Dia 1 – Fő oldal

- Köszönés.
- A következőekben szeretnék beszélni az általam készített diplomamunkámról, aminek címe Logikai rendszereket megvalósító szoftverek tesztelése.

Dia 2 – Outline

- A prezentáció során a következő fejezetekben fogom bemutatni a munkámat.
- Először is áttekintem a munkám mögötti hátteret, motivációt és problémát.
- A projekttel szemben támasztott célokat követően az elkészített megoldáshoz kapcsolódó alkalmazást és funkciókat mutatom be.
- És végül az összegzés során ismertetem az eredményeket és további lehetőségeket.

Dia 4 – Introduction

- A technológia fejlődés és technológia során egyre komplexebb alkalmazások jöttek létre, amelyeknek egyre nagyobb biztonságot és megbízhatóságot kell nyújtani.
- Szoftver tesztelés
    - A szoftver tesztelés segítségével bizonyosodhatunk meg az alkalmazás helyessége felől és növelhetjük a pontosságot és a megbízhatóságot.
    - A nem teljes tesztelés rejtett hibákat eredményezhet.
- Cause-Effect
    - Jól képesek modellezni az üzleti logikát és vizualizálni az ok-okozati kapcsolatok.
    - Viszont a tesztek készítése a gráfból többnyire manuális munka, ami nem hatékony, sok hibát tartalmazhat és nehezen skálázható.

Dia 5 – Problem Statment

- Az üzleti logika pontosságának és helyességének biztosítása és teljes tesztelés végzése kritikus és nehéz feladat.
- A cause-effect gráfok jól strukturáltak és könnyen érthetőek viszont nehéz végrehajtható tesztesetekké alakítani őket.

Dia 6 – Objectives

- A célja a diplomamunkának, hogy egy alkalmazás készüljön, ami képes feldolgozni egy specializált szöveges nyelven írt bementet gráf modellé, logikai formulákká, döntési táblázattá és végül egy tesztelésre felhasználható GPT kimenetté formázni.

Dia 8, 9 - Methodology

- A konkrét megoldás során a felhasználó definiálhatja a cause-effect gráf struktúráját szöveges formában és ezt modellé majd végül egy tesztesetek generálására alkalmas formába hozni, néhány közbelső lépést végre hajtva.
- Mindezt minimális manuális közbelépéssel támogatva nagyobb és komplexebb gráfokat is.

Dia 10, 11 – Application Arch.

- Az alkalmazás architektúrája egy a kliens oldalból és egy szerver oldalból épül fel.
- A kliens a moduláris React interfészen intuitív gráf vizualizációra és valós idejű gráf szintaxis ellenőrzésre képes, hiba visszajelzéssel.
- A szerver oldal hivatott az alkalmazás üzleti logikáját biztosítani, kezeli a gráf szabályokat és az átalakításokat.
- Telepíthetőséget támogatja a konténer struktúra is a felállított CI/CD lehetőségek.

Dia 12 – Core Features

- Több főbb funkció közül a legfontosabb és legnagyobb csomag a gráfok definíciója és annak feldolgozása és átalakítása.
    - A speciális Kotlin DSL alapú nyelven definiált szöveg feldolgozásra kerül gráf modell formájában majd ez lesz átalakítva logikai formulákká és végül pedig egy döntési táblává.
- Továbbá az alkalmazás lehetőséget biztosit gráf vizualizációra és a teszteset generálásra alkalmas kimenet biztosításra.

Dia 14 – Results

- Teljesítmény hatékonyság
    - Hatékonyság még nagyobb adathalmaz esetén is
- Transzformáció hatékonyság
    - Megbízható átalakítás komplex szabályokhoz
- Skálázhatóság és válasz
    - Nagyobb felhasználó konkurencia kezelése minimális teljesítmény leeséssel
- Használhatóság
    - Intuitív felhasználó felület, hiba visszajelzésekkel
    - Könnyű gráf definíció és gyors eredmény visszajelzés

Dia 15 – Limitations

- Nincsen adattárolás
- Szerver skálázása nagyobb felhasználó bázis kiszolgálásra
- Fejlettebb nyelvi szerver integráció szemantikai ellenőrzéshez

Dia 16 – Conclusion and Future Work

- Sikeres eszköz cause-effect gráf készítésre és tesztelésre
- Validált funkcionalitás
- Biztos alap a jövőbeli fejlesztésre és bővítésre
    - Többek között perzisztencia, felhasználó kezelés, munkamenet kezelés, fejlesztett szerkesztő funkciók, illetve integráció külső teszt eszközökkel.

Dia 17 – Questions

- Köszönöm a figyelmet és várom a felmerülő kérdéseket.