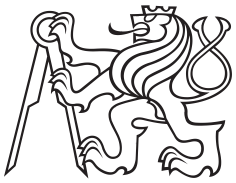
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Vytvoření aplikace pro zpracování výukových materiálů

Mariia Pasichna

Vedoucí práce: Ing. Pavel Náplava, Ph.D.
Leden 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Pasichna** Jméno: **Mariia** Osobní číslo: **495637**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Vytvoření aplikace pro zpracování výukových materiálů

Název bakalářské práce anglicky:

Implementation of an application for education materials processing

Pokyny pro vypracování:

Navrhněte a vytvořte aplikaci, která zjednoduší studentům přípravu na výuku, testy a zkoušky z existujících výukových materiálů (především slajdy a videa). Postupujte následujícím způsobem:
1) Analyzujte existující způsoby podpory výuky a typy podpůrných materiálů, které jsou v jejich rámci vytvářené a používané.
2) Vyhodnoťte silné a slabé stránky podpůrných materiálů.
3) Zaměřte se dále na slajdy a podpůrná videa. Proveďte průzkum mezi studenty, jakým způsobem se jim s těmito materiály pracuje a zda by jim při přípravě pomohla kombinace těchto materiálů v podobě aplikace, umožňující například současně hledat ve slajdech a videích, následně pak pro nalezený slajd spustit odpovídající část videa.
4) Na základě zpětné vazby aplikaci navrhněte.
5) Zvolte vhodnou platformu realizace aplikace.
6) Navrženou aplikaci vytvořte.
7) Použitelnost aplikace ověřte za pomocí uživatelských testů vybranou skupinou studentů

Seznam doporučené literatury:

1. Fleming, N.D, I'm different; not dumb. Modes of presentation (VARK) in the tertiary classroom, in Zelmer,A., (ed.) Research and Development in Higher Education [online], Proceedings of the 1995 Annual Conference of the Higher Education and Research Development Society of Australasia (HERDSA) Dostupné z: http://www.vark-learn.com/wp-content/uploads/2014/08/different_not_dumb.pdf
2. Fleming, N.D, Bonwell, C., How do I learn best?: A students guide to improved learning: VARK - visual, aural, read/write, kinaesthetic. May 2019. ISBN Number: 978-0-473-07810-2 Dostupné z: https://vark-learn.com/wp-content/uploads/2019/07/How-Do-I-Learn-Best-Sample.pdf
3. Thusi N., Costa K., Introducing NFT Model as a pedagogy using Dale's and Bloom's Taxonomy: A complementary tool for learning instruction [online]. Dostupné z: https://osf.io/rfmyp/download/?format=pdf

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Náplava, Ph.D. Centrum znalostního managementu FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **02.02.2022** Termín odevzdání bakalářské práce: **10.01.2023**

Platnost zadání bakalářské práce: **30.09.2023**

Ing. Pavel Náplava, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studentky

Poděkování

Chtěla bych poděkovat mému vedoucímu, kterým byl Ing. Pavel Náplava, Ph.D. za veškerou podporu a vedení tohoto projektu od začátku až do konce. Děkuji také mému příteli Matějovi, za to že tu pro mě byl a taky Michalce za pomoc s uvedením mé práce do gramaticky správné podoby.

Prohlášení

I declare that this work is all my own work and I have cited all sources I have used in the bibliography.

Prague, ledna 9, 2023

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 9. January 2023

Abstrakt

Hlavním cílem této bakalářské práce je zanalyzovat systém samo-výuky studentů, navrhnout a vyvinout aplikaci "ViSliCo", která zjednoduší studentům přípravu na výuku, testy a zkoušky. Vyvinutá aplikace zpracovává data z již existujících výukových materiálů, jako jsou prezentace a videozáznamy výuky. Program "ViSliCo"materiály mimo jiné rozšiřuje o titulky a možnost vyhledávání v nich pomocí textů a slajdů z prezentace.

Klíčová slova: video, prezentace, spojování, samostudium

Vedoucí práce: Ing. Pavel Náplava, Ph.D.

Abstract

The main goal of this bachelor's thesis is to analyze the self-teaching system of students to design and develop the "ViSliCo" application, which will simplify students' preparation for lessons, tests and exams. The developed application processes data from already existing teaching materials, such as presentations and video recordings of teaching. Among other things, the application expands the materials with subtitles and the possibility of searching them using texts and slides from the presentation.

Keywords: video, presentation, connecting, self-study

Title translation: Creating an application for processing educational materials

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Tato bakalářská práce se zabývá efektivitou (samo)vzdělávání se a tím, jaké množství informací lidé získávají z různých druhů materiálů, jako jsou videozáznamy a prezentace. Nápad na tuto práci jsme dostali během pandemie covid-19, kdy byly zavřené školy a veškerá výuka probíhala online. V tuto dobu začaly ve velké míře vznikat video záznamy z přednášek a cvičení, které pak sloužily a i nadále slouží, jako cenný zdroj informací k přípravě na zkoušky a učení se obecně.

Cílem naší práce je zanalyzovat potřeby studentů vysokých škol při samostudiu a prozkoumat existující podpůrné materiály a jejich použitelnost při studiu. Dále pak vyhodnotit silné a slabé stránky těchto existujících podpůrných materiálů, konkrétně videozáznamů a prezentací, a s pomocí dotazníků vyplněných studenty navrhnout aplikaci, která by jim pomohla se samostudiem a získáváním dat z těchto materiálů.

Na základě zpětné vazby pak navrhneme aplikaci, která by tyto potřeby studentů splňovala a která by mohla pomoci studentům při přípravě na výuku s používáním existujících výukových materiálů. V rámci návrhu aplikace vybrat vhodnou platformu a prostředky potřebné k její realizaci.

Následně pak daný program vytvořit a jeho funkcionalitu a použitelnost ověřit na skupince studentů.

Kapitola 2
Teoretické podklady

Učení je proces osvojování si nového porozumění, znalostí, chování, dovedností, hodnot, postojů a preferencí.[1] V této kapitole jsem se snažila zjistit, jaký způsob učení se je nejefektivnější. A taky se zabývala již existujícími výzkumy na téma vnímání různých studijních materiálů lidmi.

2.1 Výzkumy

Jsou různé teorie o tom jaký způsob učení je nejlepší.

Například existuje teorie učebních stylů VARK, která je popsaná v knize [Fle95], podle které se lidi klasifikují podle toho, jaký styl výuky jim jde nejlépe (Visuál, Aural, Read/Write and Kinesthetic, jak můžeme vidět na obrázku 2.1) a tvrdí, že každý má svůj nejlepší přístup k učení se.

Vizuály (V): Mají rádi informace ve formě grafů, tabulek a vývojových diagramů. Někdy si kreslí mapy svých učebních sekvencí nebo vytvoří vzorce informací.

Sluchový (A): Studenti preferují informace, které jsou mluvené nebo slyšené. Pro osoby s touto preferencí je důležité činit prohlášení a používat otázky.

Lidé čtenáři/zapisovatelé (R): Používají tištěné slovo jako nejdůležitější způsob předávání a přijímání informací.

Kinestetik (K): Prožívá své učení s využitím všech možných smyslů, včetně hmatu, sluchu, čichu, chuti a zraku. Preferují „skutečné“ věci, které se staly.

Detailnější popis ke každému z typů je možné najít v tomto zdroji: [Lea].

Podle N. D. Fleminga, muže který daný model navrhl, padesát procent středoškolských a vysokoškolských studentů využívá smíšený učební styl. Tento styl prakticky kombinuje dva až všechny čtyři předchozí učební styly. Což je popsáno v této publikaci: [NF19].

Existují i jiné průzkumy na toto téma, jako například Cone of Experience ("kužel zkušenosti"nebo "kužel učení") Edgara Dalea. Informace o kuželu se většinou odkazují na třetí vydání knihy Dalea (1969) “Audiovisual methods in teaching”, kterou jsem bohužel na internetu nenašla v originále, nicméně na internetu je několik interpretací této informace.

[1] Source text: Wikipedia (`https://cs.wikipedia.org/wiki/UÄDenÃŋ`)

zdroj:[Nic]

Obrázek 2.1: znázornění VARK modelu

- První interpretace je popsaná například tady [Tha15] a tvrdí, že si člověk po dvou týdnech pamatuje:
  - 10% z toho, co přečte
  - 20% z toho, co uslyší
  - 30% z toho, co uvidí
  - 50% z toho, co uvidí a uslyší
  - 70% z toho, co řekne a napíše
  - 90% z toho, co udělá.

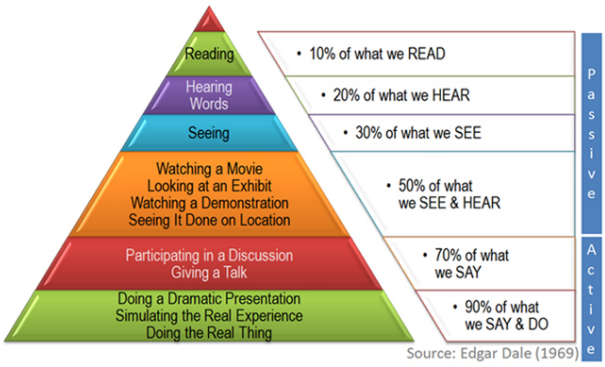
  Vizualizaci dané teorie můžeme vidět na obrázku 2.2

  Takže pokud chceme, aby si studenti zapamatovali informace co nejvíce a dlouhodobě, měli bychom je nechat „navrhnout/provést prezentaci“ nebo by měli „udělat skutečnou věc“.

  Učitelé by také měli studentům pomáhat tím, že je nechají účastnit se praktických workshopů. Je mnohem cennější sledovat ukázku než sledovat teoretické videa.

- Druhá interpretace je popsaná například v tomto článku [DVMSPBME16] a zhruba odpovídá myšlence druhého vydání Dáleho knihy [Dal54]. Tato interpretace však prezentuje jiný názor než ta první.

  Podle té je uspořádání v kuželu založeno na abstrakci a na počtu zapojených smyslů.



zdroj: [S.20]

Obrázek 2.2: kužel zkušenosti E. Dale

Podle jednoho z principů při výběru a používání vyučovacích strategií platí, že čím více smyslů je zapojeno do učení, tím efektivnější učení bude, ale to neznamená, že konkrétní zkušenost je jedinou efektivní zkušeností, kterou by pedagogové měli používat při předávání znalostí studentovi. Dale nikdy neřekl, že konkrétnější zkušenosti jsou lepší než abstraktnější. Věřil, že všechny přístupy by mohly a měly být použity v závislosti na potřebách studenta.

2.2 Materiály

Ve škole se učitelé snaží poskytnout studentům různé druhy dostupných materiálů tak, aby si každý vybral to, co je mu nejvíce příjemné, nebo co nejvíce potřebuje. Každý z využívaných materiálů při výuce má svoje plusy (+) a mínusy (-):

- Videozáznam

  \+ Formát je lehký na vnímání. Obsahuje všechno, co říkal učitel, a tak koncentruje všechny potřebné informace ke zkoušce. Student si ho může zastavit, vrátit, přehrávat v pohodlné rychlosti. Může obsahovat praktickou ukázku. Videa jsou lehká na vytvoření tím, že stačí hodinu jenom nahrát.

  \- Ne u všech předmětů jsou dostupné videozáznamy. Je časově náročnější hledat konkrétní menší informace ve videu v porovnání s jinými materiály.

- Audionahrávka

  \+ Formát je lehký na studování. Obsahuje všechno, co říkal učitel, a tak koncentruje všechny potřebné informace ke zkoušce. Student si ho může zastavit, vrátit, přehrávat v pohodlné rychlosti. Může obsahovat praktickou ukázku. Jsou lehká na vytvoření tím, že stačí hodinu jenom nahrát.

- Málokdy je nahrávají učitelé. Je časově náročnější hledat konkrétní menší informace v audiozáznamu v porovnání s jinými materiály. Chybí vizuální prvky.

- Slajdy

  + Většinou jsou veřejně dostupné u všech předmětů. Jsou rychlé na vyhledávání informací, které jsou v nich uvedeny. Obsahují důležité termíny.

  - Často málo informací. Jsou různorodé (v závislosti na autoru).

- Ukázkové příklady

  + Pomáhají s pochopením toho, jak se využívají informace v praxi.

  - Většinou dávají pouze úzký přehled (nevysvětlují proč to tak je, jak to funguje, nebo širší použití)

- Materiály navíc a jiné zdroje

  + Pomáhají podívat se na látku pod jiným úhlem. Je možné je použít pro samostudium.

  - Dávají látku v širším rozsahu, než je potřeba pro školní účely.

V souhrnu všechny průzkumy říkají, že neexistuje jediný nejlepší způsob učení se pro všechny, ale každý student má vlastní způsoby učení se a lepšího zapamatování si informací. Učitelé by se měli snažit o to, aby studenti měli dost materiálů, a taky o zapojení maximálního počtu smyslů na hodinách a v úkolech během semestru.

Kapitola 3

Vlastní průzkum studijních preferencí studentů

V této kapitole jsem se snažila zjistit studijní preference studentů v současné době. Jaká forma studia jim nejvíce vyhovuje, jak se připravují během semestru a jak se učí před zkouškami. Udělala jsem pár dotazníků, kterých se zúčastnilo 25 studentů z různých technických škol v České republice, kteří se účastní výuky prezenční formou.

3.1 Výsledky dotazníků

V prvním dotazníku byly otázky, kde respondenti odpovídali ohledně svých preferencí ohledně studia.

Většina respondentů (72%) odpověděla, že chodí na přednášky nebo je kombinují se záznamy. Cvičení ale na druhou stranu navštěvuje všech 100% respondentů.

Při seřazení způsobů učení se podle efektivity vyšly následující výsledky:

1. místo - praktická ukázka
2. místo - video
3. místo - prezentace
4. místo - text
5. místo - audio

Přesné výsledky jsou vidět na obrázku 3.1

Dále skoro všichni respondenti (89%) odpověděli, že se připravují ke zkouškám podle prezentací. Nicméně když něco ze slajdu neví, tak většina (56%) začne hledat na internetu a jenom 28% studentů se dívá na videozáznam.

Na otázku, proč data nehledají v přednáškách (podle kterých se učí během semestru), jsem dostala následující odpovědi:
“Trvá než najdu správnou odpověď.”
“Nemám jednoduchý způsob jak najít hledanou informaci ve videu.”
“video ze skoly nemaji timestampy pro rychle vyhledavani”

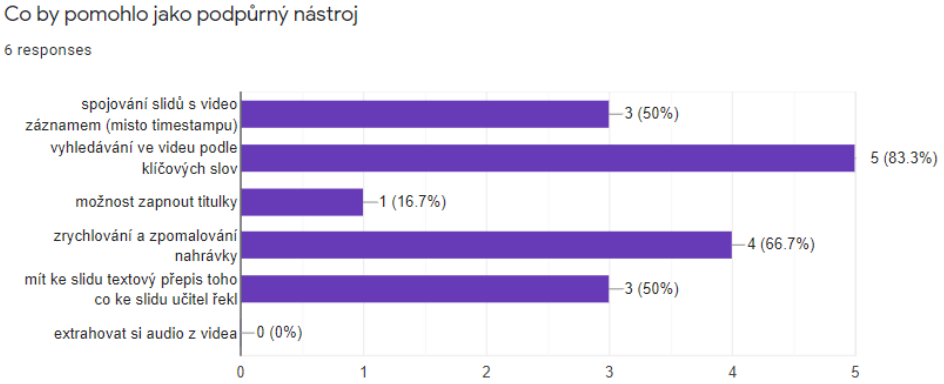
Obrázek 3.1: výsledek dotazníku

"Zdlouhavé. Kdyby člověk měl časovou osu jako na ytb, tak pak by moje první možnost byl videozáznam".

Pak jsem udělala jiný více detailní průzkum a poslala jsem ho 6 vybraným studentům, kteří dříve uvedli, že používají slajdy a videozáznamy při samostudiu. Otázky v dotazníku byli zaměřené hlavně na funkcionalitu, která by jim s tím pomohla.

Většina z nich (5 lidí) přiznali, že jim nejvíce komplikuje život při samostudiu podle slajdů a videozáznamů časová náročnost hledání informací ve videu a jeden odpověděl, že mu nic nevadí.

Při výběru funkcionality, o které si myslí, že jim se studiem pomůže, byli získané následující odpovědi 3.2:



Obrázek 3.2: výsledek z dotazníku

Dále všichni potvrdili, že by používali program s danou funkcionalitou.

Z odpovědí studentů na daný průzkum a podle studijních teorií vyplývají následující požadavky pro aplikace na podporu studia z videozáznamů:

- vyhledávání ve videu podle klíčových slov
- spojování slajdů s videozáznamem (místo timestampu)
- možnost podívat se celkově na text ke slajdu, co řekl učitel
- možnost zapnout titulky [1]

[1] V průzkumu nebyl o titulky velký zájem, ale to bude způsobeno tím, že jsem se ptala

- zrychlování a zpomalování nahrávky

Průzkum potvrzuje, že by se hodilo mít program, kde si studenti budou moct snadno vyhledávat informace z videozáznamů přednášek a cvičení, který bude obsahovat výše popsanou funkcionalitu. A tak jsem se pokusila najít takovou aplikaci.

hlavně česky/slovensky mluvících studentů. Student, který o titulky zájem potvrdil, nemá češtinu/slovenštinu jako mateřský jazyk, a proto by tato funkcionalita mohla být pomocná spíše cizincům.

Kapitola 4
Hledání alternativy existujících řešení

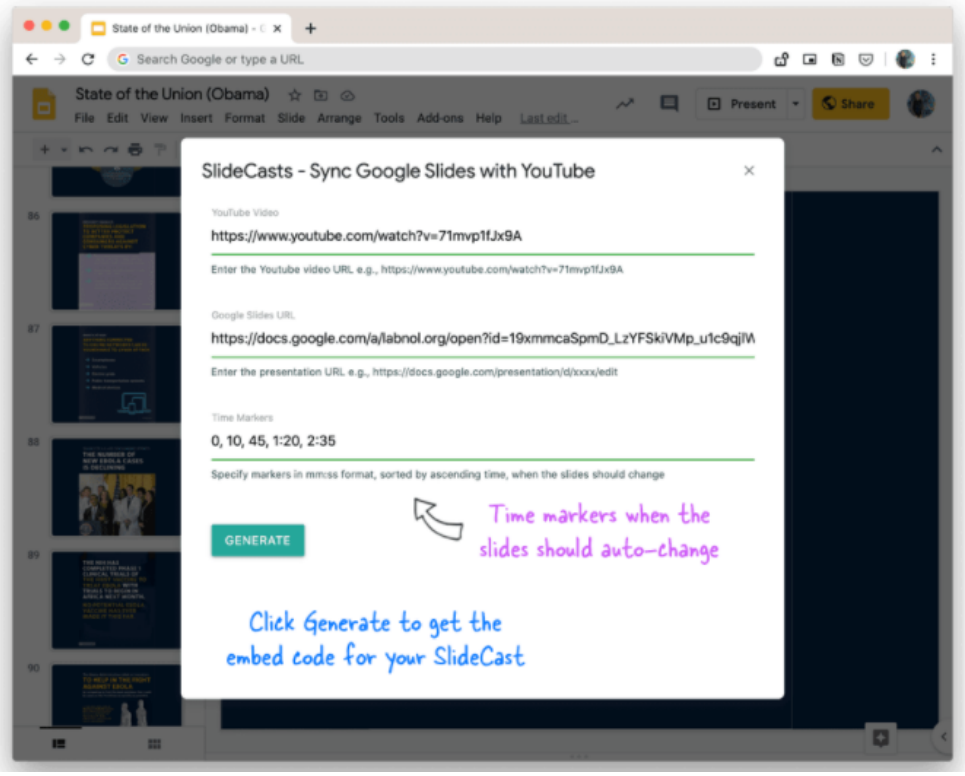
V této kapitole jsem se snažila najít již existující programy, které by úplně, nebo alespoň částečně odpovídaly našim požadavkům, které jsem uvedla v minulé kapitole.

4.1 Hledání alternativy

Hledání již existujících programů jsem prováděla v Google vyhledávači. Podle slovních kombinací:

- "search in video"
  Zde mi Google nabídl hodně různých video editorů, a některé z nich obsahovaly i možnost vytvářet titulky a hledat přes ně ve videu nebo zrychlování. A proto jsem se v dalších hledáních zaměřila hlavně na spojování slajdů se záznamem.

- "merge slides with video"
  Zde byli nástroje, do kterých uživatel může nahrát video, prezentaci a pak je ručně pomocí timestampu spojit, což ale my chceme mít automatizované. Screenshot nastavení jednoho z takových programu je vidět na obrázku 4.1



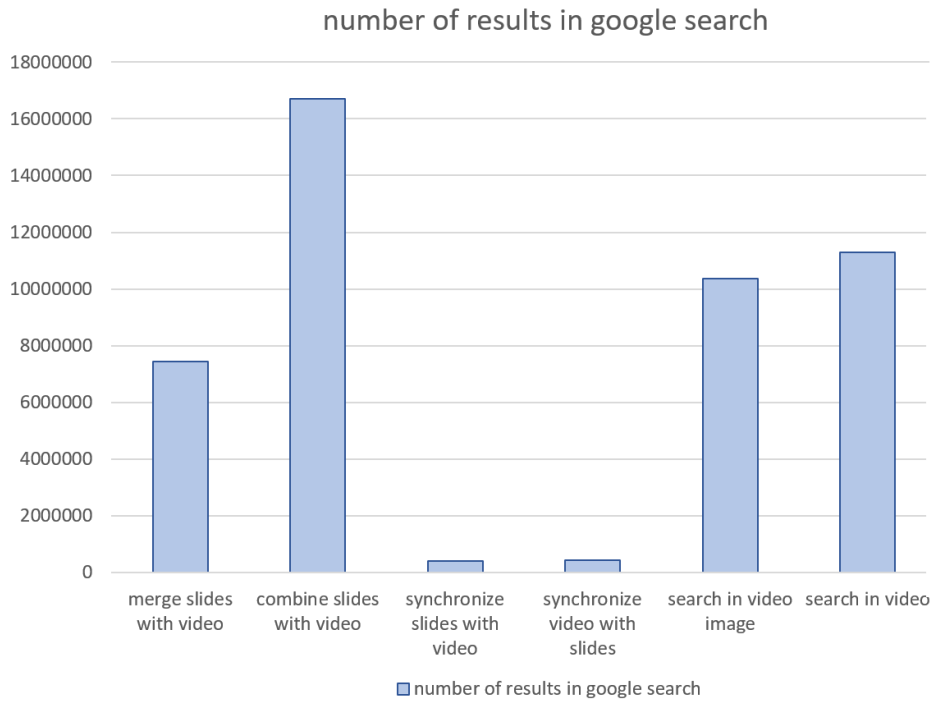
Obrázek 4.1: znázornění funkcionality nalezeného programu

- "combine slides with video"
  Zhruba stejný typ výsledků jako “merge slides with video”, ale více nástrojů.

- "synchronize slides with video"a "synchronize video with slides"
  Stejný typ výsledků jako v “combine slides with video” a “merge slides with video”.

- "search in video image"
  Tady byly nalezeny nástroje a články o hledání videa na internetu podle screenshotu z něho.

Diagram s počtem nalezených výsledků v Googlu je vidět na dalším obrázku 4.2.



Obrázek 4.2: diagram s počtem nalezených výsledků v Google

Při diskuzi s vedoucím mé práce, mi pověděl o již existujícím programu “SlidesLive”, který vytvořili studenti ČVUT. Nicméně při detailnějším prozkoumání jsem objevila, že daný program, pouze zaznamenává vaše snímky s hlasem a poté automaticky synchronizuje vaši prezentaci s videem nebo zvukem. Takže se jedná o skvělý nástroj pro vytváření nové prezentace, tak abyste nemuseli přepínat slajdy při prezentaci, ale je to něco naprosto odlišného od našich požadavků.

I po hledání programu, který by vyhovoval našim požadavkům, na internetu a po komunikaci s lidmi v mém okolí, kteří by takový program mohli znát, jsem nenašla žádný program, který by splňoval danou funkcionalitu. A proto jsem se rozhodla vytvořit nový program jménem "ViSliCo"(Video Slides Connector).

Kapitola 5

Návrh a analýza architektury aplikace

V této kapitole jsem vytvořila základní návrh programu "ViSliCo"a analyzovala jeho funkcionalitu a způsoby implementace.

5.1 Technologie a provedení

Na základě požadavků na aplikaci "ViSliCo"se musíme rozhodnout, o jaký typ aplikace se bude jednat.

Jedním z řešení by mohla být webová aplikace, nicméně v takovém případě by se proces zpracování videa zbytečně prodlužoval kvůli nahrávání videí a dat na server, a navíc by při velkém počtu uživatelů bylo potřeba zvětšovat i servery a řešit škálovatelnost. S počítačovou aplikací se vyhneme těmto problémům.

Dále pak, jelikož je aplikace určena k opakovanému spouštění nad stejnými daty, budeme potřebovat aby si již jednou zpracovaná data ukládala do lokální paměti uživatele, dále jen cachovala, pro opětovné spuštění. Proto bude nejlepším řešením vytvořit desktopovou aplikaci, která si bude cachovat již zpracovaná videa a při opětovném spuštění nad daným videem si pouze načte již dříve vygenerovaná data z lokální paměti.

5.1.1 Technologie

Při rozhodování, pomocí jakých technologií budu aplikaci psát, jsem se rozhodovala mezi Javou a Pythonem. S Javou mám sice více zkušeností a je pro ni velká podpora co se grafického rozhraní týče. Nicméně Python má mnohem širší nabídku knihoven pro zpracovaní multimediálních dat a je to jazyk, ve kterém se vytváří a jsou napsány neuronové sítě a modely, které budu při vývoji nucena použít. Dále jsem pak vyhodnotila, že pro mě bude snadnější využití a zapouzdření knihoven Pythonu do vytvořeného programu.

Uvažovala jsem i o použití jiných jazyků, jako například C# a jejich knihoven, ale ve většině případů jsem objevila pouze menší knihovny, které ne přesně odpovídaly mým požadavkům, a to buď kvůli tomu že, neposkytovaly veškerou funkcionalitu, ale i kvůli tomu, že buď byly zpoplatněné, nebo jejich budoucí podpora nebyla nijak zaručena.

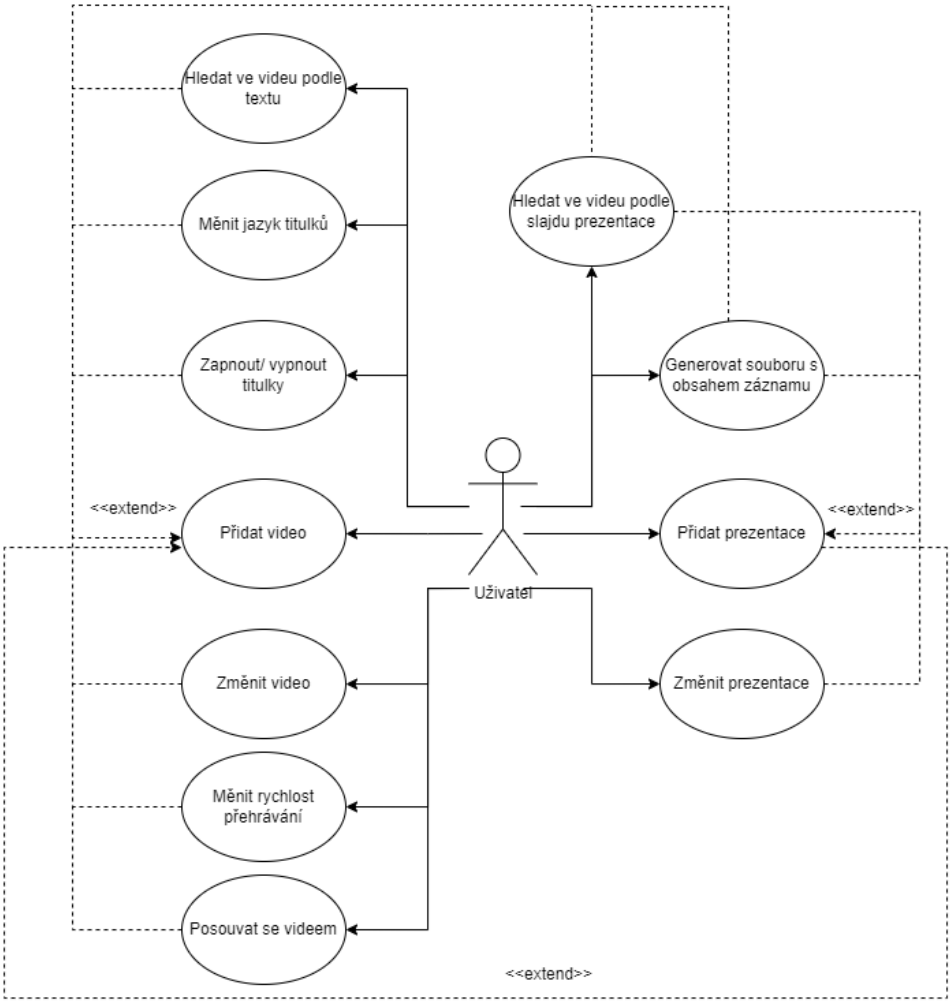
5.2 Funkcionalita

Na základe požadavků z dotazníku v podkapitole 3.1 jsem vytvořila jednoduchý use case diagram, který je vidět na obrázku 5.1.

V dotazníku jsme zjistili, že ne všichni uživatelé chtějí používat funkci spojování videozáznamu s prezentací, a proto uživatel bude moct použít generování titulků, překlad titulků a vyhledávání ve videu pomocí textu bez přidání prezentace. Pokud uživatel bude chtít využít spojení prezentace s videem nebo generování dokumentu, musí k videu nahrát i prezentaci, která je použita ve videu.

Hlavní funkcionalitou programu "ViSliCo"pak bude:

1. program umožní nahrát videozáznam
2. program vygeneruje titulky pro video
3. program bude zobrazovat titulky během přehrávání videa
4. program umožní textové vyhledávání ve videu
5. program umožní přeložit titulky do uživatelem vybraného jazyka
6. program umožní zobrazit titulky ve vybraném jazyce
7. program umožní přidat prezentaci k videozáznamu
8. program spojí slajdy z prezentace s videozáznamem
9. program umožní vyhledávání ve videu pomocí slajdů
10. program bude umět vygenerovat dokument složený ze slajdů a textu



Obrázek 5.1: use case diagram

V aplikaci "ViSliCo"není potřeba dělit uživatele na různé role. Všichni uživatelé používají v programu pouze svoje lokální data, a po nahrání potřebných materiálů používají všechna vygenerovaná data.

5.3 Algoritmus

V této kapitole si rozdělíme program na jeho hlavní části a navrhneme si, jak je budeme později implementovat.

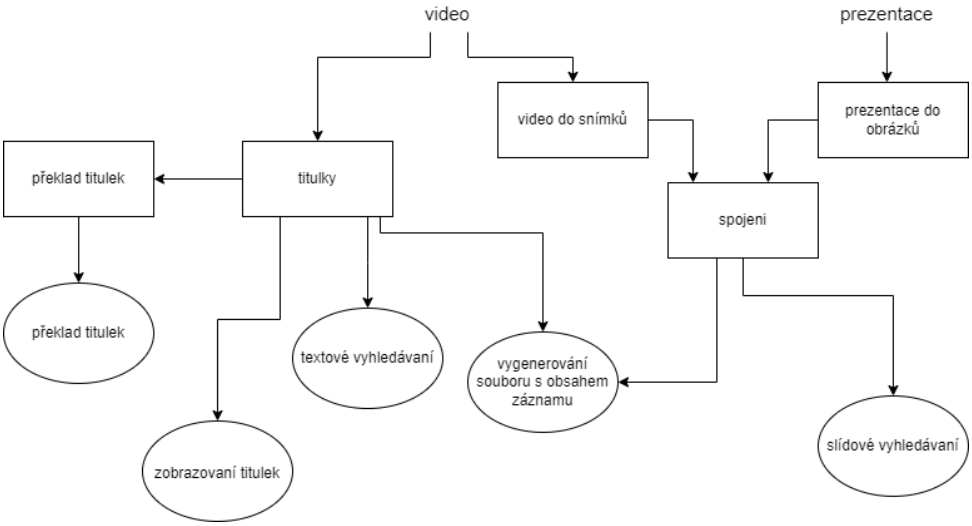
I přesto, že jsem se snažila najít nějaký již existující nástroj, který by nám vygeneroval spojení videa s prezentací, nepodařilo se mi to, a proto jsem se rozhodla, že spojení budu muset vygenerovat sama. Kvůli generování spojení si budu muset upravit běh programu a předpřipravit některá data jako jsou například snímky z videa a slajdy z prezentace.

Samotný běh a zpracování vstupních materiálů pak navrhneme takto:

1. uživatel nahraje videozáznam

   a. program umožní přehrávání videozáznamu
   b. program vyextrahuje framy pro pozdější generování spojení s prezentací
   c. program vygeneruje titulky pro video a umožní jejich zobrazení
2. program zpřístupní uživateli vyhledávání pomocí textu titulků
3. program zpřístupní uživateli překlad titulků a jejich zobrazení
4. uživatel nahraje do programu prezentaci
   a. program vygeneruje slajdy prezentace jako obrázky pro spojení s videem
   b. program najde slajdy v obrázcích z videa a vytvoří si spojení slajdu s jejich výskytem ve videu
5. program zpřístupní uživateli vyhledávání pomocí slajdů z prezentace
6. program zpřístupní uživateli možnost vygenerovat si soubor se slajdy a textem z videa

Samotný proces a zpracování dat se pak bude skládat z následujících částí, které se budou vykonávat paralelně mezi sebou a i s přehráváním videa, jak můžeme vidět na obrázku 5.2.



Obrázek 5.2: diagram zpracování dat programem

5.3.1 Generování titulků

Jelikož jedním z výstupů programu "ViSliCo"jsou titulky k videu, musíme vytvořit modul pro generování titulků z videozáznamu.

Existuje hodně nástrojů na generování textu z audio nebo videozáznamu, nicméně ne všechny nástroje generují přímo titulky, tak abychom je mohli

rovnou použít, a některé z nástrojů nefungují s českým jazykem. Nejobsáhlejší funkcionalitu, kterou bychom mohli použít nabízí `Mozilla DeepSpeech` a `AutoSub`.

Podle článku [Tal20] `Mozilla DeepSpeech` zvládne rozpoznávání textu lokálně již do několika minut, nicméně tento program má problémy s nerodilým anglickým přízvukem. A také nemá dostatečně kvalitní rozpoznávání češtiny. Řešením tohoto problému by mohlo být naučit daný model rozpoznávání potřebného jazyka podle vlastních dat.

`Autosub` podporuje různé vstupní a výstupní jazyky včetně češtiny. Dále `Autosub` obsahuje možnost překladu titulků do jiného jazyka, ale jenom s Google Translate API klíčem, který je placený. Takže na tuto funkcionalitu budeme potřebovat jinou technologii. Dalším mínusem je, že pro generování titulků potřebuje `Autosub` internetové připojení.

Vzhledem k tomu, že není problém, aby uživatelé byli připojeni k internetu při prvním sledování videa a pak už použili titulky z cache, a protože úkol učení modelu je celkem obsáhlý, rozhodla jsem se používat `Autosub`.

5.3.2 Překlad titulků

Jak jsem již zmínila výše, `Autosub` sice poskytuje možnost přeložit titulky, ale tato funkcionalita je placená, proto jsem se rozhodla využít k tomuto účelu jiných prostředků.

Při hledání knihoven, které by podporovaly velmi širokou škálu jazyků včetně češtiny, jsem objevila, že většina takových knihoven využívá v základu Google translator. Proto jsem se rozhodla použít přímo oficiální překladové API od Google - `Googletrans`, s tím že věřím, že bude i nadále podporované a funkční v budoucnosti. Hlavní nevýhodou `Googletrans` je, že stejně jako `Autosub` potřebuje k fungování internet, ale i tady uživatel může využít možnost si data dopředu nacachovat.

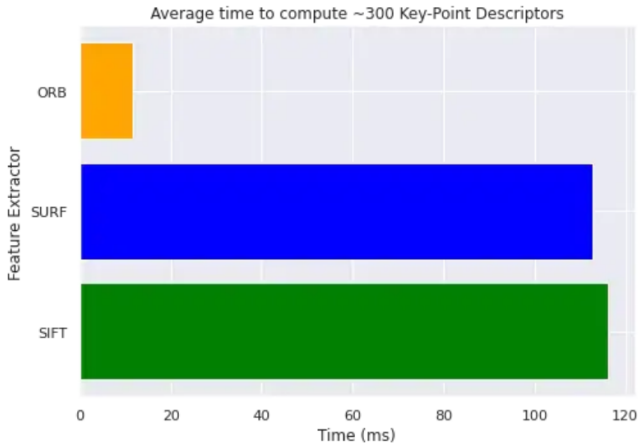
5.3.3 Spojení slajdu s video framy

Pro spojovaní videa s prezentací si vygenerujeme snímky z průběhu videa a z prezentace, a spojíme je tak, jak k sobě patří. V této kapitolce si nyní představíme základní možnosti, jak spojit slajd se snímkem z videa a připravíme si možné způsoby, jak toho dosáhnout.

Detekce vlastnosti obrázku

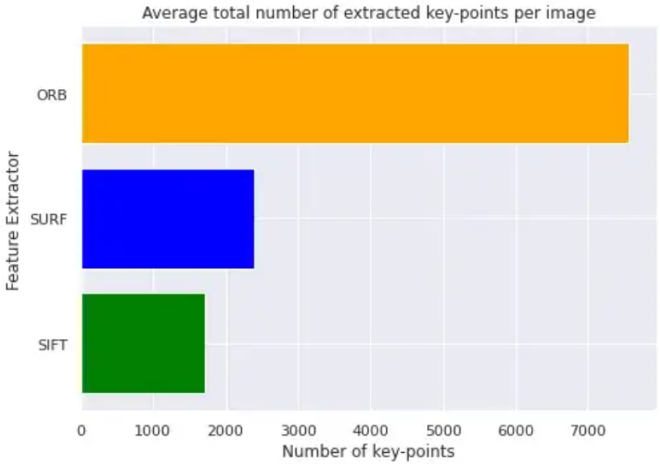
První ze způsobů porovnání dvou obrázků je pomocí detekce a porovnání jeho rysů, jako jsou například hrany a rohy. Knihovna `OpenCV` obsahuje několik různých způsobů, jak detekovat rysy obrázku.

Jak můžeme vidět na obrázcích 5.3 a 5.4 `ORB` má nejlepší výkon a nejrychlejší výpočet, navíc umí poznat změny rozměru, rotace a je odolný vůči silnějšímu jasu. Což ho dělá nejlepším kandidátem mezi algoritmy v této kategorii.



zdroj: [Ken21]

Obrázek 5.3: diagram rychlosti výpočtu



zdroj: [Ken21]

Obrázek 5.4: diagram celkového počtu detekovaných klíčových bodů

Měla jsem od daného algoritmu celkem velká očekávaní, nicméně při jeho vyzkoušení na studijních materiálech mě zklamal. `ORB` je velice rychlý, ale porovnání framů se slajdy bylo hodně nepřesné. Často docházelo k situaci, kdy slajd, který ke snímku nepatřil, měl větší shodu se snímkem než ten odpovídající. Proto jsem se rozhodla tento způsob porovnání nepoužít.

Detekce shod pomocí barev

Dalším ze způsobů, jak ověřit podobnost obrázků, je pomocí detekování jejich barevné shody.

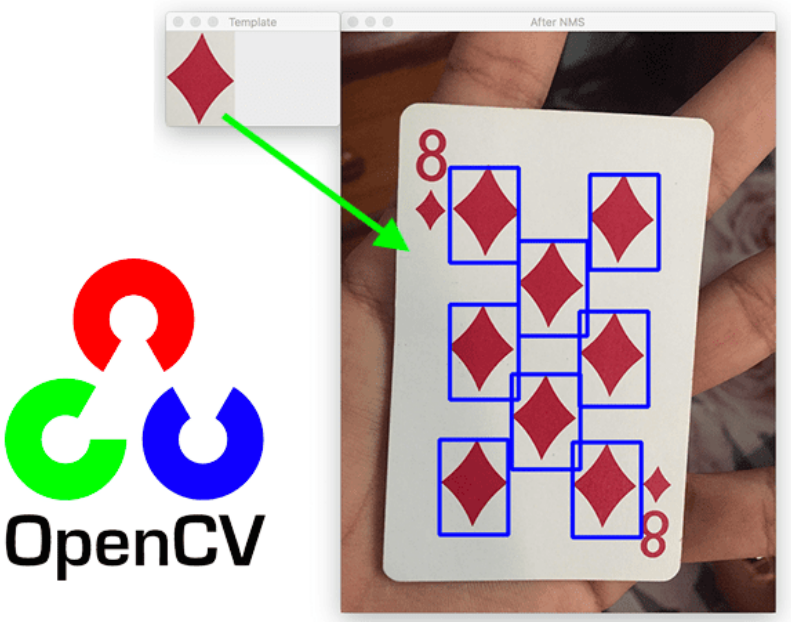
`OpenCV` nám na to poskytuje funkci *calcHist()* pro výpočet histogramů obrázků, kterou můžeme použít k výpočtu histogramu barevných kanálů (modrý, zelený a červený) obrázku. Tato funkce je poměrně rychlá, ale spíš

jen doplňková, protože jen podle jejich výsledků nemohu přesně spojit slajd se snímkem.

Detekce shod pomocí porovnání šablon

Další zajímavá funkcionalita, kterou nám poskytuje `OpenCV` je porovnání šablon. Daná funkcionalita nám definuje místa, kde se vyskytuje první obrázek ve druhém, jak je ukázáno na obrázku 5.4. Pro porovnání se nazývá funkce *matchTemplate()*, která přijímá dva obrázky a metodu. Metody se mezi sebou liší jenom vzorečkem pro výpočet. Po vyzkoušení jsem dostala nejlepší výsledky s metodou *TM_SQDIFF_NORMED*.

Hlavní nevýhodou dané operace je, že obrázky musí být stejné velikosti.



zdroj: [Ros21]

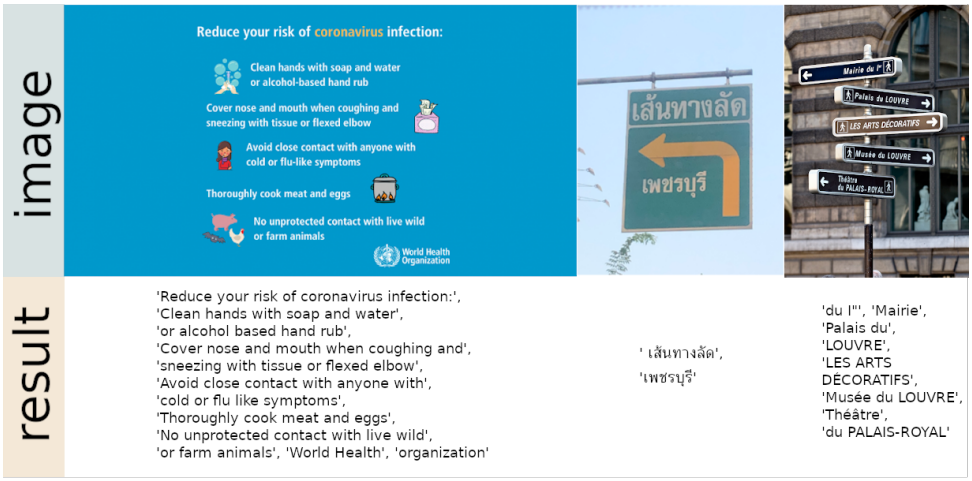
Obrázek 5.5: příklad výsledku detekce shod

Detekce shod pomocí textu

Další varianta pro porovnání obrázků je v našem případě pomocí porovnání textu, který snímky obsahují. Tedy pomocí optického rozpoznávání znaků čili OCR (Optical Character Recognition). Příklady výsledků jsou zobrazeny na obrázku 5.6.

Tady z Python knihoven, které podporují OCR, obsahující češtinu a případně fungující s horší viditelností textu, jsou dvě: `EasyOCR` a `PyTesseract`.

`EasyOCR` je skvělý příklad knihovny pro detekování textu na obrázku, který umí detekovat text pod různými úhly, podporuje většinu jazyků a lehce se instaluje do Pythonu. Nevýhodou je, že pokud je knihovna spuštěná pouze přes CPU, je velmi pomalá.



zdroj: [Jai]

Obrázek 5.6: příklad použití OCR

`PyTesseract` má stejné výhody jako `EasyOCR`, až na instalaci. Pro instalaci `PyTeseractu` na platformě Windows je potřeba si stáhnout soubory z jejich oficiálního webu.

Nicméně po osobním vyzkoušení jsem zjistila, že `PyTesseract` funguje výrazně rychleji než `EasyOCR` a výsledky na testovaných datech jsou zhruba stejně kvalitní.

Zvolený postup

Po vyzkoušení různých typů spojování na studijních materiálech, jsem dospěla k závěru, že žádná z metod sama o osobě nedává dostatečně dobrý výsledek (kvalitní spojení za krátkou časovou dobu), a tak jsem vyzkoušela spojit několik ze zmíněných funkcionalit spolu. Nejefektivnější a nejrychlejší se ukázalo spojení detekce shod pomocí textového porovnání, porovnání šablon a detekce shod pomoci barev. Takže na implementaci spojování budu používat modul `PyTesseract` a *matchTemplate()*, *calcHist()* od `OpenCV`.

5.4 GUI frameworky

Jelikož aplikace ViSliCo bude mít grafické rozhraní, musíme vybrat v čem ho budeme implementovat. Python obsahuje různé knihovny pro grafické rozhraní. Tady jsem vybrala a analyzovala ty nejpopulárnější.

Tkinter

`Tkinter` je jednoduchá knihovna, která by měla fungovat na všech platformách.

Podle [Amn20] `tkinter` nemá spolehlivé uživatelské rozhraní a občas ho může být složité debugovat. Dále pak, na platformě Windows vypadá zastarale.

PyQt

`PyQt` je taky plně multiplatformní a nabízí modul `QtDesigner` pro vytváření grafického uživatelského rozhraní. Další věcí, která mě zaujala je, že pro tuto knihovnu je hodně videotutoriálů. Zejména i na to, jak vytvořit videopřehrávač.

Nevýhodou této knihovny, podle zdroje [Amn20], je nedostatečná dokumentace pro `PyQt5` a tedy i to, že pochopení knihovny zabere spoustu času.

Kivy

Framework `Kivy` se používá hlavně pro vývoj mobilních aplikací, ale podporuje více platforem. Vývoj GUI v `Kivy` by měl být snadný a rychlý, ale nevýhodou je, že uživatelské rozhraní vypadá neznámě a z tohoto důvodu je pro uživatele obtížné platformu používat.

wxPython

`wxPython` je multiplatformní framework a má uživatelský friendly rozhraní.

Jeho nevýhody, podle zdroje [Tak19], jsou, že `wxPython` vyžaduje samostatné stažení, což může být obtížné pro uživatele při nasazení aplikace a taky to, že `wxPython` má chyby.

PySimpleGUI

`PySimpleGUI` je také multiplatformní a v daném zdroji [Cha22] říkají, že je velmi snadný na učení ve srovnání s jinými GUI.

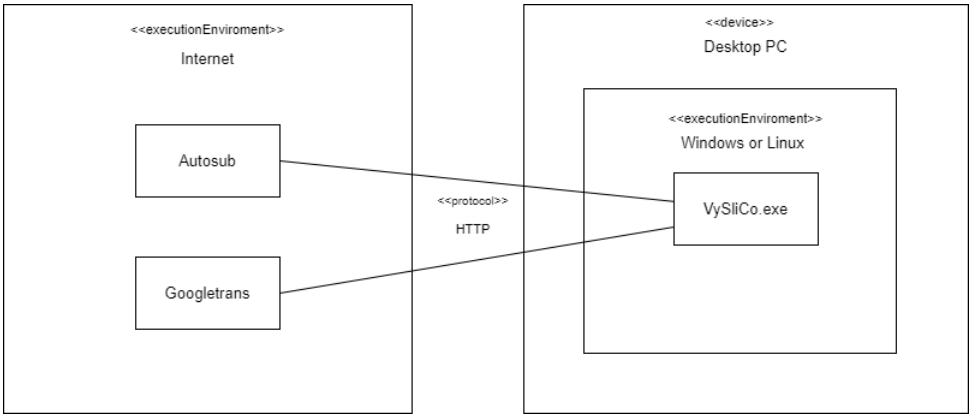
Snad jediným jeho mínusem, který jsem našla, je to, že je o něm výrazně míň informací na stránkách jako je Stack Overflow a videotutoriálů, kvůli čemu se bojím, že v případě problému bude složitější najít řešení. Ale domnívám se, že to je hlavně kvůli tomu, že se jedná o poměrně novou technologii, jelikož `PySimpleGUI` byl vytvořen v roce 2018.

5.4.1 Vybraný framework

Z analýzy jsem si vybrala `PyQt` jako framework pro implementaci GUI, a to kvůli dostatečnému množství informací o `PyQt` v prostorech internetu a taky modulu `QtDesigner`, díky kterému věřím, že proces vytváření vzhledu aplikace pro mě bude přehlednější a snad i rychlejší.

5.5 Diagram nasazení

Na obrázku 5.7 je vidět diagram nasazení. Aplikace bude běžet lokálně na počítači, až na výjimku `Autosubu` a `Googletrans`, které potřebují mít připojení k internetu pro správné fungováni funkcionality vytvoření a přeložení titulků.



Obrázek 5.7: diagram nasazení

5.5.1 Závěr

Technicky vidíme, že práce je uskutečnitelná. Nyní máme vybrané moduly pro veškerou funkcionalitu a můžeme začít s implementací.

Kapitola 6
Implementace

Tato kapitola popisuje implementaci a použité technologie programu "ViSliCo".

6.1 Funkcionalita backendu a její implementace

V této části detailněji probereme technologii a postupy, které jsme použili pro implementaci backendu.

6.1.1 Generování titulků

Ke generování titulků jsme použili knihovnu `Autosub`. Při práci s daným nástrojem jsme objevili následující omezení: `Autosub` je vybíravý vůči jménu videa, a to tak, že ve jméně videa nemůžou být mezery nebo lomítka. Jelikož si náš program neukládá video u sebe, nemůžeme měnit jeho cestu a ani jméno. Proto si před použitím `Autosubu`, pomocí nástroje `Moviepy` vygeneruje z videozáznamu dočasný audio soubor, který pak uložíme pod námi zvoleným jménem a máme nad ním kontrolu. Po vygenerování a uložení titulků audiozáznam hned smažeme.

Dalším omezením `Autosubu` je, že potřebuje definovat jazyk původu. Proto pokud uživatel přidává nové video, naše aplikace se hned po načtení uživatele zeptá na jazyk řeči v záznamu.

Jednou z největších nevýhod `Autosubu` je jeho časová náročnost generování titulků, která je mimo jiné způsobena hlavně komunikací přes internet. Celkový proces generování titulků pomocí `Autosubu` pak pro 1,5 hodinové video zabere zhruba 5 minut. Jelikož generování titulků trvá celkem dlouho, na to aby uživatel jenom čekal, spouštíme generování v novém vlákně. Nově vzniklé titulky jsou uloženy do cache v souboru s názvem *generic* ve formátu `srt`, který pak používáme jako základní soubor titulků pro toto video.

6.1.2 Překlad titulků

Titulky překládáme ze souboru *generic* do jazyka definovaného uživatelem. Překlad probíhá tak, že v prvním kroku proběhne načtení titulků do Pythonu a to pomocí knihovny `srt`. Dále pak vezmeme text z naparsovaného objektu

a postupně ho přeložíme do zvoleného jazyka pomocí modulu `Googletrans`. Nově vzniklé titulky si uložíme do cache.

Jelikož generování titulek je operace, která je svou délkou závislá na délce videa a mohla by zabrat i několik minut, je i generování samotné spuštěno v novém procesu.

6.1.3 Textové vyhledávaní

Pro textové vyhledávaní se používají titulky uživatelem zvoleného jazyka a nebo pokud nejsou ještě k dispozici, použijí se titulky ze souboru *generic*. Po zadání uživatelem hledaného textu do vyhledávacího políčka program naparsuje soubor s titulky a jako výsledek vrátí časy bloků titulků, které obsahují hledanou frázi. V aktuální implementaci program vyžaduje absolutní shodu hledaného textu s částí textu titulek.

6.1.4 Prezentace do obrázků

Při původním návrhu jsme plánovali, že program bude zpracovávat prezentace ve formátu `pptx` a `pdf`, ale během implementace jsme zjistili, že vygenerovat obrázky z `pptx` je dosti problematické.

Při řešení tohoto problému jsme objevili, že existuje několik knihoven, které sice generují obrázky z pptx, nebo převádí `pptx` do `pdf` formátu, ale mají následující omezení:

- Knihovna `Aspose` je placená a funkcionalita, kterou poskytuje zdarma je omezena a generované obrázky z `ppt` a `pptx` jsou ve špatné kvalitě s vodoznakem.
- Knihovna `pywin32` poskytuje přístup k Windows API a obsahuje funkcionalitu, kterou bychom mohli použít pro vygenerování obrázků z `pptx`, nicméně daná knihovna pracuje jenom na platformě Windows a jenom pokud má uživatel nainstalovaný Microsoft office, což by bránilo přenositelnosti aplikace "ViSliCo"mezi platformami.

Po zvážení možných řešení jsme došli k závěru, že v našem případě bude nejlepším řešením podporovat pouze načítání `pdf` souborů s tím, že uživatel si v případě zájmu může přeformátovat `pptx` prezentace sám. Dalším důvodem, proč jsme se rozhodli nezabývat se problémy s `pptx` je, že většina vyučujících zveřejňuje prezentace ve formátu `pdf`.

Pro samotné převedení `pdf` prezentace do obrázků jsme použili existující knihovnu `PDF2image`.

6.1.5 Video do framů

Pro převedení videa do framů jsme použili knihovnu `OpenCV`. Framy z videa se generují podle nastavené časové frekvence (každých n vteřin). V ideálním případě bychom generovali framy co nejčastěji pro co nejpřesnější určování spojení, nicméně počet použitých framů má přímý vliv na délku spojování

videa s prezentací, a proto je v aktuální implementaci nastavena pevná frekvence, a to 30 vteřin.

6.1.6 Spojování obrázků

Jak jsme se rozhodli již v analýze, spojování prezentace s videozáznam budeme provádět pomocí spojování framů videa se slajdy.

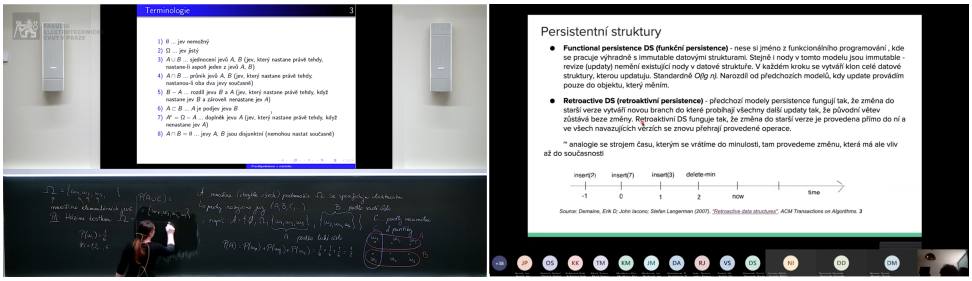
Funkcionalita spojení se dá rozdělit na 2 části, a to:

1. detekování a porovnání textu
2. detekce shod a kontrola jejich barevné podobnosti

S detekováním textu nám pomáhá modul `tesseract`. Modul `tesseract` na vstupu přijme jazyk, který pak detekuje ve vstupních obrázcích a na výstupu vrací detekovaný text z obrázku. Po detekování textu ze slajdů a framů se texty převedou do malých písmen a porovnáme je pomocí knihovny `difflib`, která nám vrátí podobnost textů v procentech. Pro označení, že slajd je zobrazen na framu, máme definovanou hodnotu podobnosti textu, kterou musí překonat. Pokud žádný slajd nedosáhne minimální hodnoty podobnosti, daný frame se zahodí a program pokračuje na další. Ze slajdů, které překonají hranici podobnosti, pak vybereme ten nejpodobnější a ten postoupí na další krok kontroly.

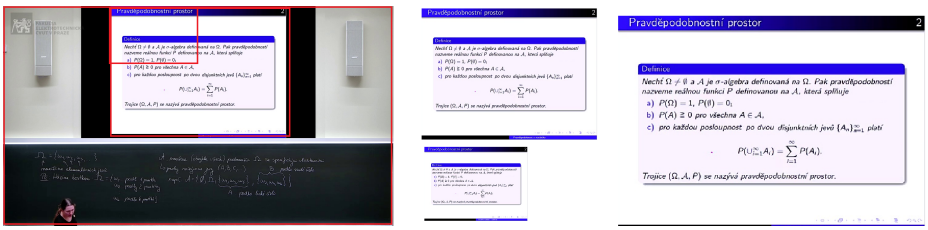
V druhém kroku kontroly, který je výpočetně náročnější, použijeme kvůli rychlosti již zmenšené obrázky. Hlavním cílem druhého kroku je odstranit falešně pozitivní spojení. Z pozorování víme, že některé slajdy, například slajdy pouze s obrázkem a záhlavím, mají málo textu a proto kontrolujeme i barevnou shodu.

Při ověřování barevné podobnosti je potřeba nejprve nalézt slajd ve framu, protože některé přednášky v sobě obsahují pouze malinký náhled na prezentaci a některé se skládají pouze ze záznamu prezentace se zvukovou stopou výkladu, jak můžeme vidět na obrázku 6.1. V některých případech se dokonce prezentace může po záznamu pohybovat.



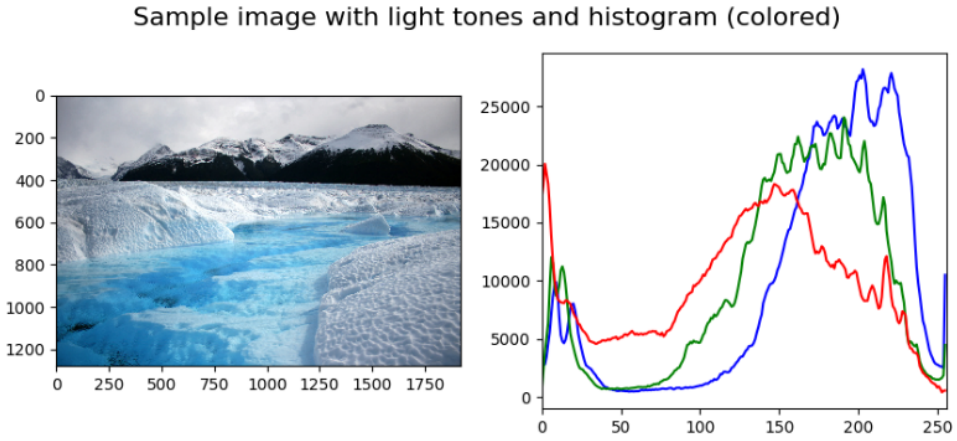
Obrázek 6.1: Příklady typu zobrazení slajdů na videozáznamech

Kvůli detekci polohy slajdu ve framu si vygenerujeme různé velikosti tohoto slajdu, jak můžeme vidět na obrázku 6.2, a pomocí metody *matchTemplate()* z knihovny `OpenCV` detekujeme nejpravděpodobnější polohu slajdu ve framu.



Obrázek 6.2: Znázornění hledání slajdu ve framu

Jakmile máme předpokládanou polohu ve framu, ověříme barevnou podobnost pomocí vytvoření a porovnání histogramu barev. Pomocí funkce *calcHist()* vytvoříme histogram pro všechny 3 barevné kanály (modrý, červený a zelený), jak je zobrazeno na obrázku 6.3 a pak spočítáme jejich podobnost funkcí *compareHist()*. Pro označení, že jsou obrázky barevně podobné, musejí opět překonat námi definovanou podobnostní hodnotu. Při nalezení první velikostní varianty slajdu, která splní podmínku, můžeme vrátit zkoumaný porovnávaný slajd jako spojení. V případě, že ani jedna z variant není dostatečně barevnostně podobná s framem, vracíme frame bez spojení.



zdroj: [D19]

Obrázek 6.3: vytvoření barevného histogramu pro každý kanál obrázku

Nakonec před zápisem spojení framu a slajdů do souboru, si data setřídíme tak, aby jsme k slajdu měli jenom začáteční čas pro každý blok. Například pokud máme se slajdem spojeny časy 0:30, 1:00, 1:30 a 5:00, tak se zapíšou jenom začáteční časy, a to 0:30 a 5:00 s tím, že ostatní už jsou jenom pokračování stejného slajdu.

Délka spojování záleží na více faktorech, například na počtu slajdů v prezentaci nebo na výkonosti počítače. Z pozorovaných běhů programu jsme pro spojování 1,5 hodinového videa naměřili časy od 3 do 20 minut. Jelikož se jedná o zdlouhavý proces, a neblokovali jsme uživatele, tak spojování spouštíme v novém procesu.

6.1.7 Vygenerování souboru

Jednou z dalších funkcionalit programu je vygenerování souboru se slajdy a textem z videozáznamu. Při implementování této funkcionality jsme narazili na dilema, v jakém formátu dokument generovat, jestli pdf nebo docx.

Python sice podporuje vytvoření dokumentu jak v docx tak i v pdf ,ale ze zkoušených knihoven, které podporují zápis do pdf ,se mi s žádnou nepodařilo dosáhnout formátování takového jaké bych chtěla. Navíc jsme si řekli že generování souborů docx bude i pro studenty praktičtější, protože si pak sami budou moci výstup upravit a případně v něm dopsat nebo odstranit co budou chtít.

Vzhledem k důležitosti formátování jsme se rozhodli udělat generování dokumentu ve formátu docx ,a to pomocí knihovny `docx`.

6.1.8 Zobrazování titulků

Na přehrávaní titulků jsme nenašli žádnou již implementovanou funkci v knihovně `PyQT5`, kterou používáme pro GUI, a tak jsme si zobrazování titulků vytvořili sami.

S pomocí knihovny `srt`, která parsuje a načítá titulky jsem vytvořila funkci, která pro zadaný čas vrací text řečený v daný čas. Pomocí této funkce pak při změně stavu videa se uživateli zobrazují titulky, pokud je má zapnuty.

6.1.9 Cachování dat

Důležitou částí našeho programu je ukládání si již dříve zpracovaných dat do lokální paměti uživatele. Díky ukládání dat do cache, nemusí program při každém používání vykonávat výpočetně náročné operace jako je například spojování slajdů s videem, a také uživateli umožňuje využívat funkcionalitu programu i bez internetového připojení.

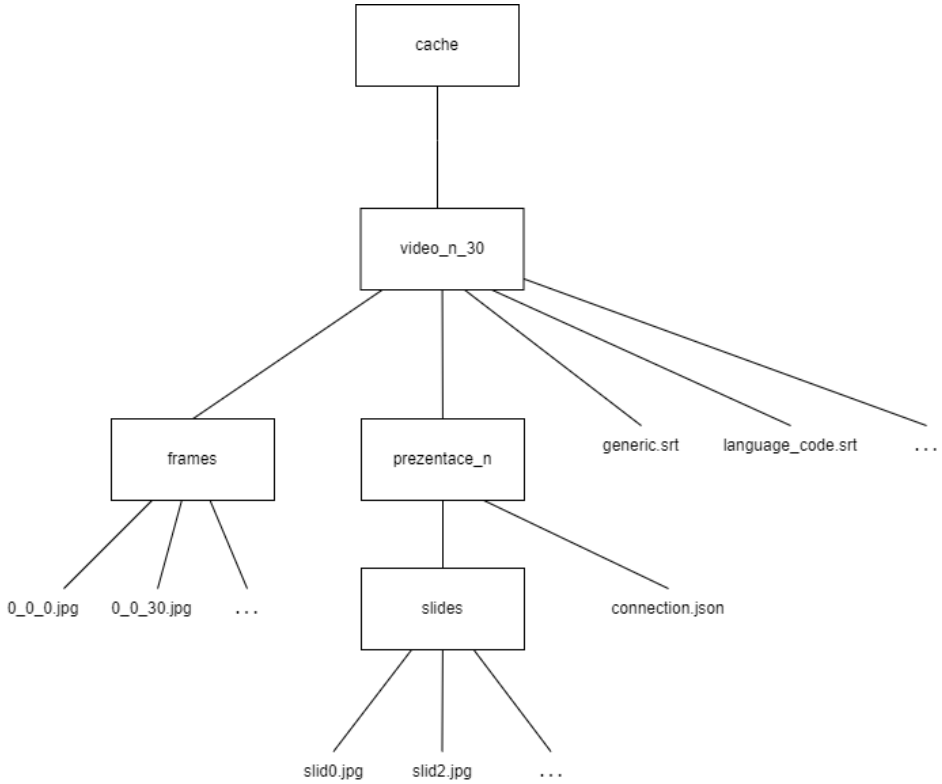
Cachování budeme dělat tak, že si programem ve složce aplikace vygenerujeme složku `cache`, do které budeme ukládat veškerá již zpracovaná data.

Prvním krokem uživatele je načtení videa, rozhodli jsme se proto všechny data k tomuto videu ukládat do složky načteného videa (nadále jen video-složky). Video-složka bude pojmenovaná podle názvu videa, formátu načteného videa a frekvence, se kterou generujeme spojení videa s prezentací. V současné podobě programu je frekvence spojování nastavena pevně na 30 vteřin, a proto je důležité, aby uživatel měl zpracovaná videa pojmenovaná unikátně. Toto omezení by neměl být problém, protože přednášky a cvičení jsou obvykle pojmenovány i číslovány různě.

Přímo do video-složky se pak ukládají vygenerované framy z videa, generované titulky, a pokud uživatel využije funkce překladu, tak pak i nově vzniklé přeložené titulky.

K videu uživatel může načíst také více prezentací, proto se jednotlivé prezentace uloží do nové složky se jménem pdf souboru, ve kterém byly načteny. Ve složce prezentace se vytvoří složka se slajdy prezentace a také

soubor definující spojení prezentace s videem. Celá struktura ukládaní dat do cache je znázorněna na obrázku 6.4.



Obrázek 6.4: diagram struktury cache

Pro managování cache máme v programu třídu `CacheSupervisor`, která vytváří nové video-složky, ověřuje obsah již vytvořených a kontroluje ukládaní dat do cache.

6.2 Uživatelské rozhraní

V této podkapitole si povíme o designu aplikace a o implementaci uživatelského rozhraní.

6.2.1 Použité technologie

Grafický design programu jsme vytvořili pomoci aplikace `Qt Designer`, která obsahuje grafické rozhraní pro vytváření uživatelského rozhraní v `PyQt5`. Z této aplikace jsme pak vygenerovali soubor v Python a vložili ho do svého programu. Tento soubor obsahuje veškeré informace ohledně použitých grafických elementů a jejich vzhledu. V dalším ručně psaném souboru jsme přidali složitější okna, jako například videoplayer a implementovali veškerou funkcionalitu. Dále pak v našem repositáři můžete najít soubor ve formátu

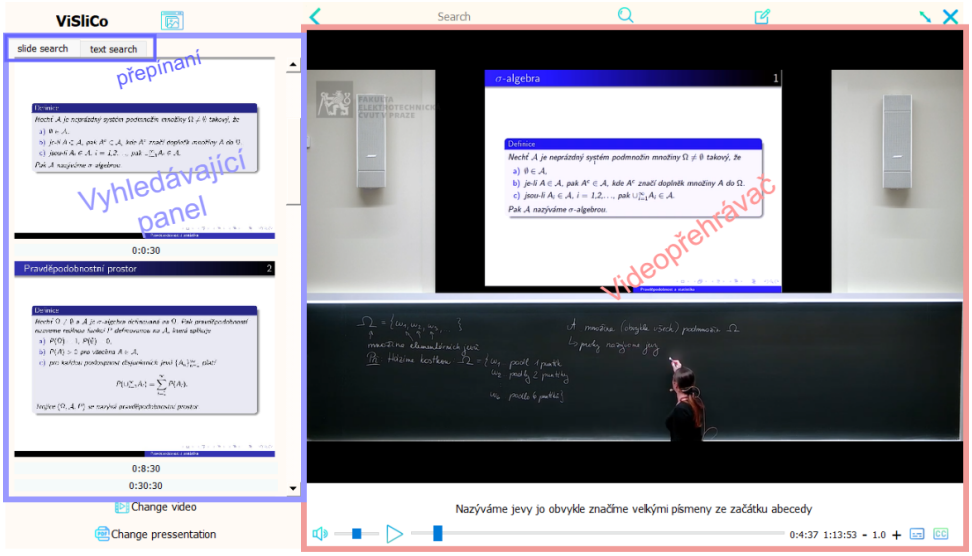
textsf.ui, který popisuje vytvořené uživatelské rozhraní ve formátu XML, a který můžete otevřít v aplikaci `Qt Designer`.

Výhodou implementování pomoci aplikace `Qt Designer` je rozdělení kódu do dvou částí, což je pak čitelnější, a také to, že hned vidíme vzhled aplikace a můžeme ho lehce opravovat a měnit dle potřeb. Nicméně i tato aplikace potřebovala čas, abych se s ní naučila pracovat.

6.2.2 Implementace GUI

Při implementaci GUI jsme se snažili napodobit již existující aplikace, obsahující videopřehrávač, tak aby se uživatelé mohli lehce orientovat již ve známém prostředí, například jako je YouTube.

Jelikož jsme pro ostatní funkcionalitu nenašli podobné aplikace, udělali jsme si pár návrhů a prokonzultovali jsme je se spolužáky. Nakonec jsme se po diskusi rozhodli pro návrh, kde vyhledávací panel bude přímo vedle video přehrávače a bude rozdělen na jednotlivé druhy vyhledávaní, jak je zobrazeno na obrázku 6.5.

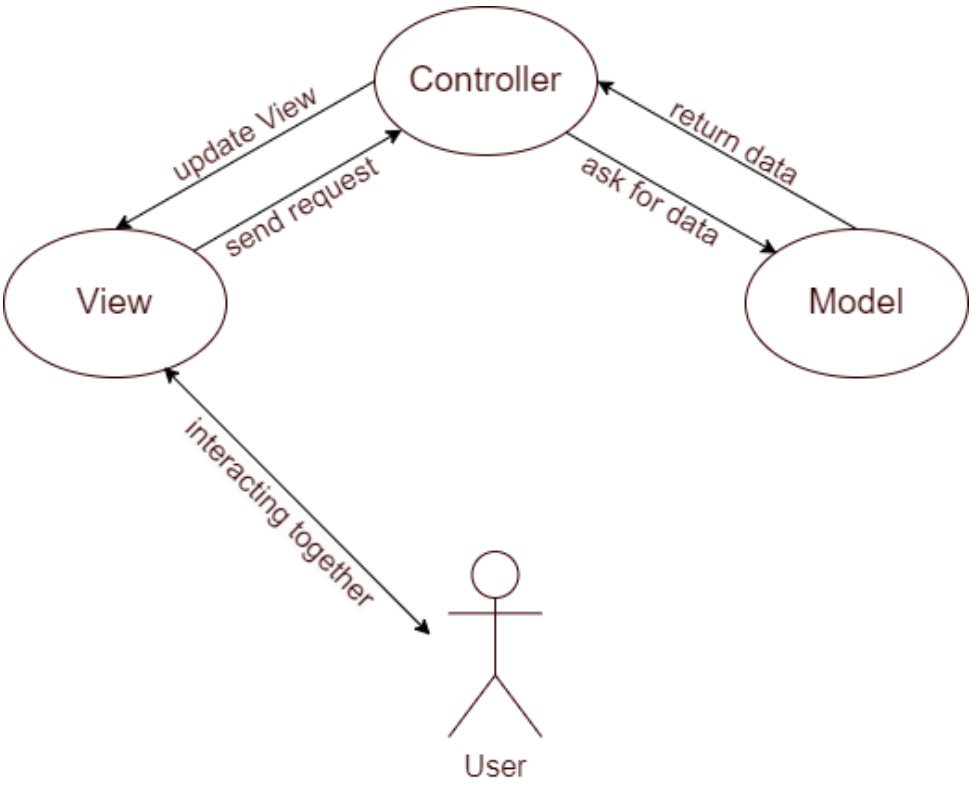


Obrázek 6.5: okno s funkcionalitou aplikace

Díky tomuto návrhu bude mít uživatel přehledné zobrazení výsledků vedle videa a umožní nám to v budoucnu lehké rozšíření o další způsoby vyhledávaní.

6.3 Architektura aplikace

Jako architekturu pro aplikaci jsme zvolili model MVC (Model View Controller). Tento pattern rozděluje veškerou funkcionalitu na 3 bloky Model, View a Controler, jak je zobrazeno na obrázku 6.6



Obrázek 6.6: diagram architektury aplikace

V naší implementaci je View blok zodpovědný za komunikaci s uživatelem a jedná se o grafické rozhraní, přes které uživatel zadává dotazy a vkládá data a které pak uživateli zobrazuje video a z něj vygenerovaná data.

Grafické rozhraní pak předá požadavky od uživatele Controlleru, který rozhodne jaká funkcionalita je potřeba a volá ji nad modelem. Typicky jde o to, že je potřeba vyhodnotit, jestli máme již data k dispozici a nebo je bude teprve potřeba vygenerovat. Pokud jsou data již k dispozici, Controller je nechá načíst z cache a vrátí View bloku. Pokud data nejsou k dispozici, Controller zavolá patřičné funkce nad modelem.

Model je u nás reprezentován třídou `Support`, která podle návrhového vzoru `Fasada`, zastřešuje funkcionalitu rozdělenou do menších tříd a funkcích nad jednotlivými knihovnami.

Kapitola 7

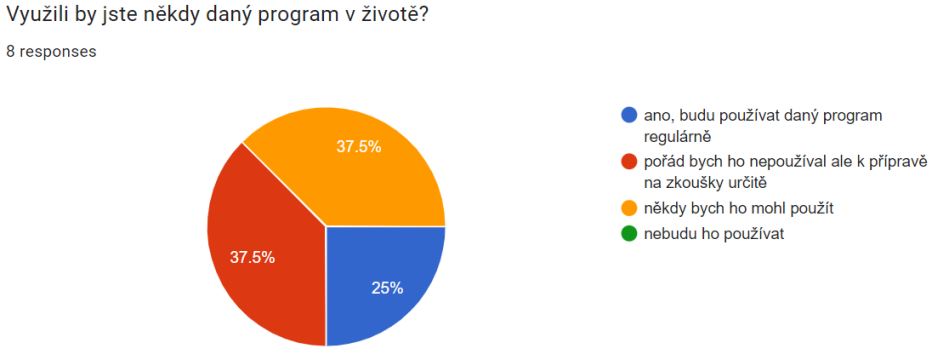
Testování a vyhodnocení programu

7.0.1 Výsledky uživatelského testování

Pro otestování funkčnosti generovaných výsledků programu "ViSliCo"jsme vytvořili dotazník a požádali několik studentů, aby nám ho vyplnili. Studenti byli vybráni jak z technických, tak i humanitních oborů.

Celkově si program stáhlo a vyzkoušelo 8 hodnotitelů. Všichni používali operační systém Windows a po instalaci potřebného balíčku program fungoval bezproblémově. Bohužel, kvůli tomu, že všichni naši hodnotitelé použili operační systém Windows, nemáme zkontrolovanou funkčnost programu na jiných platformách.

Jednou z hlavních otázek, co se nově vzniklého programu týče, bylo to, jestli by uživatelé používali náš program "ViSliCo"ve svém životě, a jak často. Na tuto otázku 25% respondentů odpovědělo, že ho budou používat pravidelně, 37,5% odpovědělo, že ho budou používat při přípravě na zkoušky a 37,5% napsalo, že by ho mohli někdy použít. Diagram s odpověďmi můžeme taky vidět na obrázku 7.1

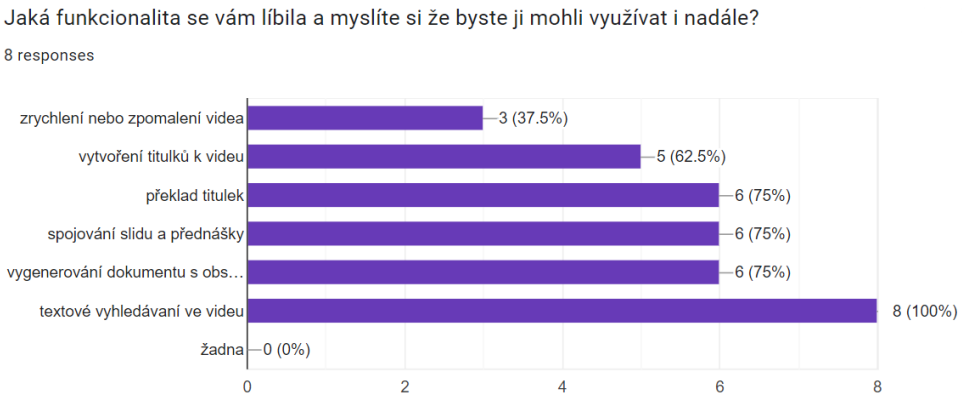


Obrázek 7.1: diagram odpovědí o používaní programu v budoucnu

Nejoblíbenější funkcí, kterou by uživatelé využívali i nadále, jak můžeme vidět na obrázku 7.2, je textové vyhledávání ve videu. Na druhou stranu, nejméně populární funkcí se stalo zrychlení a zpomalení videa, a to z velké částí kvůli tomu, že při úpravě rychlosti videa dochází k deformaci zvuku. Tento problém je aktuálně způsoben videopřehrávačem a je možné ho vyřešit

přidáním speciálních knihoven k tomu určených. Tuto funkčnost plánujeme upravit v budoucí verzi.

Názory ohledně ostatní funkcionality se rozdělily, což bude pravděpodobně kvůli využití různých materiálů studenty při učení. Například jeden z dotazovaných studentů napsal, že by v životě nepoužil generování textového dokumentu, protože pokud má k dispozici video záznam, radši si přehraje video. Další student, na druhou stranu, nechal komentář, že generování textového dokumentu je podle něho ta nejlepší funkcionalita, protože je zvyklý pracovat s psaným textem.



Obrázek 7.2: diagram odpovědí o funkcionalitě

Jedna z otázek, která nás zajímala, byla ohledně délky spojování videa s prezentací, a na ni, jak jsme i očekávali, jsme dostali různé odpovědi. Nicméně jsme rádi za výsledky, že u 7 z 8 uživatelů spojování 1,5 hodinového videa trvalo do 15 minut a téměř 40% dotazovaných ho mělo spojené do 5 minut.

Další otázkou, kterou jsme zkoumali bylo, jestli se nám podařilo udělat nativní GUI a zde 50% dotazovaných přiznalo, že některou funkcionalitu muselo hledat, a 12% uvedlo, že by bez zhlédnutí tutoriálu některé funkce ani nenašli. Pravděpodobně se většina dotazovaných na video-tutoriál podívala až v momentě, pokud nemohla něco najít, ale i tak bychom tento výsledek měli brát jako znamení, že bychom měli uživatelské rozhraní udělat více intuitivní, případně vymyslet jiný způsob jak uživatele lehce seznámit s ovládáním.

Dva z testovacích uživatelů se setkali s problémem při generování titulků. Místo hotových titulků, program zobrazil chybovou hlášku, že titulky jsou nedostupné kvůli selhání jejich generování. Tato chyba je způsobena problémem při komunikaci se serverem nebo chybou přímo na straně serveru, který používá `Autosubu`. Možným řešením je smazat soubor *generic.srt* z video-složky v cache a opakovaně načíst video pro jeho zpracování.

Dále jsme zjistili, že někteří uživatelé měli problém se zobrazováním ikonek pro spuštění videa, před tím než na ně najeli myší. Po revizi kódu jsme ale nenašli nic, co by mohlo způsobovat danou chybu.

Jeden z testujících taky napsal, že program mu občas padal, pokud se snažil udělat více akcí na jednou. Po komunikaci s tímto uživatelem jsme zjistili, že chování, při kterém chyba nastala, neodpovídá klasickému scénáři

použití naší aplikace.

7.0.2 Zpětná vazba od uživatelů

Přestože aplikace "ViSliCo", už teď obsahuje hodně funkcí, stále existují kroky pro její vylepšení. V dotazníku jsme dostali i řadu komentářů k programu a hodně nápadu na vylepšení, případně upravení stávající funkcionality.

Negativní komentáře

Někteří uživatelé (jeden) v komentářích zmínili již výše uvedené nedokonalosti programu a označili je jako něco, co by si přáli, aby bylo opraveno před tím, než by začali program aktivně používat. Příkladem těchto požadavků je například neintuitivní rozhraní nebo úprava zvuku po zrychlení a zpomalení videa.

Pozitivní komentáře

Většina dotazovaných zmiňovala užitečnost programu "ViSliCo", a to nejenom pro studijní účely. Hodně z nich chválilo širokou funkcionalitu a litovali, že naše aplikace neexistovala za covidové doby, jelikož by jim pomohla s učením a ušetřila čas. Minimálně dva z dotazovaných uvedli, že si program už v této podobě nechají a budou ho aktivně používat při studiu.

Návrhy na vylepšení

V anketě jsme dostali řadu návrhů na vylepšení, tady jsou některé z nich:

- Vygenerovat dokument k videu i pokud není přidaná prezentace. [Již implementováno.]
- Přidat ruční nastavení velikosti okna a tlačítka pro minimalizace okna.
- Ovládání programu pomocí klávesnice: skok ve videu o 10s dopředu či zpět, hlasitost, zrychlení a zpomalení, spouštění textového hledání pomocí tlačítka 'Enter'.
- Přidat uživatelům možnost měnit velikosti písma.
- Jasnější označení funkcionality ikonek. Například vyskakující textový popisek.
- Udělat klikatelný i přímo sám slajd/text. Ne jenom čas pod ním.
- Lépe uživatele informovat o stavu zpracování materiálů.
- Zobrazit celý text vedle běžícího videa.

Tato rozšiřující funkcionalita je uskutečnitelná v rámci našeho programu a může být přidána v rámci budoucích prací.

Dále pak v anketě jeden uživatel zmínil i větší návrh na rozšíření programu, a to vytáhnout slajdy z videa, aniž by uživatel poskytoval prezentaci. Vzhledem k existujícím technologiím, kdy o některých z nich jsme se již zmiňovali v 5. kapitole, tento návrh je uskutečnitelný a mohl by skvěle doplnit funkcionalitu našeho programu "ViSliCo".

7.0.3 Osobní nápady na vylepšení

Já osobně mám také několik nápadů na vylepšení aplikace, které ještě nebyly zmíněny.

S tím že hodně uživatelů odpovědělo, že by využívalo i nadále textové vyhledávání, nám přijde užitečné ho rozšířit. Například hledáním nejenom zadaných slov, ale i jejich synonym nebo zkomolených variant. Další funkcionalitou by mohlo být samodetekování použitého jazyka ve videu i prezentaci.

Co se distribuce našeho programu týče, aktuální verze není úplně dokonalá a uživatel si sám musí stáhnout chybějící balíčky, jak je popsáno v README. V budoucnu bychom chtěli tento problém vyřešit.

Kapitola 8

Závěr

Cílem této bakalářské práce bylo zanalyzovat potřeby studentů vysokých škol a identifikovat funkcionalitu, která by studentům zjednodušila přípravu na výuku, testy a učení se na zkoušky z již existujících výukových materiálů. Tento záměr jsme splnili a popsali v kapitolách 2 a 3.

Po prvotním nápadu na vytvoření aplikace, jsme si pomocí dotazníku vyplněného studenty ověřili reálný zájem o naši aplikaci a zjistili si klíčovou funkcionalitu, kterou by studenti u nově vzniklé aplikace ocenili, jak jsme popsali v kapitole 3.

Dále jsme pak udělali průzkum trhu a ověřili, že podobné nástroje ještě neexistují a vytvoření takové aplikace má tedy smysl. Tento cíl jsme splnili a popsali v kapitole 4.

Jelikož podobné aplikace ještě neexistují, ověřili jsme si, že implementace námi popsané aplikace je proveditelná. Dále jsme pak udělali základní návrh aplikace a vybrali vhodné technologie k jejímu implementování, jak jsme popsali v kapitole 5.

Po navrhnutí aplikace, jsme ji implementovali a vytvořili nový program jménem "ViSliCo", který splňuje veškeré požadavky, které jsme si určili v původním návrhu. Popis implementace je popsán v kapitole 6.

Pro otestování naší aplikace a ověření, že splňuje ve vzniklé podobě potřeby studentů, jsme aplikaci rozdistribuovali skupině studentů, která ji otestovala a poskytla nám zpětnou vazbu. Výsledky testování jsou v kapitole 7. Podle studentů aplikace splňuje požadovanou funkcionalitu a je vhodná k použití. V rámci naší práce jsme tedy naplnili veškeré cíle, které jsme si stanovili v zadání.

Do budoucna jsme si s aplikací naplánovali nové cíle, kterých bychom chtěli dosáhnout. Po diskuzi se studenty jsme získali návrhy na rozšíření funkcionality uživatelského rozhraní, a to konkrétně o ovládání pomocí klávesnice, například zesilování videa pomocí šipek, spuštění textového vyhledávání tlačítkem 'Enter', nebo přidání uživatelům možnosti měnit velikosti písma v aplikaci. Dále jsme dostali návrh na přidání funkce, která vyextrahuje použité slajdy přímo z videa, aniž by uživatel poskytoval prezentaci. Více o plánovaných rozšířeních si můžete přečíst v části 7.0.2 a 7.0.3.

Seznam použitých zkratek

Použité zkratky a jejich vyznám:

- API - Application Programming Interface
- GUI - Graphic User Interface
- MVC - Model View Controller
- OCR - Optical Character Recognition
- ViSliCo - Video Sllides Connector

Literatura

[Amn20] Amnindersingh1414, *Python gui – pyqt vs tkinter*.

[Cha22] Punition Chaetognathan, *Which python gui library should you use?*

[D19] Raghunath D, *Image histograms in opencv*.

[Dal54] E. Dale, *Audiovisual methods in teaching*, 2en ed. ed., New York, Dryden Press, Dostupné z: https://archive.org/details/audiovisualmetho00dale/page/42/mode/2up?view=theater, 1954.

[DVMSPBME16] Ph.D. Dr. V.K.Maheshwari M.A Socio Phil B.Sc. M. Ed, *Edgar dale's cone of experience*, Dostupné z: http://www.vkmaheshwari.com/WP/?p=2332, 2016.

[Fle95] N.D. Fleming, *I'm different; not dumb. modes of presentation (vark) in the tertiary classroom, in zelmer,a.*, (ed.) research and development in higher education ed., Proceedings of the 1995 Annual Conference of the Higher Education and Research Development Society of Australasia (HERDSA)[online], Dostupné z: http://www.vark-learn.com/wp-content/uploads/2014/08/different_not_dumb.pdf, 1995.

[Jai] JaidedTeam, *Easyocr*.

[Ken21] Mikhail Kennerley, *A comparison of sift, surf and orb on opencv*.

[Lea] VARK Learn, *Strategies matched to vark preferences*.

[NF19] C. Bonwell N.D. Fleming, *How do i learn best?: A students guide to improved learning: Vark - visual, aural, read/write, kinaesthetic*, [online] (ed.) ed., Research and Development in Higher Education, ISBN

Number: 978-0-473-07810-2, Dostupné z: https://vark-learn.com/wp-content/uploads/2019/07/How-Do-I-Learn-Best-Sample.pdf, 2019.

[Nic] Nicki, *What's the best way to learn english?*

[Ros21] Adrian Rosebrock, *Multi-template matching with opencv.*

[S.20] Dutta S., *Teaching of geography through dale's cone of experience.*

[Tak19] Mayuri Takle, *Advantages and disadvantages of tkinter and wxpython.*

[Tal20] Abhiroop Talasila, *Generate subtitles for any video file using mozilla deepspeech.*

[Tha15] W. Thalheimer, *Debunk this: People remember 10 percent of what they read*, Journal of Elliptic Analysis (2015).

Příloha A

Přílohy

K teto prací taky přikládám následující elektronické přílohy:

- Uživatelsky manuál pro program ViSliCo: Uzivatelsky_manual.pdf
- Krátký video manuál pro program ViSliCo: ViSliCo.mp4
- Soubor s časovou náročnosti pro jednotlivé částí práce: Harmonogram.pdf

Odkaz na git repositář :
https://gitlab.fel.cvut.cz/pasicmar/vislico_public
Ve složce PYTHON je uložen kód k mé bakalářské práci a složka EXE obsahuje spustitelnou aplikaci/distribuci mé bakalářské práce.