



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

**PROCVIČOVÁNÍ MATEMATIKY ZŠ S MOŽNOSTÍ
HLASOVÉHO VKLÁDÁNÍ ŘEŠENÍ PŘÍKLADŮ**

BASIC SCHOOL MATH DRILL PLATFORM WITH ENTERING ANSWERS BY VOICE

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

DOMINIK HORUT

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. IGOR SZÓKE, Ph.D.

BRNO 2025

Zadání bakalářské práce



164214

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
Student: **Horut Dominik**
Program: Informační technologie
Název: **Procvičování matematiky ZŠ s možností hlasového vkládání řešení příkladů**
Kategorie: Uživatelská rozhraní
Akademický rok: 2024/25

Zadání:

1. Nastudujte platformy pro výuku a procvičování matematiky pro ZŠ (např. matika.in, umimematiku.cz). Soustředte se na užití formou drilu. Nastudujte dostupné služby (API) pro diktování a zadávání dat hlasem.
2. Navrhněte platformu pro dril příkladů z matematiky. Soustředte se na uživatelskou zkušenost a jednoduchost.
3. Integrujte možnost zadávat výsledky příkladů hlasem. Ověřte a vyhodnoťte, jestli/kdy tato modalita má smysl.
4. Zhodnoťte dosažené výsledky a navrhněte směry dalšího vývoje.
5. Vytvořte A2 plakátek nebo 20-30 vteřinové video o Vašem projektu.

Literatura:

- Tidwell et al.: Designing Interfaces: Patterns for Effective Interaction Design, O'Reilly, 2020
- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN: 978-0321965516, 2013
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299, 2009
- Joel Marsh: UX for Beginners: A Crash Course in 100 Short Lessons, O'Reilly 2016

Při obhajobě semestrální části projektu je požadováno:

Body 1, 2, a část bodu 3, 4 ze zadání.

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

Vedoucí práce: **Szőke Igor, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.
Datum zadání: 1.11.2024
Termín pro odevzdání: 14.5.2025
Datum schválení: 12.11.2024

Abstrakt

Cílem této bakalářské práce je vytvořit platformu pro procvičování matematiky formou drillu, která umožňuje zadávání výsledků hlasem. Práce popisuje jednotlivé fáze vývoje – od průzkumu existujících řešení, přes specifikaci požadavků uživatelů a návrh řešení, až po samotnou implementaci platformy. Dále jsou představeny technologie využité v aplikaci, zejména systémy pro automatický přepis řeči a velké jazykové modely, které jsou v řešení využity k vyhodnocování odpovědí hlasem. Po dokončení implementace byla platforma testována cílovými uživateli, což umožnilo zhodnotit její uživatelskou přívětivost. Během testování byla rovněž shromážděna data, ze kterých vznikla datová sada pro porovnání vybraných služeb pro přepis řeči.

Abstract

The aim of this bachelor's thesis is to create a platform for practicing mathematics through drill exercises, which allows users to enter answers by voice. It details the complete development process – from analyzing existing solutions and specifying user requirements to designing and implementing the final product. Furthermore, it presents the technologies used in the application, particularly automatic speech recognition systems and large language models, which are used to interpret and evaluate answers entered by voice. Following implementation, the platform was tested by target users to assess usability and gather data. This data supported the creation of a dataset for comparing selected speech-to-text services.

Klíčová slova

UI, uživatelské rozhraní, UX, uživatelská zkušenost, ASR, Django, Vue, procvičování matematiky, Azure, LLM, Gemini, webová aplikace.

Keywords

UI, user interface, UX, user experience, ASR, Django, Vue, practicing math, Azure, LLM, Gemini, web application.

Citace

HORUT, Dominik. *Procvičování matematiky ZŠ s možností hlasového vkládání řešení příkladů*. Brno, 2025. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Igor Szóke, Ph.D.

Procvičování matematiky ZŠ s možností hlasového vkládání řešení příkladů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Igora Szökeho, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....

Dominik Horut

12. května 2025

Poděkování

Tímto bych rád poděkoval panu Ing. Igoru Szökemu, Ph.D., za odborné vedení mé bakalářské práce, cenné rady a veškeré konzultace. Poděkování patří také všem testujícím uživatelům za poskytnutí dat a zpětné vazby. V neposlední řadě děkuji mým rodičům za jejich neustálou podporu po celou dobu studia.

Obsah

1	Úvod	2
2	Průzkum a požadavky uživatelů	4
2.1	Průzkum existujících řešení	4
2.2	Uživatelský průzkum	10
3	Uživatelská rozhraní a uživatelská zkušenost	12
3.1	Pohled uživatele	12
3.2	Vzhled uživatelského rozhraní	13
3.3	Chování uživatele	13
3.4	Testování uživatelského rozhraní	14
4	Návrh	16
4.1	Specifikace požadavků	16
4.2	Návrh datového modelu	17
4.3	Návrh uživatelského rozhraní	18
4.4	Výběr technologií	21
5	Použité technologie pro zadávání výsledků hlasem	24
5.1	Automatický přepis řeči	24
5.2	Velké jazykové modely	27
6	Implementace a architektura	30
6.1	Architektura aplikace a nasazení	30
6.2	Část serveru	31
6.3	Část klienta	36
7	Testování	43
7.1	Cíle testování	43
7.2	Návrh testování	43
7.3	Vyhodnocení testování	45
7.4	Vyhodnocení zvolených řešení pro zadávání výsledků hlasem	48
8	Závěr	51
	Literatura	53
A	Obsah paměťového média	56

Kapitola 1

Úvod

Matematika je často označována jako královna přírodních věd, jelikož právě skrze ni ostatní přírodní vědy formulují své modely a teorie. Dá se tak říci, že se nachází všude kolem nás a je proto důležité, aby každý z nás porozuměl alespoň základním okruhům této vědy.

Učení se matematice si lze představit jako stavbu domu – je důležité mít pevné a kvalitní základy, protože právě na nich bude dům stát a postupně růst. Stejně tak v matematice je proto podstatné, abychom si dokonale osvojili fundamenty a díky jejich znalosti pak byli schopni porozumět složitějším a méně triviálním tématům.

Pochopení a naučení se těmto základům je náplní výuky matematiky na základní škole. Právě tam se žáci seznamují se základními matematickými operacemi, jejich vlastnostmi a také se tyto poznatky učí aplikovat v praxi v podobě řešení slovních úloh vycházejících z reálných situací. Na těchto získaných znalostech budou následně stavět při studiu matematiky na střední a případně vysoké škole.

Pokud si ovšem tyto základy žák nevštípí hned na počátku, je následné doučování se nedostatečně či chybně naučené látky náročnou a zdoluhavou prací.

Nejlepší cesta, jak tyto základy pevně a hlavně důkladně uložit v paměti žáka, je počítat příklady z problematického okruhu do té doby, dokud si postup řešení daného problému, respektive typu příkladů, neautomatizuje. Na internetu se v dnešní době vyskytuje spousta webových stránek, které se na doučování matematiky zaměřují. Většina z nich se ale na procvičování příkladů dívá hlavně z vysvětlovacího hlediska, což se často odráží na zdoluhavosti procvičování.

Téma této bakalářské práce se proto zaměřuje na tvorbu webové aplikace na procvičování matematiky na úrovni základní školy, ve které si uživatel bude látku procvičovat formou drilu. Cílem používání této aplikace tedy bude spočítat co nejvíce příkladů za co nejmenší časový úsek. Aby uživatel při počítání neztrácel čas zadáváním výsledků příkladů do aplikace ručně, bude mít možnost zadávat výsledky hlasem. Tím bude procvičování efektivnější a zejména u žáků prvního stupně nebude docházet ke zbytečným časovým prodávám spojených s používáním klávesnice. Celkově se tedy aplikace snaží brát co největší ohled na cílovou skupinu uživatelů, kteří, vzhledem ke svému věku, nemusí mít bohaté zkušenosti s používáním počítače. Řešení také bude obsahovat rozhraní umožňující tvorbu a editaci obsahu aplikace, tedy příkladů.

Text práce je rozdělen do osmi kapitol, které popisují použité technologie a jednotlivé fáze vývoje aplikace. V kapitole 2 je nejprve provedena analýza existujících řešení s podobným zaměřením a průzkum mezi cílovými uživateli. Jeho cílem bylo identifikovat nedostatky těchto řešení a zjistit požadavky na výslednou aplikaci. Kapitola 3 se věnuje teoretickým principům návrhu uživatelských rozhraní.

Na základě získaných poznatků pak vznikl návrh aplikace, který je popsán v kapitole 4. Kapitola 5 představuje technologie použité pro zadávání výsledků hlasem – automatický přepis řeči a velké jazykové modely. Tato kapitola se věnuje jak principu fungování daných technologií, tak zejména jejich vlastnostem důležitým pro potřeby vyvíjené aplikace. V kapitole 6 je popsána samotná implementace aplikace. Zdrojový kód řešení je volně dostupný jako open-source¹ a aplikace je také nasazena².

Následující kapitola 7 se věnuje testování funkčnosti a kvality aplikace. Součástí je i srovnání různých existujících systémů pro přepis řeči na základě dat od testujících uživatelů. Pomocí těchto dat byl také vyhodnocen vliv použití velkého jazykového modelu pro zvýšení přesnosti vyhodnocení odpovědí. Práce je zakončena shrnutím v kapitole 8, kde jsou navrženy také možné směry dalšího vývoje.

¹Zdrojový kód je volně dostupný na platformě GitHub na <https://github.com/PanHorut/BP>

²Aplikace je nasazena na <https://drillovacka.applikuapp.com/>

Kapitola 2

Průzkum a požadavky uživatelů

Tato kapitola je věnována průzkumu již existujících řešení, kde je každé řešení popsáno a shrnuty jsou pak jeho pozitiva a negativa. Následuje sekce pojednávající o provedeném uživatelském průzkumu. Obsahuje popis cílové skupiny, výběr respondentů a průběh samotného průzkumu. Získané poznatky a připomínky jsou pak shrnuty a dále je popsáno, jak budou reflektovány ve vytvářené aplikaci.

2.1 Průzkum existujících řešení

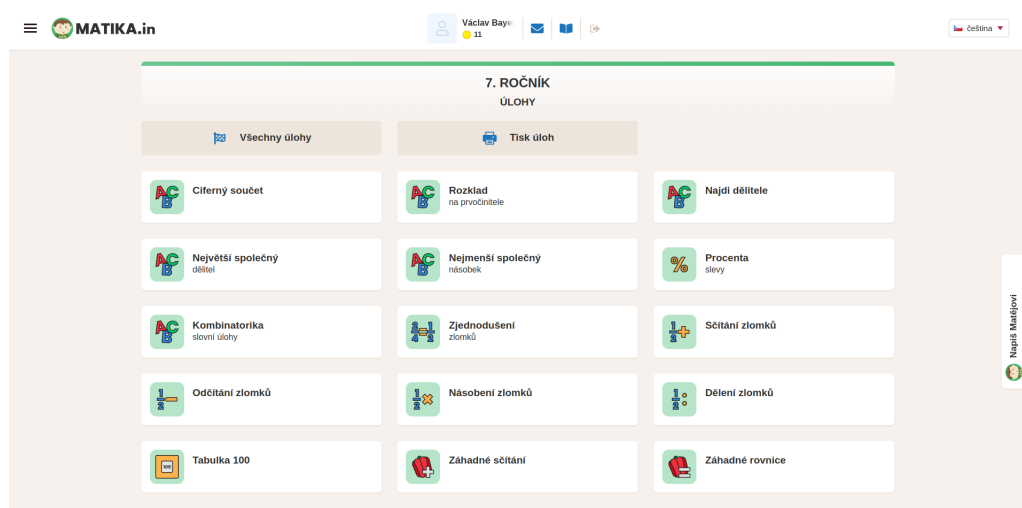
Průzkum bylo nutné rozdělit na dvě části. Protože se výuka matematiky v každé zemi člení jiným způsobem, byla první část zaměřena na české stránky, které vycházejí z kurikula matematiky vytvořeného v rámci Rámcového vzdělávacího programu pro základní vzdělávání¹. V této části byla u jednotlivých řešení věnována pozornost zejména členění jednotlivých témat a jejich přehlednost. Bylo také zkoumáno, jakým způsobem lze jednotlivá témata procvičovat a zda obsahují možnost počítat příklady formou drilu.

2.1.1 Matika.in

První zkoumanou platformou byla Matika.in². Jedná se o službu, která umožňuje procvičovat matematiku hravým způsobem. Kromě standartního počítání příkladů tak nabízí například cvičení formou doplňování pyramid či hraní jednoduchých her s matematickým tématem. Zároveň také žák během procvičování získává virtuální měnu, takzvané „zlatáky“. Aplikovány jsou tak prvky gamifikace, tedy využití her, herních principů v mimoherním kontextu pro vytvoření podobného prožitku, jako je prožitek získaný hraním her [27]. Kromě žáků mohou platformu využít také rodiče a učitelé. Ti tak skrze tuto stránku mohou svým dětem či žákům zadávat úkoly a testy a sledovat tak jejich výsledky a pokrok. Jednotlivá témata jsou na hlavní stránce rozdělena podle ročníků, díky čemuž se uživatel může rychle zorientovat, ve které sekci bude požadované cvičení hledat. Sekce pro jednotlivé ročníky pak nabízejí uživateli výčet všech cvičení adekvátních pro daný ročník (viz obrázek 2.1).

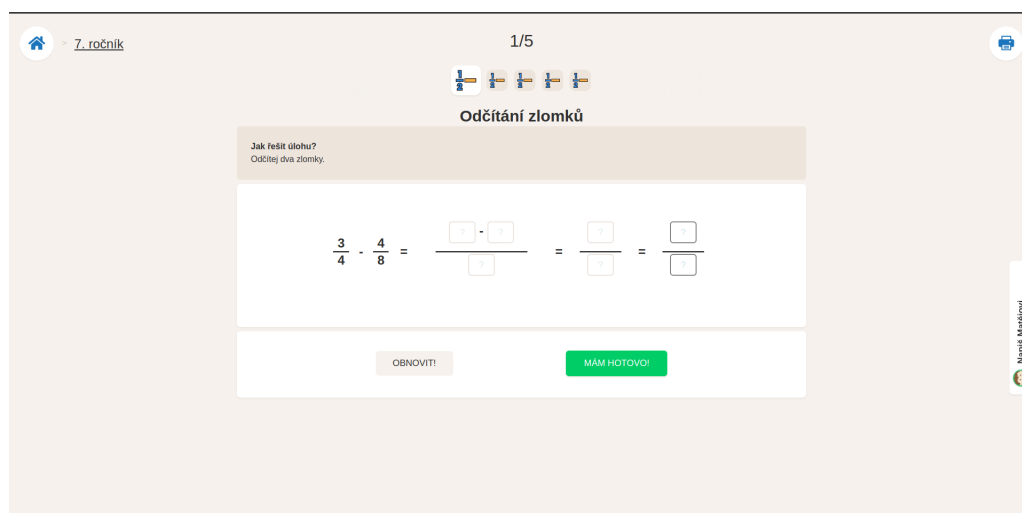
¹Platná verze RVP ZV – https://www.edu.cz/wp-content/uploads/2023/07/RVP_ZV_2023_zmeny.pdf

²Matika.in – <https://www.matika.in/cs/>



Obrázek 2.1: Jednotlivá cvičení pro 7. ročník na platformě Matika.in.

Samotné počítání příkladů je uživatelsky velice přívětivé (například u složitějších příkladů nabízí možnost celé řešení rozepsat, viz obrázek 2.2). Cvičení probíhá jako série pěti příkladů, což je z hlediska efektivity procvičování zdlouhavé a chybí tak možnost počítat vlastní množství příkladů bez přerušení. Na konci každého cvičení se nachází sumarizace celého cvičení včetně stráveného času.



Obrázek 2.2: Počítání příkladů na platformě Matika.in.

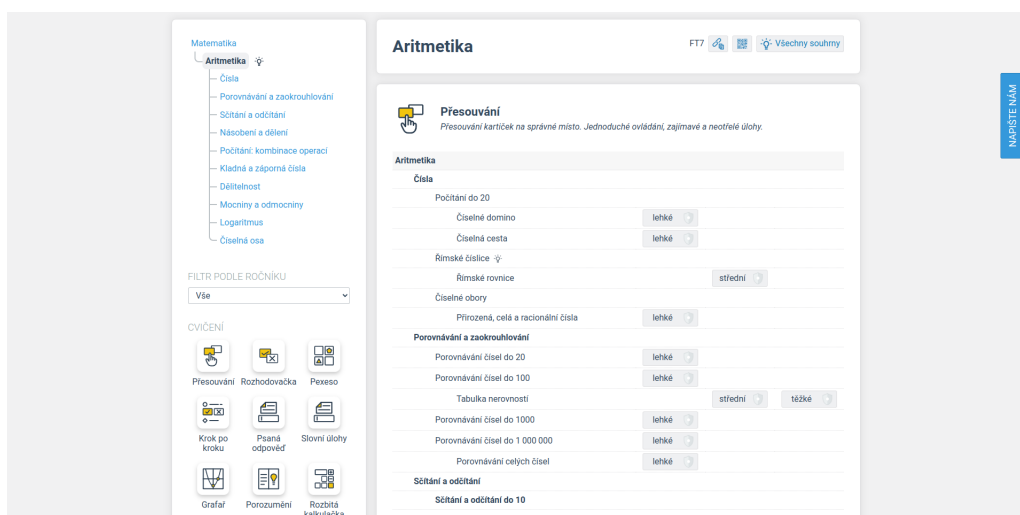
Celkově lze tuto platformu hodnotit jako kvalitní a uživatelsky přívětivou. Jako vhodná se jeví zejména pro žáky prvního stupně s přihlédnutím na využití her při procvičování. Co se ale drilování příkladů týče, obsahuje prvky, které celý proces zpomalují a není tak nejvhodnější pro tento záměr.

2.1.2 Umimematiku

Jedná se o sekci³ výukového portálu Umimeto.org, který se kromě matematiky zaměřuje i na výuku ostatních předmětů vyučovaných na základních a středních školách. Stejně jako řešení 2.1.1, se i tato služba zaměřuje na procvičování matematiky více nestandardní formou. K dispozici tak jsou opět kromě klasického počítání, kdy žák spočítá a následně zapíše výsledek, cvičení na doplňování postupu, rozhodování mezi několika výsledky a další nestandardní přístupy.

Vyhledávat jednotlivá témata může uživatel jak podle tematických celků, tak podle ročníků. Díky tomu je možné požadované téma bez větších problémů nalézt. Na stránce se také nachází sekce *Na rychlost*, kde uživatel může vybraná témata procvičovat na čas a to formou závodů proti počítači nebo jiným uživatelům. Lze to tak považovat za jistou zábavnější alternativu drilování příkladů. Zadávání výsledků je ale implementováno pouze formou rozhodování mezi dvěma výsledky, což svádí k tipování.

Tuto platformu lze tedy hodnotit jako kvalitní nástroj pro procvičování matematiky, který nabízí velké množství různorodých cvičení na všechna témata vyučovaná na základní i střední škole. Kromě klasického procvičování nabízí i počítání na rychlost, což lze považovat za jakousi formu drilu příkladů.



Obrázek 2.3: Nabídka procvičování aritmetiky na platformě Umimematiku.

Druhá část se zaměřila na zahraniční stránky. Jak již bylo zmíněno, zahraniční kurikula matematiky se v určitých aspektech od toho českého odlišují [14] a z tohoto důvodu nebyla přehlednosti a členění kapitol věnována tak velká pozornost jako u předchozích řešení. O to více se ale tato část průzkumu věnovala právě možnosti rychlého a efektivního procvičování a také zadávání výsledků hlasem, které v žádném existujícím českém řešení nebylo podporováno.

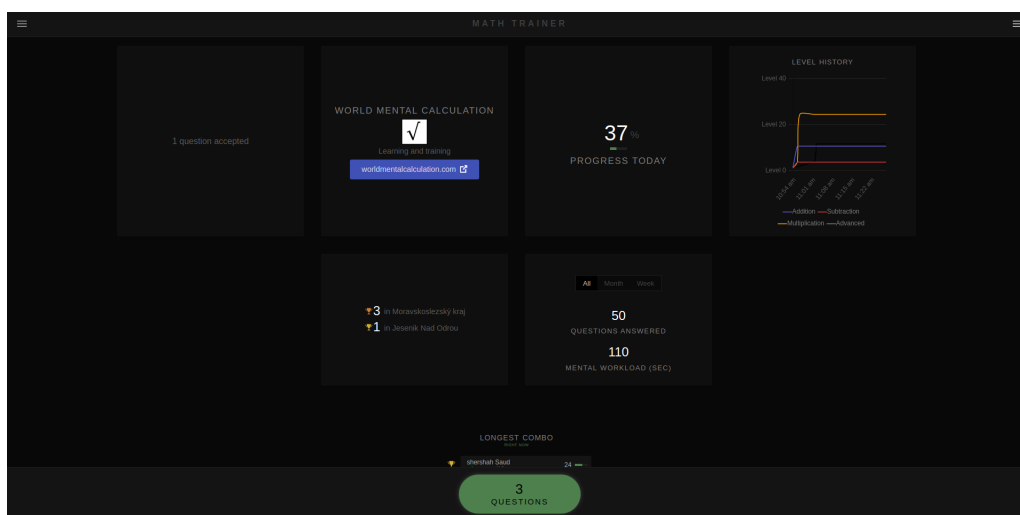
³Umimematiku – <https://www.umimematiku.cz/>

2.1.3 Mathtrainer

Do průzkumu existujících řešení byla tato webová aplikace⁴ vybrána z důvodu, že se zaměřuje na procvičování mentální matematiky [7], což lze považovat za jistou formu drilu. Příklady na sčítání, odčítání, násobení a dělení rozděluje na sto obtížnostních úrovní. Pro každý příklad je úroveň určena na základě mediánu času, který uživatelé při řešení stráví. Uživatelův pokrok se tak odvíjí na základě času, po který příklad počítal a také jeho úrovně.

Aplikace nabízí čisté a přehledné rozhraní, kde během počítání není uživatel ničím vyrušován. Typy příkladů, na kterých chce trénovat, si může navolit. V nabídce je kromě základních aritmetických operací také volba pokročilejších příkladů, které se zaměřují na procvičování mocnin a odmocnin. Přímo na hlavní stránce jsou uživateli zobrazovány jeho statistiky v podobě grafů, ve kterých je zaznamenáno, jaké úrovně dosáhl u jednotlivých aritmetických operací.

Celkově lze tedy aplikaci považovat za vhodnou inspiraci, co se uživatelského rozhraní týče, zejména v oblasti počítání příkladů, kde tato sekce neobsahuje žádné nadbytečné prvky, které by uživatele během drilu vyrušovaly.



Obrázek 2.4: Hlavní stránka webové aplikace Mathtrainer.

2.1.4 VoiceMath

Ze všech řešení zmíněných v této kapitole se jedná o jediné, které podporuje zadávání výsledků hlasem. Důvodem je neexistence jiných webových aplikací, které by tuto formu vstupu podporovaly. Jde o vcelku nové řešení⁵, ve kterém zjevně není vše dokončeno (např. nefunkční tlačítka do jistých částí aplikace). Jediná možnost jak procvičovat je formou kvízu o deseti příkladech, ve kterém uživatel řeší náhodně vygenerované příklady zaměřující se pouze na jednoduchou aritmetiku. Uživatel si tak v podstatě nemůže vybrat co by si přál procvičovat.

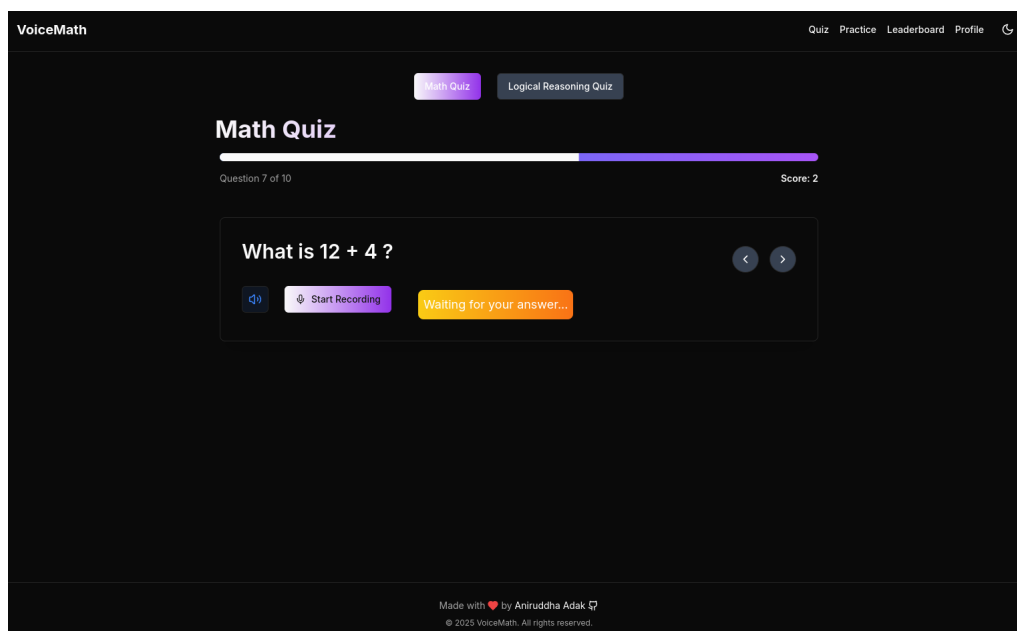
Zadat výsledek lze pouze hlasem a v anglickém jazyce. Chybí také alternativa zadání odpovědi pomocí klávesnice. Vyhodnocení odpovědi je dostatečně rychlé, ovšem nešťastnou formou notifikace (takzvaného *toastu*), která není dobře viditelná a uživatel tak nemusí

⁴Mathtrainer – <https://mathtrainer.ai/>

⁵VoiceMath – <https://voicemath.vercel.app/>

postřehnout, zda odpověděl správně či špatně. V akusticky rušném prostředí pak většinou uživatelská odpověď není zaznamenána a vyhodnocena.

Celkově se jedná o webovou aplikaci, která by v případě vylepšení a dotažení měla velký potenciál. Ve stavu, v jakém se ale aktuálně nachází, se jedná o nedostatečné řešení.



Obrázek 2.5: Procvičování ve webové aplikaci VoiceMath.

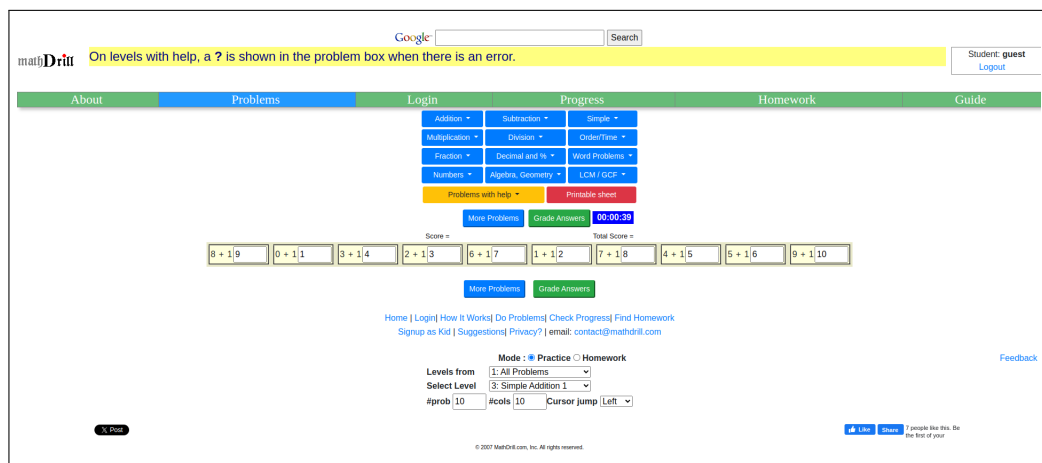
2.1.5 Mathdrill

Tato webová aplikace⁶, která se řadí ke starším řešením, nabízí v porovnání s předešlou webovou aplikací Mathtrainer.ai mnohem širší škálu témat k procvičení, jako například počítání se zlomky nebo základní algebru. Příklady jsou roztrženy na hlavní kategorie (např. číselné operace nebo zlomky), které se následně detailně dělí na podkategorie (např. násobení dvěma). Díky tomu si uživatel může velmi konkrétně vybrat, co chce procvičit.

Vhodné členění témat ale negativně vyvažuje uživatelské rozhraní aplikace a to zejména zobrazování zadání příkladů. Zadáno je totiž vždy deset příkladů z vybraného tématu, které jsou ovšem zobrazeny vedle sebe, což může uživatelům, zejména dětem, dělat problémy pro svou nepřehlednost. Řešení pak také nabízí sekci zobrazující pokrok uživatele. To je ale bohužel řešeno pouze jednoduchou tabulkou, ze které nelze nijak konkrétně vyvodit, v čem se uživatel zlepšuje či naopak zhoršuje.

Závěrem lze tedy toto řešení pochválit za velmi dobrou kategorizaci příkladů a je proto žádoucí se při návrhu členění témat inspirovat právě zde. Z hlediska uživatelské přívětivosti a přehlednosti je ale toto řešení na velmi špatné úrovni a s přihlédnutím na standardy uživatelských rozhraní v dnešní době jej lze v tomto směru označit za nedostatečné.

⁶Mathdrill – <https://www.mathdrill.com/>



Obrázek 2.6: Hlavní stránka webové aplikace Mathdrill.

Shrnutí vlastností všech představených řešení je uvedeno v tabulce 2.1. U každého řešení je vyzdvížena vlastnost, ve které oproti ostatním vyniká, a zároveň zmíněn jeho hlavní nedostatek. Dále je porovnávána možnost zadávání výsledků hlasem a podpora procvičování formou drilu.

Řešení	Klady	Zápory	Hlas	Dril
Matika.in	Uživatelsky přívětivé rozhraní pro počítání příkladů, rychlé nalezení cvičení	Zdlouhavost procvičování	✗	✗
Umimematiku	Přehledné rozdělení témat	Procvičovat formou drilu lze pouze rozhodováním mezi správnými výsledky	✗	✓
Mathtrainer	Čisté a přehledné rozhraní, statistiky o pokroku uživatele	–	✗	✓
VoiceMath	–	Nedokončené řešení, zadat výsledek lze pouze hlasem	✓	✓
Mathdrill	Kategorizace příkladů	Uživatelsky nepřívětivé a nepřehledné rozhraní	✗	✗

Tabulka 2.1: Porovnání webových aplikací pro procvičování matematiky na základě svých výhod, nedostatků a podpory zadávání výsledků hlasem a procvičování formou drilu.

V rámci průzkumu existujících řešení bylo představeno pět řešení – dvě česká a tři zahraniční. U každého z nich pak byly popsány klady a zápory. Každé z těchto řešení disponovalo jinými kvalitami, ovšem u žádného z nich nebyla nalezena kombinace širšího spektra témat a možnosti počítat formou drilu. Zadávat výsledky hlasem pak bylo možné pouze u jediného z nich. Proto bylo potřeba zaměřit se na aspekty, ve kterých jednotlivá řešení vynikala oproti ostatním. Z hlediska uživatelské přívětivosti se jako vhodná ukázala webová aplikace Matika.in, která respektuje, že uživatelskou základnu budou tvořit zejména mladší uživatelé a její uživatelské rozhraní je proto přehledné a snadno pochopitelné. Hodna inspirace je také webová aplikace Mathtrainer, a to zejména z hlediska jednoduchosti a minimalismu uživatelského rozhraní pro počítání příkladů, kde uživatel není ničím rušen a může se tak soustředit pouze na počítání. Pro jednoduché a rychlé vyhledání tématu k procvičení ve výsledné aplikaci bude vhodné navrhnout podobnou strukturu témat, jaká je použita v řešení Mathdrill.

2.2 Uživatelský průzkum

Před samotným návrhem aplikace je vždy vhodné vytyčit si, kdo je cílovou skupinou budoucího řešení a jaké jsou její požadavky. Cílová skupina jsou pochopitelně žáci prvního a druhého stupně základní školy. Je ale nutné brát ohled na to, že mezi jednotlivými uživateli této skupiny jsou věkové, ale hlavně vývojové rozdíly. Žák prvního ročníku bude mít jiné požadavky, než žák v posledním ročníku. Proto bylo žádoucí vybrat do uživatelského průzkumu co největší škálu ročníků a na základě jejich požadavků vytvořit kompromis, aby aplikace vyhovovala všem. Zvolena tak byla pětice respondentů z mého okolí, ve které byl zastoupen první, třetí, šestý, osmý a devátý ročník ZŠ. Díky tomu tak bylo možné zjistit, jak se požadavky v závislosti na věku liší nebo naopak shodují.

Průzkum byl proveden formou rozhovoru s jednotlivými respondenty. Na začátku byl respondent dotázán, jakým způsobem procvičuje matematiku, například před písemkou. Tři z pěti respondentů používají k tomuto účelu nějakou výukovou platformu na internetu. V případě respondentů z prvního a třetího ročníku probíhá procvičování počítání v pracovním sešitě. Z toho vyplývá, že žáci nižších ročníků pravděpodobně ocení možnost vytisknout si příklady a počítat přímo na papíře.

Zbylí tři respondenti byli usazeni k počítači s otevřeným webovým prohlížečem a byli požádáni, aby otevřeli webovou stránku, na které většinou procvičují. Ve všech případech si respondenti otevřeli výukový portál Umimematiku.cz, jehož popis se nachází v kapitole průzkumu existujících řešení 2.1.2. Následně byli vybídnuti k vyhledání sekce s procvičováním látky, kterou právě v hodinách matematiky probírají.

To se u dvou respondentů ukázalo jako velmi problematické, jelikož i přes to, že po chvíli objevili sekci *Témata* a v ní příslušnou kategorii, do které látka náleží, nemohli následně nalézt konkrétní procvičování. To dle respondentů způsobilo příliš velké množství různých typů cvičení. Nakonec však po delší době oba našli hledané téma k procvičení. V případě respondenta z devátého ročníku proběhlo vyhledání tématu rychle, a to z důvodu, že jako jediný použil k vyhledání tématu vyhledávací panel v horní části stránky. Je proto vhodné tento způsob vyhledávání ve vytvářené aplikaci podporovat. K výběru měli několik různých procvičování, kde se každé z nich lišilo různým způsobem zadávání výsledků. Nejprve tak respondenti počítali příklady formou rozhodování mezi dvěma výsledky. To se ale jevilo jako nevhodné, protože místo počítání občas tipovali výsledky. Následovala tak cvičení s psanou odpovědí, kde respondenti neměli výhrady. Jediná připomínka byla směřována na velikost textu příkladů.

Na závěr byli uživatelé dotázáni, aby popsali, zda byli s procvičováním spokojeni a zda by se takto dokázali připravit na písemku z dané látky. Zde byly odpovědi různorodé. Jeden z respondentů ocenil výukovou sekci portálu pro dovysvětlení látky v případě nejasností. Zbylí dva měli totožný názor, tedy že by se na písemku touto formou naučili, ale samotné zdlouhavé vyhledání procvičování na dané stránce by jim zabralo čas, který mohli strávit počítáním.

Z průzkumu tedy vyplývají tyto uživatelské požadavky:

- Mladší žáci (1. – 3. ročník) pravděpodobně ocení možnost vytisknout si příklady a počítat na papíře.
- Výběr kategorií příkladů by měl být na hlavní stránce přehlednější a více viditelný.
- Podporováno by mělo být vyhledání kategorie příkladů pomocí vyhledávacího panelu.

- Procvičování formou výběru správné odpovědi není v případě matematiky nejvhodnější.
- Zadání příkladů by mělo být dostatečně velké.

Ve vytvářené aplikaci je tak žádoucí tyto požadavky zohlednit a vytvořit aplikaci, která bude užitečná nejen žákům, kteří matematiku procvičují na různých výukových platformách, ale i pro žáky, kteří volí cestu počítání na papíru a existující řešení tak pro ně nejsou atraktivní. Dále je důležité, aby uživatelé neměli problém rychle vyhledat téma, které si chtějí procvičit. Jako vhodná forma zadávání odpovědi se ukázala psaná odpověď.

Kapitola 3

Uživatelská rozhraní a uživatelská zkušenost

Obsah této kapitoly se věnuje principům, které je dobré dodržovat při návrhu uživatelských rozhraní. Nahlíží na jejich tvorbu z pohledu uživatele, který může rozhraní vnímat zcela odlišně, než jak zamýšlel vývojář. To pak vede k horší uživatelské zkušenosti, použitelnosti a celkovému dojmu z aplikace. Následně jsou zmíněny důležité aspekty, které dopomáhají k lepší použitelnosti. Protože je potřeba informace o tom, co si uživatel o rozhraní myslí, nějakým způsobem získat, pojednává poslední část o uživatelském testování a různých přístupech k němu.

3.1 Pohled uživatele

Při tvorbě uživatelského rozhraní si vývojář či designér může myslet, že uživatel bude při používání důkladně zkoumat každou jeho část. V realitě se ale dle Steva Kruga a jeho knihy *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability* [11] jedná o pravý opak. Uživatelé totiž věnují jen letmý pohled na každou novou stránku a kliknou na první odkaz či tlačítko, které je zaujme.

To se podle něj děje skrze skutečnost, že uživatel rozhraní nečte, ale pouze skenuje. Způsobuje to například to, že uživatel přichází na webovou stránku či do aplikace s nějakým cílem, který chce splnit co nejrychleji. Jako další důvod uvádí, že uživatel sám ví, že nepotřebuje číst vše, co je na stránce vyobrazeno pro splnění svého cíle.

Z toho plyne rada, že pokud je to jen trochu možné, měla by být webová stránka na první pohled jasná a srozumitelná. Uživatel by měl být schopen pochopit, co stránka nabízí a jak ji používat, aniž by musel vynakládat jakékoli úsilí na přemýšlení o tom.

Pro příklad lze uvést jednoduché tlačítko. To, jaký text tlačítko má a jestli opravdu vypadá, že na něj lze kliknout, se na první pohled může zdát jako nepodstatné – uživatel maximálně ztratí krátkou chvíli přemýšlením. Každá taková chvíle ale zvyšuje jeho mentální zátěž a odvádí pozornost od úkolu, na který se právě soustředí. Tato vyrušení mohou být nepatrná, ale postupně se sčítají, zvláště pokud jde o činnost, kterou dělá neustále, například právě rozhodování, na co kliknout.

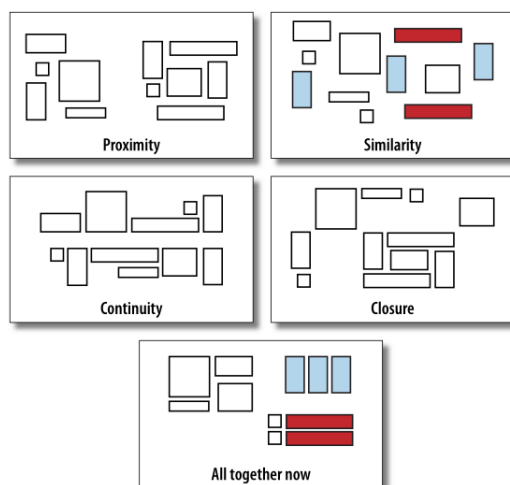
Samozřejmě ne vše lze udělat tak, aby to bylo na první pohled zřejmé. Pak je potřeba aby to bylo alespoň srozumitelné. Uživatel tak sice musí strávit trochu času přemýšlením o stránce, ale opravdu jen trochu [11]. K dosažení takového cíle dopomáhá celkový vzhled stránky a její rozvržení. Proto se následující část bude věnovat právě těmto tématům.

3.2 Vzhled uživatelského rozhraní

Důvod, proč je i tato část návrhu důležitá a přispívá k celkové použitelnosti aplikace, popisuje Jenifer Tidwell ve své knize *Designing Interfaces: Patterns for Effective Interaction* [22]. Pokud vzhled na uživatele dobře zapůsobí, bude uživatel více tolerantní vůči menším nedostatkům a obtížím, se kterými se na stránce může setkat. Zároveň bude více kreativní při hledání řešení těchto obtíží.

Dále je důležité i rozvržení stránky. Teorie o seskupování a zarovnání prvků vychází z gestaltismu, tedy z tvarové psychologie, která byla psychology zkoumána ve 20. století. Definuje několik důležitých principů o rozvržení prvků, které jsou jakýmsi způsobem přirozené lidskému vnímání. Pro rozvržení rozhraní jsou dle Tidwell důležité tyto čtyři [22]:

- **Proximita** – Prvky umístěné blízko sebe bude uživatel vnímat tak, že spolu souvisí.
- **Podobnost** – Pokud mají dva prvky stejný tvar, barvu, text či orientaci, uživatel je opět bude vnímat jako spolu související.
- **Kontinuita** – Uživatel vnímá prvky, které jsou spolu nějakým způsobem zarovnané, jako jednu spojitou linii či křivku.
- **Uzavřenost** – Uživatel má sklon vidět jednoduché uzavřené tvary, jako jsou obdélníky či mezery ve tvaru bloků i přes to, že nejsou nijak explicitně nakresleny. Seskupené prvky často vytvářejí dojem uzavřených tvarů.



Obrázek 3.1: Principy teorie o seskupování, převzato z [22].

3.3 Chování uživatele

Při tvorbě rozhraní není důležité jen to, jak uživatel rozhraní vnímá, ale také to, jak se při jeho používání typicky chová a co od něj očekává. Proto následující část popisuje pro tuto práci důležité šablony chování, které uživatelé při používání rozhraní mají.

3.3.1 Změny uprostřed činnosti

Stává se, že během vykonávání nějaké činnosti v rozhraní uživatel z ničeho nic změní svůj původní záměr, který chtěl v rozhraní uskutečnit a začne se věnovat jiné aktivitě v rozhraní. Jeho cíle se totiž během používání rozhraní mohou měnit. Tento fakt by měl být proto při návrhu respektován. Uživateli by tak mělo být dovoleno přerušit dosavadní práci, kterou v rozhraní dělal a umožnit mu vrátit se k ní později [22].

3.3.2 Zvyklosti

Pokud uživatelé v rozhraní provádějí nějakou akci velmi často, může se z provádění této akce stát zvyk. V takovémto případě již uživatel provádí takovou akci podvědomě. Pro příklad lze uvést klávesovou zkratku `Ctrl + S` pro uložení dokumentu. Díky tomu uživatelé používají rozhraní efektivněji. To má ale i svá úskalí. Může totiž nastat situace, kdy uživatel jistou akci automaticky použije, ale rozhraní ji nebude podporovat nebo v horším případě bude mít provedení akce zcela jiný výsledek, než na který je uživatel zvyklý. Při vytváření rozhraní je proto důležité brát existenci jistých zvyklostí na zřetel [22].

3.3.3 Používání klávesnice

Někteří uživatelé preferují při ovládání rozhraní pracovat pouze s klávesnicí a myš použít pouze v nutných případech. Použitelné rozhraní by mělo vyjít vstříc i těmto uživatelům a to zejména pokud se jedná o aplikaci, ve které se předpokládá časté zadávání vstupu uživatelem. Proto by rozhraní mělo nabízet použití zkratk, pohyb po vstupových elementech stránky za pomoci šipek nebo potvrzování akcí klávesou `Enter` [22].

3.4 Testování uživatelského rozhraní

Testování je důležitým pilířem při vytváření rozhraní. Jen díky němu lze zjistit, zda je navržené rozhraní opravdu použitelné a uživatelsky přívětivé. Návrhář totiž může přemýšlet o rozhraní jinak, než uživatel a díky testování si tuto skutečnost může uvědomit [11]. Testování lze v základu provést dvěma způsoby – dotazníkem či rozhovorem. Proto budou oba tyto přístupy vysvětleny a popsány budou také jejich klady a zápory.

3.4.1 Dotazník

Testování formou dotazníku spočívá ve vytvoření sady otázek zabývajících se rozhraním. Pro získání kvalitních výsledků je důležité otázky dobře konstruovat a správně formulovat. Obecně je vhodné používat tuto formu testování v případě, kdy je potřeba porovnat jednotlivé odpovědi uživatelů nebo testovat na větším vzorku uživatelů.

Klady:

- Uživatelé mohou vyplnit dotazník anonymně a být tak více upřímní.
- Každý účastník obdrží stejné otázky.
- Jednoduchý a relativně levný způsob, jak získat odpovědi od velkého množství uživatelů.

Zápory:

- Nelze se dotazovat na navazující otázky v případě nejasností.
- Je jednoduché nedopatřením ovlivnit výsledky, například špatně položenou otázkou.

3.4.2 Rozhovor

Postup při testování touto formou obnáší připravení si otázek, na které uživatelé budou dotazováni a následně provedení rozhovorů. Aby byl rozhovor co nejlépe provedený, je vhodné setkat se s uživatelem osobně. Díky tomu lze uživatele při používání rozhraní důkladně pozorovat. Při pozorování je důležité držet se několika principů, jako například sledovat nejen to, co uživatel říká, ale i jeho řeč těla a výraz. Dále je potřeba sledovat nejen to, co si uživatel vybírá (například tlačítko, na které klikne), ale také jak si vybírá. Během celého testování je důležité, aby se při používání rozhraní uživateli nijak neradilo a nepomáhalo, tím totiž přijdeme o jeho přirozené chování. Během celého rozhovoru je vhodné psát si poznámky, aby bylo možné jednotlivé rozhovory porovnat a získat z nich výstup [12].

Klady:

- Je možné dotazovat se na navazující otázky, díky kterým můžeme dosáhnout kvalitní a celistvé odpovědi.
- Uživatele můžeme sledovat naživo a všimnout si tak i nonverbálních výrazů.

Zápory:

- Uživatelé budou upravovat své chování a názory a nebudou tak upřímní.
- Testovat tímto způsobem větší počet uživatelů je obtížné.

Cílem této kapitoly bylo přiblížit, jak se uživatelé během používání rozhraní chovají a uvažují. Tyto poznatky pak bude vhodné promítnout do návrhu vytvářené aplikace. Z vizuálního hlediska je důležité zejména správné rozvržení prvků v uživatelském rozhraní a jeho příjemný a nerušivý vzhled. Důležité je také respektovat zvyklosti, které mají uživatelé zažité a standardy, které od rozhraní očekávají (například možnost používat klávesnici namísto myši). Pro vytvoření kvalitního a použitelného rozhraní je důležité testovat jej uživateli. V případě této práce tak bude potřeba nejprve zjistit, co uživatelé oceňují či naopak kritizují u existujících řešení. Ideální způsob, jak takovýchto poznatků dosáhnout je rozhovor, který umožní respondentům detailněji popsat své potřeby. Po vytvoření aplikace pak bude následovat dotazníkové šetření, které umožní získat zpětnou vazbu od většího počtu respondentů.

Kapitola 4

Návrh

Téma této kapitoly je zaměřeno na návrh vytvářené aplikace. Nejprve je proto popsáno, jaké požadavky bude aplikace splňovat a také jsou definováni uživatelé, kteří ji budou používat včetně akcí, které budou moci provádět. Nastíněna je také struktura systému dovedností a návrh datového modelu. Následně je představen návrh uživatelského rozhraní doplněný o drátěné modely jednotlivých pohledů. Poslední část je pak věnována výběru technologií, pomocí kterých bude aplikace implementována.

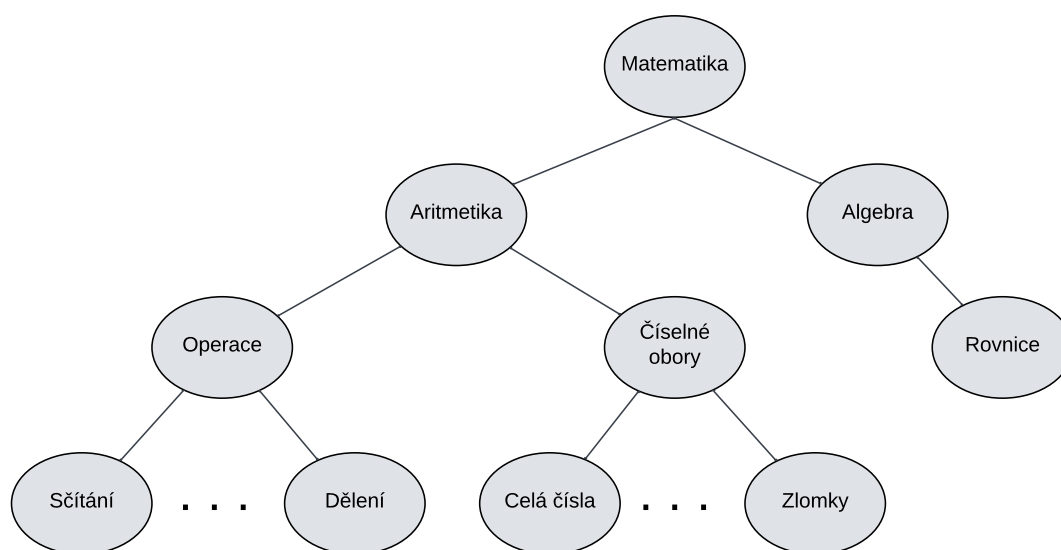
4.1 Specifikace požadavků

Na základě uživatelského průzkumu v sekci 2.2 a společných konzultací s vedoucím práce byly definovány požadavky, které budou ve výsledném řešení respektovány. Ne všechny uživatelské požadavky však budou v aplikaci promítnuty. Konkrétně se jedná o možnost tisku příkladů na papír. To z důvodu, že hlavní zaměření aplikace je zadávání výsledků hlasem, které má samo o sobě za cíl zjednodušit zadávání mladším uživatelům. Proto bylo od tohoto požadavku upuštěno.

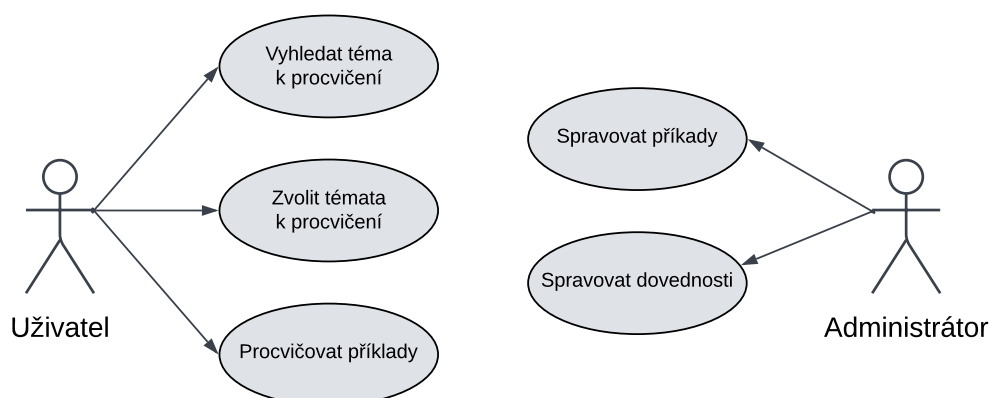
Aplikace bude uzpůsobena dvěma typům uživatelů – běžnému uživateli, který bude v aplikaci moci procvičovat příklady a také administrátorovi, který bude tvořit a spravovat příklady a dovednosti.

Z hlediska běžného uživatele musí být aplikace uživatelsky přívětivá a přehledná, aby mohl v co nejmenším možném čase od otevření aplikace začít procvičovat. Zejména se jedná o vyhledávání příslušných kategorií na hlavní stránce a následnou volbu podkategorií. Uživatel bude mít možnost vytvořit si v aplikaci účet, díky kterému bude možné sbírat data o jeho procvičování. Ta budou sbírána během počítání příkladů. Bude se například jednat o počet chyb v konkrétním cvičení nebo čas, po který příklad počítal.

Administrátor aplikace pak bude moci spravovat veškerý obsah v aplikaci. Jednat se bude zejména o správu příkladů. Ty budou sdruženy do sad, které bude mít administrátor možnost vytvářet a upravovat ve speciální sekci aplikace. Dále bude moci vytvářet či mazat dovednosti. Ty budou uspořádány ve stromové struktuře, kde kořen bude dovednost *Matematika*. Z této dovednosti se poté budou odvíjet další dovednosti jako *Aritmetika* či *Algebra*. Podrobněji je návrh struktury zobrazen na obrázku 4.1.



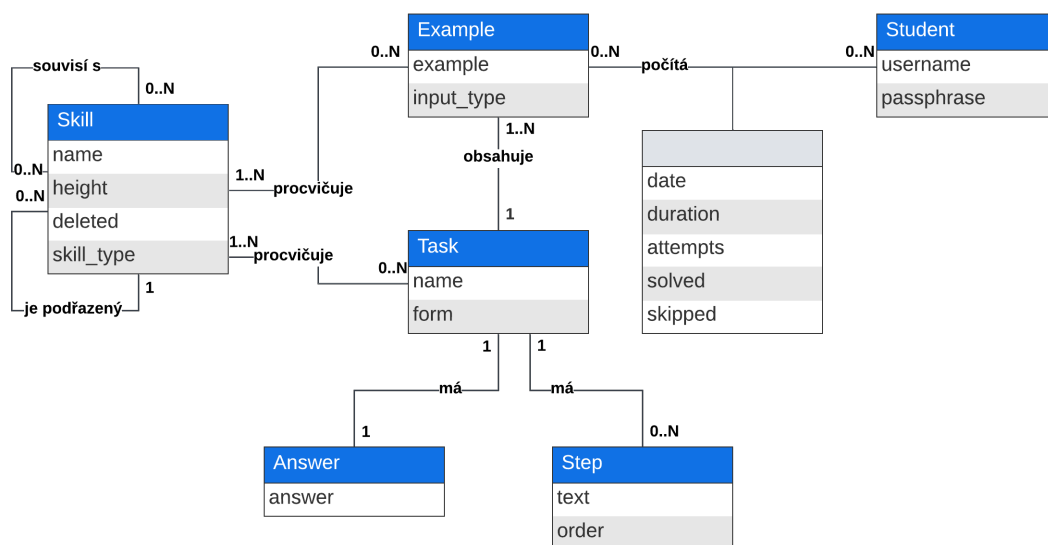
Obrázek 4.1: Návrh struktury dovedností, kde se z kořenové dovednosti *Matematika* odvíjejí její odvětví až na úroveň základních dovedností jako je *Sčítání*.



Obrázek 4.2: Diagram případů užití vytvořený na základě specifikace požadavků na výslednou aplikaci.

4.2 Návrh datového modelu

Po vytyčení typů uživatelů, kteří budou v aplikaci vystupovat, a akcí, které budou moci provádět, bylo potřeba transformovat všechny tyto požadavky a vztahy do podoby, ze které bude vycházet následná implementace datového modelu. Proto byl vytvořen jeho návrh, a to v podobě E-R (*Entity-Relationship*) diagramu, který je zobrazen na obrázku 4.3.



Obrázek 4.3: E-R diagram znázorňující návrh datového modelu aplikace. Diagram zachycuje hlavní entity a jejich vlastnosti, jako jsou uživatelé (Student), dovednosti (Skill), sady cvičení (Task), příklady (Example), odpovědi (Answer), postupy řešení (Step) a záznamy o řešení příkladů uživateli. Zobrazuje také vztahy mezi těmito entitami a jejich vzájemné vazby.

4.3 Návrh uživatelského rozhraní

Při návrhu je důležité dodržovat základní principy pro tvorbu rozhraní, ty nejdůležitější z nich pro tuto aplikaci jsou popsány v kapitole 3.

Pro návrh takzvaného drátěného modelu byla použita webová aplikace Figma¹, která se na tvorbu návrhů a prototypů specializuje. Vytvořeny byly návrhy jednotlivých částí aplikace, které byly na základě konzultací s vedoucím práce upravovány. Využity byly i interaktivní prvky umožňující přechody mezi jednotlivými okny návrhu pro lepší určení uživatelských scénářů.

4.3.1 Hlavní obrazovka

Hlavní obrazovka bude vstupním bodem do celé aplikace. Je tedy nutné, aby se na ní uživatel co nejrychleji zorientoval. Nejdůležitější částí na této obrazovce bude výčet jednotlivých dovedností k procvičení, proto bude zabírat největší část obrazovky tak, aby ji uživatel neměl problém nalézt. Nad výčtem se bude nacházet vyhledávací panel pro nalezení konkrétní podkategorie pro ještě rychlejší přístup k procvičování. V navigačním panelu, který bude umístěn v horní části obrazovky, pak bude mít uživatel možnost přejít na správu svého profilu.

¹Figma – <https://www.figma.com/>



Obrázek 4.4: Drátěný model hlavní stránky, jehož hlavní část tvoří výčet jednotlivých dovedností a jako první tak upoutá uživatelskou pozornost. Pro rychlejší nalezení požadované dovednosti k procvičení pak uživatel může použít vyhledávací panel.

4.3.2 Výběr procvičovaných témat

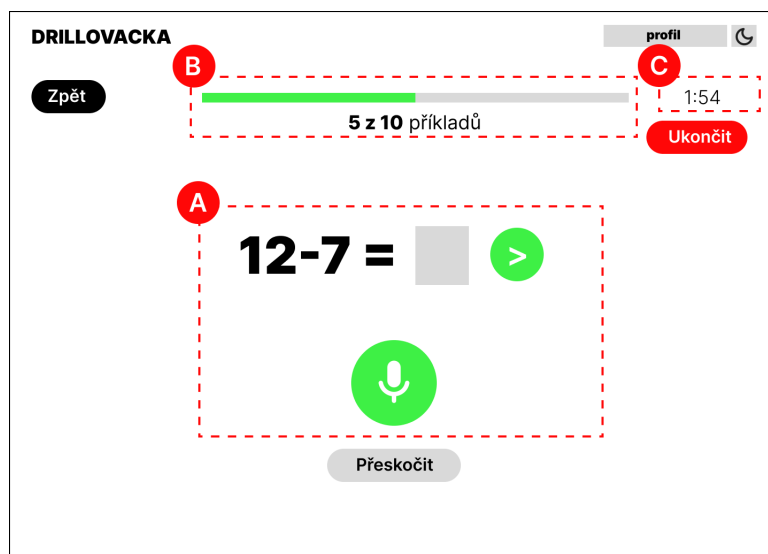
Po výběru hlavní dovednosti bude uživateli zobrazen výběr všech podkategorií, kterým je daná dovednost nadřazena. To bude řešeno formou zaškrťovacích polí, pomocí kterých si uživatel vybere, co vše chce z dané dovednosti procvičovat. Spuštění procvičování pak bude moci spustit tlačítkem *Začít procvičovat* umístěným pod výběrem kategorií.



Obrázek 4.5: Drátěný model obrazovky pro výběr konkrétních dovedností k procvičování. Výběr bude možné provést pomocí zaškrťovacích polí.

4.3.3 Počítání příkladů

Obrazovka pro počítání příkladů bude mít jednoduché a čisté rozhraní, aby uživatel při řešení nebyl ničím významně vyrušován. Velkou část obrazovky tak bude zabírat zadání příkladu a komponenta pro zadání odpovědi a to jak pomocí klávesnice, tak hlasem. Uživatel bude mít také možnost příklad přeskočit nebo celé procvičování ukončit za pomoci tlačítek se stejnými názvy. Aby měl uživatel přehled, kolik příkladů již během aktuálního procvičování spočítal a kolik mu do konce procvičování ještě zbývá, bude v horní části obrazovky umístěn indikátor postupu. Dále bude zobrazen časovač informující, jak dlouho je příklad počítán. Na tuto komponentu ale nebude použito velké vizuální zvýraznění, jelikož by uživatele při počítání spíše vyrušovalo a znervózňovalo, což zajisté není chtěný efekt.

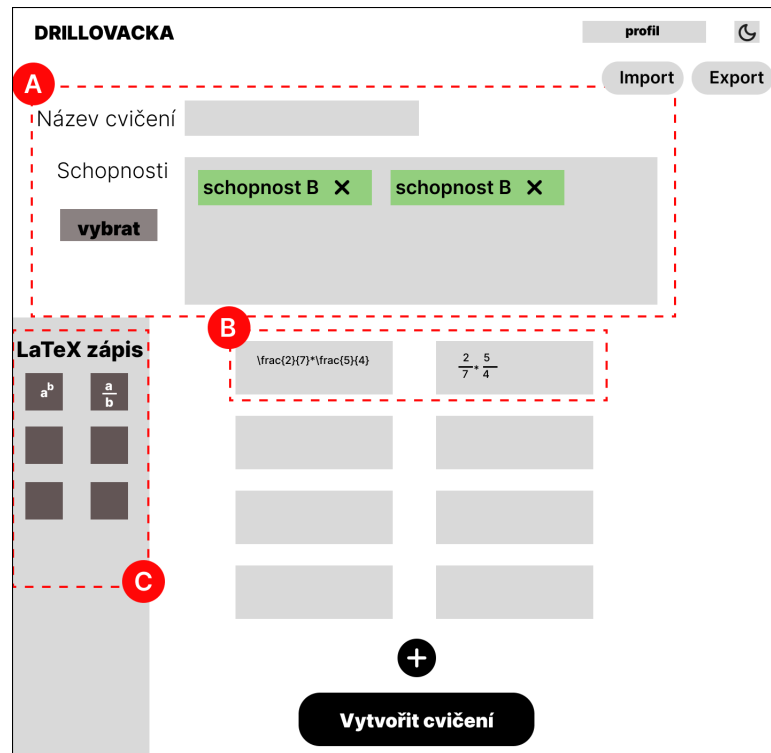


Obrázek 4.6: Drátěný model obrazovky pro počítání příkladů. **A** znázorňuje oblast se zadáním příkladu a komponentami pro odpověď pomocí klávesnice a hlasem.

B představuje indikátor postupu. **C** označuje časovač.

4.3.4 Vytváření sad příkladů

Speciální část aplikace, do které běžný uživatel nebude mít přístup, bude vytváření sad příkladů. Ta bude umožňovat jednoduché a pohodlné vytváření jednotlivých příkladů. Uživatel nejprve zvolí název a procvičené schopnosti v rámci sady a následně může začít vytvářet jednotlivé příklady. Protože je obtížné zapsat na klávesnici složitější matematické výrazy jako zlomky či mocniny tak, aby zadání bylo stále čitelné, budou se zadání příkladů zapisovat ve formátu $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$. Proto bude vedle každého vstupu pro vložení zadání a odpovědi příkladu okno vykreslující tento zápis, aby měl zadávající uživatel jistotu, že je formát správný. Je ale možné, že zadavatel nebude znát konkrétní příkazy v tomto formátu. Proto bude na levé straně obrazovky panel s různými znaky či výrazy. Po kliknutí na jeden z nich bude uživateli vložen do příslušného vytvářeného příkladu $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ zápis požadovaného výrazu, který si může následně dopravit. V pravém horním rohu se pak budou nacházet tlačítka umožňující import a export příkladů.



Obrázek 4.7: Drátěný model modulu pro vytváření příkladů. **A** znázorňuje část pro zadání názvu sady a výběru dovedností. V oblasti **B** je pak zobrazeno vstupní pole zadání příkladu v $\text{\LaTeX}$ formátu spolu s jeho vykreslením. **C** označuje panel umožňující vložení $\text{\LaTeX}$ reprezentace znaku či výrazu.

4.4 Výběr technologií

Aby bylo možné vytvořit kvalitní webovou aplikaci, je potřeba důkladně zvolit technologie, na kterých bude webová aplikace stát. Výběr technologií totiž ovlivní jak funkcionalitu, tak škálovatelnost a výkon výsledného řešení [21]. V případě správného výběru pak bude zajištěno, že aplikace bude splňovat očekávání uživatelů, její vývoj bude rychlejší a údržba snadnější [19].

Při volbě technologií je proto dobré řídit se těmito faktory [10]:

- Uživatelské potřeby.
- Velikost a komplexita projektu.
- Očekávaná flexibilita a škálovatelnost.
- Bezpečnostní požadavky.
- Očekávaná rychlost a výkon.

4.4.1 Backend

Vývoj backendu spočívá ve vytvoření serverové části aplikace, kde dochází ke správě dat a zpracování požadavků na server. Nachází se v ní tedy velká část logiky celé aplikace. V případě tohoto řešení byl na doporučení vedoucího práce zvolen framework Django, který bude nyní popsán.

Django

Jedná se o vysokoúrovňový webový framework² napsaný v jazyce Python umožňující rychlý vývoj webové aplikace. Zjednodušuje proces tvorby tím, že eliminuje potřebu vytvářet běžné funkce od základu a vývojář se tak může soustředit na tvorbu samotné aplikace bez toho, aby vše musel nejprve konfigurovat. Kód v tomto frameworku je strukturován podle osvědčených principů jako je DRY (*Don't Repeat Yourself*), který minimalizuje výskyt duplicitního kódu. Zároveň je postaven na architektuře MVC (*Model-View-Controller*), což také přispívá k přehlednosti kódu. Díky své škálovatelnosti umožňuje vytvářet jak menší, tak i větší a komplexní projekty, důkazem je použití například na platformách jako je Instagram či Disqus. Z hlediska bezpečnosti zajišťuje bezpečné zpracování uživatelských účtů a hesel pomocí šifrování [13].

4.4.2 Frontend

Tvorba klientské části aplikace, též nazývané frontend, se zaměřuje na vytvoření vizuálních a interaktivních prvků, pomocí kterých bude uživatel webovou aplikaci využívat ve svém webovém prohlížeči. Pro tvorbu této části se často používá kombinace jazyků HTML, CSS a Javascript. Pro zjednodušení a zrychlení vývoje je ale praktičtější použít framework či knihovnu jazyka Javascript, které jsou na těchto jazycích postaveny a obsahují již předpřipravené funkcionality. Následuje proto popis nejčastěji používaných.

Vue.js

Jedná se o progresivní framework³ a slouží jako lehčí alternativa k robustnějším frameworkům, jako je například React. Jeho progresivita umožňuje vkládat jej v případě potřeby do statických HTML stránek bez nutnosti sestavení, vhodný je také pro tvorbu tzv. jednostránkových aplikací (angl. *single-page application*).

Využívá tzv. deklarativní vykreslování, které rozšiřuje standardní HTML o šablonovou syntaxi. Tato syntaxe umožňuje snadno popsat výstup HTML na základě aktuálního stavu definovaného v JavaScriptu. Další klíčovou vlastností je reaktivita. Vue.js automaticky sleduje změny stavu dat v JavaScriptu a efektivně aktualizuje DOM (*Document Object Model*), jakmile ke změnám dojde. Efektivní aktualizace je umožněna použitím virtuálního DOM. Jedná se o lehkou kopii skutečného DOM, která se při změně stavu dat aktualizuje jako první. Poté se virtuální DOM porovná s reálným DOM a framework provede pouze minimální potřebné změny.

²Django – <https://www.djangoproject.com/>

³Vue.js – <https://vuejs.org/>

Souborová struktura komponenty se dělí na tři části, kde každá z nich definuje jiné aspekty [10], [23]:

- `script` obsahuje logiku komponenty, tedy JavaScript kód, který řídí chování komponenty.
- `template` definuje strukturu komponenty.
- `style` určuje styly pro komponentu, určuje vzhled jednotlivých elementů v komponentě.

React

React⁴ je aktuálně nepoužívanější knihovna jazyka Javascript a to pro svou jednoduchost při vytváření klientské části. Ta může být rozdělena do komponent, což zvyšuje znovupoužitelnost a modularitu. Stejně jako Vue.js používá deklarativní vykreslování a virtuální DOM. Z hlediska syntaxe používá syntaktické rozšíření jazyka Javascript JSX (*Javascript XML*). Toto rozšíření umožňuje kombinovat strukturu (HTML) a logiku (JavaScript) v jednom souboru, což usnadňuje práci při vytváření složitějších aplikací [10], [16].

4.4.3 Shrnutí

Díky získaným informacím o jednotlivých technologiích bylo možné určit, které z nich jsou pro toto řešení nejvhodnější. Pro stranu serveru byl vybrán framework Django a to na základě doporučení vedoucího. Pro tvorbu části klienta bude použit framework Vue.js i přes to, že se ve spoustě aspektech podobá knihovně React. Rozhodující tak byla jednoduchost a lehce strmější učicí křivka.

Jednou z hlavních funkcionalit navrženého řešení je podpora zadávání výsledků hlasem. Tato funkce bude v aplikaci zajištěna službou pro automatický přepis řeči. Po implementaci této funkcionality se však ukázalo, že pro složitější odpovědi (například ve formě zlomku) je přepis řeči často v podobě slov namísto čísel a jeho vyhodnocení se tak stává vzhledem k rozmanitosti českého jazyka obtížné a nepřesné. Z toho důvodu bude pro vyhodnocení takovýchto odpovědí použit velký jazykový model. Následující kapitola se proto zaměřuje právě na tyto technologie, popisuje jejich princip a představuje služby, které je poskytují.

⁴React – <https://react.dev/>

Kapitola 5

Použité technologie pro zadávání výsledků hlasem

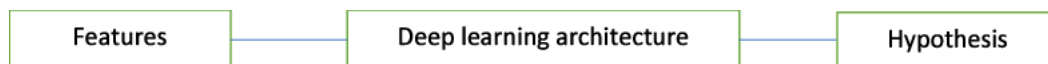
Tato kapitola se věnuje technologiím, které umožňují zadávání výsledků hlasem ve vytvořené aplikaci. Nejprve je nutné převést hlasový vstup uživatele do textové podoby, což zajišťují systémy pro automatický přepis řeči. První část kapitoly se proto zaměřuje na popis těchto systémů a představuje existující řešení. Po získání přepisu je dále nutné vyhodnotit, zda je odpověď uživatele správná. V případě složitějších formátů odpovědí je k tomuto účelu využit velký jazykový model. Druhá část kapitoly se proto věnuje právě této technologii.

5.1 Automatický přepis řeči

Automatické rozpoznávání řeči (angl. *automatic speech recognition*, dále jen ASR) je zkoumáno již od 80. let minulého století. V průběhu času se přístup k návrhu systémů umožňující přepis řeči změnil. Z tradiční vícemodulové struktury se přešlo na *End-to-End* modely [25], které celkovou architekturu zjednodušují. Představen tak bude právě tento model včetně svých výhod i omezení. Následně budou popsána existující řešení, která přepis řeči umožňují, včetně jejich srovnání na základě obecných charakteristik. Na závěr bude také představena metrika, na základě které je hodnocena přesnost modelů.

5.1.1 End-to-End model

Oproti vícemodulovému tradičnímu modelu se *End-to-End* modely skládají z jednoho spojeného modulu. Není proto nutné samostatně trénovat jednotlivé komponenty, jako jsou akustický model, jazykový model a slovník výslovnosti, ale pouze jeden společný model jako celek. Oproti klasickému přístupu tato architektura za pomoci hlubokého učení přímo mapuje sekvenci vstupních akustických charakteristik na sekvenci grafémů (nejmenších jednotek psaného jazyka) nebo slov [24].



Obrázek 5.1: Architektura *End-to-End* ASR modelu, převzato z [1].

Díky této architektuře již není nutné využívat slovník výslovnosti ani samostatný jazykový model, čímž se celý systém výrazně zjednodušuje [4]. Protože není potřeba použít

sekundární zdroj informací, jako je právě slovník výslovnosti, je možné vytvářet modely pro jazyky či oblasti, kde je získání zdrojů pro vytvoření slovníku obtížné. Sekundární zdroje navíc mohou obsahovat chyby [15]. V případě potřeby přesnějších výsledků je ale možné jazykový model do architektury zakomponovat [1].

Navzdory významnému pokroku, kterého *End-to-End* modely dosáhly z hlediska přesnosti, nemohou garantovat stoprocentní přesnost. To je způsobeno komplexitou lidské řeči, která může obsahovat nejrůznější nuance, jako je akcent, výška hlasu či rychlost mluvy a ani nejpokročilejší modely tak zatím nedokážou plně obsáhnout všechna tato specifika. I přes to, že jsou *End-to-End* modely vhodné pro aplikace vyžadující přepis řeči v reálném čase, je potřeba počítat s omezeními, jako je nižší přesnost a to zejména ve složitých situacích, jako je mluvčí s neobvyklým akcentem nebo hlučné prostředí [20].

5.1.2 Existující řešení

V této části budou představena aktuálně nejpopulárnější komerční řešení, která umožňují přepis řeči za pomoci programovacího rozhraní (angl. *application programming interface*, dále jen API). Protože je ve výsledné aplikaci potřeba přepisovat řeč v reálném čase (takzvaný *streaming*), byla vybrána pouze řešení, která tuto možnost podporují. V závěru pak budou všechna představená řešení srovnána na základě vybraných vlastností.

AssemblyAI

Modely od společnosti AssemblyAI¹ se vyznačují vcelku solidní přesností. Protože ale dva nejlepší modely nepodporují češtinu, byl pro potřeby porovnání vybrán model Nano, podporující 102 jazyků. K dispozici je jednoduché rozhraní pro integraci do aplikací, a to za pomoci SDK (*Software development kit*, vývojová sada) [6].

Microsoft Azure

Řešení od společnosti Microsoft² vyniká zejména možnostmi vytvářet vlastní modely a přizpůsobit je tak specifickým potřebám, například pro rozpoznávání odborné terminologie nebo specifickým akustickým podmínkám. Služba je dostupná prostřednictvím SDK, REST API nebo Speech CLI, což umožňuje snadnou integraci do aplikací. Pro porovnání byl vybrán model Universal, který podporuje 85 jazyků [2].

Deepgram

Hlavní výsadou tohoto řešení³ je přizpůsobitelnost svých modelů a také jejich rychlost. K dispozici jsou například modely Nova či Whisper. Pro svou dobrou integraci s různými platformami jde o vhodné řešení pro hlasové asistenty a analýzu hovorů. Protože nejnovější model Nova-3 nepodporuje češtinu, byl pro porovnání vybrán starší model Nova-2, který podporuje 36 jazyků [5], [6].

¹ AssemblyAI – <https://www.assemblyai.com/>

² Microsoft – <https://www.microsoft.com/cs-cz>

³ Deepgram – <https://deepgram.com/>

ElevenLabs

Model Scribe, vydaný touto společností⁴ v únoru 2025, je považován za jeden z aktuálně nej-
přesnějších modelů, zejména v angličtině (přesnost 96.6 %). Dále model například umožňuje
identifikaci zvuků jako smích, potlesk nebo kroky pro bohatší kontext přepisu a podporo-
váno je 99 jazyků [17].

Whisper

Jedná se o open-source řešení od společnosti OpenAI⁵. I přesto, že Whisper sám o sobě
streaming nepodporuje, je možné provozovat toto řešení přímo na serveru, což je ovšem
z hlediska integrace do aplikace poměrně náročné, nebo využít hostované řešení od společ-
nosti Deepgram. Pro porovnání byly, z důvodu výpočetní náročnosti, vybrány právě hos-
tované modely a to Whisper Small a Whisper Medium, které podporují 57 jazyků, včetně
češtiny [5].

Shrnutí

Představená řešení budou nyní porovnávána, a to na základě dvou vlastností – počtu pod-
porovaných jazyků a nákladů na využívání těchto řešení. Další relevantní vlastnost pro
porovnání by zajisté bylo zpoždění, tedy čas od okamžiku, kdy je slovo vysloveno, po oka-
mžik, kdy ho ASR rozpozná. Protože se ale většina průzkumů hodnotící modely zaměřuje
na přepis předem nahraného audia (takzvaný *batch* přepis) a samotní poskytovatelé tyto
hodnoty většinou neuvádějí, nebylo možné porovnání na základě této vlastnosti provést.

Služba (model)	Podporovaných jazyků	Náklady (USD za 1000 minut)
AssemblyAI (model Nano)	102	2.00
Azure (model Universal)	85	6.00
Deepgram (model Nova-2)	36	4.30
ElevenLabs (model Scribe)	99	6.67
Whisper Small	57	3.80
Whisper Medium	57	4.20

Tabulka 5.1: Porovnání vybraných řešení na základě podporovaných jazyků a nákladů na
provoz.

Na základě těchto výsledků lze tedy prohlásit, že z hlediska vybraných vlastností se
jako nejlepší řešení jeví model Nano od společnosti AssemblyAI. Jde ovšem pouze o malou
množinu vlastností a hlavní a nejdůležitější tak stále zůstává přesnost modelu. Pro použití
v aplikaci byl pak na základě doporučení vedoucího práce, nízkým provozním nákladům a
také díky snadné integraci do aplikace pomocí SDK vybrán model Universal od společnosti
Microsoft.

⁴ElevenLabs – <https://elevenlabs.io/>

⁵OpenAI – <https://openai.com/>

5.1.3 Metriky pro hodnocení modelů

Asi nejobecnější a nejčastěji užívanou metrikou pro hodnocení přesnosti modelů je *word error rate*, zkráceně WER. Tato metrika srovnává, jak moc se přepsaný text liší od referenčního textu, a to na úrovni slov. Chyby, které přepis může obsahovat, se řadí do tří kategorií [8]:

- **Záměny** (angl. *substitutions*) – přepsané slovo se liší od referenčního slova.
- **Vkládání** (angl. *insertions*) – přepis obsahuje slovo navíc.
- **Mazání** (angl. *deletions*) – referenční slovo v přepsaném textu chybí.

Hodnota WER je následně vypočtena pomocí následujícího vzorce. Všechny tři kategorie chyb jsou sečteny a vyděleny celkovým počtem slov N referenčního textu [8]:

$$WER = \frac{S + I + D}{N}$$

Procentuální přesnost modelu (značena jako W_{Acc} – *word accuracy*) je následně vypočtena ze získané hodnoty WER [8]:

$$W_{Acc} = 1 - WER$$

Pro hodnocení přesnosti lze využít také metriku CER (*character error rate*), která měří počet chyb na úrovni jednotlivých znaků [8]. V kontextu této práce je však vhodnější použití metriky WER, jelikož přepisy budou mít většinou formu krátkých slovních výrazů či číselných odpovědí, kde je klíčové, zda byla rozpoznána správná slova nebo čísla jako celek.

5.2 Velké jazykové modely

Použití velkých jazykových modelů v této práci nebylo původně zamýšleno. Přepisy hlasových odpovědí uživatele totiž byly původně vyhodnocovány skriptem, který z přepisu extrahoval čísla a ta byla následně porovnána se správnou odpovědí. Po počátečním testování ovšem vyšlo najevo, že vyhodnocení přepisu hlasové odpovědi uživatele je v případě odpovědi ve tvaru zlomku či hodnoty proměnné velmi nepřesné. Implementace spolehlivějšího a přesnějšího vyhodnocovacího skriptu by ovšem byla příliš komplikovaná a časově náročná. Z tohoto důvodu bylo zvoleno robustnější, ale rychlejší řešení – velký jazykový model (angl. *large language model*, dále jen LLM). V případě nasbírání dostatečně velkého množství dat by pak navíc bylo možné natrénovat jednodušší model plnící stejnou funkci. V této části proto bude princip fungování těchto modelů popsán a představena bude také technika pro efektivní využití modelů a jejich použití při vyhodnocování odpovědí v kontextu této práce.

5.2.1 Princip velkých jazykových modelů

Jde o velmi rozsáhlé modely hlubokého učení, které jsou předtrénovány na obrovském množství dat. Jejich základ tvoří architektura *transformers* – sada neuronových sítí s mechanismem *self-attention*. Díky tomu je model schopen vážit důležitost jednotlivých slov ve větě bez ohledu na jejich vzájemnou vzdálenost v textu [9].

Jednou z hlavních výhod LLM je možnost vcelku jednoduchého přizpůsobení pro specifické úlohy. Existuje několik technik jak tohoto přizpůsobení dosáhnout. Uživatelsky nejjednodušší možnost, která na rozdíl od ostatních technik nevyžaduje znalost strojového učení, protože používá přirozený jazyk, se nazývá *prompt engineering* [26].

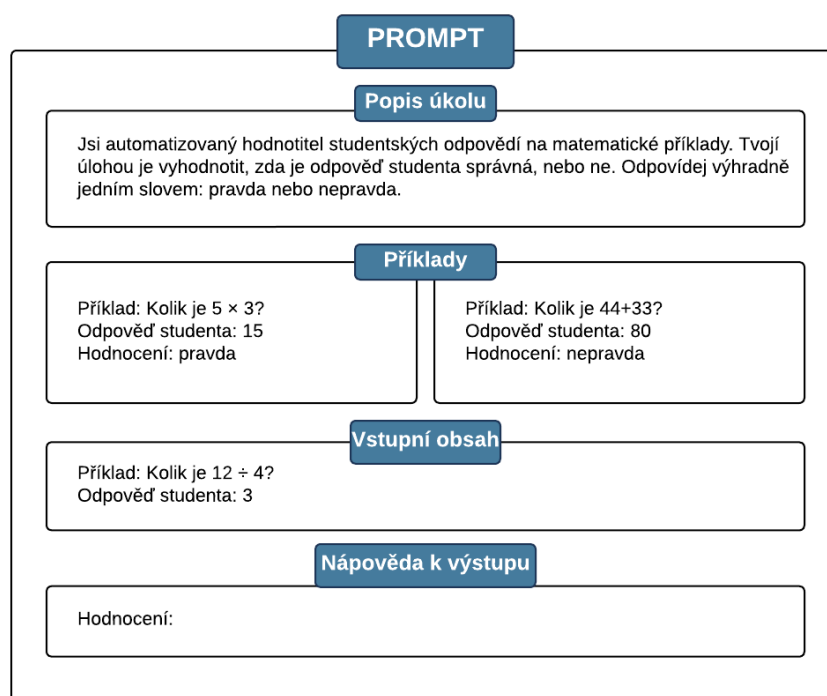
5.2.2 Prompt engineering

Jde o techniku, kdy uživatel formuluje vhodné vstupy (prompty) pro LLM tak, aby dosáhl požadovaného výstupu – například odpovědi na otázky, překlady textů nebo generování článků. Kvalita výstupu modelu totiž často výrazně závisí na tom, jak dobře je prompt navržen [26].

Ačkoli neexistuje jeden univerzální vzor správného promptu, který by vyhovoval všem úlohám, většina z nich obsahuje několik základních prvků. Služba Azure OpenAI od společnosti Microsoft popisuje vzor o pěti částech následovně [26]:

- **Popis úkolu** – obsahuje instrukci, co má LLM provést.
- **Příklady** – obsahují vstupy i očekávané výstupy a pomáhají modelu pochopit, jak má na daný úkol reagovat.
- **Podpůrný kontext** – jde o doplňující informace, které mohou zlepšit kvalitu odpovědi modelu, i když přímo nesouvisejí s úkolem.
- **Vstupní obsah** – hlavní vstup, na kterém má model pracovat, tedy text nebo dotaz, na nějž se má model zaměřit při plnění úkolu.
- **Nápověda k výstupu** – počáteční část výstupu, která pomáhá nasměrovat model ke generování odpovědi (např. aby model odpověděl na otázku, a ne jen doplnil text).

V případě této práce, ve které je potřeba vyhodnocení odpovědi uživatele, by prompt mohl vypadat například jako na obrázku 5.2.



Obrázek 5.2: Návrh promptu pro vyhodnocení odpovědi žáka vytvořený na základě vzoru. Podpůrný kontext byl v tomto případě záměrně vynechán, protože při vyhodnocení odpovědi uživatele nehraje významnou roli.

5.2.3 Zvolené řešení

Výsledná aplikace bude pro vyhodnocení odpovědí pomocí LLM využívat model Gemini (konkrétně verzi Gemini 2.0 Flash) vyvinutý společností Google⁶. Dvěma zásadními důvody pro použití tohoto řešení je vcelku jednoduchá integrace do aplikace a také nízké náklady na využívání této služby.

⁶Google – <https://www.google.com/>

Kapitola 6

Implementace a architektura

Tato kapitola pojednává o samotné implementaci webové aplikace. Popsána je tak struktura aplikace, informace o nasazení a následně je představena serverová část včetně datových modelů, koncových bodů, logiky důležitých částí implementace a komunikace s klientskou částí. Popsáno je také uživatelské rozhraní, tedy pohledy vyskytující se v aplikaci a důležité komponenty, které používá.

6.1 Architektura aplikace a nasazení

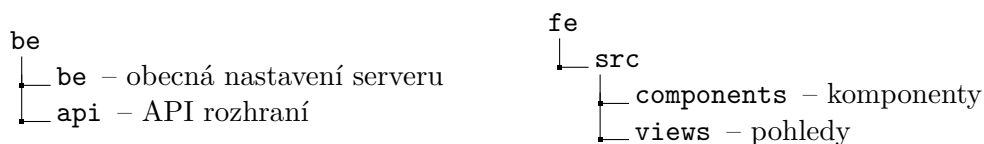
Aplikace je založena na architektuře klient-server, konkrétně používá třívrstvou architekturu (*three-tier architecture*), ta se dělí na tyto části [3]:

- **Prezentační vrstva** – vizualizuje pro uživatele informace formou grafického uživatelského rozhraní. Může provádět kontrolu vstupů uživatele, neobsahuje však zpracování dat.
- **Aplikační vrstva** – obsahuje logiku a funkce aplikace, provádí výpočty a zpracování dat.
- **Datová vrstva** – databáze.

Adresářová struktura je rozdělena na dvě hlavní části, kde první z nich implementuje server za pomoci frameworku Django a nachází se v adresáři **be**. Ten dále obsahuje dva podadresáře **be** a **api**. První z nich, adresář **be**, obsahuje nastavení serveru. V podadresáři **api** se pak nacházejí moduly implementující API rozhraní a logiku celé aplikace.

Pro klientskou část je použit framework Vue.js a nachází se v adresáři **fe**. V jeho podadresáři **src** se pak nachází zejména adresář obsahující komponenty, tedy menší, znovupoužitelné bloky uživatelského rozhraní (adresář **components**) a adresář s pohledy, které tyto komponenty využívají (adresář **views**) k vytvoření jednotlivých stránek aplikace. Pro databázi je použit relační databázový systém MySQL¹.

¹MySQL – <https://www.mysql.com/>



Obrázek 6.1: Adresářová struktura: část serveru (vlevo) a část klienta (vpravo).

Pro sestavení aplikace je použita technologie Docker², která umožňuje spouštět aplikaci v izolovaném prostředí, takzvaném kontejneru. To činí přechod z vývojového do produkčního prostředí jednodušší.

Aplikace, tedy server a databáze, je hostována na virtuálním privátním serveru (angl. *virtual private server*) poskytovaném společností Hetzner³. Samotné nasazení je pak realizováno za pomoci platformy Appliku⁴, která celý proces usnadňuje a automatizuje.

6.2 Část serveru

Jak již bylo popsáno v předešlé části, serverová část aplikace je implementována ve frameworku Django a dělí se na dvě části, kde první z nich zajišťuje nastavení aplikace, jako například způsob komunikace, připojení se k databázi či načtení proměnných prostředí. Druhá část pak obsahuje:

- definici databázových modelů (soubor `models.py`).
- modul zodpovědný za zpracování logiky při obdržení HTTP požadavku (soubor `views.py`).
- moduly zajišťující komunikaci pomocí komunikačního protokolu Websocket (soubory `consumers.py` a `consumersSurvey.py`).
- moduly poskytující pomocné funkce modulům zajišťujícím logiku (soubory `utils.py` a `answerChecker.py`).

Tyto podčásti budou nyní podrobněji popsány a vysvětleny.

6.2.1 Datové modely

V aplikaci bude vystupovat několik různých entit. Uloženy jsou v databázi MySQL a přístupováno k nim je skrze framework Django. Pro každý záznam modelu je vytvořeno unikátní identifikační číslo `id`, pomocí kterého se lze na záznam odkazovat. Vytvořeno je několik modelů, které budou nyní popsány.

Žák je v datovém modelu reprezentován modelem **Student** a obsahuje tyto atributy:

- **username** – uživatelské jméno.
- **passphrase** – heslová fráze pro autentizaci uživatele.

Tyto atributy bude žák používat pro přihlášení do aplikace. Heslová fráze je uživateli vygenerována při registraci a v databázi je uložen její hash. Podobné atributy má pak

²Docker – <https://www.docker.com/>

³Hetzner – <https://www.hetzner.com/>

⁴Appliku – <https://appliku.com/>

model reprezentující administrátora aplikace. Na rozdíl od žáka ale pro autentizaci používá klasické heslo, definované atributem **password**. Jednotlivé příklady jsou popsány modelem **Example**:

- **example** – zadání příkladu.
- **input_type** – typ vstupu, jakým bude příklad počítán.
- **task_id** – reference na sadu příkladů, do které příklad patří.

Typ vstupu určuje, jaké rozhraní pro řešení příkladů se v klientské části zobrazí. Atribut může nabývat těchto hodnot: **INLINE** – jednoduchá forma odpovědi, **FRAC** – odpověď ve tvaru zlomku a **VAR** – odpověď je hodnota proměnné či proměnných. Každý příklad může mít také svůj postup řešení. To se dělí na jednotlivé kroky, které jsou reprezentovány modelem **Step** s těmito atributy:

- **text** – text kroku řešení.
- **order** – pořadí kroku (příklad samozřejmě může mít více kroků řešení).
- **example_id** – reference na příklad, pro který je krokem řešení.

Mezi žákem a příkladem se pak nachází vztah, který reprezentuje záznam, že student daný příklad počítal. Tento vztah pak má tyto atributy:

- **date** – časové razítko.
- **duration** – čas, po který žák příklad počítal.
- **attempts** – počet pokusů na vyřešení příkladu.
- **solved** – booleovská hodnota, určující, zda žák příklad úspěšně vyřešil.
- **skipped** – booleovská hodnota, určující, zda žák příklad přeskočil.

Časové razítko označuje datum a čas, kdy byl příklad počítán a umožňuje tak vytvořit unikátní vztah mezi uživatelem a příkladem. Čas potřebný k vyřešení příkladu (**duration**) se zaznamenává v sekundách. Maximální počet pokusů pro vyřešení příkladu je tři. Po jeho překročení se atribut **solved** nastaví na hodnotu **false**. Každý příklad má také svou odpověď, kterou reprezentuje model **Answer**:

- **answer** – odpověď příkladu.
- **example_id** – reference na příklad, pro který je odpovědí.

Jednotlivé příklady se sdružují do sad příkladů. Ty jsou reprezentovány modelem **Task**, který obsahuje tyto atributy:

- **name** – název sady.
- **skills** – reference na procvičené dovednosti.
- **form** – typ zobrazení příkladů.

Atribut **skills** tedy bude určovat, které dovednosti si žák počítáním příkladů z této sady procvičí. Forma zobrazení příkladů pak může nabývat těchto hodnot:

- **classic** – zadání je v $\text{\LaTeX}$ formátu a bude vykresleno viz 6.3.7.
- **word-problem** – zadání bude zobrazeno jako prostý text.

Poslední model, který se v datovém modelu vyskytuje, reprezentuje dovednost, kterou si žáci počítáním příkladů procvičují. Tu definuje model **Skill**:

- **name** – název dovednosti.
- **height** – výška dovednosti ve stromu dovedností.
- **deleted** – booleovská hodnota umožňující mělké odstranění dovednosti.
- **parent_skill** – reference na nadřazenou dovednost.
- **skill_type** – typ dovednosti.
- **related_skills** – reference na související dovednosti.

Aby nedošlo ke ztrátě vztahů mezi dovednostmi a příklady, je místo klasického odstranění dovednosti použit atribut **deleted** umožňující takzvané mělké odstranění (angl. *soft delete*). Protože dovednosti tvoří stromovou strukturu (viz kapitola 4.1), má každá dovednost svou výšku v tomto stromě, na základě které se následně zobrazuje na straně klienta při výběru dovedností k procvičení. Stejně tak obsahuje odkaz na dovednost, která je jí nadřazená. Pokud je v jedné sadě příkladů procvičeno více dovedností z různých větví stromu, pak jsou tyto dovednosti navzájem uloženy v atributu **related_skills**. Pro rozlišení, zda se jedná například o operaci či číselný obor, slouží atribut **skill_type**. Mezi dovednostmi a příkladem existuje vztah popisující, jaké dovednosti budou daným příkladem procvičeny.

6.2.2 Koncové body API

Pro modifikaci a přístup k datům z klientské části aplikace byly vytvořeny koncové body API. Každý z těchto bodů má svou URL (*Uniform Resource Locator*) adresu, ke které klient přistupuje pomocí požadavků HTTP (*Hypertext Transfer Protocol*). Použity jsou čtyři metody, které protokol HTTP nabízí – metoda GET pro získání dat, metoda POST pro manipulaci s daty, metoda DELETE pro odstranění dat a metoda PATCH pro částečnou úpravu dat.

Všechny koncové body lze rozdělit do několika kategorií, kde každá pracuje s jiným datovým modelem či modely. Pro práci se sadami příkladů a s příklady samotnými slouží koncové body popsané v tabulce 6.1.

URL	METODA	FUNKCE
tasks/	GET	Získání všech sad včetně jejich příkladů
create-task/	POST	Vytvoření sady příkladů
edit-task/	POST	Úprava existující sady
tasks/<int:task_id>/delete/	DELETE	Odstranění sady včetně příkladů
examples/?topics={id}	GET	Získání příkladů pro id daných dovedností
example/<int:example_id>/delete/	POST	Vytvoření příkladu
check-answer/	POST	Vyhodnocení odpovědi uživatele

Tabulka 6.1: Koncové body API pro práci se sadami a příklady.

S daty, která se během procvičování o uživateli sbírají v podobě záznamů o počítání příkladu, lze manipulovat skrze tyto koncové body:

URL	METODA	FUNKCE
<code>create-record/</code>	POST	Vytvoření záznamu
<code>update-record/</code>	POST	Doplnění záznamu (změna počtu pokusů aj.)
<code>delete-record/</code>	POST	Odstranění záznamu
<code>skip-example/</code>	POST	Označení příkladu v záznamu jako přeskočený

Tabulka 6.2: Koncové body API pro práci se záznamy o počítání.

Další a nejpočetnější kategorie koncových bodů se zabývá prací s dovednostmi. Kromě základní manipulace s dovednostmi totiž tyto body slouží zejména k jejich zobrazení uživateli, a to v několika variacích. To z důvodu, že dovednosti jsou uspořádány formou stromu (viz kapitola 4.1), ale každá dovednost má navíc svůj vlastní seznam dovedností, se kterými souvisí (například dovednosti *Sčítání* a *Celá čísla*). Při zobrazování je proto nutné brát v potaz všechny tyto vztahy.

URL	METODA	FUNKCE
<code>create-skill/</code>	POST	Vytvoření dovednosti
<code>skills/<int:skill_id>/delete/</code>	PATCH	Označení dovednosti jako smazané
<code>skill/<int:skill_id>/</code>	GET	Získání konkrétní dovednosti
<code>skill-tree/</code>	GET	Získání stromové struktury dovednosti
<code>landing-page-skills/</code>	GET	Získání dovedností pro hlavní stránku
<code>skills/<int:skill_id>/tree/related</code>	GET	Získání souvisejících dovedností
<code>skills/<int:skill_id>/tree/children</code>	GET	Získání poddovedností
<code>skills/<int:skill_id>/operations</code>	GET	Získání souvisejících operací
<code>skills/search/</code>	GET	Vyhledání dovednosti
<code>skills/paths/sandbox/</code>	GET	Vyhledání dovednosti v podstromu
<code>skills/related-counts/</code>	GET	Získání příkladů a sad obsahující dovednost

Tabulka 6.3: Koncové body API pro práci s dovednostmi.

Ostatní koncové body se zabývají registrací a přihlašování uživatelů, zpracováním odpovědí uživatelů týkajících se průzkumu během používání aplikace (viz kapitola 7) a také komunikací pomocí protokolu Websocket, který se používá při zadávání výsledků příkladů hlasem a pro odpovídání hlasem na otevřené otázky v již zmíněném průzkumu.

URL	METODA	FUNKCE
<code>register/student/</code>	POST	Registrace nového uživatele
<code>login/student</code>	POST	Přihlášení uživatele
<code>login/admin</code>	POST	Přihlášení administrátora
<code>survey-answer/</code>	POST	Zpracování odpovědi na uzavřenou otázku v průzkumu
<code>ws/speech/</code>	–	Zadávání výsledků hlasem
<code>ws/survey/</code>	–	Zpracování odpovědi na otevřenou otázku v průzkumu

Tabulka 6.4: Koncové body API pro přihlašování, průzkum a komunikaci skrze protokol Websocket. Poslední dva koncové body používají protokol WebSocket místo běžné HTTP metody.

6.2.3 Vyhodnocování odpovědí

Uživatel má v aplikaci možnost zadat výsledek příkladu dvěma způsoby – klávesnicí nebo hlasem. V této části bude proto popsán princip, jakým jsou uživatelské odpovědi v obou případech zpracovávány a také modul, který zajišťuje samotné vyhodnocení odpovědi. Ten se nachází v souboru `answerChecker.py` a reprezentován je třídou `AnswerChecker`, která poskytuje základní funkce pro vyhodnocení, jako například porovnání hodnot či validace formátu odpovědi. Z této třídy pak dědí její potomci, kteří budou popsáni v dalších částech této kapitoly.

Zadávání výsledku klávesnicí

V případě, že uživatel zadá výsledek pomocí klávesnice a vyplní tak odpovědní okénko (viz obrázek 6.6), je zadaná hodnota (či hodnoty) odeslána na server spolu s příkladem, pro který má být řešením. Následně se je základě atributu příkladu `input_type` určeno, která třída vyhodnocení provede. Jedná se o tyto třídy:

- `InlineAnswerChecker` pro jednoduché (jednořádkové) odpovědi.
- `FractionAnswerChecker` pro odpovědi ve tvaru zlomku.
- `VariableAnswerChecker` je-li odpověď hodnota proměnné (či proměnných).

Zaslaná odpověď je vždy nejprve přetypována na desetinné číslo a následně porovnána se správnou odpovědí. Podle výsledku porovnání je pak upraven záznam o počítání daného příkladu uživatelem a následně je výsledek zaslán na stranu klienta, kde proběhnou příslušné akce pro informování uživatele o výsledku vyhodnocení odpovědi.

Zadávání výsledku hlasem

Na rozdíl od vyhodnocování, které je popsáno v předešlé části a probíhalo za pomoci HTTP požadavků, je pro zadávání odpovědi hlasem použit protokol WebSocket, pomocí kterého je na server zasílán uživatelský hlas a také data potřebná pro vyhodnocení. Komunikace na straně serveru je obsluhována třídou `SpeechRecognitionConsumer` nacházející se v souboru `consumers.py`.

Pokud si uživatel zvolí na straně klienta možnost zadávat výsledky hlasem, je mezi klientem a serverem otevřeno spojení a zaslána metadata jako například `id` studenta a příkladu a aktuálně používaný jazyk. Následně se na straně klienta spustí nahrávání zvuku, který se postupně ukládá do bufferu o velikosti 4096 vzorků. Po jeho naplnění jsou data převedena do formátu PCM, odeslána na server a celý proces se opakuje. Na straně serveru se zasláná data okamžitě předávají službě Azure Speech-to-Text, která provádí přepis do chvíle, kdy uživatel přestane mluvit. Samotný přepis je pak opět na základě atributu příkladu `input_type` předán jedné z tříd pro vyhodnocení odpovědi:

- `InlineSpeechAnswerChecker` pro jednoduché odpovědi jako například celá či desetinná čísla.
- `FractionSpeechAnswerChecker` pro odpovědi ve tvaru zlomku.
- `VariableSpeechAnswerChecker` je-li odpověď hodnota proměnné (či proměnných).

Jelikož ale v případě odpovědí ve složitějším formátu jako jsou zlomky docházelo k přepisování zlomků na slova a tím tak k chybnému a nepřesnému vyhodnocení, je pro tyto typy příkladů použita speciální třída `LLMAnswerChecker`, která pro vyhodnocení používá velký jazykový model Gemini. Díky tomu je vyhodnocení příkladů, jejichž odpověď má složitější formát přesnější a tím také spolehlivější. Tento model je ale omezen limitem počtu dotazů za minutu (angl. *requests per minute*). Proto jsou v případě vypršení tohoto limitu opět použity již zmíněné třídy `FractionSpeechAnswerChecker` a `VariableSpeechAnswerChecker`. Podle výsledku vyhodnocení je pak opět upraven záznam o počítání příkladu uživatelem a výsledek je poté zaslán zpět na stranu klienta.



Obrázek 6.2: Diagram znázorňující komunikaci mezi moduly během vyhodnocování výsledků zadaných hlasem. Během odpovědi uživatele je zvukový záznam odeslán klientem na server, kde dochází k jeho přepisu. Na základě typu příkladu je následně vybrána příslušná třída pro vyhodnocení přepisu a výsledek je odeslán zpět klientovi.

6.3 Část klienta

V této části se nachází implementace uživatelského rozhraní. Ta je realizována pomocí frameworku Vue.js. Použita jsou ale také rozšíření tohoto frameworku, konkrétně Vue.js Router⁵ zajišťující navigaci mezi jednotlivými pohledy a rozšíření Pinia⁶, které umožňuje sdílení dat mezi komponentami. Dále jsou využity dvě knihovny třetích stran – Axios⁷ pro přehlednější tvoření HTTP požadavků a knihovna Tailwind CSS⁸ umožňující stylování komponent a prvků za pomoci předem definovaných tříd. Popsány tedy budou jednotlivé pohledy uživatele a administrátora aplikace a také komponenty, ze kterých se skládají.

6.3.1 Hlavní stránka

Hlavní stránka je první pohled, který je uživateli při vstupu do aplikace zobrazen. Hlavní část tvoří nabídka základních dovedností, které následně uživatele přesměrují na podrobnější výběr. Dále pohled obsahuje také vyhledávač dovedností pro co nejjednodušší nalezení požadovaného cvičení. Pro potřeby testování a sbírání dat je podporována také anglická verze aplikace a to včetně zadávání výsledků hlasem. Uživatel může jazyk aplikace změnit kliknutím na ikonu vlajky reprezentující příslušný jazyk.

⁵Vue.js Router – <https://router.vuejs.org/>

⁶Pinia – <https://pinia.vuejs.org/>

⁷Axios – <https://axios-http.com/>

⁸Tailwind CSS – <https://tailwindcss.com/>



Obrázek 6.3: Uživatelské rozhraní hlavní stránky. **A** označuje nabídku základních dovedností, které lze vyhledat také pomocí vyhledávacího panelu (oblast **B**). Zobrazení anglické verze aplikace lze pomocí ikony v části **C**.

6.3.2 Výběr procvičených dovedností

Na tento pohled je uživatel přesměrován po výběru základní dovednosti a umožňuje mu přesněji specifikovat dovednosti, které chce procvičit. Samotná struktura pohledu se liší v závislosti na typu zvolené hlavní dovednosti.

Jedná-li se totiž o operaci (například *Sčítání*), jsou zobrazeny dovednosti (případně i jejich poddovednosti), které jsou s touto operací obsaženy v alespoň jedné sadě příkladů. Pokud ale uživatel zvolí dovednost, která je číselným oborem (například *Celá čísla*), jsou zobrazeny jak její poddovednosti, tak operace, se kterými souvisí (viz obrázek 6.4). Uživatel pak může označit všechny dovednosti, které chce procvičit.

Aby měl uživatel představu o počtu vybraných příkladů, je vedle každé dovednosti také zobrazeno, kolik příkladů ve vybrané kombinaci obsahuje.



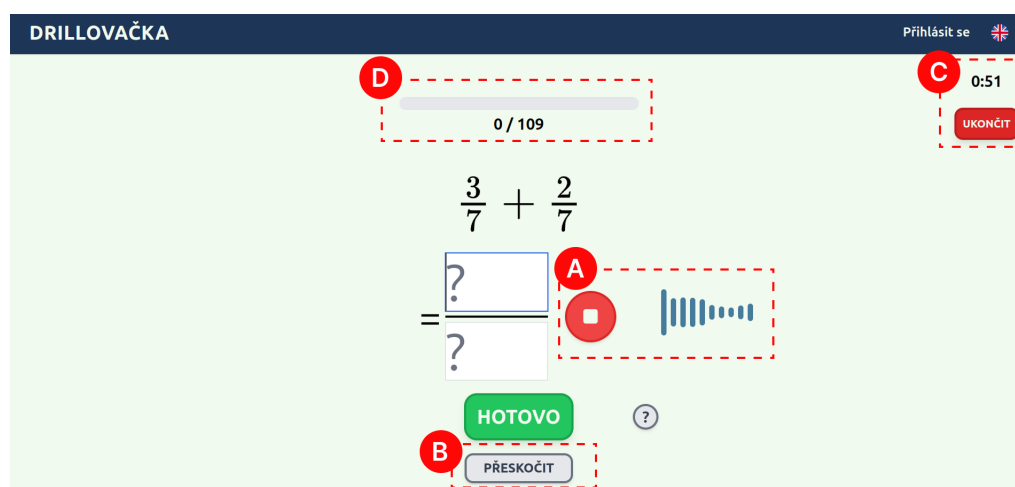
Obrázek 6.4: Výběr dovedností k procvičení. **A** označuje výběr operací se kterými zvolená dovednost souvisí. V části **B** se pak nachází výběr poddovedností.

6.3.3 Procvičování

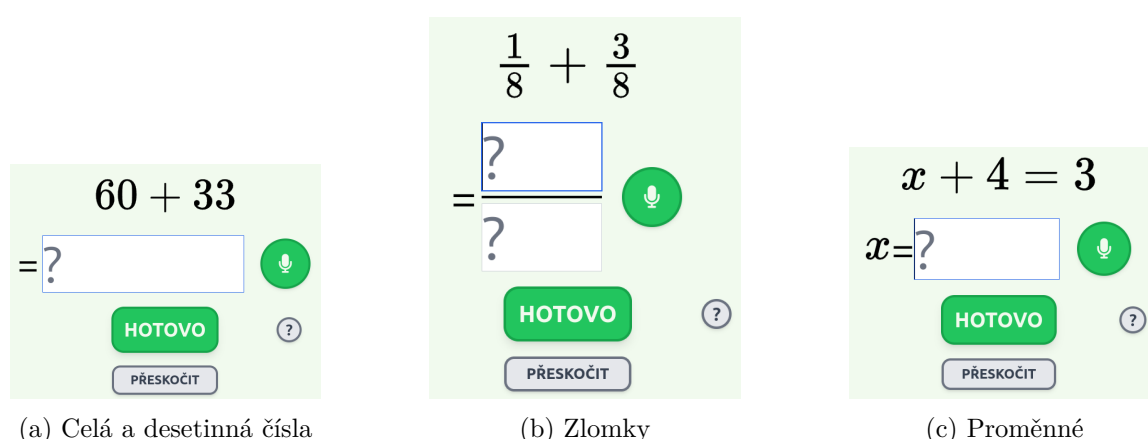
Tento pohled slouží k samotnému procvičování příkladů. Hlavní část pohledu je určena zadání příkladu a také příslušným komponentám pro zadání výsledku. Ty se na základě toho, o jaký typ příkladu se jedná, mohou měnit (viz obrázek 6.6).

Vedle vstupního pole pro zadání výsledku se pak nachází ikona mikrofonu, která umožňuje spustit zadávání výsledků hlasem. Aby byl uživatel informován o tom, že nahrávání zvuku je v pořádku a detekuje jeho hlas, nachází se vedle ikony také komponenta zajišťující vizualizaci zvukového signálu. V případě, že uživatel není schopen příklad vypočítat, má možnost dané zadání přeskočit za pomoci tlačítka *Přeskočit*, případně také ukončit celé procvičování tlačítkem *Ukončit*. Horní část zobrazuje počet spočítaných příkladů z jejich celkového počtu. V pravém horním rohu může uživatel také vidět čas, po který příklad počítá.

V případě, že uživatel zadá správnou odpověď, je zobrazena ikona znázorňující tuto skutečnost spolu s krátkým zvukovým znamením. To samé platí pro odpověď nesprávnou a v případě, že uživatel zadá nesprávnou odpověď třikrát za sebou, je mu zobrazena správná odpověď a příklad je přeskočen. Pokud příklad obsahuje i kroky řešení, jsou uživateli v případě chybné odpovědi tyto kroky postupně zobrazovány ve formě nápovědy.



Obrázek 6.5: Uživatelské rozhraní pro procvičování příkladů. **A** označuje tlačítko pro zadávání výsledků hlasem spolu s komponentou pro vizualizaci hlasu uživatele. Tlačítka v sekci **B** a **C** umožňují uživateli přeskočit příklad, respektive ukončit procvičování. V části **D** je zobrazen počet vyřešených příkladů a také jejich celkový počet.



Obrázek 6.6: Komponenty umožňující zadání odpovědi pro různé typy příkladů.

Po procvičení všech zvolených příkladů či ukončením procvičování stisknutím tlačítka je zobrazena komponenta se shrnutím právě proběhlého procvičování. Uživateli je prezentováno celkové množství spočítaných příkladů. Dále komponenta zobrazuje poměr příkladů, ve kterých uživatel neudělal chybu s příklady, kde chyboval nebo které přeskočil. Pod tímto shrnutím se pak nachází komponenta pro ohodnocení celého procvičování pro potřeby testování celé aplikace, kterým se zabývá kapitola 7.



Obrázek 6.7: Komponenta zobrazující výsledky procvičování. **A** označuje shrnutí procvičování. Sekce **B** umožňuje uživateli ohodnotit proběhlé procvičování za účelem testování.

6.3.4 Přihlášení a registrace uživatele

I přes to, že pro používání aplikace není vyžadováno vytvoření uživatelského účtu, pro případné zpracování statistik o pokroku uživatele je tato funkcionality implementována. Uživatel se do aplikace přihlašuje přezdívkou, kterou si sám vybere a také heslem respektive tříslovnou heslovou frází která je uživateli vygenerována za pomoci souboru `passphraseGenerator.js`. Uživatelé budou zejména mladšího věku a je žádoucí, aby bylo heslo jednoduché. Sestává proto ze tří slov ve formátu *vlastnost-barva-zvíře* (např. *vesely-modry-jezek*), díky čemuž si jej uživatel lépe zapamatuje, na rozdíl od klasického hesla. Generátor je v současné době schopen vytvořit přibližně 39 tisíc různých kombinací, což znamená, že vygenerovaná hesla jsou relativně slabá. Vzhledem k tomu, že uživatelský účet neobsahuje žádné citlivé údaje a důraz je kladen hlavně na jednoduchost přihlášení, je však tato úroveň zabezpečení v současné chvíli považována za dostatečnou.

Přihlásit se

Přezdívka

Přístupový kód

- -

Přihlásit se

[Ještě nemám účet - registrovat se](#)

[Jsem administrátor aplikace](#)

Registrovat se

Přezdívka

Přístupový kód

bystry-stříbrny-kocour

Vygenerovat kód

Registrovat se

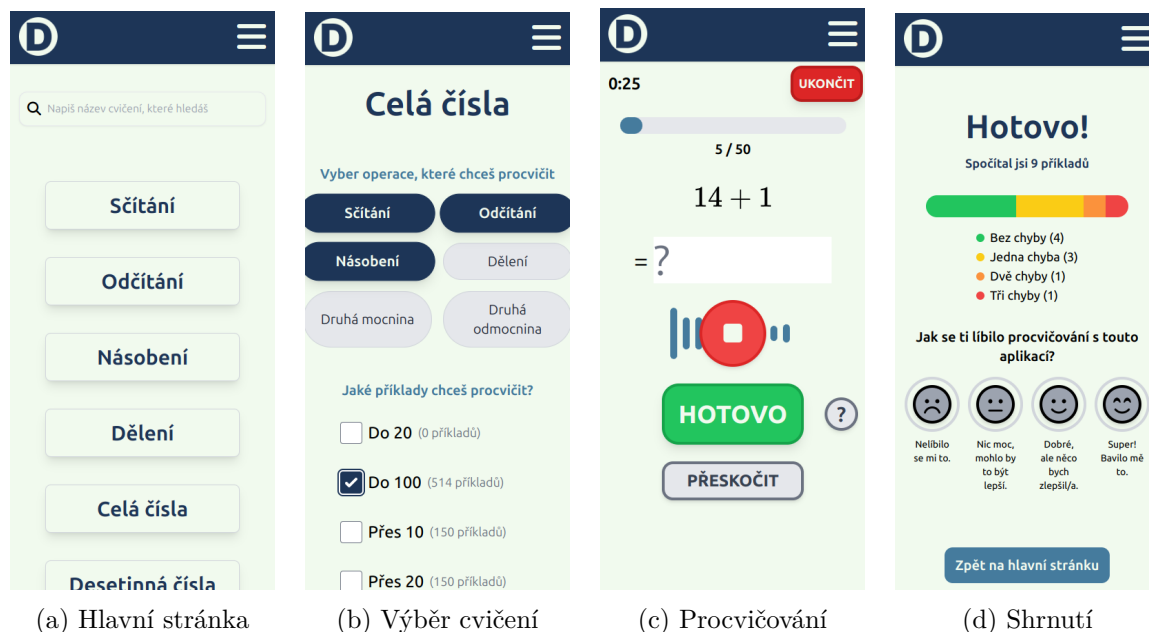
[Už mám účet - přihlásit se](#)

[Jsem administrátor aplikace](#)

Obrázek 6.8: Komponenty pro přihlášení a registraci uživatele.

6.3.5 Mobilní verze

Je nutné počítat s tím, že cílový uživatel, tedy žák navštěvující základní školu, nebude mít vždy přístup k počítači, ale naopak téměř každý bude mít k dispozici mobilní telefon. Proto je celá aplikace (vyjma pohledů administrátora) responzivní a lze ji tedy používat i na mobilním telefonu či tabletu.



Obrázek 6.9: Rozhraní aplikace určené pro mobilní telefon.

6.3.6 Seznam sad příkladů

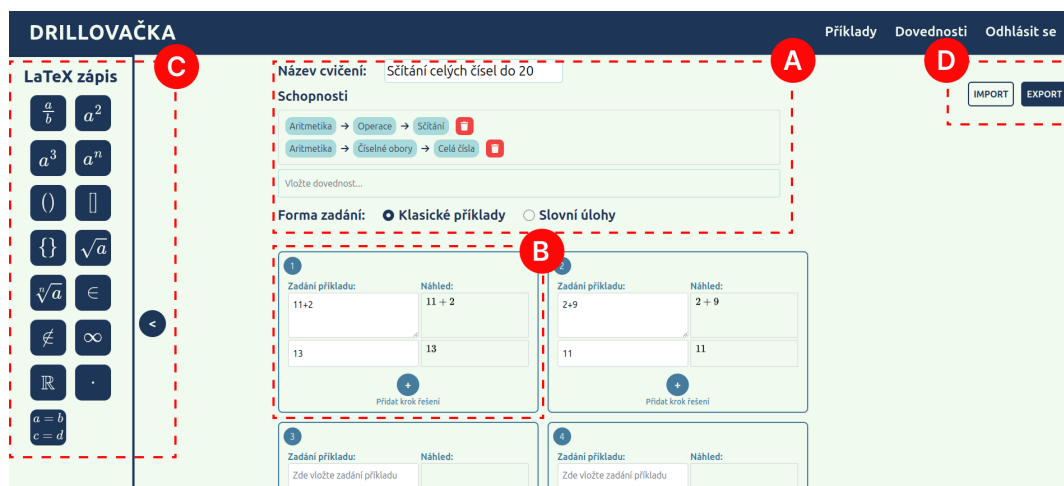
Aby měl administrátor aplikace přehled o existujících sadách, byl implementován pohled pro zobrazení všech existujících sad příkladů. Ten umožňuje rychlý náhled sady bez nutnosti použití editoru. Pokud jsou ovšem nutné úpravy sady, je možné přejít s danou sadou do editoru a zde ji upravit. V tomto pohledu lze také provést odstranění sady a příkladů, které zaštiťuje.

6.3.7 Vytváření a úprava příkladů

Pro vytváření či úpravu příkladů, respektive jejich sad, obsahuje aplikace speciální modul. V něm administrátor kromě jména sady musí vybrat také dovednosti, které budou vytvořené příklady procvičovat. Dále pak zvolit, zda bude zadání v klasické $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ formě, nebo se bude jednat o slovní úlohy zadané prostým textem.

Hlavní část obrazovky již pak obsahuje jednotlivá vstupní pole pro vložení příkladů a jejich řešení. Jak již bylo zmíněno v návrhu aplikace (viz kapitola 4.3.4), budou příklady vkládány v $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ formátu. Aby si uživatel ověřil, že zadání bude správně vykresleno, nachází se vedle vstupních polí jejich vykreslení. To je realizováno pomocí knihovny MathJax⁹, která slouží pro vykreslování $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ či MathML zápisu v prohlížečích. V případě, že uživatel nezná konkrétní příkaz v tomto formátu, může použít boční panel obsahující nejčastěji užívané výrazy. Po kliknutí na konkrétní výraz bude do aktivního vstupního pole vložen ekvivalentní $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ zápis. K zadání příkladu lze také přidat kroky řešení, které jsou uživateli zobrazovány v případě nesprávné odpovědi jako nápověda.

Aby nebylo nutné všechny příklady zadávat ručně, je implementován import ale také export celé sady, což vytváření příkladů výrazně zjednodušuje a samotný modul pak může sloužit zejména ke kontrole a editaci detailů. Sadu lze následně vytvořit za pomoci tlačítka *Vytvořit cvičení* respektive *Uložit změny*.



Obrázek 6.10: Uživatelské rozhraní modulu pro tvorbu sad příkladů. **A** označuje část pro nastavení jména sady, procvičených dovedností a také formy zadání. V sekci **B** se pak nachází vstupní pole příkladu a také jeho vykreslená verze. Pro vložení $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ zápisu výrazu je možné využít panel v části **C**. Pro import a export příkladů slouží tlačítka v části **D**.

⁹MathJax – <https://www.mathjax.org/>

6.3.8 Správa dovedností

V kompetenci administrátora je také vytváření dovedností, případně jejich odstranění. K tomu slouží pohled, který dovednosti zobrazuje ve stromové struktuře. Pohled tak umožňuje do-
tvářet celkovou podobu struktury stromu a přidávat nové větve dovedností. V případě, že
chce administrátor odstranit dovednost, je upozorněn formou modální okna oznamující,
s kolika příklady a sadami je tato dovednost spjata.



Obrázek 6.11: Uživatelské rozhraní modulu pro tvorbu dovedností.

Kapitola 7

Testování

Po implementaci a nasazení aplikace následovala fáze testování. Bylo tedy potřeba, aby cíloví uživatelé aplikaci vyzkoušeli a během jejího používání také odpověděli na otázky, na základě kterých pak bude možné vyhodnotit silné a slabé stránky implementovaného řešení. Tato kapitola proto popisuje, jaké informace bylo potřeba za pomoci testování získat a také způsob, jakým bylo realizováno. Na závěr kapitoly pak budou představeny výsledky testování a také srovnání technologií použitých pro zadávání výsledků hlasem, a to na základě dat nasbíraných od testujících uživatelů.

7.1 Cíle testování

Pro vytyčení, čeho má být skrze testování aplikace dosaženo, bylo nutné vycházet jak z požadavků uživatelů (viz kapitola 2.2), tak z funkcionality, kterou se aplikace liší od jiných řešení, která se na procvičování matematiky zaměřují – vkládání výsledků hlasem. Z tohoto hlediska je tedy potřeba zjistit, jak je uživateli tato funkcionality vnímána v porovnání s klasickým zadáváním pomocí klávesnice.

Cílem testování bylo získat odpovědi na tyto otázky:

- Jak rychle uživatel našel téma, které chtěl procvičit?
- Preferoval uživatel zadávat výsledky hlasem nebo klávesnicí?
- Byla aplikace dostatečně rychlá a přesná z hlediska zadávání výsledků hlasem?
- Jaký celkový dojem uživatelé z používání aplikace mají?

7.2 Návrh testování

K dosažení cílů vytyčených v předchozí části byl zvolen uživatelský průzkum v podobě dotazníku. Navržena proto byla sada devíti otázek, při jejichž návrhu bylo dbáno na dodržení doporučených postupů, kterých se při tvorbě dotazníku držet. Vyjmenovat lze například přiměřený počet otázek (5 až 10) či použití různých typů otázek (kombinace otevřených a uzavřených otázek) [18]. Dále byl také brán ohled na věk dotazovaných, proto byly otázky formulovány jednoduše a srozumitelně tak, aby je mladší uživatelé bez problémů pochopili.

Uživatelé tak budou dotazováni na tyto otázky:

- „Jak rychle jsi našel cvičení, které jsi chtěl procvičit?“ (*škálová otázka*)
- „Na jakém zařízení aplikaci používáš?“ (*výběrová otázka*)
- „Bylo něco, co tě při počítání rušilo nebo ti vadilo?“ (*otevřená otázka*)
- „Vyhovovalo ti více zadávat výsledky hlasem nebo klávesnicí?“ (*výběrová otázka*)
- „Stalo se ti, že aplikace řekla, že máš chybu, i když jsi měl/a správný výsledek? Pokud ano, jak často?“ (*škálová otázka*)
- „Kdybys mohl/a něco v aplikaci změnit, co by to bylo?“ (*otevřená otázka*)
- „Jak rychle aplikace reagovala, když jsi řekl výsledek?“ (*škálová otázka*)
- „Přišlo ti rychlejší říkat výsledky nahlas, nebo je psát na klávesnici?“ (*výběrová otázka*)
- „Stalo se ti, že něco v aplikaci nefungovalo tak, jak mělo? Kde přesně?“ (*otevřená otázka*)

Aby mohli uživatelé své odpovědi zadávat již během používání aplikace a sdělení jejich dojmů tak bylo co nejvíce spontánní, byly otázky zobrazovány přímo v aplikaci. Pro tuto potřebu byla dodatečně implementována komponenta, která otázku položila po každých deseti spočítaných příkladech. Ta, vzhledem k již zmíněnému nižšímu věku respondentů, umožňuje mimo jiné odpovídat na otevřené otázky hlasem, aby uživatelé ztratili během zodpovídání otázek co nejméně času.



Obrázek 7.1: Komponenty určené pro zobrazení jednotlivých typů otázek.

Po implementaci a integraci dotazníku do aplikace bylo osloveno 15 žáků základní školy a spolu s instrukcemi jim byl sdílen odkaz k aplikaci. Po uplynutí tří týdnů pak bylo testování ukončeno a provedena analýza odpovědí. Její výsledky budou představeny v následující části.

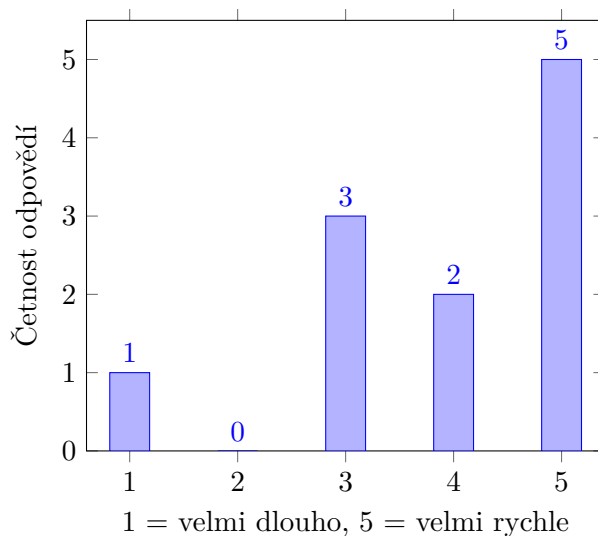
7.3 Vyhodnocení testování

Celkově se do testování zapojilo deset žáků z patnácti oslovených, kde většina z nich navštěvovala první stupeň základní školy. Proto bylo nejvíce odpovědí získáno zejména pro jednodušší příklady zaměřující se na počítání s celými čísly. I přes to ale menší množství respondentů vyzkoušelo složitější příklady na počítání s desetinnými čísly či se zlomky.

7.3.1 Výsledky testování

Z hlediska zařízení, která uživatelé pro testování aplikace použili se ukázalo, že polovina z nich, tedy pět, použila mobilní telefon, druhá polovina pak počítač. Z toho vyplývá, že responzivita aplikace byla nutností a uživatelé ji berou již jako samozřejmost. To, jak dobře se uživatelé v aplikaci orientují, respektive jak rychle dokážou nalézt cvičení, které chtějí procvičit, je zobrazeno na histogramu 7.2.

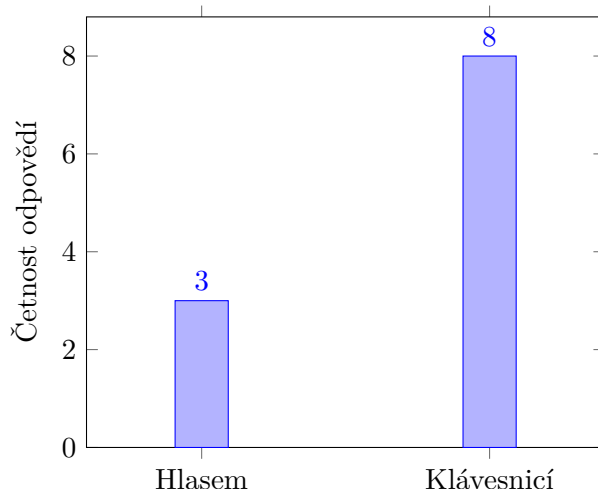
Jak rychle jsi našel cvičení, které jsi chtěl procvičit?



Obrázek 7.2: Histogram odpovědí zobrazující, jak rychle byli uživatelé schopni nalézt požadované cvičení.

Jak lze z histogramu vyčíst, ve většině případů nedělalo uživatelům nalezení požadovaného cvičení problém. Lze tedy prohlásit, že požadavky uživatelů na aplikaci byly v tomto případě úspěšně promítnuty ve výsledném řešení. Následovalo vyhodnocení preferencí uživatelů mezi zadáváním výsledků hlasem a klávesnicí. Výsledky jsou zobrazeny na histogramu 7.3.

Vyhovovalo ti více zadávat výsledky hlasem nebo klávesnicí?



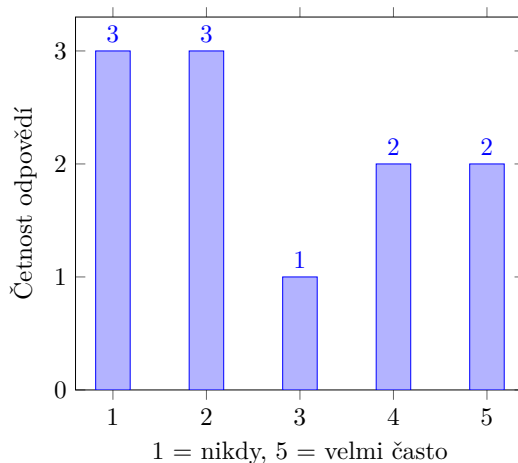
Obrázek 7.3: Histogram preference zadávání výsledků.

Je patrné, že uživatelé upřednostňovali zadávat výsledky pomocí klávesnice. Zadávání hlasem uživatelé preferovali pouze u příkladů, které byly jednodušší, jako například počítání s celými čísly. U složitějších typů příkladů, jako počítání se zlomky preferovali výhradně zadávání klávesnicí.

Podobně odpovídali uživatelé i na otázku „*Přišlo ti rychlejší říkat výsledky nahlas, nebo je psát na klávesnici?*“, kde u jednodušších příkladů volili možnost zadávat výsledky nahlas a naopak u složitějších se přikláněli k zadávání na klávesnici. Z toho lze vyvodit, že zadávat výsledky hlasem uživatelé ocení zejména u příkladů, které budou schopni rychle vyřešit, tedy například cvičení z aritmetiky prvního stupně základní školy.

Dále bylo vyhodnoceno samotné zadávání výsledků hlasem. Důležitá byla zejména přesnost, s jakou byly odpovědi vyhodnocovány. Hodnocení přesnosti uživateli je zobrazeno na následujícím histogramu.

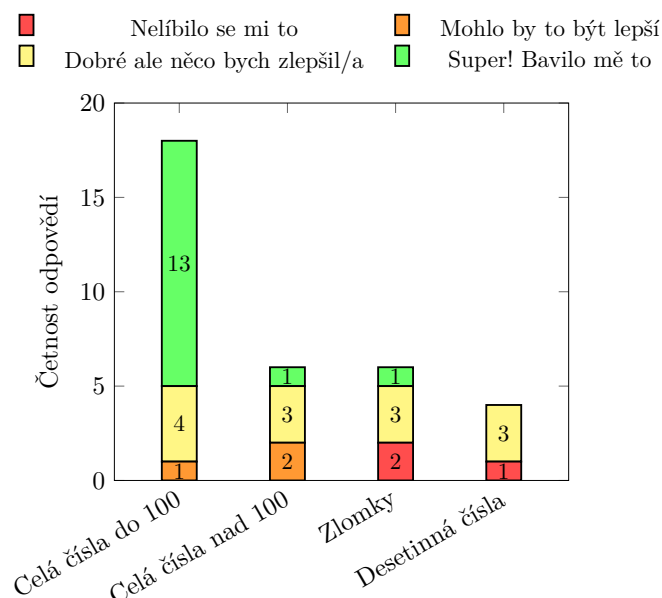
Stalo se ti, že aplikace řekla, že máš chybu, i když jsi měl/a správný výsledek? Pokud ano, jak často?



Obrázek 7.4: Histogram přesnosti vyhodnocování odpovědí hlasem.

Na histogramu lze vidět rovnoměrné zastoupení odpovědí. Vyšší přesnost uživatelé uváděli opět u jednodušších typů příkladů, naopak nižší opět u složitějších typů. Z hlediska rychlosti, kterou aplikace na jejich hlasové vstupy reagovala, byly kladněji hodnoceny typy příkladů u jejichž vyhodnocení nemusel být použit velký jazykový model (viz 6.2.3).

Poslední bod, který byl vyhodnocen, je celkový dojem uživatelů během používání aplikace. V histogramu níže se proto nachází ohodnocení kategorií příkladů, které byly uživateli testovány.



Obrázek 7.5: Histogram dojmů uživatelů z aplikace.

Je patrné, že nejlépe a také nejčastěji bylo uživateli hodnoceno procvičování příkladů na celá čísla a to také proto, že tuto kategorii vyzkoušelo nejvíce testujících uživatelů. Méně uspokojivé výsledky na druhou stranu přinesly příklady se zlomky a desetinnými čísly.

Na otevřené otázky uživatelé odpovídali spíše strohým způsobem. Z hlediska věcí, které by v aplikaci mohly být jinak, neměli respondenti žádné nápady ani výhrady. Jediné výtky byly vedeny k zadávání hlasem, které dle uživatelů občas nereagovalo zcela správně.

7.3.2 Shrnutí

Pro vyhodnocení výsledné aplikace bylo provedeno testování. Nejprve byly vytyčeny vlastnoti aplikace, jejichž kvalita bude uživateli otestována. Následně byla navržena sada otázek, na které budou uživatelé během používání aplikace dotazováni a také implementována komponenta, která zobrazování otázek v aplikaci umožní. Osloveno bylo 15 žáků základní školy. Do testování se však zapojilo pouze 10 z nich. Po uplynutí lhůty pro otestování aplikace byly odpovědi vyhodnoceny a vyvozeny z nich tyto závěry:

- Uživatelé se v aplikaci byli schopni vcelku solidně orientovat a byl tedy splněn jejich požadavek na přehlednost a jednoduchost vyhledání požadovaného cvičení.
- Preferováno bylo zadávání výsledků pomocí klávesnice, a to zejména u složitějších příkladů.

- Možnost zadávat výsledky hlasem a přesnost této formy vstupu uživatelé kladně hodnotili spíše u jednodušších příkladů.
- Ohlasy uživatelů na aplikaci byly ve většině pozitivní, ovšem s prostorem pro zlepšení.

Aplikace byla rovněž prezentována na studentské konferenci Excel@FIT 2025, kde bylo možné získat zpětnou vazbu od odborné veřejnosti. Návštěvníky zaujala jak originalita aplikace na českém trhu, tak i její možné rozšíření na základní školy. Mezi podněty ke zlepšení zazněl například návrh na natrénování vlastního ASR modelu přizpůsobeného specifikům této aplikace.

Celkově lze tedy považovat testování za úspěšné, jelikož je aplikace uživateli vnímána pozitivně i přes drobné výtky. Jako slabá stránka aplikace se jeví zadávání hlasem u složitějších příkladů a přesnost vyhodnocení odpovědi hlasem. Možným řešením tohoto nedostatku je nasazení výkonnějšího jazykového modelu, který by mohl zajistit vyšší přesnost, ačkoliv by to znamenalo vyšší provozní náklady.

7.4 Vyhodnocení zvolených řešení pro zadávání výsledků hlasem

Cílem testování bylo mimo jiné ověřit, jak dobře si zvolená řešení pro zadávání výsledků hlasem vedly při porovnání s jinými. Jak řešení pro přepis řeči (model Universal od společnosti Microsoft Azure), tak velký jazykový model pro vyhodnocení odpovědi (model Gemini od společnosti Google) totiž byly zvoleny na základě jejich obecných vlastností a doporučení vedoucího práce.

Z tohoto důvodu byla z hlasových vstupů, které uživatelé během testování zadávali do aplikace, vytvořena datová sada. Pomocí této sady totiž bude možné vyhodnotit, zda zvolené řešení pro přepis řeči dosahuje uspokojivé přesnosti, nebo zda bude potřeba přejít na jiné, přesnější řešení.

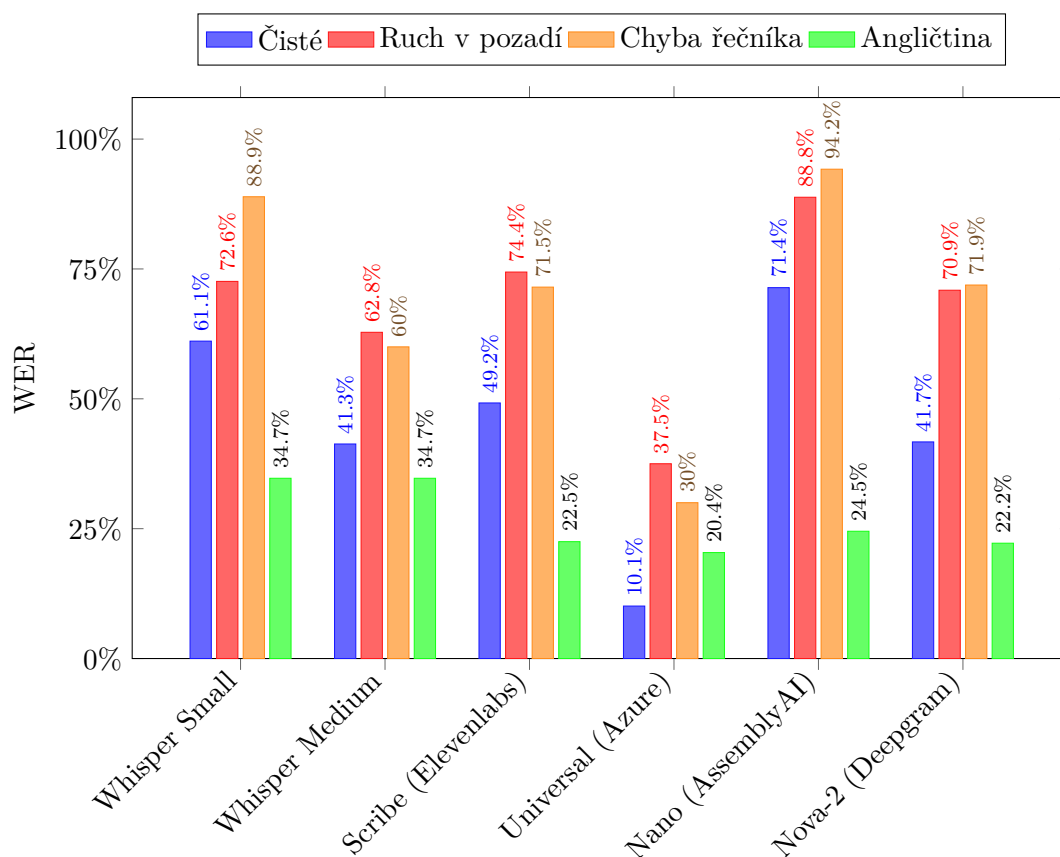
V průběhu testování tak bylo nasbíráno celkem 800 audio nahrávek (v průměru 5 sekund dlouhých), ve kterých uživatelé vyslovovali odpovědi na matematické příklady zobrazované v aplikaci. V nahrávkách vystupuje celkem deset různých řečníků (v případě nahrávek v češtině), v angličtině pak tři řečníci. Všechny nahrávky byly následně manuálně anotovány a rozděleny do čtyř kategorií:

- **Čisté** — nahrávky obsahující jasně a zřetelně vyslovenou odpověď bez rušivého hluku v pozadí (přibližně 450 nahrávek).
- **Ruch v pozadí** — delší nahrávky, které kromě odpovědi uživatele obsahují také řeč či zvuky z okolí (přibližně 150 nahrávek).
- **Chyba řečníka** — nahrávky, ve kterých uživatel kromě odpovědi vyslovuje další slova, případně hovoří s výraznými výslovnostními chybami (přibližně 100 nahrávek).
- **Angličtina** — nahrávky, ve kterých uživatel odpovídá v anglickém jazyce (přibližně 50 nahrávek).

Vybrané ASR systémy budou hodnoceny samostatně pro každou kategorii. Díky tomu bude možné například určit, které řešení si lépe poradí s rušným prostředím nebo které je vhodnější pro hlasové vstupy v angličtině.

7.4.1 Výsledky vybraných ASR na datové sadě

Všechny nahrávky byly přepsány pomocí každého z vybraných ASR systémů, které jsou blíže popsány v podkapitole 5.1. V případě nahrávek s ruchem v pozadí byl přepis často doplněn o zvuky z okolí, proto byly tyto přepisy manuálně ořezány tak, aby obsahovaly pouze část, ve které uživatel skutečně vyslovuje odpověď. Po vytvoření přepisů následovala závěrečná fáze vyhodnocení. Přepisy byly nejprve normalizovány (např. odstraněním diakritiky, přepisem čísel na slova apod.) a poté porovnány s referenčním textem. Na základě těchto porovnání byla pro každou kategorii vypočtena hodnota metriky WER, která určuje přesnost modelů na vytvořené datové sadě. Přehled výsledků vyhodnocení je zobrazen na následujícím grafu.



Obrázek 7.6: Srovnání vybraných ASR modelů dle metriky WER na vytvořené datové sadě.

Srovnání modelů ukázalo, že ve všech kategoriích dosahuje nejlepší přesnosti řešení Microsoft Azure s modelem Universal, které je již v aplikaci využíváno. Výsledky tedy potvrzují, že není nutné přecházet na jiné řešení. Naopak nejhorší přesnost vykázalo řešení od společnosti AssemblyAI a model Whisper Small. Ostatní testované modely rovněž nedosáhly uspokojivých výsledků, a proto nejsou vhodnými kandidáty pro použití v rámci vytvořené aplikace.

7.4.2 Přesnost vyhodnocení odpovědi při použití LLM

V aktuální verzi aplikace je LLM využíván pouze pro vyhodnocování odpovědí ve formě zlomků a výrazů s proměnnými. Je však žádoucí ověřit, zda by jeho použití nemohlo být přínosné i u jiných typů odpovědí. Při tvorbě datové sady, popsané v předchozí části, byla proto ke každé audio nahrávce ukládána doprovodná data obsahující přepis odpovědi generovaný aplikací, zadání příkladu a správné řešení.

Tato data byla následně anotována – bylo určeno, zda by měla být daná odpověď uživatele vyhodnocena jako správná či nesprávná – a roztríděna do tří kategorií podle typu odpovědi: celá čísla, desetinná čísla a zlomky. Pro každou z těchto kategorií bylo provedeno vyhodnocení pomocí dvou přístupů: jednak klasickým skriptem s příslušnou třídou (popsány v kapitole 6.2.3) a pomocí jazykového modelu Gemini. Přesnost obou přístupů byla vyhodnocena jako poměr správně posouzených odpovědí k jejich celkovému počtu. Výsledky srovnání jsou popsány v tabulce 7.1.

Přístup	Celá čísla	Desetinná čísla	Zlomky
Skript	96.0 %	68.4 %	75.9 %
Gemini	95.5 %	97.5 %	85.7 %

Tabulka 7.1: Přesnost vyhodnocení skriptu a LLM Gemini.

Z tabulky je patrné, že největší zlepšení přesnosti díky použití LLM nastalo v případě desetinných čísel, a to až o 30 %. Je proto jistě žádoucí používat pro vyhodnocení v této kategorii LLM. Mírné zlepšení o 10 % také přineslo použití LLM u zlomků, což pouze potvrzuje nutnost používat jej v této kategorii. Naopak mírně nižší přesnost vykazoval velký jazykový model u celých čísel. Přesnost skriptu je navíc v této kategorii uspokojivě vysoká a není tak třeba způsob vyhodnocení měnit.

Kapitola 8

Závěr

Cílem této práce bylo vytvořit webovou aplikaci pro procvičování matematiky, která bude umožňovat efektivní a uživatelsky přívětivé počítání příkladů s možností zadávat výsledky hlasem.

V úvodní fázi byly nastudovány principy tvorby uživatelských rozhraní – tedy jak uživatelé rozhraní vnímají, jaké mají zažité zvyklosti a jaké standardy od něj očekávají. Dále byly porovnány metody testování rozhraní, konkrétně pomocí rozhovorů a dotazníků.

Následně byl proveden průzkum existujících řešení zaměřených na procvičování matematiky. U vybraných platforem byly popsány jejich nedostatky, ale v několika případech také jejich pozitivní vlastnosti, které sloužily jako inspirace při návrhu vyvíjené aplikace. Pro definování uživatelských potřeb bylo následně uskutečněno pět rozhovorů s cílovými uživateli – žáky základní školy. Díky tomu bylo možné identifikovat i takové potřeby, které z průzkumu existujících řešení nevyplynuly nebo nebyly na první pohled zřejmé. Na základě všech získaných poznatků pak byl vytvořen návrh aplikace a také vybrány technologie, za pomoci kterých bude aplikace vytvořena.

Na základě vytvořeného návrhu bylo následně přistoupeno k implementaci řešení, přičemž jeho zdrojový kód je volně dostupný jako open-source¹. Vytvořená aplikace umožňuje jak procvičování matematiky základní školy formou drillu, tak tvorbu příkladů a dovedností prostřednictvím administrátorské části rozhraní. Pro podporu zadávání výsledků hlasem bylo nezbytné integrovat službu umožňující automatický přepis řeči v reálném čase (tzv. streamování). Po prozkoumání dostupných možností a na doporučení vedoucího práce bylo zvoleno řešení Azure Speech-to-Text od společnosti Microsoft, a to především díky jednoduché integraci. Model pro přepis řeči však nedokázal převést odpovědi ve složitějším formátu, například zlomky, do číselné podoby. Vyhodnocovací skript, který tyto přepisy zpracovával, se proto v těchto případech ukázal jako nespolehlivý. Z tohoto důvodu byl pro vyhodnocení složitějších odpovědí využit velký jazykový model Gemini.

Implementované řešení bylo následně nasazeno² a vývoj přešel do fáze testování. To probíhalo formou dotazníku, jehož otázky byly uživatelům zobrazovány přímo v aplikaci během jejího používání. Testování se zúčastnilo deset žáků základní školy a na základě jejich odpovědí byla aplikace z hlediska uživatelské zkušenosti hodnocena převážně pozitivně. Jedinou opakující se výtkou byly občasné nepřesnosti při vyhodnocování odpovědí zadaných hlasem. Dalším cílem práce bylo porovnat existující řešení pro automatický přepis řeči. Během testování proto byly ukládány hlasové vstupy uživatelů, z nichž byla vytvořena datová

¹Zdrojový kód je volně dostupný na platformě GitHub: <https://github.com/PanHorut/BP>

²Aplikace je dostupná na <https://drillovacka.aplikuapp.com/>

sada sloužící k vyhodnocení jednotlivých řešení pomocí metriky WER. Nejlepších výsledků dosáhlo řešení Azure Speech-to-Text, které je v aplikaci aktuálně využíváno, a není proto nutné uvažovat o jeho nahrazení. Kromě audio nahrávek hlasových vstupů byl ukládán také jejich přepis, zadání příkladu a správný výsledek. Díky tomu bylo možné porovnat přesnost vyhodnocení pomocí skriptu a velkého jazykového modelu. Největší zlepšení přesnosti přineslo využití jazykového modelu u odpovědí ve formě desetinných čísel a zlomků. Naopak u celých čísel postačovalo klasické vyhodnocení skriptem, které dosahovalo velmi uspokojivé přesnosti.

I přes to, že aplikace v současné podobě umožňuje vytváření uživatelských účtů, nebyla část týkající se uživatelského profilu a statistik o jeho pokroku, po dohodě s vedoucím, implementována. Místo toho byla pozornost věnována srovnání různých služeb pro přepis řeči na základě vytvořené datové sady. Do budoucna se tak nabízí dokončení této části aplikace a zaměření se na způsoby vizualizace uživatelského pokroku.

Literatura

- [1] ALDARMAKI, H.; ULLAH, A.; RAM, S. a ZAKI, N. Unsupervised Automatic Speech Recognition: A review. *Speech Communication*. Elsevier BV, duben 2022, sv. 139, s. 76–91. ISSN 0167-6393. Dostupné z: <http://dx.doi.org/10.1016/j.specom.2022.02.005>.
- [2] AZURE. *What is speech to text?* online. 18. března 2025. Dostupné z: <https://docs.azure.cn/en-us/ai-services/speech-service/speech-to-text>. [cit. 2025-05-02].
- [3] BURGET, R. *Informační systémy – Architektury informačních systémů* online. 2024. [cit. 2024-12-26].
- [4] CHABLANI, M. *ASR ML Systems: Overview and latest model architectures: Transducers, TDT* online. 2024. Dostupné z: <https://medium.com/@ManishChablani/asr-ml-systems-overview-and-latest-model-architectures-transducers-tdt-b547bf6db366>. [cit. 2024-12-30].
- [5] EDEN AI. *Best Speech-to-Text APIs in 2025* online. 2025. Dostupné z: <https://www.edenai.co/post/best-speech-to-text-apis>. [cit. 2025-05-02].
- [6] GLADIA. *Best speech-to-text APIs in 2025* online. 7. ledna 2025. Dostupné z: <https://www.gladia.io/blog/best-speech-to-text-apis-in-2025>. [cit. 2025-05-02].
- [7] HODNETT, B. *What is mental math?* online. Dostupné z: https://www.edu.cz/wp-content/uploads/2023/07/RVP_ZV_2023_zmeny.pdf. [cit. 2024-12-19].
- [8] HUGGING FACE. *Evaluation metrics for ASR* online. Dostupné z: <https://huggingface.co/learn/audio-course/chapter5/evaluation>. [cit. 2025-01-01].
- [9] JAIMAN, A. *Large Language Models [LLMs]* online. 28. dubna 2024. Dostupné z: <https://ashishjaiman.medium.com/large-language-models-llms-260bf4f39007>. [cit. 2025-05-02].
- [10] KARAKHAINKO, K. *Choosing the best tech stack for web application development: a 2023 guide* online. 29. října 2023. Dostupné z: <https://techwings.com/blog/choosing-the-best-tech-stack-for-web-app-development>. [cit. 2024-12-22].
- [11] KRUG, S. *Don't make me think, revisited*. 3. vyd. Upper Saddle River, NJ: New Riders Publishing, prosinec 2013.

- [12] MARSH, J. *UX For Beginners*. Sebastopol, CA: O'Reilly Media, leden 2016.
- [13] MDN CONTRIBUTORS. *Django introduction* online. 2024. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Django/Introduction. [cit. 2024-12-23].
- [14] MORAVCOVÁ, V.; SURYNKOVÁ, P. a HROMADOVÁ, J. Comparison of lower secondary school education of mathematics in the Czech Republic and selected countries with respect to curriculum documents. *Sci. Educ.* Charles University in Prague, Karolinum Press, prosinec 2019, sv. 10, č. 3.
- [15] PRABHAVALKAR, R.; HORI, T.; SAINATH, T. N.; SCHLÜTER, R. a WATANABE, S. *End-to-End Speech Recognition: A Survey*. arXiv, 2023. Dostupné z: <https://arxiv.org/abs/2303.03329>.
- [16] REACT. *Writing Markup with JSX* online. Dostupné z: <https://react.dev/learn/writing-markup-with-jsx>. [cit. 2024-12-23].
- [17] SCHNEIDER, F. a KÄNEL, T. v. *Meet Scribe* online. 26. února 2025. Dostupné z: <https://elevenlabs.io/blog/meet-scribe>. [cit. 2025-05-02].
- [18] SOEGAARD, M. *User Experience (UX) Surveys: The Ultimate Guide*. 6. October 2023. Dostupné z: <https://www.interaction-design.org/literature/article/ux-surveys>. 2025-05-05.
- [19] SOFTWARE MIND. *Choosing the Right Web Development Technologies* online. 4. října 2023. Dostupné z: <https://softwaremind.com/blog/choosing-the-right-web-development-technologies/>. [cit. 2024-12-22].
- [20] SRIVASTAVA, N. *An Overview of End-to-End Automatic Speech Recognition* online. 2. dubna 2025. Dostupné z: <https://www.e2enetworks.com/blog/an-overview-of-end-to-end-automatic-speech-recognition>. [cit. 2025-05-02].
- [21] THEY MAKE DESIGN. *Web Development Stacks: Choosing the Right Technologies* online. 24. listopadu 2023. Dostupné z: <https://medium.com/theymakedesign/what-is-web-development-stack-a9a01c5ab9e8>. [cit. 2024-12-22].
- [22] TIDWELL, J. *Designing Interfaces*. 2. vyd. Sebastopol, CA: O'Reilly Media, prosinec 2010.
- [23] VUE. *Introduction* online. Dostupné z: <https://vuejs.org/guide/introduction.html>. [cit. 2024-12-23].
- [24] WANG, S. a LI, G. Overview of end-to-end speech recognition. *Journal of Physics: Conference Series*. IOP Publishing, duben 2019, sv. 1187, č. 5, s. 052068. ISSN 1742-6596. Dostupné z: <http://dx.doi.org/10.1088/1742-6596/1187/5/052068>.
- [25] ZHANG, S.; KONG, J.; CHEN, C.; LI, Y. a LIANG, H. Speech GAU: A Single Head Attention for Mandarin Speech Recognition for Air Traffic Control. *Aerospace*. MDPI AG, červenec 2022, sv. 9, č. 8, s. 395. ISSN 2226-4310. Dostupné z: <http://dx.doi.org/10.3390/aerospace9080395>.

- [26] ZI, W.; ASRI, L. E. a PRINCE, S. *A High-level Overview of Large Language Models* online. 12. července 2023. Dostupné z: <https://rbcborealis.com/research-blogs/a-high-level-overview-of-large-language-models/>. [cit. 2025-05-02].
- [27] ZORMANOVÁ, L. Gamifikace – nový fenomén ve výuce. *Metodický portál: Články*, Březen 2022. ISSN 1802-4785. Dostupné z: <https://clanky.rvp.cz/clanek/22995/GAMIFIKACE-NOVY-FENOMEN-VE-VYUCE.html>. [cit. 2024-12-19].

Příloha A

Obsah paměťového média

/	
└─ implementace/	
└─ be/.....	zdrojové kódy serveru (Django)
└─ fe/.....	zdrojové kódy klientské části aplikace (Vue.js)
└─ README.md.....	popis adresářové struktury aplikace a návod ke spuštění
└─ LICENSE.....	licence vytvořené aplikace
└─ excel/	
└─ abstrakt.pdf.....	rozšířený abstrakt pro konferenci Excel@FIT 2025
└─ poster.pdf.....	plakát práce pro konferenci Excel@FIT 2025
└─ latex_src/.....	zdrojové soubory k technické zprávě
└─ xhorut01.pdf.....	PDF technické zprávy
└─ README.md.....	soubor s obsahem paměťového média