Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Využití umělé inteligence při vytváření testovacích scénářů pro manuální testování webových aplikací

Jakub Balšánek

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jakub Balšánek

Informatika

Název práce

Využití umělé inteligence při vytváření testovacích scénářů pro manuální testování webových aplikacích

Název anglicky

Leveraging Artificial Intelligence in the Generation of Test Cases for Manual Testing of Web Applications.

Cíle práce

Bakalářská práce je tematicky zaměřena na problematiku využití umělé inteligence v oblasti tvorbě testů. Hlavním cíle práce je zhodnocení možností využití umělé inteligence pro psaní testovacích scénářů pro manuální testování webových aplikacích
Dílčí cíle jsou:
- Charakterizovat problematiku testování webových stránek a umělé inteligence.
- Analyzovat vhodné postupy pro tvorbu testovacích scénářů pomocí umělé inteligence.
- Navrhnout doporučení pro využití umělé inteligence při psaní testovacích scénářů.

Metodika

V teoretické části práce budou studiem literatury získány potřebné znalostí pro zhotovení samotné práce.

V praktické části bude vytvořena analýza současného stavu testování webových aplikací a identifikace problémů spojených s manuálním vytvářením testovacích scénářů. Poté bude navrhnuta a implementace webové stánky pro provedení testů. V následujícím kroku budou provedeny experimenty, spočívající v zhodnoceny schopnosti umělé inteligence odhalit chyby na testovacích scénářích. Následně budou zhodnoceny úspory při vytváření testovacích scénářů pomocí umělé inteligence a tradičních metod. Na základě syntézy teoretických poznatků a výsledků praktické časti budou formulovány závěry práce.

Doporučený rozsah práce

30–40 stran

Klíčová slova

Umělá inteligence, Testování webových aplikací, Testovací scénáře, Automatické generování scénářů

Doporučené zdroje informací

BAHRINI, Aram, Mohammadsadra KHAMOSHIFAR, Hossein ABBASIMEHR, Robert J. RIGGS, Maryam ESMAEILI, Rastin Mastali MAJDABADKOHNE a Morteza PASEHVAR. ChatGPT: Applications, Opportunities, and Threats. In: 2023 Systems and Information Engineering Design Symposium (SIEDS) [online]. IEEE, 2023, 2023-4-27, s. 274-279. ISBN 979-8-3503-0064-2. Dostupné z: doi:10.1109/SIEDS58326.2023.10137850

FONTES, Afonso a Gregory GAY. The integration of machine learning into automated test generation: A systematic mapping study. Software Testing, Verification and Reliability [online]. 2023, 33(4) 0960-0833. Dostupné z: doi:10.1002/stvr.1845

IOANNIDES, Charalambos a Kerstin I. EDER. Coverage-Directed Test Generation Automated by Machine Learning – A Review. ACM Transactions on Design Automation of Electronic Systems [online]. 2012, 17(1), 1-21 ISSN 1084-4309. Dostupné z: doi:10.1145/2071356.2071363

JHA, Nisha a Rashmi POPLI. Artificial Intelligence For Software Testing-Perspectives And Practices. In: 2021 Fourth International Conference on Computational Intelligence and Communication Technologies (CCICT) [online]. IEEE, 2021, 2021, s. 377-382. ISBN 978-1-6654-2392-2. Dostupné z: doi:10.1109/CCICT53244.2021.00075

PATTON, Ron. Testování softwaru. Praha: Computer Press, 2002. Programování. ISBN 8072266365.

Předběžný termín obhajoby

2023/24 LS – PEF

Vedoucí práce

Ing. Michal Stočes, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 4. 7. 2023

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 3. 11. 2023

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 15. 03. 2024

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci „Využití umělé inteligence při vytváření testovacích scénářů pro manuální testování webových aplikací“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3.2024 *Jakub Balšánek*

Poděkování

Rád bych touto cestou poděkoval svému vedoucímu práce panu doktorovi Michalu Stočesovi za jeho rady a připomínky, které mi pomohly k vypracování této práce. Dále bych chtěl poděkovat své rodině za podporu, kterou mi poskytla.

Využití umělé inteligence při vytváření testovacích scénářů pro manuální testování webových aplikací

Abstrakt

Tato bakalářská práce se zaměřuje na využití modelu GPT-4.0 od OpenAI pro automatizované generování testovacích scénářů softwarových aplikací. Cílem bylo prozkoumat potenciál a omezení použití pokročilých modelů umělé inteligence v kontextu testování softwaru. Experimentální část práce prokázala, že GPT-4.0 je schopen efektivně identifikovat a generovat testovací scénáře, což vede k významným časovým a ekonomickým úsporám v procesu testování. Nicméně, práce také odhalila určité omezení v detekci chyb, což zdůrazňuje nutnost kombinace umělé inteligence s lidskými testovacími postupy pro zajištění komplexního pokrytí a kvality softwaru. Výsledky práce poukazují na významný potenciál integrace AI do softwarového testování a zdůrazňují důležitost dalšího výzkumu a vývoje v této oblasti.

Klíčová slova: umělá inteligence, testování softwaru, GPT-4.0, generování testovacích scénářů, automatizace testování, kvalita softwaru, detekce chyb, umělé neuronové sítě, testovací strategie, efektivita testování

Leveraging artifficial intelligence in the generation of test cases for manual testing of web applications

Abstract

This bachelor thesis focuses on the use of the GPT-4.0 model by OpenAI for the automated generation of test scenarios for software applications. The aim was to explore the potential and limitations of using advanced artificial intelligence models in the context of software testing. The experimental part of the thesis demonstrated that GPT-4.0 is capable of efficiently identifying and generating test scenarios, leading to significant time and economic savings in the testing process. However, the thesis also revealed certain limitations in error detection, highlighting the need for a combination of artificial intelligence with human testing procedures to ensure comprehensive coverage and software quality. The results of the thesis point to the significant potential of integrating AI into software testing and emphasize the importance of further research and development in this area.

Keywords: artificial intelligence, software testing, GPT-4.0, test scenario generation, testing automation, software quality, error detection, artificial neural networks, testing strategies, testing efficiency

Obsah

1 Úvod

V dnešní době, kdy se technologie vyvíjí bleskovou rychlostí a softwarové aplikace se stávají stále složitějšími, se kvalita a efektivita testování software ukazují jako klíčové faktory určující úspěch nebo neúspěch technologických produktů. V této bakalářské práci se zaměřuji na revoluční přístup k testování software, který využívá umělou inteligenci, k automatickému generování testovacích scénářů. Tento přístup představuje potenciální paradigmatickou změnu v metodologiích softwarového testování, která může zásadně ovlivnit jak rychlost, tak kvalitu procesu vývoje software.

Cílem této práce je nejen zkoumat teoretické podklady a technologie umělé inteligence aplikované na testování, ale také prakticky evaluovat efektivitu a účinnost velkých jazykových modelů v generování testovacích scénářů na příkladu webové aplikace GlobeTrotter. GlobeTrotter je fiktivní platforma pro rezervaci cestovních služeb, která slouží jako modelový příklad k demonstraci potenciálu a výzev spojených s použitím AI v testování.

Práce je rozdělena do několika klíčových kapitol. Po úvodním přehledu problematiky testování software a role umělé inteligence v tomto procesu se věnuji detailnímu popisu jednotlivých modelů, jejich architektury a možností aplikace v generování scénářů. Také je Poté je provedena vícekriteriální analýza jednotlivých modelů a výběr toho nejlepšího kandidáta GTP 4.0. Dále je představena webová aplikace GlobeTrotter, včetně specifikace jejích funkcí a potenciálních chyb, které mají být detekovány.

V experimentální části je popsán proces generování testovacích scénářů s využitím GPT-4.0, včetně analýzy dosažených výsledků, identifikace odhalených chyb a hodnocení úspěšnosti modelu v detekci různých typů chyb. Zvláštní pozornost je věnována také diskuzi o výzvách a omezeních spojených s použitím AI v testování, stejně jako možnostem dalšího výzkumu a vývoje v této oblasti.

Závěrem této práce reflektuji nad zjištěními a dopadu těchto technologií na budoucí praxi testování software. S přihlédnutím k dynamickému vývoji v oblasti umělé inteligence a automatizace testování zdůrazňuji potřebu kontinuálního vzdělávání a adaptace profesionálů v oblasti testování software, aby mohli plně využít potenciál těchto inovativních nástrojů.

2 Cíl práce a metodika

2.1 Cíl práce

Bakalářská práce je tematicky zaměřena na problematiku využití umělé inteligence v oblasti tvorbě testů. Hlavním cíle práce je zhodnocení možností využití umělé inteligence pro psaní testovacích scénářů pro manuální testování webových aplikacích

Dílčí cíle jsou:
- Charakterizovat problematiku testování webových stránek a umělé inteligence.
- Analyzovat vhodné postupy pro tvorbu testovacích scénářů pomocí umělé inteligence.
- Navrhnout doporučení pro využití umělé inteligence při psaní testovacích scénářů

2.2 Metodika

V teoretické části práce budou studiem literatury získány potřebné znalostí pro zhotovení samotné práce. V teoretické části bude vytvořena analýza současného stavu testování webových aplikací a identifikace problémů spojených s manuálním vytvářením testovacích scénářů. Poté bude navrhnuta implementace webové stánky pro provedení testů. Pomocí vícekriteriální analýzy bude vybrán nejlepší model. V následujícím kroku budou provedeny experimenty, spočívající v zhodnoceny schopnosti umělé inteligence odhalit chyby na testovacích scénářích. Následně budou zhodnoceny úspory při vytváření testovacích scénářů pomocí umělé inteligence a tradičních metod. Na základě syntézy teoretických poznatků a výsledků praktické časti budou formulovány závěry práce.

3 Teoretická východiska

3.1 Základy testování

Testování softwaru je klíčovým prvkem vývojového cyklu softwaru, zajišťujícím, že výsledný produkt splňuje specifikované požadavky a je bez chyb. Tento proces nejen odhaluje přítomnost chyb v softwaru, ale také ověřuje a potvrzuje kvalitu softwarového produktu z hlediska funkčnosti, výkonnosti, bezpečnosti a dalších kritických faktorů. (Patton, 2001)

3.1.1 Základní principy a terminologie

V oblasti testování softwaru je zásadní porozumět klíčovým pojmům a terminologii, které definují a formují procesy zajištění kvality softwarových produktů. Mezi tyto základní pojmy patří testovací scénáře, testovací případy, chyby a selhání, které společně tvoří základ pro efektivní testování a analýzu softwaru.

Testovací scénáře (test scenario)

Podle Jöckelové (2021) jsou testovací scénáře vysokoúrovňové popisy toho, co má být testováno, a slouží jako vodítko pro vytváření detailnějších testovacích případů. Scénáře obvykle obsahují sérii kroků nebo operací, které simulují uživatelské interakce s aplikací nebo systémové procesy, s cílem ověřit specifické funkcionality nebo chování softwaru. Typicky vycházejí z případů užití, ale nemusí to být nutně pravidlo.

Testovací případy (test case)

Detailní specifikace vstupů, prováděcích podmínek, testovacích kroků a očekávaných výsledků, které jsou navrženy tak, aby systematicky ověřily funkčnost a výkon softwaru pod konkrétními podmínkami. Každý testovací případ je zaměřen na konkrétní aspekt softwaru, ať už jde o funkci, výkon, bezpečnostní aspekt nebo jinou kvalitativní charakteristiku.

Testovací scénáře a případy jsou generovány na základě požadavků a specifikací softwaru, což zahrnuje funkční specifikace, uživatelské příběhy, designové dokumenty a další technické materiály. Tento proces vyžaduje důkladné porozumění softwarovému produktu a jeho očekávanému chování v různých situacích. (Jöckel *et al.*, 2021)

Pokrytí testů (test coverage)

Metrika, která se používá v softwarovém testování k určení, do jaké míry testovací sada pokrývá kód aplikace nebo softwaru. Jedná se o důležitý ukazatel kvality softwarového testování, protože poskytuje přehled o tom, které části kódu byly testovány a které ne. Cílem je dosáhnout co nejvyššího pokrytí testů, aby se minimalizovalo riziko, že v softwaru zůstanou neodhalené chyby. Pokrytí testů může být měřeno různými způsoby, například pokrytím kódu (měří, kolik kódu je spuštěno během testů), pokrytím funkčnosti (zda testy pokrývají všechny funkční požadavky) nebo pokrytím větví (zda testy pokrývají všechny možné větve výkonného kódu). Vysoce kvalitní testovací sady, které dosahují vysokého stupně pokrytí testů, jsou klíčem k odhalení chyb v raných fázích vývoje softwaru a přispívají k větší spolehlivosti a bezpečnosti softwarových produktů. (Lee, Kang, Jung, 2020)

Chyba (defekt)

Termín používaný k označení jakékoli nesrovnalosti nebo odchylky od očekávaného chování softwaru, které může být způsobeno chybami v kódu, designu, konfiguraci nebo jiných aspektech softwarového vývoje. Identifikace a oprava chyb je klíčovým cílem testovacího procesu. (tutorialspoint)

Selhání (failure)

Selhání se vyskytuje, když software nedokáže provést požadovanou funkci nebo operaci podle specifikací. Selhání je obvykle výsledkem jedné nebo více chyb v softwaru a může mít různou závažnost, od drobných problémů po kritické chyby ovlivňující celý systém. (tutorialspoint)

3.1.2 Základy testování webových stránek

V digitálním věku, kdy internetové stránky představují klíčový prvek v komunikaci, marketingu a obchodu, je testování webových stránek nezbytné pro zajištění jejich funkčnosti, uživatelské přívětivosti a bezpečnosti. Tato část, podle Pattona (2001), se zaměřuje na základní principy a metody testování webových stránek, které jsou nezbytné pro vývojáře, testery a všechny, kdo se podílejí na vytváření a správě webového obsahu.

Základní prvky webových stránek

Webové stránky jsou v zásadě multimediální dokumenty obsahující text, obrázky, zvuky, videa a hypertextové odkazy. Umožňují uživatelům navigovat klikáním na odkazy, vyhledávat informace a interagovat s různými formami obsahu. Dvě klíčové vlastnosti, které internetové stránky odlišují od statických multimediálních dokumentů, jsou jejich schopnost propojovat uživatele s informacemi po celém světě a snadnost, s jakou může průměrný člověk vytvořit základní webovou stránku.

Testování webových stránek

Testování webových stránek je komplexní proces, který zahrnuje ověřování různých aspektů webové stránky, od funkčnosti a výkonu po uživatelskou přívětivost a bezpečnost. Testeři by měli přistupovat k testování s tzv. "tester mentality", což znamená aktivní hledání chyb a potenciálních problémů, které by mohly uživatele odradit nebo ohrozit bezpečnost webové stránky.

Black-box testing

Black-box testování je metoda, při které tester nezná vnitřní strukturu testovaného systému. Při testování webových stránek se tester zaměřuje na interakci uživatele s webovou stránkou, aniž by měl přístup k jejímu kódu. Tento přístup umožňuje identifikovat chyby v uživatelském rozhraní a funkčnosti webové stránky.

Funkční a nefunkční testování

Funkční testování se zaměřuje na ověření, zda webová stránka splňuje specifikované požadavky a zda všechny její funkce pracují podle očekávání. Nefunkční testování naopak hodnotí aspekty jako je výkon, bezpečnost, kompatibilita s různými prohlížeči a zařízeními, a uživatelská přívětivost.

Kompatibilita a konfigurace

Jedním z klíčových aspektů testování webových stránek je zajištění jejich správné funkčnosti a vzhledu na různých zařízeních, v různých prohlížečích a na různých operačních systémech. To zahrnuje testování na mobilních telefonech, tabletech, desktopových počítačích a v různých verzích prohlížečů.

3.1.3 Nejčastější chyby v softwaru

Funkční chyby

Funkční chyba v softwaru nastává, když aplikace nebo webová stránka nefunguje podle očekávání nebo specifikací. Jedná se o situace, kde softwarové funkce selhávají v plnění svých předem definovaných úkolů. To může zahrnovat širokou škálu problémů, od nereagujících tlačítek a formulářů, které nejsou správně odeslány, po chyby v logice

vyhledávání nebo neočekávané pády aplikace. Tyto problémy mohou negativně ovlivnit uživatelskou zkušenost a důvěru v produkt. (test.io)

Četnost funkčních chyb

Funkční chyby představují nejvýznamnější část, tvoří 41,3 % všech analyzovaných problémů. Catolino (et al., 2019) tvrdí, že chyby jsou přímým důsledkem implementace nových funkcí nebo úprav existujících, což je v souladu s očekáváním, že většina problémů vzniká právě v této fázi vývoje softwaru.

Typické příklady funkčních chyb

Příklady funkčních chyb mohou zahrnovat, ale nejsou omezeny na:

1. Tlačítko, které neodesílá formulář **-** Uživatel očekává, že po kliknutí na tlačítko dojde k odeslání dat formuláře, ale nic se nestane.
2. Vyhledávání nereaguje na uživatelský vstup - Při pokusu o vyhledávání aplikace nevrací žádné výsledky nebo nereaguje.
3. Neočekávané ukončení aplikace - Aplikace se bez varování nebo zjevného důvodu zavře. (test.io)

3.1.4 GUI související problémy – vizuální chyby

V této kapitole podle Catolino (et al., 2019) jsou vizuální chyby problémy spojené s grafickým uživatelským rozhraním, které mohou negativně ovlivnit estetický dojem a intuitivnost aplikací nebo webových stránek. Vizuální chyba se stává funkční chybou v okamžiku, kdy zasáhne do funkčnosti softwaru nebo aplikace, i když samotná funkce není defektní. Klíčovým aspektem pro identifikaci tohoto přechodu je rozpoznání, že vizuální problém brání uživatelům v plnohodnotném využívání produktu.

Četnost vizuálních chyb

GUI související problémy tvoří 17 % všech analyzovaných případů, což je důsledek rostoucí závislosti na grafických uživatelských rozhraních v mnoha softwarových systémech. S narůstající komplexností GUI a jeho interakcemi s uživateli se zvyšuje pravděpodobnost vzniku chyb. Tento fakt podtrhuje význam zaměření na pečlivé testování a návrh GUI, aby bylo zajištěno bezproblémové fungování aplikací a kvalitní uživatelská zkušenost.

Typické příklady vizuálních chyb

1. Problémy s rozložením: například špatně zarovnané texty nebo elementy.
2. Problémy s responzivním designem: element se zobrazuje na jednom mobilním zařízení, ale na jiném nikoliv.
3. Překryvy textů nebo elementů: když se objekty na stránce nechtěně překrývají.
4. Ořezané texty nebo elementy: části obsahu jsou neviditelné nebo nedostupné kvůli špatnému umístění nebo rozměrům.

3.1.5 Konfigurační chyby

Problémy s konfigurací odkazují na chyby spojené s nastavením konfiguračních souborů v softwarových projektech. Tyto chyby jsou často způsobeny dvěma hlavními faktory:

1. Problémy s externími knihovnami – Tyto problémy nastávají, když externí knihovny, na které se software spoléhá, vyžadují aktualizaci nebo opravu kvůli nekompatibilitě nebo zastaralosti.

2. Chybné cesty k adresářům nebo souborům – Chyby v definici cest k souborům nebo adresářům v XML nebo manifestových artefaktech mohou způsobit, že software nebude schopen najít potřebné zdroje nebo závislosti.

Četnost konfiguračních chyb

Konfigurační chyby 16 % celkového počtu chyb, které byly identifikovány v softwarových projektech. Také poukazuje na to, že tyto chyby souvisí s nevhodnou konfigurací aplikací, používáním externích knihoven a API. S rostoucím využíváním těchto externích zdrojů se zvyšuje frekvence konfiguračních chyb, což zdůrazňuje potřebu správného nastavení a pravidelných aktualizací konfiguračních souborů a závislostí, aby byla zajištěna hladká funkcionalita aplikací.

3.1.6 Testovací kód

V rámci softwarového inženýrství je nezbytné věnovat pozornost nejen chybám v produkčním kódu, ale i problémům, které se vyskytují v testovacím kódu. Chyby v testovacím kódu se typicky týkají několika oblastí:

1. Problémy s prováděním, opravou nebo aktualizací testovacích případů - Tyto problémy mohou zahrnovat chyby v logice testů, nesprávné použití testovacích frameworků nebo potřebu aktualizace testů kvůli změnám v produkčním kódu.

2. Intermitentní testy - Testy, které selhávají nekonzistentně pod různými podmínkami nebo při různých spuštěních, což komplikuje diagnostiku a opravy chyb.

3. Neschopnost testu najít lokalizované chyby - Situace, kdy testy nedokážou odhalit chyby v kódu kvůli jejich neefektivnímu návrhu nebo špatnému pokrytí kódu.

Četnost testovacího kódu

Chyby v testovacím kódu představují 7 % všech identifikovaných problémů, což podtrhuje důležitost efektivního testování pro minimalizaci výskytu těchto chyb. Efektivní testovací strategie a postupy jsou klíčové pro zajištění, že software bude fungovat podle očekávání a že budou včas odhaleny a opraveny jakékoli potenciální problémy. Tímto způsobem lze předejít mnoha problémům, které by jinak mohly uživatele frustrovat a negativně ovlivnit jejich zkušenost s produktem.

3.1.7 Výkonnostní problémy

Tyto problémy zahrnují širokou škálu defektů, od nadměrného využití paměti a úniků energie až po metody způsobující nekonečné smyčky. Tyto chyby mohou významně ovlivnit rychlost, dobu reakce, stabilitu a spotřebu zdrojů softwaru, což negativně působí na uživatelskou spokojenost a celkovou efektivitu softwarových řešení. (Polyuno, 2021)

Četnost výkonnostních problémů

Výkonnostní problémy, tvořící 4 % všech zjištěných problémů, i když nejsou mezi nejčastějšími, mohou mít značný vliv na spokojenost uživatelů a celkovou efektivitu softwaru. Catalino zdůrazňuje význam sledování a optimalizace výkonnosti aplikací, protože dokonce i malé zlepšení může vést k výraznému zvýšení uživatelské spokojenosti a výkonnosti aplikace. V tomto kontextu je nezbytné věnovat pečlivou pozornost nejen funkčnosti, ale i výkonnosti softwaru, aby se zajistilo, že koneční uživatelé mají kvalitní a plynulé uživatelské zážitky. (Catolino, et al., 2019)

3.2 Úvod do automatizace testování

V neustále se vyvíjející krajině vývoje softwaru se efektivita a spolehlivost dodávaných produktů staly nejvyššími prioritami. Uprostřed mnoha etap, které tvoří životní cyklus vývoje softwaru (SDLC), se testování jeví jako klíčová fáze, zajišťující, že konečný produkt splňuje požadované normy a funguje podle zamýšlení. Tradičně tento proces značně spoléhal na manuální metodiky testování, při nichž testery pečlivě provádějí testy případ po případu. Avšak s rostoucí složitostí a rozsahem softwarových aplikací se stávají omezení manuálního testování stále zřetelnějšími. Tento scénář otevírá cestu pro nástup automatizace testování transformačního přístupu, který je připraven předefinovat paradigma zajištění kvality softwaru.

Automatizace testování se odkazuje na použití specializovaných softwarových nástrojů k provedení předem naprogramovaných testů na softwarové aplikaci před jejím uvedením do produkce. Tento inovativní přístup si klade za cíl zefektivnit proces testování, významně snížit čas a úsilí vyžadované pro vyčerpávající manuální testování. Automatizované testování nejen urychluje cyklus testování, ale také zvyšuje přesnost výsledků testů, eliminuje potenciál pro lidské chyby, který je vlastní manuálním testovacím praktikám. Schopnost opakovaně spouštět testy kdykoliv během dne poskytuje úroveň flexibility a efektivity, kterou manuální testování jednoduše nemůže dosáhnout. (Sharma, Ango, 2014)

3.2.1 Automatizace testování a umělá inteligence

Začlenění umělé inteligence (AI) do automatizace testování softwaru představuje významný skok vpřed v evoluci praktik softwarového inženýrství. Jak digitální prostředí neustále expanduje, s aplikacemi zasahujícími do každého aspektu našich životů od domácích spotřebičů po sofistikovaná rozhraní virtuální reality, poptávka po rychlých, efektivních a spolehlivých metodikách testování softwaru nikdy nebyla kritičtější.

Úvod do AI v testování softwaru

Tempo, jakým technologie přetváří naše profesní a osobní prostředí, je ohromující, a vyžaduje si stálou adaptaci, aby bylo možné držet krok s pokroky. V tomto kontextu se testování softwaru jeví jako klíčová fáze v cyklu vývoje softwaru, zajišťující, že aplikace fungují bezchybně na různých platformách a zařízeních, což nakonec zlepšuje zážitek koncového uživatele. Avšak tradiční metody manuálního testování jsou stále méně schopné vyhovět požadavkům dnešních rychlých cyklů dodávání softwaru, charakterizovaných častými vydáními a aktualizacemi. Tato mezera otevřela cestu pro přijetí automatizace testování, ještě více posílené schopnostmi umělé inteligence (AI) a strojového učení (ML).
AI v testování softwaru překračuje omezení konvenční automatizace tím, že zavádí schopnost učit se z dat bez explicitního programování. Tato schopnost umožňuje nástrojům pro automatizaci testování řízeným AI identifikovat vzory, dělat předpovědi a s časem zlepšovat testovací strategie, což významně zvyšuje efektivitu a účinnost testovacího procesu.

Evoluce od manuálního k automatizovanému testování s podporou AI

Cesta od manuálního k automatizovanému testování, a nyní k automatizaci testování s podporou AI, odráží neustálé hledání efektivnějších, přesnějších a nákladově efektivnějších metodik testování. Selenium, uznávaný jako jeden z průkopnických nástrojů v automatizaci testování, označuje důležitý milník v této evoluci. Avšak příchod AI do

automatizace testování nás přiblížil k dosažení konečného cíle testování softwaru: zajištění vysoké kvality softwaru s minimálním lidským zásahem.
Úloha AI v automatizaci testování softwaru je mnohostranná, řeší nejen provedení testů, ale také údržbu, která byla historicky významnou výzvou. Jak se zvyšuje složitost softwaru, roste i zátěž spojená s udržováním komplexního a účinného souboru testů. Algoritmy AI a ML nabízejí řešení automatizací generování a údržby testovacích případů, snižují čas a úsilí potřebné k udržení testovacích sad aktuálních a relevantních.

Dopad AI na efektivitu a kvalitu testování

Začlenění AI do nástrojů pro testování softwaru učinilo cyklus vývoje softwaru snadněji zvládnutelným, automatizuje při tom nudné a opakující se úkoly, které tradičně vyžadovaly manuální úsilí. Tato automatizace se rozšiřuje daleko za jednoduché provedení testů, zahrnuje generování testovacích případů, optimalizaci testovacích procesů a dokonce i prediktivní analýzu pro předvídání potenciálních problémů, než vzniknou.

Jednou z klíčových výhod AI v automatizaci testování softwaru je její schopnost zvýšit přesnost. Automatizované testy, poháněné AI, mohou provádět opakující se úkoly s vysokou přesností, zajišťující konzistentní a spolehlivé výsledky testů. Tato přesnost, spolu se schopností provádět vyčerpávající testování nad rámec možností manuálního úsilí, významně zlepšuje celkovou kvalitu softwarového produktu.

Budoucí trendy a vývoj v automatizaci testování s AI

S výhledem do budoucna je začlenění AI do testování softwaru připraveno předefinovat krajinu vývoje a zajištění kvality softwaru. Jak se technologie AI vyvíjí, můžeme očekávat sofistikovanější aplikace AI v testování, včetně schopnosti testovat, diagnostikovat a dokonce opravovat software autonomně. Tato vize plně automatizovaného, inteligentního testování není jen možností, ale nevyhnutelným pokrokem v zajištění kvality softwaru. (Battina, 2019)

3.2.2 Generování testovacích scénářů pomocí umělé inteligence LLM

Rozsáhlé jazykové modely (LLM) jsou typem umělé inteligence, která dokáže generovat text podobný lidskému. Tyto modely jsou trénovány na obrovských množstvích dat a mohou být využity pro řadu úkolů, včetně překladu jazyka, sumarizace textu a generování textu. V posledních letech byly LLM využívány pro generování testovacích scénářů, které jsou používány pro testování softwarových aplikací. (Wei, 2022)

Přínosy testovacích scénářů generovaných LLM

Generování testovacích scénářů pomocí rozsáhlých jazykových modelů (LLM) přináší značné výhody v oblasti softwarového inženýrství, zejména z hlediska efektivity a adaptability. Díky schopnosti LLM rychle zpracovávat a generovat textový obsah mohou tyto modely produkovat komplexní sady testovacích scénářů, které pokrývají širokou škálu situací, od běžně používaných funkcí po méně časté okrajové případy. Tato rychlost a rozmanitost generování umožňuje týmům urychlit proces testování a zároveň zajistit, že testy jsou komplexní a pokrývají potenciálně problematické oblasti, které by mohly být při manuálním vytváření scénářů přehlédnuty.
Kromě toho LLM nabízí možnost personalizace testovacích scénářů pro specifické testovací požadavky. Týmy mohou modely instruovat, aby se zaměřily na konkrétní funkční oblasti nebo uživatelské scénáře, což umožňuje detailnější a cílenější testování. Tato schopnost adaptace je obzvláště cenná v projektech, které vyžadují specifické testování pro unikátní nebo složité funkce.

Omezení testovacích scénářů generovaných LLM

Přestože využití LLM pro generování testovacích scénářů nabízí řadu přínosů, je důležité si být vědom také potenciálních omezení. Jedním z hlavních problémů je, že tyto modely mohou někdy generovat scénáře, které nejsou zcela relevantní nebo realistické vzhledem k testovanému softwaru. Může to být způsobeno omezením v trénovacích datech, na kterých byl model vyvinut, což může vést k neschopnosti správně interpretovat specifické požadavky nebo kontext aplikace.

Dalším omezením je, že scénáře vygenerované LLM mohou někdy postrádat hloubku a komplexnost. Ačkoli modely mohou identifikovat širokou škálu testovacích situací, mohou se zaměřit spíše na povrchové aspekty než na hlubší logické nebo funkční vazby. Toto omezení může ovlivnit schopnost odhalit sofistikovanější chyby nebo problémy, které vyžadují komplexní interakce systémových komponent. (Codemify, 2023)

3.3 Integrace do mé práce

V rámci mé bakalářské práce budou tyto poznatky ověřeny a rozšířeny. Zaměřím se na praktické nasazení LLM pro generování testovacích scénářů a prozkoumám, jak efektivně lze tyto modely využít ve skutečných testovacích procesech. Budu zkoumat, do jaké míry mohou LLM generovat relevantní a komplexní testovací scénáře, a jak mohou tato omezení ovlivnit kvalitu a spolehlivost softwarového testování. Tímto přístupem poskytnu hlubší vhled do potenciálu a výzev spojených s využíváním LLM v procesu testování softwaru.

3.4 Umělá inteligence

Umělá inteligence je oblastí informatiky, která se primárně zaměřuje na přenášení antropomorfní inteligence a myšlení do strojů, jež mohou na mnoha úrovních asistovat lidem. Termín umělá inteligence poprvé použil John McCarthy v roce 1956. Od té doby se AI postupně rozvíjela a získávala na síle v mnoha oblastech, jako jsou inženýrství, matematika, fyzika a technologie, což vše vedlo k současnému obrovskému posunu v tomto oboru, kterého jsme nyní svědky.

Tato myšlenka navrhuje, že stroje mohou získat inteligenci. Zahrnuje oblasti, jako je schopnost strojů učit se samostatně, přizpůsobit se konkrétním okolnostem a opravit vlastní chyby. Jinými slovy, stroje mohou myslet samy o sobě bez nutnosti být naprogramovány specifickými příkazy. (Karthikeyan, 2021)

3.4.1 Princip fungování velkých jazykových modelů LLM

Velké jazykové modely (LLM), jako je GPT (Generative Pre-trained Transformer), představují jednu z nejvýznamnějších a nejpokročilejších technologií v oblasti umělé inteligence (AI) a strojového učení. Tyto modely jsou schopné generovat text, který je nápadně podobný textu psanému člověkem, a nacházejí uplatnění v široké škále aplikací, od automatického generování textu po překlad jazyka a mnoho dalších. Klíčem k jejich úspěchu je kombinace pokročilých technik strojového učení, obrovských datasetů a výpočetní síly.

3.4.2 Architektura neuronových sítí

Jazykové modely využívají specifický typ neuronové sítě známý jako transformátory. Tyto sítě jsou navrženy tak, aby efektivně zpracovávaly sekvence dat, jako je text, a umožňovaly modelu získat hluboké porozumění kontextu a významu slov v rámci

dlouhých textů. Transformátory dosahují tohoto porozumění pomocí mechanismů pozornosti, které modelu umožňují vážit význam různých částí textu při generování odpovědí. (Vaswani, el al., 2017)

3.4.3 Trénování a učení

Jazykové modely jsou trénovány na obrovských datasetech textu získaných z internetu, knih, článků a dalších zdrojů. Během trénovacího procesu se model učí rozpoznávat vzorce v jazyce, včetně gramatiky, slovní zásoby a stylistických prvků. Trénování modelu probíhá pomocí techniky zvané zpětná propagace, kde model neustále upravuje své interní parametry na základě chyb vygenerovaných ve výstupu, aby se jeho predikce co nejvíce přiblížily skutečným datům. (Goodfellow, et al., 2016)

3.4.4 Generování textu

Po trénování je podle Radforda (et al., 2019) model schopen generovat text na základě vstupních podnětů, které dostává. To dělá tak, že na základě kontextu poskytnutého ve vstupním textu vybírá nejpravděpodobnější slovo nebo frázi, která by měla následovat. Tento proces se opakuje pro každé nové slovo v sekvenci, čímž model generuje soudržný a relevantní text odpovídající zadání.

3.5 Chat-GPT 4.0

Vývoj GPT-4 představuje zásadní milník v oblasti umělé inteligence. Jako multimodální model schopný zpracovávat obrazy a texty a produkovat textové výstupy, GPT-4 ukazuje výkony srovnatelné s lidskými na různých profesionálních a akademických benchmarkách. Překonává předchozí modely, včetně GPT-3.5, a dosahuje skóre v top 10% testovaných na simulované advokátní zkoušce. Tento model, založený na architektuře transformátorů, byl před trénován s cílem předpovídat další token v dokumentu.

Přestože GPT-4 překonává existující modely a v mnoha případech i systémy na špičce technologie, stále sdílí omezení s předchozími modely GPT: není plně spolehlivý, má omezené kontextové okno a neučí se z vlastní zkušenosti. (OpenAI, 2023)

Chat-GPT 4.0 architektura

Cretu (Scalable Path, 2023) definuje ChatGPT jako rozsáhlý jazykový model vyvinutý společností OpenAI, který využívá architekturu Transformer pro úlohy zpracování přirozeného jazyka. Architektura Transformer je navržena tak, aby zpracovávala a generovala odpovědi pro jakoukoli sekvenci znaků, které dávají smysl, a je založena na architektuře neuronové sítě, která se může učit z velkých datových sad a generalizovat nové situace.

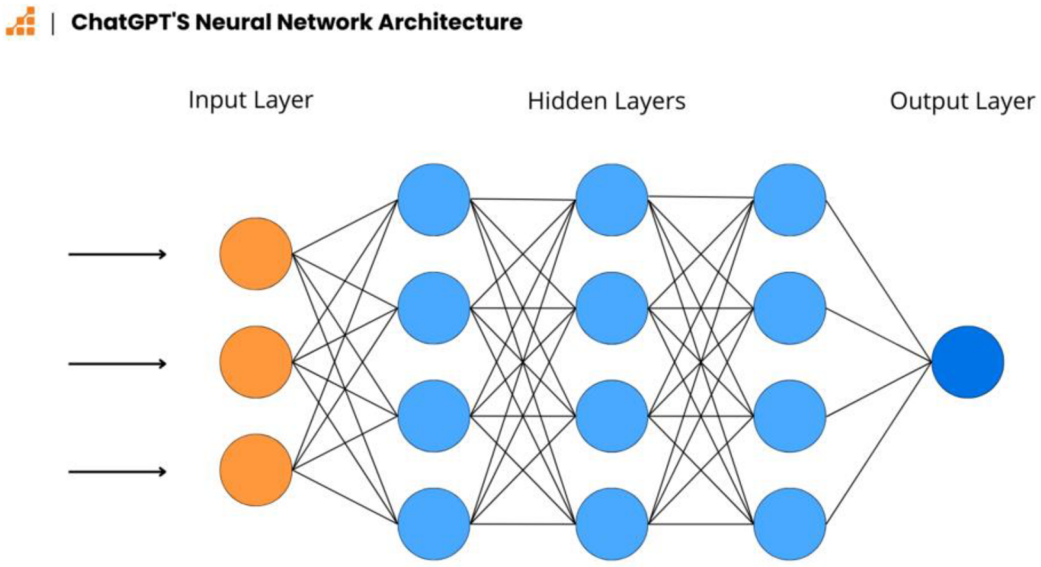
Architektura Transformer se skládá z enkodéru a dekodéru, které zpracovávají a generují odpovědi pro vstupní sekvenci. Enkodér zpracovává vstupní sekvenci a generuje sekvenci skrytých stavů, které reprezentují význam vstupní sekvence. Dekodér poté generuje sekvenci výstupů na základě těchto skrytých stavů, používající mechanismus nazývaný pozornost, aby se soustředil na různé části vstupní sekvence a přikládal větší váhu určitým tokenům, které jsou pro úkol relevantnější.

Mechanismus pozornosti v architektuře Transformer pomáhá síti zpracovávat a chápat složitá data tím, že identifikuje a zdůrazňuje nejrelevantnější informace. Toho dosahuje výpočtem váhy pro každý token ve vstupní sekvenci na základě jeho podobnosti se skrytými

stavy generovanými enkodérem. Síť poté používá tyto váhy k generování sekvence výstupů, které jsou pravděpodobnější, že budou pro úkol relevantní.

Proces trénování Chat-GPT

Proces trénování ChatGPT zahrnuje jemné ladění před trénovaného modelu na menší datové sadě specifické pro úkol, pro který je trénován. Tento proces zahrnuje tři kroky: supervizované jemné ladění, model odměn a posilovací učení. Při supervizovaném jemném ladění je model trénován na menší datové sadě označené odpovídajícími výstupními hodnotami, což mu pomáhá generovat lepší výstupy, a nakonec lépe plnit daný úkol. Ve fázi modelu odměn je model odměňován za generování výstupů, které jsou blíže požadovanému výstupu, což mu pomáhá učit se optimalizovat svůj výstup tak, aby maximalizoval signál odměny, který obdrží. V procesu posilovacího učení je model trénován pomocí technik posilovacího učení, přičemž obdrží zpětnou vazbu ve formě odměn nebo trestů, které mu pomáhají zlepšit jeho výkon.

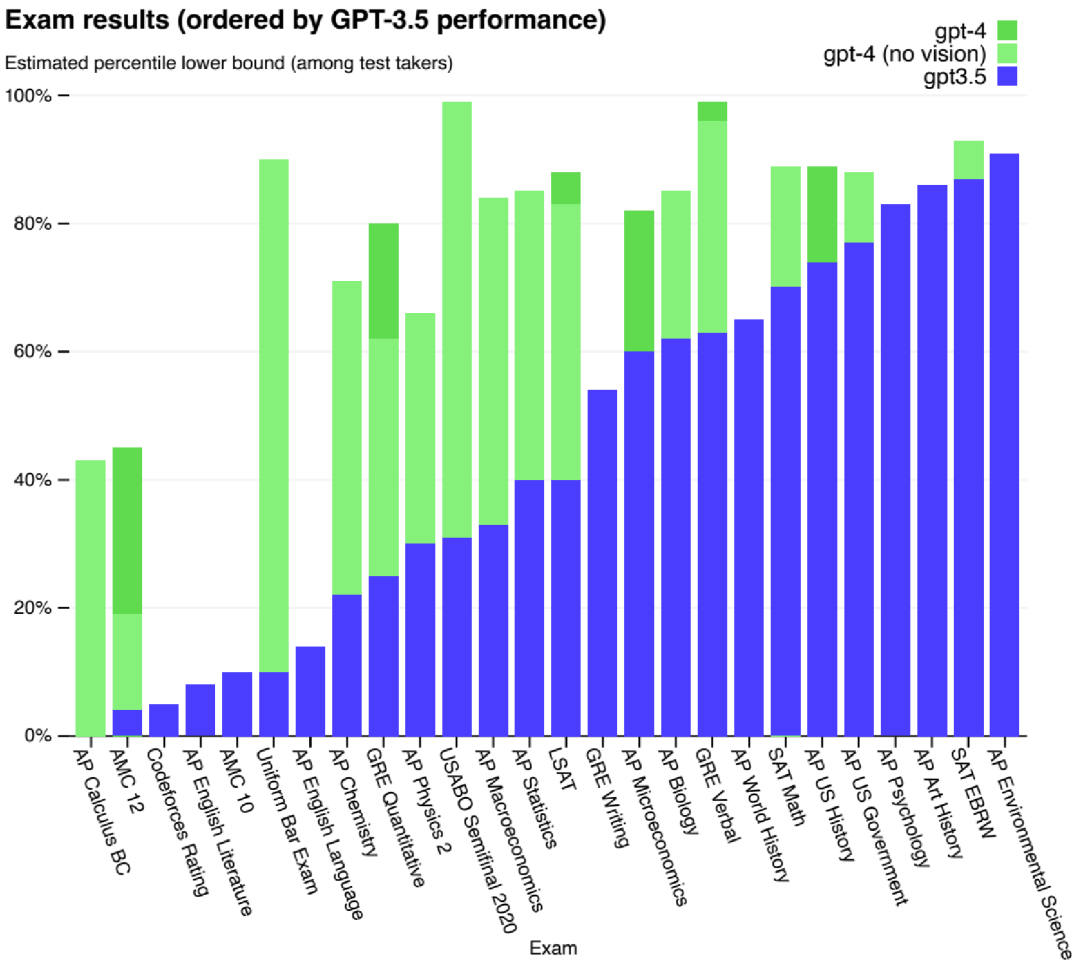


Obrázek 1 - Architektura neurální sítě Chatu GPT (Cretu,2023)

Výkon Chatu-GPT 4.0 a 3.5 v různých testech

V rámci zkoumání potenciálu umělé inteligence v procesu tvorby testovacích scénářů pro webové aplikace se nabízí zajímavý pohled na nedávné výsledky experimentů provedených s jazykovými modely GPT-4.0 a GPT-3.5. Graf, který předkládáme v této kapitole, poskytuje porovnání výkonnosti těchto modelů v různých akademických zkouškách, jež mohou mít nepřímý vztah k pochopení a testování webových aplikací.

Na grafu jsou zkoušky uspořádané podle výkonu GPT-3.5, přičemž jsou zahrnuty výsledky pro modely GPT-4.0 s vizuálními schopnostmi a GPT-4.0 bez těchto schopností. Přestože graf přímo nezahrnuje zkoušky specifické pro webové technologie. (OpenAI, 2023)



Obrázek 2 - Tabulka výsledků testů (OpenAI, 2023)

Implikace pro testování webových aplikací

Zkoušky zahrnuté v grafu, i když nepřímo, mohou odrážet schopnosti AI v logickém uvažování a algoritmickém myšlení. Tyto dovednosti jsou klíčové při návrhu testovacích scénářů, které vyžadují nejen pochopení kódu a jeho funkce, ale také predikci chování systému pod různými podmínkami a zátěží. Toto chování ověřím v praktické části této práce. (OpenAI, 2023)

3.6 Model Llama 2

V této kapitole podle Mety a GenAI (2023) je Llama 2 aktualizovaná verze původního modelu Llama 1, která byla vytvořena s cílem zlepšit předchozí výkon a funkčnost. Model je trénován na novém mixu veřejně dostupných dat a zahrnuje zvýšení velikosti datasetu o 40%, zdvojnásobení délky kontextu a adopci tzv. grouped-query attention (GQA), což je technika zlepšující škálovatelnost při inferenci u větších modelů.

Architektura modelu Llama 2

Llama 2 využívá optimalizovaný auto-regresivní transformátorový model, který je základem pro mnoho moderních LLM. Model kombinuje robustní čištění dat s aktualizovanými mixy dat a je trénován na větším množství tokenu. Kromě toho zdvojnásobuje délku kontextu, což umožňuje modelu lépe pochopit a zapojit se do delších dialogů nebo textů.

Proces trénování LLama 2

Vývoj modelů Llama 2 zahrnoval inovace oproti jejich předchůdcům, s cílem dosáhnout lepšího výkonu a efektivity. V procesu byly implementovány pokročilé techniky čištění dat, rozšíření datových sad, prodloužení kontextu a použití skupinové dotazové pozornosti pro lepší škálovatelnost větších modelů.

Předtrénování a příprava dat

Modely byly předtrénovány na rozsáhlém korpusu veřejně dostupných dat, zatímco data obsahující osobní informace byla pečlivě vyřazena. Předtrénování zahrnovalo intenzivní čištění a výběr dat, aby se zvýšila kvalita a přesnost modelů. Byla zavedena nová metoda skupinové dotazové pozornosti, která zlepšila schopnost modelů zpracovávat dlouhé texty a zvýšila efektivitu inferenčních výpočtů.

Fine-tuning

Fine-tuning (dolaďování) je proces, při kterém se předtrénovaný model dále upravuje pro konkrétní úlohy nebo datové sady. Llama 2 prošla několika fázemi fine-tuning, kde byly modely speciálně upraveny pro lepší pochopení a generování textu na základě nových nebo specifických datových sad. Tento proces umožnil modelům dosáhnout větší přesnosti a relevanci v různých aplikacích.

Reward Modeling

Reward Modeling je klíčová součást tréninkového procesu, která se zaměřuje na vývoj modelů odměn, jež hodnotí výstupy modelu na základě preferencí uživatelů nebo specifických kritérií. Při této technice jsou vytvářeny modely odměn, které slouží k posouzení a zlepšení chování jazykového modelu, a to tak, aby co nejlépe odpovídalo lidským preferencím. Tato metoda je zásadní pro zajištění, že modely produkují bezpečné, relevantní a užitečné odpovědi.

Výkon Llama 2 a její potenciální aplikace pro testování

Llama 2 byla testována na různých akademických benchmarkových testech a ukázala srovnatelné nebo lepší výsledky ve srovnání s ostatními uzavřenými modely, jako je GPT- 3.5, PaLM a dokonce i PaLM-2-L. Tato výkonnost naznačuje, že Llama 2 může být užitečná pro různé aplikace, včetně vytváření chatbotů, automatizované odpovědi na otázky, překladu a mnoha dalších úloh, kde je potřeba hluboké porozumění jazyku jako je třeba při psaní testovacích scénářů.

3.7 Model Mixtral 8x7B

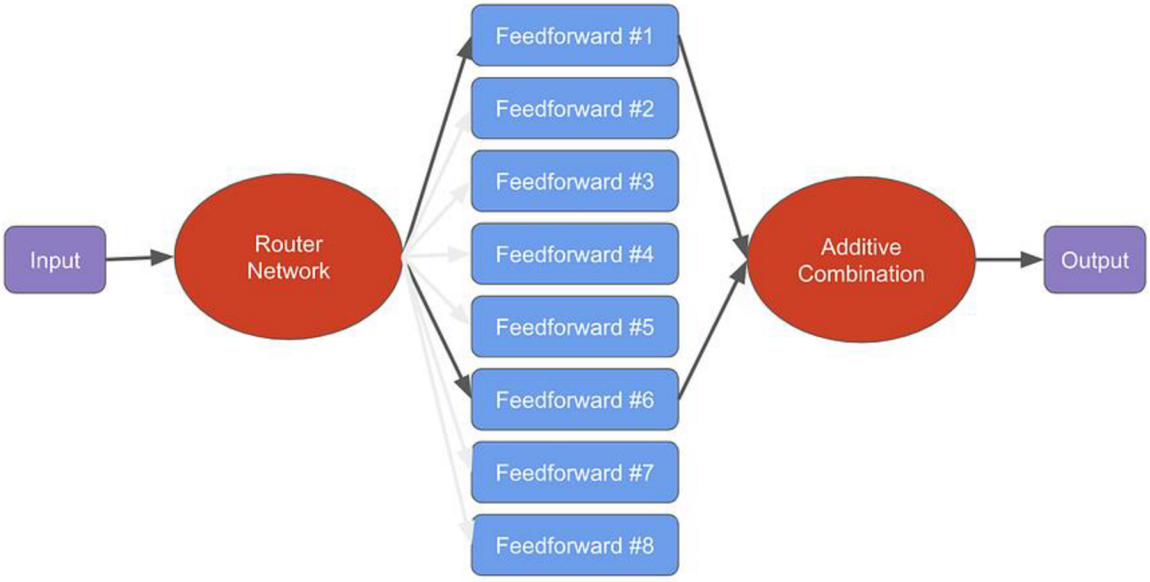
V této části podle Marie (2023) je model Mistral, konkrétněji Mixtral 8x7B, je otevřený model s licencí Apache 2.0, který představuje inovativní přístup v návrhu a škálovatelnosti. Tento model v sobě kombinuje rozptýlenou architekturu s unikátním způsobem výběru expertních skupin, které jsou vybírány a kombinovány pro každou vrstvu a token.

Obrovským benefitem je také možnost tento model spustit lokálně což zabraňuje úniku potenciálně citlivých dat v případě napadení hostitele jazykového modelu.

Vzhledem k otevřené povaze modelu a jeho licenci Apache 2.0, Mixtral poskytuje vědcům a vývojářům možnost testovat a adaptovat model pro specifické účely bez omezení. To otevírá dveře pro širokou škálu aplikací a výzkumných projektů, a to nejen v komerčním sektoru, ale i v akademickém prostředí.

Architektura Mixtral 8x7B

Mixtral se prezentuje jako model pouze s dekodérem, což představuje určitý odchyl od tradičních sekvenčních modelů. Jeho specifika spočívají v tom, že pro zpracování informací využívá osm různých skupin parametrů. Tyto skupiny, často označované jako "experti", jsou selektivně využívány díky směrovací síti, která pro každý token efektivně vybírá a aditivně kombinuje výstupy dvou skupin. Tento proces umožňuje modelu zpracovávat data s vysokou účinností a přitom zachovat nízké náklady na výpočetní zdroje, což je zásadní vzhledem k celkovému počtu 46,7 miliard parametrů modelu.



Obrázek 3-A sparse mixture of experts (volný překlad: řídká směs odborníků) (Marie, 2023)

Výkon Mixtral 8x7B a jeho potenciální aplikace pro testování

Výkonnostní testy ukazují, že Mixtral překonává mnohé své předchůdce, včetně modelů jako je Llama 2 70B, a to jak v kvalitě výstupu, tak v efektivitě využití výpočetních zdrojů. V benchmark testech dosahuje srovnatelných nebo lepších výsledků než GPT-3.5, což demonstruje jeho schopnost generovat kód a zpracovávat rozsáhlý kontext až 32 tisíc tokenů. Kromě toho je Mixtral schopen pracovat s více jazyky, což z něj činí vícejazyčnou platformu s dobrou znalostí angličtiny, francouzštiny, italštiny, němčiny a španělštiny. Tohle všechno dělá z Mixtralu 8x7b zajímavého kandidáta na psaní testovacích scénářů. (Mistral AI team, 2023)

4 Vlastní práce

4.1 Vícekriteriální analýza modelů

Z dostupných dat jsem vytvořil tabulku porovnávající modely v určitých testech, bohužel žádný test přímo neimplikuje schopnost vytvářet testovací scénáře, ale z tabulky můžeme vyčíst určitý vývoj ve schopnostech v čase a hrubě modely porovnat.
Data vychází z teoretické části práce doplnil jsem také test MBPP chat-GPT 4.0
(Papers with Code)

Název testu	LLaMA 2 70B	GPT - 3.5	Mixtral 8x7B	GPT - 4.0
MMLU (MCQ in 57 subjects)	69.9%	70.0%	70.6%	86.4%
HellaSwag (10-shot)	87.1%	85.5%	86.7%	95.3%
ARC Challenge (25-shot)	85.1%	85.2%	85.8%	96.3%
WinoGrande (5-shot)	83.2%	81.6%	81.2%	87.5%
MBPP (pass@1)	49.8%	52.2%	60.7%	87.5%
GSM-8K (5-shot)	53.6%	57.1%	58.4%	92.0%
MT Bench (for Instruct Models)	6,86	8,32	8,30	X
Cena	1	1	1	4
Otevřenost modelu	1	2	1	2

Obrázek 4- Tabulka výsledků (vlastní práce)

Název testu	LLaMA 2 70B	GPT - 3.5	Mixtral 8x7B	GPT - 4.0
MMLU (MCQ in 57 subjects)	4	3	2	1
HellaSwag (10-shot)	2	4	3	1
ARC Challenge (25-shot)	4	3	2	1
WinoGrande (5-shot)	2	3	4	1
MBPP (pass@1)	4	3	2	1
GSM-8K (5-shot)	4	3	2	1
Otevřenost modelu	1	2	1	2
cena	1	1	1	4

Obrázek 5 – Tabulka pořadí (vlastní práce)

4.1.1 Popis a váha kritérií

Metodologie hodnocení

Analýza byla provedena použitím metody váženého skórování, kde každé kritérium bylo ohodnoceno a následně vynásobeno specifickou váhou, která odráží jeho důležitost pro kontext vytváření testovacích scénářů. Váhy byly pečlivě zvoleny s ohledem na to, jak významně každé kritérium přispívá k celkové užitečnosti modelu pro daný účel.
Detaily a popis kritérií:

1. **MMLU (MCQ in 57 subjects):**
   - Toto kritérium odráží výkon modelu v testu s výběrem z více odpovědí na různá témata. Vyšší skóre znamená lepší porozumění a adaptabilitu ve širokém rozsahu témat.
   - Váha: 0.20 (předpokládá významnou schopnost modelu zvládat široké spektrum znalostí)
2. **HellaSwag (10-shot):**
   - Toto kritérium hodnotí schopnost modelu predikovat správné zakončení vět na základě krátkého kontextu.
   - Váha: 0.05 (odráží schopnost modelu pochopit kontext a generovat relevantní obsah)
3. **ARC Challenge (25-shot):**

- Odráží schopnost modelu řešit otázky založené na znalostech, které vyžadují porozumění a aplikaci komplexních informací.
- Váha: 0.15 (značí důležitost schopnosti modelu aplikovat znalosti v obtížnějších situacích)

4. **WinoGrande (5-shot):**
   - Testuje schopnost modelu porozumět a vyřešit věty, které obsahují dvojznačnosti, což je kritické pro generování přirozeně znějícího textu.
   - Váha: 0.10 (důležité pro přirozenou interakci, ale méně než výkonnost v širším kontextu)
5. **MBPP (pass@1):**
   - Měří, jak dobře model dokáže psát a debuggovat kód, což může být velmi užitečné pro vytváření softwarových testů.
   - Váha: 0.15 (specifické pro vývoj softwaru, ale nemusí být primární pro všechny testovací scénáře)
6. **GSM-8K (5-shot):**
   - Hodnotí, jak dobře model řeší slovní úlohy, což ukazuje na jeho matematické a logické uvažování.
   - Váha: 0.10 (důležité pro analytické úlohy, ale může být méně relevantní pro některé typy testů)
7. **Otevřenost modelu:**
   - Odráží, jak snadno může být model upraven nebo integrován do různých systémů a projektů.
   - Váha: 0.10 (důležité pro adaptabilitu a integraci, ale sekundární vůči výkonnostním měřítkům)
8. **Cena:**
   - Reprezentuje finanční náklady spojené s používáním modelu, což je kritický faktor pro mnoho organizací.
   - Váha: 0.10 (náklady jsou důležité, ale nemělo by to převážit výkonnost a schopnosti)

Název testu	Váha	body
MMLU (MCQ in 57 subjects)	0,2	2
HellaSwag (10-shot)	0,05	0,5
ARC Challenge (25-shot)	0,2	2
WinoGrande (5-shot)	0,1	1
MBPP (pass@1)	0,15	1,5
GSM-8K (5-shot)	0,1	1
Otevřenost modelu	0,1	1
cena	0,1	1
SUMA	1	10

Obrázek 6- (vlastní práce)

Název testu	LLaMA 2 70B	GPT - 3.5	Mixtral 8x7B	GPT - 4.0
MMLU (MCQ in 57 subjects)	2	4	6	8
HellaSwag (10-shot)	1,5	0,5	1	2
ARC Challenge (25-shot)	2	4	6	8
WinoGrande (5-shot)	3	2	1	4
MBPP (pass@1)	1,5	3	4,5	6
GSM-8K (5-shot)	1	2	3	4
Otevřenost modelu	4	3	4	3
cena	4	4	4	1
SUMA	19	22,5	29,5	36

Obrázek 7- (vlastní práce)

4.1.2 Výsledky hodnocení

Podle provedené analýzy dosáhl model GPT-4.0 nejvyššího celkového skóre 36, což jej činí nejvýše hodnoceným modelem v mojí práci. Tento model vyniká ve všech testech výkonnosti, zejména v MMLU (MCQ in 57 subjects), ARC Challenge a MBPP, což ukazuje na jeho schopnost účinně zpracovávat a generovat přirozený jazyk, stejně jako na jeho schopnost řešit složité úlohy. Nicméně, je důležité poznamenat, že model GPT-4.0 byl také ohodnocen nejnižší hodnotou v kategorii ceny, což naznačuje, že přestože je výkonnostně nejsilnější, může být také nejnákladnější možností. Momentální cena na měsíc činí 20 dolarů.

Na druhém místě se umístil model Mixtral 8x7B s celkovým skóre 29,5. Tento model ukázal silné výsledky ve výkonnostních testech a byl hodnocen stejně jako GPT-4.0 v kategorii otevřenosti modelu a ceny. To naznačuje, že ačkoliv je méně výkonný než GPT-4.0, může nabízet lepší rovnováhu mezi cenou a otevřeností.

Model GPT-3.5 dosáhl celkového skóre 22,5 a předvedl solidní výkonnost v testech. Přestože je tento model překonán novějšími modely, jeho výsledky naznačují, že stále zůstává konkurenceschopný, zvláště pokud jsou zváženy faktory jako cena a otevřenost.

LLaMA 2 70B, s celkovým skóre 19, představuje model s nejnižším hodnocením v naší analýze. Zatímco tento model zaznamenal dobré výsledky v kategorii otevřenosti modelu, jeho výkonnost v testech byla nižší ve srovnání s ostatními modely.

Diskuse

Při výběru modelu pro vytváření testovacích scénářů je klíčové zvážit nejen výkonnost v izolovaných testech, ale také celkovou otevřenost modelu a náklady na jeho používání. Model GPT-4.0 jasně vede v kategorii výkonnosti, ale jeho vyšší cena může být limitujícím faktorem pro některé projekty. Na druhou stranu, modely jako Mixtral 8x7B nabízejí více vyváženou možnost.

V kontextu mé bakalářské práce jsem se rozhodl využít nejsilnější dostupný nástroj – CHAT GPT 4.0. Toto rozhodnutí bylo učiněno na základě této vícekriteriální analýzy, v níž GPT-4.0 výrazně převyšoval ostatní hodnocené modely, což signalizuje jeho potenciál pro vytváření testovacích scénářů.

Výběr CHAT GPT 4.0

GPT-4.0 prokázal ve všech sledovaných kategoriích mimořádné schopnosti, zvláště v klíčových testech výkonnosti, které jsou nezbytné pro generování komplexních a realistických testovacích scénářů. Tyto schopnosti zahrnují pokročilé pochopení přirozeného jazyka, adaptabilitu při řešení nejasností a komplexních problémových otázek, stejně jako schopnost efektivního programování a ladění kódu.

4.2 Návrh webové aplikace

4.2.1 Úvod

V rámci této kapitoly bakalářské práce se zaměřuji na návrh webové aplikace GlobeTrotter, který slouží jako základ pro simulaci testů a tvorbu testovacích scénářů generovaných umělou inteligencí CHAT GPT-4. Hlavním cílem aplikace GlobeTrotter je umožnit uživatelům rezervovat letenky, hotely a vozidla na jednom místě, což z ní činí komplexní nástroj pro plánování cest. Přestože v rámci této práce nebudu aplikaci skutečně implementovat, je důležité podrobně specifikovat její funkce a možnosti, aby bylo možné na ní efektivně simulovat testovací scénáře.

4.2.2 Domovská stránka

Úvodní stránka GlobeTrotteru je prvním kontaktním bodem mezi uživatelem a platformou, přičemž je klíčová pro zajištění pozitivního prvního dojmu. Je pečlivě navržena s cílem okamžitě zaujmout pozornost uživatelů a intuitivně je vést k dalším krokům jejich cestovního plánování.

Centrálním prvkem úvodní stránky je vyhledávací panel, který je umístěn strategicky na význačném místě, aby uživatelé mohli snadno zahájit proces vyhledávání letů, hotelů nebo vozidel k pronájmu. Panel podporuje jasné a jednoduché rozhraní, které minimalizuje počet kroků potřebných k iniciaci vyhledávání.

Dalším významným prvkem je karusel (banner s cyklicky rotujícím obsahem) obrázků, který prezentuje atraktivní destinace, speciální nabídky a doporučení, sloužící jako inspirace pro uživatele přemýšlející o svých budoucích cestách. Obrázky jsou doplněny stručnými a poutavými popisky, které podněcují zvědavost a motivují k dalšímu prozkoumávání.

Navigační lišta je dalším klíčovým prvkem, který umožňuje uživatelům snadný přístup k různým částem webu, jako jsou lety, hotely, pronájem aut, speciální nabídky a uživatelský profil. Je navržena tak, aby byla intuitivní a snadno srozumitelná pro všechny uživatele, nezávisle na jejich předchozích zkušenostech s podobnými platformami.

4.2.3 Vyhledávač letů

Vyhledávač letů je zásadní komponentou aplikace, která umožňuje uživatelům vyhledávat lety podle různých kritérií, jako jsou destinace, datum odletu a návratu, počet cestujících a třída letenky. Funkce autofill a inteligentní návrhy zvyšují uživatelský komfort tím, že usnadňují vyplňování formulářů a nabízejí možnosti založené na předchozích vyhledáváních a oblíbených destinacích. Flexibilita ve výběru data umožňuje uživatelům najít lety s nejlepší cenou nebo v nejvhodnější čas, zatímco pokročilé filtry umožňují uživatelům přizpůsobit vyhledávání na základě specifických preferencí, jako jsou přímé lety, doba letu nebo letecká společnost.

4.2.4 Rezervace hotelů

Systém rezervace hotelů na GlobeTrotteru je navržen tak, aby poskytoval uživatelům plynulý a intuitivní proces vyhledávání a rezervace ubytování. Tento systém je klíčovou součástí platformy, jelikož umožňuje uživatelům snadno najít a rezervovat hotely, které nejlépe vyhovují jejich potřebám a preferencím.

4.2.5 **Rezervace auta**

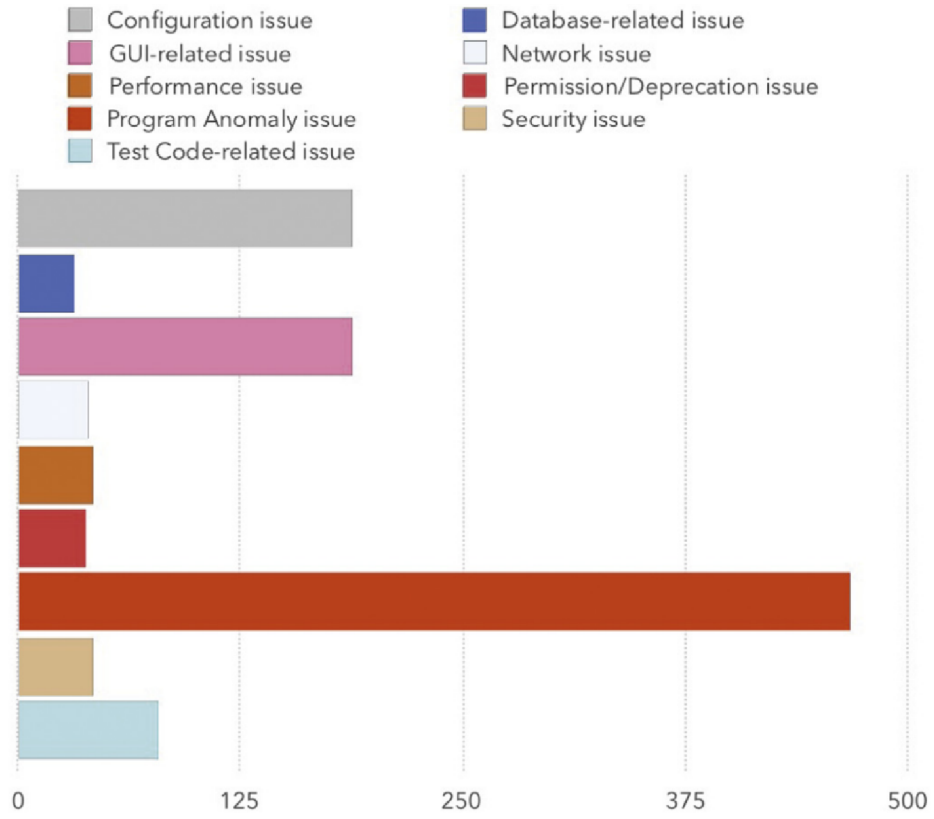
Služba půjčování aut na platformě GlobeTrotter je navržena tak, aby poskytovala uživatelům snadný a přehledný způsob, jak si půjčit vozidlo pro jejich cesty. Stránka půjčování aut je klíčovou součástí této služby a je přizpůsobena tak, aby odpovídala potřebám a představám širokého spektra uživatelů.

4.2.6 **Případy užití**

Jednotlivé případy užití, které jsou použity pro generování testovacích scénářů naleznete v příloze.

4.3 Chyby ve webové aplikaci

V rámci vývoje zmíněné webové aplikace je nezbytné zahrnout simulaci chyb, které by měly co nejvíce reflektovat skutečné situace, s nimiž se uživatelé mohou setkat. Abych zajistil, že simulované chyby budou co nejvíce realistické a přínosné pro testování aplikace, opírám se o studii představenou v teoretické části této práce. Jak uvádí Catolino (et al., 2019) ve své studii, struktura a rozvržení simulovaných chyb by mělo následovat specifické uspořádání, které je detailně popsáno a analyzováno. Toto uspořádání je klíčové pro efektivní implementaci testovacích scénářů a pro zajištění, že testování bude co nejblíže reálným podmínkám a výzvám, které mohou v praxi nastat.
Jedinou výjimkou jsou chyby testovacího kódu, jelikož cílem je vytvářet scénáře pro manuální testování a z toho důvodu nejsou tyto chyby relevantní.

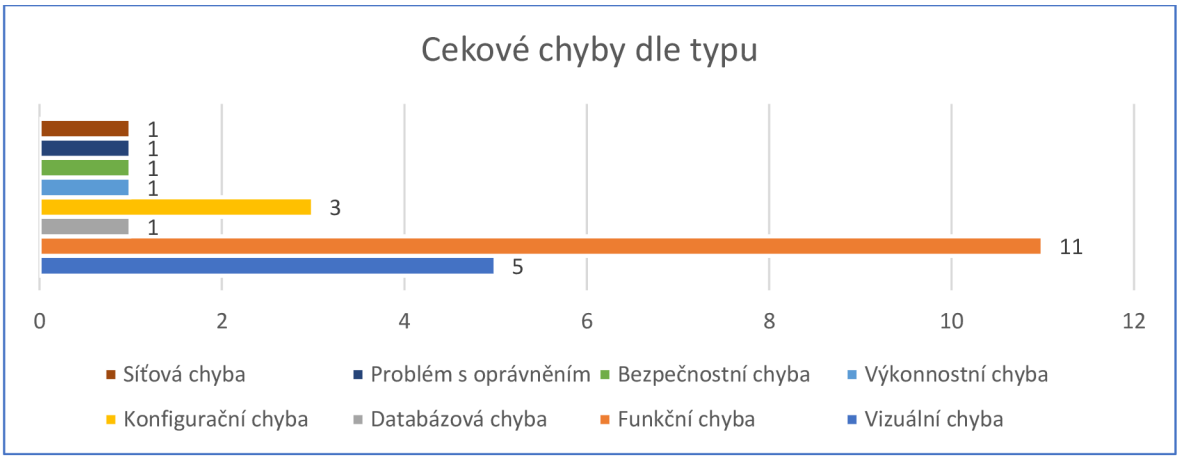


Obrázek 8- Četnost každé kategorie primární příčiny chyb (Catolino, 2019)

V rámci testování pro bakalářskou práci bylo navrženo celkem 27 chyb s detailním rozložením zaměřeným na různé aspekty funkcionality a bezpečnosti webové aplikace. Funkční chyby představují největší podíl s 51,85 %, což poukazuje na významné místo, které funkčnost zaujímá v celkové stabilitě a použitelnosti aplikace. Vizuální chyby související s grafickým uživatelským rozhraním mají podíl 18,52 %, zdůrazňující důležitost estetiky a intuitivního designu pro uživatelský zážitek. Konfigurační chyby tvoří 11,11 %, což signalizuje potenciální komplikace v správném nastavení a konfiguraci systému. Síťové chyby, výkonnostní chyby, bezpečnostní chyby, databázové chyby a problémy s oprávněním mají stejný podíl, každá kategorie zaujímá 3,70 %. Tato rozmanitost chyb zdůrazňuje potřebu komplexního přístupu k testování, kde je nutné adresovat široké spektrum

potenciálních problémů, od síťové komunikace přes výkon, zabezpečení, integritu dat až po správné řízení uživatelských oprávnění.

Tato strategie testování reflektuje realistické scénáře a výzvy, se kterými se vývojáři a testeři mohou setkat při práci na webových aplikacích.



Obrázek 9- (vlastní práce)

4.3.1 Chyby na domovské stránce

Chyba #1: Poškozené obrázky v karuselu

- Popis: Karusel na domovské stránce má integrovány obrázky s vysokým rozlišením prezentující populární destinace. Chyba se projevuje tak, že místo obrázků se uživateli zobrazují ikony poškozeného obrázku.
- Kategorizace: Vizuální chyba

Chyba #2: Sekce s uživatelskými recenzemi

- Popis: Sekce s uživatelskými recenzemi obsahuje textové bloky a obrázky, které se nekorektně překrývají kvůli chybám v CSS stylování nebo nedostatkům v responzivním designu.
- Kategorizace: Vizuální chyba

Chyba #3: Nefunkční odkazy v navigačním baru

- Popis: Navigační bar obsahuje odkazy vedoucí na různé části webu. U některých odkazů dochází k chybě, kdy jsou uživatelé přesměrováni na neexistující stránky, což vyústí v chybu 404.
- Kategorizace: Funkční chyba

Chyba #4: Formulář pro přihlášení k odběru novinek

- Popis: Uživatel může zadat svůj e-mail do formuláře pro odběr novinek. Formulář akceptuje e-mail, ale systém selže při odesílání potvrzovacího e-mailu uživateli.
- Kategorizace: Funkční chyba

Chyba #5: Po kliknutí na článek tipů na cestování stránka přesměruje jinam, než má

- Popis: Odkazy na články v sekci nejnovějších cestovních zpráv mají za následek zobrazení článku, jenž není ten, na který bylo kliknuto
- Kategorizace: Funkční chyba

Chyba #6: Zastaralá knihovna způsobí pád aplikace po kliknutí na obrázek v Karuselu

- Popis: Knihovna, jenž má za úkol spravovat rotaci karuselu způsobí pád aplikace po kliknutí na jeden z rotujících obrázků.
- Kategorizace: Konfigurační chyba

4.3.2 Chyby na stránce rezervace hotelů

Chyba #7: Nesprávné výsledky vyhledávání

- Popis: Při vyhledávání hotelů podle města, data příjezdu a odjezdu a počtu osob někdy systém zobrazuje výsledky, které neodpovídají zadaným kritériím.
- Kategorizace: Funkční chyba

Chyba #8: Zpomalení při načítání mapy

- Popis: Interaktivní mapa zobrazující polohu hotelů je velmi pomalá při načítání a někdy se dokonce nenačte vůbec, což komplikuje proces výběru hotelu na základě jeho umístění.
- Kategorizace: Výkonnostní chyba (Performance)

Chyba #9: Nesprávné zobrazení informací o pokoji

- Popis: Po výběru typu pokoje a pokračování k rezervaci se občas stane, že detaily a cena pokoje neodpovídají původně vybraným specifikacím.
- Kategorizace: Funkční chyba

Chyba #10: Exponovaná platební data

- Popis: Na stránce pro zadání platebních informací může dojít k odhalení citlivých dat v URL adrese, což představuje vážné bezpečnostní riziko.
- Kategorizace: Bezpečnostní chyba

Chyba #11: Oprávnění při přidávání recenzí

- Popis chyby: Na stránce rezervace hotelů mají uživatelé možnost přidávat recenze na hotely, ve kterých se ubytovali. Implementovaná chyba spočívá v tom, že určitá skupina uživatelů narazí na problém s oprávněními, když se pokusí odeslat recenzi. Systém jim oznámí, že nemají dostatečná oprávnění pro přidání recenze, přestože mají být přihlášeni a měli by mít tuto možnost jako součást svých uživatelských práv. Tento problém může být způsoben nesprávnou konfigurací serveru nebo chybou v logice aplikace, která ověřuje oprávnění uživatelů.
- Kategorizace: Problém s oprávněním

Chyba #12: Problémy s galerií obrázků hotelu

- Popis: Galerie obrázků hotelu trpí problémy s načítáním a někdy zobrazuje obrázky v nesprávném poměru stran nebo vůbec nezobrazuje náhledy.
- Kategorizace: Vizuální chyba

Chyba #13: Problém s načítáním recenzí z externího API

- Popis: Stránka hotelu zahrnuje sekci s uživatelskými recenzemi, které jsou načítány prostřednictvím externího API. Chyba se projevuje jako neschopnost stránky správně zobrazit recenze kvůli problémům s timeoutem nebo přerušením síťového připojení. Tento problém simuluje reálné situace, kdy externí služby nejsou dostupné nebo je jejich odezva pomalá, což ovlivňuje celkovou funkcionalitu a uživatelskou zkušenost na stránce.
- Kategorizace: Síťová chyba

Chyba #14: Problém s aktualizací dostupnosti pokojů

- Popis: Po provedení rezervace by měl systém automaticky aktualizovat dostupnost pokojů v daném hotelu. V tomto případě chyba v databázi způsobí, že ačkoliv uživatel dokončí proces rezervace a obdrží potvrzení, stav dostupnosti pokojů v databázi se neaktualizuje správně. To může vést k situaci, kdy je stejný pokoj nabídnut více uživatelům, což způsobuje konflikty v rezervacích a možné přebookování.
- Kategorizace: Databázová chyba

4.3.3 Chyby na stránce Vyhledávač letů

Chyba #15: Autofill nesprávných letišť
- Popis: Vyhledávací formulář poskytuje možnost autofill pro vstupní pole letišť. Chyba se projevuje tím, že návrhy někdy obsahují nesprávná nebo neexistující letiště.
- Kategorizace: Funkční chyba

Chyba #16: Výběr data v minulosti
- Popis: Kalendář umožňuje uživatelům vybrat datum odletu, které je v minulosti, což by mělo být omezeno na aktuální a budoucí data.
- Kategorizace: Funkční chyba

Chyba #17: Filtr přímých letů
- Popis: Filtr pro vyhledávání přímých letů někdy zobrazí lety s mezipřistáními, což je v rozporu s očekáváním uživatele.
- Kategorizace: Funkční chyba

Chyba #18: Filtr cenového rozmezí
- Popis: Cenový posuvník nesprávně filtruje lety podle ceny, ať už nezobrazuje nejlevnější nabídky nebo nerespektuje nastavený cenový limit.
- Kategorizace: Funkční chyba

Chyba #19: Skryté detaily letů
- Popis: Rozbalovací sekce s detaily letů má chybu, která někdy způsobí, že se obsah nezobrazí a zůstane skrytý před uživatelem.
- Kategorizace: Vizuální chyba

Chyba #20: Tlačítko pro rezervaci letu
- Popis: Tlačítko „Rezervovat let“ občas nereaguje nebo způsobuje pád stránky, což uživatele znemožňuje dokončit rezervaci.
- Kategorizace: Funkční chyba

Chyba #21: Ztráta uložených letů
- Popis: Funkce pro ukládání vyhledaných letů je nespolehlivá a někdy dojde ke ztrátě informací o uložených letech kvůli problémům se správou relací.
- Kategorizace: Konfigurační chyba

4.3.4 Chyby na stránce rezervace aut

Chyba #22: Výběr místa vyzvednutí a vrácení auta
- Popis: Pole pro výběr místa vyzvednutí a vrácení vozidla někdy nezaznamenává zadané informace, nebo se informace resetují po odchodu uživatele ze stránky.
- Kategorizace: Funkční chyba

Chyba #23: Výběr data a času vyzvednutí a vrácení
- Popis: Systém umožňuje uživatelům vybrat čas vrácení vozidla, který je před časem vyzvednutí, což by nemělo být možné.
- Kategorizace: Funkční chyba

Chyba #24: Resetování preferencí vozidla
- Popis: Preferované nastavení vozidla, jako je model nebo přídavné vybavení, se resetuje po navigaci uživatele pryč ze stránky a návratu zpět.
- Kategorizace: Konfigurační chyba

Chyba #25 Kalkulace celkové ceny
- Popis: Celková cena vypočtená při pokladně je nesprávná a neodráží výběry učiněné uživatelem, což vede k nedorozuměním ohledně očekávané ceny pronájmu.
- Kategorizace: Funkční chyba

Chyba #26: Text s podmínkami pronájmu

- Popis: Text s podmínkami pronájmu se překrývá s ostatními prvky na stránce, což znemožňuje uživatelům jejich čtení a pochopení.
- Kategorizace: Vizuální chyba

Chyba #27: Potvrzovací e-mail

- Popis: Po dokončení rezervace systém selhává při odesílání potvrzovacího e-mailu, což uživatele zbavuje jistoty, že byla jejich rezervace úspěšně zaznamenána.
- Kategorizace: Funkční chyba

4.4 Analýza procesů generování scénářů

4.4.1 Analýza existujících postupů

V této kapitole shrneme klíčové postupy a strategie pro efektivní využití prompt engineeringu v kontextu generování textů s využitím modelů umělé inteligence, jak jsou prezentovány ve studii. Tato shrnutí reflektují nejnovější poznatky a doporučení pro optimalizaci interakcí s jazykovými modely, a tím pro zvýšení kvality a relevancí vygenerovaných odpovědí.

1. Jasné a specifické instrukce: Jeden z klíčů k úspěchu při práci s AI generátory textu je poskytování jasných a specifických instrukcí. Tento přístup minimalizuje nedorozumění a umožňuje modelu generovat cílenější a přesnější odpovědi na základě jasného zadání.

2. Experimentování s kontextem a příklady: Zahrnutí kontextu a konkrétních příkladů do promptů může významně zlepšit kvalitu a relevanci odpovědí. Poskytnutím dodatečných informací nebo příkladů může uživatel lépe nasměrovat AI k požadovaným výstupům.

3. Iterativní testování a vylepšování: Úspěch v prompt engineeringu často závisí na schopnosti iterativně testovat a upravovat prompty. Tento proces umožňuje identifikovat a odstranit slabiny ve vygenerovaných odpovědích a postupně zdokonalovat interakci s modelem.

4. Vyvážení uživatelského záměru a kreativity modelu: Při navrhování promptů je důležité nalézt rovnováhu mezi přímým záměrem uživatele a prostor pro kreativní generování modelu. Umožněním modelu, aby projevil určitou míru kreativity, můžeme často dosáhnout překvapivě inovativních a užitečných výstupů.

5. Využití externích zdrojů a API: Integrace externích zdrojů a API do promptů může významně rozšířit možnosti a užitečnost generovaných textů. Tímto způsobem můžeme překlenout mezery v znalostech modelu a zpřístupnit aktualizované informace nebo specializovaná data.

6. Zajištění etického používání a vyhýbání se předsudkům: Etika a odpovědnost hrají zásadní roli v prompt engineeringu. Je nutné pečlivě monitorovat a upravovat prompty tak, aby se minimalizovala možnost generování zkreslených, nevhodných nebo citlivých textů. (Ekin, 2023)

4.4.2 Přístupy k generování testovacích scénářů s využitím AI

Přístup, který jsem zvolil pro generování testovacích scénářů v praktické části mé bakalářské práce, odráží aplikaci nejlepších postupů v oblasti prompt engineeringu a testování softwaru, jak jsou diskutovány v předchozím shrnutí. Tento proces je příkladem, jak se může můj postup překrývat s doporučenými metodami pro efektivní využití AI při optimalizaci kvality a rozsahu testování.

1. Jasné a specifické instrukce: Poskytnutím podrobného kontextu webové aplikace a specifikace jednotlivých případů užití jsem zajistil, že instrukce pro AI byly jasné a zaměřené. Tím jsem usnadnil generování přesných a relevantních testovacích scénářů, což je v souladu s principem poskytování jasných a specifických pokynů.

2. Experimentování s kontextem a příklady: Integrací detailního kontextu webové aplikace a specifických případů užití do promptů pro AI jsem efektivně experimentoval s kontextem, což umožnilo modelu vygenerovat scénáře, které lépe odpovídají specifickým potřebám a charakteristikám testovaného systému.

3. Iterativní testování a vylepšování: Metodika "Pokračuj v generování", zaměřená na dosažení 100% pokrytí testování, reflektuje iterativní přístup k vytváření a zlepšování testovacích scénářů. Tento iterativní proces je klíčem k identifikaci a zaplnění mezer v testovacím pokrytí.
4. Vyvážení uživatelského záměru a kreativity modelu: Použitím AI k generování testovacích scénářů na základě případů užití, zatímco je zároveň poskytnut prostor pro "Pokračuj v generování" promptů, jsem umožnil modelu projevit kreativitu při navrhování scénářů, které možná přímo nevyplynuly z předdefinovaných případů užití. To vede k vyvážení mezi specifickými požadavky a potenciálem modelu přicházet s novými a inovativními scénáři.
5. Využití externích zdrojů a API: Ačkoli můj přístup přímo nezahrnuje integraci externích API pro generování testovacích scénářů, otevřenost k využití AI a promptů pro rozšíření testovacích možností odráží podobnou filozofii adaptace a využití dostupných technologických nástrojů.
6. Zajištění etického používání a vyhýbání se předsudkům: Mým záměrem je vytvářet testovací scénáře, které jsou inkluzivní a eticky odpovědné. I když to není přímo zmíněno, tento princip je důležitým aspektem při každé aplikaci AI a generování obsahu.

5 Výsledky a diskuse

5.1 Úvod

V kapitole "Výsledky" se zaměřím na analyzování a prezentaci klíčových zjištění z experimentu využívajícího umělou inteligenci GPT-4.0 pro generování testovacích scénářů. Cílem experimentu bylo prozkoumat potenciál a omezení strojového učení v kontextu automatizace testování softwaru, zejména při psaní detailních a specifických testovacích scénářů pro různé případy užití. V této kapitole budu detailně zkoumat, jak umělá inteligence GPT-4.0 přistoupila k této úloze, jaké byly výsledky její práce, a jaké výzvy a příležitosti se v průběhu procesu objevily.

Zaměřím se na několik klíčových aspektů: detailnost a specifičnost vygenerovaných testovacích scénářů, jejich pokrytí a schopnost detekovat chyby, bezpečnostní rizika a přehlédnuté oblasti, nutnost supervize a lidského zásahu. Přestože výsledky experimentu ukázaly, že AI je schopná generovat detailní a velmi specifické testovací scénáře, nebylo dosaženo úplného pokrytí testů, a některé chyby zůstaly nedetekovány. Zároveň byly identifikovány oblasti, kde je třeba dalšího vývoje a supervize.

Tato kapitola poskytne podrobný pohled na to, co AI dokáže v současné době nabídnout v oblasti psaní testovacích scénářů, a jaké jsou její limity.

5.2 Popis a analýza testovacích scénářů

Výsledné vygenerované scénáře naleznete v příloze, scénáře jsou rozděleny po jednotlivých webových stránkách celkově jich je 538 a pokrývají širokou škálu možných situací, ale témata těchto scénářů jsou omezena pouze na dodané testovací případy.

5.2.1 Detailnost scénářů

Detailní popis scénářů je klíčem k identifikaci a nápravě chyb, které by mohly být v průběhu vývoje a udržování softwaru přehlédnuty. GPT-4.0 poskytuje rozsáhlé a detailní testovací scénáře, které nejen specifikují kroky potřebné k ověření funkčnosti jednotlivých prvků aplikace, ale také podávají hloubkovou analýzu možných variant uživatelského chování a interakcí. Tím se odlišují od tradičního přístupu, kde by například testování navigace na webové stránce bylo pokryto jediným scénářem s požadavkem "zkontroluj, že veškeré odkazy na stránce fungují správně". Umělá inteligence však přistupuje k této úloze detailnějším způsobem a vytváří pro každou stránku a funkci zvláštní testovací scénář, což umožňuje důkladnější kontrolu a vede k větší pravděpodobnosti odhalení chyb.

8	Navigation to Flights Page	Test navigation from the homepage to the Flights page via the navigation bar.	Selecting "Flights" from the navigation bar should redirect the user to the Flights page.
9	Navigation to Hotels Page	Test navigation from the homepage to the Hotels page via the navigation bar.	Selecting "Hotels" from the navigation bar should redirect the user to the Hotels page.
10	Navigation to Car Rentals Page	Test navigation from the homepage to the Car Rentals page via the navigation bar.	Selecting "Car Rentals" from the navigation bar should redirect the user to the Car Rentals page.
11	Navigation to Special Offers Page	Test navigation from the homepage to the Special Offers page via the navigation bar.	Selecting "Special Offers" from the navigation bar should redirect the user to the Special Offers page.
12	Navigation to Customer Support Page	Test navigation from the homepage to the Customer Support page via the navigation bar.	Selecting "Customer Support" from the navigation bar should redirect the user to the Customer Support page.
13	Responsive Design Check	Verify the homepage's responsiveness on different devices.	The homepage should render correctly and remain fully functional on desktops, tablets, and smartphones.

Obrázek 10- (vlastní práce)

Ukázkové testovací scénáře, jako je navigace na stránku letů, hotelů, pronájmu aut, speciálních nabídek a zákaznické podpory, jsou každý koncipovány tak, aby testovaly specifické funkce a cesty uživatelů. Toto detailní rozdělení usnadňuje práci člověku, jenž bude dělat ze scénářů testovací případy, ale na druhou stranu by bylo pro člověka podstatně

pracnější. Umělá inteligence produkuje text, ale tak rychle že to nepovažuji za špatnou vlastnost těchto scénářů.

5.3 Zaměření na specifické případy užití

Zatímco výše uvedený přístup může být považován za příliš granulární a časově náročný, jeho výhodou je, že umožňuje testovacím týmům podrobně porozumět jednotlivým funkcím aplikace a způsobům, jakými jsou tyto funkce prověřeny. Nicméně je důležité si uvědomit, že takto zúžený pohled může vést k situacím, kdy některé chyby zůstanou nedetekovány, protože testy byly psány opravdu pouze pro dané případy užití.

V případě aplikace umělé inteligence, jako je GPT-4.0, při tvorbě testovacích scénářů můžeme narazit na omezení v schopnosti provádět širší syntézu úloh, což je běžně v rámci lidských schopností. Například, zatímco lidský tester by intuitivně navázal scénář pro přidávání recenzí na hotelové stránce se scénářem pro jejich čtení, GPT-4.0 nemusí automaticky učinit stejný deduktivní skok. To může být důsledkem toho, že AI generuje scénáře založené striktně na definovaných případech užití bez přihlédnutí k širším souvislostem. V této konkrétní situaci GPT-4.0 sice vytvořil podrobné scénáře pro dané úkony, ale nedokázal plně zohlednit všechny potenciální interakce uživatelů, což může vést k přehlédnutí důležitých testovacích případů. To poukazuje na potřebu integrace AI s analytickým myšlením a předvídavostí, které jsou charakteristické pro lidské testery, a zdůrazňuje význam jejich role v procesu testování.

Je pravda, že pokud bychom přistupovali k formulaci promptů pro GPT-4.0 méně strukturovaným způsobem, než je definice případů užití, mohlo by to vést k vytváření testovacích scénářů z širší perspektivy. Přestože by takový přístup mohl teoreticky stimulovat generování testů s větším rozsahem funkcionalit, současně by se zvyšovalo riziko, že některé navrhované testy budou neplatné nebo irelevantní vzhledem k funkcím skutečně dostupným na testované stránce.

GPT-4.0 by mohl, v reakci na méně konkrétní prompt, navrhovat testovací scénáře, které neodpovídají aktuálnímu stavu nebo schopnostem aplikace. To by mohlo vyústit ve ztrátu času a zdrojů při zkoušení funkcionalit, které aplikace nenabízí, nebo které nejsou v rámci předpokládaného uživatelského prostředí relevantní.

Proto je klíčové najít rovnováhu mezi poskytnutím dostatečného vedení AI pro generování relevantních a přesných testů a zároveň zachováním flexibility potřebné pro identifikaci méně zřejmých, ale potenciálně kritických, testovacích scénářů. Tento přístup může zahrnovat kombinaci dobře definovaných případů užití s obecnějšími otázkami nebo podněty, které mohou AI podnítit k uvažování o širších souvislostech a možných interakcích, jež by jinak mohly zůstat opomenuty.

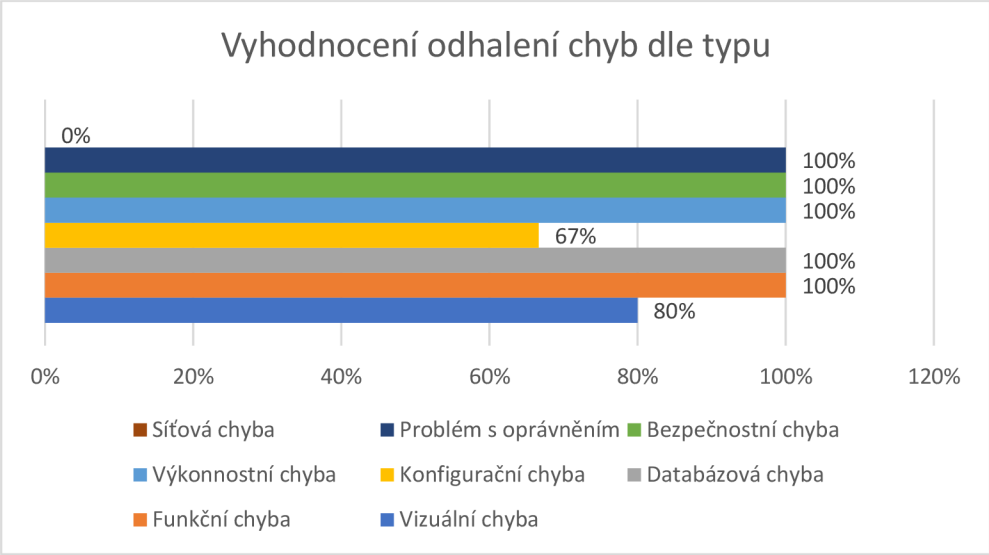
5.3.1 Závislost na kvalitě zadání

Výsledky ukázaly, že úspěšnost detekce chyb značně závisí na detailnosti a specifičnosti zadání poskytnutého GPT-4.0. Absence explicitních instrukcí pro určité testovací scénáře může vést k přehlédnutí relevantních chyb. Tento aspekt vyžaduje zvýšenou pozornost při přípravě zadání pro generování scénářů.

5.4 Hodnocení úspěšnosti GPT-4.0

5.4.1 Celková úspěšnost

Experiment ukázal, že GPT-4.0 byl schopen identifikovat přibližně 88,9% z navržených chyb (24 z 27), což demonstruje vysokou účinnost tohoto AI modelu v detekci softwarových chyb prostřednictvím automatizovaně generovaných testovacích scénářů. Celková doba potřebná k generování testovacích scénářů pro všechny stránky byla zhruba 1,5 hodiny, což je výrazně méně času, než by bylo potřeba pro manuální generování ekvivalentního množství scénářů, odhadem 2-3 člověkodny.



Obrázek 11- (vlastní práce)

5.4.2 Hodnocení nalezení chyby

Pro účinné hodnocení schopnosti scénářů odhalovat potenciální chyby bylo nezbytné přistupovat k této úloze s relativní přísností. Aby byl scénář klasifikován jako úspěšný v nalezení chyby, musel explicitně zmínit komponentu, které se chyba týkala. Očekávalo se, že zkušený tester by na základě takového scénáře mohl chybu snadno identifikovat. V případě, že byla definice scénáře příliš vágní a nebylo jasné, zda by chyba byla odhalena, daná chyba nebyla hodnocena jako nalezená.

5.4.3 Bezpečnostní chyby

V rámci testovací fáze mé bakalářské práce byla implementována kontrola jedné bezpečnostní chyby, exponovaná platební brána, která byla úspěšně detekována. Zajímal jsem se také o to, zda vytvořené testovací scénáře pokryjí běžné typy útoků jako SQL injection a XSS, které patří mezi relativně časté metody útoků na webové stránky. Testovací scénáře zaměřené na prevenci těchto útoků byly generovány pro tři ze čtyř testovaných stránek. Tento výsledek zdůrazňuje význam neustálého dohledu ze strany odborníků a potřebu doplnění testů v případě, že by stránka obsahovala tuto bezpečnostní slabinu a scénáře by tuto chybu neodhalily.

Pominutí takového problému by mohlo vést k úniku citlivých dat a jejich potenciální ztrátě. Je tedy zřejmé, že i při využití pokročilých AI systémů v procesu generování

testovacích scénářů je klíčová kombinace automatizovaných testů a lidského přístupu k zajištění maximální možné bezpečnosti.

5.4.4 Funkční chyby

Bylo identifikováno všech 14 předpokládaných funkčních chyb. GPT-4.0 prokázal dobré porozumění očekávané funkcionalitě aplikací, což ukazuje na jeho potenciál ve vyhledávání a identifikaci funkčních problémů. Přestože by výsledky mohly být různé pro aplikace s vysokým stupněm unikátnosti, v tomto testování byla potvrzena schopnost GPT-4.0 detekovat funkční chyby v běžných aplikacích.

Zvláštní pozornost si zaslouží chyba s identifikačním číslem 16, která umožňuje výběr dat v minulosti – což je častý problém u komponent umožňujících výběr data. Tento scénář byl GPT-4.0 úspěšně odhalen, což ilustruje schopnost modelu rozpoznat a pochopit důležitost logického omezení kalendářových komponent, které by měly umožňovat výběr pouze relevantních, budoucích datových intervalů.

5.4.5 Databázové chyby

Databázová chyba byla jenom jedna, ta se projevovala problémem s aktualizací databáze po rezervaci hotelového pokoje, specifický scénář, navržený za účelem kontroly aktualizace dostupnosti pokojů, úspěšně tento problém odhalil.

Je však důležité si uvědomit, že méně zřejmé databázové chyby, které se nemusí projevovat tak očividně, by mohly modelu GPT-4.0 uniknout. Tento fakt poukazuje na nezbytnost synergie mezi člověkem a umělou inteligencí. Lidský prvek v procesu testování zůstává klíčový pro zajištění komplexnosti a hloubky testů, zejména při odhalování a opravování chyb s méně jasným projevem.

5.4.6 Konfigurační chyby

Konfigurační chyby byly v rámci mé práce odhaleny s úspěšností dvě třetiny. Příkladem je situace, kdy zastaralá knihovna vedla k pádu aplikace po uživatelově kliknutí na obrázek, což bylo umělou inteligencí identifikováno s relativní lehkostí. Nicméně celková úspěšnost detekce konfiguračních chyb poukazuje na určité nedostatky v schopnostech AI odhalovat tyto specifické typy chyb.

5.4.7 Problém s oprávněním

GPT 4.0 úspěšně identifikovalo chyby oprávnění uživatelů při přidávání recenzí. AI vytvořila testovací scénář, který byl schopen odhalit problémy v procesu recenzování. Scénář definovaný AI umožňoval uživatelům vybrat si minulou rezervaci, následně vložit numerické hodnocení a písemnou zpětnou vazbu, a potvrdit, že tyto kroky mohou být úspěšně dokončeny a recenze odeslána.

5.4.8 Síťová chyba

GPT-4.0 nebyl schopný detekovat síťovou chybu podrobný popis a odůvodnění je v sekci Nedostatky v pokrytí testů.

5.4.9 **Výkonnostní chyba**

V této oblasti byla chyba také odhalena, což lze částečně přičíst skutečnosti, že byly v zadání pro GPT-4.0 explicitně požadovány výkonnostní testy. To podtrhuje význam pečlivě formulovaných instrukcí, které jsou předkládány umělé inteligenci. Správně specifikované požadavky jsou klíčové pro úspěšné využití AI v procesu testování a zdůrazňují potřebu jasné a přesné komunikace s těmito systémy, aby byly dosaženy očekávané výsledky.

5.4.10 **Vizuální chyba**

V sekci týkající se detekce vizuálních chyb bylo zjištěno, že GPT-4.0 dosáhl úspěšnosti 80% při identifikaci těchto chyb, což je považováno za silný výsledek. Nicméně stále existuje prostor pro zlepšení a je pravděpodobné, že inovativní přístupy v generování testů by mohly vést k ještě lepší detekci.

Bylo by zvláště zajímavé zaměřit se na vizuální chyby izolovaně a provést porovnání výsledků s alternativní metodou testování, kde by se GPT-4.0 poskytly HTML a CSS dokumenty, jež definují danou webovou stránku. Takový přístup by mohl modelu umožnit hlubší porozumění struktuře a stylu stránky a potenciálně odhalit jemnější vizuální nedostatky, které by mohly být přehlédnuty při tradičních testech. Tato metodologie by mohla umožnit GPT-4.0 lépe simulovat vizuální zpracování, které provádí uživatelé při interakci se stránkou, a nabídnout tak holistický pohled na uživatelskou zkušenost.

5.5 Diskuse

GPT-4.0 měl zejména problémy s chybami vyžadujícími hlubší kontextové porozumění a chybami souvisejícími s vizuálními aspekty aplikací. Tyto nálezy poukazují na to, že přestože GPT-4.0 exceluje v mnoha oblastech, jeho schopnosti nejsou všestranné a jisté typy chyb unikají jeho detekčním algoritmům. Například, chyby, které jsou závislé na uživatelské interakci nebo estetice UI, nemusí být model AI schopen adekvátně vyhodnotit, protože mu chybí schopnost vizuálního vnímání a subjektivního hodnocení, které je pro tyto úkoly klíčové.

Tato omezení naznačují, že pro zvýšení efektivity využití GPT-4.0 v testovacích procesech je zapotřebí dalšího zlepšování v oblasti zadávání úkolů a jeho instrukcí. Může to zahrnovat specifičtější a detailnější popis situace a kontextu chyby, aby se usnadnilo modelu lepší porozumění problému. Dále to také ukazuje na možnou potřebu integrace GPT-4.0 s jinými nástroji, které se specializují na vizuální testování, nebo s metodami, které mohou lépe simulovat uživatelskou interakci.

Kombinace GPT-4.0 s dalšími technologiemi a testovacími přístupy by mohla vést k vytvoření komplexnějšího a robustnějšího testovacího frameworku, který by mohl kompenzovat tyto identifikované slabiny. Výzkum v této oblasti by měl pokračovat, s cílem vytvářet synergetické nástroje, které využívají nejlepší vlastnosti AI i lidských testerů, aby se maximalizovala kvalita a bezpečnost softwarových produktů.

5.6 Časové úspory

Významným přínosem použití GPT-4.0 je efektivita a časové úspory při generování testovacích scénářů. Tento přístup nejen zkracuje celkový čas potřebný k přípravě testovacího procesu, ale také umožňuje rychlejší iterace a zlepšování softwaru. Doba, která by byla potřebná k doplnění scénářů pro dosažení plného pokrytí chyb, je odhadována na 0,5 až 1 člověkoden, což je stále výrazně efektivnější ve srovnání s úplně manuálním procesem.

5.7 Nedostatky v pokrytí testů

Z 27 očekávaných chyb byly pomocí vygenerovaných scénářů identifikovány pouze 24, což nám dává úspěšnost detekce na úrovni přibližně 88,9%. I když by se mohlo zdát, že je to poměrně vysoká úspěšnost, tento výsledek nám zároveň ukazuje omezení GPT-4.0.

5.7.1 Chyba číslo 13 - Načítání recenzí z externího API

Popis a kontext chyby

Chyba spojená s načítáním recenzí z externího API. Tato chyba, pokud by byla přítomna ve skutečné aplikaci, by mohla mít značné negativní důsledky pro uživatelskou zkušenost, neboť některé recenze by nebyly zobrazeny.

Specifikace chyby a důsledky

Chyba spočívala v tom, že systém, místo aby korektně zobrazil všechny recenze, by některé z nich přehlédl. To by vedlo k situaci, kde uživatelé nemají přístup ke všem recenzím, což by mohlo negativně ovlivnit jejich rozhodovací proces a důvěru v aplikaci.

Analýza absence detekce chyby GPT-4.0

GPT-4.0 obdržel případ užití zaměřený na psaní recenzí, avšak instrukce pro čtení recenzí chyběly. Tato specifika zadání odhalila klíčové omezení v přístupu k testování GPT-4.0, jako generátor textu, vykazuje neschopnost širokého pojetí bez explicitně

definovaných scénářů. Tento případ ukazuje, že přesná specifikace úkolů je pro úspěšnou generaci testovacích scénářů nezbytná, jelikož model má tendenci přehlížet aspekty, které nejsou jasně určeny.

Doporučení pro zlepšení

Pro zlepšení detekce chyb v podobných situacích by bylo vhodné rozšířit zadání pro AI modely o všechny možné aspekty funkcionality aplikace, včetně těch, které se mohou zdát samozřejmé. Integrace podrobnějších a komplexnějších případů užití může pomoci překlenout mezeru mezi omezeními generátorů textu a potřebami komplexního softwarového testování.

5.7.2 Chyba číslo 26 - Vizuální překrytí textu

Popis a kontext chyby

Chyba číslo 26, se týkala vizuálního překrytí textu v aplikaci.

Důsledky chyby a analýza

Vizuální chyby, jako je překrytí textu, mohou vážně ovlivnit uživatelský dojem a použitelnost aplikace. V tomto případě GPT-4.0 dokázal na jiných stránkách generovat scénáře pro kontrolu vizuálních aspektů, ale selhal zde, což odhaluje nekonzistentnost v generování scénářů pro vizuální testování. Tato nekonzistentnost může být připsána variabilitě v interpretaci zadání modelu.

Doporučení pro zlepšení

Zlepšení v detekci vizuálních chyb by mohlo spočívat v explicitnějším formulování zadání pro vizuální testy a v integraci vizuálních testovacích nástrojů, které by mohly model GPT-4.0 doplnit. Vzhledem k omezení textově založených AI modelů by mělo být zváženo začlenění lidských testerů pro finální kontrolu vizuálních aspektů aplikace.

5.7.3 Chyba č. 24 - Resetování preferencí vozidla

Detaily chyby a její důsledky

Chyba zahrnuje resetování uživatelsky definovaných preferencí vozidla, jako jsou model vozidla nebo přídavné vybavení, když uživatel naviguje pryč ze stránky a poté se na ni vrátí. To může vést k frustraci uživatelů, protože je nutí opakovat své výběry, což snižuje celkovou snadnost použití a efektivitu webu.

Analýza nepodařené detekce chyby

Nejbližší scénář, který byl pro tuto funkčnost vygenerován (ID scénáře 10), se zaměřoval na možnost uživatelů uložit si kritéria vyhledávání pronájmu pro budoucí návštěvy, avšak nepokrýval konkrétně situaci, kdy se preferované nastavení vozidla resetuje po opuštění a návratu na stránku. Toto naznačuje, že i přes schopnost GPT-4.0 generovat užitečné testovací scénáře, jeho úspěch silně závisí na specifičnosti a úplnosti zadání, které obdrží.

Doporučení pro zlepšení

Zlepšení detekce takových chyb by vyžadovalo detailnější zadání testovacích scénářů, které explicitně zahrnují různé uživatelské interakce s webem, včetně navigace pryč a zpět. Kombinace širšího spektra testovacích scénářů, které zahrnují specifické uživatelské chování, spolu s pravidelným přezkumem a aktualizací těchto scénářů, by mohla výrazně zvýšit schopnost AI identifikovat podobné chyby v budoucnosti.

Závěr

V této bakalářské práci jsem se věnoval analýze a využití modelu GPT-4.0 pro automatizované generování testovacích scénářů softwarových aplikací. V teoretické části jsem zhodnotil dosavadní postupy ve vytváření testovacích scénářů. Následně jsem zanalyzoval, jakým způsobem nejlépe zadávat pokyny ke generování pomocí umělé inteligence. Praktická část práce ukázala, že GPT-4.0 má významný potenciál pro identifikaci a detekci softwarových chyb, což bylo prokázáno úspěšnou detekcí 88,9% z předem definovaných chyb v testované aplikaci. Výsledky tak potvrzují hypotézu, že využití pokročilých modelů umělé inteligence může výrazně přispět k efektivitě a rychlosti testovacích procesů, což je v současném rychlém vývoji softwaru nezbytné.

Doba potřebná k vygenerování testovacích scénářů pomocí GPT-4.0 byla výrazně kratší ve srovnání s tradičními manuálními metodami, což představuje významné časové a potažmo ekonomické úspory. Tato efektivita umožňuje testovacím týmům rychleji reagovat na potřeby vývojových cyklů a zvyšuje možnosti iterativního vylepšování a optimalizace softwarových produktů.

Přestože výsledky experimentu jsou povzbudivé, odhalily také určité omezení a výzvy spojené s používáním AI v procesu testování. Specificky, absenci detekce některých chyb, což může být důsledkem omezení v zadáních pro AI nebo inherentních omezeních modelu GPT-4.0 v kontextu softwarového testování. Tyto zjištěné nedostatky zdůrazňují potřebu kombinace AI s tradičními metodami testování a důležitost supervize lidskými experty, aby bylo dosaženo maximálního pokrytí a kvality testování.

Vzhledem k rychlému pokroku v oblasti umělé inteligence a jejím aplikacím je jasné, že budoucnost softwarového testování bude značně ovlivněna těmito technologiemi. Další výzkum a vývoj v této oblasti by měl směřovat k optimalizaci integračních procesů mezi AI a testovacími metodikami, zlepšení přesnosti a spolehlivosti AI generovaných testů, a rozšíření schopností AI modelů k detekci širšího spektra chyb.

Zájem o tuto problematiku a potenciál umělé inteligence v oblasti testování softwaru mě osobně fascinuje a hodlám se tématu dále věnovat i v budoucnu. Neustálý vývoj v oblasti velkých jazykových modelů, jako je GPT-4.0, a jejich aplikací v praxi otevírá nové možnosti a výzvy, které jsou nejen akademicky zajímavé, ale mají i přímý dopad na průmyslové procesy. Je nezbytné tento vývoj dále zkoumat, aby bylo možné plně využít potenciál AI v testování a zároveň identifikovat a řešit její omezení a výzvy.

V závěru je třeba zdůraznit, že přestože využití modelů umělé inteligence, jako je GPT-4.0, představuje slibný pokrok ve způsobech, jakými přistupujeme k testování softwaru, lidský faktor zůstává nezbytným komponentem v procesu zajištění kvality softwaru. Synergie mezi umělou inteligencí a lidskými odborníky nabízí nejlepší cestu k dosažení vysoce kvalitního, efektivního a bezpečného softwaru, který vyhovuje požadavkům a očekáváním uživatelů ve stále se vyvíjejícím digitálním světě.

6 Seznam použitých zdrojů

1. BATTINA, Dhaya Sindhu. Artificial Intelligence in Software Test Automation: A Systematic Literature Review. *JETIR*. 2019, č. 6, s. 1329-1332. ISSN 2349-5162.
2. CATOLINO, Gemma; PALOMBA, Fabio; ZEIDMAN, Andy a FERRUCCI, Filomena. Not All Bugs Are the Same: Understanding, Characterizing, and Classifying the Root Cause of Bugs. Online. *Journal of Systems and Software*. 2019, article 152, s. 165-181. Dostupné z: https://doi.org/10.1016/j.jss.2019.03.002. [cit. 2024-02-10].
3. CODEMIFY. *ChatGPT for QA: how to use*. Online. 2023. Dostupné z: https://www.youtube.com/watch?v=iOheLWYbILw&t=1860s. [cit. 2024-02-10].
4. EKIN, Sabit. Prompt Engineering For ChatGPT: A Quick Guide To Techniques, Tips, And Best Practices. Online. 2023, s. 11. Dostupné z: https://doi.org/10.36227/techrxiv.22683919.v2. [cit. 2024-02-15].
5. GOODFELLOW, I., BENGIO, Y., & COURVILLE, A. (2016). *Deep Learning*. MIT Press. Dostupné z: http://www.deeplearningbook.org/
6. PATTON, Ron. *Software Testing*. USA: Sams Publishing, 2001. ISBN 0-672-31983-7.
7. PAPERSWITHCODE.COM. *Code Generation on MBPP*. Online. Dostupné z: https://paperswithcode.com/sota/code-generation-on-mbpp. [cit. 2024-03-1].
8. POLYUNO. *Types of Bugs in Software Testing*. Online. 2.2.2021. Dostupné z: https://polyuno.com/blog/types-of-bugs-in-software-testing/. [cit. 2024-02-10].
9. RADFORD, A., WU, J., CHILD, R., LUAN, D., AMODEI, D., & SUTSKEVER, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*. Dostupné z: https://openai.com/blog/better-language-models/
10. JÖCKEL, L. *et al.* (2021) "Towards a common testing terminology for software engineering and data science experts," in *Product-Focused Software Process Improvement*. Cham: Springer International Publishing (Lecture notes in computer science), pp. 281–289. doi: 10.1007/978-3-030-91452-3_19.
11. KARTHIKEYAN, Dr.J. *Learning outcome of a classroom reasearch*. India: L ORDINE NUOVO PUBLICATION, 2021. ISBN 978-93-92995-15-6.
12. LEE, J., KANG, S. a JUNG, P. (2020) "Test coverage criteria for software product line testing: Systematic literature review," Information and software technology. Elsevier BV, 122(106272), p. 106272. doi: 10.1016/j.infsof.2020.106272.
13. MARIE, Benjamin. Mixtral-8x7B: Understanding and Running the Sparse Mixture of Experts: How to efficiently outperform GPT-3.5 and Llama 2 70B. Online. In: . 2023. Dostupné z: https://towardsdatascience.com/mixtral-8x7b-understanding-and-running-the-sparse-mixture-of-experts-0e3fc7fde818. [cit. 2024-02-10].
14. META, GENAI. Llama 2: Open Foundation and Fine-Tuned Chat Models. Online. 2023. Dostupné z: https://arxiv.org/pdf/2307.09288.pdf. [cit. 2024-02-10].
15. MISTRAL AI TEAM. Mixtral of experts: A high quality Sparse Mixture-of-Experts. Online. In: . Dostupné z: https://mistral.ai/news/mixtral-of-experts/. [cit. 2024-03-10].

16. OPENAI. Online. 2023. Dostupné z: https://arxiv.org/pdf/2303.08774.pdf. [cit. 2024-02-10]. Please cite this work as “OpenAI (2023)”
17. TEST.IO ACADEMY. *Functional Bugs*. Online. 2024, 2024-02-29. Dostupné z: https://www.tutorialspoint.com/software_testing_dictionary/failure.htm. [cit. 2024-03-09].
18. TUTORIALSPOINT. *Defect*. Online. Dostupné z: https://www.tutorialspoint.com/software_testing_dictionary/defect.htm. [cit. 2024-02-03].
19. TUTORIALSPOINT. *Failure*. Online. Dostupné z: https://www.tutorialspoint.com/software_testing_dictionary/failure.htm. [cit. 2024-02-03].
20. VASWANI, A., SHAZEER, N., Parmar, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł., & POLOSUKHIN, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (str. 5998-6008). Dostupné z: https://arxiv.org/abs/1706.03762
21. WEI, Jason a , et al. Emergent Abilities of Large Language Models. Online. *Transactions on Machine Learning Research*. Roč. 2022, č. 8. Dostupné z: https://arxiv.org/pdf/2206.07682.pdf. [cit. 2024-02-10].

7 Seznam obrázků, tabulek, grafů a zkratek

7.1 Seznam obrázků

7.2 Seznam použitých zkratek

AI - artificial intelligence (umělá inteligence)

API – application programming interface (aplikační programové rozhraní)

GPT - generative pre-trained transformer (generativní předtrénovaný transformátor)

GUI – graphical user interface (grafické uživatelské rozhraní)

GQA – grouped-query attention (pozornost při skupinovém dotazu)

LLM – large language model (velký jazykový model)

URL - uniform resource locator (jednotný lokátor zdroje)

XML - extensible markup language (rozšiřitelný značkovací jazyk)

Přílohy

Příloha A: Případy užití
Popis: Dokument obsahující detailní popis případů užití. Dokument je formátován v Microsoft Word a poskytuje komplexní přehled funkcionalit a interakcí uživatelů se systémem v anglickém jazyce.
Umístění: Příloha v digitální formě vložená do sekce příloh této práce pod názvem Případy užití.doc.

Příloha B: Vygenerované scénáře
Popis: Tabulka ve formátu Microsoft Excel obsahující scénáře vygenerované pro testování systému XYZ. Každý scénář je podrobně popsán včetně očekávaných vstupů a výstupů.
Umístění: Příloha v digitální formě vložená do sekce příloh této práce pod názvem "Vygenerované scénáře.xlsx".