben bővül az alkalmazás bővebb tudásbázissal, azt is azon keresztül tudják elérni. Azonban a jó felhasználói élmény nyújtása érdekében szükséges egy manuálisan kezelhető naptárat is biztosítani. Ez alapértelmezetten heti nézetben jelenik meg, azonban lehet váltani havi, illetve napi megjelenésre is.

A naptár alatt a kilépést biztosító gomb található “Exit” felirattal, valamint egy óra. A gomb segítségével vissza lehet lépni az azonosítási oldalra, ahol meg kell adni a felhasználónevét és a jelszót. Erről is hamarosan több információ olvasható.

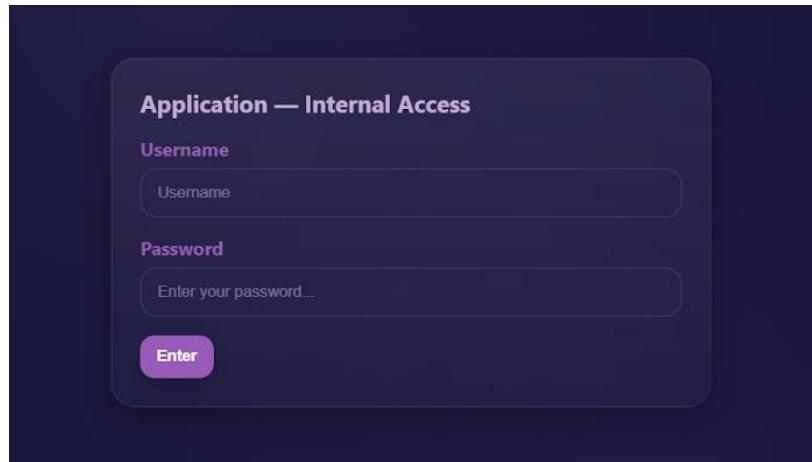
A menedzseri felületen, a személyes naptár és az exit gomb között még egy elem található. Az alkalmazottak naptárai egy lapozható carousel-ben. Két nyíllal lehet navigálni az egyes naptárak között és az adott naptárhoz tartozó felhasználónév középen, felette látható. Ha a menedzser felvesz egy eseményt a chat felületen és újra töltjük az oldalt, az meg is jelenik a naptárakban. A két különböző felületet a 8. és 9. ábrák tartalmazzák.



8-9. ábra: Az alkalmazás felhasználói felületei. Balra: alkalmazotti nézet, Jobbra: menedzseri nézet

#### 4.4.2. Azonosítás

Mielőtt megjelenne bármelyik felület, a felhasználóknak azonosítaniuk kell magukat. Erre egy külön felület szolgál, ami a 10. ábrán látható.



10. ábra: Bejelentkezési felület

Itt meg kell adniuk a felhasználóknak a felhasználónévüket és jelszavukat, majd az Enter gombra kattintva be tudnak lépni az oldalra. Két felhasználó típus van, egy a menedzsernek és egy az alkalmazottaknak. Ez segít majd a megfelelő felületek megjelenítésében. Ezen túl más fontos szerepe is van a felhasználó típusoknak. Amint már felmerült, az alkalmazottak nem hozhatnak létre értekezleteket és nem delegálhatnak feladatot. Annak ellenőrzésében, hogy ki menedzser és ki nem, a típusoknak van szerepe. Amikor egy felhasználó elküld egy üzenetet, meta adatként elküldésre kerül az üzenetet küldő felhasználó neve és szerepköre is. A teljes bemenet, amit a munkafolyam megkap így néz ki:

```
[  
 {  
   "action": "sendMessage",  
   "sessionId": "4d4fc6dd-ffffd-42e4-aec4-fd048e6e9431",  
   "chatInput": "Do i have anything on monday?",  
   "metadata": {  
     "username": "kozmaszabolcs",  
     "role": "user",  
     "page":  
       "/C:/Users/User/Documents/Saj%C3%A1lt%20projektek/Management/page.h  
tml"  
   }  
 }]  
 ]
```

Látható a begépelt szöveg, az üzenet azonosítója és a metaadatok, köztük a felhasználónévvel és a szerepkörrel. Az AI agent ezt a szerepkört felhasználva ellenőrzi, hogy jogosult-e a felhasználó a tevékenység végrehajtására.

#### **4.4.3. Használt eszközök, nyelvek**

A weboldal és a felületek konkrét kódjának ismertetése és bemutatása nem része a dolgozatnak. Ennek legfőbb oka, hogy a fókusz az automatizáló platformon készített munkafolyamaton van, illetve annak a menedzseri feladatokban való felhasználásán és nem a webes fejlesztésen. Maga a kód több mint ezer sor, így csak a fontosabb részeket kiemelve is túl nagy terjedelmet igényelne.

Ennek ellenére a felhasznált eszközökről és programozási nyelvekről röviden érdemes ejteni pár szót. A fő felhasznált technológiák a HTML5 (Hypertext Markup Language 5), CSS3 (Cascading Style Sheet Level 3), JavaScript, valamint külső könyvtárak és szolgáltatások. A teljes weboldalt leíró kód 21%-ban HTML, 52%-ban CSS és 27%-ban Java.

HTML képezte a szemantikus felépítés alapját: különböző szekciók, fejlécek, láblécek stb. A beágyazott naptár kódját is egy HTML kód részletben adja meg a Google Calendar. Ezt egy iframe elemben kellett létrehozni.

A CSS-t a témák és színek egységes kezelésére lehetett a legjobban használni. Segítségével reszponzív tipográfiát és elrendezést lehetett kialakítani, vagyis különböző méretű eszközökön is jól jelenik meg. Támogatja a glassmorphism-szerű megjelenést (áttetsző hátterek, finom outline/árnyékok), animációkat és a fókusz-állapotokat, vagyis az egyes elemek, mint az exit gomb vagy a carousel-t irányító nyilak stílusváltozását, amikor rajtuk van az egér.

A JavaScript adta az eszközöket a chat beszélgetés beágyazásához, az állapotkezeléshez (különböző felhasználó típusok), az időzítéshez és UI frissítéshez (óra) valamint az egyszerű azonosításhoz.

## **5. TESZTELÉS**

Az alkalmazás tervezett funkciói egyértelműen hasznosak egy vállalkozásban a menedzser számára. Nagyban lerövidíti a feladatok kiosztásának folyamatát, megspórol a dolgozóknak számos meetinget, pontosabbá teszi az információáramlást, és nyomon követhetővé teszi a működést.

Azonban ahhoz, hogy kiderüljön, hogy az elkészített prototípus is alkalmas-e mindenre, vagyis, hogy minden funkció megfelelően működik-e jól kiterjedt tesztelésre van szükség. Ennek a fejezetnek a célja pedig pontosan ez. Először ismerteti a tesztelés menetét: a tesztelés módját/ módszerét, a tesztelési metrikákat, valamint az egyes teszteseteket. Végül pedig az összesített eredményeket is bemutatja, amiből kiderül az alkalmazás teljesítőképessége.

### **5.1. Tesztelés módja**

Az alkalmazás teszteléséhez a black box tesztelés volt a legalkalmasabb. A black box tesztelés során előre rögzítik a bemenetet és az elvárt kimenetet a specifikáció alapján úgy, hogy a tesztelő nem ismeri a belső működést. Ennek az ellentéte a white box tesztelés, amikor a tesztelő ismeri a belső működést, azonban ez ebben a projektben nem alkalmazható. A white box tesztelésnél a forráskódot tesztelik, az egyes függvényeket, metódusokat stb. Azonban jelen esetben a függvények működését az n8n elfedi. (Nidhra és Dondeti, 2012)

### **5.2. Tesztelési metrikák**

A tesztelés során használt legfontosabb metrikák a use-case (felhasználói út) lefedettség, hiba sűrűség és a tesztek sikereségi aránya. (Burnstein, 2003, Amland, 1999) A bemenet chat jellegéből adódóan nem illenek az alkalmazásra a további jellemző black box testing metrikák. Szokás az említett metrikákon felül a különböző bemeneti és kimeneti érték variációkat tesztelni, valamint a határértékeket vizsgálni, azonban jelen esetben ezek nem alkalmazhatók. Mivel a bemenet természetes nyelvi formátumú, nincsenek egyértelmű bemeneti értékek és határértékek. A bemeneti üzenet minőségében lehet különbséget tenni, például tökéletes, kicsit zajos, nagyon zajos esetek külön tesztelése. Ezek külön use-case-ként kerültek felvételre az egyértelműség kedvéért.

### **5.3. Teszt esetek**

A tesztesetek meghatározásánál minden funkcionálitáshoz össze kell szedni az elvárt működés szerinti lehetséges lefutásokat, hibalehetőségeket és kritikus pontokat. Az alkalmazás

fő funkciói, amelyek mentén a tesztelés történik a meeting létrehozás, feladat delegálás és az információ lekérdezés.

Minden műveletnél szempont, hogy ki végzi azt, egy menedzser vagy egy alkalmazott. Erre a jogosultságok tesztelése miatt van szükség, hogy minden esetben jól működő és biztonságos az alkalmazás. Az értekezlet létrehozás és a feladat delegálás természetesen csak a menedzser hatóköre, azonban információt minden dolgozó kérdezhet le.

Fontos aspektus még a megadott információk pontossága és a helyesírás. Az információk pontossága azt jelenti, hogy például egy értekezlet létrehozásához minden információ rendelkezésre áll-e az első üzenetben. Amennyiben vannak hiányos, nem említett részek, két lehetősége van a rendszernek. Vagy visszakérdez és bekéri a hiányzó adatokat, vagy a System promptban megadott alapértelmezett információkat használja. Például, ha nincsen megadva feladatdelegálásnál időablak, akkor a következő 48 órát veszi számításba alapértelmezettként. Ilyenkor ez az elvárt kimenet. A helyesírás a bemeneti üzenet minőségére utal, hogy a nevek pontosan vannak-e megadva (vagy becenével), az egyes szavak helyesen vannak-e leírva, illetve a mondatszerkezet mennyire érthető.

Értekezlet létrehozása során le kell tesztelni, hogy van-e különbség, ha csak egy embert adunk meg, vagy többet, esetleg mindenki hivatalos rá. Ugyanezt meg kell tenni a feladat delegálás esetében is. Ott a két eset, hogy valaki egyedül végez egy feladatot, vagy többen végzik azt projekt jellegben. Ezknél a teszteknél az elvárt kimenet, hogy ugyanazokkal az adatokkal létrejön minden megjelölt ember naptárában az esemény. (Winder, 2024)

A fent említett szempontok számos variációt hoznak létre, amit le kell ellenőrizni. Összesen 108 teszteset került definiálásra. Ezeket a 2. függelék tartalmazza. Az egyes oszlopokban láthatóak a tesztesetet meghatározó attribútumok, illetve az utolsó oszlopban, hogy az elvárt eredményt hozta-e vagy sem.

#### **5.4. Eredmények**

A 2. függelékben látható, mely esetek hozták a várt viselkedést és melyeknél tért el attól. Az összesen 108 teszt-esetből négy volt, ahol a rendszer nem volt képes végrehajtani az elköpzelt műveletet a bemenet alapján. Ez alapján egyértelműen elmondható, hogy az alkalmazás rendkívül jól működik és képes a feladatok ellátására. Azonban a cél mindenig a tökéletes működés, így közelebbről is meg kell vizsgálni a hibás eseteket.

Mind a négy esetben a bemeneti üzenetnek ugyanaz a tulajdonsága okozta a problémát, ami nem más, mint a helyesírás összevisszásága. Ha valaki jobban megnézi a 2. függelékben

található táblázatot és a hibás eseteket összeveti azokkal az esetekkel, ahol a helyesírás kategóriája ugyanaz, azonban a rendszer jól kezelte, felmerülhet benne, hogy mégis miért térnek el ezen esetek eredményei.

Erre a válasz érdekes és a probléma is könnyen orvosolható. Azokban az esetekben, ahol az alkalmazás hibázott, mindenhol az egyik megnevezett személy, akinek a feladatot ki kellett osztani, vagy akit meg kellett hívni a meetingre Kozma Szabolcs volt, azonban nem a teljes névvel referált rá az üzenet, hanem Szabi becenévvel. Ez megzavarta a rendszert, mivel ilyenkor rendre Szabó József-re gondolt az alkalmazás, neki vette fel az eseményeket a naptárba. Ennek a magyarázata az lehet, hogy a Szabó névtől csak egy karakterrel tér el a bemenet, míg a Szabolcs névtől négy karakterrel. Söt, mivel angolul folyt a tesztelés és a System prompt is angolul van megfogalmazva, feltehetőleg ezt erősítette, az angol nyelvet beszélő nemzeteknél elterjedt szokás, hogy a vezetéknévén referálnak emberekre, a keresztnévük helyett.

Amint említtettem, ez a probléma könnyen orvosolható. Egyik módja, hogy a modellnek valamilyen módon tudja, hogy egy személynek milyen becenevei vannak, így le tudja kezelní ezeket az eseteket is. Erre megoldás, ha a system prompt, vagy a Google Sheet (egy új attribútumban) tartalmazza az egyes emberek lehetséges beceneveit. Ezen túl, amint a tesztelés is mutatja, az is megoldja az anomáliát, ha a dolgozókra a teljes nevükkel, vagy legalább a helyesen leírt vezetéknévekkel vagy keresztnévekkel referálunk. Ez azonban nem minden esetben elvárható a felhasználótól, hiszen az alkalmazás célja, hogy lerövidítse a feladatkiadásra és az értekezlet összehívásra fordított időt, így a gyorsan begépelt, pontatlan üzeneteket is jól kell tudnia kezelní a modellnek.

Összességében az alkalmazás 108 esetből 104-ben az elvárt módon viselkedett, ezzel 96.3%-os pontosságot mutatott. Ez rendkívül jó eredménynek számít egy prototípushoz képest, figyelembe véve azt is, hogy az elkövetett hibák könnyen korrigálhatóak. Természetesen egy kiterjedtebb tesztelés során, ahol több ezer vagy több tízezer tesztesetet vizsgálnak meg, felmerülhetnek további hibás esetek. Azonban egy ekkora volumenű teszteléshez rendkívül sok időre és erőforrásra lenne szükség. Az elvégzett tesztelés során egy eset elvégzése átlagosan 5000 tokent használt fel. Ezt, ha arányosítjuk, egy 10 000 tesztet végrehajtó projektre, annak az API költsége is legalább 37\$, de a kritikusabb probléma az idő, ami alatt ezt el lehet végezni.

Az eredmények tovább javíthatók egy erősebb modell használatával. A tesztelés során az OpenAI GPT-4o mini modelljét használta az alkalmazás, ami rendkívül költséghatékony, azonban egy kiterjedtebb tesztelés során előjöhetnek olyan problémák, amik egy jobb modell

használatát igényelhetik. Ez viszont jelentős pénzügyi ráfordítással valósítható meg, mivel a GPT-4o modell közel húszszoros drágulást jelent ( $\sim 740\$ = \sim 240'000$  HUF).

## 6. KONKLÚZIÓ

Egy szervezet életében kulcs fontosságú az új technológiák és trendek követése. Aki hamarabb alkalmazza a legfrissebb újításokat, versenyelőnybe kerülhet konkurenseivel szemben, ami meghatározhatja a piaci helyzetét. Napjainkban a legnagyobb érdeklődésnek örvendő új technológia, a mesterséges intelligencia, mely évről évre formálja át szinte az összes létező iparágat. Hihetetlenül gyors fejlődésének köszönhetően egyes szervezetek képesek eddig még nem tapasztalt mértékű növekedést elérni. Más, akár évtizedek óta fennálló cégek azonban feledésbe is merülhetnek ennek köszönhetően, mivel az új konkurensek jobban aknázzák ki az új, dinamikusan változó, világ által nyújtott lehetőségeket.

A dolgozat célkitűzése az volt, hogy megvizsgálja a kereskedelmi LLM modellek felhasználhatóságát a menedzseri folyamatok automatizálásában. Korábbi kutatások már bizonyították, hogy a menedzsment, illetve a HR területén is rendkívüli hatékonyság növekedés érhető el, bizonyos munkafolyamatok automatizálásával. A dolgozatban bemutatott kutatás és prototípus célja azonban a legújabb technológiák felhasználása ezen a területen. Egy olyan alkalmazás elkészítése, ami a mesterséges intelligenciát használja fel arra, hogy terhet vegyen le a menedzser válláról.

A feladatdelegálás, értekezlet összehívás és az ezekhez szükséges információ lekérdezés kerültek a fókuszba. Ezek a tevékenységek heti szinten több tíz órát is elvezetnek egy vezető munkaidejéből. Egy feladat delegálása előtt, figyelembe kell venni az azonos pozícióban dolgozó alkalmazottak aktuális feladatait, a terhelésüket és a beosztásukat. Manuálisan végezve ez a folyamat felettébb monoton. Egy értekezlet összehívásánál is hasonló a helyzet. Külön egyeztetni minden alkalmazottal, kezelní az ütközéseket, akár az utolsó pillanatban újra szervezni nem egyszerű, mentálisan megterhelő lehet és sok időt igényel. Rendelkezésre állnak olyan eszközök, ahol nyomon lehet követni a kollégák napirendjét, azonban a folyamat időigényén ez nem sokat változtat.

A projekt célkeresztjében egy olyan alkalmazás létrehozása állt, amiben egy chat felületen keresztül lehet a tevékenységeket szervezni. Naprakész riportokat lehet lekérni, feladatot lehet rajta keresztül delegálni, illetve értekezleteket lehet létrehozni. Mindössze meg kell fogalmaznia a felhasználónak az igényét és a rendszert működtető AI agent lekezelni a feladatot.

Fontos szempont volt, hogy minden generáció számára könnyen használható legyen az alkalmazás. Ennek az igénynek az említett chat megoldás eleget tesz. Nincsen szükség a sok különböző menüpont, illetve kombináció megjegyzésére, mivel minden természetes nyelven ugyanazon a felületen kell megfogalmazni. Nincsen kockázat akkor sem, ha hiányos

információt ad meg a felhasználó, vagy pontatlanul ír le valamit. Amennyiben a modell nem tudja értelmezni a bemenetet, válaszüzenetben kéri az adatok pontosítását és az újonnan megadott információkkal kiegészítve dolgozik tovább.

Az említett felületet egy egyszerű, tesztelés céljából készült, webalkalmazás tartalmazza. Funkcionalitás és letisztultság szempontjából megfelelő. Viszont, mivel csak tesztelésre készült, így nem biztosít valódi védelmet, ami éles használat esetén nem lenne biztonságos. Fontos megjegyezni azonban, hogy a projekt fókusza nem a biztonságos webalkalmazás fejlesztésen, annál inkább a mögötte álló rendszer kialakításán volt.

A rendszer architektúrájának kialakítására az n8n platform szolgált. Ez egy „low code” platform révén elfedi az egyes építő elemek mögötti függvényeket, így a tervező az optimális kialakításra fókuszálhat a szintaktika helyett. Itt lehetett a rendszerhez hozzácsatolni a Google Calendar és Google Sheet elemeket, amik az adattátolást végezték. Az alkalmazást mozgató AI agent-et is az n8n-en belül volt lehetőség személyre szabni. Egyedi rendszerüzenetet és toolokat megadva fel lehetett készíteni minden eset helyes kezelésére.

A platformnak ebből a jellegéből adódóan nem lehetett white box tesztelést végezni, ami szoftvertesztelésnél bevett szokás, mivel olyankor az egyes függvények megfelelő működését tesztelik különböző bemenetekkel. Black box tesztelést kellett végezni, aminek az ismérve, hogy a rendszer belső működésének ismerete nélkül határozzák meg a teszteseteket, illetve az azokhoz tartozó elvárt eredményt. Az elvárt eredményeket a specifikációban lefektetett követelmények adják meg.

Összesen 108 teszteset került definiálásra. Ez a szám a különböző szempontok különböző lehetőségeinek kombinációjaként állt össze. Figyelembe kellett venni az elvégzett műveletet (3 lehetőség), a műveletet végrehajtó felhasználó típusát (2 lehetőség), a bemeneti üzenetben nyújtott információmennyiséget (3 lehetőség), a helyesírás minőségét (3 lehetőség), illetve a megjelölt személyek számát (2 vagy 3 lehetőség a művelettől függően). Ezeknek a variációiként jött létre az említett 108 eset.

Négy tesztesetnél nem működött megfelelően az alkalmazás, így 96,3%-os eredményt ért el. A hibás eseteket megvizsgálva, majd egy kisebb kiegészítést eszközölve a rendszer promptba, sikerült kijavítani a hiányosságot. A hibás eseteket újra lefuttatva már helyes működést produkált a rendszer, így a 2. függelék táblázatában látható tesztkészleten vett végső eredmény 100% lett. Azonban fontos megjegyezni, hogy ahogyan a 108 elemű tesztkészlet, úgy egy nagyobb és több szempontra kiterjedő tesztkészlet is felhívhatja a figyelmet hibákra. A projekt

során viszont nem volt lehetőség ennél részletesebb tesztelést végezni, mivel annak mind az API költségei, mind az időigénye jelentősen nagyobb lett volna.

A kapott eredmények azt mutatják, hogy már prototípus szinten is olyan eredményeket lehet elérni, ami bizonyítja, hogy a kereskedelmi forgalomban lévő LLM modellekre lehet olyan alkalmazást építeni, ami képes helyettesíteni vagy megkönyíteni bizonyos menedzseri feladatokat. Jelen prototípus, mint segédeszköz segít felgyorsítani, igazságosabbá és pontosabbá tenni az említett folyamatokat.

Nagyobb és hatékonyabb modellek felhasználásával, illetve kevésbé absztrakt automatizáló platform használatával rendkívüli eredményeket lehetne elérni ezen a területen. Ezen túl számos olyan felettébb hasznos funkció is van, amivel tovább bővíthető az alkalmazás. Ilyen az email fiókok kezelése, vagy a szervezeti tudásbázis létrehozása.

Egy központi tudásbázis kialakításával meg lehetne szüntetni az információhiányt, amivel sok cégnél küzdenek. Emberi hibából kifolyólag, gyakran maradnak le adatok a „briefing” -ek, vagyis tájékoztatások során, ami okán egymás utáni felesleges akadályokkal kell az alkalmazottaknak megküzdeniük és a szervezeten belüli munkamorál is csökkenhet. A tudásbázis ugyanúgy a chat felületen keresztül lehetne használható. Az az információ, amit egyszer egy felhasználó megad bemenetként, például egy éppen folyamatban lévő projekttel kapcsolatban, elérhetővé válik a szervezet többi dolgozója számára is. Így, ha valaki becsatlakozik egy projektbe, minden összes azzal kapcsolatos információt egy jól értelmezhető formába. Ezzel szintén sok hasznos óra felszabadulna, amit konstruktívabb, értékteremtő feladatokra tudnak fordítani a beosztottak. Ez tehát a legfontosabb jövőbeli fejlesztés az alkalmazással kapcsolatban.

A dolgozat során kifejtett koncepcióval nem csupán időt, de jelentős mennyiséggű anyagi forrást is meg lehet takarítani. A menedzserek fizetése nagyban függ a konkrét területtől, ahol dolgoznak, illetve a pontos munkakörüktől. A további számításokhoz a Hayes 2025-ös kimutatásai fognak alapul szolgálni. A kutatás alapján a termékmenedzsment, projektmenedzsment vagy változás- és projektmenedzsment kategóriákat tekintve a bér átlagát 550 000 forinttól (junior termékmenedzser) 2 500 000 forintig (engineering manager) terjed. Mivel az alkalmazás minden területen alkalmazható, ezeknek a kategóriáknak és tapasztalati szinteknek az átlagát érdemes venni, ami 1 376 000 forint. Ez tehát egy havi átlagos bér, ami 160 munkaóráért jár. Ez 8605 forintos órabér jelent.

Feltéve, hogy egy menedzsernek heti szinten 6 órával kevesebbet kell a korábban említett automatizálható feladatokkal foglalkoznia (naponta egy órával kevesebbet), már 206 520 forinttal felérő idő szabadult fel havonta. (Kothadiya, 2024) Ez egy nagyobb szervezet, vagy nagyobb időfelszabadítás esetén sokszorozódik. A bevezetés kisvállalkozásokat említett, mint célcsoport. Ez továbbra is így van, az említett kutatási adatok, csupán egy közelítő értéket hivatottak adni ahhoz, hogy ábrázolni lehessen, mekkora potenciált rejtenek a hasonló AI alkalmazások. Kis és középvállalkozások esetében viszont a megtakarított idő még értékesebbnek bizonyulhat. Gyakran a cég tulajdonosa látja el valamilyen szinten a menedzseri feladatokat, így a cég bevételre nála összpontosul, így egy munkaórára levetítve jelentősen felértékeli annak értékét.

Mindent egybe véve vizsgálataim során bebizonyosodott, hogy a menedzsment feladatokban, úgy a feladatdelegálásban és az értekezletek szervezésében, valamint az információ áramlásában jelentős támogatást nyújthat egy AI alapú automatizáló alkalmazás használata. Az n8n automatizáló platform felhasználásával készített prototípussal első tesztelésre 96.3%-os, javítva a hibákat pedig 100%-os eredményt sikerült elérni 108 teszeseten. Ez a terület nagy lehetőségeket rejti, mivel nincsen hasonló könnyen alkalmazható és használatba vehető eszköz.

## IRODALOMJEGYZÉK

- 1 Amland, S. (1999): Risk-based testing and metrics. International Software Testing Conference Proceedings.
- 2 Anthropic. (n.d.): Claude 3 Haiku – Generative AI on Anthropic. <https://www.anthropic.com/clause/haiku>, 2025. 10. 30.
- 3 Boris D. (2024): A Brief History of GPT Models. Edlitera. <https://www.edlitera.com/blog/posts/gpt-models-history>, 2025. 10. 30.
- 4 Bernzweig M. (2024): Business Process Automation: Latest Statistics and Trends. Software Oasis. <https://softwareoasis.com/business-process-automation>, 2025. 10. 29.
- 5 Burnstein, I. (2003): Practical Software Testing. Springer, New York.
- 6 Ekuma, K. (2024): Artificial Intelligence and Automation in Human Resource Development: A Systematic Review. SAGE Open. <https://journals.sagepub.com/doi/full/10.1177/15344843231224009>, 2025. 10. 29.
- 7 Epoch AI. (n.d.): Trends in training dataset sizes. Epoch AI. <https://epoch.ai/data-insights/dataset-size-trend>, 2025. 10. 30.
- 8 Firawi, A. M. (2024): The Impact of Automation on Human Resource Management Practices: Latest. International Journal of Business and Applied Sciences, GAFTIM. <https://www.journals.gaftim.com/index.php/ijbas/article/view/325>, 2025. 10. 29.
- 9 Fortune Business Insights. (2025): Artificial Intelligence (AI) Market Size, Growth & Trends by 2032. Fortune Business Insights, Pune. <https://www.fortunebusinessinsights.com/industry-reports/artificial-intelligence-market-100114>, 2025. 10. 30.
- 10 GeeksforGeeks. (2025): What is an API Call? GeeksforGeeks. <https://www.geeksforgeeks.org/websites-apps/what-is-an-api-call/>, 2025. 10. 30.
- 11 Gemini Team. (2024): Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv preprint arXiv:2403.05530. <https://arxiv.org/abs/2403.05530>, 2025. 10. 30.
- 12 Google Cloud Vertex AI. (n.d.): Gemini 2.5 Flash – Generative AI on Vertex AI. Google Cloud. <https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash>, 2025. 10. 30.
- 13 Google Developers. (n.d.): Manage quotas | Google Calendar API Guides. Google. <https://developers.google.com/workspace/calendar/api/guides/quota>, 2025. 10. 30.

- 14 Google Support. (n.d.): Limits for Google Calendar usage. <https://support.google.com/a/answer/2905486?hl=en&sjid=10805118400427645624-EU>, 2025. 10. 30.
- 15 Hiter, S. (2024): Best Workflow Automation Software. TechnologyAdvice. <https://technologyadvice.com/blog/project-management/best-workflow-automation-software/>, 2025. 10. 30.
- 16 Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., ... & Sifre, L. (2022): Training Compute-Optimal Large Language Models. arXiv preprint arXiv:2203.15556. <https://arxiv.org/abs/2203.15556>, 2025. 10. 30.
- 17 Kothadiya, A. (2024): The Time Spent on Managing (Not Attending) Meetings is More Than 15%. Avoma Blog. <https://www.avoma.com/blog/time-spent-on-managing-meetings>, 2025. 11. 02.
- 18 Mohamed, S. A., Mahmoud, M. A., Mahdi, M. N., és Mostafa, S. A. (2022): Improving Efficiency and Effectiveness of Robotic Process Automation in Human Resource Management. Sustainability, 14 (7), 3920. <https://www.mdpi.com/2071-1050/14/7/3920>, 2025. 11. 29.
- 19 n8n Chat. (n.d.): @n8n/chat – as npm package. npm. <https://www.npmjs.com/package/%40n8n/chat>, 2025. 10. 30.
- 20 n8n Docs: Connections. (n.d.) n8n. <https://docs.n8n.io/workflows/components/connections/>, 2025. 10. 30.
- 21 n8n Docs: Node settings. (n.d.) n8n. <https://docs.n8n.io/workflows/components/nodes/#node-settings>, 2025. 10. 30.
- 22 n8n Docs: Sticky Notes. (n.d.) n8n. <https://docs.n8n.io/workflows/components/sticky-notes/>, 2025. 10. 30.
- 23 Nidhra, S., Dondeti, J. (2012): Black box and white box testing techniques – a literature review. International Journal of Embedded Systems and Applications (IJESA), 2 (2), 29–50.
- 24 O’Brien, M. (2024): „AI ‘gold rush’ for chatbot training data could run out of human-written text”. The Associated Press. <https://www.apnews.com/article/ai-artificial-intelligence-training-data-running-out-9676145bac0d30ecce1513c20561b87d>, 2025. 10. 30.
- 25 OpenAI. (2024): Hello GPT-4o. OpenAI. <https://openai.com/index/hello-gpt-4o/>, 2025. 10. 30.

- 26 OpenAI. (2025): Introducing GPT-5. OpenAI. <https://openai.com/index/introducing-gpt-5/>, 2025. 10. 30.
- 27 Shakudo. (2025): Top 9 Workflow Automation Tools as of October 2025. Shakudo. <https://www.shakudo.io/blog/top-9-workflow-automation-tools>, 2025. 10. 30.
- 28 Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017): Attention Is All You Need. arXiv preprint arXiv:1706.03762. <https://arxiv.org/abs/1706.03762>, 2025. 10. 30.
- 29 Winder, P. (2024): Testing and Evaluating Large Language Models in AI Applications. Winder AI. <https://winder.ai/testing-evaluating-large-language-models-ai-applications/>, 2025. 10. 30.

## FÜGGELÉK

### 1. függelék:

#Role

You are a helpful management assistant. You are helping managers save time: provide employee information (email addresses, availability, already scheduled meetings etc...), delegate tasks and create new meetings according to the time schedule of the manager and the employees.

#Task

Your task are the following:

##Enroll the task delegation or meeting schedule into the calendar that is connected as a tool.

##Provide information about the availability of the employees, already scheduled meetings using the tools connected.

#Tools

Use these tools to get information and to delegate tasks or create meetings.

Logging events:

##append\_data

Use this tool to save event information. In one row only one name should appear. If there is a meeting with more people create separate rows. Store the length of the event, the date, the description, the status (active) and the ID of the event. Use this every time an event is created/deleted/updated. Only use this after creating the event in the calendar.

##delete\_data

When an event is deleted, use this tool to set its status to "Deleted"

Getting emails and roles:

##attendee\_emails

Use this tool to get the email addresses and the roles of the employees.

Getting availability:

##get\_banyaszemma

Use this tool to check Banyasz Emma availability.

Get all of the events in the period.

Also use in case of deleting to get the event ID.

##get\_magyarkaroly

Use this tool to check Magyar Karoly availability.

Get all of the events in the period.

Also use in case of deleting to get the event ID.

##get\_horvathmiklos

Use this tool to check Horvath Miklos availability.

Get all of the events in the period.

Also use in case of deleting to get the event ID.

##get\_szabojozsef

Use this tool to check Szabo Jozsef availability.

Get all of the events in the period.

Also use in case of deleting to get the event ID.

##get\_kovacsistvan

Use this tool to check Kovacs Istvan availability.

Get all of the events in the period.

Also use in case of deleting to get the event ID.

##get\_manager

Use this tool to check the manager availability.

Get all of the events in the period.

Also use in case of deleting to get the event ID.

Creating events:

##create\_banyaszemma

Use this tool to create an event in Banyasz Emma's calendar. Details in: {{ \$json.chatInput }}

##create\_magyarkaroly

Use this tool to create an event in Magyar Karoly's calendar. Details in: {{ \$json.chatInput }}

##create\_horvathmiklos

Use this tool to create an event in Horvath Miklos's calendar. Details in: {{ \$json.chatInput }}

##create\_szabojozsef

```
Use this tool to create an event in Szabo Jozsef's calendar. Details
in: {{ $json.chatInput }}
```

```
##create_kovacsistvan
```

```
Use this tool to create an event in Kovacs Istvan's calendar. Details
in: {{ $json.chatInput }}
```

```
##create_manager
```

```
Use this tool to create an event in the manager's calendar. Details
in: {{ $json.chatInput }}
```

Deleting events:

```
##delete_banyaszemma
```

Use this tool to delete any event from Banyasz Emma's calendar. Use  
the get\_banyaszemma tool to get the events and identify which one to  
delete.

```
##delete_manager
```

Use this tool to delete any event from Manager's calendar. Use the  
get\_manager tool to get the events and identify which one to delete.

```
##delete_magyarkaroly
```

Use this tool to delete any event from Magyar Károly's calendar. Use  
the get\_magyarkaroly tool to get the events and identify which one to  
delete.

```
##delete_horvathmiklos
```

Use this tool to delete any event from Horváth Miklós's calendar.  
Use the get\_horvathmiklos tool to get the events and identify which  
one to delete.

```
##delete_szabojozsef
```

Use this tool to delete any event from Szabo Jozsef's calendar. Use  
the get\_szabojozsef tool to get the events and identify which one to  
delete.

```
##delete_kovacsistvan
```

Use this tool to delete any event from Kovács István's calendar. Use  
the get\_kovacsistvan tool to get the events and identify which one to  
delete.

Updating events:

```
##update_banyaszemma
```

Use this tool to update any event in Banyasz Emma's calendar. Use the get\_banyaszemma tool to get the events and identify which one to update.

##update\_manager

Use this tool to update any event in Manager's calendar. Use the get\_manager tool to get the events and identify which one to update.

##update\_magyarkaroly

Use this tool to update any event in Magyar Károly's calendar. Use the get\_magyarkaroly tool to get the events and identify which one to update.

##update\_horvathmiklos

Use this tool to update any event in Horváth Miklós's calendar. Use the get\_horvathmiklos tool to get the events and identify which one to update.

##update\_szabojozsef

Use this tool to update any event in Szabo József's calendar. Use the get\_szabojozsef tool to get the events and identify which one to update.

##update\_kovacsistvan

Use this tool to update any event in Kovács István's calendar. Use the get\_kovacsistvan tool to get the events and identify which one to update.

#Information needed:

##Type: is it a task delegation or a meeting schedule

##Time: around when should the activity be done

If the time window is not mentioned in any ways, it should in the next 48 hours

##Description: exact description about the topic of the task or meeting

#Creating meetings:

Only create meetings when all the attendees are available.

For checking the availability use the get\_ tools.

Check for every attendee separately, the meetings and tasks should not be in conflict.

Create an event for the meeting in every employees calendar seperately.

**##Employee(s):** The employees mentioned in connection with the meeting / task / information gathering . If the activity is a task, this is not required, but for meeting and information gathering it is required.

#### #IMPORTANT

Don't make up things on your own, only provide information based on the get\_ and attendee\_emails tools.

#### #Response

Provide short, summarized responses, without any unnecessary details.

**#Date and time:** {{ \$now }}

Use Europe/Budapest time zone.

Use the YYYY-MM-DDTHH:mm:ssZ structure.

#### #Work time

Only create events from 8:30 until 17:00 every day.

#### #Email addresses:

Bányász Emma	banyaszemma@email.com
Kovács István	kovacsistvan@email.com
Szabó József	szabojozsef@email.com
Horváth Miklós	horvathmiklos@email.com
Magyar Károly	magyarkaroly@email.com
Manager	manager@email.com

#### #Undo

If the user wants to undo an event, get the information from the previous message and delete that event.

#### #Rights and actions

Only the manager can create events.

Check {{ \$json.metadata.role }} in the input metadata to know if the user has the rights for the action they want to do. Only create/delete/ update the meetings if the role is "manager".

## 2. függelék:

Sorszám	Ki végzi	Művelet	Megadott információk	Helyesírás	Résztvevők száma (értekezlet)	Megjelölt személyek száma (delegálás)	Sikeresség
1	menedzser	értekezlet létrehozás	pontos	helyes	1	-	<input checked="" type="checkbox"/>
2	menedzser	értekezlet létrehozás	pontos	helyes	több	-	<input checked="" type="checkbox"/>
3	menedzser	értekezlet létrehozás	pontos	helyes	mindenki	-	<input checked="" type="checkbox"/>
4	menedzser	értekezlet létrehozás	pontos	kisebb hibák	1	-	<input checked="" type="checkbox"/>
5	menedzser	értekezlet létrehozás	pontos	kisebb hibák	több	-	<input checked="" type="checkbox"/>
6	menedzser	értekezlet létrehozás	pontos	kisebb hibák	mindenki	-	<input checked="" type="checkbox"/>
7	menedzser	értekezlet létrehozás	pontos	össze-vissza	1	-	<input type="checkbox"/>
8	menedzser	értekezlet létrehozás	pontos	össze-vissza	több	-	<input type="checkbox"/>
9	menedzser	értekezlet létrehozás	pontos	össze-vissza	mindenki	-	<input checked="" type="checkbox"/>
10	menedzser	értekezlet létrehozás	hiányos	helyes	1	-	<input checked="" type="checkbox"/>
11	menedzser	értekezlet létrehozás	hiányos	helyes	több	-	<input checked="" type="checkbox"/>
12	menedzser	értekezlet létrehozás	hiányos	helyes	mindenki	-	<input checked="" type="checkbox"/>
13	menedzser	értekezlet létrehozás	hiányos	kisebb hibák	1	-	<input checked="" type="checkbox"/>
14	menedzser	értekezlet létrehozás	hiányos	kisebb hibák	több	-	<input checked="" type="checkbox"/>
15	menedzser	értekezlet létrehozás	hiányos	kisebb hibák	mindenki	-	<input checked="" type="checkbox"/>
16	menedzser	értekezlet létrehozás	hiányos	össze-vissza	1	-	<input checked="" type="checkbox"/>
17	menedzser	értekezlet létrehozás	hiányos	össze-vissza	több	-	<input checked="" type="checkbox"/>
18	menedzser	értekezlet létrehozás	hiányos	össze-vissza	mindenki	-	<input checked="" type="checkbox"/>
19	menedzser	értekezlet létrehozás	semmilyen	helyes	1	-	<input checked="" type="checkbox"/>
20	menedzser	értekezlet létrehozás	semmilyen	helyes	több	-	<input checked="" type="checkbox"/>
21	menedzser	értekezlet létrehozás	semmilyen	helyes	mindenki	-	<input checked="" type="checkbox"/>
22	menedzser	értekezlet létrehozás	semmilyen	kisebb hibák	1	-	<input checked="" type="checkbox"/>
23	menedzser	értekezlet létrehozás	semmilyen	kisebb hibák	több	-	<input checked="" type="checkbox"/>

24	menedzser	értekezlet létrehozás	semmilyen	kisebb hibák	mindenki	-	<input checked="" type="checkbox"/>
25	menedzser	értekezlet létrehozás	semmilyen	össze-vissza	1	-	<input checked="" type="checkbox"/>
26	menedzser	értekezlet létrehozás	semmilyen	össze-vissza	több	-	<input checked="" type="checkbox"/>
27	menedzser	értekezlet létrehozás	semmilyen	össze-vissza	mindenki	-	<input checked="" type="checkbox"/>
28	alkalmazott	értekezlet létrehozás	pontos	helyes	1	-	<input checked="" type="checkbox"/>
29	alkalmazott	értekezlet létrehozás	pontos	helyes	több	-	<input checked="" type="checkbox"/>
30	alkalmazott	értekezlet létrehozás	pontos	helyes	mindenki	-	<input checked="" type="checkbox"/>
31	alkalmazott	értekezlet létrehozás	pontos	kisebb hibák	1	-	<input checked="" type="checkbox"/>
32	alkalmazott	értekezlet létrehozás	pontos	kisebb hibák	több	-	<input checked="" type="checkbox"/>
33	alkalmazott	értekezlet létrehozás	pontos	kisebb hibák	mindenki	-	<input checked="" type="checkbox"/>
34	alkalmazott	értekezlet létrehozás	pontos	össze-vissza	1	-	<input checked="" type="checkbox"/>
35	alkalmazott	értekezlet létrehozás	pontos	össze-vissza	több	-	<input checked="" type="checkbox"/>
36	alkalmazott	értekezlet létrehozás	pontos	össze-vissza	mindenki	-	<input checked="" type="checkbox"/>
37	alkalmazott	értekezlet létrehozás	hiányos	helyes	1	-	<input checked="" type="checkbox"/>
38	alkalmazott	értekezlet létrehozás	hiányos	helyes	több	-	<input checked="" type="checkbox"/>
39	alkalmazott	értekezlet létrehozás	hiányos	helyes	mindenki	-	<input checked="" type="checkbox"/>
40	alkalmazott	értekezlet létrehozás	hiányos	kisebb hibák	1	-	<input checked="" type="checkbox"/>
41	alkalmazott	értekezlet létrehozás	hiányos	kisebb hibák	több	-	<input checked="" type="checkbox"/>
42	alkalmazott	értekezlet létrehozás	hiányos	kisebb hibák	mindenki	-	<input checked="" type="checkbox"/>
43	alkalmazott	értekezlet létrehozás	hiányos	össze-vissza	1	-	<input checked="" type="checkbox"/>
44	alkalmazott	értekezlet létrehozás	hiányos	össze-vissza	több	-	<input checked="" type="checkbox"/>
45	alkalmazott	értekezlet létrehozás	hiányos	össze-vissza	mindenki	-	<input checked="" type="checkbox"/>
46	alkalmazott	értekezlet létrehozás	semmilyen	helyes	1	-	<input checked="" type="checkbox"/>
47	alkalmazott	értekezlet létrehozás	semmilyen	helyes	több	-	<input checked="" type="checkbox"/>
48	alkalmazott	értekezlet létrehozás	semmilyen	helyes	mindenki	-	<input checked="" type="checkbox"/>
49	alkalmazott	értekezlet létrehozás	semmilyen	kisebb hibák	1	-	<input checked="" type="checkbox"/>
50	alkalmazott	értekezlet létrehozás	semmilyen	kisebb hibák	több	-	<input checked="" type="checkbox"/>
51	alkalmazott	értekezlet létrehozás	semmilyen	kisebb hibák	mindenki	-	<input checked="" type="checkbox"/>
52	alkalmazott	értekezlet létrehozás	semmilyen	össze-vissza	1	-	<input checked="" type="checkbox"/>

53	alkalmazott	értekezlet létrehozás	semmilyen	össze-vissza	több	-	<input checked="" type="checkbox"/>
54	alkalmazott	értekezlet létrehozás	semmilyen	össze-vissza	mindenki	-	<input checked="" type="checkbox"/>
55	menedzser	feladat delegálás	pontos	helyes	-	1	<input checked="" type="checkbox"/>
56	menedzser	feladat delegálás	pontos	helyes	-	több	<input checked="" type="checkbox"/>
57	menedzser	feladat delegálás	pontos	kisebb hibák	-	1	<input checked="" type="checkbox"/>
58	menedzser	feladat delegálás	pontos	kisebb hibák	-	több	<input checked="" type="checkbox"/>
59	menedzser	feladat delegálás	pontos	össze-vissza	-	1	<input checked="" type="checkbox"/>
60	menedzser	feladat delegálás	pontos	össze-vissza	-	több	<input checked="" type="checkbox"/>
61	menedzser	feladat delegálás	hiányos	helyes	-	1	<input checked="" type="checkbox"/>
62	menedzser	feladat delegálás	hiányos	helyes	-	több	<input checked="" type="checkbox"/>
63	menedzser	feladat delegálás	hiányos	kisebb hibák	-	1	<input checked="" type="checkbox"/>
64	menedzser	feladat delegálás	hiányos	kisebb hibák	-	több	<input checked="" type="checkbox"/>
65	menedzser	feladat delegálás	hiányos	össze-vissza	-	1	<input checked="" type="checkbox"/>
66	menedzser	feladat delegálás	hiányos	össze-vissza	-	több	<input checked="" type="checkbox"/>
67	menedzser	feladat delegálás	semmilyen	helyes	-	1	<input checked="" type="checkbox"/>
68	menedzser	feladat delegálás	semmilyen	helyes	-	több	<input checked="" type="checkbox"/>
69	menedzser	feladat delegálás	semmilyen	kisebb hibák	-	1	<input checked="" type="checkbox"/>
70	menedzser	feladat delegálás	semmilyen	kisebb hibák	-	több	<input checked="" type="checkbox"/>
71	menedzser	feladat delegálás	semmilyen	össze-vissza	-	1	<input checked="" type="checkbox"/>
72	menedzser	feladat delegálás	semmilyen	össze-vissza	-	több	<input checked="" type="checkbox"/>
73	alkalmazott	feladat delegálás	pontos	helyes	-	1	<input checked="" type="checkbox"/>
74	alkalmazott	feladat delegálás	pontos	helyes	-	több	<input checked="" type="checkbox"/>
75	alkalmazott	feladat delegálás	pontos	kisebb hibák	-	1	<input checked="" type="checkbox"/>
76	alkalmazott	feladat delegálás	pontos	kisebb hibák	-	több	<input checked="" type="checkbox"/>
77	alkalmazott	feladat delegálás	pontos	össze-vissza	-	1	<input checked="" type="checkbox"/>
78	alkalmazott	feladat delegálás	pontos	össze-vissza	-	több	<input checked="" type="checkbox"/>
79	alkalmazott	feladat delegálás	hiányos	helyes	-	1	<input checked="" type="checkbox"/>
80	alkalmazott	feladat delegálás	hiányos	helyes	-	több	<input checked="" type="checkbox"/>
81	alkalmazott	feladat delegálás	hiányos	kisebb hibák	-	1	<input checked="" type="checkbox"/>

82	alkalmazott	feladat delegálás	hiányos	kisebb hibák	-	több	<input checked="" type="checkbox"/>
83	alkalmazott	feladat delegálás	hiányos	össze-vissza	-	1	<input checked="" type="checkbox"/>
84	alkalmazott	feladat delegálás	hiányos	össze-vissza	-	több	<input checked="" type="checkbox"/>
85	alkalmazott	feladat delegálás	semmilyen	helyes	-	1	<input checked="" type="checkbox"/>
86	alkalmazott	feladat delegálás	semmilyen	helyes	-	több	<input checked="" type="checkbox"/>
87	alkalmazott	feladat delegálás	semmilyen	kisebb hibák	-	1	<input checked="" type="checkbox"/>
88	alkalmazott	feladat delegálás	semmilyen	kisebb hibák	-	több	<input checked="" type="checkbox"/>
89	alkalmazott	feladat delegálás	semmilyen	össze-vissza	-	1	<input checked="" type="checkbox"/>
90	alkalmazott	feladat delegálás	semmilyen	össze-vissza	-	több	<input checked="" type="checkbox"/>
91	menedzser	információ lekérdezés	pontos	helyes	-	-	<input checked="" type="checkbox"/>
92	menedzser	információ lekérdezés	pontos	kisebb hibák	-	-	<input checked="" type="checkbox"/>
93	menedzser	információ lekérdezés	pontos	össze-vissza	-	-	<input checked="" type="checkbox"/>
94	menedzser	információ lekérdezés	hiányos	helyes	-	-	<input checked="" type="checkbox"/>
95	menedzser	információ lekérdezés	hiányos	kisebb hibák	-	-	<input checked="" type="checkbox"/>
96	menedzser	információ lekérdezés	hiányos	össze-vissza	-	-	<input checked="" type="checkbox"/>
97	menedzser	információ lekérdezés	semmilyen	helyes	-	-	<input checked="" type="checkbox"/>
98	menedzser	információ lekérdezés	semmilyen	kisebb hibák	-	-	<input checked="" type="checkbox"/>
99	menedzser	információ lekérdezés	semmilyen	össze-vissza	-	-	<input checked="" type="checkbox"/>
100	alkalmazott	információ lekérdezés	pontos	helyes	-	-	<input checked="" type="checkbox"/>
101	alkalmazott	információ lekérdezés	pontos	kisebb hibák	-	-	<input checked="" type="checkbox"/>
102	alkalmazott	információ lekérdezés	pontos	össze-vissza	-	-	<input checked="" type="checkbox"/>
103	alkalmazott	információ lekérdezés	hiányos	helyes	-	-	<input checked="" type="checkbox"/>
104	alkalmazott	információ lekérdezés	hiányos	kisebb hibák	-	-	<input checked="" type="checkbox"/>
105	alkalmazott	információ lekérdezés	hiányos	össze-vissza	-	-	<input checked="" type="checkbox"/>
106	alkalmazott	információ lekérdezés	semmilyen	helyes	-	-	<input checked="" type="checkbox"/>
107	alkalmazott	információ lekérdezés	semmilyen	kisebb hibák	-	-	<input checked="" type="checkbox"/>
108	alkalmazott	információ lekérdezés	semmilyen	össze-vissza	-	-	<input checked="" type="checkbox"/>

### 3. függelék:

