

Využití strojového učení v procesu vývoje software

Dominik Machalík



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	Dominik Machalík
Osobní číslo:	A21342
Studijní program:	B0613A140020 Softwarové inženýrství
Forma studia:	Prezenční
Téma práce:	Využití strojového učení v procesu vývoje software
Téma práce anglicky:	Using Machine Learning in the Software Development Process

Zásady pro vypracování

1. Popište princip fungování strojového učení a možnosti jeho uplatnění v různých odvětvích vývoje software.
2. Vytvořte přehled nástrojů a technologií založených na strojovém učení, které je aktuálně možné využít v procesu vývoje software.
3. Vytvořte konkrétní případovou studii vývoje software, kde budou použity vybrané nástroje z přehledu výše. Vhodně popište způsob jejich aplikace na konkrétních případech užití.
4. Zhodnoťte přínosy v oblasti efektivity a urychlení vývojového procesu ve srovnání s klasickým přístupem bez využití zmíněných nástrojů.
5. Vyhodnoťte také možná rizika navrženého přístupu (bezpečnost, bias, soukromí...) a zvažte budoucí vývoj či potenciální rozšíření vámi navrženého řešení.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. LINHART, Zdeněk. Úvod do deep learning. Vysoká škola ekonomie a managementu, 2021. ISBN 9788088330196.
2. GHOSH, Pinaki, S. KIRAN, J. MAHALAKSHMI a SK ALTHAF HUSSAIN BASHA. 2023. Understanding Machine Learning [elektronická kniha] AG Publishing House (AGPH Books), 220 stran. Dostupné z: https://www.google.cz/books/edition/Understanding_Machine_Learning/ufC9EAAAQBAJ?hl=cs&gbpv=0&kptab=overview
3. MIKELSTEN, Daniel, TEIGENS, Vasil a SKALFIST, Peter. Umělá inteligence: Čtvrtá průmyslová revoluce [online]. Cambridge Stanford Books, 2020 [cit. 2023]. Dostupné z: <https://www.everand.com/book/476431120/Um%C4%9Bla-intelligence-%C4%8Ctvrta-pr%C5%AFmyslova-revoluce>
4. CHELOUAH, Rachid. Optimization and Machine Learning: Optimization for Machine Learning and Machine Learning for Optimization. 1st ed. Wiley, 2022. ISBN 978-1-78945-071-2.
5. ZELINKA, Ivan. Umělá inteligence: hrozba nebo naděje? Praha: BEN – technická literatura, 2003. ISBN 8073000687.
6. ALPAYDIN, Ethem. Introduction to machine learning. 2nd ed. Massachusetts: MIT Press, c2010. ISBN 9780262012430.
7. COECKELBERGH, Mark. Etika umělé inteligence. Přeložil Sylva FICOVÁ. Dnešní svět. Praha: Filosofia, 2023. ISBN 978-80-7007-746-7.
8. OLEJ, Vladimír a HÁJEK, Petr. Úvod do umělé inteligence: klasická umělá inteligence : distanční opora. Pardubice: Univerzita Pardubice, 2009. ISBN 9788073952419.

Vedoucí bakalářské práce:

Ing. Radek Vala, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce:

5. listopadu 2023

Termín odevzdání bakalářské práce:

13. května 2024

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan



prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 5. ledna 2024

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.
- že při tvorbě této práce jsem použil nástroje generativních modelů AI:
 - ChatGPT 4; <https://chat.openai.com/>
 - GitHub Copilot; <https://github.com/features/copilot>
 - Google Gemini; <https://gemini.google.com/>
 - Uizard; <https://uizard.io/>
- Nástroje byly využity za účelem ukázkového generování kódu, uživatelského rozhraní a návrhu aplikace, jelikož se tomuto tématu tato práce věnuje. Také byly využity při vyhledávání a zpracování informací z Internetu.

- Veškerý kód vygenerovaný nástroji generativních modelů AI je v práci označen popiskem „Vygenerovaný kód“ a na konci práce se nachází seznam všech těchto kódů. Tento kód není započítán v celkové délce práce.
- Po použití těchto nástrojů jsem provedl kontrolu obsahu a přebírám za něj plnou zodpovědnost.

Ve Zlíně, dne 9. května 2024

Dominik Machalík, v. r.

podpis studenta

ABSTRAKT

Tato práce se zabývá využitím nástrojů strojového učení při vývoji software. V teoretické části jsou představeny základní principy fungování strojového učení, jsou představeny největší jazykové modely a aplikace, které nad nimi běží. V praktické části jsou některé tyto nástroje využity pro návrh funkcionalit, tvorbu uživatelského rozhraní, psaní kódu a tvorbu automatických testů. Dále se soustředí na zhodnocení výstupů a popsání přínosů a rizik spojených s využitím těchto nástrojů.

Klíčová slova: Strojové učení, Umělá inteligence, Vývoj software, Efektivita, Kvalita software, ChatGPT, GitHub Copilot

ABSTRACT

This thesis focuses on the use of machine learning tools in software development. In the theoretical part, the basic principles of machine learning are introduced, the most common language models and applications that run on top of them are presented. In the practical part, some of these tools are used for creating functionalities, creating user interfaces, writing code and creating automated tests. In the end it focuses on evaluating the outputs and describing the benefits and risks associated with using these tools.

Keywords: Machine learning, Artificial intelligence, Software development, Efficiency, Software quality, ChatGPT, GitHub Copilot

Tímto bych chtěl poděkovat panu Ing. Radku Valovi, Ph. D. za jeho ochotnou pomoc a připomínky při tvorbě práce a také za přijetí vlastního tématu práce. Dále bych chtěl poděkovat své rodině, přítelkyni a přátelům, kteří mi byli při tvorbě práce oporou.

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST.....	10
1. STROJOVÉ UČENÍ.....	11
1.1. ZÁKLADNÍ PRINCIPY FUNGOVÁNÍ STROJOVÉHO UČENÍ.....	11
1.2. TYPY STROJOVÉHO UČENÍ	11
1.2.1. Učení s učitelem	11
1.2.2. Učení bez učitele	12
1.2.3. Učení s částečným dohledem	12
1.2.4. Zpětnovazebné učení	12
1.3. GENERATIVNÍ UMĚLÁ INTELIGENCE	12
1.4. VELKÝ JAZYKOVÝ MODEL	12
1.4.1. Token.....	13
1.4.2. GPT	15
1.4.3. Gemini.....	15
1.4.4. Llama.....	15
2. NÁSTROJE ZALOŽENÉ NA STROJOVÉM UČENÍ	16
2.1. CHATBOTI	16
2.1.1. Kontext.....	16
2.1.2. Prompt	16
2.1.3. ChatGPT.....	16
2.1.4. Gemini.....	17
2.1.5. Claudie	18
2.1.6. Uizard.....	18
2.1.7. GitHub Copilot Workspace	19
2.2. POROVNÁNÍ CHATBOTŮ	19
2.3. IMPLEMENTACE NÁSTROJŮ A TECHNOLOGIÍ STROJOVÉHO UČENÍ DO VÝVOJOVÝCH PROSTŘEDÍ	20
2.3.1. GitHub Copilot.....	20
2.3.1.1. GitHub Copilot Chat.....	20
2.3.1.2. GitHub Copilot – Code completion	20
2.3.2. Amazon Q Developer.....	21
2.3.3. JetBrains AI	21
3. PŘÍNOSY A RIZIKA SPOJENÁ S VYUŽITÍM STROJOVÉHO UČENÍ VE VÝVOJI SOFTWARE	22
3.1. SPOKOJENOST VÝVOJÁŘE	22
3.2. RYCHLOST TVORBY APLIKACE	22
3.3. HLEDÁNÍ INFORMACÍ.....	22
3.4. KVALITA GENEROVANÝCH ODPOVĚDÍ	23
3.5. HALUCINACE.....	23
3.6. ZTRÁTA KONTEXTU	23
3.7. ZPŘÍSTUPNĚNÍ DAT JINÝM SPOLEČNOSTEM	23
3.8. NAHRAZENÍ ČLOVĚKA V PRACOVNÍM PROCESU	24
II PRAKTICKÁ ČÁST	25

4. PŘÍPADOVÁ STUDIE S VYUŽITÍM STROJOVÉHO UČENÍ	26
4.1. PODROBNOSTI O POUŽÍVANÝCH NÁSTROJÍCH A TECHNOLOGIÍCH.....	26
4.1.1. ChatGPT.....	26
4.1.2. GitHub Copilot.....	26
4.1.3. Uizard.....	26
4.2. PŘÍPRAVA PROSTŘEDÍ.....	26
4.3. PŘÍPADOVÁ STUDIE - TODO LIST	27
4.3.1. Návrh funkcionalit aplikace	27
4.3.1.1. Sestavení promptu pro vygenerování funkcionalit	27
4.3.1.2. Vygenerovaná odpověď dle promptu pro funkcionality.....	28
4.3.1.3. Zhodnocení vygenerovaného výsledku	28
4.3.2. Návrh uživatelského rozhraní.....	29
4.3.2.1. Sestavení promptu pro vygenerování uživatelského rozhraní	29
4.3.2.2. Vygenerovaný výstup dle promptu pro uživatelské rozhraní	30
4.3.2.3. Zhodnocení vygenerovaného uživatelského rozhraní.....	30
4.3.3. Vývoj aplikace – funkcionality	31
4.3.3.1. Sestavení promptu pro vygenerování kódu aplikace	32
4.3.3.2. Vygenerovaný výstup dle promptu pro tvorbu aplikace.....	32
4.3.3.3. Zhodnocení vygenerovaného kódu aplikace.....	41
4.3.4. Vývoj aplikace – uživatelské rozhraní	43
4.3.5. Úprava aplikace.....	46
4.3.5.1. Úprava výpisu priorit a deadlinu	46
4.3.5.2. Úprava práce se šítky.....	46
4.3.5.3. Úprava úkolů.....	47
4.3.6. Testování aplikace.....	47
4.3.6.1. Testování aplikace pomocí GitHub Copilota.....	48
4.3.6.2. Testování aplikace pomocí ChatGPT 4	48
4.3.6.3. Vygenerování testů	49
4.3.6.4. Zhodnocení vygenerovaných testů	49
5. ZHODNOCENÍ PŘÍNOSŮ VYUŽITÍ NÁSTROJŮ STROJOVÉHO UČENÍ.....	53
5.1. NAVRHOVÁNÍ FUNKCIONALIT APLIKACE	53
5.2. NAVRHOVÁNÍ UŽIVATELSKÉHO ROZHRAŇÍ APLIKACE	53
5.3. VÝVOJ APLIKACE	54
5.4. TESTOVÁNÍ APLIKACE	55
5.5. ROZŠÍŘITELNOST A BUDOUCÍ VÝVOJ	55
ZÁVĚR	56
SEZNAM POUŽITÉ LITERATURY.....	58
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	61
SEZNAM VYGENEROVANÝCH KÓDŮ.....	62
SEZNAM OBRÁZKŮ	63
SEZNAM TABULEK.....	64
SEZNAM PŘÍLOH.....	65

ÚVOD

Od zpřístupnění chatbota ChatGPT pro širokou veřejnost v listopadu 2022 se začalo více rozšiřovat využití nástrojů strojového učení do mnoha průmyslových odvětví. Výjimkou nezůstalo ani odvětví zabývající se návrhem, vývojem a testováním software. V současné době jsou tyto nástroje zatím relativně nové a další se neustále objevují. Zatím se však příliš neprojeví přínosy a rizika spojené s využitím těchto nástrojů.

V teoretické části jsem si dal za cíl vysvětlit základní pojmy z oblasti strojového učení a také představit principy, které s tímto tématem souvisí. Dále představit největší jazykové modely a také aplikace nad nimi, které jsou v dnešní době používány nejen při vývoji software, ale i v mnoha dalších odvětvích. Práce také popisuje některé nástroje, které se při vývoji mohou používat. V neposlední řadě jsou představeny hlavní přínosy a rizika spojená s využitím těchto nástrojů.

Praktická část práce poté ukazuje případovou studii, která ukazuje, jak mohou být nástroje strojového učení prakticky využity při vývoji jednoduché webové aplikace pro správu úkolů. Z procesu vývoje software je zde návrh funkcionalit aplikace, tvorba uživatelského rozhraní, psaní kódu a tvorba automatizovaných testů.

Cílem práce je představit některé nástroje strojového učení využívané při vývoji software a ukázat, jak fungují. Představím přínosy spojené s jejich využitím během vývoje, ale zaměřím se i na body, které tyto nástroje příliš nezvládají a vysvětlím i další problémy a rizika spojená s vývojem s těmito nástroji.

I. TEORETICKÁ ČÁST

1. STROJOVÉ UČENÍ

Ada Lovelace ve své poznámce k vynálezu „Analytického stroje“ v roce 1843 poznamenala, že nemá nároky na to, aby něco vytvořil, ale že může vykonat cokoli, co je schopna definovat. Tento účel měla být podpora v oblasti matematické analýzy pomocí mechanických operací, což vysvětluje název stroje. Její poznámka se později stala citací pro Alana Turinga, který ve svém klíčovém článku z roku 1950 přinesl Turingův test a základní myšlenky o umělé inteligenci. Otázka, kterou Ada Lovelace položila a Turing rozvinul, se týkala toho, zda by počítače mohly být schopny učení a originality, a tím otevřela dveře pro koncept strojového učení. Tento nový přístup k programování se liší od klasického modelu, kde programátoři definují pravidla. Ve strojovém učení jsou data a očekávané odpovědi vstupem, z něhož se odvozují pravidla, která mohou být aplikována na nová data. Tento přístup posouvá hranice toho, co mohou počítače dělat, a klade otázku, zda by mohly být schopny překvapit svými schopnostmi. [1]

1.1. Základní principy fungování strojového učení

Strojové učení je odvětvím informatiky, které se zaměřuje na vývoj algoritmů schopných automaticky se učit a zlepšovat své schopnosti na základě dat. Tato data mohou být získána z různých zdrojů, včetně přírody, lidského vytváření nebo generování pomocí jiných algoritmů. Proces strojového učení lze chápat jako dvoufázový postup, kde se nejprve sbírají relevantní data a následně se na základě těchto dat vytváří statistický model či algoritmus, který je následně aplikován k řešení praktických problémů. Tento statistický model se předpokládá, že bude schopen adaptovat se na nové informace a zlepšovat své výsledky v průběhu času. [1]

1.2. Typy strojového učení

Dle způsobu, jak s těmito daty algoritmus pracuje se pak dělí strojové učení na několik typů. Mezi tyto typy patří učení s učitelem, učení bez učitele, učení s částečným dohledem a zpětnovazební učení.

1.2.1. Učení s učitelem

Učení s učitelem v oblasti strojového učení je postup, při kterém model trénuje na datech obsahujících jak vstupy, tak odpovídající výstupy. Jeho hlavním cílem je naučit se předpovídat správné výstupy pro nové vstupy. U tohoto postupu je potřeba kvalita poskytnutých

dat, které musí v dostatečné míře a hlavně v dostatečné kvalitě obsáhnout co nejvíce případů. Z toho důvodu může být tento přístup často nákladný. [2]

1.2.2. Učení bez učitele

Ve strojovém učení bez učitele se algoritmus snaží vyhledávat různé vzory či podobnosti ve vstupních datech bez poskytnutí jakéhokoliv výstupu jako u učení s učitelem. Mezi učení se dá také řadit například vizualizace dat. [3]

1.2.3. Učení s částečným dohledem

V případě učení s částečným dohledem jsou poskytnuta data, která jsou jak označená, tak i neoznačená. Využívá se hlavně v případech, kdy je k dispozici velké množství neoznačených dat a menší množství označených dat. Hlavní výhodou je snížení nákladů na označování dat, jelikož cílem této metody je umožnit trénování modelu s využitím malého množství označených dat na velké množství dat neoznačených. [4]

1.2.4. Zpětnovazebné učení

U zpětnovazebného učení se snaží model naučit co nejlepší výsledek na základě toho, jakou dostává zpětnou vazbu. Jedná se o formu skóre, kterou získává během toho, jak se postupně dostává k požadovanému výsledku. Tyto způsoby mohou být nefunkční či neefektivní, ale také mohou být správné. [5]

1.3. Generativní umělá inteligence

Generativní umělá inteligence je odvětví strojového učení, které umožňuje systémům učit se z dat a na základě nich generovat nové datové instance. Tento typ umělé inteligence se využívá k vytváření různých typů dat, jako jsou texty, obrázky, hudba nebo dokonce video, což je zásadní rozdíl oproti tradičním systémům AI, které se opírají o pevně stanovená pravidla a algoritmy a nemohou se učit ani se zlepšovat s časem. Generativní AI, díky své schopnosti učení z velkého množství dat, nachází uplatnění v různých aplikacích, od tvorby realistického uměleckého obsahu po vývoj autonomních systémů schopných imitace lidského chování a myšlení. [6]

1.4. Velký jazykový model

Velký jazykový model je algoritmus, který se učí na obrovském množství dat, nejčastěji na obsahu z Internetu. Z těchto dat se tvoří sady, na kterých se LLM učí pro svůj účel, například

pro vytvoření či vylepšení chatbota či jazykového překladače. Učí se například tím, že rozpozná, jak se některá slova ve větách používají za sebou. S větším množstvím poskytnutých dat se správnost použití dalšího slova zvětšuje. Model tak předpovídá další znak či znaky, které by se v daném textu mohly objevit. Pokud model nevybere správné slovo, změní se neuronová síť uvnitř modelu tak, aby se příště vybral správný znak či skupina znaků. [7]

1.4.1. Token

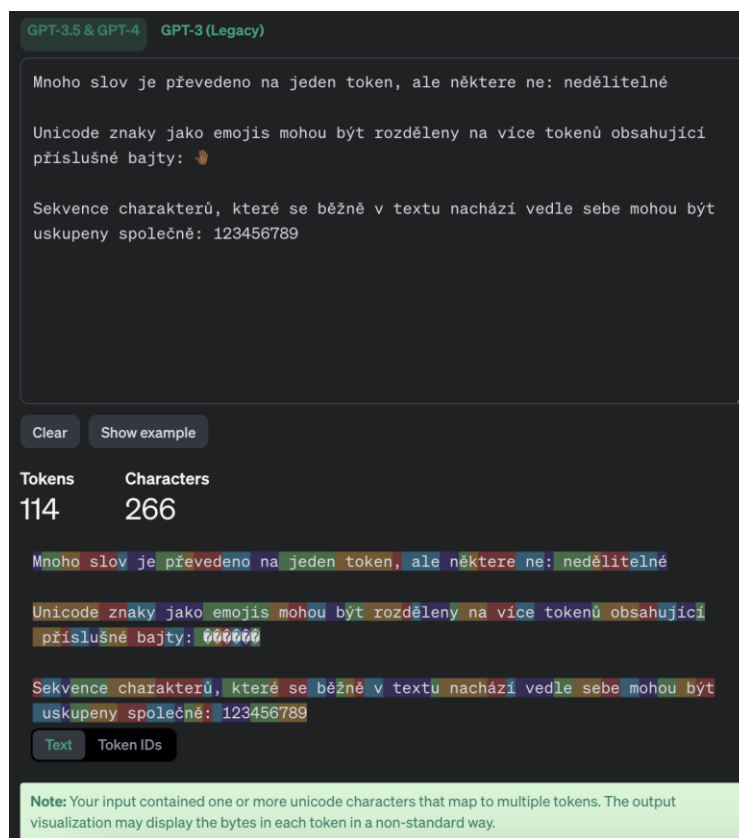
Velké jazykové modely nepracují v neuronové síti přímo se znaky, ale s takzvanými tokeny.

Čech, Rus a Němec přijdou do|

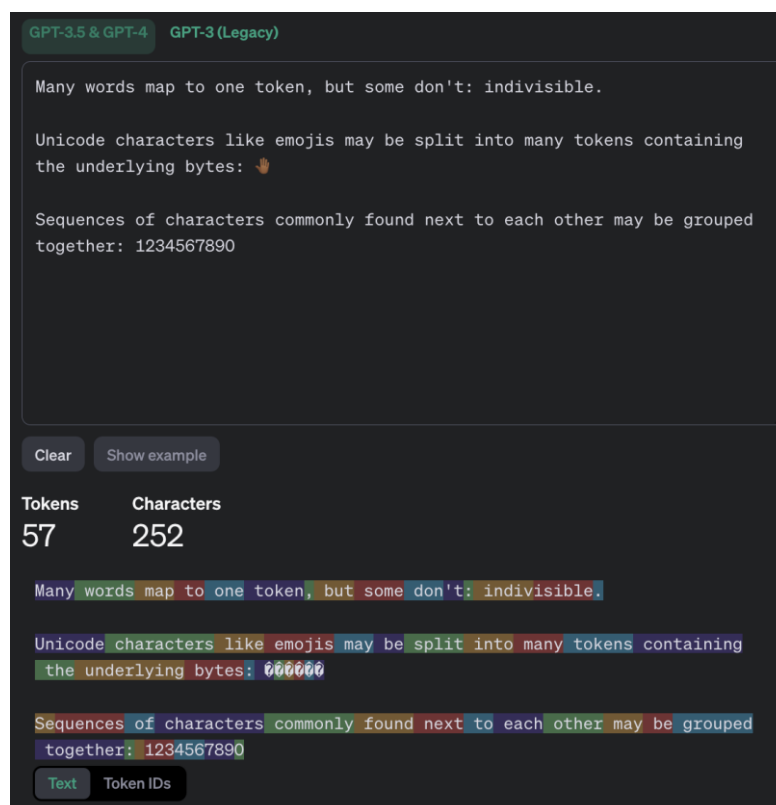


Obrázek 1 – Předpovídání dalšího tokenu [23]

V anglickém jazyce představuje token většinou jedno slovo, někdy jen jeho část. Naopak v češtině se jedná většinou o nižší jednotky znaků na jeden token, u písmen s diakritikou se může jednat dokonce i o jeden či více tokenů na znak. Firma OpenAI poskytuje volně dostupný nástroj Tokenizer, který po zadání textu vypíše jeho délku a počet tokenů u jeho LLM modelu se jménem GPT. [9] Jak jde vidět níže na obrázcích 2 a 3, tak při téměř stejné délce textu je u angličtiny počet tokenů poloviční.



Obrázek 2 – Výstup nástroje Tokenizer po převodu českého textu na tokeny



Obrázek 3 – Výstup nástroje Tokenizer po převodu anglického textu na tokeny

1.4.2. GPT

GPT je anglická zkratka pro „Generative Pre-trained Transformer“, kterou jde do češtiny přeložit jako generativní předtrénovaný transformátor. Jedná se o nejznámější LLM, který představila nezisková organizace OpenAI v roce 2018. [8] Jednotlivá písmena ve zkratce GPT pak vysvětlují, jak tento LLM funguje. Písmeno G je zkráceno z „Generative“ a popisuje generování nového obsahu. Písmeno P zkráceno z „Pre-Trained“ vysvětluje, že model prošel procesem učení z velkého množství dat. „Pre“ také referuje na to, že je možné tyto data dále upravovat pro specifické účely. Písmeno T je zkráceno z „Transformer“, což je typ modelu umělé inteligence, kde je text převeden na číselné reprezentace v podobě tokenu, a poté je pomocí práce s vektory dále předpovídán další text.

Nejaktuálnější verze je GPT-4 vydaná v březnu 2023. Není přesně jasné na kolika parametrech byl GPT trénován, avšak jeho předchozí verze byla trénována na 175 miliardách parametrů. Je pravděpodobné, že u novější verze je toto číslo o dost vyšší. [11]

1.4.3. Gemini

V prosinci 2023 představil Google svůj LLM s názvem Gemini. Jedná se o nástupce LLM LaMDA a PaLM 2, kterým chce Google konkurovat OpenAI a jejich GPT. Google tvrdí, že v několika benchmark testech překonává ostatní LLM včetně GPT. Byly uvedeny 3 verze Gemini. První z nich se jmenuje „Nano“, která je určena na koncová zařízení uživatelů. Druhá z nich je „Pro“, která je dostupná skrze chatbota zdarma. Třetí verze s označením „Ultra“ je nejschopnější a měla by sloužit pro vysoce komplexní úkoly. [12]

1.4.4. Llama

Llama 2 je LLM od společnosti Meta, která spravuje největší sociální síť na světě: Facebook a Instagram. Představuje kolekci jazykových modelů, která má k dispozici od 7 miliard do 70 miliard parametrů. Llama 2 Chat modely jsou speciálně navrženy pro interaktivní dialog a dosahují vynikajících výsledků v porovnání s konkurenčními modely. Díky pečlivému ladění a implementaci bezpečnostních opatření nabízí uživatelům možnost využívat pokročilé jazykové modely s vysokou úrovní efektivity a bezpečnosti. Meta tento model poskytla jako open-source projekt. [13]

2. NÁSTROJE ZALOŽENÉ NA STROJOVÉM UČENÍ

Nejznámější část nástrojů založených na strojovém učení jsou pro veřejnost v dnešní době jednoznačně chatboti. Tato práce se však zajímá i o to, jak mohou být chatboti implementováni dále, například do vývojových prostředí.

2.1. Chatboti

Chatbot je software, který simuluje přirozený jazyk v konverzaci s uživatelem, se kterým lze komunikovat pomocí webového rozhraní či přes aplikaci v telefonu. Primárně se využívá textová forma komunikace, ovšem v dnešní době se také zdokonalují formy komunikace pomocí vzájemné verbální interakce.

2.1.1. Kontext

LLM mají dané omezení skrze jejich kontextové okno, což je pro ně krátkodobá paměť pro dané otázky a odpovědi. Kontext se udává v převážně v tisících tokenů, což odpovídá v češtině jednotkám normostran psaného textu. Kontext je jeden z hlavních parametrů, který říká, jak může být chatbot vhodný a kvalitní pro různé účely. [10]

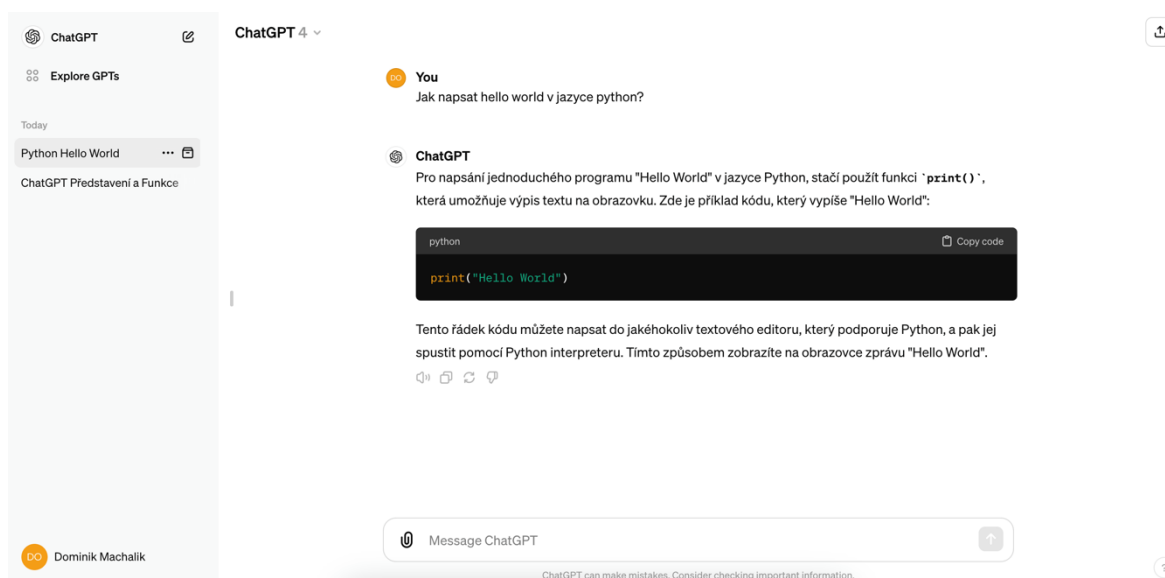
2.1.2. Prompt

Aby mohly chatboti, nebo obecně LLM fungovat, potřebují vždy ke svému dalšímu fungování dostat zadání. Toto zadání se u velkých chatbotů nazývá prompt. Chatboti pak vrací textový výstup, který může být obyčejným textem, ale také například kódem, tabulkou, zvukem či obrazem. K dosažení co nejlepšího výsledku je právě správné formulování promptu klíčovým krokem. Neexistuje oficiální manuál, jak by měl prompt pro chatbota vypadat. Je však doporučeno se držet několika jednoduchých kroků. Mezi tyto kroky patří určení role, jakou má chatbot zastávat, důkladný popis problému či požadavku s jeho kontextem a definování požadovaného výstupu, jenž lze doplnit i o případnou ukázkou. [10]

2.1.3. ChatGPT

Nejznámější chatbot současné doby je jednoznačně ChatGPT. Ten vydala firma OpenAI v listopadu 2022 a během dvou měsíců jej využívalo více jak 100 milionu lidí po celém světě. V té době se ihned stal nejrychleji rostoucí aplikací z hlediska uživatelů v historii. Až v posledním roce jej překonala aplikace Threads od Mety, která této mety dosáhla během necelého týdne. [14] ChatGPT lze po registraci používat bezplatně ve verzi GPT-3.5

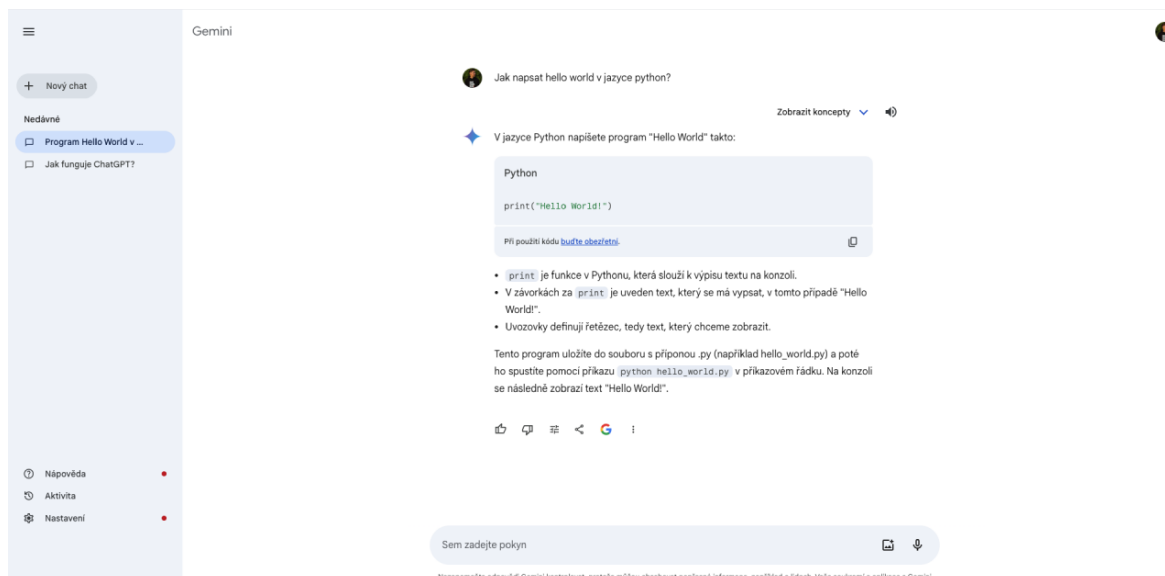
s neomezeným množstvím zpráv. Verze GPT-4 je v současné době zpoplatněna za 20 amerických dolarů bez DPH. Oproti nižší verzi může chatbot generovat obrázky pomocí modelu DALL-E, procházet Internet a spouštět kód v jazyce Python. Ovšem v tomto případě je omezeno používání na 40 zpráv každé 3 hodiny. Nadále je však dostupné přepnutí na neomezenou verzi GPT-3.5. Úvodní stránka ihned nabízí náměty ke konverzaci. Na levé straně se nachází panel, kde uživatel může překlíkávat mezi historií chatů. Chatbot v jednom chatovém okně nezná obsah a kontext v jiném.



Obrázek 4 – Rozhraní webové aplikace ChatGPT

2.1.4. Gemini

Gemini je chatbot od společnosti Alphabet, jenž vlastní Google. Byl zpřístupněn v únoru 2024 jako nástupce předchozího konverzačního systému se jménem Bard. Samotný Bard běžel od února 2023. Jedná se o odpověď Googlu na ChatGPT, který postupně do svého vyhledávače Bing začal nasazovat konkurenční Microsoft. Funguje na velmi podobném principu jako ChatGPT, kdy je nasazen jako webová aplikace obsahující chatový vstup a odpovídá jako v chatovací aplikaci. Také nabízí již předpřipravené náměty ke konverzaci a v levé části obrazovky má historii předešlých chatů. [15]



Obrázek 5 – Rozhraní webové aplikace Google Gemini

2.1.5. Claudie

Claudie představila společnost Anthropic v březnu 2023 jako asistenta nové generace. Jedná se o LLM přístupný přes stejnojmenné chatovací rozhraní, nebo vývojářům přes API. Je vhodná k sumarizaci, vyhledávání informací v textu, psaní textu, dotazům či programování. Na začátku byly k dispozici dvě verze: „Standard“, která nabízela nejvyšší výkon, a „Instant“, která byla cenově dostupnější a rychlejší. [16] Je schopná přepisovat a analyzovat obrázky, ručně psané poznámky či fotografie. Umí také generovat webové stránky, převádět obrázky do strukturovaných JSON dat a nebo debugging. V neposlední řadě je schopna také překladu či procvičování gramatiky. [17]

2.1.6. Uizard

Uizard je webová aplikace, která je určena pro designery, vývojáře, manažery i startupy. Slouží k navrhování uživatelských rozhraní, vytváření prototypů a k celkovému usnadnění a urychlení návrhu UX a UI dané aplikace. Rychlost je v tomto případě zařízena díky umělé inteligenci, která dokáže samotné výstupy generovat. Tvůrci dokonce tvrdí, že není potřeba mít žádné zkušenosti s navrhováním designu. Skrze aplikaci lze vytvářet také jednotlivé komponenty, které lze opakovaně využívat. Aplikace obsahuje funkci Screenshot Scanner, která umí převádět snímky obrazovky nebo drátěné modely na samotný návrh aplikace společně s funkčním grafickým prototypem. Dále také obsahuje funkci Autodesigner, která navrhne celé uživatelské rozhraní pouze na základě popisu od uživatele. Nástroj je pro hobby nadšence a studenty zdarma, kteří mají ve Free verzi možnost vytvářet až 10 šablon ve 2

projektech. Neomezené šablony a projekty lze využívat ve verzi Pro, která stojí pro jednotlivce 12 amerických dolarů na měsíc. [24] [25]

2.1.7. GitHub Copilot Workspace

GitHub Copilot Workspace je nová služba pro programátory, která dokáže pracovat s kódem pomocí promptů v přirozeném jazyce. Vyšla na konci dubna 2024. Tato služba navazuje na nástroje GitHub Copilot a GitHub Copilot Chat, které již v minulosti dokázaly zvýšit produktivitu vývojářů. Umožňuje vývojářům plánovat, testovat i spouštět kód pouze na základě promptů, čímž dokáže celý vývoj urychlit. Je také schopen opravovat chyby v kódu buď přímo procházením kódu, nebo dle nahlášených bug reportů přes službu GitHub. Umí analyzovat, kontrolovat a vylepšovat také pull requesty, implementovat nové funkcionality a vytvářet a nastavovat celý projekt. Veškeré informace jsou získávány na základě integrace s GitHubem, tudíž je možné projekt vyvíjet jednoduše i skupinově. V neposlední byl také kladen důraz na možnost provádět tyto úkoly jednoduše i z mobilního telefonu. [26]

2.2. Porovnání chatbotů

Chatboty lze mezi sebou i porovnávat. Existuje organizace Large Model Systems, která provozuje projekt Chatbot Arena. V tom mezi sebou zápasí chatboti v různých kategoriích na základě crowdsourcingu. Výsledky jsou poté založeny na základě Elo žebříčku. Jednou z kategorií je i programování, kde se v nejlepší desítce drží hned 4 různé podverze GPT-4.

Rank* (UB)	Delta	Model	Arena Elo	95% CI	Votes	Organization	License	Knowledge Cutoff
1	0	GPT-4-Turbo-2024-04-09	1269	+7/-7	8300	OpenAI	Proprietary	2023/12
1 ↑	1	GPT-4-1106-preview	1260	+7/-6	13031	OpenAI	Proprietary	2023/4
2	0	Claude 3 Opus	1255	+6/-5	18919	Anthropic	Proprietary	2023/8
2 ↑	1	GPT-4-0125-preview	1248	+7/-7	12478	OpenAI	Proprietary	2023/12
4 ↓	-2	Gemini 1.5 Pro API-0409-Preview	1235	+8/-8	11769	Google	Proprietary	2023/11
6 ↑	2	Claude 3 Sonnet	1218	+7/-6	15928	Anthropic	Proprietary	2023/8
6	0	Llama-3-70b-Instruct	1208	+7/-7	14281	Meta	Llama 3 Community	2023/12
6 ↑	1	Reka-Core-20240501	1205	+10/-13	3717	Reka AI	Proprietary	Unknown
7 ↑	3	GPT-4-0314	1198	+8/-8	8640	OpenAI	Proprietary	2021/9
7 ↑	3	Qwen-Max-0428	1195	+13/-14	1940	Alibaba	Proprietary	Unknown

Tabulka 1 – Žebříček chatbotů v programování [27]

2.3. Implementace nástrojů a technologií strojového učení do vývojových prostředí

Tato podkapitola se bude věnovat implementaci některých nástrojů a technologií strojového učení do přímo do daných vývojových prostředí.

2.3.1. GitHub Copilot

GitHub Copilot je doplněk pro vývojová prostředí, který pomáhá z kontextu kódu s tvořením kódu, řešením problémů, psaním testů a implementováním API bez hledání na Internetu. Byl představen v červnu 2021. [18] Je možno jej použít ve vývojových prostředích Visual Studio Code, Visual Studio, Neovim a ve vývojových prostředích od firmy JetBrains. Byl představen hlavně pro jazyky JavaScript, TypeScript, Ruby a Go. Doplněk je zpoplatněn a dostupný ve třech různých verzích. Verze Individual pro samostatné vývojáře, freelancery je stojí měsíčně 10 amerických dolarů nebo je také dostupná za 100 dolarů ročně. Verze Business pro menší organizace stojí 19 dolarů měsíčně a verze pro velké firmy je dostupná otevřenější verze pro personalizované úpravy za 39 dolarů na měsíc. Pro učitele či studenty je dostupná verze Individual bezplatně po prokázání studia či zaměstnání ve vzdělávací instituci. [19]

2.3.1.1. GitHub Copilot Chat

GitHub Copilot Chat je chatovací rozhraní, které umožňuje uživateli klást otázky a získávat na ně odpovědi týkající se programování přímo přes github.com a podporovaná vývojová prostředí Visual Studio Code, Visual Studio a JetBrains IDEs. Umožňuje získávat informace bez potřeby procházet dokumentaci či procházet fóra na Internetu. Využívá zpracování přirozeného jazyka a strojového učení k analýze dotazu a generování odpovědi. Odpovědi se v tomto případě rozumí generování či vysvětlení kódu. Obsahuje však filtry na detekci nežádoucího obsahu. Funguje v češtině i v jakémkoliv dalším jazyce, ovšem primární podporovaný jazyk je angličtina. [20]

2.3.1.2. GitHub Copilot – Code completion

GitHub Copilot – Code completion je doplněk přímo do vývojových prostředí, který přímo pomáhá s psaním kódu. Jedná se o asistenta, který pomáhá psát kód rychleji a s menším úsilím. Nabízí přímo během psaní návrhy v kódu. Jedná se například o dokončení řádku, generování celého bloku kódu. Též dokáže generovat kód jen na základě napsání komentáře

a Copilot pod komentář již navrhne možnost. Lze přijímat celý návrh, jeho část a nebo také jej celý odmítnout a pokračovat ve psaní. [21]

2.3.2. Amazon Q Developer

Amazon Q Developer, též dříve známý jako Amazon CodeWhisperer je nástroj navržen pro podporu vývojářů při psaní kódu ve vývojovém prostředí nebo příkazové řádce. Poskytuje návrhy pro úpravu kódu od jednořádkových změn až po celé funkce. Umí také skenovat bezpečnost aplikace či sledovat open-source licence. Také obsahuje interaktivního asistenta na základě umělé inteligence, který tyto změny dokáže sám provádět. Také je možné s tímto asistentem vést dialog o dané aplikaci. [28]

2.3.3. JetBrains AI

JetBrains AI je doplněk do vývojových prostředí od společnosti JetBrains, mezi které patří například IntelliJ Idea, PyCharm, PhpStorm, ReSharper nebo Fleet. Tento doplněk nabízí chatování přímo ve vývojovém prostředí o samotném kódu v libovolném programovacím jazyce. Pomáhá se psaním kódu skrze našeptávání podobně jako GitHub Copilot, umí pomoci s vhodným pojmenováváním proměnných či funkcí. Nabízí i navrhování popisu změn při příspěvcích do verzovacího systému, jako je například Git. Dokáže také přímo rozpoznat chyby při kompilování či interpretování a dle nich navrhnout jejich samotné opravení. Rozšíření je zpoplatněno na 314 Kč měsíčně nebo 3416 Kč ročně. [22]

3. PŘÍNOSY A RIZIKA SPOJENÁ S VYUŽITÍM STROJOVÉHO UČENÍ VE VÝVOJI SOFTWARE

Jako každá technologie i strojové učení má své přínosy a rizika, která přináší do odvětví IT. Díky strojovému učení dokážou lidé rychleji a efektivněji řešit některé problémy, které se mohou vyskytnout. Ovšem s využitím této technologie přicházejí i rizika. Mezi ty lze zařadit například odesílání informací či dat jiným společnostem a získávání špatných odpovědí. Tyto rizika se objeví hlavně při nesprávném využití těchto nástrojů. V této kapitole představím některé parametry, kterých si budu v praktické části později všímat.

3.1. Spokojenost vývojáře

Výzkum od společnosti GitHub tvrdí, že vývojáři používající GitHub Copilot uvedli, že se cítí až o 75% více naplnění svou prací, cítí se méně frustrovaní a mohou se tak soustředit na svou práci. Jedná se o jednu ze základních metrik, která zvyšuje efektivitu samotných vývojářů. Téměř stejné množství vývojářů také uvedlo, že jim tento nástroj pomáhá zůstat ve flow a zachovat úsilí u opakujících se úkolů. [29]

3.2. Rychlost tvorby aplikace

Ve stejném výzkumu se zjistilo, že vývojáři dokázali splnit zadaný úkol o 55 % rychleji, než ti, kteří jej nevyužijí. V této části výzkumu bylo najato 95 profesionálních vývojářů, ti poté byli rozděleni náhodně do dvou skupin a dostali za úkol naprogramovat HTTP server v programovacím jazyce JavaScript. Skupině bez Copilota trval úkol v průměru 2 hodiny a 41 minut, skupině s Copilotem to v průměru trvalo o 90 minut méně – 1 hodinu a 11 minut. Také míra dokončení úkolu byla vyšší o 8 %. [29]

3.3. Hledání informací

Vývojáři také některé své znalosti musí dohledávat na Internetu a tím mohou zvýšit celkovou časovou náročnost dokončení různých úkolů. Prohledávání dokumentací či fór, jako je například StackOverflow rozhodně patří do tohoto problému a s tímto problémem dokáže právě Copilot pomoci. Až 77 % programátorů uvedlo, že díky němu nemuselo na Internetu dohledávat tolik informací. [29]

3.4. Kvalita generovaných odpovědí

Bylo zjištěno, že samotné ChatGPT dokázalo během pár měsíců snižovat kvalitu svých odpovědí. V období od března 2023 do června 2023 se zjistilo, že novější verze ChatGPT si s různými druhy problémů si ChatGPT poradilo daleko hůře než ta starší. Je proto důležité, aby jednotlivci či společnosti zkoumali ve svých konkrétních případech, jestli jim opravdu nástroje strojového učení pomáhají, a to hlavně v dlouhodobém měřítku. V posledních měsících se například matematické schopnosti opět zlepšují, obzvláště díky možnosti spouštět kód v jazyce Python, který dokáže vypočítat jednoduché i složitější matematické operace a nemusí tak předpovídat své odpovědi. Také napojení na Internet pomáhá zlepšit kvalitu odpovědí uživateli. [30] [31]

3.5. Halucinace

LLM je učen na obrovském množství dat, které by mu měli přinést co nejvíce informací pro řešení problémů, ovšem není možné naučit model naprosto vše. Jelikož modely předpovídají další tokeny při generování odpovědi, tak se může stát, že tvrdí znalost daného problému i když je jeho odpověď nesprávná. Cílem je vygenerovat přirozeně znějící text nebo přirozeně vypadající odpověď, jenž se však nemusí zakládat na pravdě. Takovému chování se u LLM říká halucinace. [32]

3.6. Ztráta kontextu

Již bylo zmíněno, že dané LLM mají dané kontextové okno, ve kterém si pamatují vstupy uživatele a své odpovědi. Pokud by konverzace o aplikaci, například přes ChatGPT pokračovala příliš dlouho, je možné, že časem bude ztracen kontext celé aplikace. V ten moment může poté chatbot začít pracovat s částmi aplikace, které nemusí ani existovat. [32]

3.7. Zpřístupnění dat jiným společnostem

Riziko využití nástrojů strojového učení spočívá také v tom, že LLM jsou často trénovány na datech, které poskytnou uživatelé skrze samotného chatbota. Společnost OpenAI tvrdí, že ChatGPT a další jejich služby jsou učeny skrze 3 hlavní pilíře. Jedná se o data veřejně dostupná na Internetu, data poskytnutá jinými společnostmi na základě licencí a data poskytnutá jejich uživateli či lidskými trenéry modelu. Podstatný je v tomto případě hlavně třetí pilíř. Ten říká, že poskytnutá data uživatelů v konverzacích jsou dále zpracovávána. Proto může být rizikové do chatbota vkládat hesla, API klíče či díla pod dohodami o mlčenlivosti,

které mnoho zaměstnanců v různých společnostech podepisuje. Tímto zpřístupněním citlivých informací vzniká riziko, že chatbot může tyto informace někdy v budoucnosti využít při generování odpovědí u dalších uživatelů. Možným řešením tohoto problému je zakoupení plánu ChatGPT Team nebo ChatGPT Enterprise, u kterých OpenAI zaručuje, že na datech uživatelů s tímto předplatným své modely netrénuje. [33]

3.8. Nahrazení člověka v pracovním procesu

Nabízí se i otázka, jestli dokáže v budoucnu nějaký nástroj strojového učení nahradit programátory, testery, grafiky, IT architektky, datové analytiky a další profese z odvětví IT. V současné době nejsou dostatečně kvalitní nástroje, které by dokázaly takové profese plně nahradit. Lidé v IT často řeší komplexnější problémy, které na sebe úzce navazují. Tyto problémy by pak bylo nutné řešit velmi kvalitním promptem pro daného chatbota. Právě kvalitní prompt dokáže napsat hlavně osoba, jenž se v dané problematice vyzná. U laika by taková práce byla přinejmenším pomalá a velmi pravděpodobně i nekvalitní. Proto budou nejspíš chatboti sloužit jako nástroje, které práci nebudou brát, ale budou měnit její kvalitu a rychlost. [34]

II. PRAKTICKÁ ČÁST

4. PŘÍPADOVÁ STUDIE S VYUŽITÍM STROJOVÉHO UČENÍ

V praktické části mé práce se pokusím představit některé z nástrojů popsanych v části teoretické. Grafické návrhy budu generovat pomocí aplikace Uizard. Co se týče návrhu aplikace, programování a testování, půjde zejména o nástroje GitHub Copilot a ChatGPT.

Pokusím se kombinovat jejich schopnosti a střídavě je využívat na různé části aplikace. Pokud to bude možné, v některých případech se pokusím využít více nástrojů současně. V tom případě bych vybral nakonec tu odpověď, která mi přijde subjektivně vhodnější.

Nakonec všechny výstupy kriticky zhodnotím a případně je dle potřeby upravím.

4.1. Podrobnosti o používaných nástrojích a technologiích

Pro tvorbu projektu jsem využil jsem využil několik technologií. Vybral jsem si ty nejvíce používané, jelikož hlavně na ně bude mít vývoj v odvětví strojového učení vliv.

4.1.1. ChatGPT

ChatGPT jsem již představil v teoretické části práce. Zde pouze zmíním, že se budu držet doporučených rad komunitou, které se týkají správného sestavení promptu. Budu používat aplikaci ve webovém prohlížeči, jenž využívá model ChatGPT 4 s podverzí „Feb 13, 2024“. Jelikož využívám nejnovější verzi ChatGPT, bylo nutné si předplatit plán „Plus“.

4.1.2. GitHub Copilot

GitHub Copilota budu používat skrze rozšíření do vývojového prostředí Visual Studio Code. Aktuální verze je „1.184.0.“ K rozšíření GitHub Copilot jsem se dostal díky GitHub Student Developer Pack, který je poskytován téměř celosvětově zdarma studentům po dobu jejich studia a také i učitelům po dobu jejich pracovního poměru u vzdělávací instituce.

4.1.3. Uizard

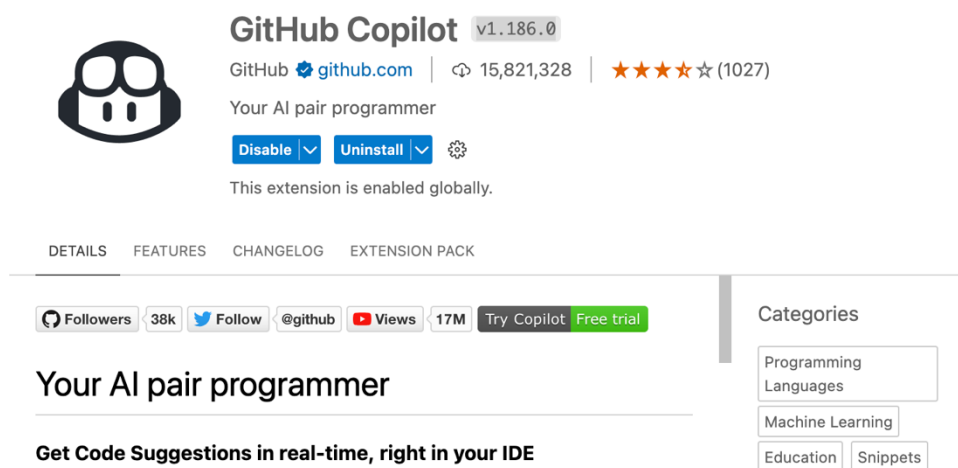
Aplikaci Uizard budu využívat skrze verzi 1.5 nástroje Autodesigner.

4.2. Příprava prostředí

Příprava na projekt není nikterak náročná. Nejprve je potřeba mít nainstalovaný webový prohlížeč, nejlépe s některou z novějších verzí renderovacího jádra prohlížeče. V mém případě budu využívat prohlížeč Brave ve verzi 1.65.123, který využívá jádro Chromium ve verzi 124.0.6367.91.

Dále je potřeba nainstalovat vhodné vývojové prostředí. Jelikož budu pracovat s jazyky JavaScript a HTML, tak jsem si vybral Visual Studio Code. Jeho aktuální verze je 1.88.1.

Přes menu ve vývojovém prostředí Visual Studio Code se dostanu do záložky Extensions, kde vyhledám doplněk GitHub Copilot. Ten budu využívat ve verzi v1.186.0.



Obrázek 6 – Stránka doplňku GitHub Copilot pro IDE VSCode

4.3. Případová studie - Todo List

Jako ukázkový projekt jsem si vybral vytvoření jednoduché webové aplikace pro správu úkolů. Tento projekt jsem si zvolil, jelikož se jedná o oblíbenou volbu mnoha začínajících programátorů, kteří si osvojují základy programování webových aplikací. Aplikace bude vytvářena v jazyce JavaScript, k otestování pak použiji testovací framework Jest.

4.3.1. Návrh funkcionalit aplikace

Základním bodem aplikace je její samotný návrh. V tom budu specifikovat veškeré požadavky, které bude aplikace splňovat. Již tento návrh zadám do ChatGPT, který se pokusí funkcionality sestavit. Ze začátku je třeba definovat roli, kterou by měl zastávat, popsat požadovaný problém a jeho kontext, a nakonec definovat požadovaný vstup. V tomto případě budu chtít, aby odpověď byla generována jako odpověď programátora, který dostal za úkol udělat todo list.

4.3.1.1. Sestavení promptu pro vygenerování funkcionalit

V tomto případě bude ještě požadovaným výstupem textový popis aplikace. Prompt jsem nakonec sestavil následovně:

„Chovej se jako zkušený programátor, který dostal za úkol naprogramovat webovou aplikaci todo listu v jazyce JavaScript. Vypiš mi v bodech funkcionality, které by měla taková aplikace zvládnout.“

4.3.1.2. Vygenerovaná odpověď dle promptu pro funkcionality

Odpověď ChatGPT ze zadaného promptu byla následovná:

Aplikace typu todo list (úkolník) je skvělým projektem pro procvičení dovedností v JavaScriptu. Tady je seznam funkcionalit, které by měla kvalitní webová aplikace todo listu obsahovat:

1. **Přidávání úkolů** - Umožnit uživatelům zadávat nové úkoly. Každý úkol by měl mít alespoň základní popis.
2. **Odstraňování úkolů** - Umožnit uživatelům odstranit úkoly, které již byly dokončeny nebo jsou již zastaralé.
3. **Editace úkolů** - Umožnit úpravu již existujících úkolů. Toto by mohlo zahrnovat změnu textu, priority nebo jiných atributů úkolu.
4. **Zaškrtnutí dokončených úkolů** - Umožnit uživatelům označit úkol jako dokončený. Dokončené úkoly by mohly být vizuálně odlišeny (například přeškrtnutím).
5. **Filtrování úkolů** - Nabídnout filtry pro zobrazení všech úkolů, pouze aktivních úkolů nebo dokončených úkolů.
6. **Ukládání stavu** - Ukládat seznam úkolů do lokálního úložiště prohlížeče (localStorage), aby uživatelé mohli opustit stránku a vrátit se k ní bez ztráty dat.
7. **Prioritizace úkolů** - Umožnit uživatelům nastavit prioritu úkolům, aby mohli lépe řídit, na čem chtějí pracovat jako první.
8. **Datové značky** - Možnost přidávat datumy k úkolům, včetně deadline nebo data vytvoření.
9. **Notifikace** - Volitelně zahrnout upozornění nebo připomínky pro blížící se deadliny úkolů.
10. **Responsivní design** - Aplikace by měla být použitelná na různých zařízeních, od stolních počítačů až po mobilní telefony.
11. **Export a import úkolů** - Umožnit uživatelům exportovat jejich seznam úkolů do souboru a následně jej importovat zpět do aplikace.
12. **Mnohojazyčná podpora** - Možnost přepínat mezi různými jazyky, pokud je aplikace určena pro mezinárodní publikum.
13. **Téma a personalizace** - Nabídnout možnost změny vzhledu aplikace prostřednictvím různých barevných schémát nebo témat.

Tyto funkcionality tvoří základ robustní a uživatelsky přívětivé webové aplikace todo listu v JavaScriptu.

4.3.1.3. Zhodnocení vygenerovaného výsledku

Výsledek promptu nakonec obsahuje všechny podstatné body takové aplikace. Všechny body, které jsem měl původně v plánu, se ve výstupu nacházely. K nim dokázal navíc vymyslet i další body, které by mě třeba na začátku projektu nenapadly a spíše bych k nim

došel až během samotného programování. Mezi to patří třeba funkce exportu seznamu úkolů do souboru, mnohojazyčná podpora anebo personalizace aplikace.

4.3.2. Návrh uživatelského rozhraní

Dalším krokem byl samotný návrh uživatelského rozhraní. K tomu jsem využil aplikaci Uizard a jeho funkcionalitu Autodesigner.

4.3.2.1. Sestavení promptu pro vygenerování uživatelského rozhraní

V tomto kroku bylo potřeba popsat celý projekt v angličtině ve 300 znacích. Proto jsem zvolil následující prompt:

A todo list web application - users to manage their tasks. Users can add new tasks, each with a description. He can remove no longer needed tasks or edit details of existing tasks. A key feature is the ability to mark tasks as completed, which changes their appearance. Also tasks can be filtered.

Následně bylo potřeba zvolit klíčová slova vztahující se k samotnému projektu, k popisu designu nebo obojího. Zde jsem zvolil slova: todo, list, tasks, minimalistic, modern. V poslední řadě jsem zaškrtnl další fráze, které se ke generaci projektu vztahují. Zde se jednalo o slova Light a Modern.

Use simple text to generate your project using Uizard's AI-powered Autodesigner

Which device are you designing for?

☐ Mobile ☐ Tablet ☒ Desktop

Describe your project in plain English [Try example](#)

A todo list web application - users to manage their tasks. Users can add new tasks, each with a description. He can remove no longer needed tasks or edit details of existing tasks. A key feature is the ability to mark tasks as completed, which changes their appearance. Also tasks can be filtered.

297/300

Describe a design style, pick keywords, or both

todo, list, tasks, minimalistic, modern

39/150

Light Dark Modern Artsy Techy Young

Corporate Formal Elegant Hand-drawn

Generate my project Beta

Obrázek 7 – Promptové okno aplikace Uizard se sepsaným promptem

4.3.2.2. *Vygenerovaný výstup dle promptu pro uživatelské rozhraní*

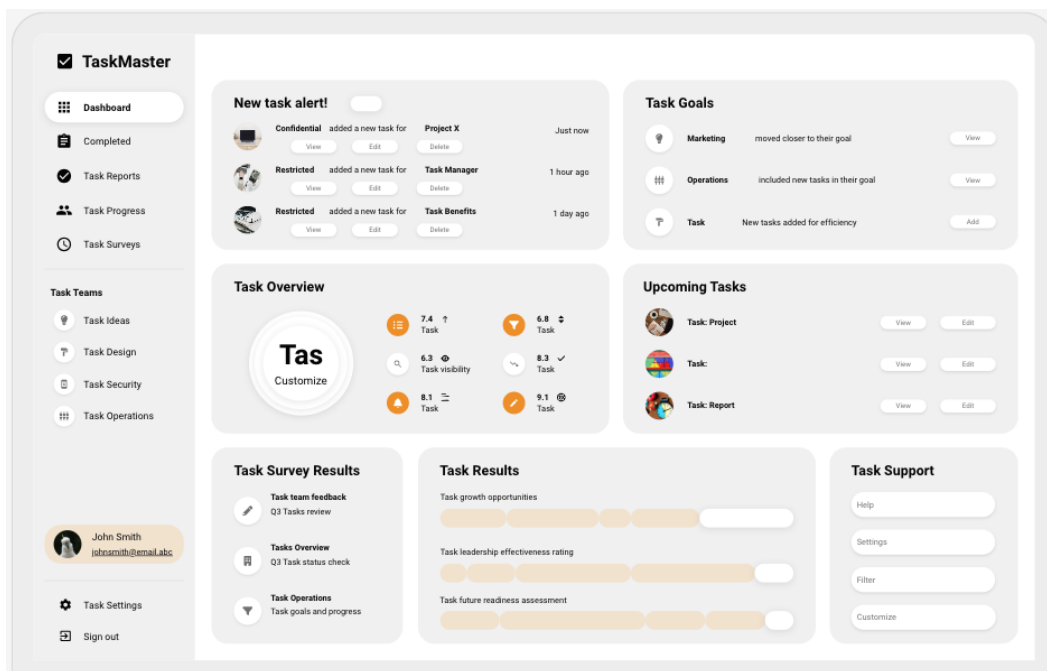
Aplikace se po zadání promptu přepnula na loading screen a poté bylo třeba už jenom čekat. Celý proces generování zabral 3 minuty a po jeho dokončení mám k dispozici 5 návrhů s různými částmi aplikace.

Výsledek obsahuje návrh domovské stránky s přehledem různých částí aplikace, stránku se samotnými úkoly, stránku pro vytvoření úkolu, stránku s notifikacemi a také stránku s nastavením profilu. Všechny stránky jsou propojeny v preview funkci a v novém okně lze spustit simulaci aplikace, kde lze mezi jednotlivými stránkami překlíkávat.

4.3.2.3. *Zhodnocení vygenerovaného uživatelského rozhraní*

Vygenerovaný návrh na první pohled vypadá dobře. Obsahuje některé podstatné části, které bych od aplikace očekával. Mezi ty patří například logo v levém horním rohu, menu v levé části stránky nebo domovská stránka. Ovšem po důkladnějším průzkumu nacházím docela nesmyslné části návrhu.

Z hlediska UI jsem si ihned všiml, že Uizard využívá ve výstupu ikony Material designu od Googlu a také známý font Roboto. Prvky jsou navrženy většinou pěkně, moderně a jednoduše. Dokonce je v návrhu i potencionální název a logo aplikace v levém horním rohu. U některých tlačítek a vstupů pro text mi však vadí nedostatečný padding, jako například tlačítka menu v pravém dolním rohu na obrázku 8.



Obrázek 8 – Návrh uživatelského rozhraní aplikací Uizard

Větší problém je UX vygenerovaného řešení. Po důkladném prozkoumání návrhu jde vidět, že vygenerované texty, tlačítka a hlavně umístění elementů často nedává smysl. Opět to jde vidět na obrázku 8, kde u menu „Task Support“ jsou tlačítka „Filter“ nebo „Customize“, které v tomto kontextu nedávají smysl. Také u komponenty „Task Results“ nedávají smysl progress bary a ty jsou špatně navrženy i z hlediska UI. Stránka je pro uživatele dle mého názoru neintuitivní a spousta věcí je navržena špatně.

Proto jsem se rozhodl, že z tohoto výstupu využiji vzhled některých elementů, font a ikony. Co se týče konečného rozmístění prvků a práci na intuitivním UX, tak tento krok budu řešit pouze na základě svých zkušeností bez využití nástrojů strojového učení.

4.3.3. Vývoj aplikace – funkcionality

Po návrhu funkcionalit aplikace a uživatelského rozhraní je nyní možné začít samotnou tvorbou kódu. Jelikož se bude jednat webovou aplikaci, rozhodl jsem se pro využití programovacího jazyku JavaScript.

4.3.3.1. Sestavení promptu pro vygenerování kódu aplikace

Jako požadovaný výstup budu chtít kód v jazyce JavaScript. Dle toho budu sepisovat prompt. Také se zeptám na to, jak by bylo vhodné kód dělit na různé soubory a složky a jak jej dát dohromady tak, aby z něj vznikla funkční aplikace. Do ChatGPT jsem tedy zadal prompt:

Chovej se jako programátor zkušený v jazyce JavaScript. Sestroj mi aplikaci pro správu úkolů (todo list), která bude mít následující funkcionality:

- *Lze přidávat úkoly - každý úkol má název, popis a datum vytvoření*
- *Nepovinně má každý úkol také prioritu a deadline*
- *Nepovinně půjde úkolům dávat štítky. Štítky bude možno jednoduše vytvořit, nebo vybrat z těch, které byly již použity*
- *Úkoly půjde zaškrtnout a tím je označit jako hotové*
- *Úkoly půjde upravovat a mazat*
- *Úkoly budou uloženy v localStorage daného prohlížeče, aby zde zůstaly i po jeho zavření*
- *Úkoly půjde řadit dle priority, dle počtu dnů do deadline a také dle toho, kolik dní je již úkol vytvořen*
- *Také je bude možno filtrovat dle štítků, priority, dnů do deadline a dnů od vytvoření. Pokud jsou filtry zapnuty, bude to jasně vidět a bude možné jednoduše filtr odstranit*

Nejprve mi napiš, jak mám postupovat a ke kroku mi případně vygeneruj funkční, testovatelný a bezpečný kód v jazyce JavaScript.

4.3.3.2. Vygenerovaný výstup dle promptu pro tvorbu aplikace

Jako odpověď jsem dostal prompt rozdělený na odpovědi. ChatGPT vždy vypsalou některou z částí odpovědi a poté ke konci se vždy aplikace zeptala, zda může pokračovat. Jakmile jsem napsal, že ano, tak vypisování odpovědi pokračovalo dále a navazovalo na předchozí odpověď.

Nejprve jsem dostal vygenerovanou HTML strukturu aplikace.

```
<!DOCTYPE html>
<html lang="cs">
<head>
  <meta charset="UTF-8">
  <title>Todo List App</title>
</head>
<body>
  <h1>Todo List App</h1>
  <form id="new-task-form">
    <input type="text" id="task-title" placeholder="Název úkolu"
required>
    <textarea id="task-description" placeholder="Popis úkolu"></textarea>
    <input type="date" id="task-deadline">
    <input type="number" id="task-priority" placeholder="Priorita (1-5)">
    <button type="submit">Přidat úkol</button>
  </form>

  <div id="tasks-container"></div>

  <script src="app.js"></script>
</body>
</html>
```

Vygenerovaný kód 1 – HTML struktura aplikace

Následně už začalo generování samotného kódu v JavaScriptu. Nejprve bylo potřeba vytvořit soubor *app.js*, který bude obsahovat logiku aplikace. Vytvoření souboru mi ChatGPT připomnělo také. První částí JavaScriptového kódu byl eventListener pro přijetí vstupů formuláře a vytvoření úkolu. V této funkci ChatGPT definovalo samotný úkol jako objekt.

```
document.getElementById('new-task-form').addEventListener('submit', function(event)
{
    event.preventDefault();
    const title = document.getElementById('task-title').value;
    const description = document.getElementById('task-description').value;
    const deadline = document.getElementById('task-deadline').value;
    const priority = document.getElementById('task-priority').value;

    const task = {
        id: Date.now(), // jedinečné ID úkolu
        title,
        description,
        created: new Date().toISOString(),
        deadline,
        priority,
        completed: false
    };

    saveTask(task);
    displayTask(task);
    this.reset(); // reset formuláře
});
```

Vygenerovaný kód 2 – Vytvoření nového úkolu

Ten poté uložilo pomocí funkce *saveTask* zobrazil pomocí funkce *displayTask*. Tyto funkce byly definovány až pod `addEventListener`. Správně byla také definována část pro ukládání úkolu do `localStorage`.

```
function saveTask(task) {
    const tasks = JSON.parse(localStorage.getItem('tasks')) || [];
    tasks.push(task);
    localStorage.setItem('tasks', JSON.stringify(tasks));
}

function displayTask(task) {
    const container = document.getElementById('tasks-container');
    const element = document.createElement('div');
    element.innerHTML = `
        <div>
            <input type="checkbox" ${task.completed ? 'checked' : ''} onchange="toggleTaskCompletion(${task.id})">
            <strong>${task.title}</strong> - ${task.description} (Deadline: ${task.deadline || 'N/A'}, Priority: ${task.priority || 'N/A'})
            <button onclick="deleteTask(${task.id})">Smazat</button>
        </div>
    `;
    container.appendChild(element);
}
```

Vygenerovaný kód 3 – Uložení a zobrazení úkolu

Na konci své odpovědi ještě navrhl `addEventListener` pro výpis úkolů.

```
document.addEventListener('DOMContentLoaded', function() {  
    const tasks = JSON.parse(localStorage.getItem('tasks')) || [];  
    tasks.forEach(displayTask);  
});
```

Vygenerovaný kód 4 – Výpis úkolů

Před ukončením odpovědi navrhl další kroky, které bude ještě potřeba splnit. Jednalo se o označení úkolu jako dokončený, o mazání úkolů, šítky a řazení. Zde odpověď skončila a poté jsem byl požádán o definování dalšího postupu.

Do chatovacího okna jsem pouze vepsal: „Pokračuj s dvěma funkcemi a pak budeme pokračovat dále.“ ChatGPT pokračovalo s generováním aplikace. Tentokrát se jednalo o funkcionality pro označení úkolů jako dokončeného.

```
function toggleTaskCompletion(taskId) {  
    const tasks = JSON.parse(localStorage.getItem('tasks'));  
    const task = tasks.find(t => t.id === taskId);  
    task.completed = !task.completed; // přepne stav hotovosti  
    localStorage.setItem('tasks', JSON.stringify(tasks)); // uloží aktualizované  
    úkoly  
    document.getElementById(`task-  
${taskId}`).querySelector('input[type="checkbox"]').checked = task.completed;  
}  
  
function displayTask(task) {  
    const container = document.getElementById('tasks-container');  
    const element = document.createElement('div');  
    element.setAttribute('id', `task-${task.id}`);  
    element.innerHTML = `  
        <div>  
            <input type="checkbox" ${task.completed ? 'checked' : ''}  
onchange="toggleTaskCompletion(${task.id})">  
            <strong>${task.title}</strong> - ${task.description} (Deadline:  
${task.deadline || 'N/A'}, Priority: ${task.priority || 'N/A'})  
            <button onclick="deleteTask(${task.id})">Smazat</button>  
        </div>  
    `;  
    container.appendChild(element);  
}
```

Vygenerovaný kód 5 – Dokončení úkolu

Zde přibyla jedna nová funkce sloužící k přepínání dokončení úkolu. Opět se validně stav úkolu také upravuje v localStorage prohlížeče.

Po implementaci tohoto úkolu bylo potřeba ještě upravit kód pro výpis, jelikož je nutné zobrazit checkbox úkolu. Po dokončení přepsání výpisu bylo potřeba přidat odstraňování úkolů.

```
function deleteTask(taskId) {  
  let tasks = JSON.parse(localStorage.getItem('tasks'));  
  tasks = tasks.filter(t => t.id !== taskId);  
  localStorage.setItem('tasks', JSON.stringify(tasks));  
  document.getElementById(`task-${taskId}`).remove();  
}
```

Vygenerovaný kód 6 – Odstraňování úkolů

Tohle byl poslední kód ve druhé odpovědi. Poté opět ChatGPT navrhlo pokračovat s další funkcionalitou. Tento krok jsem mu potvrdil.

Ve své třetí odpovědi ChatGPT vrátilo přidávání štítků k úkolům. Nejprve bylo potřeba do HTML formuláře přidat vstup pro štítky.

```
<input type="text" id="task-tags" placeholder="Štítky (oddělené čárkou)">
```

Vygenerovaný kód 7 – Vstup pro přidání štítků

Poté bylo potřeba upravit také funkci pro přidávání štítků, která nyní bere data ze vstupu pro tagy a ukládá je do objektu úkolu. Také bylo potřeba provést úpravu výpisu, kde se nyní mohou štítky vypisovat.

```
const tags = document  
  .getElementById('task-tags')  
  .value.split(',')  
  .map(tag => tag.trim())  
  .filter(tag => tag);
```

Vygenerovaný kód 8 – Zpracování štítků

```
const tagsHtml = task.tags.map(tag => `<span class="tag">${tag}</span>`).join(' ');
// A poté v element.innerHTML:
<div>Tags: ${tagsHtml}</div>
```

Vygenerovaný kód 9 – Výpis štítků

Nyní zbývá vyhledávání na základě zadaných štítků. K tomu bylo opět potřeba přidat input a tlačítko do HTML kódu.

```
<input type="text"
      id="filter-tags"
      placeholder="Filtrovat podle štítků (oddělené čárkou)">
<button onclick="filterTasksByTags()">Filtrovat</button>
```

Vygenerovaný kód 10 – Vstup pro přidání štítku s tlačítkem

K uvedeným elementům pak ještě bylo potřeba přidat funkci pro filtrování.

```
function filterTasksByTags() {
  const filterTags = document
    .getElementById('filter-tags')
    .value.split(',')
    .map(tag => tag
      .trim())
    .filter(tag => tag);

  const tasks = JSON
    .parse(localStorage.getItem('tasks'));
  const filteredTasks = tasks
    .filter(task => filterTags
      .every(filterTag => task.tags.includes(filterTag)));
  document.getElementById('tasks-container')
    .innerHTML = '';
  filteredTasks.forEach(displayTask);
}
```

Vygenerovaný kód 11 – Filtrování úkolů dle štítků

Toto byla opět poslední funkcionalita, která byla ve výstupu vygenerována. Poté se opět ChatGPT dotázalo, zda má pokračovat.

Po odsouhlasení pokračování bylo tentokrát přidáno řazení úkolů dle parametrů. Parametry půjde řadit priority, zbývajícím počtu dní do deadlinu, nebo počtu dní uplynulých od vytvoření. Opět bylo potřeba nejdříve přidat potřebné HTML elementy.

```
<select id="sort-criteria">
  <option value="priority">Priorita</option>
  <option value="deadline">Dny do deadlinu</option>
  <option value="creation">Dny od vytvoření</option>
</select>
<button onclick="sortTasks()">Seřadit úkoly</button>
```

Vygenerovaný kód 12 – Vstupy pro seřazení

Dále k těmto vstupům byly přidány odpovídající funkce v JavaScriptu, které samotné řazení umožňují.

```
function sortTasks() {
  const criteria = document.getElementById('sort-criteria').value;
  let tasks = JSON.parse(localStorage.getItem('tasks'));

  if (criteria === 'priority') {
    tasks.sort((a, b) => (a.priority || 5) - (b.priority || 5));
  } else if (criteria === 'deadline') {
    tasks.sort((a, b) => new Date(a.deadline) - new Date(b.deadline));
  } else if (criteria === 'creation') {
    tasks.sort((a, b) => new Date(a.created) - new Date(b.created));
  }

  document.getElementById('tasks-container').innerHTML = '';
  tasks.forEach(displayTask);
}
```

Vygenerovaný kód 13 – Funkce pro seřazení úkolů

Předposlední vygenerovanou funkcionalitou je rozšíření původních filtrů. Několik kroků zpět aplikace vygenerovala pouze filtrování úkolů na základě štítků. Tehdy implementace dalších filtrů nebyla vůbec vygenerována.

Projevilo se buď špatné sestavení promptu, kdy jsem veškeré filtry nepopsal do jedné odřádky, a nebo špatné pochopení promptu chatbotem, který odpověď vygeneroval pouze v pořadí, které jsem mu předložil a nedokázal si je propojit.

Každopádně bylo opět vygenerováno několik vstupů pro uživatele.

```
<input type="number" id="filter-priority" placeholder="Filtr podle priority">
<input type="date" id="filter-deadline" placeholder="Filtr podle data deadlinu">
<input type="number" id="filter-days-since-creation" placeholder="Filtr podle dnů od vytvoření">
<button onclick="applyFilters()">Aplikovat filtry</button>
```

Vygenerovaný kód 14 – Vstupy pro filtrování

K nim byla vygenerovaná funkce *applyFilter*, která postupně projde všechny vyplněné vstupy k filtrování a vypíše pouze ty, které filtrem prošly.

```
function applyFilters() {
  const filterPriority = document.getElementById('filter-priority').value;
  const filterDeadline = document.getElementById('filter-deadline').value;
  const filterDaysSinceCreation = document.getElementById('filter-days-since-creation').value;
  let tasks = JSON.parse(localStorage.getItem('tasks'));

  if (filterPriority) {
    tasks = tasks.filter(task => task.priority === parseInt(filterPriority));
  }
  if (filterDeadline) {
    tasks = tasks.filter(task => new Date(task.deadline).toDateString() === new Date(filterDeadline).toDateString());
  }
  if (filterDaysSinceCreation) {
    tasks = tasks.filter(task => {
      const daysSinceCreation = (new Date() - new Date(task.created)) / (1000
* 60 * 60 * 24);
      return daysSinceCreation >= parseInt(filterDaysSinceCreation);
    });
  }

  document.getElementById('tasks-container').innerHTML = '';
  tasks.forEach(displayTask);
}
```

Vygenerovaný kód 15 – Funkce pro filtrování

V tento moment to vypadalo, že aplikace pomalu začala přepřínovat své kontextové okno, jelikož sama nenavrhovala další funkcionality, které jsem jí v promptu zadal. Víceméně všechny body byly již splněny, ovšem část jednoho bodu zůstala nekompletní.

U štítků bylo specifikováno, že by mělo jít použít už ty, které byly někdy použity. Ovšem z dosavadních odpovědí tato funkcionality stále implementována nebyla.

Jelikož jsem tento fakt věděl, chtěl jsem vyzkoušet, jestli aplikace pořád pozná, co se jí stále nepovedlo vygenerovat. Zadal jsem prompt: „*Implementuj další funkcionality, kterou jsem ti na začátku zadal a zatím v aplikaci není.*“

K mému překvapení opravdu aplikace rozpoznala, že ještě nebyla implementována možnost využívat již vytvořené štítky. Opět tedy bylo přidáno několik HTML elementů, které slouží k vytvoření nových štítků, výběru štítků a tlačítko pro přidání štítku. Již zde vidím jeden potencionální problém, který více popíšu ve zhodnocení výstupu.


```
<label for="task-tags">Štítky:</label>
<input type="text" id="new-tag" placeholder="Přidat nový štítek">
<select id="existing-tags">
  <option value="">Vyberte existující štítky</option>
</select>
<button type="button" onclick="addNewTag()">Přidat štítek</button>
```

Vygenerovaný kód 16 – Práce se štítky

K práci se štítky byly přidány dále další 2 funkce. První funkce slouží ke zpracování inputu a uložení samotných štítků.

```
function addNewTag() {
  const newTag = document.getElementById('new-tag').value.trim();
  if (!newTag) return;

  const tags = new Set(JSON.parse(localStorage.getItem('tags')) || []);
  tags.add(newTag);
  localStorage.setItem('tags', JSON.stringify(Array.from(tags)));

  updateTagsDropdown();
  document.getElementById('new-tag').value = '';
}
```

Vygenerovaný kód 17 – Přidání nového štítku

Druhá z nich poté upravuje select se štítky, které byly již dříve vytvořeny.

```
function updateTagsDropdown() {
  const tags = JSON.parse(localStorage.getItem('tags')) || [];
  const dropdown = document.getElementById('existing-tags');
  dropdown.innerHTML = '<option value="">Vyberte existující štítky</option>';
  tags.forEach(tag => {
    dropdown.innerHTML += `<option value="${tag}">${tag}</option>`;
  });
}

document.addEventListener('DOMContentLoaded', function() {
  updateTagsDropdown();
});
```

Vygenerovaný kód 18 – Update štítků

Toto byl konec zprávy a bylo potřeba opět připomenout ChatGPT, že pořád nebyly splněny veškeré funkcionality. Použil jsem tedy stejný prompt jako při minulém konci zprávy.

ChatGPT si opět uvědomilo, která část není hotová. Tentokrát šlo o správu filtrů. Ty bylo potřeba nejprve zobrazit. K tomuto účelu byla vytvořena rozsáhlá funkce

displayActiveFilters. Ta nejprve získala hodnoty z již implementovaných inputů daných filtrů. Poté tyto hodnoty filtry postupně prošla a postupně vytvořila elementy, které mohou tyto filtry jednoduše smazat. Zároveň díky tomu budou filtry i lépe viditelné a tím i více přehledné.

```
Object.keys(filterValues).forEach(key => {  
  const value = filterValues[key];  
  if (value) {  
    const filterDiv = document.createElement('div');  
    filterDiv.textContent = `${key}: ${value}`;  
    const removeButton = document.createElement('button');  
    removeButton.textContent = 'Odstranit';  
    removeButton.onclick = function() {  
      document.getElementById(`filter-${key}`).value = '';  
      applyFilters(); // Reapply filters after removal  
    };  
    filterDiv.appendChild(removeButton);  
    filtersList.appendChild(filterDiv);  
  }  
});
```

Vygenerovaný kód 19 – Správa aktivních filtrů

Poté jen stačilo na konec stávající funkce *applyFilters* vložit volání funkce *displayActiveFilters*. Poslední částí byla implementace zrušení všech filtrů, kde pouze stačilo resetovat hodnotu inputů a zavolat opět původní funkci *applyFilters*. Zde odpověď skončila.

V tento moment chybí již poslední bod zadání, a to je úprava úkolů. Opět jsem ChatGPT zadal již potřetí prompt vyzývající k dokončení dosud neimplementovaných funkcionalit.

V odpovědi však ChatGPT začalo psát o dynamické správě štítků. V ten moment jsem generování zastavil a tentokrát již chatbotovi pomohl.

Za své nedorozumění se omluvil a poté začal s implementací. U HTML stačilo přidání tlačítka do tvorby elementu ve funkci *displayTask*. V JavaScriptu poté bylo nutné vytvořit celou funkci *editTask* pro úpravu. Ta využívá původní formulář pro tvorbu úkolů, který změní do módu pro úpravu a poté do něj dosadí hodnoty z úkolu vybraného pro úpravu.

Tímto je celý kód funkcionalit aplikace hotový. Byly implementovány veškeré funkcionality, které byly zadány v původním promptu pro generování.

4.3.3.3. Zhodnocení vygenerovaného kódu aplikace

Z počátku byly generované výstupy kvalitní. Veškerý kód dával smysl a splňoval dobře zadání, které mu bylo v promptu popsáno. Body promptu byly splňovány postupně, čímž byl

vývoj s generováním celkově zpomalen, avšak šlo díky tomu lépe rozumět aplikaci jako celku. Byly nejprve popsány základní funkce a objekt samotného úkolu a poté se dříve generované výstupy doplnily. Pokud by uživatel byl zkušenějším programátorem, určitě by šlo do promptu přidat větu, která by požadovala výpis celého kódu a tím celý proces urychlit.

Přidávání úkolů bylo implementováno správně, stejně jako označování úkolů jako hotových. Problém se neobjevil ani u mazání úkolů. Menší problém se ukázal u jejich výpisu, kde pokud se nevyplnilo nepovinné pole s prioritou a deadline, tak se ve výpisu zobrazil text ‚N/A‘. Nejedná se však o velký problém, to v úpravě aplikace vyřeší jednoduchá podmínka při výpisu úkolu.

Poté začalo implementování štítků, u kterých se již jejich implementace neprovedla příliš ideálně. Bylo sice možné přidávat i více štítků na jeden úkol, ovšem jen zadáním štítků do vstupu, přičemž zde byly odděleny čárkami. Takové vkládání není u podobných rozšířených aplikací obvyklé a spíše je nepraktické. Ovšem jelikož v promptu nebylo přesně definováno, jak se mají štítky přidávat, tak to nelze považovat za kritickou chybu.

S tím souvisí i filtrování úkolů podle štítků. Úkoly lze filtrovat i pomocí více štítků, ale opět je potřeba je vepsat všechny do vstupu a v něm je oddělit čárkou.

Správa štítků taky není implementována správně, jelikož pokud vytvářím nový úkol, a vybírám jej z těch již vytvořených, tak jejich seznam je v selectu, kde lze vybrat štítek jen jeden. Ovšem je správně ošetřeno, že pokud do nových štítků vložím štítek, který je již vytvořený, tak se nevytvoří znovu a není tím vytvořena duplicita. Navíc pokud štítek vytvářím u úkolu, a ne předem ve správě štítků, tak se pak neukáže ve filtru.

Co se týče správy aktivních filtrů, tak ta funguje také správně, ovšem řazení lze zapnout jen dle jednoho parametru, čímž je tato funkcionalita omezena. Ovšem i zde ChatGPT neporušil původní prompt, jelikož tato informace nebyla explicitně zadána.

Posledním bodem implementace bylo upravování úkolů. Zde vytvořil chatbot celou funkci, která však pracovala se vstupy pro vytvoření úkolů. Tato funkce však nefungovala správně a úkol nakonec upravit vůbec nejde.

Základ aplikace je připraven relativně dobře. Avšak jsou zde části, které nefungují, jak mají a také jsou zde části, které mohou poškozovat celkovou UX aplikace. Na tyto části se budu zaměřovat během opravování aplikace, a následně později v u jejího testování.

Todo List App

Filtry

Aktivní filtry:

Štítky

Štítky:

Seřazení

Nový úkol

☐ **ahoj1** - popis1
Tags: stitek1
(Deadline: 2024-06-03, Priority: 1)

☐ **ahoj2** - popis2
Tags: stitek2
(Deadline: 2020-10-10, Priority: 4)

☐ **ahoj3** - popis3
Tags: stitek3
(Deadline: 2022-12-12, Priority: 4)

Obrázek 9 – Aplikace před úpravami UX a UI

4.3.4. Vývoj aplikace – uživatelské rozhraní

Nyní je čas na celkovou úpravu vzhledu aplikace. Aplikaci budu stylovat pomocí CSS, jelikož se nejedná o příliš velkou aplikaci a není potřeba přidávat přehlednější rozšíření jako SASS nebo SCSS.

Bylo proto potřeba vytvořit soubor style.css, který se poté napojí na samotný HTML soubor. Ihned po vytvoření souboru mi po odentrování za titulkem HTML stránky Copilot napověděl samostatné napojení CSS souboru na HTML.

```
<> index.html > html > head
1  <!DOCTYPE html>
2  <html lang="cs">
3  <head>
4  |   <meta charset="UTF-8">
5  |   ✨ <title>Todo List App</title>
6  |   <link rel="stylesheet" href="style.css">
7  | </head>
8  | <body>
9  | |   <h1>Todo List App</h1>
10 |
```

Obrázek 10 – Našeptání Copilota pro připojení CSS souboru

Pomocí tabulátoru jsem provedení našeptání provedl a tím jej implementoval. Dále bylo potřeba dle grafického návrhu implementovat prvky, se kterými se nadále bude pracovat. V tomto případě se jedná o font a ikony.

Do GitHub Copilot Chatu jsem tedy zadal prompt: „Implementuj mi do mého projektu *Material Design Ikony*.“ Jelikož jsou tyto ikony dostupné přes CDN, tak opravdu stačilo přidat jenom jeden řádek do HTML kódu. Podobný postup fungoval i pro vložení fontu Roboto. Copilot tedy neměl žádný problém ani s poskytnutím validní URL adresy.

1. Přidejte odkaz na Material Design Icons CSS do hlavičky vašeho HTML souboru. Tento odkaz zahrnuje všechny ikony z knihovny Material Design Icons.

```
<link href="https://fonts.googleapis.com/icon?family=Materi
```

2. Nyní můžete používat ikony v HTML kódu. Ikony se přidávají pomocí názvu ikony obklopeného třídou `material-icons`.

```
<i class="material-icons">face</i>
```

Výše uvedený kód přidá ikonu s obličejem. Změnou textu uvnitř elementu `<i>` na název jiné ikony změníte zobrazenou ikonu.

Obrázek 11 – Výstup GitHub Copilot Chatu na prompt pro implementaci ikon

Dále jsem začal se samotným stylováním jednotlivých elementů stránky. Prvním z nich bylo rozvržení stránky na levé menu a pravou hlavní část. Situaci jsem chtěl původně použitím

CSS gridů, ale poté jsem si zkusil opět vygenerovat odpověď. Copilot mi navrhl použití `display:flex` s nastavením šířky stránky.



Obrázek 12 – Implementace rozložení stránky

Dalšími úpravami ve stylech stránky bylo upravení tlačítek a inputů. V tomto případě se nejednalo o žádné složité změny. Stačilo zde správně nastavit pouze barvy, paddingy a bordery. Bylo pouze nutné se řídit podle grafického návrhu, nebo se dle něj alespoň inspirovat. Tento krok by také bylo možné provést v GPT pomocí image input funkce, která dokáže rozpoznávat obrázky. Nejednalo se ovšem o úkol, který by zabral podstatnou část času.

Jakmile byly navrženy hlavní prvky aplikace, mohl jsem nyní už jenom pracovat na UX samotné aplikace. Pro rychlejší fungování uživatele jsem filtry, řazení a práci se štítky přesunul do pravé části aplikace. V případě responzivního designu by tak bylo jednoduché pouze element nastavit jako overlay a udělat z něj další menu.

Dále už byly úpravy spíše o přemístění některých prvků, což byla jen jednoduchá práce s HTML souborem. Jednalo se například o posouvání či shlukování tlačítek a inputů a jejich správné seřazení.

Obrázek 13 – Aplikace po dokončení UX a UI úprav

4.3.5. Úprava aplikace

Aplikace po vygenerování obsahovala několik chyb, které bylo třeba vyřešit. Některé funkcionality fungovaly, jiné fungovaly jen částečně. Některé části byly i nefunkční, nebo velmi špatně vyřešené.

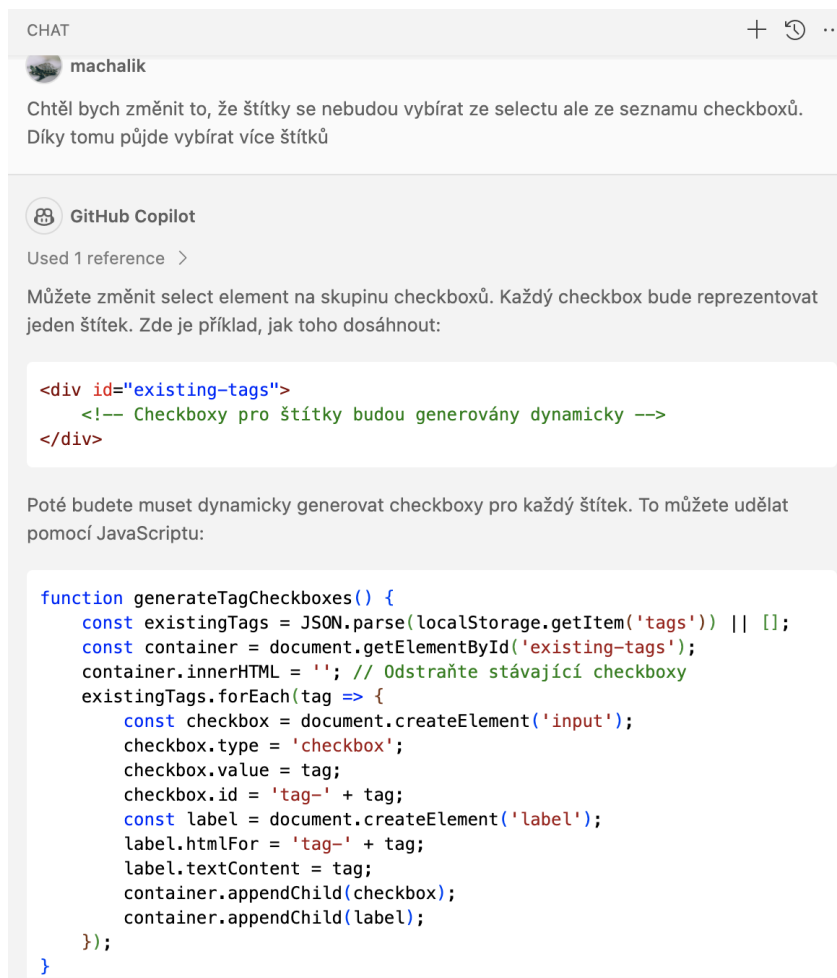
4.3.5.1. Úprava výpisu priorit a deadlinu

Jako první jsem v chybách po generování zmínil vypisování N/A u chybějících volitelných parametrů. Tento problém jsem vyřešil nakonec pouze vypisováním pomlčky, která je v tomto případě přehledná. Přišlo mi to jako vhodnější řešení pro orientaci uživatele v úkolech. Problém jsem chtěl nejprve řešit přidáním podmínky, která by vypisovala parametry pouze, pokud by byly vyplněny. Takto mi přijde, že uživateli je jasnější, že parametry nevyplnil a dodatečně je může vyplnit.

4.3.5.2. Úprava práce se štítky

Štítky nebyly původně implementovány zcela špatně, ale spíše neprakticky. Hlavním problémem bylo to, že při filtrování štítků nebylo možné filtrovat štítky ve větším množství, ale jen po jednom. To z toho důvodu, že do filtry byl implementován HTML select, který v tomto případě výběr více štítků nedovolil.

Úprava tedy spočívala pouze v tom, že bylo potřeba kód upravit tak, aby štítky nebyly jako select, ale jako seznam checkboxů. S tím si dokonce poradil Copilot Chat, který dokázal připravit funkci `generateTagCheckboxes`, která tuto funkcionalitu řeší.



Obrázek 14 – Změna selectu na checkboxy u štítků

4.3.5.3. Úprava úkolů

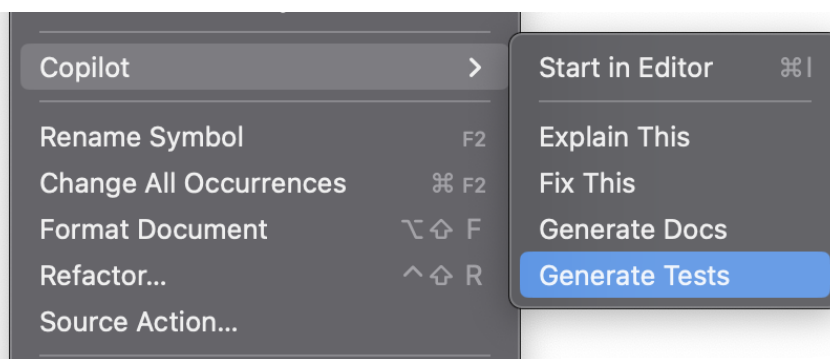
Posledním bodem bylo upravování úkolů, které však příliš nefungovalo. Problém byl nakonec způsoben pouze špatným implementováním do kódu, kde nebylo správně napsáno přepínání módu. Nakonec stačilo pouze dopsat funkci *preventDefault* k funkci pro vytváření úkolu.

4.3.6. Testování aplikace

Posledním krokem při vývoji aplikace je její testování. V tomto případě jsem se rozhodl pro automatizované testování, protože mým cílem v této práci je předvést co nejvíce generování kódů. Avšak ani s manuálním testováním a jeho návrhem by nemělo ChatGPT problém. Původně jsem pro tento úkol vybral framework Jest, který slouží pro testování aplikací naprogramovaných v jazyce Javascript. Nejprve jsem však chtěl vyzkoušet, jak si s tímto úkolem poradí GitHub Copilot integrovaný do samotného vývojového prostředí.

4.3.6.1. Testování aplikace pomocí GitHub Copilota

GitHub Copilot nabízí generování testů již po přidání tohoto doplňku do Visual Studio Code. Nejprve jsem tedy vyzkoušel tuto možnost pro generování testů. Copilot během několika vteřin dokázal vygenerovat test pro uložení nového úkolu do localStorage prohlížeče. Vygenerovaný test však obsahuje problémy, které mi hned na první pohled vadily.



Obrázek 15 – Nabídka Copilota ve Visual Studio Code

Prvním problémem bylo mockování localStorage. Ihned na první pohled jsem poznal, že vygenerovaný test byl právě testem frameworku Jest, který jsem chtěl použít později. Ovšem Jest má problémy s testováním localStorage. Nelze příliš jednoduše používat localStorage jako v Javascriptovém kódu, jelikož pro Jest testy je používána knihovna jsdom. Ta práci s localStorage pomocí getterů a setterů jako samotný Javascript neumí. Řešením mohlo být Proto jsem tedy vyzkoušel jiný způsob testování. Další myšlenkou bylo, že budu pouze vkládat úkoly pomocí již implementované Javascript funkce, a tyto úkoly budu vyhledávat přímo na stránce a ne pouze v localStorage dané aplikace. Zde jsem však narazil na další problém. Úkol je v původní funkci vytvářen tak, že je nastaven eventListener na tlačítko pro submit.

Pokud je u formuláře zavolán submit, tak se úkol uloží do localStorage a její obsah se poté vypíše. Není zde tedy nastavený parametr HTML tagu *onclick*. Jelikož si Jest nedokáže jednoduše poradit ani s eventListenerem, vyzkoušel jsem tedy jiný způsob.

4.3.6.2. Testování aplikace pomocí ChatGPT 4

Nakonec jsem se rozhodl změnit samotný způsob, jak budu aplikaci testovat. Rozhodl jsem se tedy testovat pomocí Robot Framework, který již znám a vím, že pro tento úkol bude možná nakonec i efektivnější. Robot Framework funguje na základě WebDriverů, respektive v mém případě ChromeDriveru, který má samotnou programovou kontrolu nad webovým

prohlížečem. To tedy ve výsledku znamená, že test otevře samotný prohlížeč a přímo s daným oknem interaguje.

Jako úvodní prompt jsem vložil obsah souborů *index.html* a *app.js* a poté jsem dopsal, aby byly vygenerovány 3 automatizované testy pomocí Robot Frameworku.

Jako odpověď jsem dostal celý postup instalace Robot Frameworku pomocí správce balíčků pip, včetně příkazů pro vložení do příkazové řádky. Jelikož již Robot Framework i Chromedriver v počítači mám, tento krok jsem přeskočil. Poté již začalo samotné generování testů. Na konci bylo napsáno, aby všechny testy byly vloženy do jednoho souboru a poté spuštěny pomocí příkazu *robot* v příkazové řádce.

Jelikož jsem celou strukturu testů ještě rozdělit, zeptal jsem se, zda mají všechny testy v jednom souboru. Po položení tohoto dotazu jsem dostal odpověď, která doporučovala organizované a udržitelné rozdělení úkolů.

4.3.6.3. *Vygenerování testů*

Na začátku jsem obdržel strukturu souborů pro testy. Ta byla rozdělena na *resources* a *test_cases*. Ve složce *resources* by se měly nacházet pomocné soubory pro testování. Soubor *keywords.robot* obsahoval klíčová slova pro otevření a zavření okna. Zde se pouze využilo funkcionality Robot Frameworku, a tak bylo klíčové slovo „Open Browser“ s parametry URL a prohlížečem nahrazeno za vlastní klíčové slovo „Open Todo List“. Klíčové slovo „Close Browser“ bylo nahrazeno za „Close Todo List“. V souboru *variables.robot* pak byla pouze jedna proměnná, a to ta s URL stránky localhostu, kterou jsem musel přepsat.

Testy byly napsány téměř správně. Byly vygenerovány 3 testy, které obsahovaly příklady testování úkolů a štítků.

4.3.6.4. *Zhodnocení vygenerovaných testů*

Jako první byl vygenerovaný test pro tvorbu úkolů, který byl téměř celý správně. Název, popis i štítky byly vygenerovány ve správném formátu, stejně jak priorita. Jedinou chybou bylo vygenerování termínu. Klasický HTML input typu date v českém prostředí prohlížeče je ve formátu „d-m-Y“, vygenerovaná odpověď byla ve formátu „Y-m-d“, tudíž zde bylo potřeba datum pouze upravit do správného formátu – původně by se totiž vložilo datum 20. února 40427 (řetězec byl „2024-04-27“).

```
*** Settings ***
Resource    ../resources/keywords.robot
Resource    ../resources/variables.robot

*** Test Cases ***
Add New Task
    Open Todo List
    Input Text    id:task-title    "Koupit mléko"
    Input Text    id:task-description    "Koupit mléko v obchodě"
    Input Text    id:task-tags    "nákupy, domov"
    Input Text    id:task-deadline    "08-05-2024"
    Input Text    id:task-priority    3
    Click Button    css:button[type="submit"]
    Page Should Contain    Koupit mléko
    Close Todo List
```

Vygenerovaný kód 20 – Test pro vytváření úkolů

Druhý test sloužil po vytvoření nového štítku. U tohoto testu byl problém tentokrát s nalezením správně XPath tlačítka pro vytvoření štítku. Proto bylo potřeba upravit cestu k tlačítku. Po úpravě na správnou cestu již test procházel v pořádku.

```
*** Settings ***
Resource    ../resources/keywords.robot
Resource    ../resources/variables.robot

*** Test Cases ***
Add New Tag
    Open Todo List
    Input Text    id:new-tag    "online"
    Click Button    //button[@id='addNewTagButton']
    Page Should Contain    online
    Close Browser
```

Vygenerovaný kód 21 – Test pro vytváření štítků

Poslední test bylo smazání úkolu. Nejprve ChatGPT vygenerovalo test pouze stisknutí tlačítka pro smazání a kontrolu obsahu, který by se na stránce neměl vyskytovat. Zde však test nebyl vypsán správně, jelikož se test pokoušel smazat úkol, který byl na stránce vytvořen v prvním testu. Ten však není uložen v localStorage jako v klasické aplikaci, jelikož se při spouštění testu vytváří kompletně nová instance prohlížeče, která nemá uložené žádné data v cookies nebo právě v localStorage. Tento problém jsem opravil nakonec tím, že jsem pouze do posledního testu zkopíroval obsah testu prvního. I v tomto testu byla potřeba upravit XPath na správné tlačítko ve formuláři, které úkol smaže. Tento test tedy také nebyl vygenerován správně a bylo potřeba jej upravit.

```
*** Settings ***
Resource    ../resources/keywords.robot
Resource    ../resources/variables.robot

*** Test Cases ***
Delete Task
    Open Todo List
    Input Text    id:task-title    "Koupit mléko"
    Input Text    id:task-description    "Koupit mléko v obchodě"
    Input Text    id:task-tags    "nákupy, domov"
    Input Text    id:task-deadline    "2024-05-08"
    Input Text    id:task-priority    3
    Click Button    css:button[type="submit"]
    Page Should Contain    Koupit mléko
    Click Button    //button[@id='deleteTaskButton']
    Page Should Not Contain    Koupit mléko
    Close Browser
```

Vygenerovaný kód 22 – Test pro mazání úkolů

Testy byly z větší části tedy vygenerovány správně, avšak ve všech případech bylo potřeba je správně upravit tak, aby fungovaly. Na konci testu bylo potřeba ještě upravit soubor `package.json`, který specifikoval spouštění testů ze složky `test_cases` a ukládání výsledků do složky `results`.

```
"scripts": {
  "test": "robot -d ./results ./tests/test_cases"
}
```

Obrázek 16 – Úprava souboru package.json

Testy jde nyní spouštět jednoduše přes příkazovou řádku přímo ve vývojovém prostředí pomocí příkazu `npm test`. Po provedení všech příkazů byly vygenerovány i výstupní soubory `log.html`, `output.xml` a `report.html`, které obsahují veškeré informace o provedených testech.

Test Cases Report

Summary Information

Status:

All tests passed

Start Time:

20240507 22:21:11.070

End Time:

20240507 22:21:22.782

Elapsed Time:

00:00:11.712

Log File:

[log.html](#)

Test Statistics

Total Statistics	Total	Pass
All Tests	3	3

Statistics by Tag	Total	Pass
No Tags		

Statistics by Suite	Total	Pass
Test Cases	3	3
Test Cases . 1 Add Task	1	1
Test Cases . 2 Add Tag	1	1
Test Cases . 3 Remove	1	1

Obrázek 17 – Vygenerovaný výstup do souboru report.html

Ve výsledku se dá říct, že testy byly vygenerovány poměrně dobře, bylo potřeba však upravit některé části testovacího kódu.

5. ZHODNOCENÍ PŘÍNOSŮ VYUŽITÍ NÁSTROJŮ STROJOVÉHO UČENÍ

V poslední části této práce se pokusím shrnout veškeré poznatky, které jsem během její tvorby získal. Budu se soustředit hlavně na body, které jsem zmiňoval v kapitole o přínosech a rizicích spojených s využitím strojového učení ve vývoji software. Jedná se totiž o body, na které jsem se během práce soustředil, protože jsou s tímto tématem v současné době nejčastěji spojovány. Rozhodl jsem se každý z těchto bodů rozebrat u každé části tvorby aplikace.

5.1. Navrhování funkcionalit aplikace

Během návrhu aplikace jsem nenarazil na nějaké větší problémy. Výstup byl kvalitní, dokonce i s větším množstvím návrhů, než které mě původně k aplikaci napadly. Odpověď byla vygenerována v rádech sekund. V tomto ohledu musím říct, že ChatGPT dokázalo velmi efektivně pomoci s vytvářením aplikace.

Odpověď měla lehce přes 800 tokenů, což pro kontextové okno čítající až 32 tisíc tokenů u ChatGPT 4 bylo dost pod limitem. V tomto případě jsem se tedy nemusel bát ztráty kontextu. Další konverzace o návrhu by byla bezpečná a chatbot by s vysokou pravděpodobností nehalucinoval. V porovnání bez využití chatbota by tento úkol zabral podstatně více času a určitě by při zavolání dalšího promptu přišel rychle s dalšími nápady na rozšíření.

Další nápady mohly přijít po prozkoumání jiných nejvíce používaných aplikací pro správu úkolů, avšak v tomto případě to nebylo nutné. Nahrazení v tomto odvětví vývoje není dle mě pravděpodobné, ale určitě dokáže práci značně urychlit v dostatečné kvalitě.

5.2. Navrhování uživatelského rozhraní aplikace

U navrhování uživatelského rozhraní má spokojenost nebyla taková, jak u návrhu aplikace. Vytvoření promptu bylo sice stejně jednoduché, avšak v tomto případě by i lepší prompt nedokázal vygenerovat o moc lepší odpověď. Samotná příprava promptu nebyla náročná, stačilo jen shrnout funkcionality aplikace. Výstup ovšem v tomto případě nebyl kvalitní.

Uizard sice dokázal přijít se zajímavými prvky jako byly tlačítka, logo nebo rozmístění stránky. Tyto prvky vypadaly moderně a do aplikace se i hodily, avšak jejich rozmístění nedávalo smysl. Texty v návrhu byly nelogické a jednalo se spíše jen o výplň. Uživatelský

zážitek takové aplikace by byl velice špatný. Návrh prvků trval několik málo minut, avšak zde by ani více času na tento úkol příliš nepomohlo.

Proto si myslím, že nástroje strojového učení v blízké době nedokážou nahradit zkušené UX nebo UI designery ani jejich spolupráci se softwarovými vývojáři. Nadále bude potřeba vzájemná spolupráce při tvorbě UX a UI, které budou pro uživatele nejvhodnější.

5.3. Vývoj aplikace

Vývoj aplikace byl hlavním bodem mé práce. Spokojenost s celkovým kódem byla poměrně dobrá. Vygenerování celé aplikace sice v tomto případě trvalo kolem 10 minut, ovšem opravy vyskytlých problémů později času zabralo daleko více. Kvalita projektu nebyla ideální, avšak pomocí vytvořených funkcionalit a z nich vytvořeného promptu vyšla kvalitní kostra, která šla dobře upravovat. Ke konci generování aplikace došlo již ke ztrátě kontextu, které způsobilo halucinování. To se stalo u vytvoření poslední funkce pro úpravu úkolu, kde bylo při její úpravě volána funkce *addTask*, i když se tato funkce v původní aplikaci vůbec nevyskytovala, a proto bylo potřeba funkci upravit.

Úprav v aplikaci bylo o něco více. Jednalo se o úpravu štítků. Původně se štítky daly filtrovat pouze pomocí *selectu*, kde byl vybrán pouze jeden štítek. Nakonec bylo toto vyřešeno pomocí vytvoření checkboxů se štítky. Celkově úpravy zabraly i s implementováním uživatelského rozhraní hodinu a půl. Tato doba by byla bez použití nástrojů strojového učení dost individuální dle znalostí daného programátora.

U začátečníka by mohl nastat problém například v tom, že by nevěděl, proč mu daná část aplikace nefunguje a nemohl by na to dostatečně reagovat. Dá se říct, že čas, který by byl využit na programování aplikace byl nahrazen za čas, který byl vynaložen na opravu chyb či nesprávně implementovaných částí kódu.

Nástroje strojového učení také dokážou pomoci s vyhledáváním různých informací, které by jinak musely být hledány například na Internetových fórech. Po nějaké době také začíná ztráta kontextu aplikace, jelikož u psaní kódu lze velmi rychle dosáhnout maximálního kontextového okna z důvodu častého dotazování nebo vkládání částí kódu bez širšího kontextu.

V této aplikaci se nenacházel žádný citlivý kód obsahující například různá hesla, API klíče nebo šifrovací algoritmy, jelikož se jedná o aplikaci, která byla vytvořena již nespočetněkrát. Proto zde nehrozí riziko v zpřístupnění kódu další společnosti. Dle mého názoru by šlo tuto aplikaci vytvořit lépe, pokud by byl počáteční prompt více rozsáhlý. I přes to si myslím, že

by se postupně s růstem aplikace ztrácel kontext a vznikala potřeba neustále kód například do chatbota vkládat.

5.4. Testování aplikace

Testování aplikace bylo možné v tomto případě také urychlit díky nástrojům strojového učení. Vytvoření jednoduchých testů nebylo příliš náročné, trvalo také celkově pár minut a až na úpravu několika XPath k HTML elementům v *.robot* souborech bylo testování rychlé. S testovacími soubory jsem byl téměř naprosto spokojený.

Nemusel jsem v tomto případě ani po roce procházet dokumentaci Robot Frameworku. Pokud bych chtěl vytvořit další testy, s promptem popisující podobu testu by si ChatGPT 4 určitě poradilo. I když kontextové okno bylo dostatečné, i tak chatbot nedokázal získat správnou XPath k elementům a bylo potřeba ji doplnit manuálně.

Co se týče šíření kódu, tak i v tomto případě existuje riziko úniku soukromých dat, pokud by chatbot procházel celý projekt, který obsahuje citlivá data.

5.5. Rozšiřitelnost a budoucí vývoj

Pokud se podívám na projekt jako celek, je dle mého názoru lehce rozšiřitelný. Veškeré funkcionality, které byly při jejich návrhu na začátku aplikace vygenerovány, tak nebyly implementovány.

Návrh uživatelského rozhraní i přes nekvalitní výstup dokáže poskytnout možnou inspiraci, které části by mohly být dále vyvíjeny a jakým způsobem by toho šlo dosáhnout. To poskytuje dostatečný základ pro to, aby mohl být vyvíjen další kód, který by byl do aplikace později přidán.

Nevýhodu vidím v tom, že je celá logika aplikace napsána v jednom souboru. Při dokončení měl JavaScriptový soubor kolem 250 řádků a HTML soubor kolem 80. V pozdějších fázích vývoje by tento způsob mohl být ještě více nepřehledný, než je teď. Ovšem i toto by dle mého názoru šlo pomocí chatbota jako je ChatGPT vyřešit a myslím, že by dokázal na uspokojivé úrovni vymyslet vhodné řešení pro přehlednost celého projektu.

Tím by se mohl urychlit další vývoj a také celková efektivita vývojáře či celého týmu, který by na projektu pracoval.

ZÁVĚR

V teoretické části práce jsem představil, jak nástroje strojového učení fungují a co znamenají nejznámější fráze z tohoto odvětví. Také byly zmíněny jazykové modely, které jsou hlavními motory chatbotů dnešní doby. Tyto nástroje jsou nesmírně komplexní a vyvíjí je velké týmy vývojářů v největších světových firmách. Pro jejich správné fungování je potřeba obrovské množství dat, které tyto firmy získávají po světě různými způsoby.

Jejich využití při tvorbě aplikace v praktické části přineslo spoustu výhod. Celkové tvoření aplikace bylo záživnější a nebylo potřeba se soustředit na vytváření jednoduchých částí, se kterými si například ChatGPT snadno poradí. Jednalo se například o vytváření jednoduchých funkcí, či tvorba HTML kódu s elementy dané stránky. Tím se urychlilo celé vytváření aplikace. Bylo mi navrženo také několik funkcionalit, se kterými bych na začátku projektu třeba nepočítal a bylo možné dle toho upravovat kód. Nebylo potřeba na Internetu dohledávat některé funkcionality JavaScriptových funkcí, které se týkaly například eventListenerů. Jednoduché části kódu mohly být tím pádem pouze generovány. To samé platí například i u návrhu vzhledu, kde jsem dostal zajímavé návrhy, jelikož jsem nevyužíval CSS knihovny jako například Bootstrap.

S využitím těchto nástrojů se vyskytlo i několik problémů. UX aplikace nebylo navrženo prakticky vůbec a v tomto ohledu bych byl u většího projektu závislý na designerech. Některé části kódu například vůbec nefungovaly, a proto musely být opravovány. Čas, který jsem ušetřil na psaní kódu jsem tedy poté musel z velké části vynaložit na jeho opravu. Pokud jsem se tázal příliš dlouho, chatbot začal v pár případech halucinovat.

Aplikace je však ve výsledku docela lehce rozšiřitelná, dále by šlo implementovat například více oken, například pro práci se štítky, podporu více jazyků nebo například import a export úkolů a štítků.

Jelikož se nepracovalo s databází, hesly či šifrováním, nebylo potřeba příliš řešit bezpečnost aplikace. Pokud by se však jednalo o větší aplikaci, bylo by nutné zvážit, které části kódu budou moci nástroje strojového učení zpracovávat a zda je možné, že by mohla uniknout citlivá data, na kterých by se jazykové modely dále učily. V tomto případě by bylo potřeba zvážit individuální případy a buď některé části kódu vůbec neposkytovat, nebo zvolit vhodné předplatné, které zaručuje, že se modely na těchto datech učit dále nebudou.

Profese vyskytující se u vývoje software nejspíše v nejbližších letech dle mého názoru nezmizí, ovšem budou poznat rozdíly mezi lidmi, kteří tyto nástroje dokážou efektivně a správně využívat, a mezi těmi, kteří je buď nepoužívají vůbec, nebo špatným způsobem. V nejbližších měsících budou programátoři nuceni vybírat ke své práci správné nástroje a správně je používat, například zvolením kvalitních promptů.

SEZNAM POUŽITÉ LITERATURY

- [1] CHOLLET, François. *Deep learning v jazyku Python: knihovny Keras, Tensorflow*. Přeložil Rudolf PECINOVSKÝ. Knihovna programátora (Grada). Praha: Grada Publishing, 2019. ISBN 978-80-247-3100-1.
- [2] BURKOV, Andriy. *The Hundred-Page Machine Learning Book*. 2019. ISBN 978-1999579517.
- [3] Typy strojového učení. *Elements of AI* Online. 2024. Dostupné z: <https://course.elementsofai.com/cs/4/1>. [cit. 2024-04-07].
- [4] *Semi-Supervised Learning in Artificial Intelligence*. Online. DATAROBOT. Datarobot.com. 2024. Dostupné z: <https://www.datarobot.com/blog/semi-supervised-learning/>. [cit. 2024-04-07].
- [5] *What is reinforcement learning?* Online. AMAZON. Aws.amazon.com. 2024. Dostupné z: <https://aws.amazon.com/what-is/reinforcement-learning/>. [cit. 2024-04-07].
- [6] SAP. *Co je generativní AI?* Online. 2024. Dostupné z: <https://www.sap.com/cz/products/artificial-intelligence/what-is-generative-ai.html>. [cit. 2024-04-07].
- [7] David Slouka. *Modely v modelu: In-context learning posouvá hranice LLM*. Online. Prg.ai. 2023. Dostupné z: <https://prg.ai/modely-v-modelu/>. [cit. 2024-04-07].
- [8] *Improving language understanding with unsupervised learning*. Online. Openai.com. 2018. Dostupné z: <https://openai.com/research/language-unsupervised>. [cit. 2024-04-07].
- [9] *Tokenizer*. Online. Openai.com. 2023. Dostupné z: <https://platform.openai.com/tokenizer>. [cit. 2024-04-07].
- [10] *GPT a další velké jazykové modely*. Online. Michal Kašpárek. 2024. Dostupné z: <https://michalkasperek.cz/llms.html>. [cit. 2024-04-07].
- [11] *Understanding AI*. Online. Understanding AI. 2024. Dostupné z: <https://www.understandingai.org/p/large-language-models-explained-with>. [cit. 2024-04-07].
- [12] *Welcome to the Gemini era*. Online. GOOGLE. Google Gemini. 2024. Dostupné z: <https://deepmind.google/technologies/gemini/#gemini-1.0>. [cit. 2024-04-07].
- [13] *Llama 2: Open Foundation and Fine-Tuned Chat Models*. Online. META. Llama 2: Open Foundation and Fine-Tuned Chat Models. 2024. Dostupné

- z: <https://ai.meta.com/research/publications/llama-2-open-foundation-and-fine-tuned-chat-models/>. [cit. 2024-04-07].
- [14] FORBES. Threads Now Fastest-Growing App In History—With 100 Million Users In Just Five Days. Online. FORBES. Forbes.com. 2024. Dostupné z: <https://www.forbes.com/sites/siladityaray/2023/07/10/with-100-million-users-in-five-days-threads-is-the-fastest-growing-app-in-history/>. [cit. 2024-04-08].
- [15] LUPA.CZ. Google představil svou odpověď na ChatGPT, konverzační AI Bard nasadí do vyhledávání. Online. LUPA.CZ. Lupa.cz. 2024. Dostupné z: <https://www.lupa.cz/aktuality/google-predstavil-svou-odpoved-na-chatgpt-konverzacni-ai-bard-nasadi-do-vyhledavani/>. [cit. 2024-04-08].
- [16] ANTHROPIC. *Introducing Claude*. Online. ANTHROPIC. Anthropic.com. 2023. Dostupné z: <https://www.anthropic.com/news/introducing-claude>. [cit. 2024-04-08].
- [17] ANTHROPIC. *Meet Claude*. Online. GITHUB. Github.com. 2023. Dostupné z: <https://www.anthropic.com/claude>. [cit. 2024-04-08].
- [18] GITHUB. Introducing GitHub Copilot: your AI pair programmer. Online. GITHUB. Github.blog. 2021. Dostupné z: <https://github.blog/2021-06-29-introducing-github-copilot-ai-pair-programmer/>. [cit. 2024-04-08].
- [19] GITHUB. *The world's most widely adopted AI developer tool*. Online. GITHUB. Github.com. 2023. Dostupné z: <https://github.com/features/copilot>. [cit. 2024-04-08].
- [20] GITHUB. *About GitHub Copilot Chat*. Online. GITHUB. Docs.github.com. 2023. Dostupné z: <https://docs.github.com/en/copilot/github-copilot-chat/about-github-copilot-chat>. [cit. 2024-04-08].
- [21] ANTHROPIC. *About GitHub Copilot*. Online. GITHUB. Github.com. 2023. Dostupné z: <https://docs.github.com/en/copilot/about-github-copilot>. [cit. 2024-04-08].
- [22] JETBRAINS. *JetBrains AI*. Online. JETBRAINS. JetBrains.com. 2023. Dostupné z: <https://www.jetbrains.com/ai/>. [cit. 2024-04-08].
- [23] *GPT-Restaurace*. Online. In: GRUDL, David. Umelligence.cz. Dostupné z: <https://www.umelligence.cz/media/gpt-restaurace.avif>. [cit. 2024-05-01].
- [24] *Uizard – Product*. Online. Uizard.io. 2024. Dostupné z: <https://uizard.io/product/>. [cit. 2024-05-01].

- [25] *Uizard – Pricing*. Online. Uizard.io. 2024. Dostupné z: <https://uizard.io/pricing/>. [cit. 2024-05-01].
- [26] *GitHub Copilot Workspace: Welcome to the Copilot-native developer environment*. Online. Github.blog. 2024. Dostupné z: <https://github.blog/2024-04-29-github-copilot-workspace/>. [cit. 2024-05-01].
- [27] *LMSYS Chatbot Arena Leaderboard*. Online. Lmsys.org. 2024. Dostupné z: <https://chat.lmsys.org/?leaderboard>. [cit. 2024-05-01].
- [28] *Amazon CodeWhisperer*. Online. Amazon.com. 2024. Dostupné z: <https://aws.amazon.com/codewhisperer/>. [cit. 2024-05-01].
- [29] GITHUB. *Research: quantifying GitHub Copilot's impact on developer productivity and happiness*. Online. Github.blog. 2022. Dostupné z: <https://github.blog/2022-09-07-research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/>. [cit. 2024-05-01].
- [30] CHEN, Lingjiao; ZAHARIA, Matei a ZOU, James. *How is ChatGPT's behavior changing over time?* Palo Alto: Stanford University, 2023.
- [31] Coding on Copilot 2023 – Data Shows Downward Pressure on Code Quality. Online. 2024, s. 24. Dostupné z: <https://gitclear-public.s3.us-west-2.amazonaws.com/Coding-on-Copilot-2024-Developer-Research.pdf>. [cit. 2024-05-01].
- [32] *První kroky s ChatGPT*. Online. GRUDL, David. Umelligence.cz. 2023. Dostupné z: <https://www.umelligence.cz/chatgpt-zaklady>. [cit. 2024-05-09].
- [33] OPENAI. *How ChatGPT and our language models are developed*. Online. Openai.com. 2024. Dostupné z: https://help.openai.com/en/articles/7842364-how-chatgpt-and-our-language-models-are-developed#h_2df02d4917. [cit. 2024-05-09].
- [34] *Žijeme s AI: dopady, přínosy a rizika*. Online. GRUDL, David. Umelligence.cz. 2023. Dostupné z: <https://www.umelligence.cz/rizika-prinosy>. [cit. 2024-05-01].

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AI	Artificial Intelligence
LLM	Large Language Model
GPT	Generative Pre-Trained Transformer
LaMDA	Language Model for Dialogue Applications
PaLM	Pathways Language Model
DPH	Daň z přidané hodnoty
API	Application Programming Interface
JSON	JavaScript Object Notation
UX	User Experience
UI	User Interface
IDE	Integrated Development Environment
HTTP	Hypertext Transfer Protocol
IT	Informační Technologie
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
XPath	XML Path Language

SEZNAM VYGENEROVANÝCH KÓDŮ

Vygenerovaný kód 1 – HTML struktura aplikace	33
Vygenerovaný kód 2 – Vytvoření nového úkolu	34
Vygenerovaný kód 3 – Uložení a zobrazení úkolu	34
Vygenerovaný kód 4 – Výpis úkolů	35
Vygenerovaný kód 5 – Dokončení úkolu	35
Vygenerovaný kód 6 – Odstraňování úkolů	36
Vygenerovaný kód 7 – Vstup pro přidání štítků	36
Vygenerovaný kód 8 – Zpracování štítků	36
Vygenerovaný kód 9 – Výpis štítků	37
Vygenerovaný kód 10 – Vstup pro přidání štítku s tlačítkem	37
Vygenerovaný kód 11 – Filtrování úkolů dle štítků	37
Vygenerovaný kód 12 – Vstupy pro seřazení	38
Vygenerovaný kód 13 – Funkce pro seřazení úkolů	38
Vygenerovaný kód 14 – Vstupy pro filtrování	38
Vygenerovaný kód 15 – Funkce pro filtrování	39
Vygenerovaný kód 16 – Práce se štítky	40
Vygenerovaný kód 17 – Přidání nového štítku	40
Vygenerovaný kód 18 – Update štítků	40
Vygenerovaný kód 19 – Správa aktivních filtrů	41
Vygenerovaný kód 20 – Test pro vytváření úkolů	50
Vygenerovaný kód 21 – Test pro vytváření štítků	50
Vygenerovaný kód 22 – Test pro mazání úkolů	51

SEZNAM OBRÁZKŮ

Obrázek 1 – Předpovídání dalšího tokenu [23]	13
Obrázek 2 – Výstup nástroje Tokenizer po převodu českého textu na tokeny	14
Obrázek 3 – Výstup nástroje Tokenizer po převodu anglického textu na tokeny	14
Obrázek 4 – Rozhraní webové aplikace ChatGPT	17
Obrázek 5 – Rozhraní webové aplikace Google Gemini.....	18
Obrázek 6 – Stránka doplňku GitHub Copilot pro IDE VSCode	27
Obrázek 7 – Promptové okno aplikace Uizard se sepsaným promptem.....	30
Obrázek 8 – Návrh uživatelského rozhraní aplikací Uizard	31
Obrázek 9 – Aplikace před úpravami UX a UI	43
Obrázek 10 – Našeptání Copilota pro připojení CSS souboru	44
Obrázek 11 – Výstup GitHub Copilot Chatu na prompt pro implementaci ikon	44
Obrázek 12 – Implementace rozložení stránky.....	45
Obrázek 13 – Aplikace po dokončení UX a UI úprav	46
Obrázek 14 – Změna selectu na checkboxy u štítků.....	47
Obrázek 15 – Nabídka Copilota ve Visual Studio Code	48
Obrázek 16 – Úprava souboru package.json	51
Obrázek 17 – Vygenerovaný výstup do souboru report.html.....	52

SEZNAM TABULEK

Tabulka 1 – Žebříček chatbotů v programování [27]	19
---	----

SEZNAM PŘÍLOH

Příloha 1 – CD se zdrojovými kódy a textem práce