VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INTELIGENTNÍ PROSTŘEDÍ PRO ROZŠIŘOVÁNÍ ZNALOSTÍ PROGRAMOVÁNÍ V JAZYCE PYTHON FORMOU SAMOSTUDIA

INTELLIGENT ENVIRONMENT FOR EXTENDING PYTHON PROGRAMMING KNOWLEDGE VIA SELF-LEARNING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. JAN KREJČÍ**
AUTHOR

VEDOUCÍ PRÁCE — **doc. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

154307

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Krejčí Jan, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Softwarové inženýrství
Název:	**Inteligentní prostředí pro rozšiřování znalostí programování v jazyce Python formou samostudia**
Kategorie:	Informační systémy
Akademický rok:	2023/24

Zadání:

1. Seznamte se s možnostmi výuky a zdokonalování v programování s využitím aktuálních nejlepších konverzačních agentů typu ChatGPT a interakce se začínajícími programátory přímo při vytváření kódu.
2. Zpracujte přehled možností generování individuálních testovacích příkladů pro procvičování základních i pokročilých témat s vazbou na výuku v rámci předmětu Skriptovací jazyky (ISJ) na FIT VUT v Brně.
3. Navrhněte a implementujte systém, který bude schopen zadávat a vyhodnocovat příklady studentům, kteří se chtějí zdokonalit v programování v Pythonu, případně se lépe připravit na zkoušku ISJ.
4. Vyhodnoťte vytvořený systém na základě interakce se studenty kurzu ISJ, případně dalšími, a diskutujte míru zlepšení kvality postupně odevzdávaných kódů.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- dle doporučení vedoucího

Při obhajobě semestrální části projektu je požadováno:

- funkční prototyp řešení

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Smrž Pavel, doc. RNDr., Ph.D.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2023
Termín pro odevzdání:	17.5.2024
Datum schválení:	21.12.2023

Abstrakt

Tato práce se zaměřuje na vytvoření inteligentního prostředí pro rozšiřování znalostí programování v jazyce Python formou samostudia. Klíčovým prvkem práce je implementace mechanismů poskytování zpětné vazby. Pro tento účel byly analyzovány možnosti a omezení velkých jazykových modelů. Vytvořený systém využívá klasifikačních modelů, které na základě analýzy studentských projektů poskytují personalizovanou zpětnou vazbu. Systém byl nasazen a testován v rámci kurzu Skriptovací jazyky na FIT VUT v Brně a měl pozitivní ohlasy od studentů. Výsledek práce představuje ucelený a funkční systém, který splnil svůj původní záměr a přispěl k efektivnějšímu a interaktivnějšímu procesu vzdělávání v oblasti programování v jazyce Python.

Abstract

This thesis aims to create an intelligent environment for extending the knowledge of Python programming through self-study. A key element of this work is the implementation of feedback mechanisms. For this purpose, the capabilities and limitations of large language models have been analyzed. The developed system uses classification models to provide personalized feedback based on the analysis of student projects. The system has been deployed and tested in the Scripting Languages course at FIT BUT and received positive feedback from students. The outcome presents a comprehensive and functional system that has fulfilled its original intention and contributed to a more effective and interactive Python programming education process.

Klíčová slova

Python tutor, inteligentní výukový systém, poskytování zpětné vazby, analýza kódu, jazykový model, klasifikační model, jazyk Python, Docker

Keywords

Python tutor, intelligent tutoring system, providing code feedback, code analysis, language model, classification model, Python language, Docker

Citace

KREJČÍ, Jan. *Inteligentní prostředí pro rozšiřování znalostí programování v jazyce Python formou samostudia.* Brno, 2024. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Pavel Smrž, Ph.D.

Inteligentní prostředí pro rozšiřování znalostí programování v jazyce Python formou samostudia

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doc. RNDr. Pavla Smrže, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jan Krejčí
13. května 2024

Poděkování

Touto cestou děkuji vedoucímu mé práce panu Doc. RNDr. Pavlu Smržovi, Ph.D. za vedení a odborné konzultace. Děkuji také své rodině za podporu při celém studiu.

Obsah

Kapitola 1

Úvod

V dnešní době se vzdělávání a rozvoj dovedností stávají stále důležitějšími faktory úspěchu v moderním světě. Zejména v oblasti informačních technologií se neustále vyvíjejí nové metody a nástroje, které umožňují efektivnější a interaktivnější vzdělávání. S nárůstem zájmu o tuto oblast se rovněž zvyšuje potřeba efektivních a inovativních metod výuky, které by umožnily studentům získat a zdokonalit své programátorské dovednosti. Nabývá tak významu využití inteligentních systémů, kde hraje klíčovou roli poskytování zpětné vazby, které umožňuje studentům nejen lépe porozumět dané látce, ale také se aktivně zapojit do procesu učení a zdokonalování svých dovedností.

Tato práce si klade za cíl vytvořit inteligentní prostředí pro rozšiřování znalostí programování v jazyce Python formou samostudia. Python se v poslední době stal jedním z nejrozšířenějších a nejpopulárnějších programovacích jazyků. Klíčovým prvkem této práce je implementace mechanismů poskytování zpětné vazby za pomoci předtrénovaných modelů, které uživatelům umožní získat cenné informace o jejich řešení. Tímto způsobem mohou studenti sledovat svůj pokrok v rámci kurzu Skriptovací jazyky na FIT VUT v Brně a lépe se připravit na závěrečnou zkoušku.

Text práce je strukturován do několika hlavních kapitol, které postupně rozvíjejí jednotlivé aspekty navrhovaného systému. Úvodní kapitoly tvoří teoretickou část, kterou bylo nutné nastudovat pro navržení systému pro poskytování zpětné vazby. Zabývá se analýzou problematiky inteligentních výukových systémů a důležitostí poskytování zpětné vazby v procesu vzdělávání. Také jsou zde představeny možnosti a omezení dnešních velkých jazykových modelů. Dále je v kapitole 3 popsán proces zpracování dat a dotrénování klasifikačních modelů, které generují specifickou zpětnou vazbu k studentským projektům. Následující kapitola 4 obsahuje analýzu požadavků, návrh uživatelského rozhraní a architekturu webové aplikace. Implementace v kapitole 5 se zabývá postupem realizace celého systému, popisuje vývojové prostředí, použité technologie a další zásadní části systému. V závěru práce je popsán průběh a výsledky testování a také možné rozšíření systému do budoucna.

Kapitola 2

Rozbor řešené problematiky

Úvod této kapitoly se zaměřuje na inteligentní výukové systémy s důrazem na výuku programování. Představuje Bloomovu taxonomii jako rámec pro definici vzdělávacích cílů a zdůrazňuje význam podrobné zpětné vazby v procesu vzdělávání, zejména pak v oblasti programování, kde přispívá k rozvoji dovedností studentů. Dále popisuje poskytování zpětné vazby na základě různých možností analýzy kódu.

V druhé části kapitoly je popsán význam umělé inteligence, zejména možnosti a omezení velkých jazykových modelů, v kontextu vzdělávání a online výuky. Tyto modely mají schopnost poskytovat personalizovanou podporu, automatizovat procesy a zlepšovat efektivitu výuky. Následně se kapitola detailněji zaměřuje na konkrétní jazykové modely, které v současnosti nalézají široké uplatnění.

Závěr této kapitoly se věnuje existujícím řešením v oblasti vzdělávání v programování v jazyce Python, které poskytují prostředí pro interaktivní učení a procvičování dovedností.

2.1 Inteligentní výukový systém

Inteligentní výukový systém (dále jen ITS z anglického Intelligent Tutoring System) je podle publikace [19] definovaný jako systém v oblasti vzdělávání určený k výuce studentů na individuální úrovni, kterým automaticky poskytuje zpětnou vazbu. Tato technologie má kořeny sahající až do 60. let 20. století, ačkoliv první oficiální definice ITS byla formulována až v 90. letech téhož století. V současné době existuje několik různých ITS, které pokrývají široké spektrum vzdělávacích oborů a pomáhají tak s výukou statisíců studentů každý rok.

Podle Rus a Graesser [24] je jednou z hlavních výzev v oblasti ITS zaměřených na interakci v přirozeném jazyce proces vyhodnocování odpovědí studentů. Během dialogu mezi systémem a studentem je sledováno, zda student dokáže správně formulovat každý z očekávaných kroků, které představují ideální řešení na daný problém. Posuzování odpovědí v podobě studentských zdrojových kódů je komplexní a vyžaduje sofistikované techniky a algoritmy pro analýzu a vyhodnocení.

2.1.1 ITS pro výuku programování

Publikace [10] a [26] poukazují, že ITS se ukázal jako velmi efektivní pro osvojení dovedností v různých oblastech, a to především v programování, kde je integrace dovedností rozhodující i v úvodních kurzech. Zvláště individuální procvičování je považováno za klíčový faktor při získávání odborných znalostí, kdy se výuka soustředí na speciálně navržené úkoly s následnou zpětnou vazbou a opakováním za účelem zdokonalení kritických oblastí. ITS mohou

tyto cvičení poskytovat v optimalizované podobě tím, že reagují na potřeby jednotlivých studentů a umožňují opakování dle potřeby.

Existují důkazy o efektivitě těchto cvičení. Studie [7, 14] prokázaly, že začátečníci, kteří studovali kurz s důrazem na programovací vzory a systematické osvojení si dovedností, dosahovali lepších výsledků v řešení komplexnějších problémů než ti, kteří studovali tradičním způsobem.

Bloomova taxonomie

Tato podsekce popisující populární Bloomovu taxonomii vychází z publikací [2] a [28]. Bloomova taxonomie (dále jen BT) o cílech vzdělávání je jedním z klíčových konceptů v oblasti pedagogiky. Tato taxonomie slouží jako rámec pro definování a kategorizaci cílů výuky a hodnocení vzdělávacího procesu. Jedním z význačných prvků této taxonomie je rozdělení kognitivních dovedností do šesti úrovní, jejich grafické znázornění je na obrázku 2.1. Ideálně by měly být tyto dovednosti rozvíjeny v daném pořadí, neboť dovednosti na vyšší úrovni se opírají o dovednosti nižších úrovní.

1. **Pamatování (Knowledge)**: Na nejnižší úrovni BT se nachází pamatování. Tato úroveň se zaměřuje na schopnost studenta vybavit si informace a fakta. Příklady zahrnují například paměťové testy definic či postupů.

2. **Porozumění (Comprehension)**: Na této úrovni je žádoucí, aby si studenti nejen pamatovali požadované informace, ale také aby je chápali. Klasická demonstrace porozumění zahrnuje vysvětlování pojmů, srovnávání souvisejících fakt a ilustraci.

3. **Aplikace (Application)**: Aplikace se týká schopnosti studenta použít získané znalosti a dovednosti v konkrétních situacích. To zahrnuje řešení problémů v praxi a použití abstraktních informací v konkrétních situacích. Její hlavní rozdíl od úrovně porozumění spočívá v tom, že při prokazování porozumění jsou žákovi naznačeny abstrakce (pojmy), které by měl použít, zatímco úroveň aplikace vyžaduje pochopení pojmů, které se týkají konkrétního úkolu.

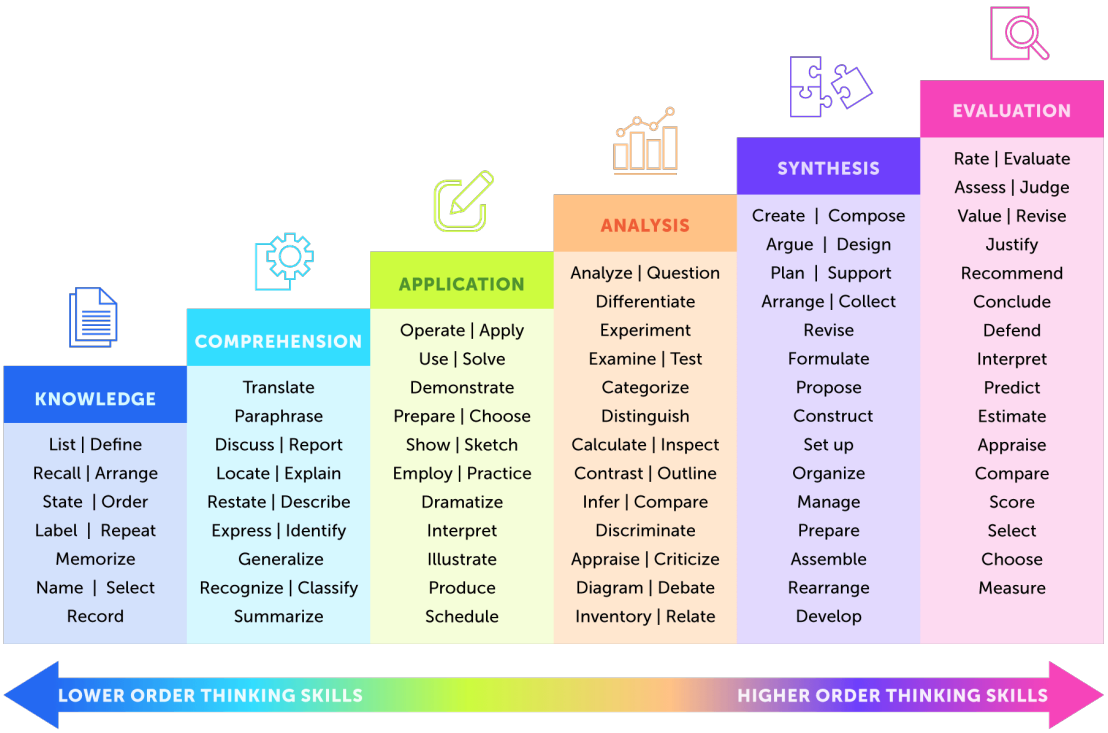
4. **Analýza (Analysis)**: Tato úroveň zahrnuje rozklad komplexních problémů na menší části, identifikaci vzorů a vztahů mezi nimi a nakonec vyvozování závěrů.

5. **Syntéza (Synthesis)**: Do syntézy spadá schopnost kombinovat různé prvky a informace k vytvoření nového celku. Studenti na této úrovni navrhují, formulují a vytvářejí nové nápady, produkty nebo řešení.

6. **Hodnocení (Evaluation)**: Nejvyšší úroveň BT se věnuje schopnosti studenta hodnotit a posuzovat informace, argumenty, koncepty nebo řešení. To zahrnuje kritické myšlení, rozhodování a formulaci vlastních názorů a představ na základě důkazů.

Na obrázku 2.1 jsou graficky znázorněná *slovesa* spojená s kognitivními procesy jednotlivých vzdělávacích úrovní. Tyto slovesa slouží k popisu konkrétních činností, které jsou potřebné pro dosažení vzdělávacích cílů, které odpovídají jednotlivým úrovním.

Učební úlohy na prvních třech úrovních BT vzdělávacích cílů se obvykle charakterizují jako relativně jednoduché. Pro jejich řešení stačí dodržet základní posloupnost kroků. Naopak úlohy na vyšších úrovních často vyžadují strategické myšlení a plánování pro dosažení



Obrázek 2.1: Vzdělávací úrovně sloužící k popisu konkrétních činností potřebných pro dosažení vzdělávacích cílů BT. Převzato z [3].

cíle. Na úrovních pamatování, porozumění a aplikace je tedy rozdíl mezi kognitivními (sledují studenta krok za krokem při řešení problému) a omezujícími (kontrolují stavy řešení, zda neporušují omezující podmínky) ITS zanedbatelný.

V kontextu vzdělávání v oblasti programování můžeme tyto úrovně vzdělávacích cílů interpretovat následovně [29]:

- **Pamatování**: Tato úroveň se zaměřuje na schopnost naučit se symboly a syntaxi programovacího jazyka, aby bylo možné napsat spustitelný kód.
- **Porozumění**: Porozumění znamená pochopení podstaty funkcí a příkazů programovacího jazyka.
- **Aplikace**: Tato úroveň představuje schopnost psát kód podle požadavků konkrétní úlohy a definovat objekty a jejich chování pomocí programovacího jazyka.
- **Analýza**: Analyzování je spojeno se schopností najít logické nebo syntaktické chyby v kódu a provádět diagnostiku problémů.
- **Syntéza**: Syntéza znamená schopnost implementovat v programovacím jazyce složité struktury, jako jsou moduly, třídy a funkce, které vzájemně komunikují a řeší komplexnější úlohy.
- **Hodnocení**: Poslední úroveň se váže k rozhodování o přístupu k řešení konkrétního problému. Studenti na této úrovni musí rozhodnout, jaké datové struktury, progra-

mové vzory a algoritmy použít pro efektivní řešení úlohy, což vyžaduje pokročilou úroveň zkušeností.

Moderní ITS pro cvičení programování poskytují širokou škálu funkcí pro podporu procesu učení v různých tématických oblastech, jako jsou programovací úlohy se zpětnou vazbou, kvízy, pseudokódy algoritmů, referenční materiály a mnoho dalších. Jsou zaměřeny na různé úrovně cílů BT. Zpětná vazba se pohybuje od hodnocení typu splněno/nesplněno, přes výpis chybových hlášení až po předdefinované nápovědy orientované přímo na daný úkol. Většina systémů používá natvrdo naprogramované modely pro konkrétní úkoly.

Mnoho těchto platforem je vytvořeno pro jednu tématickou oblast. Častým problémem jsou chybějící úrovně BT, kdy ITS poskytují studentům pouze úlohy na vysoké úrovni, aniž by poskytovaly adekvátní podporu pro rozvoj kognitivních dovedností na nižších úrovních. K plnému využití adaptivních schopností potřebuje ITS velkou banku různých úloh.

Mnoho výukových systémů využívá vizualizaci jako účinný prostředek ilustrace úlohy a jejího řešení pro snazší pochopení problému. Interaktivní vizualizace může poskytovat dodatečnou zpětnou vazbu v průběhu řešení úlohy a usnadnit tak žákovi jeho postup. Užitečná může být i textová zpětná vazba. K odhalení plné síly adaptivního hodnocení potřebuje ITS velkou banku různých úloh, ideálně generované nově dle potřeb studenta.

2.1.2 Výhody ITS

Mezi hlavní výhody ITS dle [26] patří:

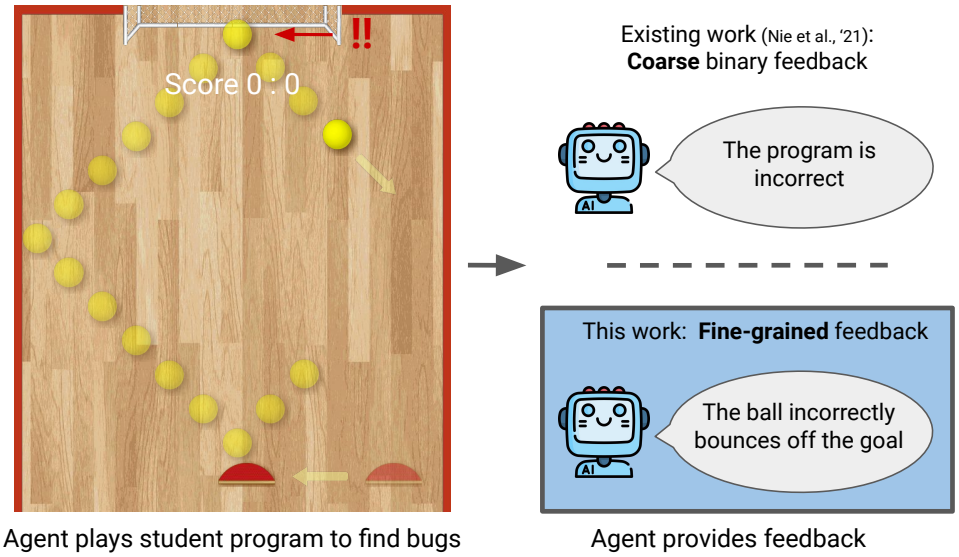
- **Ekonomická efektivita**: Z perspektivy dlouhodobé udržitelnosti lze konstatovat, že ITS představují ekonomicky efektivnější alternativu ve srovnání s tradičním pedagogickým přístupem. Například proces vyhodnocení úkolů a následné poskytnutí zpětné vazby lektorem v tradičním prostředí vyžaduje kontinuální finanční náklady. V případě ITS jsou jedinými náklady spojenými s tímto systémem náklady na jeho vývoj a potřebnou technickou podporu. Tato investice do vývoje ITS je však jednorázová a následně lze systém využívat opakovaně. Důležitým aspektem je také schopnost poskytovat personalizovanou výuku a adaptovat se na potřeby jednotlivých studentů.
- **Dostupnost**: Další významnou výhodou ITS je jeho vysoká dostupnost. Na rozdíl od lektorů a běžných učeben, které mají omezený provozní čas, je výukový systém neustále k dispozici studentům, což jim umožňuje přizpůsobit vzdělávací proces individuálním potřebám a preferencím.
- **Analýza dat**: Poslední zmíněnou výhodou ITS je možnost sběru a analýzy dat o pokroku studentů, což umožňuje kontinuální vylepšování výukových materiálů a strategií. To vede k vyšší efektivitě výuky a zlepšení výsledků studentů.

Celkově lze tedy konstatovat, že ITS představují ekonomicky výhodnou a v některých aspektech lepší alternativu k tradičnímu přístupu výuky a mohou hrát klíčovou roli v budoucnosti vzdělávání.

2.2 Poskytování zpětné vazby

Poskytování zpětné vazby hraje zásadní roli při vzdělávání. Kvalitní a pro studenta přínosná zpětná vazba však vyžaduje vysokou odbornost a poměrně značnou časovou náročnost.

Některé interaktivní programy (například počítačové hry) mohou dle [17] zahrnovat stochastické prvky, které nelze testovat tradičními unit testy a musí být hodnoceny ručně. To zabere poměrně velké úsilí opravujícím. U programů tohoto typu je tak z pohledu automatizace složitější poskytnout podrobnou zpětnou vazbu, jak je uvedeno na příkladu na obrázku 2.2.



Obrázek 2.2: Ukázka poskytnutí binární a podrobné zpětné vazby. Převzato z [17].

Zdroje [16, 17] se shodují, že podrobná zpětná vazba představuje klíčový prvek v procesu vzdělávání a rozvoje dovedností. Její důležitost spočívá v možnosti porozumět chybám. Kvalitní zpětná vazba nejen identifikuje konkrétní nedostatky v kódu, ale také vysvětluje, proč tyto nedostatky vznikly, a poskytuje návrhy na možné zlepšení. Tímto způsobem se studenti aktivně zapojují do učebního procesu, rozvíjejí svou schopnost samostatného myšlení a systematicky zdokonalují své dovednosti. Kromě toho podrobná zpětná vazba umožňuje učiteli přizpůsobit metodiku výuky podle individuálních potřeb studentů, což přispívá k efektivnějšímu vzdělávacímu procesu.

S rostoucí zájmem o vzdělání v oblasti informatiky a nárůstem online vzdělávání se ruční klasifikace stává stále méně proveditelnou. Například publikace [17] z roku 2022 uvádí, že na jedné z populárních platforem code.org[1] je registrováno více než 70 milionů studentů. Vzhledem k tomu, že ruční hodnocení odevzdaného úkolu může trvat až několik minut, nemohou tyto platformy zatím poskytovat podrobnější zpětnou vazbu o tom, zda je zadání interaktivního úkolu splněno či nikoliv.

2.2.1 Zpětná vazba při programování

Poskytování zpětné vazby studentům v úvodních kurzech programování představuje klíčový prvek v procesu jejich učení a rozvoje dovedností v této oblasti. V tradičních úvodních kurzech programování studenti obvykle obdrží zpětnou vazbu na své projekty až po uplynutí lhůty pro jejich odevzdání. Tato zpětná vazba je často vytvářena ručně hodnotiteli nebo vyučujícími asistenty a její zpracování může vyžadovat značný čas a úsilí. Jak uvádí [16],

[1] https://code.org

pokud studenti obdrží zpětnou vazbu až po termínu odevzdání a nejsou povoleny žádné opravy, omezuje to jejich možnost učit se, protože neexistuje vnější motivace ke zlepšení řešení na základě zpětné vazby.

Jednou z běžných metod zpětné vazby využívanou v současných kurzech programování je generování zpětné vazby pomocí nástrojů, které poskytují studentům automatické hodnocení. Přijetí automatických korektorů bylo alespoň zčásti pragmaticky motivováno rostoucím počtem zájemců o studium programování. Hodnotící struktury, které využívají automatický korektor, mají tu výhodu, že umožňují opakované odevzdávání a umožňují studentům okamžitou zpětnou vazbu, kterou mohou využít k vylepšení svých řešení. Studie [8, 15] prokázaly, že studenti mají tendenci odevzdávat své práce častěji v blízkosti termínů. Tím, že někteří studenti s plněním úkolů otálejí, jsou jejich výkony negativně ovlivněny. Řešením může být zařazení průběžných termínů, kdy studenti dostanou včasnou zpětnou vazbu o svých dosavadních řešeních a mohou své projekty upravit před konečným termínem hodnocení. Ačkoli je toto řešení dobrovolné, mnoho studentů tuto včasnou zpětnou vazbu vítá a zlepšují tak kvalitu své práce v každé iteraci. Okamžitá a neomezená zpětná vazba však může vést k přílišnému spoléhání se na ni. Snížit tuto závislost může omezení maximálního počtu pokusů odevzdávání či sankce za nadměrné testování. Nicméně neexistuje jasná shoda na tom, jak by měla být zpětná vazba studentům poskytována, aby se maximalizovaly její přínosy.

Publikace [17] dále uvádí, že automatické systémy často poskytují pouze binární zpětnou vazbu o tom, zda je program napsán správně, nebo ne, což je typicky dáno průchodem testů. Studenti však k lepšímu pochopení svých chyb potřebují podrobnější zpětnou vazbu vztahující se ke konkrétním konstrukcím ve svém programu.

2.3 Velké jazykové modely

Úvod této sekce vychází z publikace [13]. Velké jazykové modely (dále jen LLM z anglického Large Language Models) v posledních letech dosáhly významného pokroku v oblasti zpracování přirozeného jazyka. Tyto modely jsou natrénovány na rozsáhlém korpusu textových dat a dokáží generovat text srovnatelný s lidským, reagovat na otázky a vykonávat další jazykové úlohy s vysokou přesností.

Jedním z klíčových trendů vývoje v této oblasti je využití transformačních architektur a základního mechanismu pozornosti, které výrazně zlepšily schopnost LLM zpracovávat závislosti v přirozeném jazyce. Konkrétně architektura transformátoru využívá mechanismus vlastní pozornosti k určení relevance různých částí vstupu při generování predikcí. To modelu umožňuje lépe porozumět vztahům mezi slovy ve větě bez ohledu na jejich pozici.

Přestože LLM v posledních letech zaznamenaly velký pokrok, stále existuje mnoho omezení, která je třeba překonat. Jedním z hlavních omezení spočívá v nedostatečné schopnosti interpretace, neboť je obtížné porozumět logice, která stojí za modelovými predikcemi. To znamená, že model poskytuje výsledky, ale je obtížné pochopit, jak a proč k nim došel. Existují také etické aspekty, jako jsou obavy z dopadu na zaměstnanost, zneužití, neadekvátní nebo neetického nasazení, ztráta integrity a mnoho dalších.

2.3.1 Fine-tuning

Tato část je založena na dokumentaci *OpenAI* [20], *Google Generative AI* [9] a *Hugging Face* [11]. V kontextu strojového učení představuje jemné doladění (fine-tuning nebo také dotrénování) proces adaptace existujícího modelu pro specifickou úlohu nebo oblast. Proces

spočívá v dodatečném trénování modelu na konkrétním datasetu s cílem optimalizovat jeho výstupy. Tím lze dosáhnout lepšího přizpůsobení modelu dané problematice a nastavení specifického stylu, tónu nebo formátu odpovědí. Takto doladěný model redukuje počet tokenů a dosahuje srovnatelné výkonnosti s novějšími a rozsáhlejšími modely, čímž efektivně optimalizuje náklady bez ztráty na kvalitě.

Podle autorů [22] je tato metodologie využívána zejména v situacích, kdy slovní popis není dostačující a je zapotřebí intuitivního pochopení. Například je obtížné abstrahovat vlastnosti kvalitního článku od zkušeného spisovatele pouhým popisem. V těchto případech je snazší demonstrovat požadovaný výsledek pomocí vhodné datové sady. Klíčovým faktorem úspěšného jemného doladění jazykových modelů je dostatečný počet kvalitních trénovacích dat, přičemž optimální velikost trénovací sady závisí na konkrétní aplikaci. I při omezeném množství dat (v řádu desítek vzorků) může model dosahovat kvalitních výsledků.

Fine-tuning je dle [13] obvykle prováděn pomocí techniky učení s učitelem (supervised learning), ale existují také techniky pro doladění modelu pomocí učení s částečným dohledem (weak supervision) nebo na základě zpětné vazby od člověka.

Trénovací data pro model by měla obsahovat rozmanité ukázky konverzací, které reflektují situace v reálném provozu. Tato data by měla zahrnovat i okrajové případy, které se mohou vyskytnout až v produkčním prostředí, aby bylo zajištěno, že model dokáže adekvátně reagovat i na nečekané dotazy. Hrozí zde také riziko přetrénování, kdy model ztrácí schopnost generalizace na dosud neviděné situace.

2.3.2 RAG

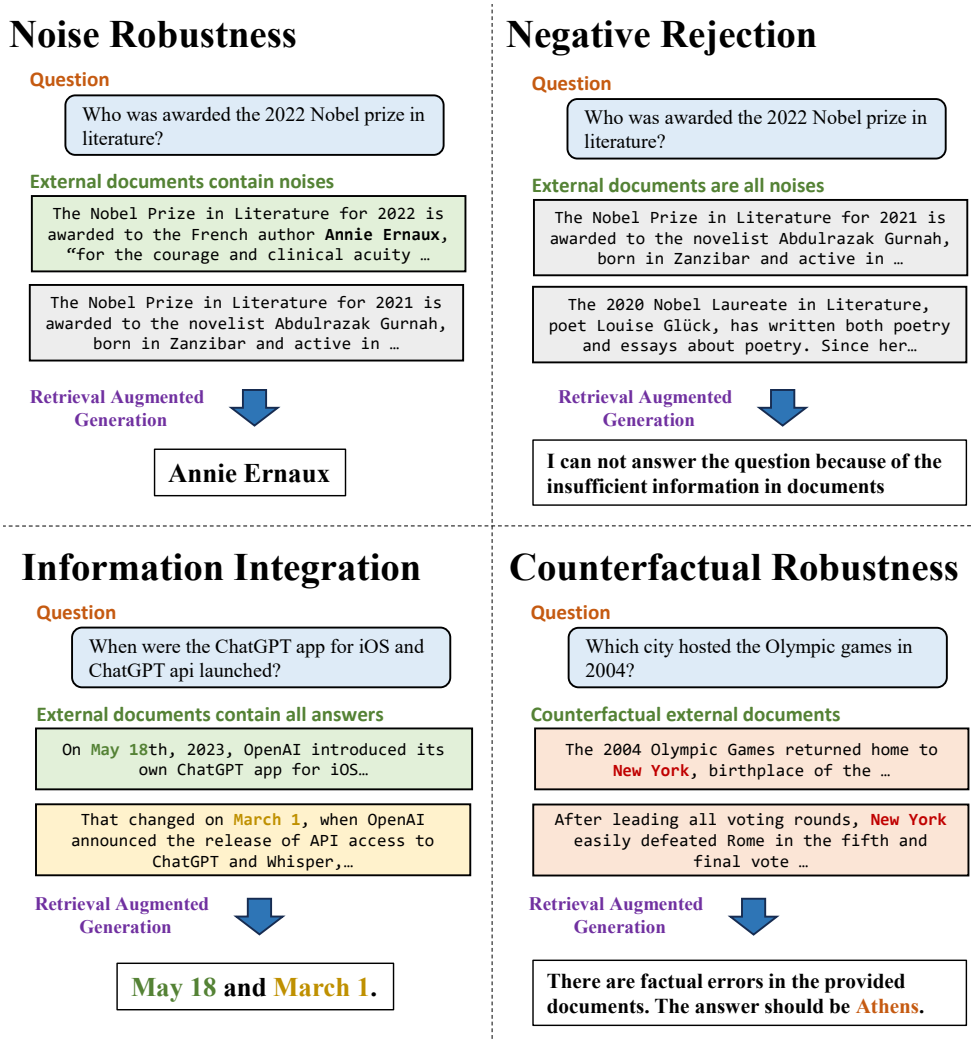
Ačkoli LLM prokázaly pozoruhodné obecné schopnosti, stále čelí zásadním výzvám, jako je zastaralost a nedostatek odborných znalostí v konkrétních oblastech. Koncept generování rozšířené o vyhledávání [5] (dále jen RAG z anglického Retrieval Augmented Generation) nabízí řešení na výše uvedené problémy. RAG totiž umožňuje modelům přístup k externím informacím, což může výrazně rozšířit jejich schopnosti.

Znalosti LLM jsou fixovány na okamžik posledního tréninku, což znamená, že nejsou schopny reagovat na aktuální události nebo nové informace. LLM využívající RAG mají možnost přistupovat k aktuálním informacím, což zvyšuje jejich přesnost a spolehlivost. Avšak použití RAG může mít i negativní stránky. Internet obsahuje množství falešných informací, což může vést k nepřesnostem a nekonzistencím ve vygenerovaných odpovědích. Ukázky takových situací jsou demonstrovány na obrázku 2.3. Zabezpečení aktuálních a relevantních dat může být náročné, zejména v dynamických oblastech. Další výzvou je optimalizace procesu vyhledávání a integrace externích informací do generovaného obsahu tak, aby byl v souladu s původním kontextem a případně s požadavky na jeho kvalitu a důvěryhodnost.

Skutečný potenciál RAG spočívá v jeho aplikacích v reálném světě. V odvětvích, jako je například zdravotnictví, kde jsou přesné a aktuální informace klíčové, nebo v oblasti financí, kde se situace na trhu mohou měnit během několika minut.

2.3.3 Umělá inteligence ve vzdělávání

Dle [25] má umělá inteligence (dále jen AI z anglického Artificial Intelligence) stále rostoucí potenciál pro zlepšení vzdělávání a může sloužit jako cenný nástroj pro učitele ve výuce. Její přínos spočívá především v analýze dat, personalizaci výuky, automatizaci administrativních úkolů a poskytování okamžité zpětné vazby studentům. Tato technologie



Obrázek 2.3: Ukázka čtyř situací a jejich ideálního řešení, které mohou nastat u LLM s rozšířeným vyhledáváním v externích zdrojích. Převzato z [5].

dokáže identifikovat individuální potřeby žáků a nabídnout jim přizpůsobené výukové materiály, což vede ke zlepšenému učebnímu procesu. Nicméně je důležité zdůraznit, že AI nemůže a neměla by nahradit klíčový lidský aspekt výuky. Učitelé mají jedinečnou schopnost vytvářet emocionální vazby se studenty, podporovat jejich sociální a emocionální rozvoj a rozvíjet dovednosti, které AI zatím nenahradí. Lidský učitel poskytuje inspiraci, motivaci, povzbuzení a vedení, které jdou nad rámec schopností AI. Integrace AI do výuky by měla být vnímána jako doplňující nástroj, který pomáhá učitelům lépe plnit svoji roli a zvýšit efektivitu výuky, nikoli jako náhrada za lidský pedagogický přístup.

Podle článku [18] existují hlavní role, které by AI mohla hrát ve vzdělávacím procesu. Mezi ně patří mentor, tutor, spolužák či simulátor. Pro každou roli je nezbytné poskytnout podrobné zadání, také označované jako *prompt*. Jak popisuje [22], prompt představuje dotaz nebo pokyn, který uživatel zadá s cílem získat optimální odpověď nebo generovaný obsah. Toto zadání lze použít ke konkrétnímu účelu dané role spolu s dalšími relevantními pokyny jako je styl tón nebo formát odpovědi, ošetřování chyb a krajní případy.

Autoři [13] popisují, že LLM mohou studentům a odborníkům ve všech fázích vzdělávání přinést řadu výhod a možností. Mohou pomáhat při rozvoji čtenářských, psacích, matematických, vědeckých či jazykových dovedností a také poskytovat studentům personalizované materiály k procvičování, shrnutí a vysvětlení, což může pomoci zlepšit výkony studentů. Kromě toho mohou LLM pomáhat také při řešení výzkumných, písemných a problémových úloh a poskytovat odbornou přípravu pro danou oblast. Jak však bylo uvedeno v úvodu této sekce, při používání těchto modelů je třeba postupovat opatrně, neboť mají i svá omezení, jako je nedostatečná interpretovatelnost, riziko zkreslení nebo nežádoucí zjednodušení.

2.4 Konkrétní jazykové modely

V návaznosti na předchozí sekci 2.3, kde jsou představeny základní koncepty LLM, jsou zde popsány aktuální modely od společností OpenAI a Google. V závěru této sekce jsou popsány i open-source alternativy k těmto komerčním modelům. U každé platformy jsou představeny vlajkové modely a jejich způsob dotrénování.

2.4.1 ChatGPT od OpenAI

ChatGPT (Generative Pre-trained Transformer) je pokročilý jazykový model vyvinutý technologickou společností OpenAI[2]. Tento model je specializovaný na generování lidského textu na základě zadaných textových vstupů. Byl trénován na obrovském množství textových dat a je schopen generovat text podobný lidskému, odpovídat na otázky a plnit další úlohy s vysokou přesností. Na popularitě nabral zveřejněním modelu GPT-3.5 v roce 2022, který dokázal generovat souvislé a komplexní odpovědi, což pro širší veřejnost představoval významný pokrok v oblasti strojového učení a zpracování přirozeného jazyka. V roce 2023 byla představena vylepšená a zpoplatněná verze GPT-4.0. Tato sekce vychází převážně z OpeanAI dokumentace [20].

Tokeny

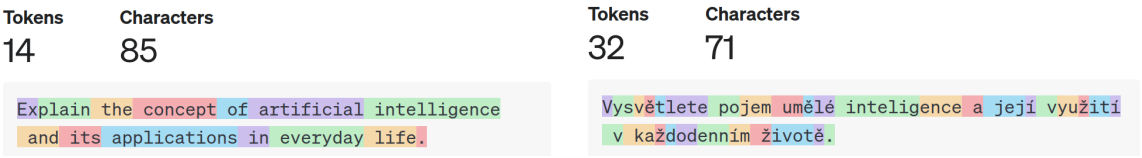
Token se považuje za základní jednotku zpracování textu. V kontextu přirozeného jazyka může token reprezentovat slova, části slov nebo dokonce jedno písmeno, v závislosti na úrovni granularity, kterou model nastavuje.

Zajímavostí je rozdíl v tokenizaci českého a anglického textu, což je dáno gramatickými a morfologickými rozdíly mezi oběma jazyky. Jelikož čeština nabízí bohatší morfologii a syntaxi, může vyžadovat komplexnější tokenizaci než angličtina, což vede k odlišnému počtu tokenů pro stejný text. Testy ukázaly, že počet tokenů pro český text může být ve srovnání s anglickým textem více než dvojnásobný. Ilustrativní příklad tokenizace z nástroje Tokenizer[3] je uveden na obrázku 2.4.

Dotrénování modelů

Koncept fine-tuning (dotrénování modelů) je podrobně popsán v podsekci 2.3.1, která se rovněž zabývá teorií, která naznačuje, že i starší specificky upravené modely mohou dosáhnout srovnatelných výsledků v porovnání s těmi novějšími. Modely od OpenAI toto potvrzují, jak je popsáno v následující kapitole 3. V případě, že jsou dosaženy dobré výsledky s aktuálně nejnovějším modelem GPT-4, je možné dosáhnout podobné kvality pomocí modelu

[2] https://openai.com
[3] https://platform.openai.com/tokenizer

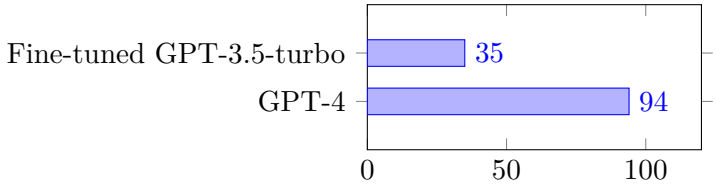


(a) Tokenizace anglického textu.

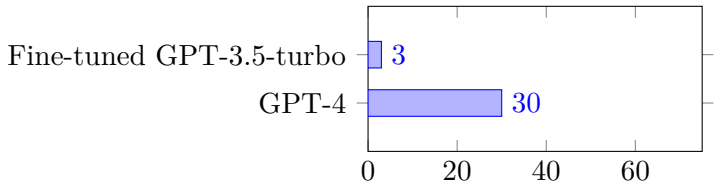
(b) Tokenizace českého textu.

Obrázek 2.4: Příklad rozdílů v tokenizaci anglického a českého textu v jazykovém modelu GPT-4 pomocí nástroje Tokenizer[3].

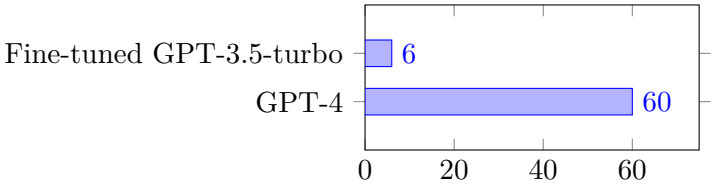
GPT-3.5-turbo, který byl upraven pomocí metody fine-tuning. Navíc, jak dokazují grafy (a) a (b) na obrázku 2.5, použití staršího modelu může výrazně snížit provozní náklady. Model GPT-3.5-turbo je téměř trojnásobně rychlejší a může ušetřit až 90% nákladů za použití jeho API v porovnání s novějším GPT-4.



(a) Rychlost odezvy na generovaný token [ms]



(b) Cena za jeden milion tokenů (vstup) [usd]



(c) Cena za jeden milion tokenů (výstup) [usd]

Obrázek 2.5: Tři grafy porovnávající výkonnost a náklady mezi dotrénovaným modelem GPT-3.5-turbo a GPT-4. Grafy zobrazují (a) rychlost odezvy na generovaný token, cenu za jeden milion tokenů (b) vstupních dat a (c) výstupních dat. Data byla získána k listopadu 2023 z oficiálního ceníku OpenAI [21].

Trénovací data pro jazykové modely obvykle obsahují různé typy informací, které pomáhají modelu lépe porozumět. Pro fine-tuning modelů od OpenAI jsou tato data strukturována do tří položek:

- **Systém (system)**: Tato informace slouží k definování charakteru a chování modelu. Systémové informace mohou obsahovat metadata o konverzaci, informace o uživatelském rozhraní a další relevantní údaje.
- **Uživatel (user)**: Tato položka obsahuje uživatelský vstup, což může zahrnovat dotazy, požadavky nebo zprávy od uživatele.
- **Asistent (assistant)**: Poslední položka obsahuje ideální odpovědi, které by měl model generovat na základě uživatelského vstupu a kontextu konverzace. Tyto odpovědi jsou často ručně připraveny a slouží jako referenční standard pro trénink modelu.

Na ukázce kódu 2.1 je formát trénovacích dat pro doladění modelu GPT-3.5-turbo, který ilustruje strukturu dat a jejich vztah k jednotlivým položkám.

```
{"messages": [
    {"role": "system", "content": "You are a sarcastic chatbot."},
    {"role": "user", "content": "Why did the chicken cross the road?"},
    {"role": "assistant", "content": "To get to the other side, duh."}
]}
```

Ukázka kódu 2.1: Formát datové sady pro doladění modelu GPT-3.5-turbo.

V sekci 2.3.1 je popsáno, že úspěšnost dotrénování modelu závisí na množství trénovacích dat. U OpenAI modelů je doporučeno začít s 50 kvalitně připravenými testovacími vzory a sledovat, zda model po vyladění vykazuje známky zlepšení. Celkový počet ale záleží na konkrétní aplikaci.

Průběh a parametry dotrénování

Při doladění modelu je klíčové pečlivě nastavit různé parametry trénování, jako je počet epoch, velikost trénovací dávky (batch size), rychlost učení (learning rate) a další hyperparametry. Správná volba těchto parametrů má zásadní vliv na rychlost a úspěšnost výsledného modelu. Je proto nezbytné je individuálně přizpůsobit konkrétnímu úkolu a dostupným datům.

Během samotného trénování modelu probíhají opakované iterace, nazývané epochy, během nichž je model opakovaně vystaven trénovacím datům a upravuje své váhy a parametry na základě chyby, kterou generuje při generování textu. Dále je taktéž sledována tzv. *training loss*, což je míra chyby modelu při generování textu na trénovacích datech. Cílem je minimalizovat tuto chybu pomocí optimalizace váh a parametrů modelu. Sledování training loss je klíčové pro posouzení pokroku trénování a detekci případných problémů nebo nedostatků v modelu.

Po ukončení každé epochy je často provedena validace modelu na validačních datech, aby bylo možné sledovat jeho úspěšnost a případně upravit parametry trénování. Po ukončení celého trénovacího procesu je model obvykle otestován na testovacích datech, aby bylo možné posoudit jeho schopnosti generovat kvalitní text na nových datech, které nebyly součástí trénovacích nebo validačních dat.

ChatGPT ve vzdělávání

Mnoho článků, včetně Loukidese [18], poukazuje na omezení ChatGPT jakožto mentora ve vzdělávacím procesu. Zatímco chatbot vyniká v poskytování základních rad, v mnoha případech nedokáže nabídnout doporučení nad rámec elementárního pochopení daného tématu,

které by studenti od mentora potřebovali. Až po konkrétním dotazu ze strany studenta, zdali by určitá vylepšení implementace nepřinesla přínos, ChatGPT dokáže takový dotaz zhodnotit a adekvátně zdůvodnit své stanovisko. ChatGPT tak dokáže rozpoznat, kdy uživatel předkládá dobré návrhy, ale to nestačí. Od mentora se očekává, že by takové návrhy měl předkládat sám, bez nutnosti explicitního vybízení ze strany studenta.

ChatGPT se dále ukázal být vynikajícím nástrojem při vysvětlování funkcionality částí kódu, kterým chce student porozumět. Jeho schopnost interpretovat a dekonstruovat složité programovací úseky do srozumitelných segmentů s vysvětlením může být pro začínající studenty zvláště cenná. Navíc je schopen dobře reagovat na dotazy ze strany uživatele.

V ideálním případě by mentor neměl sám od sebe generovat kód. To lze snadno napravit v úvodním promptu. Pokud by však student požadoval, aby ChatGPT generoval kód implementující jeho návrhy, musí pečlivě kontrolovat, zda nedošlo k chybám, z nichž některé mohou být nepatrnými změnami i v částech programu, které nesouvisejí s novým vylepšením. Student ale nemusí mít dostatečné znalosti k detekci těchto chyb, a proto je klíčové, aby byl výstup pečlivě kontrolován.

Jak popisuje Schwartz [25], současný stav schopností ChatGPT má svá omezení, ale jeho rychlý vývoj naznačuje zcela jisté a výrazné zlepšení. AI má potenciál změnit způsob výuky tím, že umožní personalizovaný přístup k vzdělávání pro každého studenta. Nicméně by neměla nahradit učitele, ale spíše jim sloužit jako užitečný nástroj. Učitelé mají jedinečnou schopnost poskytnout lidský rozměr a empatii, kterou stroje zatím nemohou replikovat.

2.4.2 Generativní AI od Google

Google je jedním z předních technologických gigantů, který aktivně vyvíjí pokročilé jazykové modely. V únoru 2023 představila společnost svého chatbota Bard[4], jehož cílem je konkurovat jazykovému modelu ChatGPT od OpenAI, který je v předchozí podsekci 2.4.1 podrobněji popsán [23].

Na konci roku 2023 byl uveden nový model Gemini[5], který je schopen efektivně zpracovat rozsáhlé objemy textových, vizuálních a zvukových dat v rámci jediného dotazu. Jednou z jeho klíčových výhod tohoto modelu je přístup k aktuálním informacím z internetu. Tato vlastnost však s sebou nese i potenciální riziko šíření dezinformací. Tento koncept má společné vlastnosti s RAG, který je detailněji popsán v podsekci 2.3.2.

Dotrénování modelů

Platforma **Vertex AI**[6], která sjednocuje veškeré cloudové služby Googlu, poskytuje uživatelům přístup k rozsáhlým generativním modelům a umožňuje jejich vytváření, testování a nasazení. Samotný proces dotrénování modelu (na platformě Vertex AI nazývaný jako Tuning) je obdobný jako u OpenAI modelů již popsaných v podsekci 2.4.1. Na ukázce kódu 2.2 je struktura datové sady pro ladění.

```
{
    "input_text": "Why did the chicken cross the road?",
    "output_text": "To get to the other side, duh."
}
```

Ukázka kódu 2.2: Formát datové sady pro doladění modelů na platformě Vertex AI.

[4] https://bard.google.com/
[5] https://gemini.google.com/
[6] https://cloud.google.com/vertex-ai

Dokumentace [9] uvádí, že k úspěšnému doladění modelu je potřeba minimálně 10 testovacích případů, optimální počet je pak kolem 100. Přesný počet však opět závisí na konkrétní úloze.

2.4.3 Hugging Face Hub

V posledních letech roste trend zaměřený na vývoj a sdílení projektů spojených s umělou inteligencí prostřednictvím specializovaných platforem využívajících systém git. Jednou z nejpopulárnějších platforem v oblasti zpracování přirozeného jazyka (dále jen NLP z anglického Natural Language Processing) a strojového učení je **Hugging Face Hub**[7], jak uvádí [1]. Hugging Face Hub vznikl s cílem usnadnit sdílení modelů a aplikací s nimi vytvořených. Tato platforma obsahuje statisíce modelů, datových sad a ukázkových aplikací, které jsou open-source a volně dostupné na přehledné platformě. V této podsekci jsou stručně popsány knihovny Transformers a SetFit, více informací o knihovnách lze nalézt v oficiální dokumentaci [11].

Transformery

V oblasti NLP se transformery staly jedním z klíčových typů neuronových sítí, získávajících v poslední době značnou popularitu. Jejich výjimečnost tkví v jejich schopnosti zachytit komplexní vzory a závislosti v textových datech, což jim umožňuje dosahovat vynikajících výsledků v široké škále úkolů. Mezi tyto úkoly patří například strojový překlad, rozpoznávání entit či generování textu. Díky jejich univerzální aplikovatelnosti jsou transformery vhodné i pro analýzu kódu, což je jeden z cílů této práce.

Open-source knihovna **Transformers**[8] je balíček jazyka Python, který obsahuje open-source implementace modelů transformátorů pro textové, obrazové a zvukové úlohy nebo jejich kombinace. V oblasti NLP umí knihovna Transformers provádět různé úlohy jako je klasifikace textu, rozpoznávání pojmenovaných entit, modelování jazyka, sumarizace, překlad, vícenásobný výběr či generování textu. Díky své kompatibilitě s knihovnami pro hluboké učení, jako je *PyTorch* či *TensorFlow*, je knihovna flexibilní a umožňuje snadnou integraci do projektů. Jednou z hlavních výhod této knihovny je snadné použití. Poskytuje rozhraní pro stažení a použití předtrénovaných modelů, umožňuje jednoduché doladění modelů na vlastních datech a následné sdílení vylepšených modelů s komunitou na platformě Hugging Face Hub.

Dotrénování modelů

Použití předtrénovaného modelu má značné výhody. Snižuje náklady na výpočet a umožňuje používat modely nejvyšší kvality, které již obsahují základní znalosti a vzory relevantní pro danou úlohu. Platforma nabízí přístup k tisícům předtrénovaných modelů, mezi které patří populární BERT, GPT2, Llama, Phi nebo Mistral, pokrývajících širokou škálu úloh v oblasti NLP.

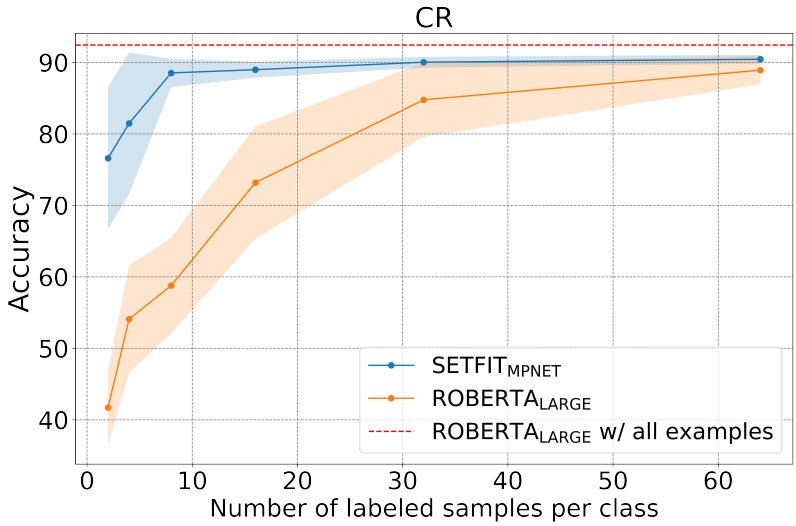
Dotrénování může být provedeno pomocí nativních knihoven (PyTorch, TensorFlow) nebo pomocí **Transformers Trainer**[9]. Tento nástroj poskytuje jednoduché, ale funkčně kompletní rozhraní pro trénování modelů nad knihovnou PyTorch. Při použití tohoto mo-

[7] https://huggingface.co/
[8] https://github.com/huggingface/transformers
[9] https://huggingface.co/docs/transformers/en/main_classes/trainer

dulu je možné nastavit hyperparametry, definovat trénovací a testovací data, volit metriky a evaluovat výkonnost modelů.

SetFit

SetFit[10] (z anglického Sentence Transformers Fine-tuning) představuje open-source nástroj určený k dotrénování modelů typu Sentence Transformers pomocí omezeného počtu trénovacích příkladů. Tato metoda, nazývaná few-shot fine-tuning, je charakterizována schopností dosahovat vysoké přesnosti s minimálním množstvím trénovacích dat (autoři uvádějí 8 vzorků na třídu), jak ukazuje obrázek 2.6. Na rozdíl od velkých modelů jako Llama nebo GPT-4, SetFit přináší efektivní řešení pro situace, kdy jsou k dispozici pouze omezená anotovaná data nebo kdy je zapotřebí rychle adaptovat menší modely na specifické požadavky. Na obrázku 2.7 je zobrazeno srovnání nákladů a výkonu mezi velkým a menším modelem, který využívá metodu SetFit. Je dosaženo téměř stejné přesnosti obou modelů, přitom náklady (časové i finanční) na trénování menšího modelu jsou výrazně nižší.



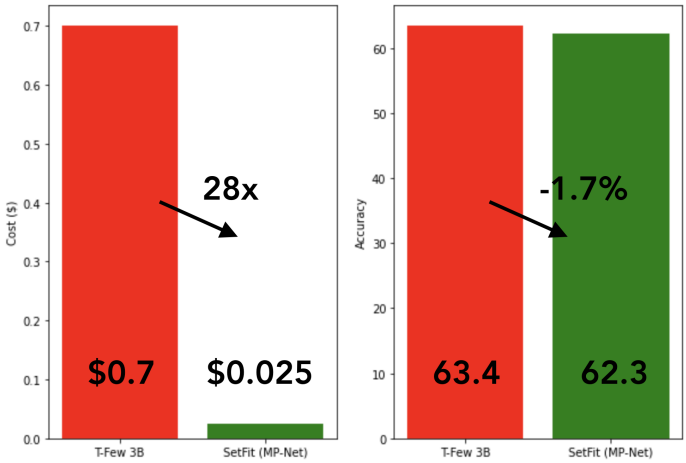
Obrázek 2.6: Porovnání přesnosti modelů v závislosti na počtu trénovacích dat při dotrénování. Převzato z [30].

Klíčový rozdíl SetFit spočívá v tom, že modely se učí rozeznávat podobnosti i rozdíly mezi trénovacími daty. Namísto tradičního instruování modelu, aby klasifikoval obrázky jako kočky nebo psy, je model instruován, aby pochopil společné rysy a rozdíly mezi různými zvířaty. Tento přístup, který se zaměřuje na pochopení podobností a odlišností v rámci vstupních dat, se běžně označuje jako metaučení.

2.5 Analýza kódu

V oblasti vývoje softwaru hraje analýza kódu klíčovou roli při zvyšování kvality, identifikaci potenciálních zranitelností a zajišťování dodržování standardů. Existuje mnoho metod a nástrojů pro analýzu kódu a poskytování zpětné vazby, které jsou více popsány ve této sekci. Pro komplexní zpětnou vazbu je nezbytné využívat kombinaci různých metod analýzy kódu.

[10] https://github.com/huggingface/setfit



Obrázek 2.7: Srovnání nákladů na trénování a průměrné úspěšnosti modelů T-Few 3B a SetFit (MPNet) s 8 anotovanými příklady na třídu. Převzato z [30].

2.5.1 Abstraktní syntaktický strom

Dle [6] je abstraktní syntaktický strom (dále jen AST z anglického Abstract Syntax Tree) stromová reprezentace syntaktické struktury zdrojového kódu programu, která odstraňuje detaily implementace a zachycuje pouze jeho syntaktickou strukturu. AST je vytvářen syntaktickým či lexikálním analyzátorem, který prochází zdrojový kód a transformuje ho do hierarchické struktury, kde každý uzel představuje určitý konstrukt programovacího jazyka, jako je například příkaz, výraz nebo deklarace proměnné.

Analýza kódu pomocí AST umožňuje provádět různé druhy analýz a manipulací s kódem, což má široké uplatnění v různých oblastech softwarového inženýrství, včetně poskytování zpětné vazby v procesu výuky programování. Na základě strukturálních charakteristik kódu může například identifikovat nadbytečné smyčky, zbytečné podmíněné příkazy či příležitosti k refaktorizaci.

Některá omezení analýzy kódu pomocí AST uvedené v publikaci [27]:

- **Nedostatečná sémantika:** AST zachycuje pouze syntaktickou strukturu kódu a nenabízí informace o jeho sémantice, což může být omezením při provádění některých druhů analýz.
- **Náročnost na implementaci**: Implementace analýzy pomocí AST může být náročná, zejména pokud je vyžadována detailní analýza sémantiky kódu.
- **Přesnost analýzy:** AST může poskytovat pouze omezenou přesnost analýzy kódu, což může vést k nepřesným výsledkům nebo chybám v poskytované zpětné vazbě.

2.5.2 Velké jazykové modely

Velké jazykové modely, které jsou podrobně popsány v sekci 2.3, představují další možnost analýzy kódu a poskytování zpětné vazby. Díky tomu, že jsou trénovány na obrovském množství textových dat a jsou schopny generovat text podobný tomu lidskému, mohou být využity k interpretaci a generování kódu, vysvětlování jeho částí a poskytování návrhů a doporučení. Dotrénování modelů specifickým případům zvyšuje jejich schopnost porozumět

a analyzovat kód, díky čemuž mohou poskytovat návrhy na vylepšení kódu, opravy chyb a optimalizace výkonu v kontextu.

2.5.3 Regulární výrazy

Regulární výrazy jsou sekvence znaků, které definují vzor pro vyhledávání a manipulaci s textem. Jsou tvořeny speciálními znaky a operátory, které umožňují definovat složité vzory textu, jako jsou například specifické řetězce znaků, znakové třídy nebo kvantifikátory. Přestože analýza založená na regulárních výrazech není z hlediska sémantického porozumění tak komplexní jako analýza AST, vyniká při identifikaci specifických vzorců. Analýzu lze navíc integrovat do automatizovaných procesů kontroly kódu, což umožňuje rychlou zpětnou vazbu o kvalitě kódu bez nutnosti rozsáhlého rozboru nebo sémantické analýzy.

Regulární výrazy mají omezenou schopnost vyjadřovat komplexní vzory a nedokáží kontrolovat sémantiku kódu. Jsou také náchylné k chybám a nepřesnostem, což může vést k nepřesným výsledkům analýzy. Na ukázce kódu 2.3 je demonstrována obtížná čitelnost při použití regulárních výrazů, která ztěžuje údržbu kódu.

```
def find_set_definitions(code):
    pattern = r'\b(\w+)\s*=\s*\{(?:[^{}]+|(?R))*\}'
    matches = re.finditer(pattern, code)
```

Ukázka kódu 2.3: Nalezení definice množiny pomocí regulárních výrazů.

2.5.4 Kontrola kvality kódu

Kromě výše zmíněných metod analýzy existuje celá řada dalších nástrojů a technik, které mohou být využity k optimalizaci a zlepšení kódu. Jak uvádí publikace [4], kontrola kvality kódu je důležitým procesem ve vývoji softwaru, který zahrnuje různé metody a nástroje pro zajištění konzistence, čitelnosti a bezpečnosti zdrojového kódu.

Statická analýza kódu

Statická analýza kódu se zaměřuje na provádění analýzy zdrojového kódu bez jeho spuštění. Tento proces může odhalit různé typy chyb a potenciální problémy, jako jsou nepoužité proměnné, nedostatečná dokumentace, redundance kódu nebo porušení definovaných konvencí.

Linting je proces kontroly zdrojového kódu na základě definovaných pravidel a konvencí, který identifikuje potenciální problémy a chyby v kódu. Formátování kódu se pak zabývá úpravou kódu tak, aby odpovídal určitým konvencím nebo stylu psaní kódu, což zajišťuje jeho konzistenci a čitelnost.

Mezi populární nástroje pro statickou analýzu kódu v Pythonu patří:

- **PyLint**[11]: Rozšířený nástroj pro statickou analýzu kódu, který provádí rozsáhlou kontrolu na základě definovaných pravidel a konvencí. Poskytuje důkladnou kontrolu zdrojového kódu, včetně detekce nepoužitých proměnných, nedostatečné dokumentace, redundance kódu a dalších potenciálních problémů.
- **Mypy**[12]: Nástroj pro statickou typovou kontrolu Python kódu. Pomáhá odhalit chyby spojené s typovou nekonzistencí v kódu a zlepšuje srozumitelnost a bezpečnost apli-

[11] https://www.pylint.org/
[12] https://mypy-lang.org/

kace tím, že umožňuje explicitní specifikaci typů proměnných, argumentů funkcí a návratových hodnot.

- **Bandit**[13]: Nástroj speciálně navržený pro analýzu zabezpečení Python kódu. Prohledává zdrojový kód a identifikuje potenciální bezpečnostní chyby a zranitelnosti, jako jsou SQL injection, XSS (Cross-Site Scripting), pevně vložené (hard-coded) přístupové údaje a další.
- **Flake8**[14] Komplexní nástroj, který kombinuje kontrolu již zmíněného PEP 8 s analýzou syntaxe.

2.6 Existující řešení

V oblasti ITS a online vzdělávání je dostupná široká škála kvalitních kurzů zaměřených na programování v jazyce Python. Tyto kurzy nabízejí personalizované a adaptivní prostředí pro studenty různých úrovní znalostí. Mezi tyto kurzy patří například platforma Codecademy či Leetcode, které jsou v této sekci popsány podrobněji. V závěru této sekce je představen Python Tutor, který již pár let slouží studentům předmětu Skriptovací jazyky (dále jen ISJ) na FIT VUT v Brně jako nástroj pro kontrolu a hodnocení projektů.

2.6.1 Codecademy

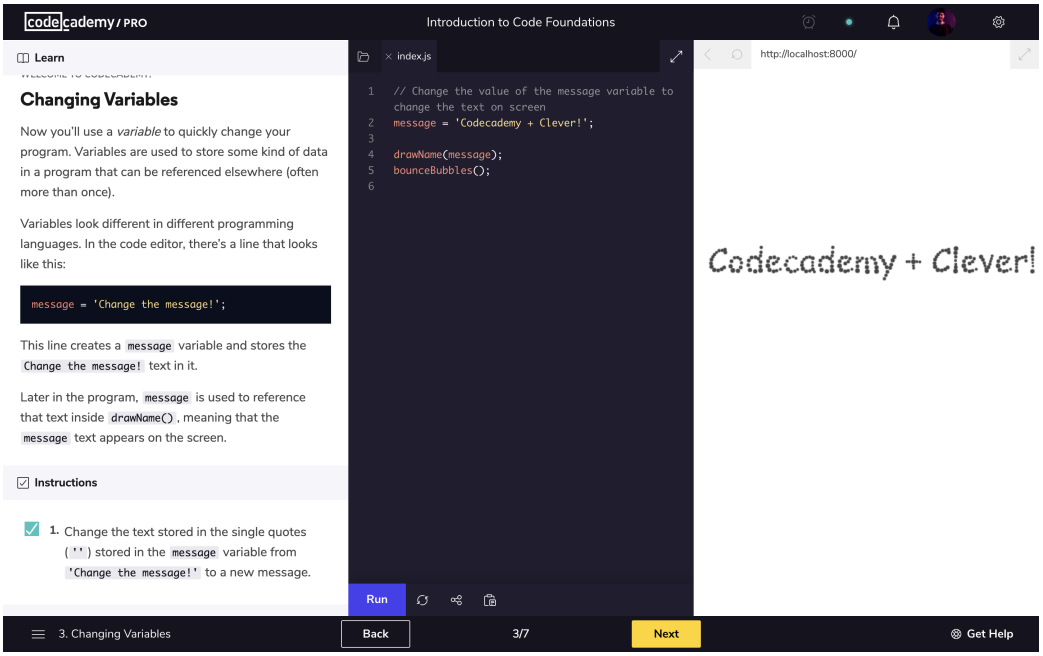
Jedním ze způsobů, jak se seznámit s programováním v Pythonu, je prostřednictvím populární platformy Codecademy[15]. Tato platforma nabízí interaktivní kurzy, které umožňují studentům procvičovat své dovednosti přímo v prohlížeči. Díky velkému množství materiálů obsahujících praktické cvičení si studenti postupně osvojují základní principy programování v Pythonu. Uživatelské rozhraní, zobrazené na obrázku 2.8, nabízí interaktivní editor kódu, který umožňuje okamžitou zpětnou vazbu, čímž studentům umožňuje rychle pochopit a korigovat své chyby. Codecademy je ideální pro začátečníky, kteří se chtějí seznámit s jazykem Python a získat pevný základ v programování.

Codecademy dále nabízí Python Cheatsheets[16], což jsou velmi stručné a přehledné příručky, ve kterých jsou jednotlivá témata Pythonu popsána formou tipů a triků. Tyto cheatsheety jsou skvělým doplňkem k interaktivním kurzům a poskytují studentům rychlý přehled o klíčových konceptech a syntaxi Pythonu. Kromě toho platforma nabízí také články k jednotlivým oblastem, které studentům poskytují další informace a hlubší porozumění.

2.6.2 LeetCode

LeetCode[17] představuje pro studenty, kteří již ovládají základy programování v Pythonu, vysoce ceněnou platformu pro získání pokročilých dovedností. Specializuje se na výzvy a soutěže, které vyzkoušejí dovednosti uživatelů při řešení různých algoritmů a problémů. Uživatelé se mohou setkat s širokou škálou úloh, které pokrývají oblasti jako algoritmické strategie, efektivní manipulaci s datovými strukturami a optimalizaci kódu. Tato rozmanitost a náročnost úloh poskytuje uživatelům skvělou příležitost k rozvoji a zdokonalení svých programátorských dovedností v Pythonu.

[13] https://bandit.readthedocs.io/
[14] https://flake8.pycqa.org/en/latest/
[15] https://www.codecademy.com/catalog/language/python
[16] https://www.codecademy.com/resources/cheatsheets/all
[17] https://leetcode.com/



Obrázek 2.8: Snímek obrazovky z interaktivního prostředí Python kurzu v Codecademy.

LeetCode je oblíbenou platformou mezi profesionálními programátory, studenty a také při přípravě na technické pracovní pohovory. Jeho prostředí umožňuje uživatelům procvičovat algoritmické myšlení a hledat efektivní řešení pro různé programátorské úlohy. Díky komplexní zpětné vazbě a diskusním fórům mohou uživatelé sdílet své přístupy a zkušenosti s ostatními, což přispívá k interaktivnímu a komunitnímu prostředí.

Vzhledem k tomu, že LeetCode nabízí úlohy na různých úrovních obtížnosti, od začátečníků po pokročilé uživatele, je vhodným nástrojem pro neustálé zdokonalování a růst v oblasti programování v Pythonu. Díky praktickému využití, které nabízí, umožňuje uživatelům aplikovat své znalosti a dovednosti v reálných situacích, což je klíčové pro úspěch v oblasti softwarového inženýrství.

2.6.3 Python tutor

V rámci bakalářské práce [12] byl vyvinut systém, který se již několik let využívá v kurzu Skriptovací jazyky. Jeho hlavním cílem je poskytnout studentům zpětnou vazbu na jejich projekty.

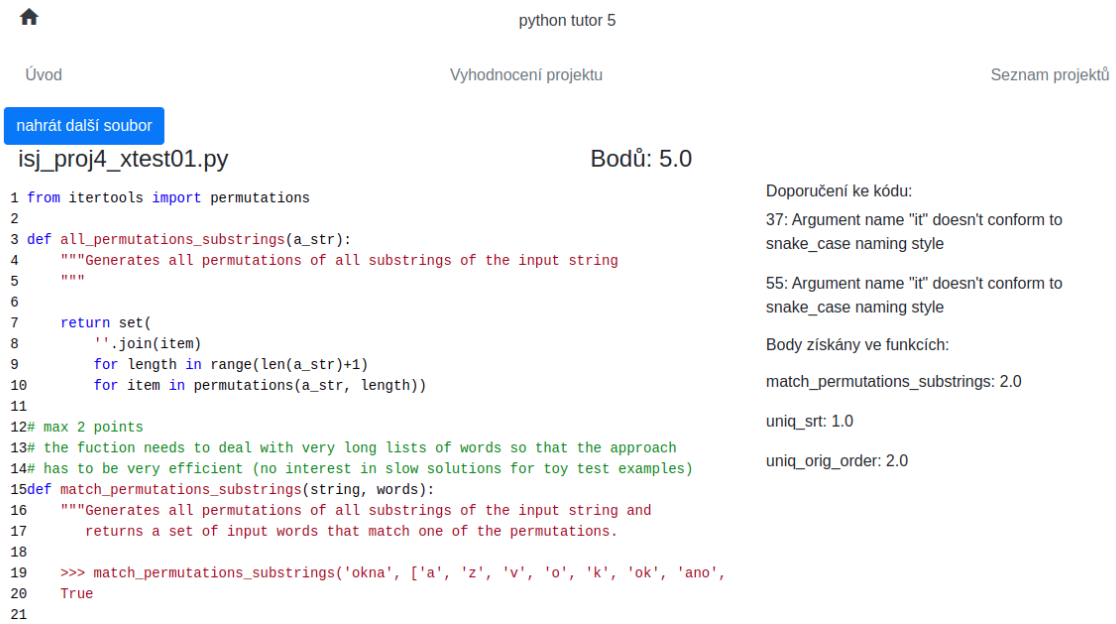
Poskytování zpětné vazby

Systém nabízí dva typy zpětné vazby. První typ se zaměřuje na kvalitu kódu, která je pro každý projekt hodnocena stejným způsobem. Druhý typ zpětné vazby poskytuje doporučení specifická pro každý projekt na základě analýzy abstraktních syntaktických stromů, které jsou sestaveny z odevzdaných projektů studentů. Tato metoda analýzy kódu je sepsána v podsekci 2.5.1.

Uživatelské rozhraní

Webové rozhraní je velmi jednoduché a přehledné. Aplikace je rozdělena na část uživatelskou a administrační.

Webové rozhraní pro uživatelskou část obsahuje jednoduchý formulář pro nahrání souborů k hodnocení a stránku s přehledem projektů, kde je uvedena dostupnost projektů. Na obrázku 2.9 je zobrazena stránka s výsledkem hodnocení. V levé části je nahraný python skript a v pravé části je samotná zpětná vazba.



Obrázek 2.9: Snímek obrazovky uživatelského rozhraní stránky vyhodnocení projektu z aplikace vytvořené v bakalářské práci [12].

Pro přístup do administrativní části aplikace je nezbytné přihlášení. Administrativní část zahrnuje správu projektů a testů. Editace informací o projektu se provádí pomocí formulářů. Jednotlivé testovací případy jsou přímo implementované v testovacích skriptech, a to ke každému projektu zvlášť. Testy ke každému projektu jsou tak pevně zakódovány, což znesnadňuje jednoduché rozšíření báze projektů.

Kapitola 3

Data a dotrénování modelů

Úvodní část této kapitoly podrobně představuje proces sběru a přípravy dat pro trénink a validaci jazykových modelů, které se používají k poskytování zpětné vazby na studentské projekty. Tento proces zahrnuje rozdělení existujících projektů do trénovacích a validačních skupin, manuální selekci řešení a přípravu datových sad pro trénování modelů. Následuje diskuse ohledně dotrénování modelů, která zahrnuje jak placené modely na komerčních platformách, tak i hledání alternativ zdarma a využití klasifikačních modelů. Klíčovými aspekty dotrénování jsou evaluace výsledků, identifikace nedostatků a následné přijetí opatření k jejich odstranění. Tato systematická analýza a zlepšování výkonu modelů přispívají k poskytování kvalitní zpětné vazby na studentské projekty, což je klíčovým cílem celého procesu.

3.1 Zpracování dat

Pro trénink jazykových modelů byly využity projekty studentů, realizovaných v rámci kurzu ISJ v průběhu posledních tří let (2023, 2022, 2021). Na základě teorie uvedené v podsekci 2.4.1 byly projekty rozděleny do dvou skupin: trénovací a validační. Tento přístup umožňuje validovat a testovat výkon modelu na nezávislých datech a zabránit přetrénování modelu na trénovacích datech. Projektů je celkem 9 a typicky obsahují více funkcí, které se testují samostatně.

3.1.1 Trénovací datová sada

U každého projektu byla provedena manuální selekce řešení, reprezentujících různé typy implementací. Existuje celá řada řešení projektů, které sice dostaly plný počet bodů, ale implementace neodpovídala zadání. Typicky je žádoucí velmi efektivní řešení či použití konkrétní metody, která se pro dané zadání hodí. Takže i když projekt splní všechny testy a dostane tak plný počet bodů, stále může být poskytnuta zpětná vazba navádějící k vylepšení kódu v tomto směru. Tato řešení jsou rozdělena do skupin, které zahrnují správná řešení i neoptimální přístupy k danému problému. Každé skupině je přiřazena adekvátní zpětná vazba na základě nastudované teorie ze sekce 2.2, sloužící jako cílová informace pro dotrénování jazykových modelů. Ve zpětné vazbě se nevyskutuje doporučení, které lze získat pomocí statické analýzy kódu popsané v podsekci 2.5.4. Kód v trénovacích datech je také očištěn o komentáře a docstringy.

V ukázce 3.1 je jeden příklad z trénovací sady se zpětnou vazbou v požadovaném formátu pro fine-tuning popsaný v podsekci 2.4.1. Důležitý je také popis chování modelu, který značně ovlivňuje výslednou odpověď.

```
{"messages": [
    {
        "role": "system",
        "content": "Act as expert tutor, providing short and very specific
               feedback on code efficiency in Python"
    },
    {
        "role": "user",
        "content": "perms = all_permutations_substrings(string)\n
               return set(filter(lambda x: x in perms, words))"
    },
    {
        "role": "assistant",
        "content": "Use specific set operation for more efficient and
               readable solution. There is no need to use lambda or filter."
    }
]}
```

Ukázka kódu 3.1: Příklad z trénovací sady se zpětnou vazbou v požadovaném formátu pro fine-tuning modelu.

3.1.2 Validační datová sada

Validační datová sada slouží výhradně k ověření schopnosti natrénovaného modelu poskytovat zpětnou vazbu na základě nových vstupů. Tato sada je plně oddělena od trénovacích dat a zajišťuje nezávislé testování výkonu modelu.

3.2 Dotrénování modelů

Tato sekce detailně popisuje průběh a výsledky dotrénování jazykových modelů pro generování zpětné vazby na studentské projekty. Nejdříve jsou testovány komerční modely, poté je popsán proces hledání alternativ k těmto placeným variantám. V závěru je popsán průběh a výběr klasifikačního modelu.

3.2.1 Placené modely

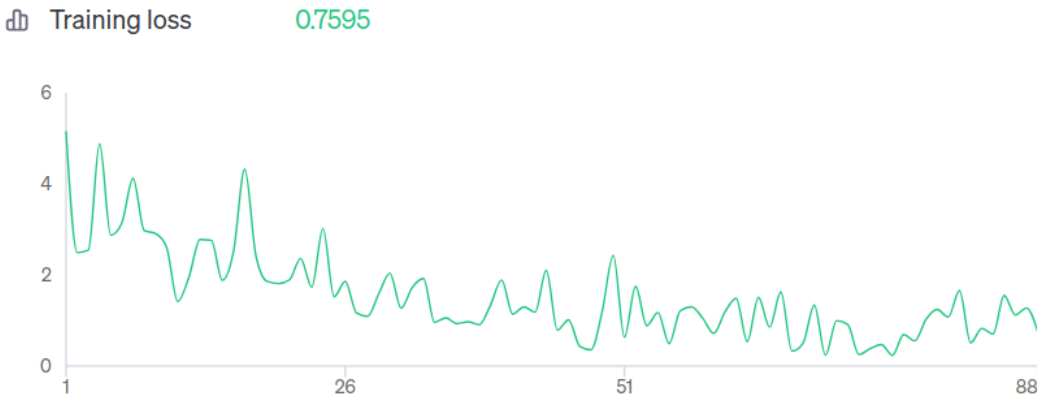
První fáze zkoumání se zaměřila na schopnosti analýzy kódu pomocí modelů na platformě *OpenAI*. Konkrétně byly testovány chatboti využívající modely GPT-3.5 a GPT-4. Při zadání obecného úkolu, kterým bylo analyzovat kód a navrhnout optimalizace pro zvýšení jeho efektivity, byly zaznamenány nedostatky, které se shodovaly s těmi popsanými v podsekci 2.4.1. Modely nedokázaly adekvátně interpretovat úkol bez přesně definovaného kontextu a specifikací. Také zpětná vazba a doporučení nebyly optimální. Tento nedostatek vedl k nekonzistentním odpovědím na identické vstupy. Každý projekt navíc vyžaduje jiný způsob analýzy, takže obecný prompt pro všechny nefunguje.

Dalším krokem bylo provedení dotrénování velkých jazykových modelů na komerčních platformách *OpenAI* a *Vertex AI*, které jsou popsány v podsekcích 2.4.1 a 2.4.2. Cílem této fáze bylo zhodnotit schopnosti aktuálních velkých modelů. V případě úspěchu by se následně hledala alternativa zdarma.

Pro fine-tuning byly vybrány konkrétní jazykové modely: **gpt-3.5-turbo-1106** a **text-bison@001**. Průběh a výsledky dotrénování obou modelů byly srovnatelné, proto jsou v následující podsekci popsány dohromady. Hlavním rozdílem mezi nimi byla cena a doba trvání dotrénování. Zatímco fine-tuning na platformě OpenAI trval několik desítek minut a stál do jednoho dolaru, na Vertex AI trval přes 2 hodiny a náklady se pohybovaly v nižších stovkách dolarů. Z tohoto důvodu byla platforma OpenAI preferována pro snazší ladění a rychlejší iterace dotrénování modelů. Příprava dat pro fine-tuning je detailně popsána v předchozí sekci 3.1.

Průběh dotrénování a výsledky

U obou modelů byly dosaženy dobré výsledky již při prvním pokusu. Během procesu dotrénování se postupně snižovala hodnota ztrátové funkce (training loss), která je vysvětlena v podsekci 2.4.1. Graf 3.1 zobrazuje její průběh během jednoho z trénování. Tento pokles signalizuje adaptaci modelů na trénovací data.



Obrázek 3.1: Graf znázorňující průběh ztrátové funkce (training loss) během trénování modelu na platformě OpenAI. Na horizontální ose jsou kroky trénování, na vertikální ose je hodnota ztrátové funkce.

Při následné evaluaci na validačních datech však byly zjištěny nedostatky. Bylo nezbytné výrazně omezit nežádoucí kreativitu modelů při generování výstupů. Modely začaly analyzovat kód i v oblastech, které nebyly pokryty trénovacími daty. Například navrhovaly přejmenování proměnných a úpravy kódu, které by implementaci spíše zhoršily. V některých případech chatbot dokonce navrhoval přímé řešení v podobě použí konkrétní funkce. Tento přístup nebyl obsažen v testovacích datech a v instrukcích modelu byl vysloveně zakázán. Cílem této zpětné vazby totiž není poskytnout konkrétní řešení, ale spíše studenta nasměrovat správným směrem.

Tato problematická kreativita jazykových modelů byla vyřešena snížením hodnoty parametru *temperature*, úpravou úvodních instrukcí (prompt) a větší generalizací zpětné vazby.

Snížení parametru *temperature* pomohlo eliminovat nekonzistentní a náhodně generované texty, což vedlo ke snížení tendence modelů produkovat nové, často nekorektní infor-

mace. Dalším klíčovým faktorem pro zlepšení výstupů modelů byla revize instrukcí s důrazem na přesnější formulaci očekávaných výstupů a stanovení limitů. Nakonec byla také upravena zpětná vazba v trénovacích datech, která byla dříve rozsáhlejší. Nyní se zpětná vazba zaměřuje pouze na zásadní problémy, což vede k přesnějším a účinnějším výsledkům.

Během několika málo iterací se podařilo dotrénovat modely, které generují kvalitní zpětnou vazbu na studentský kód. Tato zpětná vazba je přesnější než analýza kódu pomocí AST, která je využívána v aktuálně používaném řešení, jak je popsáno v podsekci 2.6.3. Jazykové modely jsou schopny detekovat situace, kdy je nějaká datová struktura v kódu použita nesprávně či neefektivně. Například mohou rozpoznat, že i přesto, že byla množina v analyzovaném kódu definována, nebyla využita dle očekávaní. Na rozdíl od toho analýza kódu pomocí AST pouze identifikuje přítomnost této struktury, ale není schopna posoudit, zda je použita správně.

3.2.2 Alternativy zdarma

Pro účely bezplatného dotrénování jazykových modelů byla vybrána platforma Hugging Face, která je detailněji popsána v posekci 2.4.3. Byla zkoumána celá škála modelů. Od rozsáhlých, jako je *OpenHermes-2.5-Mistral-7B*[1] (přes 7 miliardami parametrů), přes kompaktní *TinyLlama-1.1B-Chat-v1.0*[2], až po zcela malé modely.

Omezení a nedostatky

Byly vyzkoušeny desítky modelů, ale žádný z nich nedosáhl úrovně těch komerčních. Existuje několik faktorů, které mohou být odpovědné za tento nedostatek výkonu. Za prvé, tyto modely vyžadují více trénovacích dat, což může být problematické z důvodu ručního vytváření dat a potenciálních problémů s rozšiřitelností. Dále, menší modely trpí nedostatkem kontextu, což znamená, že nemohou efektivně porozumět úloze na základě pouhých pár ukázkových dat. Na rozdíl od toho, větší modely mají větší kontext, takže trénovací data spíše slouží k ukázce formátu a stylu odpovědi, která je požadována.

Celý proces výběru, stahování, trénování a vyhodnocování je časově i výpočetně velmi náročný[3]. Pro tyto účely byly využity školní server Sophie2 a servery virtuální organizace MetaCentrum[4].

3.2.3 Klasifikační modely

V rámci hledání alternativních přístupů k dotrénování velkých jazykových modelů byla zkoumána možnost využití klasifikačních modelů. Tyto modely se zaměřují na přiřazení textového vstupu jedné z předem definovaných tříd na základě naučených vzorů, což může být v tomto případě interpretováno jako generování konkrétní zpětné vazby. Díky zjednodušení zpětné vazby v trénovacích datech pro LLM (kvůli problematické kreativitě modelů, jak je popsáno v podsekci 3.2.1), je nyní možné tuto zpětnou vazbu rozdělit do klasifikačních tříd.

[1] https://huggingface.co/teknium/OpenHermes-2.5-Mistral-7B

[2] https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0

[3] *Hledání alternativ výrazně zasahovalo do plánovaného časového rámce a způsobilo značné zpoždění. Tato situace vyvolala nutnost hledat nové alternativní přístupy v podobě možnosti využití klasifikačních modelů.*

[4] https://metavo.metacentrum.cz/

Klasifikační modely, na rozdíl od modelů pro generování textu, jsou často mnohem menší a mohou být efektivně dotrénovány na specifické úkoly pomocí metod jako je SetFit, která je popsána v podsekci 2.4.3.

Evaluace a výběr modelu

Před výběrem konkrétního klasifikačního modelu proběhla evaluace několika kandidátů na dvou projektech obsahující čtyři samostatné funkce k analýze. Vyhodnocení výsledků je prezentováno v tabulce 3.2. Je důležité zdůraznit, že modely, které dosáhly výsledku nad 93% by byly 100%, protože do evaluačních dat byly přidány i velmi okrajové případy, které se nevyskytují v reálných projektech studentů. Cílem bylo zjistit, jak jednotlivé modely reagují na nestandardní situace a zvládají zpracovat atypické vstupy.

model	size	Project 8			Project 4			avg
		1 label	3 labels	6 labels	1 label	1 label	1 label	(1 label)
sentence-transformers/all-mpnet-base-v2	438 MB	97%	94%	100%	100%	94%	100%	97%
sentence-transformers/distiluse-base-multilingual-cased-v2	539 MB	97%	90%	100%	100%	94%	96%	96%
intfloat/e5-large-v2	1.34 GB	97%	87%	100%	100%	94%	100%	96%
sentence-transformers/paraphrase-mpnet-base-v2	438 MB	94%	94%	93%	94%	94%	100%	95%
sentence-transformers/all-MiniLM-L6-v2	90.9 MB	94%	87%	100%	100%	88%	100%	95%
flax-sentence-embeddings/st-codesearch-distilroberta-base	329 MB	94%	84%	87%	100%	94%	100%	93%
intfloat/multilingual-e5-large	2.24 GB	94%	87%	93%	100%	94%	92%	93%
intfloat/e5-small-v2	133 MB	94%	90%	93%	100%	88%	92%	93%
sentence-transformers/all-MiniLM-L12-v2	134 MB	92%	87%	87%	100%	94%	100%	93%
krlvi/sentence-msmarco-bert-base-dot-v5-nlpl-code_search	438 MB	92%	87%	93%	94%	81%	100%	91%
AISE-TUDelft/python-usage-classifier	438 MB	75%	90%	100%	100%	88%	92%	91%
AISE-TUDelft/python-summary-classifier	438 MB	69%	84%	93%	94%	81%	85%	84%

Obrázek 3.2: Srovnání úspěšnosti různých klasifikačních modelů.

Mezi vybranými modely se jako nejlepší ukázal obecný model **all-mpnet-base-v2**[5] (přes 100 milionů parametrů). Tento model, začleněný do rodiny sentence-transformers, byl trénován na obrovském množství textových dat a je schopen vektorizovat věty do 768-dimenzionálního prostoru. Tímto způsobem zachycuje sémantické informace obsažené v textu, což umožňuje jeho využití pro informační vyhledávání, shlukování nebo měření podobnosti vět.

Dotrénování klasifikačního modelu

Pro realizaci dotrénování klasifikačního modelu byl použit Python skript, který využívá knihovnu *datasets* pro práci s daty a knihovny *setfit* a *sentence_transformers* pro trénování modelů. Průběh trénování spočíval v načtení trénovacího a validačního datasetu, inicializaci modelu a následném trénování s využitím optimalizační funkce Cosine Similarity Loss[6] a nastavením počtu iterací na 20. Tato volba parametrů byla motivována potřebou dosažení dostatečného přesnosti modelu a minimalizací časové náročnosti procesu. Na závěr trénování byly vyhodnoceny výsledné metriky, které poskytují informaci o úspěšnosti modelu na validačním datasetu.

[5] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[6] https://github.com/UKPLab/sentence-transformers/blob/master/sentence_transformers/losses/CosineSimilarityLoss.py

Kapitola 4

Návrh

V této kapitole je představen návrh systému pro rozšiřování znalostí programování v jazyce Python. Nejprve jsou analyzovány a specifikovány požadavky na systém, následně je prozkoumáno existující řešení. V další části jsou navrženy a popsány prvky uživatelského rozhraní a použité webové technologie. Na závěr je představena architektura, typ úložiště, metody zabezpečení a strategie testování.

4.1 Analýza a specifikace požadavků

Hlavním cílem této práce je vytvořit webovou aplikaci, která poskytne studentům zpětnou vazbu na jejich projekty. Tento nástroj by měl umožnit snadné nahrávání projektů, automatizované testování a komplexní a především užitečnou zpětnou vazbu. Tato zpětná vazba bude zahrnovat výsledky testování, konkrétní doporučení na základě AI modelu, výsledky statické analýzy kódu a odkazy na externí zdroje, jako jsou kurzy a články. Tímto způsobem systém poskytne užitečné doporučení a zdroje na jednom místě a podpoří tak studenty v jejich učení a rozvoji dovedností v oblasti skriptovacích jazyků.

4.1.1 Diagram případů užití

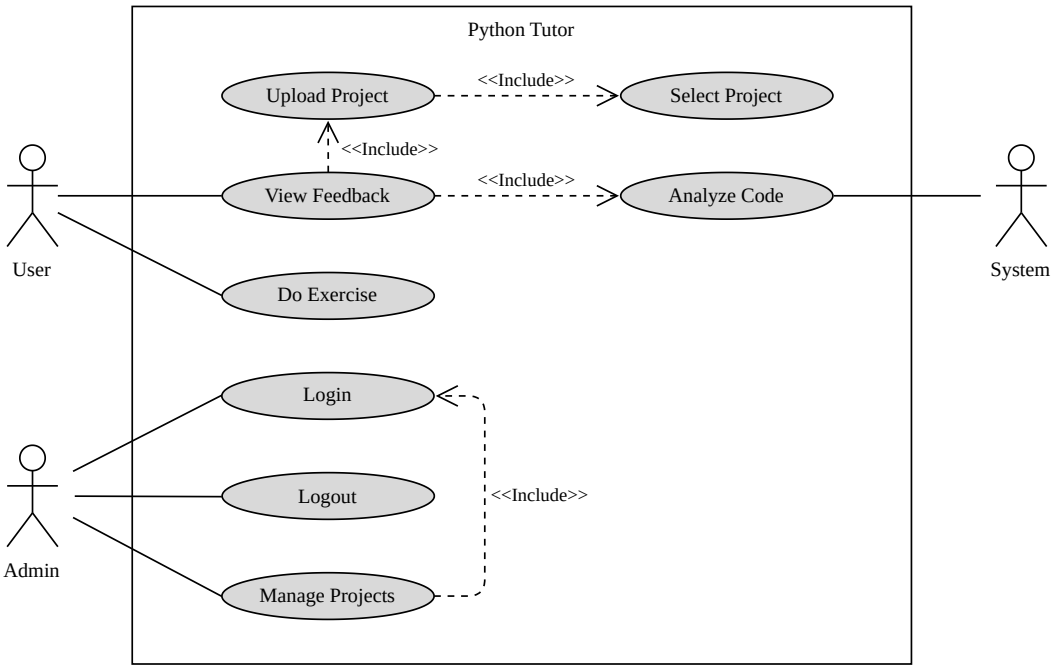
Diagram případů užití na obrázku 4.1 pro navrhovaný systém obsahuje tři hlavní aktéry a jejich interakce:

- **Student:** Má možnost nahrávat projekty, získávat zpětnou vazbu a vykonávat cvičení k jednotlivým tématům programování v Pythonu.
- **Administrátor:** Po autentizaci do systému má pravomoc spravovat projekty a systém.
- **Systém:** Systém je zodpovědný za vyhodnocování projektů a poskytování zpětné vazby studentům.

4.1.2 Analýza existujícího systému

Současné řešení systému, které je popsáno v podsekci 2.6.3, má několik nedostatků, které byly identifikovány na základě zpětné vazby od studentů[1]. Tyto připomínky byly peč-

[1]Studentská zpětná vazba byla získána především ze studentského Discord serveru.



Obrázek 4.1: Diagram případů užití navrhovaného systému.

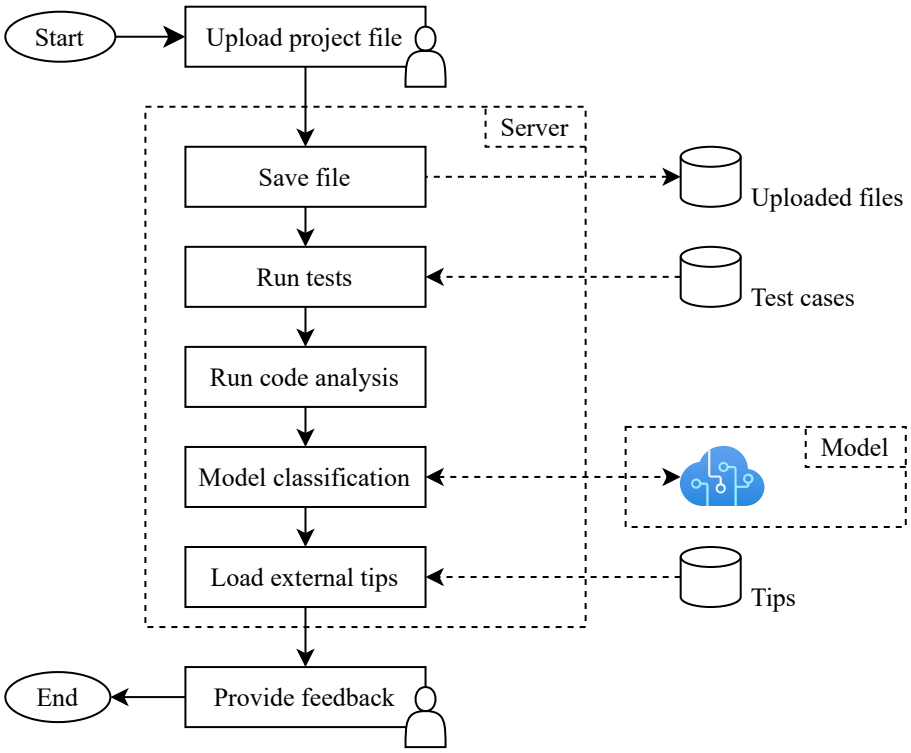
livě zhodnoceny a zohledněny při návrhu nového systému. Zde je uveden výčet některých z těchto nedostatků:

- *aplikace je často nedostupná,*
- *zbytečné doporučení týkající se části kódu, které student neimplementoval,*
- *výhrady k designu a rozhraní webové aplikace,*
- *povedlo se získat hodnotící testy když systém vracel chybu,*
- *chybí informace, že testy se shodují/neshodují s těmi, které budou tvořit bodové hodnocení projektu,*
- *aplikace je pomalá (vyhodnocení projektů trvá v závislosti na projektu 4 a více sekund),*
- *výsledky testů jen v podobě bodového zisku, žádné informace které testy selhaly a proč,*
- *chyba upozorňující, že pro tento soubor neexistují testy upozorňuje na špatné pojmenování odevzdaného souboru,*
- *stránka se zpětnou vazbou není přehledná.*

Jak bylo naznačeno v předchozí kapitole v podsekci 3.2.1, systém má určitý nedostatek ohledně detekce a použití konkrétních datových struktur v projektech. Analýza kódu je založena na AST, což omezuje schopnost systému identifikovat pouze výskyty hledaných prvků v kódu, aniž by poskytovala dostatečné informace o jejich kontextuálním využití. Tento nedostatek může vést k poskytnutí nepřesné zpětné vazby, neboť nebere v úvahu správnost použití identifikovaných prvků.

4.1.3 Proces evaluace projektů

Na obrázku 4.2 je diagram přehledně znázorňující pořadí funkčních požadavků potřebných k evaluaci projektů a jejich práci se zdroji.



Obrázek 4.2: Proces evaluace studentských projektů systémem Python Tutor.

Zpětná vazba

V souladu s teorií o poskytování zpětné vazby při programování, popsané v sekci 2.2, by měl navrhovaný systém umožňovat studentům odevzdávat své projekty a získávat podrobnou zpětnou vazbu. Klíčové vlastnosti aplikace zahrnují:

- **Automatické hodnocení s podrobnou zpětnou vazbou:** Systém by měl automaticky analyzovat odevzdaný kód a poskytnout studentům podrobnou zpětnou vazbu. Tato zpětná vazba by měla identifikovat konkrétní nedostatky v kódu a poukázat na neefektivnost řešení.
- **Možnost opakovaného odevzdání:** Studenti by měli mít možnost opakovaně odevzdávat své projekty a získávat opakovanou zpětnou vazbu. Tímto způsobem by mohli postupně vylepšovat svá řešení na základě získané zpětné vazby a učit se z vlastních chyb.
- **Rozšířená zpětná vazba:** Namísto pouhé binární zpětné vazby by systém měl poskytovat rozšířenou zpětnou vazbu, která se vztahuje ke konkrétním částem kódu. To studentům umožní lépe pochopit své chyby a nedostatky.

4.2 Uživatelské rozhraní

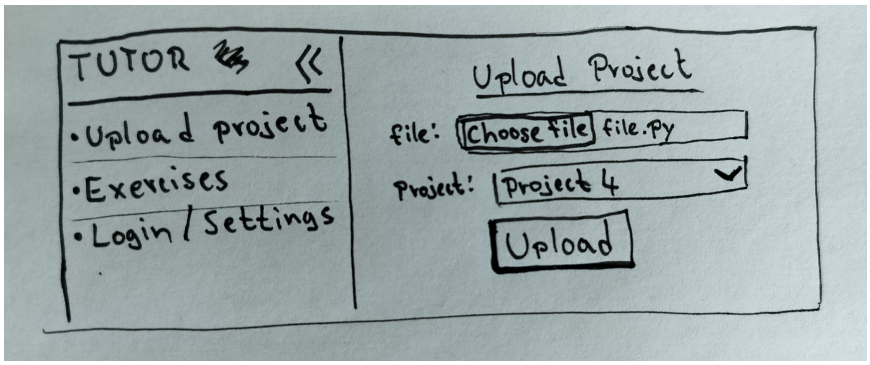
Grafické uživatelské rozhraní webové aplikace je navrženo s důrazem na jednoduchost, intuitivnost a přehlednost. Stránka je primárně určena pro použití na stolních počítačích, ale je také plně responzivní, což umožňuje použití na mobilních zařízeních a tabletech. Jako primární jazyk systému je zvolena angličtina, která je běžně používaným jazykem v oblasti informačních technologií. Okrajově také proto, aby byla zachována jednotnost a srozumitelnost pro potenciální uživatele nemluvící česky.

V této sekci jsou navrženy a především načrtnuty hlavní prvky uživatelského rozhraní včetně navigačního menu, rozvržení stránky s vyhodnocením projektů či administrační nastavení systému.

4.2.1 Nahrání projektu a menu

Na této stránce je k dispozici jednoduchý formulář pro nahrání souboru s vypracovaným řešením a výběr z dostupných projektů, na který chce student poskytnout zpětnou vazbu.

Na obrázku 4.3 je spolu s formulářem pro odevzdávání navržen i postranní panel obsahující menu s jednotlivými stránkami aplikace, jako je odevzdání, sekce se cvičeními nebo přihlášení do administrace. Uživatelé mají možnost skrýt toto menu a tím zvětšit prostor pro zobrazení obsahu.

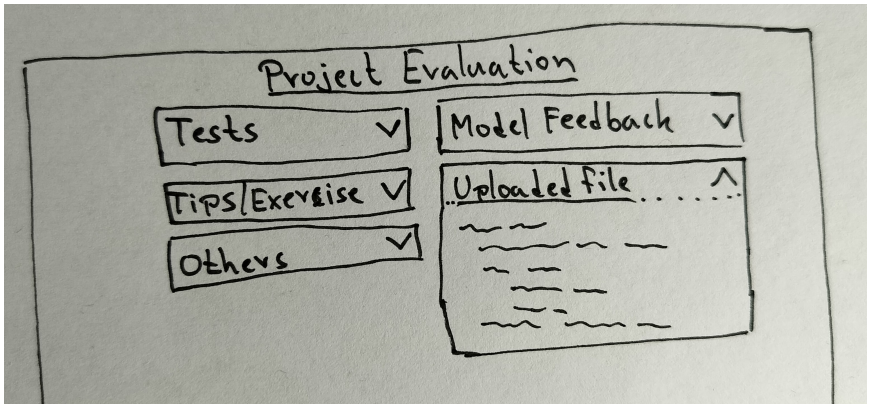


Obrázek 4.3: Návrh postranního panelu a stránky pro nahrání projektů.

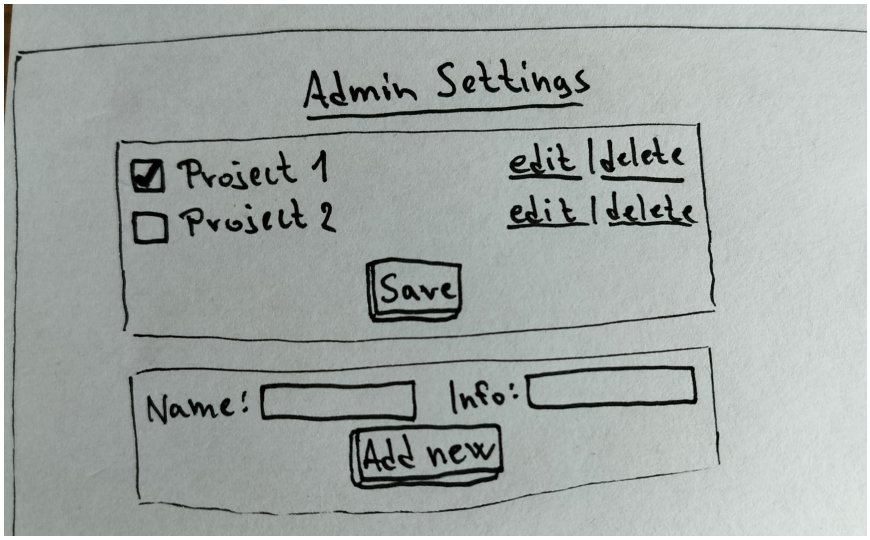
4.2.2 Zpětná vazba a administrace

Zpětná vazba se skládá z několika segmentů, které jsou popsány v podsekci 4.3.6. Na obrázku 4.4 je znázorněno, že student má možnost interaktivně rozkliknout jednotlivé segmenty a prozkoumávat tak části zpětné vazby podrobněji. Design stránky je navržen s ohledem na uživatelskou přívětivost. Tato optimalizace byla provedena v reakci na výhrady uživatelů ohledně nedostatečné přehlednosti obdobné stránky v existujícím řešení, jak je dokumentováno v podsekci 4.1.2.

Na obrázku 4.5 je zobrazeno uživatelské rozhraní pro administrátora aplikace, které obsahuje několik jednoduchých formulářů umožňujících správu projektů. Podobné rozhraní bude k dispozici také pro správu testů a modelů. Administrativní část systému představuje klíčový prvek z hlediska rozšiřitelnosti a správy aplikace.



Obrázek 4.4: Návrh stránky pro vyhodnocení projektů.



Obrázek 4.5: Návrh stránky pro nastavení administrátora.

4.3 Návrh webové aplikace

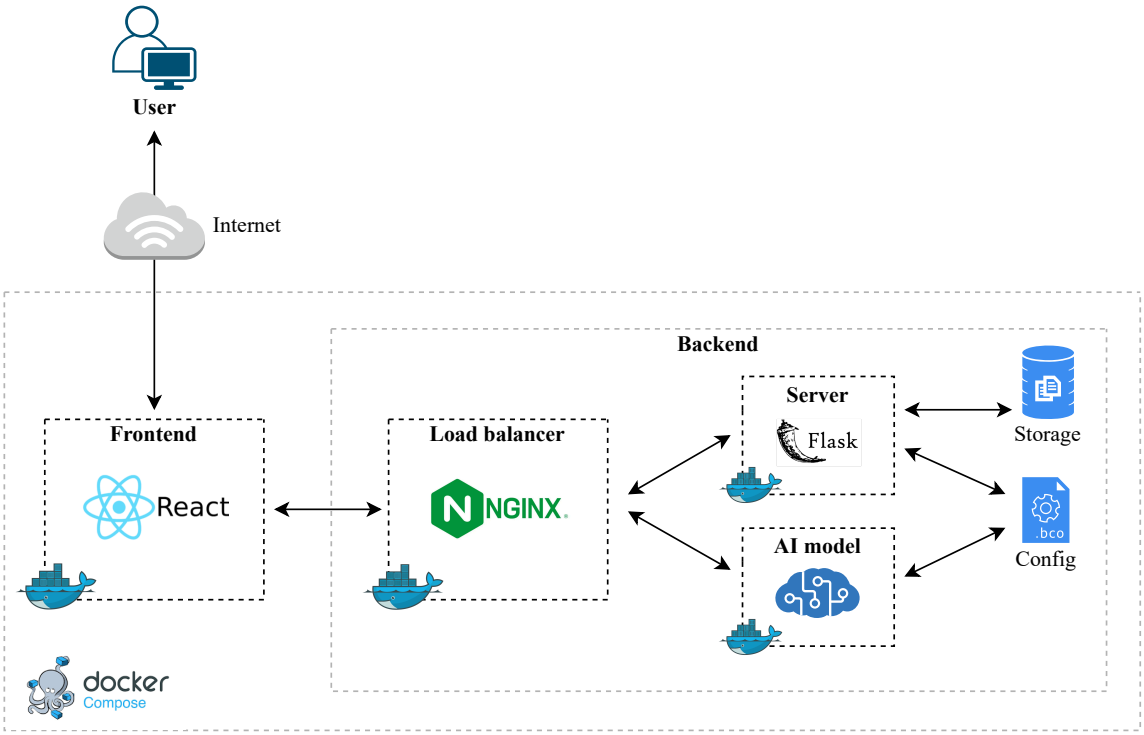
Tato sekce popisuje webovou architekturu, použité technologie, úložiště dat, metody zabezpečení a strategii testování navržené aplikace.

4.3.1 Webové technologie

Pro implementaci webové aplikace jsou zvoleny osvědčené technologie, které by měly zajistit efektivní a bezproblémový provoz celého systému. Architektura aplikace je znázorněna na diagramu 4.6. Během vývoje aplikace bude použit verzovací systém Git pro správu zdrojových kódů.

Pro hlavní část backendu je zvolen **Flask**[2] server, což je lehký a flexibilní framework napsaný v jazyce Python. Tato volba je motivována jeho jednoduchostí a schopností rychle vytvářet webové aplikace. Pro zajištění spolehlivosti a škálovatelnosti backendu je také im-

[2] https://flask.palletsprojects.com



Obrázek 4.6: Architektura navrhovaného systému Python Tutor.

plementován **Gunicorn**[3], WSGI (Web Server Gateway Interface) pro Flask, který umožňuje obsluhovat více žádostí současně.

Pro vyrovnávání zátěže a zajištění dostupnosti v době zvýšeného provozu při uzávěrce projektů je do architektury integrován **Nginx**[4] jako load balancer. Tato technologie umožňuje rovnoměrně rozdělovat zátěž mezi více serverů, čímž se zvyšuje celková dostupnost a výkon systému. Nginx také poskytuje možnost rychle reagovat na dynamické změny zátěže, což je klíčové pro stabilní provoz aplikace.

Pro izolaci a snadnou distribuci aplikace je použit **Docker Compose**[5], který umožňuje vytvářet a spravovat kontejnery. Tato technologie poskytuje bezpečné spouštění potenciálně nebezpečných studentských kódů a umožňuje jednoduché škálování backendu. Díky integraci s Nginx je možné snadno přesměrovávat žádosti na různé instance backendových serverů, což usnadňuje správu a optimalizaci výkonu.

Pro implementaci frontendu je zvolen **ReactJS**[6], což je populární knihovna pro tvorbu uživatelských rozhraní. Tato volba je motivována jednoduchostí použití a rozsáhlou komunitou, což usnadňuje rychlý vývoj a udržování frontendové části aplikace. ReactJS navíc nabízí efektivní správu stavu aplikace a umožňuje jednoduchou integraci s backendem.

[3] https://gunicorn.org
[4] https://nginx.com
[5] https://docs.docker.com/compose
[6] https://react.dev

4.3.2 Úložiště

Namísto využití klasické databáze pro uchování informací v aplikaci, jsou používány konfigurační soubory obsahující klíčová data a nastavení nezbytná pro provoz systému. Tento přístup nabízí snadnou modifikaci a minimalizuje složitost celé aplikace.

Pro ukládání souborů bude využito běžného souborového systému na serveru, který poskytuje prostor pro ukládání a správu uživatelských projektů. Oba typy úložiště jsou znázorněny na diagramu na obrázku 4.6.

4.3.3 Bezpečnost

Tato podsekce se zaměřuje na klíčové aspekty zabezpečení webové aplikace, zejména v kontextu spouštění neznámých a potenciálně nebezpečných skriptů v jazyce Python.

Prvním krokem k zajištění bezpečnosti aplikace je kontejnerizace, která odděluje spouštění nebezpečného kódu od zbytku aplikace. Tato technologie umožňuje izolaci běžících procesů a poskytuje bezpečný prostředek pro provádění kritických operací. Implementace mechanismů monitorování, které jsou podrobněji popsány v následující podsekci 4.3.4, umožňuje identifikovat podezřelé aktivity a potenciální bezpečnostní incidenty.

4.3.4 Logování

Při návrhu aplikace je plánováno použití systému logování s cílem zachytit důležité události, stavové informace a případné chyby. Tento přístup umožní efektivnější diagnostiku, ladění aplikace a monitorování výkonu. Záznamy z logování budou zásadní pro posouzení pokroku studentů v rámci opakovaného odevzdávání projektů a jejich postupného zlepšování se v čase.

Kromě ukládání logů do souborů na serveru je aplikace navržena s funkcionalitou pro odesílání upozornění administrátorovi prostřednictvím e-mailu v případě detekce kritických chyb, což zajistí okamžitou reakci a minimalizuje potenciální dopad.

4.3.5 Testování

Testování aplikace zahrnuje ostré testování prováděné v průběhu semestru v rámci kurzu ISJ, stejně jako testování uživatelských scénářů a funkcionalit. Součástí testování budou také unit testy, které již při vývoji ověří správnost funkcionality jednotlivých částí aplikace.

Během ostrého testování budou studenti mít příležitost aplikaci aktivně využívat a porovnávat ji s existujícím řešením. Tato fáze testování umožní získat podrobnou zpětnou vazbu od uživatelů ohledně funkčnosti, uživatelského rozhraní a výkonu aplikace. Získaná zpětná vazba bude důkladně analyzována a případné nedostatky či návrhy na vylepšení budou integrovány do aplikace v průběhu semestru.

Na závěr bude provedeno vyhodnocení celkového přínosu aplikace prostřednictvím dotazníku, který umožní studentům poskytnout svůj názor a hodnocení aplikace z hlediska uživatelské zkušenosti, použitelnosti a efektivity. Tato kombinace ostrého testování, získávání zpětné vazby od uživatelů a systematického vyhodnocení pomůže zajistit efektivní fungování aplikace v reálném provozu.

4.3.6 Analýza kódu

V reakci na připomínky studentů ohledně statické analýzy kódu, jak je uvedeno v podsekci 4.1.2, bylo zjištěno, že doporučení poskytovaná statickou analýzou se často vztahují

k částem skriptu, které studenti neimplementovali. Tento typ zpětné vazby tedy není příliš užitečný. Původní plán zahrnoval možnost přidání určitých doporučení na blacklist. Tento seznam by však byl pro každý projekt jiný, což je nepraktické. Místo toho bude vytvořeno upozornění, které zdůrazňuje, že výsledky statické analýzy a její doporučení mohou být relevantní pouze pro určité typy řešení a slouží spíše jako informativní zdroj. Studenti by měli tyto informace brát s rezervou. Během implementace budou provedeny testy nástrojů pro statickou analýzu, které jsou představeny v podsekci 2.5.4. Na základě těchto testů bude vybrán optimální nástroj nebo jejich kombinace, která nejlépe splňuje požadavky na kvalitu analýzy.

Dalším krokem v návrhu je implementace generování konkrétních doporučení prostřednictvím natrénovaného klasifikačního modelu přímo pro každý projekt. Tento přístup umožní poskytnout studentům cílenější a relevantnější zpětnou vazbu, která lépe odpovídá potřebám jednotlivých projektů.

4.3.7 Příklady na procvičování

Cílem této části je rozšířit kurzy ISJ o praktické příklady, které budou aplikovat stejné principy jako projekty, ale na konkrétní praktické situace. Tímto způsobem se studentům lépe přiblíží praktické využití a zdůvodní se, proč je v zadání například požadováno optimalizované a efektivní řešení, i když by projekt bylo možné implementovat méně efektivním způsobem. Je důležité zdůraznit, že i neoptimální řešení projektu projde testy a obdrží plný počet bodů. Cílem těchto cvičení je tedy zdůraznit, proč je efektivní řešení tak důležité, a ukázat studentům, jak mohou tyto principy aplikovat v reálných situacích.

V případech, kdy jsou projekty zaměřeny na základní principy skriptování, není vhodné vytvářet nové příklady na procvičování, protože jsou dostupné kvalitní zdroje online. Studenti tak budou odkázáni na již existující materiály, jako jsou interaktivní cvičení a přehledy, které pokrývají danou oblast. Tyto možnosti jsou zpracovány v sekci 2.6.

Kapitola 5

Implementace

V rámci této kapitoly je představena realizace systému navrženého v předchozí kapitole 4. Nejdříve jsou popsány detaily konfigurace a nasazení aplikace, struktura projektu a kroky nutné k instalaci a spuštění. Další část se věnuje API rozhraní serveru a způsobu, jakým jsou logovány události pro správu a monitorování chodu aplikace. Dále jsou představeny technické detaily frontendové části aplikace. Kapitola se dále zaměřuje na uživatelské rozhraní aplikace a jeho jednotlivé prvky. Důraz je kladen na přehlednost a intuitivnost uživatelského rozhraní s ohledem na návrh. V neposlední řadě je pozornost věnována sekci s příklady na procvičení a samotné administraci, která slouží pro správu projektů, testů a jazykových modelů. Každá sekce je doprovázena relevantními obrázky z výsledného řešení a schématy, které usnadňují porozumění implementovaným funkcionalit a procesů.

5.1 Konfigurace a nasazení aplikace

V této sekci je popsán způsob implementace webové aplikace Python Tutor. Nejprve je vysvětlena konfigurace pomocí Docker Compose. Dále je uvedena struktura projektu a postup nasazení na produkční server. V této sekci jsou dále vysvětleny drobné změny ve volbě technologií, které byly provedeny oproti návrhu architektury z podsekce 4.3.1. Poslední část ilustruje API rozhraní serveru a způsob, jakým jsou logovány události pro efektivní správu a monitorování chodu aplikace.

5.1.1 Docker Compose

Implementace pomocí Docker Compose je klíčovým prvkem nasazení aplikace, umožňující jednoduché a spolehlivé spouštění různých částí aplikace v kontejnerech. Konfigurace využívající Docker Compose obsahuje čtyři služby, z nichž každá má svůj vlastní `Dockerfile`. Tyto služby jsou nezbytné pro správnou funkčnost systému a zahrnují:

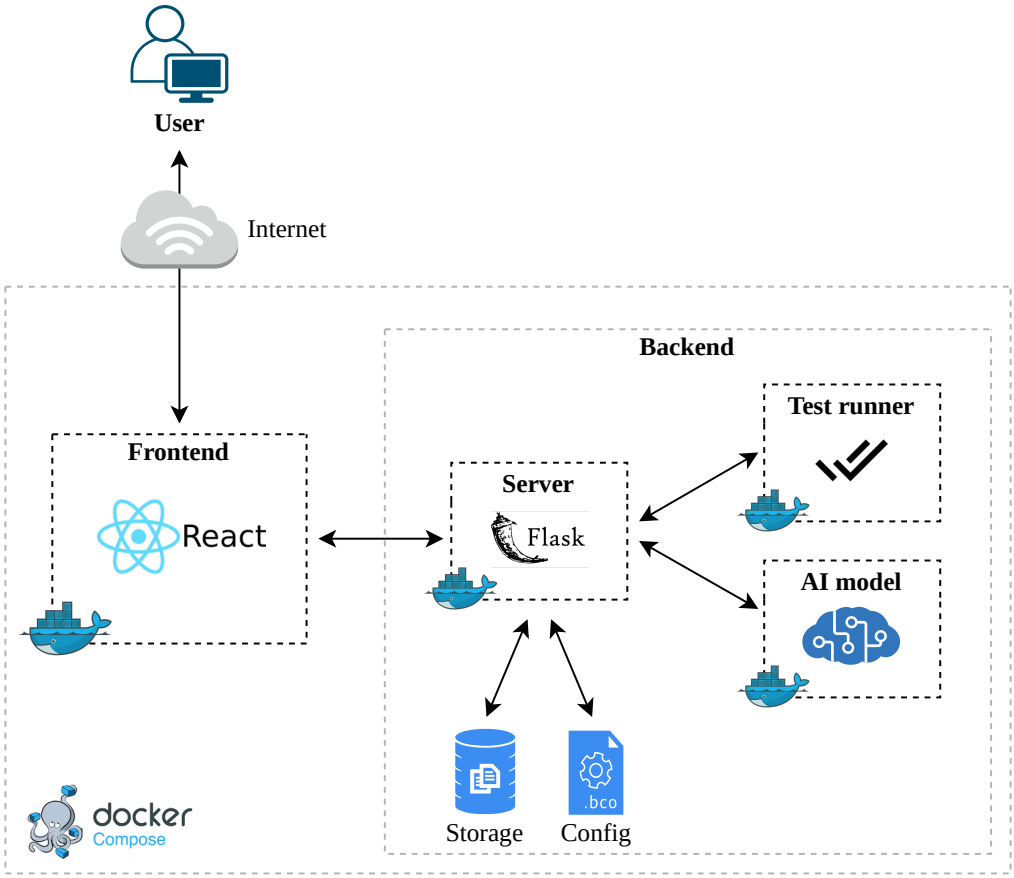
- **Flask server**: Tato služba spouští hlavní server aplikace. Konfigurace obsahuje mapování portu na hostitelský systém a umožňuje sdílení adresářů pro ukládání logů, nahraných projektů a přístup ke konfiguračním souborům.
- **Model server**: Služba zajišťující běh serverové části, která generuje zpětnou vazbu poskytovanou AI modely.
- **Test server**: Tento server zajišťuje izolované prostředí pro bezpečné testování nahraných skriptů.

- **React client**: Služba ReactJS klienta pro frontend aplikace. Tato služba závisí na službě Flask serveru a obsahuje konfiguraci pro mapování portu na hostitelský systém.

Výsledná architektura implementovaného systému Python Tutor je vyobrazena na diagramu 5.1. Servery Test a Model jsou přístupné pouze na lokální síti v rámci kontejnerů, zatímco kontejnery pro backend a frontend aplikace jsou dostupné i zvenčí.

Pro začátek byla spuštěna jedna instance serveru Flask, která během provozu prokázala konzistentní výkon, aniž by bylo nutné přidávat další instance. Na základě této stabilní provozní zkušenosti bylo rozhodnuto neimplementovat dynamické škálování. Nginx load balancer tedy není součástí výsledné architektury.

Oproti původnímu návrhu byl pro zvýšení bezpečnosti testování vytvořen nový Test server. Tento izolovaný kontejner má velmi omezené funkce a nedostává přístup ke konfiguračním souborům na hostitelském systému. Kontejner je určen k spouštění nahraných skriptů studentů, které mohou být potenciálně nebezpečné. Bezpečnostní prvky zahrnují omezené oprávnění, což pomáhá minimalizovat riziko neautorizovaného přístupu a bezpečnostních hrozeb. Nastavení volby `no-new-privileges` v `docker-compose.yml` brání kontejneru získat nová privilegia po spuštění.



Obrázek 5.1: Architektura implementovaného systému Python Tutor.

5.1.2 Struktura aplikace

V rámci struktury projektu jsou některé složky obsahující data, která nejsou vhodná pro verzování, explicitně vyloučeny z verzování pomocí konfigurace souboru `.gitignore`. Mezi tyto složky patří `./config`, `./logs` a `./uploads`. Projekt má následující hierarchii:

```
python-tutor
├── client
├── config
├── data
├── logs
├── server
│   ├── flask
│   ├── model
│   └── test
├── tests
└── uploads
```

Jednotlivé složky obsahují:

- `client` – Zdrojové kódy frontendové části aplikace v Reactu.
- `config` – Konfigurační soubory.
- `data` – Složka obsahující data a skripty pro trénování modelů.
- `logs` – Soubory logování.
- `server` – Serverová část aplikace.
  - `flask` – Hlavní Flask server.
  - `model` – Server obsluhující modely.
  - `test` – Server k testování projektů.
- `tests` – Složka obsahující skript pro spuštění testů.
- `uploads` – Složka obsahující nahrané soubory studentů.

5.1.3 Instalace a spuštění

Kompletní postup pro spuštění ve vývojovém prostředí, nasazení na produkčním serveru, vytvoření datasetu pro model a samotné dotrénování klasifikačního modelu je podrobně popsán v manuálu, který je k dispozici v příloze A. Aplikace běží na serveru *ivs.fit.vutbr.cz*, komunikuje přes specifické porty, které bylo třeba povolit ve firewallu. Díky konfiguraci pomocí Docker Compose je možné jednoduše spustit všechny služby a zajistit jejich vzájemnou komunikaci a správu prostřednictvím kontejnerů. Pro spuštění aplikace stačí v kořenovém adresáři provést příkaz `docker-compose up`.

5.1.4 API rozhraní

V této podsekci je popsáno REST API rozhraní serveru Flask, který představuje jediné rozhraní pro komunikaci s frontendem aplikace. Tento server je nejrozsáhlejší a obsahuje kompletní sadu funkcionalit. Flask server komunikuje se zbylými servery (Model a Test) v rámci lokální sítě. Tyto servery mají jednodušší API, které je implementováno obdobně.

Na Flask serveru jsou API endpointy definovány v modulu `api`. Při inicializaci Flask aplikace je povoleno Cross-Origin Resource Sharing (CORS), což umožňuje přístup klientům z různých domén k API. Soubor `routes.py` obsahuje definice jednotlivých endpointů a jejich přiřazení k odpovídajícím funkcím z ostatních modulů aplikace, které zpracovávají logiku jednotlivých API požadavků. Tento přístup zlepšuje modularitu, znovupoužitelnost a testovatelnost kódu. Každý modul přebírá zodpovědnost za specifické operace, což přináší vyšší úroveň abstrakce a usnadňuje správu a vývoj aplikace. Použití proměnné `API_VERSION` umožňuje jednoduchou správu a rozlišení různých verzí API.

5.1.5 Logování

Pro efektivní správu a monitorování chodu aplikace je ve všech flask serverech implementováno logování událostí. Logování je realizováno pomocí vestavěného modulu `logging` v jazyce Python. Cílem logování je zachytit důležité události, stavové informace a případné chyby, které se vyskytnou během provozu aplikace. Tato podsekce popisuje konfiguraci logování, formát záznamů a způsob, jakým jsou logy ukládány a udržovány.

Konfigurace

Konfigurace logů je implementována v souboru `logging_config.py`. Pro konfiguraci je využit modul `logging.config`, který umožňuje nastavení různých parametrů logovacího systému, jako jsou formáty záznamů, úroveň logování a způsob ukládání logů. Záznamy logů obsahují informace jako je datum a čas, úroveň logování, název modulu a zprávu, jak je ukázáno na příkladu 5.1.

```
2024-02-23 19:23:24 : INFO : 127.0.0.1 - "POST /api/v1/upload HTTP/1.1" 200
2024-02-23 19:23:24 : DEBUG : upload : File uploaded: isj_proj4.py
2024-02-23 19:23:24 : DEBUG : test : Running tests for proj4
2024-02-23 19:23:24 : DEBUG : app : Calling model for proj4
2024-02-23 19:23:25 : ERROR : model : Model classification failed
```

Ukázka kódu 5.1: Ukázka logování při nahrávání a testování projektu.

Udržování historie logů

Logy jsou ukládány do složky `/logs`. Pro držení historie logů a prevenci ztráty dat je použit `RotatingFileHandler`. Tento zapisovač umožňuje nastavení maximální velikosti souboru a maximálního počtu zálohovaných souborů. Pokud je překročena maximální velikost, je vytvořen nový soubor pro další záznamy. Pro archivaci logů (například po skončení každého semestru) je vytvořen skript `archive_logs.sh` v kořenovém adresáři, který provádí kompresi logovacích souborů a jejich uložení do archivního souboru. Soubor je pojmenován podle aktuálního roku a tato archivace umožňuje budoucí analýzy.

5.2 Frontend

Pro frontendovou část aplikace byl zvolen framework React. Základní stavební prvky aplikace tvoří komponenty a stránky. Komponenty jsou opakovaně použitelné části aplikace, které definují jednotlivé UI prvky a jejich logiku. Stránky představují specifické části aplikace, které shlukují související funkčnost a obsah. Struktura adresářového stromu frontendové části aplikace je následující:

```
client
├── public
├── src
│   ├── assets
│   ├── components
│   ├── pages
│   ├── App.js
│   ├── AuthContext.js
│   ├── PrivateRoute.js
│   └── ServerConfig.js
├── package.json
└── Dockerfile
```

Složka `/client` obsahuje klíčové prvky frontendu:

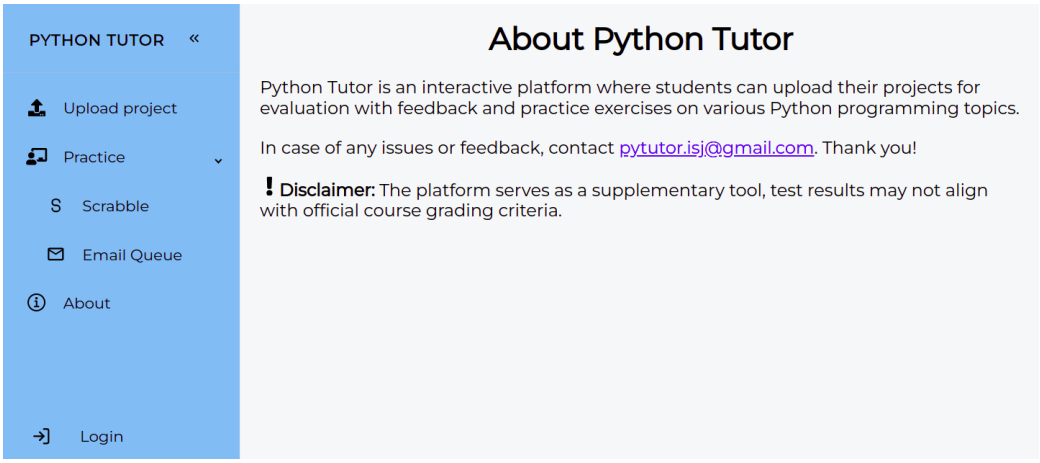
- `public` – Obsahuje veřejné soubory, jako je vstupní HTML soubor `index.html` a ikony.
- `src` – Jádro aplikace, které obsahuje veškerý zdrojový kód.
  - `assets` – Složka obsahující statické zdroje, jako jsou styly, obrázky a fonty.
  - `components` – Složka obsahující opakovaně použitelné komponenty.
  - `pages` – Složka obsahující jednotlivé stránky aplikace.
  - `App.js` – Soubor definující hlavní komponentu aplikace. Pro směrování a navigaci mezi jednotlivými stránkami využívá knihovnu `react-router-dom`.
  - `AuthContext.js a PrivateRoute.js` – Soubory definující mechanismy pro autentizaci a autorizaci v aplikaci.
  - `ServerConfig.js` – Soubor obsahující konfiguraci serveru pro komunikaci s backendem aplikace, včetně základních informací o API.
- `package.json` – Definuje metadata a závislosti aplikace.
- `Dockerfile` – Dockerfile definuje postup pro vytvoření Docker image. Je zvolen image Node.js ve verzi 16 s operačním systémem Alpine, což je minimalistická distribuce Linuxu, která umožňuje rychlejší spuštění kontejneru.

5.2.1 Uživatelské rozhraní

Uživatelské rozhraní je implementováno dle návrhu, který je představen v sekci 4.2. Na obrázku 5.2 je snímek obrazovky ze stránky *About*, která poskytuje uživatelům základní informace o Python Tutoru. V levé části se nachází postranní panel obsahující hlavní navigační menu. Tento boční panel je implementován jako komponenta, která poskytuje možnost

skrytí či rozbalení panelu, což umožňuje maximalizovat prostor pro obsah stránky podle potřeby uživatele. Nabídka menu se mění v závislosti na přihlášení uživatele. jednotlivé položky menu jsou reprezentovány ikonami a textem. To poskytuje uživatelům intuitivní navigaci a snadný přístup k různým funkcím a stránkám. Tyto ikony, používané po celé aplikaci, zajišťují snadnou identifikaci. Pro zachování konzistentního vizuálního stylu byly ikony vybrány z knihovny `react-icons`.

Ostatní části rozhraní aplikace jsou představeny ve zbytku této kapitoly, kde jsou podrobněji popsány jednotlivé funkcionality systému.



Obrázek 5.2: Uživatelské rozhraní stránky *About*.

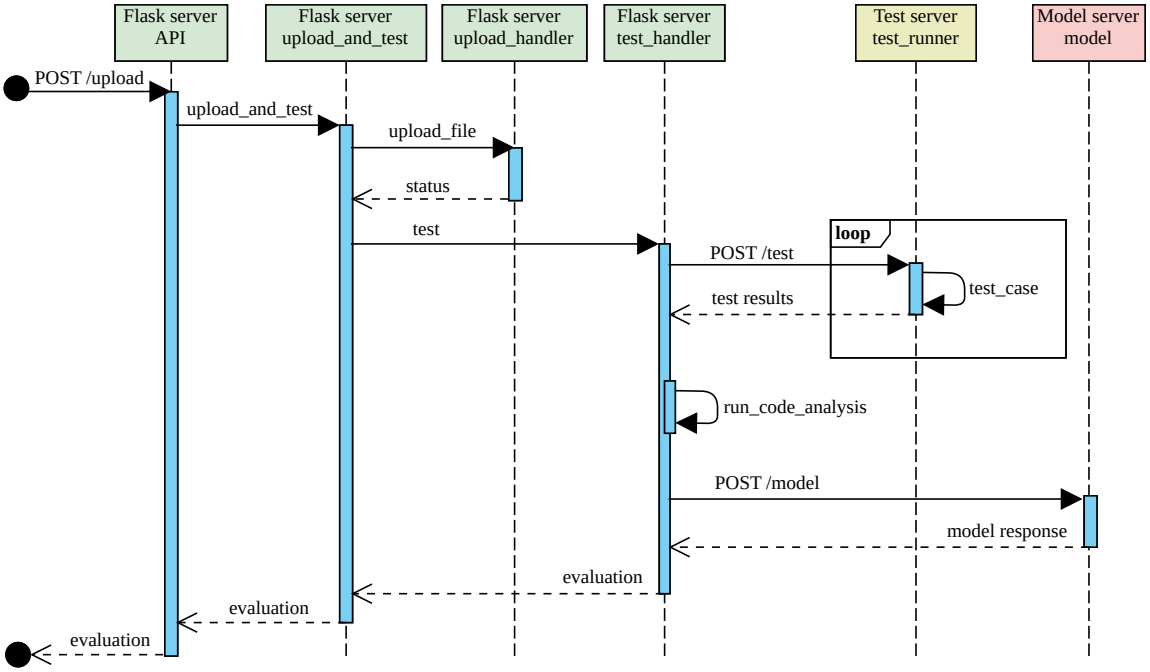
5.3 Poskytnutí zpětné vazby

Celý proces generování zpětné vazby na studenstké projekty vychází z návrhu popsaného v podsekci 4.1.3. Pro lepší porozumění fungování systému je vytvořen sekvenční diagram 5.3, který zjednodušeně vizualizuje interakce mezi jednotlivými servery a jejich moduly v čase. Evaluace projektu se skládá z výsledků automatických testů, hodnocení předtrénovaným modelem a statické analýzy kódu. V této sekci jsou popsány implementační detaily jednotlivých částí.

5.3.1 Nahrání projektu

V této podsekci je popsán proces nahrávání projektu do systému. Celý proces začíná akcí uživatele, který prostřednictvím webové stránky nahrává své řešení projektu. Systém provádí první kontroly již na úrovni frontendu a v případě zjištěných nedostatků je o nich uživatel informován.

Nahraný soubor je následně přenesen na server, kde jej zpracovává modul `uploads`. Tento modul se stará o další validaci a ukládání souboru do systému. Před samotným uložením je prověřena správnost formátu souboru a jeho maximální povolená velikost. Jméno souboru je standardizováno a v případě, že by došlo ke kolizi s existujícím souborem, je k němu připojen unikátní identifikátor. Tuto a další funkcionality zajišťují pomocné procedury implementované v souboru `uploads/upload_utils.py`. Soubory jsou ukládány na server do adresáře `/uploads/{ROK}/{PROJEKT}/`. Důležité parametry, jako je cesta k adre-



Obrázek 5.3: Sekvenční diagram zobrazující interakce mezi servery a moduly.

sáři pro ukládání souborů či maximální povolená velikost souboru, jsou uloženy v konfiguračním souboru `uploads_config.py`.

5.3.2 Testování projektů

Pro přípravu a vykonání testů slouží modul `tests`. Tento modul má za úkol správu testovacích procedur a komunikaci s testovacím serverem. Části nahraného projektu jsou podrobeny testování a následně je vyhodnocena úspěšnost provedených testů. Testovací data jsou uložena v konfiguračním souboru. Samotné spouštění testů probíhá na Test serveru, kterému je přes API zaslán projekt a příslušná testovací sada. Po provedení testů je výsledek zpětně vrácen. Tento výsledek obsahuje informace o úspěšnosti testů a případné komentáře k jejich průběhu. Tyto komentáře obsahují detaily o neúspěšných testech jako je identifikace funkcí, vstupní parametry a očekávaný výstup. Tento přístup je reakcí na připomínky uživatelů, kteří vyjadřovali nespokojenost s existujícím řešením, které poskytuje pouze konečný počet bodů bez dalších informací, jak je uvedeno v podsekci 4.1.3.

Některé projekty vyžadují specifické přístupy k testování, které nelze aplikovat obecně. Tyto projekty jsou identifikovány a testovány pomocí specializovaných částí kódu, které jsou navrženy s ohledem na jejich jedinečné charakteristiky. Například, projekty manipulující s objekty tříd, vyžadují testování, které zohledňuje specifika těchto objektů. Generické testování umožňuje spouštění testů na libovolné funkce s různými argumenty základních datových typů.

Testovací server obsahuje skript, který dynamicky importuje funkce nebo třídy ze souboru a spouští testy nad těmito importovanými funkcemi nebo třídami. Pro každou nahranou funkci či třídu se postupně prochází testovací sady a testy. Pokud se importovaná funkce nebo třída nenačte úspěšně, je do komentáře k testům zaznamenáno, že daná funkce nebyla nalezena.

Každý test obsahuje vstupní data, očekávaný výstup a informace o typech dat. Po přípravě vstupních dat a volání importované funkce nebo metody se výsledek porovná s očekávaným výstupem. Pokud se výsledky shodují, test je označen jako úspěšný. V opačném případě je do komentáře zaznamenáno, který test selhal a jaké byly očekávané a skutečné výsledky.

5.3.3 Statická analýza

Statická analýza kódu je realizována prostřednictvím souboru `static_analysis.py` a využívá nástroj *Pylint*, který je představen v podsekci 2.5.4. Výstup statické analýzy poskytuje komplexní zprávu o nalezených chybách, doporučeních a varováních, které jsou klíčové pro porozumění kvality kódu. Tomuto výstupu je definován vlastní strukturovaný formát.

5.3.4 Klasifikace modelem

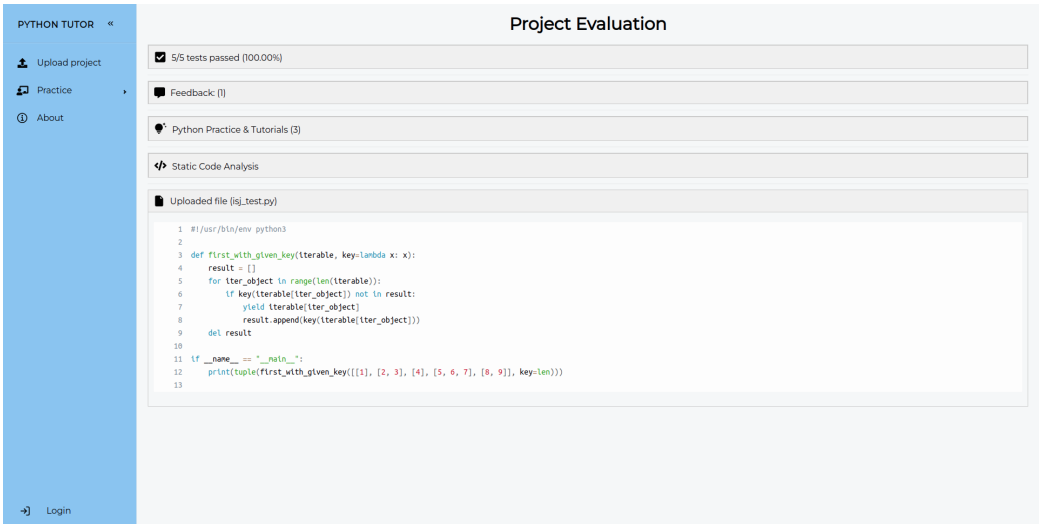
Klasifikace je provedena na odděleném serveru Model. Během jeho inicializační fáze jsou staženy a načteny do paměti všechny klasifikační modely. Tím se minimalizuje doba odpovědi na následné požadavky od hlavního Flask serveru. Před provedením samotné klasifikace je nezbytné načíst konfigurační soubor projektu, který obsahuje důležité parametry a nastavení, které ovlivňují samotný proces klasifikace. Po úspěšném načtení konfiguračního souboru je aplikace připravena k provedení klasifikace nad poskytnutým vstupním řetězcem. Tímto vstupním řetězcem je studentův nahraný projekt, který byl předem očištěn od nadbytečných prvků a připraven pro klasifikační proces. Funkce pro čištění zdrojového kódu odstraňuje komentáře, prázdné řádky a koncové bílé znaky. Při klasifikaci model zpracovává vstupní data a generuje číselnou hodnotu, která reprezentuje klasifikační třídu. Tato číselná hodnota je následně mapována na odpovídající textovou formu, což je výsledek klasifikace, který je prezentován uživateli.

5.3.5 Doporučení na další materiály

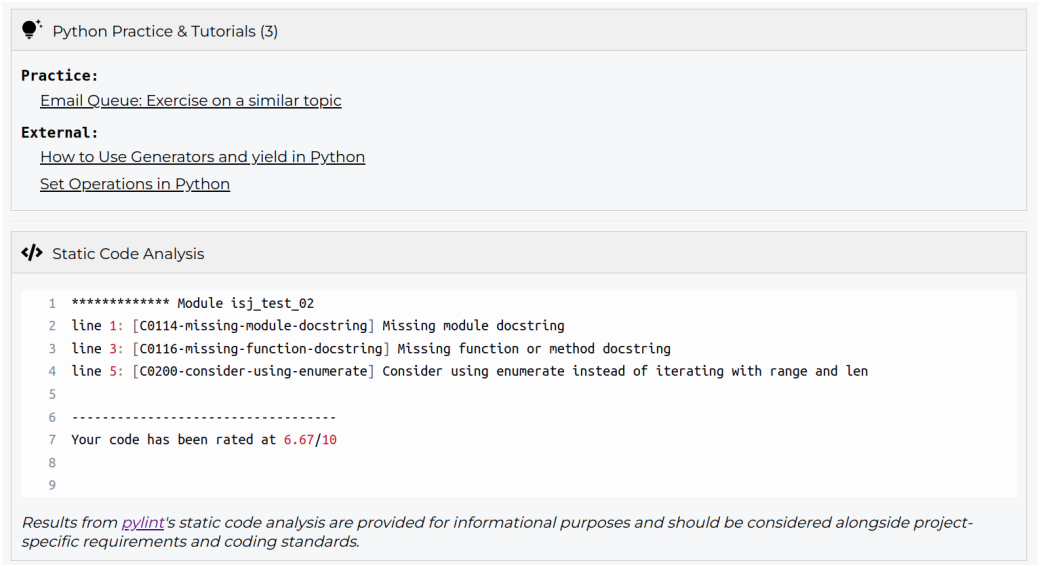
Poslední část zahrnuje doporučení na další materiály, jako jsou interní příklady k procvičení, online kurzy či přehledy k tématům vztahujících se k jednotlivým projektům. Tyto tipy jsou načítány po dokončení testů z konfiguračního souboru ve formátu YAML. Každý projekt v souboru obsahuje odkazy a textové popisy k příslušným zdrojům. Existují dva typy: již zmíněné externí zdroje, které odkazují na materiály mimo systém, a interní praktické úlohy k procvičení, jež jsou detailněji popsány v sekci 5.5.

5.3.6 Zobrazení zpětné vazby

Výsledné rozvržení stránky zobrazující zpětnou vazbu je zobrazeno na obrázku 5.4. Vychází z konceptu navrženého v podsekci 4.2.2. Jednotlivé prvky zpětné vazby jsou v samostatných rozkliknutelných kontejnerech, které zajišťují přehlednost. Tímto způsobem není uživatel při evaluaci přehlcen velkým množstvím textu. Namísto toho má možnost prozkoumat pouze tu část zpětné vazby, která ho zajímá. Již v názvu každé sekce se uživatel dozví dostatek informací, aby se mohl rozhodnout, zda danou sekci rozklikne. Například výsledky testů jsou prezentovány procentuálně. Až po rozkliknutí je možné získat podrobnější informace o tom, který test selhal a s jakými vstupními hodnotami. Obrázek 5.5 názorně ukazuje část rozkliknuté zpětné vazby, konkrétně doporučení na další materiály a výsledky statické analýzy kódu.



Obrázek 5.4: Snímek obrazovky zobrazující stránku se zpětnou vazbou na odevzdaný projekt.



Obrázek 5.5: Snímek obrazovky zobrazující doporučení na další materiály a výsledky statické analýzy kódu.

5.4 Administrační rozhraní

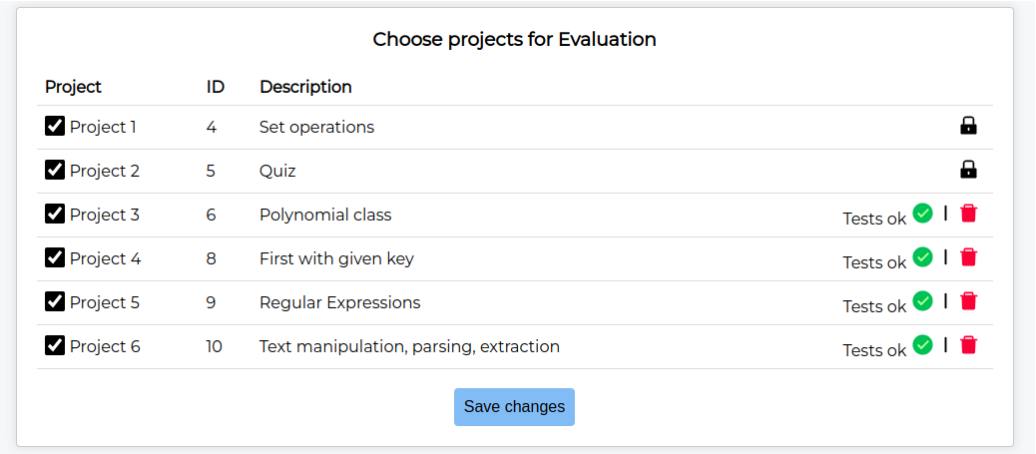
Administrační rozhraní hraje klíčovou roli v kontextu rozšiřitelnosti a kontinuální správy systému. Umožnuje správu projektů, testů a jazykových modelů, které poskytují zpětnou vazbu. Uživatelské rozhraní této sekce systému je navrženo s důrazem na jednoduchost a přehlednost, aby umožnilo snadné provedení potřebných úkonů.

Nastavení je přístupné po přihlášení administrátora. Autentizace probíhá na serveru v modulu `admin`. Důležitým aspektem zabezpečení je oddělení administračního API od veřejného rozhraní.

Na úrovni frontendu je implementována komponenta `PrivateRoute`, která zajišťuje, že pouze přihlášení uživatelé mají přístup k chráněným částem aplikace. Tato komponenta je integrována do routeru aplikace, který je implementován v kořenovém souboru `App.js`.

5.4.1 Správa projektů

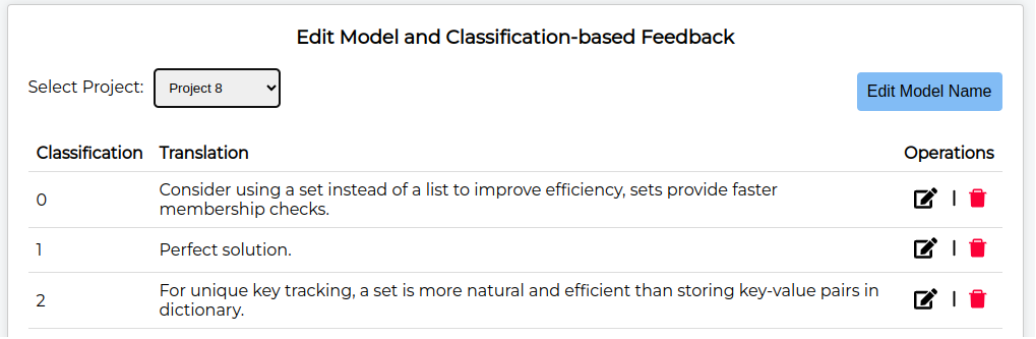
Administrační rozhraní správy projektů umožňuje vytvářet, upravovat, odstraňovat a zpřístupňovat projekty uživatelům. Existují projekty, které nelze genericky testovat a tudíž je není možné přes administrační rozhraní upravovat. Jsou označeny odpovídající ikonou, jak je zobrazeno na snímku obrazovky 5.6. Ve výčtu projektů je také uvedena informace, jestli pro daný projekt existují testy. Při vytváření projektu je možné specifikovat jeho název a popis.



Obrázek 5.6: Snímek obrazovky pro správu projektů.

5.4.2 Správa modelů

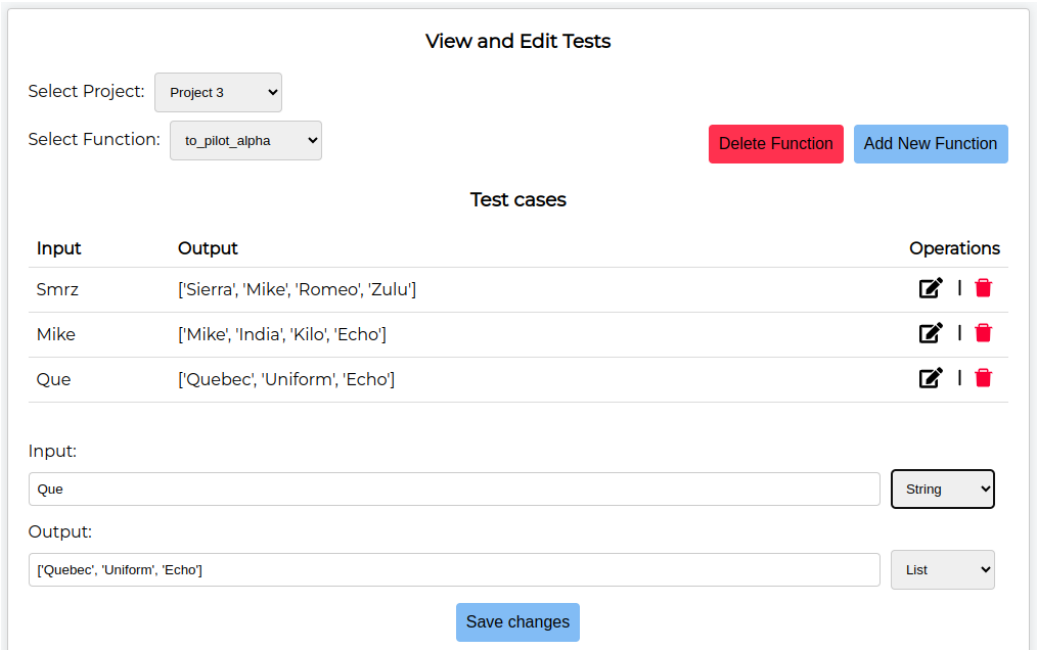
Další částí administrace systému je konfigurace jazykových modelů a jejich zpětné vazby. Ke každému projektu je možné přidat natrénovaný klasifikační model, který je nahraný na platformě HuggingFace. Dále je zde sekce pro mapování výstupu klasifikace na text, který reprezentuje konkrétní zpětnou vazbu.



Obrázek 5.7: Snímek obrazovky pro správu modelů.

5.4.3 Správa testů

Každý projekt obvykle zahrnuje několik funkcí, které jsou testovány. Z tohoto důvodu je nutné pro úpravu testů vybrat projekt a konkrétní funkci. Poté se zobrazí jednotlivé testovací případy v podobě vstupu a očekávaného výstupu, jak ukazuje obrázek 5.8. Název funkce a testovací případy je možné upravovat, přidávat a mazat. Pro definici vstupů a výstupů jsou k dispozici základní datové typy jazyka Python. Několik projektů z minulých let bylo úspěšně vytvořeno právě pomocí tohoto rozhraní.



Obrázek 5.8: Snímek obrazovky pro správu testů.

Testy jsou uloženy v konfiguračním souboru ve formátu JSON. Každý projekt má svůj vlastní blok obsahující název funkce k testování. Ta má své testovací případy. Každý testovací případ obsahuje vstupní hodnotu, typ vstupu, očekávaný výstup a typ výstupu. Na ukázce kódu 5.2 je příklad formátu konfiguračního souboru s testy.

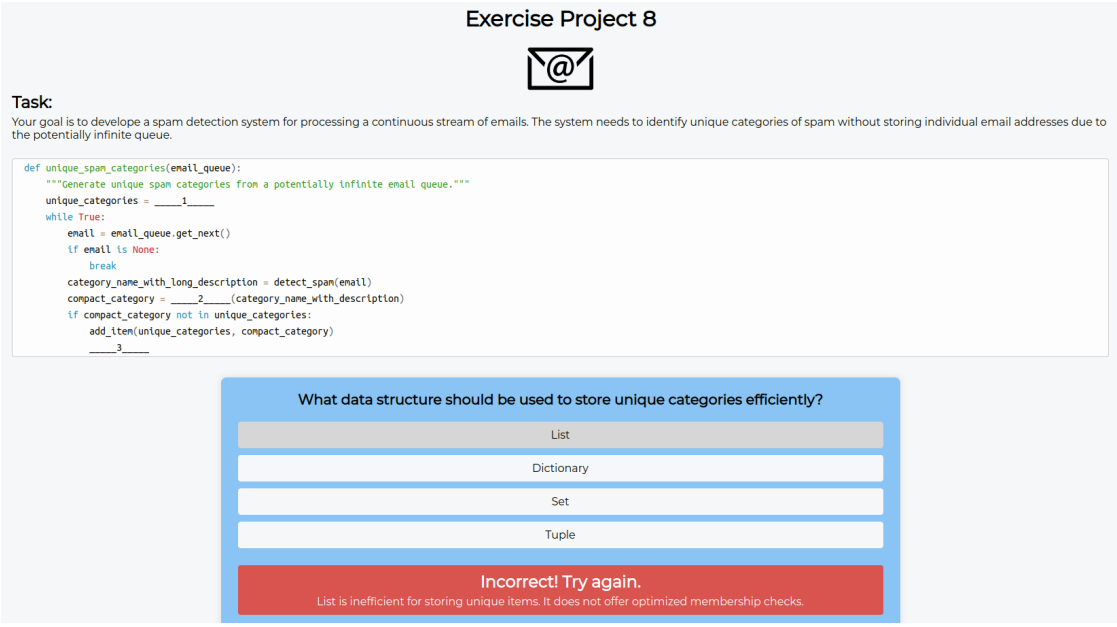
```
{
    "proj3": {
        "to_pilot_alpha": [
            {
                "in": "Smrz",
                "inputType": "string",
                "out": "['Sierra', 'Mike', 'Romeo', 'Zulu']",
                "outputType": "list"
            }
        ]
    }
}
```

Ukázka kódu 5.2: Konfigurační soubor pro testy.

5.5 Příklady na procvičení

V souladu s konceptem navrženým v podsekci 4.3.7 je vytvořena sekce s praktickými úkoly na procvičení, které aplikují principy z projektů. I když bylo původně plánováno vytvořit širší spektrum cvičení, nakonec bylo realizováno pouze několik z nich. Důvodem je, že při vytváření konceptů bylo objeveno mnoho kvalitních materiálů pokrývajících témata projektů, které jsou studentům dostupné online. Část zpětné vazby *Tipy*, která je popsána v podsekci 5.3.5, obsahuje odkazy jak na tyto interní příklady k procvičení tak i veřejně dostupné materiály.

Jedním z typů úkolů je doplňování chybějícího kódu do funkce formou kvízu, jak je zobrazeno na obrázku 5.9. Každý úkol obsahuje krátké zadání, nedokončenou funkci a kvízové otázky. Správná i špatná odpověď v kvízu je zdůvodněna, aby uživatel dostal okamžitou zpětnou vazbu na svůj výběr. Kvízové otázky, odpovědi a jejich zdůvodnění jsou uloženy v konfiguračním YAML souboru, takže jejich správa a rozšiřitelnost je jednoduchá. Tyto data jsou ze serveru zaslána frontendu, který je zpracuje a zajistí kompletní funkcionalitu. K tomuto účelu je vytvořena znovupoužitelná komponenta `Quiz`.



Obrázek 5.9: Snímek obrazovky z úkolu k procvičení.

Kapitola 6

Testování

Testování je klíčovou fází vývoje softwaru, zaměřenou na získání cenné zpětné vazby od uživatelů a odhalení potenciálních nedostatků, které mohou vzniknout v důsledku chybného návrhu či realizace. Identifikované nedostatky jsou řešeny v dalších iteracích vývoje a systém je tak postupně zdokonalován. V této kapitole jsou detailněji popsány různé přístupy k testování systému Python Tutor, jehož návrh a implementace jsou podrobně popsány v předchozích kapitolách 4 a 5. V závěru kapitoly je popsáno vyhodnocení výsledků dotazníkového průzkumu.

6.1 Testování při vývoji

V této sekci jsou popsány různé formy testování prováděné během vývoje aplikace, jako jsou automatizované unit testy, manuální testování nebo zátěžové testy.

6.1.1 Vývojové prostředí a průběžné testování

Během vývoje byly backend (Flask) i frontend (React) služby spouštěny v prostředí vývojových serverů. Tento přístup umožňuje rychlou iteraci vývoje a okamžitou zpětnou vazbu při úpravách kódu. Vývojové servery poskytují automatické načítání změn bez nutnosti ručního restartování aplikace, což výrazně zrychluje vývojový cyklus.

Systém byl již v raných fázích vývoje nasazován na serveru pomocí kontejnerizace, konkrétně pomocí služby Dockeru a jeho orchestrace pomocí Docker Compose. Při vývoji konkrétních funcionalit bylo manuálně provedeno ověření integrity modulů a správného propojení jednotlivých částí aplikace. Dalším klíčovým prvkem vývoje pro zaručení správnosti fungování systému bylo průběžné testování na reálných datech, konkrétně na řešeních projektů studentů z minulých let.

6.1.2 Automatizované testování

Automatizované testování je zaměřeno na testování jednotlivých backendových komponent a funkcí aplikace izolovaně od ostatních částí systému. Pro tento typ testování, označovaný jako *unit testing*, byl zvolen framework **Pytest**[1]. Implementace tohoto frameworku je realizována v modulu `test`. PyTest automaticky detekuje a spouští všechny testovací funkce a třídy, které splňují stanovené konvence pro pojmenování souborů a funkcí. Testování je primárně zaměřeno na ověření správnosti logiky jednotlivých částí systému, správnou

[1] https://docs.pytest.org/

funkcionalitu API rozhraní a validaci vstupních dat. Unit testy byly pravidelně přidávány a spouštěny při každé větší změně kódu a především při nasazování systému na produkční server.

Pro snadné a automatické spouštění všech jednotkových testů v projektu je implementována funkce `run_tests()` v `tests/test_runner.py`, která prochází seznam adresářů serverů a pro každý server spouští unit testy. Složka `tests` na každém serveru obsahuje testovací skripty.

6.1.3 Zátěžové testy

Pro zajištění spolehlivosti a výkonnosti systému je nezbytné provádět testování zátěže, které simuluje různé scénáře použití a množství uživatelů. Jedním z nástrojů, který byl využit pro provádění testů zátěže, je **Locust**[2].

Locust je open-source nástroj pro testování zátěže napsaný v jazyce Python, který umožňuje snadno vytvářet a spravovat testovací scénáře. Umožňuje definovat chování simulovaných uživatelů a monitorovat výkon systému za různých podmínek zátěže.

Pro vytvoření testovacích scénářů s Locustem byly identifikovány klíčové funkcionality systému, které měly být testovány za různých podmínek zátěže. Mezi tyto funkcionality patřilo například nahrávání a zpracování projektů, získávání zpětné vazby nebo zpracování úkolů k procvičení. Na základě identifikovaných scénářů byly vytvořeny testovací skripty pomocí Locustu. Tyto skripty simulovaly chování různého počtu uživatelů provádějících různé operace současně. Během testování bylo monitorováno chování systému vůči různým metrikám jako je průměrná odezva, vytížení serveru, nebo počet úspěšných a neúspěšných požadavků.

Výsledkem testování zátěže je, že systém s jednou instancí backend serveru je schopný obsluhovat větší nápor (nižší stovky) uživatelů v jednom čase. Díky tomu, že studentů v kurzu ISJ je kolem sta, není třeba spouštění dalších instancí, jak bylo navrhováno v podsekci 4.3.1. Toto rozhodnutí se také potvrdilo během semestru, kdy jediná instance serveru zvládla bezprobému obsluhovat všechny studenty.

6.2 Uživatelské testování

Uživatelské testování hraje klíčovou roli při zajišťování kvality aplikace a slouží k identifikaci potenciálních nedostatků a oblastí pro zlepšení. Jeho cílem je získat zpětnou vazbu na celkovou funkčnost a uživatelskou přívětivost od skutečných uživatelů, kteří s používáním systému nemají žádné zkušenosti.

Po dokončení vývoje funkčního prototypu systému bylo zahájeno uživatelské testování, které probíhalo v pravidelných intervalech. Během testování byly uživatelům předloženy detailní scénáře, které pokrývaly několik klíčových částí systému a různé možné cesty, které uživatel mohl při interakci se systémem zvolit. Tyto scénáře zahrnovaly:

- **Nahrání projektů a získání zpětné vazby:** Uživatel měl za úkol nahrát projekt do systému a prozkoumat poskytnutou zpětnou vazbu, která se skládala z několika částí. Cílem bylo ověřit, zda je uživatel schopen správně interpretovat a využít poskytnuté informace.

[2] https://locust.io/

- **Procvičování úkolů:** Uživatel byl požádán, aby vykonával různé úkoly určené k procvičení. Cílem bylo ověřit, zda je systém schopen poskytnout uživateli vhodné úkoly a zda je uživatel schopen je efektivně splnit.
- **Administrace:** Po krátkém zaškolení byl uživatel vyzván k přihlášení a prozkoumání možností administrace systému, včetně správy projektů, testů a modelů.

Tímto způsobem bylo zajištěno důkladné pokrytí všech klíčových funkcí systému a umožněno uživatelům pochopit a využít různé aspekty aplikace. Interakce uživatelů se systémem byla pečlivě sledována a následně analyzována.

Díky relativně přímočaré povaze aplikace nedocházelo během těchto testů k situacím, kdy by uživatel nevěděl, jak postupovat. Na základě získané zpětné vazby byly provedeny drobné úpravy uživatelského rozhraní stránky se zpětnou vazbou a také úkolů k procvičení.

Někteří uživatelé měli potíže s pochopením administračního nastavení. Na základě této zpětné vazby byly provedeny odpovídající úpravy s cílem zvýšit intuitivnost nastavení. Celkově ale nebyl na tuto zpětnou vazbu kladen takový důraz, neboť se předpokládá, že administrátor bude se systémem řádně seznámen. Systém byl také pravidelně prezentován garantovi předmětu ISJ.

6.2.1 Ostré testování

Aplikace byla nasazena a testována přímo studenty kurzu ISJ, pro které je také určena. Systém byl zveřejněn studentům v průběhu letního semestru 2024 a ti jej aktivně využívali. Po celou dobu semestru nebyly zaznamenány žádné vážné komplikace. Systém byl na základě zpětné vazby průběžně vylepšován.

Díky pečlivému logování byly zachyceny opakující se pokusy některých uživatelů o nahrání neobvyklých skriptů. Tyto skripty měly za cíl získat citlivé informace ze serveru, stáhnout kód z externích zdrojů nebo byly příliš velké (desítky MB). Bezpečnostní opatření implementovaná v systému úspěšně předešla většině těchto pokusů. Pouze v jednom případě došlo k úspěšnému získání informací o testovacím serveru. A to konkrétně u testování negenerického projektu, u kterého nebylo ošetřeno vracení chybových zpráv na frontend. Toho útočník využil a vypsal adresář kontejneru. Tento incident byl vyřešen tím, že systém již na žádném místě nezobrazuje konkrétní chybové zprávy. Další incidenty zaznamenány nebyly.

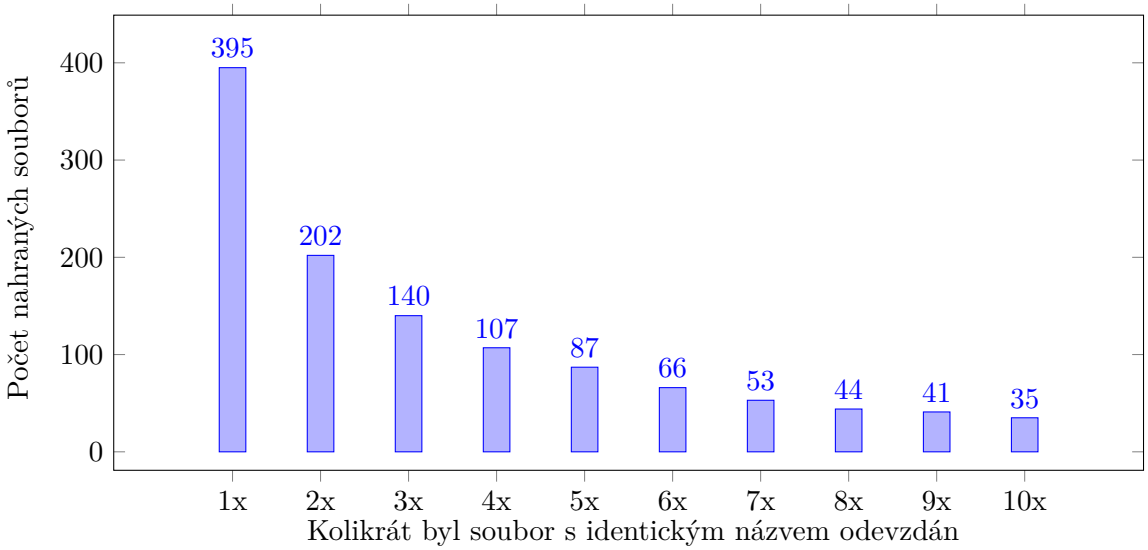
Analýza odevzdaných řešení

Analýza odevzdaných řešení byla umožněna díky ukládání všech nahraných souborů od uživatelů a podrobnému logování. Hlavním cílem této analýzy bylo zkoumání míry zlepšení kvality postupně odevzdávaných projektů na základě poskytnuté zpětné vazby.

Žádný obecný skript na analýzu zlepšení odevzdávaných řešení vytvořen nebyl, protože každý projekt je odlišný. Analýza byla prováděna pomocí jednoduchých shellových skriptů a také ručně, tj. procházením a porovnáním postupně odevzdaných souborů a logů. Ruční analýzou bylo zjištěno, že studenti s každou iterací vylepšovali svá řešení převážně na základě výsledků testů a doporučení nástroje provádějící statickou analýzu kódu. V menší míře byly také zjištěny změny provedené na základě doporučení modelu. Několik uživatelů také testovalo samotnou zpětnou vazbu poskytovanou modely, kdy modifikovali i optimální řešení, jen aby vyzkoušeli různé reakce modelu.

Celkově bylo do systému během semestru nahráno 1378 souborů k 8 různým projektům, z nichž 395 bylo dle názvu jedinečných. Systém tedy využila přibližně polovina ze zapsaných

studentů kurzu ISJ. Graf na obrázku 6.1 zobrazuje distribuci opakovaných nahrání souborů s identickým názvem. Z grafu také vyplívá, že téměř 50 % uživatelů nahrálo své řešení projektu více než jednou. Domnívám se, že tento výrazný rozdíl mezi prvním odevzdáním a opakovaným může být způsoben tím, že během semestru měli studenti k dispozici již existující řešení. Mnoho z nich tedy nový systém využívalo pouze k poslední kontrole své práce. Výsledná čísla jsou spíše orientační, protože nahrávání projektů je anonymní a jediný způsob identifikace je na základě názvu souboru a času odevzdání. Názvy souborů typicky obsahovaly login studenta, protože takto pojmenované soubory se do systému odevzdávaly k hodnocení. Při ruční kontrole se obecně pojmenovaných souborů (například *projekt.py* či *isj.py*) nacházel minimální počet.



Obrázek 6.1: Graf zobrazující distribuci opakovaných nahrání souborů s identickým názvem. Na horizontální ose je uvedeno kolikrát byl soubor s identickým názvem odevzdán, zatímco vertikální osa udává počet nahraných souborů.

6.2.2 Dotazník

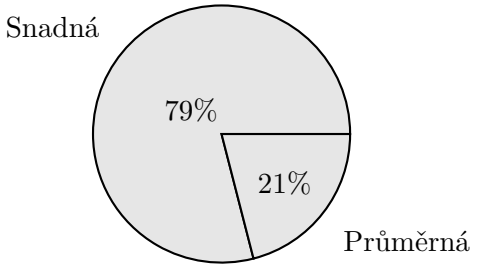
Cílem dotazníku bylo získat užitečnou zpětnou vazbu od uživatelů ohledně jejich zkušenosti s používáním systému a jeho uživatelského rozhraní. Dotazník obsahuje poměrně široké spektrum otázek zaměřených na různé aspekty uživatelské zkušenosti, včetně celkového hodnocení práce se systémem, uživatelského rozhraní, poskytované zpětné vazby a případných možností vylepšení.

Dotazník byl studentům distribuován na konci semestru a vyplnilo ho celkem 19 respondentů. Kromě standardních otázek umožňoval i volné textové odpovědi, aby studenti mohli své názory zdůvodnit. Kompletní data získaná během dotazníkového průzkum jsou k dispozici v příloze B.

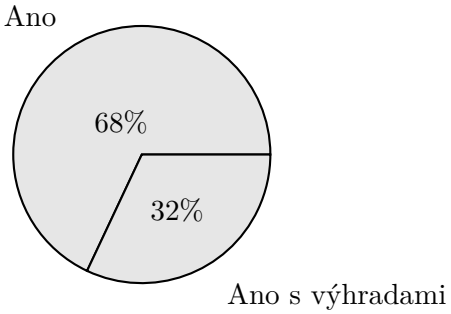
Uživatelské rozhraní

Studenti hodnotí práci se systémem jako snadnou nebo průměrnou. Uživatelské rozhraní získalo převážně pozitivní hodnocení. Výsledky této části dotazníku jsou zaznamenány v koláčových grafech na obrázku 6.2. Připomínky studentů ohledně uživatelského rozhraní byly

různorodé. Někteří kritizovali, že je příliš jednoduché, a upřednostňovali by modernější design. Naopak jiní si jeho jednoduchost chválili.



(a) Otázka: *Jak hodnotíš celkovou práci se systémem?*

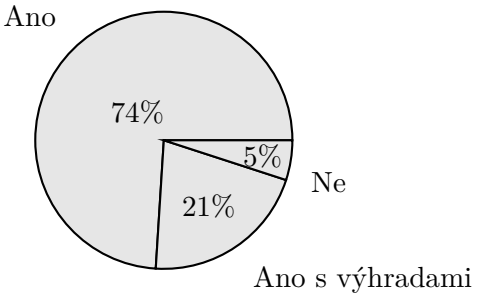


(b) Otázka: *Líbí se ti uživatelské rozhraní systému?*

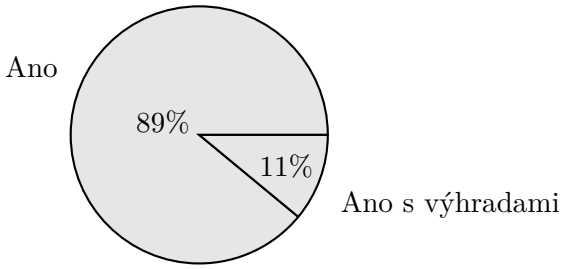
Obrázek 6.2: Grafy zahrnující výsledky dotazníku ohledně práce se systémem a uživatelského rozhraní.

Zpětná vazba

Většina respondentů vnímá zpětnou vazbu jako užitečnou, jak je patrné z grafu na obrázku 6.3. Někteří studenti vyjádřili výhrady, především kvůli nedostatečně detailním výsledkům testů. Vyzdvihli by, kdyby komentář k výsledkům testů obsahoval nejen vstup a očekávaný výstup, ale i výstup funkce. Ten však z bezpečnostních důvodů záměrně nebyl implementován. Většina respondentů potvrdila, že doporučení od dotrénovaných modelů fungovalo bez problémů. Někteří studenti si všimli absence tohoto typu zpětné vazby u prvních projektů. Tato absence však byla zcela záměrná, jelikož se jednalo o relativně jednoduché úkoly, u kterých tento typ zpětné vazby nebyl nezbytný. Do budoucna je možné dotrénovat modely na všechny projekty, aby se zpětná vazba sjednotila.



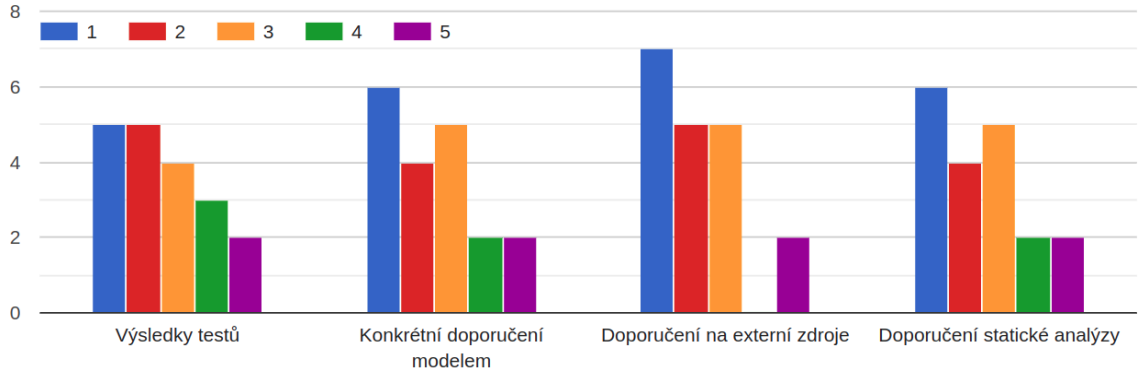
(a) Otázka: *Přišla ti zpětná vazba celkově užitečná?*



(b) Otázka: *Fungovalo doporučení od dotrénovaného modelu správně?*

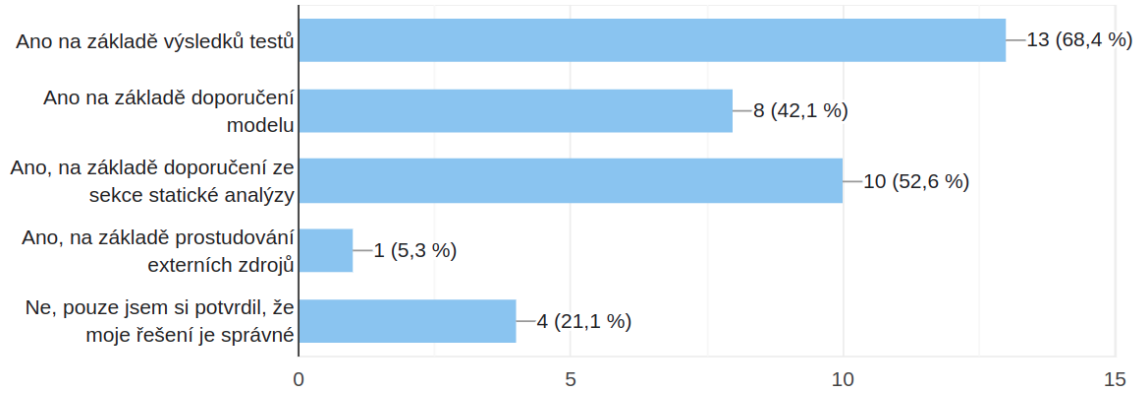
Obrázek 6.3: Grafy zahrnující výsledky dotazníku ohledně užitečnosti zpětné vazby, kterou systém poskytuje na projekty.

Na obrázku 6.4 jsou prezentovány výsledky hodnocení užitečnosti jednotlivých částí zpětné vazby. Celkově je hodnocení pozitivní a mezi jednotlivými částmi zpětné vazby nejsou významné rozdíly.



Obrázek 6.4: Graf zobrazující výsledky hodnocení užitečnosti jednotlivých částí zpětné vazby. Respondenti oznámkovali symboly od 1 do 5 jako ve škole.

Dále většina respondentů uvedla, že na základě poskytnuté zpětné vazby vylepšila své řešení. Nejčastěji se vylepšení týkala reakce na výsledky testů a statickou analýzu, jak ukazuje graf na obrázku 6.5. Tento výsledek potvrdila i ruční analýza postupně odevzdávaných řešení, která je popsána v podsekci 6.2.1. Procentuálně malé množství respondentů uvádí, že své řešení vylepšilo na základě studia externích či interních materiálů.

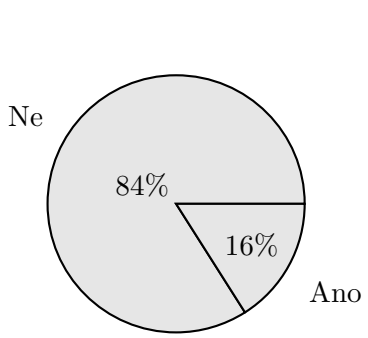


Obrázek 6.5: Graf zobrazující výsledky z dotazníku na otázku: *Vylepšil jsi svoje řešení na základě zpětné vazby?*

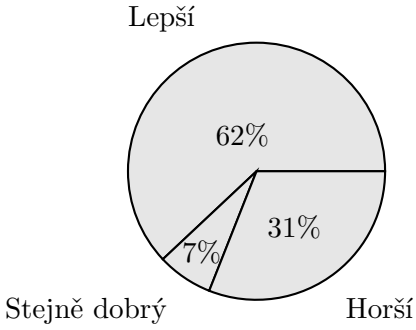
Úkoly k procvičení a porovnání s existujícím řešením

Pouze 16 % respondentů vyzkoušelo úkoly k procvičení. To může být zapříčeněné nulovou vnější motivací, špatným zpracováním úkolů nebo tím, že úkolů je málo a v průběhu semestru nebyly dále rozšiřovány, jak bylo diskutováno v sekci 5.5. Výsledky jsou zachyceny v grafech na obrázku 6.6.

Většina respondentů (62 %) hodnotí nový systém Python Tutor jako lepší v porovnání s existujícím řešením. Mezi hlavní výhody nového systému uváděli: rychlost, lepší uživatelské rozhraní a detailnější zpětná vazba v případě selhání testů. Respondenti také oceňují, že systém byl oproti existujícímu řešení dostupný po celou dobu semestru.



(a) Otázka: *Vyzkoušel jsi některý z úkolů k procvičení?*



(b) Otázka: *Jaký ti přijde nový systém v porovnání s tím existujícím?*

Obrázek 6.6: Grafy zahrnující výsledky dotazníku ohledně úkolů k procvičení a porovnání nového a existujícího systému.

6.2.3 Shrnutí

Na základě testování byly provedeny zásadní i drobné úpravy, které vedly k tomu, aby výsledná aplikace vyhovovala co možná nejvíce uživatelům. Z celkové zpětné vazby vyplynulo, že navigace v celé aplikaci je jednoduchá a srozumitelná. Nikdo z testovaných neměl problém s použitím systému v žádném z testovacích scénářů. Uživatelské testování přineslo cenné poznatky o funkčnosti a uživatelské přívětivosti systému. Nebyl nalezen žádný závažnější problém, byly vytýkány spíše drobnosti.

Dotazník poskytl cenné poznatky o uživatelském vnímání a zkušenostech s novým systémem pro poskytování zpětné vazby v rámci výuky programování v jazyce Python. Z výsledků vyplývá, že většina respondentů hodnotí nový systém pozitivně, přičemž jako jeho hlavní výhody uvádějí rychlost, lepší uživatelské rozhraní a detailnější zpětnou vazbu v případě selhání testů. Existující řešení bylo často nedostupné během semestru, což ovlivnilo jejich preference ve prospěch nového systému. Zároveň bylo zjištěno, že úkoly k procvičení byly využity pouze menšinou respondentů. Tyto poznatky jsou klíčové pro další vylepšení systému a optimalizaci jeho využití v rámci vzdělávacího procesu v kurzu ISJ.

Kapitola 7

Závěr

Cílem této práce bylo vytvořit inteligentní prostředí, které studentům poskytuje personalizovanou zpětnou vazbu a podporu při jejich učebním procesu programování v jazyce Python. Implementace tohoto systému byla úspěšně realizována díky kombinaci webových technologií a využití jazykových modelů, které umožnily poskytnutí zpětné vazby na studentské projekty. Při analýze možných řešení byly zkoumány možnosti a omezení velkých jazykových modelů. Pro realizaci systému byly zvoleny klasifikační modely typu Sentence Transformers. Během testování systému se potvrdilo, že dotrénování tohoto typu modelu přispívá k vylepšení kvality poskytované zpětné vazby. Systém byl nasazen během výuky kurzu Skriptovací jazyky na FIT VUT v Brně, kde si získal pozitivní ohlasy od studentů. Pozitivní byly také výsledky dotazníkového průzkumu, který byl proveden na konci semestru. Systém byl průběžně vylepšován na základě jejich zpětné vazby. Nově navržený systém přináší výrazné vylepšení oproti stávajícímu řešení, což bylo potvrzeno také prostřednictvím testování, dotazníku a zpětné vazby od uživatelů.

Možnosti dalšího rozšíření zahrnují například detekci kódu vytvořeného pomocí umělé inteligence. Tento trend je v současné době mezi studenty velmi rozšířený, zejména v kontextu implementace základních funkcí. Dalším možným rozšířením je přidání chatbota s dostatečnou znalostí kurzu, který by byl schopný poskytovat adekvátní informace o projektech a efektivním programování. Dále je možné rozšířit sbírku úkolů k procvičení nebo vylepšit uživatelské rohraní administrace. Posledním navrhovaným rozšířením je vytvoření jednoho rozsáhlejšího projektu, který by se zaměřil na komplexnější praktický problém. Cílem by bylo propojit všechny úkoly k procvičení do jednotného celku. Tento projekt by zahrnoval jednotlivé části, které by pokrývaly témata dosavadních projektů. Studenti by tak měli možnost postupně během semestru implementovat celý projekt, což by jim umožnilo lépe porozumět významu efektivního kódu a praktické využití jednotlivých projektů.

Záměr práce byl splněn a studenti byly s výsledným systémem spokojeni. Výsledkem je ucelený a funkční systém, který splnil svůj původní záměr a přispěl k efektivnějšímu a interaktivnějšímu vzdělávacímu procesu v oblasti programování v jazyce Python. Jsem otevřený i drobným úpravám a správě systému do budoucna, které by mohly být třeba v důsledku používání v dalších semestrech výuky.

Literatura

[1] Ait, A., Izquierdo, J. L. C. a Cabot, J. HFCommunity: A tool to analyze the hugging face hub community. In: IEEE. *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 2023, s. 728–732.

[2] Anderson, L. W. a Krathwohl, D. R. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives: complete edition*. 1st. Addison Wesley Longman, Inc., 2001. ISBN 978-080131903.

[3] Andreev, I. *Bloom's Taxonomy*. Květen 2023. [cit. 2023-09-27]. Dostupné z: https://www.valamis.com/hub/blooms-taxonomy.

[4] Burkhardt, J. *A Dynamic Analysis-Based Linter for Python*. 2023. 91 s. Diplomová práce. University of Stuttgart, Faculty of Computer Science, Electrical Engineering, and Information Technology, Germany. Dostupné z: http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=MSTR-2023-82&engl=1.

[5] Chen, J., Lin, H., Han, X. a Sun, L. Benchmarking large language models in retrieval-augmented generation. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2024, sv. 38, č. 16, s. 17754–17762.

[6] Cui, B., Li, J., Guo, T., Wang, J. a Ma, D. Code comparison system based on abstract syntax tree. In: IEEE. *2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*. 2010, s. 668–673.

[7] De Raadt, M. *Teaching programming strategies explicitly to novice programmers*. 2008. Disertační práce. University of Southern Queensland.

[8] Denny, P., Whalley, J. a Leinonen, J. Promoting early engagement with programming assignments using scheduled automated feedback. In: *Proceedings of the 23rd Australasian Computing Education Conference*. 2021, s. 88–95.

[9] *Google Cloud Documentation*. Google, 2023. [cit. 2023-11-28]. Dostupné z: https://cloud.google.com/vertex-ai/generative-ai/docs/.

[10] Huang, Y., Brusilovsky, P., Guerra, J., Koedinger, K. a Schunn, C. Supporting skill integration in an intelligent tutoring system for code tracing. *Journal of Computer Assisted Learning*. Wiley Online Library. 2023, sv. 39, č. 2, s. 477–500.

[11] *Hugging Face Hub documentation*. Hugging Face, 2023. [cit. 2023-12-05]. Dostupné z: https://huggingface.co/docs.

[12] John, P. *Výuka pokročilých konstrukcí jazyka Python na základě poskytování zpětné vazby ke studentským kódům*. Brno, CZ, 2020. Bakalářská práce. Brno University of Technology, Faculty of Information Technology. Dostupné z: https://www.fit.vut.cz/study/thesis/22441/.

[13] Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D. et al. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and individual differences*. Elsevier. 2023, sv. 103, s. 102274.

[14] Lahtinen, E., Ala Mutka, K. a Järvinen, H. A study of the difficulties of novice programmers. Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education. *ACM, New York*. 2005, sv. 14, 10.1145, s. 1067445–1067453.

[15] Leinonen, J., Castro, F. E. V. a Hellas, A. Does the early bird catch the worm? Earliness of students' work and its relationship with course outcomes. In: *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. 2021, s. 373–379.

[16] Leinonen, J., Denny, P. a Whalley, J. A comparison of immediate and scheduled feedback in introductory programming projects. In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1*. 2022, s. 885–891.

[17] Liu, E., Stephan, M., Nie, A., Piech, C., Brunskill, E. et al. Giving Feedback on Interactive Student Programs with Meta-Exploration. *Advances in Neural Information Processing Systems*. 2022, sv. 35, s. 36282–36294.

[18] Loukides, M. Automated Mentoring with ChatGPT. October 2023. [cit. 2023-10-25]. Dostupné z: https://www.oreilly.com/radar/automated-mentoring-with-chatgpt/.

[19] Moraes, L. O. a Pedreira, C. E. Designing an Intelligent Tutoring System Across Multiple Classes. In: *CSEDM@ EDM*. 2020.

[20] *OpenAI API Documentation*. OpenAI, 2023. [cit. 2023-11-22]. Dostupné z: https://platform.openai.com/docs.

[21] OpenAI. *OpenAI Pricing*. 2023. [cit. 2023-11-05]. Dostupné z: https://openai.com/pricing/.

[22] Qihuang, Z., Ding, L., Liu, J., Du, B. a Tao, D. Can ChatGPT Understand Too? A Comparative Study on ChatGPT and Fine-tuned BERT. Únor 2023. DOI: 10.48550/arXiv.2302.10198.

[23] Ram, B. a Verma, P. Artificial intelligence AI-based Chatbot study of ChatGPT, Google AI Bard and Baidu AI. *World Journal of Advanced Engineering Technology and Sciences*. 2023, sv. 8, č. 01, s. 258–261.

[24] Rus, V. a Graesser, A. C. Deeper natural language processing for evaluating student answers in intelligent tutoring systems. In: Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. *Proceedings of the national conference on artificial intelligence*. 2006, sv. 21, č. 2, s. 1495.

[25] Schwartz, S. What ChatGPT Could Mean for Tutoring. May 2023. [cit. 2023-10-28]. Dostupné z: https://www.edweek.org/technology/what-chatgpt-could-mean-for-tutoring/2023/05.

[26] Sharma, P. a Harkishan, M. Designing an intelligent tutoring system for computer programing in the Pacific. *Education and Information Technologies*. Springer. 2022, sv. 27, č. 5, s. 6197–6209.

[27] Spirin, E., Bogomolov, E., Kovalenko, V. a Bryksin, T. Psiminer: A tool for mining rich abstract syntax trees from code. In: IEEE. *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. 2021, s. 13–17.

[28] Sychev, O., Penskoy, N., Anikin, A., Denisov, M. a Prokudin, A. Improving comprehension: Intelligent tutoring system explaining the domain rules when students break them. *Education Sciences*. MDPI. 2021, sv. 11, č. 11, s. 719.

[29] Taggart, M. CS Ed Week, Part 3: Programming and Bloom's Taxonomy. *The Forever Student*. Prosinec 2017. [cit. 2023-09-27]. Dostupné z: https://theforeverstudent.com/cs-ed-week-part-3-programming-and-blooms-taxonomy-151cfc0d550f.

[30] Tunstall, L., Reimers, N., Jo, U. E. S., Bates, L., Korat, D. et al. SetFit: Efficient Few-Shot Learning Without Prompts. *Hugging Face Blog*. September 2022. [cit. 2023-12-15]. Dostupné z: https://huggingface.co/blog/setfit.

Příloha A

Manuál na spuštění systému

Projekt je dostupný ve veřejném repozitáři na githubu[1] nebo na přiloženém paměťovém médiu. Struktura celého projektu je popsána v příloze C. Po stažení adresáře s projektem stačí jednotlivé služby spustit v závislosti na typu prostředí. Návody k vývojovému a produkčnímu prostředí jsou sepsány níže. Logy se vypisují do souborů ve složce `/logs`.

Vývojové prostředí

Pro spuštění vývojového prostředí bez využití Dockeru je třeba mít nainstalován **Node.js** a **npm** (Dockerfile má image node:16-alpine, který používá npm 7.19.1). Dále **Python** (Dockerfile má image python:3.9-slim, který používá Python 3.9) a správce balíčků **pip**.

Frontend

Ve složce `/client` se nachází React frontend. Nejdříve je třeba instalace závislostí příkazem:

```
npm install
```

a následné spuštění pomocí:

```
npm start
```

Backend

Všechny backend servery jsou Flask projekty. Po navigaci do adresáře serveru, například `/server/model`, je třeba instalovat závislosti pomocí pip:

```
pip install -r requirements.txt
```

a následné spuštění pomocí:

```
flask run
```

nebo alternativně:

```
python3 server/model/app.py
```

[1] https://github.com/xkrejc70/python-tutor

Produkční prostředí

Pro nasazení aplikace na produkci je nutné mít nainstalovaný Docker a Docker Compose. Dále je nutné mít na serveru otevřené porty pro client a flask server, aby byla aplikace dostupná pro uživatele zvenčí.

Sestavení a spustění kontejnerů proběhne pomocí následujícího příkazu:

```
docker compose up -d --build
```

přepínač `-d` indikuje, že kontejnery budou spuštěny v tzv. *detach* režimu, takže poběží na pozadí. Pro zastavení všech kontejnerů aplikace je možné spustit:

```
docker compose down
```

a pro zobrazení běžících kontejnerů:

```
docker ps
```

Trénování modelů

K dotrénování modelu `all-mpnet-base-v2` slouží skript `data/setfit-train.py`. Je třeba přidat testovací a validační dataset a repozitář Hugging Face, kam se model po dotrénování nahraje. Dataset obsahující klasifikační třídy lze vytvořit přímo na platformšě Hugging Face.

Osobně doporučuji při začlenění nového typu projektu zpětnou vazbu na základě hodnocení modelu první rok neposkytovat. Až po skončení semestru analyzovat všechny odevzdané řešení a vytvořit zpětné vazby na základě různých neoptimálních typů řešení studentů.

Příloha B

Výsledky dotazníkového průzkumu

V tabulkách B.1, B.2, B.3 a B.4 jsou uvedena kompletní data získaná během dotazníkového průzkumu, který proběhl v rámci testování systému Python Tutor vyvinutého v rámci této práce. Podrobné vyhodnocení dotazníku je popsáno v podsekci 6.2.2.

Jak hodnotíš celkovou práci se systémem?	Máš nějaké **výhrady** k práci se systémem?	Líbí se ti uživatelské rozhraní systému?	Výhrady k uživatelskému rozhraní?
Snadná		Ano	
Průměrná		Ano s výhradami	Je dost jednoduché
Snadná		Ano	
Snadná	ne	Ano	žádné
Snadná	n	Ano	
Průměrná		Ano	
Snadná		Ano s výhradami	Chybí mi eye candy :'(
Snadná		Ano	
Průměrná	u failnutych testu neukazuje got, jen expected, kvuli cemu si myslim ze jsou testy spatne, u 8. projektu get_urls() jsem test case spustil na svojem zarizeni vysledek byl stejny jako expected	Ano s výhradami	občas matoucí/nepřehledne
Snadná		Ano	
Snadná	System by mohol ukazovat vysledky testov presnejsie, napr. ak nejaky test nepresiel, tak ukaze ocakavany vystup a podobne. Mozno to tak funguje, ale v poslednej ulohe som nic take nevidel.	Ano s výhradami	Velmi jednoduchy dizajn, pacilo by sa mi nieco modernejsie
Snadná		Ano	
Snadná		Ano	
Průměrná		Ano s výhradami	Nic extra
Snadná		Ano	Pri Upload som musel kliknúť 2x krát
Snadná		Ano	
Snadná	Je skvělý	Ano	Je velmi jednoduché, ale to je na
Snadná		Ano	
Snadná		Ano s výhradami	

Obrázek B.1: Výsledky dotazníku (1/4).

Přišla ti zpětná vazba celkově **užitečná**?	Jak hodnotíš užitečnost jednotlivých částí zpětné vazby? Oznámkuj jako na základce. [Výsledky testů]	Jak hodnotíš užitečnost jednotlivých částí zpětné vazby? Oznámkuj jako na základce. [Konkrétní doporučení modelem]	Jak hodnotíš užitečnost jednotlivých částí zpětné vazby? Oznámkuj jako na základce. [Doporučení na externí zdroje]	Jak hodnotíš užitečnost jednotlivých částí zpětné vazby? Oznámkuj jako na základce. [Doporučení statické analýzy]
Ano	1	1	2	4
Ano	2	2	2	2
Ano, s výhradami	3	3	3	1
Ano	1	1	1	1
Ano	1	1	1	1
Ano	4	3	3	3
Ano	1	1	1	1
Ano	4	5	5	5
Ne	5	3	3	3
Ano, s výhradami	3	1	1	1
Ano, s výhradami	3	5	3	5
Ano	5	4	2	4
Ano	1	1	1	1
Ano	2	2	2	2
Ano	3	3	1	3
Ano	2	2	1	2
Ano, s výhradami	4	4	5	3
Ano	2	3	3	3
Ano	2	2	2	2

Obrázek B.2: Výsledky dotazníku (2/4).

Máš nějaké **výhrady** k některé části zpětné vazby?	Fungovalo doporučení od dotrénovaného modelu správně? (sekce **Feedback**)	Máš nějaké **výhrady** k doporučení od dotrénovaného modelu?	Vylepšil jsi svoje řešení na základě zpětné vazby?	Vyzkoušel jsi **úkoly k procvičení**?	Máš nějaké **výhrady** k úkolům k procvičení?
	Ano, nezaznamenal jsem žádný problém	U některých projektů tato zpětná vazba chybí	Ano na základě výsledků testů, Ano na základě doporučení modelu	Ne	
	Ano, s výhradami	Nevšiml jsem si této zpětné vazby	Ne, pouze jsem si potvrdil, že moje řešení je správné	Ne	
	Ano, nezaznamenal jsem žádný problém		Ne, pouze jsem si potvrdil	Ne	
	Ano, nezaznamenal jsem žádný problém		Ano na základě výsledků testů, Ano na základě doporučení modelu, Ano, na základě doporučení ze sekce statické analýzy	Ano	
	Ano, nezaznamenal jsem žádný problém		Ano na základě výsledků testů, Ano, na základě doporučení ze sekce statické analýzy	Ne	
	Ano, s výhradami		Ne, pouze jsem si potvrdil, že moje řešení je správné	Ne	
	Ano, nezaznamenal jsem žádný problém		Ano na základě doporučení modelu	Ne	
Pri poslednom projekte bola nezhoda očakávaného výstupu v testoch a v zadaní	Ano, nezaznamenal jsem žádný problém		Ano na základě výsledků testů, Ano na základě doporučení modelu	Ne	
u failnutych testu neukazuje got, jen expected, kvuli cemu ai	Ano, nezaznamenal jsem žádný problém		Ne, pouze jsem si potvrdil, že moje řešení je správné	Ne	
	Ano, nezaznamenal jsem žádný problém		Ano na základě výsledků testů, Ano na základě doporučení modelu, Ano, na základě doporučení ze sekce statické analýzy	Ne	
malo detailne vysledky testov	Ano, nezaznamenal jsem žádný problém		Ano, na základě doporučení ze sekce statické analýzy	Ne	
	Ano, nezaznamenal jsem žádný problém		Ano na základě výsledků testů, Ano, na základě doporučení ze sekce statické analýzy	Ne	
	Ano, nezaznamenal jsem žádný problém		Ano na základě výsledků testů, Ano na základě doporučení modelu, Ano, na základě doporučení ze sekce statické analýzy, Ano, na základě prostudování externích zdrojů	Ano	
	Ano, nezaznamenal jsem žádný problém		Ano na základě výsledků	Ano, mám k nim	Misto kvizu bych d
	Ano, nezaznamenal jsem žádný problém		Ano na základě výsledků testů, Ano na základě doporučení modelu, Ano, na základě doporučení ze sekce statické analýzy	Ne	
	Ano, nezaznamenal jsem žádný problém		Ano na základě výsledků testů, Ano, na základě doporučení ze sekce statické analýzy	Ne	
	Ano, nezaznamenal jsem žádný problém		Ano na základě výsledků testů, Ano, na základě doporučení ze sekce statické analýzy	Ne	
	Ano, nezaznamenal jsem žádný problém		Ano na základě výsledků testů, Ano, na základě doporučení ze sekce statické analýzy	Ne	
	Ano, nezaznamenal jsem žádný problém		Ano na základě výsledků testů	Ne	

Obrázek B.3: Výsledky dotazníku (3/4).

Jaký ti přijde nový systém v porovnání s tím předchozím? (isj.fit.vutbr.cz)	Proč je nový systém lepší/horší?
Nový systém je lepší	Rychlejší, lepší UI, lepší doporučení
Nový systém je lepší	Vrací i vstupy testů, které selžou
Oba systémy jsou dobré	testy fungují asi stejně
Nový systém je lepší	funguje podstatně lépe
Oba systémy jsou dobré	Obojí plní svůj účel
Oba systémy jsou dobré	-
Nový systém je lepší	Vypadá líp, je přehlednější, více informací.
Nový systém je lepší	Funguje (502 bad gate pri starom)
Nový systém je horší	Starý/aktualní systém mně přijde přehlednější, a lepšími testy, ale tam
Nový systém je lepší	Však ten starý polku času nefungoval
Nový systém je lepší	Rychlost + stary ani nefunguje :D
Oba systémy jsou dobré	nepoznam stary system
Nový systém je lepší	Je lepší protože funguje a je jednoduché se v něm orientovat.
Nový systém je lepší	funguje :D
Oba systémy jsou dobré	Príde mi to rovnaké
Nový systém je lepší	Hezci uzivatelske rozhrani
Nový systém je lepší	Vždy fungoval a byl velice rychlý
Oba systémy jsou dobré	.
Nový systém je lepší	-

Obrázek B.4: Výsledky dotazníku (4/4).

Příloha C

Obsah přiloženého paměťového média

Paměťové médium přiložené k práci má následující hierarchii:

```
SD Card
├── doc
├── poster
└── python-tutor
```

Jednotlivé složky obsahují:

- **doc** – zdrojové soubory závěrečné práce v systému LaTeX.
- **poster** – zdrojové soubory plakátu v systému LaTeX.
- **python-tutor** – zdrojové soubory webové aplikace, struktura je detailněji popsána v podsekci 5.1.2.