

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

EDITOR AUTOMATICKÉHO PŘEPISU ŘEČI V REÁLNÉM ČASE

EDITOR FOR REAL TIME SPEECH TRANSCRIPTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **BORIS HATALA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. IGOR SZŐKE, Ph.D.**
SUPERVISOR

BRNO 2026

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

169903

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Hatala Boris**
Program:	Informační technologie
Název:	**Editor automatického přepisu řeči v reálném čase**
Kategorie:	Webové aplikace
Akademický rok:	2025/26

Zadání:

1. Seznamte se s frameworkem React (nebo podobným) a stávajícími platformami pro přepis a anotaci zvuku. Zaměřte se na UX/UI. Prostudujte také existující API pro backendy automatického přepisu v reálném čase.
2. Spolu s vedoucím identifikujte vhodné komponenty a případy použití.
3. Navrhněte a implementujte vybrané komponenty. Během implementace pravidelně vyhodnocujte jejich UX a integrujte své řešení s existujícím API pro přepis.
4. Otestujte navržený frontend na vhodné skupině uživatelů. Ověřte uživatelskou přívětivost a snadnost použití rozhraní.
5. Diskutujte o dosažených cílech a navrhněte směry dalšího vývoje.
6. Vytvořte plakát formátu A2 a 30sekundové video prezentující výsledky vaší práce.

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
- Next, according to the supervisor suggestions

Při obhajobě semestrální části projektu je požadováno:
Body 1-3 za 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:	1.11.2025
Termín pro odevzdání:	13.5.2026
Datum schválení:	7.11.2025

Abstrakt

Táto práca sa zaoberá návrhom a implementáciou editora automatického prepisu reči v reálnom čase.

Riešenie je klientská webová aplikácia, ktorá využíva externú službu pre automatický prepis reči. Aplikácia umožňuje nahrávanie z mikrofónu alebo nahratie súboru a automatický prepis reči v reálnom čase s možnosťou súčasnej editácie. Okrem toho podporuje správu rečníkov a záložiek a obojsmerne synchronizuje prepis so zvukovou nahrávkou na úrovni slov.

Aplikácia bola overená testovaním použiteľnosti, pričom výsledkom je webová aplikácia, ktorá nevyžaduje serverovú infraštruktúru a umožňuje editáciu prepisu priamo počas jeho vkladania.

Abstract

This thesis presents the design and implementation of a real-time automatic speech transcription editor.

The solution is a client-side web application that uses an external service for automatic speech recognition. It supports microphone recording or audio file upload, with real-time automatic transcription and simultaneous editing. The application also provides speaker and bookmark management, and bidirectionally synchronizes the transcript with the audio recording at the word level.

The application was validated through usability testing, resulting in a web application that requires no server infrastructure and enables editing the transcript directly as it is being inserted.

Kľúčové slová

prepis reči, automatický prepis reči, rozpoznávanie reči, webová aplikácia, editor, UI, UX, testovanie použiteľnosti

Keywords

speech transcription, automatic speech recognition, web application, editor, UI, UX, usability testing

Citácia

HATALA, Boris. *Editor automatického přepisu řeči v reálném čase*. Brno, 2026. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Igor Szőke, Ph.D.

Editor automatického přepisu řeči v reálném čase

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Igora Szőke, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal. Pre jazykovú revíziu technickej správy som použil nástroj Claude. Pri tvorbe odovzdaného programového kódu som využíval nástroj Claude ako asistenta pri návrhu a implementácii.

.....
Boris Hatala
12. mája 2026

Poďakovanie

Chcem sa poďakovať vedúcemu práce pánovi Ing. Igorovi Szőke, Ph.D. za odborné vedenie a cenné rady, mojej priateľke Else za mentálnu podporu a všetkým testerom aplikácie za pomoc pri vývoji.

Obsah

1 Úvod **5**

2 Editory prepisu reči **6**
2.1 Beey 6
2.2 Speechnotes 8
2.3 SpokenData 9
2.4 Rev 10
2.5 Záver 13

3 Návrh riešenia **14**
3.1 Požiadavky aplikácie 14
3.2 Architektúra systému 15
3.3 Dátový model 16
3.4 Záver 17

4 Užívateľské rozhranie **18**
4.1 Rozloženie hlavnej stránky 18
4.2 Vývoj užívateľského rozhrania 19
4.3 Funkčné komponenty 23
4.4 Záver 27

5 API služby pre prepis reči **28**
5.1 Gladia 28
5.2 Azure AI 29
5.3 Web Speech API 30
5.4 Porovnanie 31
5.5 Záver 32

6 Frontendové knižnice a frameworky **33**
6.1 React 33
6.2 Angular 34
6.3 Vue.js 35
6.4 Knižnice UI komponentov 36
6.5 Knižnice pre rich-text editory 37
6.6 Knižnice pre vizualizáciu zvukových vĺn 38
6.7 Záver 38

7 Implementácia **39**

7.1 Voľba implementačných technológií 39
7.2 Kontexty aplikácie 40
7.3 Rich-text editor 40
7.4 Ukladanie dát 41
7.5 Prehrávanie audio záznamu 42
7.6 Synchronizácia audia a textu 42
7.7 Dávkový prepis 43
7.8 Export prepisu 43
7.9 Záver 43

8 Testovanie použiteľnosti **44**
8.1 Testovanie použiteľnosti desktopovej verzie 44
8.2 Úpravy po testovaní použiteľnosti desktopovej verzie 49
8.3 Testovanie použiteľnosti mobilnej verzie 53
8.4 Úpravy po testovaní použiteľnosti mobilnej verzie 55
8.5 Záver 55

9 Záver **57**

Literatúra **59**

Zoznam obrázkov

2.1 Screenshot webovej aplikácie Beey (editor) 7
2.2 Screenshot webovej aplikácie Beey na mobile (editor). 8
2.3 Screenshot rozhrania funkcie diktovania v aplikácii Speechnotes 9
2.4 Screenshot aplikácie SpokenData. 10
2.5 Screenshot webovej aplikácie Rev 11
2.6 Mobilná aplikácia Rev počas nahrávania z mikrofónu. 12
2.7 Zobrazenie prepisu v mobilnej aplikácii Rev po ukončení nahrávania 12
2.8 Alternatívna podoba UI v mobilnej aplikácii Rev podobná webovej verzii. . 12

3.1 Tok dát v aplikácii. Pri nahrávaní z mikrofónu (a) je možné nahrávanie opakovane spúšťať. Každý cyklus doplní prepis na koniec dokumentu. Real-time spracovanie súboru sleduje rovnaký tok bez možnosti opätovného spustenia nahrávania alebo pridania ďalšieho súboru do dokumentu. V dávkovom režime (b) sa čaká na kompletný prepis pred vložením do editora. 16

4.1 Prvá iterácia návrhu UI: bočný panel naľavo, panel nástrojov nad dokumentom, ovládací panel napravo a audio prehrávač v dolnej časti 20
4.2 Druhá iterácia návrhu UI: ovládací panel presunutý z pravej strany bližšie k ľavému okraju v súlade s *F-pattern* modelom skenovania 21
4.3 Tretia iterácia návrhu UI: tmavé pozadie s bielymi plochami dokumentu a panelov pre lepší kontrast, rozloženie sústredené na ľavú stranu obrazovky . 22
4.4 Mobilná verzia rozhrania editora. 23
4.5 Bočný panel otvorený ako prekrývajúca zásuvka. 23
4.6 Panel nástrojov na formátovanie pri otvorení klávesnice. 23
4.7 Farebne zvýraznený blok textu jedného rečníka s ikonou prehrávania a časovou značkou začiatku úseku 24
4.8 Vyskakovacie okno správy rečníkov zobrazené po kliknutí na meno rečníka; umožňuje premenovanie, zmenu farby a priradenie rečníka k danému úseku 24
4.9 Audio prehrávač s vizualizáciou zvukových vĺn; kliknutím na priebeh signálu možno skočiť na ľubovoľné miesto v nahrávke 25
4.10 Synchronizácia prepisu s nahrávkou: odznené slová sú zvýraznené čiernou farbou, zvyšok prepisu je zošednutý 25
4.11 Záložka zobrazená ako ikona priamo v tele prepisu bez narušenia čitateľnosti textu 26
4.12 Kontextové menu rozdelené do troch skupín: akcie editora prepisu, operácie so schránkou a formátovanie textu 26

8.1 Stav aplikácie počas testovania použiteľnosti. 46
8.2 Upravený modul záložiek so zvýraznením textu a odkazom na sekciu. 49

8.3 Pole pre nahratie súboru so zmeneným kurzorom a slovnou indikáciou pre kliknutie. 49
8.4 Indikácia funkcionality editácie v reálnom čase v počiatočnom stave editora pred spustením prepisu reči. Najskôr sa zobrazí text vyzývajúci používateľa na editáciu, po ktorom nasleduje animácia demonštrujúca priebežnú úpravu prepisu. 50
8.5 Upravený prehrávač s tlačidlami skoku a záložkami vyznačenými vo vlnovom zobrazení. 50
8.6 Slovná indikácia v tele editora, ktorá upozorní používateľa, že môže rozprávať pri prepise z mikrofónu. 51
8.7 Rozdelená akcia vloženia rečníka: nový rečník alebo výber z existujúcich. . 51
8.8 Moduly zobrazené ako plávajúce okná prekrývajúce dokument. 52
8.9 Stav aplikácie po zapracovaní úprav plynúcich z testovania použiteľnosti. . . 52
8.10 Tlačidlá pre výber režimu prepisu súboru pred úpravou 55
8.11 Tlačidlá pre výber režimu prepisu súboru po úprave 55
8.12 Modul záložiek po testovaní použiteľnosti 55

Kapitola 1

Úvod

Editor prepisu reči predstavuje softvérový nástroj, ktorý umožňuje vytvárať prepis hovoreného slova a následne tento prepis editovať. Takéto nástroje sú užitočné v rôznych odvetviach ako sú vzdelávanie, žurnalistika, médiá, tvorba titulkov, archivácia zvukových záznamov alebo zlepšenie prístupnosti obsahu pre osoby so sluchovým znevýhodnením.

Existujúce riešenia v tejto oblasti prevažne fungujú ako komplexné platformy s vlastnou serverovou infraštruktúrou, ktorá zabezpečuje nielen prepis, ale aj ukladanie a správu súborov. Takýto prístup prináša niekoľko nevýhod: používateľské dáta sú uložené na vzdialených serveroch a samotná aplikácia je závislá na službách bežiacich v pozadí. Tieto nástroje navyše väčšinou nepodporujú editáciu v reálnom čase. Používateľ musí čakať na dokončenie celého prepisu, až potom môže pristúpiť k jeho úpravám. Motiváciou tejto práce je navrhnúť editor, ktorý tieto obmedzenia prekonáva. Celá aplikácia beží v zariadení používateľa: nahrávky, dokumenty aj editačná logika sú spracúvané lokálne. Výnimkou je automatický prepis reči, ktorý zabezpečuje vzdialená služba tretej strany.

Cieľom tejto bakalárskej práce je návrh a implementácia webovej aplikácie pre editáciu prepisu reči v reálnom čase. Hlavný dôraz je kladený na jednoduché a intuitívne užívateľské rozhranie a na to, aby aplikácia nevyžadovala samostatnú infraštruktúru bežiacu v pozadí. Navrhnuté riešenie je následne overené testovaním s cieľovými používateľmi.

Práca je členená nasledovne. Najprv sú analyzované existujúce editory prepisu reči z hľadiska ich užívateľského rozhrania, pracovného toku a funkčných vlastností s cieľom identifikovať ich silné stránky aj nedostatky. Na základe zistení sú definované požiadavky na aplikáciu a navrhnutá jej architektúra, dátový model a užívateľské rozhranie. Následne sú preskúmané dostupné služby pre automatické rozpoznávanie reči a technológie vhodné pre vývoj reaktívnych webových aplikácií. Jadro práce tvorí popis implementácie kľúčových častí riešenia. Záver je venovaný používateľskému testovaniu a zhodnoteniu dosiahnutých výsledkov.

Kapitola 2

Editory prepisu reči

Editor prepisu reči je softvérový nástroj, ktorý umožňuje vytvárať textový záznam hovoreného slova a následne ho upravovať. Na rozdiel od bežných textových editorov je úzko prepojený so zvukovým alebo audiovizuálnym záznamom. Umožňuje používateľovi prehrávať nahrávku a súčasne upravovať zodpovedajúci text. Editory zvyknú podporovať aj vizuálnu synchronizáciu medzi slovami v prepise a príslušným úsekom záznamu.

Základné funkcie editora prepisu reči zahŕňajú načítanie alebo nahranie zvukového záznamu, editáciu prepisu, prehrávanie záznamu a export výsledného prepisu do štandardných formátov. Pokročilejšie nástroje pridávajú funkcie ako automatické rozpoznávanie rečníkov (diarizácia), tvorba titulkov alebo integráciu s umelou inteligenciou (AI).

Táto kapitola analyzuje v súčasnosti dostupné nástroje tohto typu so zameraním na ich užívateľské rozhranie (UI) a užívateľský zážitok (UX). Cieľom je identifikovať ich silné stránky aj obmedzenia, ktoré poskytnú základ pre návrh vlastného riešenia.

2.1 Beey

Beey[1] je webová aplikácia zameraná na automatický prepis reči z audio a video nahrávok. Služba podporuje viac ako 30 jazykov.

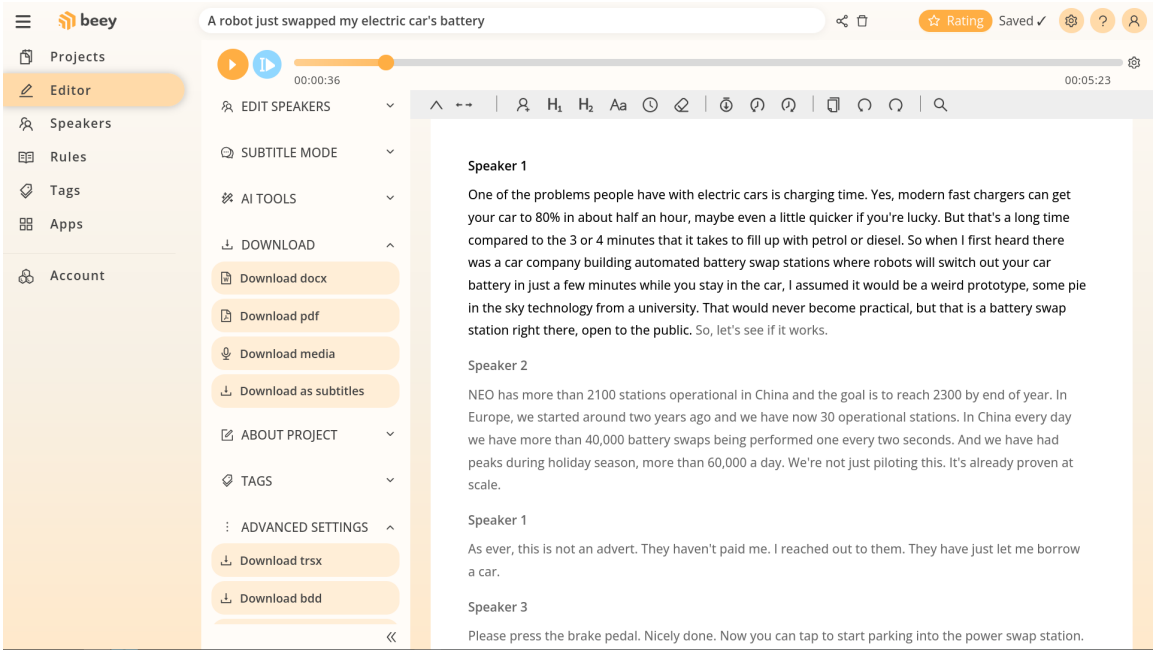
Pracovný postup

Beey podporuje aj živé nahrávanie z mikrofónu priamo v prehliadači – prepis sa zobrazuje v reálnom čase a text je možné editovať už počas nahrávania. Používateľ môže taktiež nahrať audio alebo video súbor (napr. vo formáte MP3, MP4, WAV) priamo cez webové rozhranie. Súbor je odoslaný na servery Beey, kde prebehne jeho automatické spracovanie. Používateľ musí počkať na dokončenie celého prepisu, až potom získa prístup k editoru (na obrázku 2.1). Platforma ponúka aj doplnkové funkcie AI ako automatický súhrn prepisu a preklad do iných jazykov. Hotový prepis je možné exportovať do formátov PDF alebo DOCX, prípadne použiť na vytvorenie titulkov v štandardných formátoch (SRT, VTT a podobne). Používateľ môže takisto stiahnuť aj zdrojový záznam.

UI a UX

Pracovný priestor editora Beey je rozdelený do dvoch hlavných oblastí: textový editor s prepisom a prehrávač nahrávky. Synchronizácia je jednosmerná – počas prehrávania je aktuálne

[1] Domovská stránka aplikácie Beey: https://www.beey.io/en/



Obr. 2.1: Screenshot webovej aplikácie Beey (editor)

hovorené slovo v texte zvýraznené, avšak presunutie kurzora v editore polohu prehrávača neovplyvní.

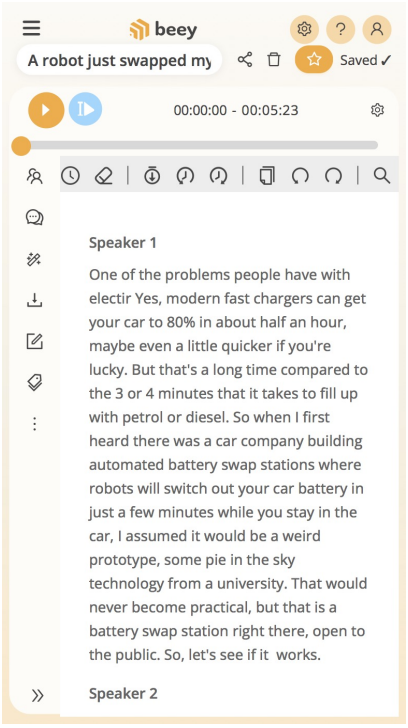
Pozitívnym prvkom je použitie klávesy `Tab` na spustenie a pozastavenie prehrávania, čo umožňuje plynulé striedanie počúvania a písania bez zbytočných pohybov myši. Editor takisto umožňuje manuálnu korekciu časových značiek jednotlivých slov, čo je užitočné najmä pri tvorbe titulkov, kde je presné časové zaradenie kľúčové.

Slabšou stránkou je časté využívanie vyskakovacích okien (*modal*) pre rôzne úkony vrátane úpravy rečníkov v texte. Tento prístup narúša plynulosť práce, keďže používateľ musí opakovane opúšťať kontext editácie a pracovať s modálnymi dialógmi.

Editor neponúka bežné nástroje formátovania textu ako tučné písmo (*bold*) alebo kurzívu (*italic*). K dispozícii sú iba štrukturálne prvky ako nadpisy (*Heading 1*, *Heading 2*).

Celkovo je Beey prepracovaný nástroj s pokročilými funkciami ako diarizácia rečníkov, korekcia časových značiek alebo export titulkov. Pri spracovaní nahratého súboru zostáva obmedzením nutnosť čakania na dokončenie prepisu pred začatím editácie.

Beey je prístupné aj na mobilných zariadeniach (na obrázku 2.2) – nejde o samostatnú aplikáciu, ale o webové rozhranie s upraveným rozložením pre menšie obrazovky. Funkčné vlastnosti zostávajú totožné s desktopovou verziou, rozhranie je však prispôsobené menšej pracovnej ploche – bočný panel sa mení na vysúvací panel a zvislý panel s nástrojmi je zúžený iba na ikony.



Obr. 2.2: Screenshot webovej aplikácie Beey na mobile (editor).

2.2 Speechnotes

Ďalším z existujúcich riešení je webová aplikácia Speechnotes[2] (na obrázku 2.3). Táto aplikácia ponúka diktovanie, prepis reči z audiovizuálnych záznamov alebo generáciu hlasu z textu. V kontexte tejto práce je relevantná predovšetkým jej funkcia diktovania. Na rozdiel od ostatných analyzovaných nástrojov Speechnotes nevyužíva vlastný backendový model. Prepis prebieha priamo v prehliadači prostredníctvom rozhrania Web Speech API[3], čo ju obmedzuje len na prehliadače s jeho podporou.

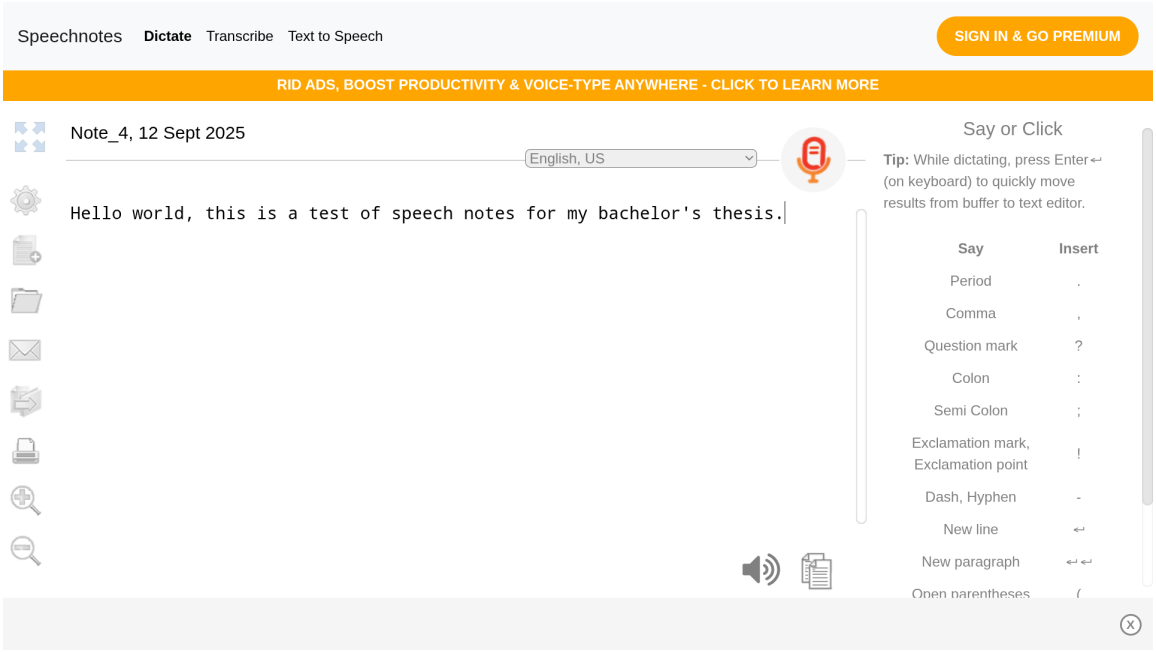
Pracovný postup

Používateľ spustí nahrávanie stlačením tlačidla mikrofónu a začne hovoriť. Počas rozprávania sa zobrazuje priebežný náhľad rozpoznaného textu, ktorý však ešte nie je vložený do editora. K vloženiu dôjde až v momente, keď používateľ na chvíľu prestane hovoriť. Následne je možné text upraviť alebo v diktovaní pokračovať. Výsledný prepis je možné skopírovať, stiahnuť alebo vytlačiť.

UI a UX

Rozhranie Speechnotes je minimalistické, tvorí ho prevažne textová plocha a tlačidlo mikrofónu. Táto jednoduchosť uľahčuje novým užívateľom sa zoznámiť s aplikačným prostredím, ale zároveň obmedzuje možnosti pokročilejšej editácie.

[2]Odkaz na aplikáciu Speechnotes: https://speechnotes.co
[3]Web Speech API je štandardizované rozhranie prehliadača pre rozpoznávanie a syntézu reči.



Obr. 2.3: Screenshot rozhrania funkcie diktovania v aplikácii Speechnotes

Keďže text je do editora vkladaný až po odmlčaní, nie je možná plynulá editácia v reálnom čase počas diktovania. Používateľ tak nemôže okamžite opravovať chyby v práve diktovanom texte.

Indikácia aktívneho nahrávania je málo výrazná, tlačidlo mikrofónu prakticky nezmení vzhľad a jediným signálom je malý animovaný krúžok v jeho blízkosti. Pre používateľa tak nemusí byť na prvý pohľad zrejmé, či nahrávanie práve prebieha.

Editor nepodporuje formátovanie textu, chýbajú bežné prvky ako tučné písmo, kurzíva či nadpisy. Ide o čistý textový editor bez podpory bohatého formátovania (*rich text*).

Aplikácii chýba prepojenie s nahraným zvukom. Po skončení diktovania nie je možné nahrávku prehrať ani sa k nej vrátiť. Speechnotes je preto vhodná predovšetkým pre jednoduché diktovanie poznámok, nie pre editáciu existujúcich záznamov.

Speechnotes je dostupná aj ako mobilná aplikácia s rovnakou funkcionalitou ako webová verzia. Napriek tomu, že webová stránka uvádza podporu pre iOS aj Android, aplikácia je v skutočnosti dostupná len pre Android.

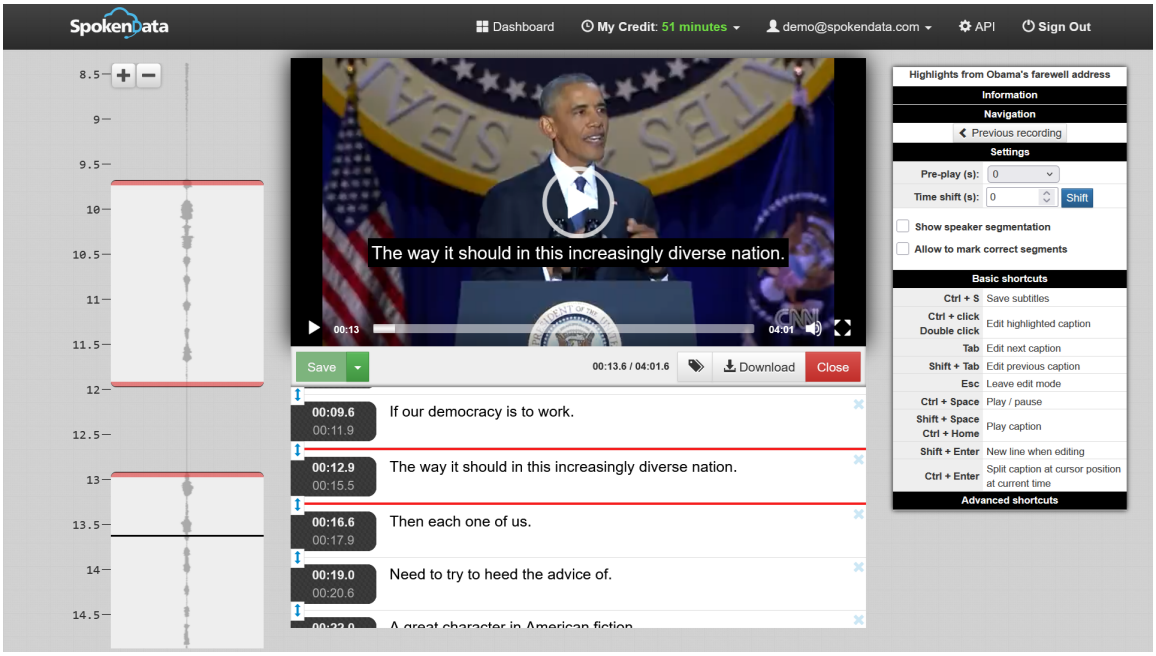
2.3 SpokenData

SpokenData[4] je cloudová služba automatického prepisu reči od spoločnosti ReplayWell[5], zameraná predovšetkým na tvorbu a editáciu titulkov. Podporuje 10 jazykov.

Pracovný postup

Používateľ nahrá audio alebo video súbor, prípadne poskytne odkaz na YouTube video. Prepis vzniká na strane servera a po jeho dokončení príde používateľovi emailová notifiká-

[4]Odkaz na domovskú stránku SpokenData: https://www.spokendata.com/
[5]Odkaz na domovskú stránku ReplayWell: https://www.replaywell.com/



Obr. 2.4: Screenshot aplikácie SpokenData.

cia. Výsledný prepis je rozdelený do segmentov s časovými značkami, ktoré možno editovať, dopĺňať alebo manuálne priradiť jednotlivým rečníkom – automatická diarizácia nie je podporovaná. Výstup je exportovateľný do formátov TXT, SRT, VTT alebo DOCX.

UI a UX

Pracovný priestor (na obrázku 2.4) je rozdelený do troch oblastí: na ľavej strane sa nachádza zvuková stopa (*waveform*), v strede hore prehrávač zdrojového média a pod ním segmenty prepisu v podobe obdĺžnikov s časovými značkami. Časové zaradenie jednotlivých segmentov je možné upravovať priamo vo zvukovej stope. Rovnako je možné pridávať úplne nové segmenty.

Rozhranie jasne odráža primárne určenie nástroja – je optimalizované pre presnú prácu s časovými kódmi, nie pre plynulú editáciu textu.

Výrazným nedostatkom je notifikačný model. Používateľ sa o dokončení prepisu dozvie až z emailu, bez akejkoľvek indikácie postupu spracovania v aplikácii samotnej. SpokenData je tak vhodná skôr pre používateľov, ktorí potrebujú presnú kontrolu nad časovým zaradením segmentov, než pre bežnú editáciu prepisu. Aplikácia nepodporuje mobilné rozhranie, na menších obrazovkách nie je rozloženie prispôsobené a nástroj je určený výhradne pre desktopové použitie.

2.4 Rev

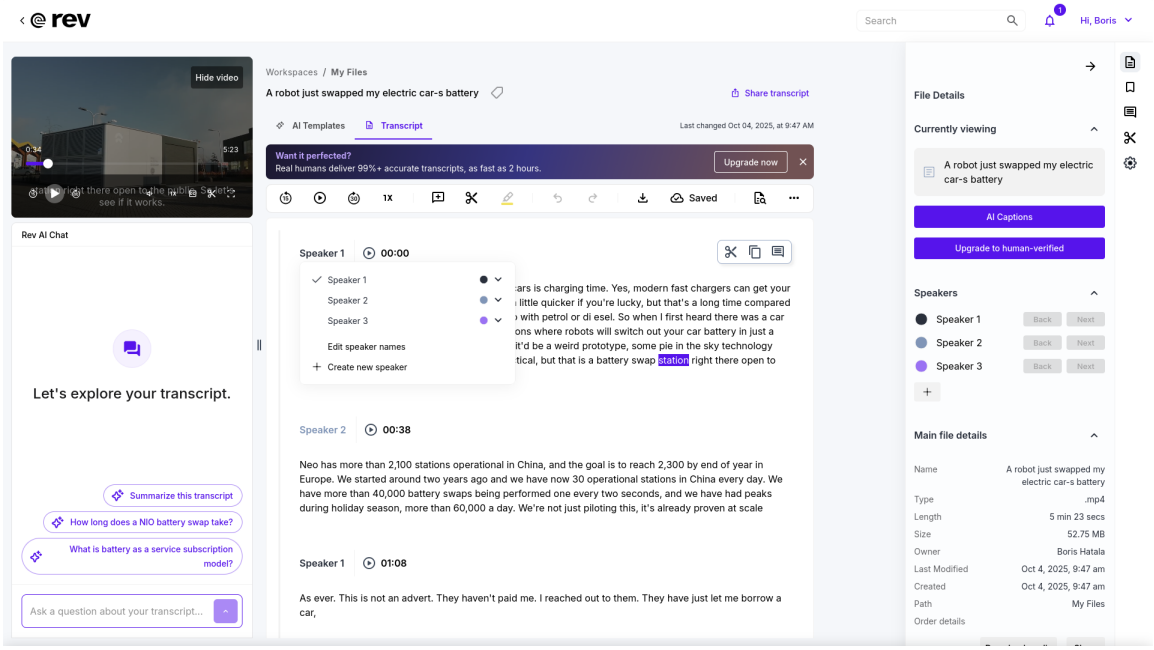
Rev[6] je komplexná platforma pre prepis a analýzu hovoreného slova. Oproti ostatným analyzovaným riešeniam sa odlišuje dôrazom na integráciu umelej inteligencie. Okrem samotného

[6]Domovská stránka služby Rev: https://www.rev.com/

prepisu automaticky generuje súhrn obsahu a umožňuje konverzáciu s AI asistentom priamo v kontexte dokumentu.

Pracovný postup

Webová aplikácia podporuje nahrávanie z mikrofónu priamo v prehliadači; prepis sa zobrazuje v reálnom čase, avšak editácia textu počas nahrávania nie je možná. Používateľ môže taktiež nahrať audiovizuálny súbor alebo poskytnúť odkaz na YouTube video. Rev záznam automaticky spracuje a odošle na prepis. Spracovanie prebieha na strane servera, takže používateľ musí počkať na dokončenie celého prepisu a až potom získa prístup k editoru. Čakacia doba závisí od dĺžky nahrávky a aktuálneho zaťaženia služby. Po dokončení Rev automaticky vygeneruje aj AI súhrn obsahu. Výsledný prepis je možné exportovať vo formátoch PDF, DOCX alebo TXT, prípadne využiť na tvorbu titulkov vo formátoch SRT a VTT.



Obr. 2.5: Screenshot webovej aplikácie Rev

UI a UX

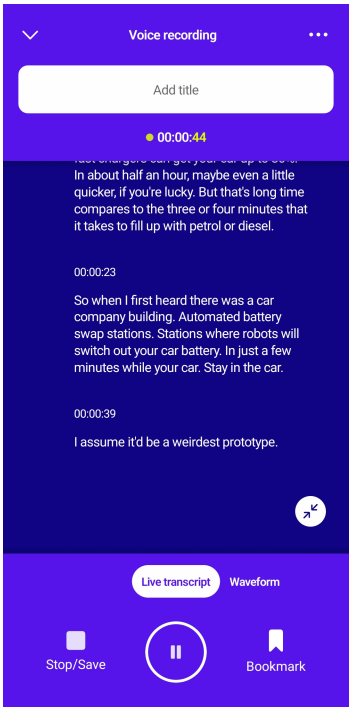
Pracovný priestor webovej aplikácie (na obrázku 2.5) je rozdelený do troch panelov. V ľavej časti sa nachádza video/audio prehrávač a AI chat, cez ktorý môže používateľ klásť otázky k obsahu prepisu alebo požiadať o jeho sumarizáciu. Stred zaberá samotný editor prepisu a pravý panel združuje doplnkové nástroje ako sú záložky, komentáre a klipy, čo sú vyznačené úseky prepisu s odkazom na príslušné miesto v nahrávke.

Synchronizácia audia a prepisu je obojsmerná: počas prehrávania je aktuálne hovorené slovo v texte zvýraznené, a naopak, kliknutím na slovo v prepise sa prehrávač presunie na príslušné miesto v nahrávke. Toto rozvrhnutie umožňuje pracovať s prepisom, AI asistentom aj poznámkami bez prerušenia pracovného toku. Správa rečníkov je riešená veľmi prakticky,

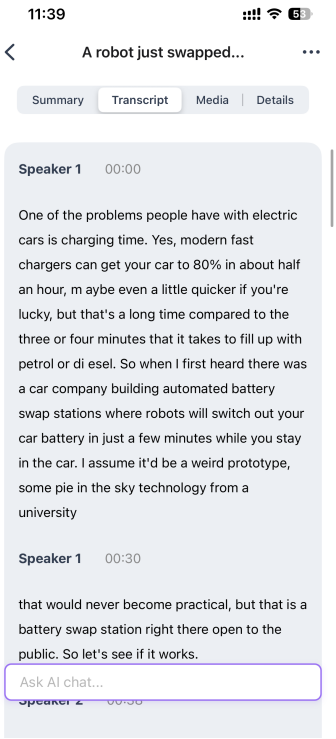
priamo v editore formou vyskakovacieho okna pri príslušnom segmente, nie cez samostatný modálny dialóg ako napríklad v Beey.

Rev je spomedzi analyzovaných nástrojov funkčne najbohatší, kombinuje editáciu prepisu, AI analýzu a anotačné nástroje v jednom rozhraní. Je vhodný pre používateľov, ktorí potrebujú nielen prepis, ale aj ďalšiu prácu s jeho obsahom.

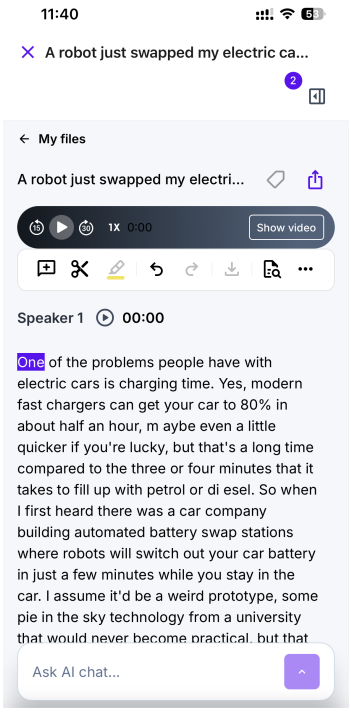
Rev je dostupný aj ako mobilná aplikácia (na obrázku 2.6), ktorá kladie dôraz na živé nahrávanie – prepis sa zobrazuje v reálnom čase počas nahrávania, avšak editácia textu počas nahrávania nie je možná. Po ukončení nahrávania Rev automaticky vygeneruje AI súhrn. Výsledný prepis je v predvolenom zobrazení (na obrázku 2.7) možné iba prehliadať, nie priamo upravovať. Aplikácia umožňuje prepnúť do zobrazenia podobného webovej verzii (na obrázku 2.8), ktoré sprístupňuje editor prepisu, avšak niektoré funkcie sú v tomto režime nedostupné – napríklad pridávanie záložiek.



Obr. 2.6: Mobilná aplikácia Rev počas nahrávania z mikrofónu.



Obr. 2.7: Zobrazenie prepisu v mobilnej aplikácii Rev po ukončení nahrávania .



Obr. 2.8: Alternatívna podoba UI v mobilnej aplikácii Rev podobná webovej verzii.

2.5 Záver

Analyzované nástroje pokrývajú širokú škálu prístupov k prepisu reči a líšia sa nielen funkciami, ale aj cieľovou skupinou používateľov. Porovnanie kľúčových vlastností je zhrnuté v tabuľke 2.1.

Kategória / Nástroj	**Beey**	**Speechnotes**	**SpokenData**	**Rev**
Real-time prepis	✓	✓	×	✓
Automatická diarizácia	✓	×	×	✓
Práca s časovými značkami	✓	×	✓	✓
Vytváranie titulkov	✓	×	✓	✓
Prehranie audio záznamu	✓	×	✓	✓
Prehranie video záznamu	×	×	✓	✓
AI funkcie	✓	×	×	✓

Tabuľka 2.1: Porovnanie funkcionalít editorov prepisu reči.

Speechnotes predstavuje minimalistické riešenie bez serverovej závislosti; táto jednoduchosť sa však prejavuje absenciou pokročilejších funkcií – chýba synchronizácia so zvukom, formátovanie textu aj práca s existujúcimi nahrávkami. SpokenData je naopak optimalizovaná na presnú prácu s časovými kódmi a segmentáciou, čo z nej robí nástroj vhodný skôr na úpravu časovania než na plynulú editáciu textu. Beey a Rev predstavujú komplexnejšie riešenia: Beey sa sústreďuje na editáciu prepisu s podporou diarizácie a titulkov, Rev navyše integruje AI chat aj anotačné nástroje do jedného pracovného priestoru. Dostupnosť mobilnej verzie, ktorú demonštruje práve Rev, výrazne rozširuje použiteľnosť nástroja.

Napriek týmto rozdielom zdieľa väčšina nástrojov dve spoločné obmedzenia. Prvým je závislosť od vzdialených serverov – nahrávky aj prepisy sú ukladané mimo zariadenia používateľa. Druhým je nemožnosť editovať prepis počas jeho vytvárania pri spracovaní existujúcich nahrávok. Tieto zistenia sú priamou motiváciou pre návrh vlastného riešenia.

Kapitola 3

Návrh riešenia

Táto kapitola sa venuje návrhu aplikácie. Vychádza z analýzy existujúcich editorov prepisu reči. Na ich základe sú najprv definované požiadavky na výsledné riešenie, následne je navrhnutá architektúra systému a dátový model.

3.1 Požiadavky aplikácie

Aplikácia by mala byť jednoduchá na používanie a zvládnuteľná aj pre ľudí s minimálnou počítačovou gramotnosťou.

Hlavným cieľom aplikácie je poskytovať nástroj na vytváranie prepisu hovoreného slova s pokročilými editačnými možnosťami v reálnom čase. Aplikácia musí podporovať spracovanie existujúcich audiovizuálnych súborov, ako aj živé nahrávanie.

Funkčné požiadavky

- **Prepis reči:** Automatický prepis hovoreného slova do textovej podoby v reálnom čase, s podporou nahrávania z mikrofónu aj spracovania existujúcich audio súborov. Pri spracovaní súboru sú podporované dva režimy: priebežný real-time prepis, pri ktorom výsledky prichádzajú postupne a je možné ich okamžite editovať, a dávkový prepis, ktorý počká na kompletný výsledok vrátane automatického rozpoznania rečníkov.
- **Diarizácia:** Rozpoznanie a vizuálne odlíšenie jednotlivých rečníkov v prepise.
- **Editor prepisu:** Úprava textu prepisu so základným formátovaním a správou rečníkov.
- **Synchronizácia audia a textu:** Počas prehrávania záznamu je aktuálne hovorený úsek zvýraznený v texte; presunutím kurzora v editore sa zmení pozícia prehrávania.
- **Audio prehrávač:** Prehrávanie nahraného záznamu so štandardnými ovládacími prvkami.
- **Záložky:** Možnosť označiť kľúčové miesta v prepise previazané s časovou pozíciou v zázname.
- **Export:** Export finálneho prepisu do bežných textových formátov a možnosť skopírovania textu do schránky a stiahnutie zvukového záznamu.

Nefunkčné požiadavky

- **Lokálne ukladanie dát:** Audio nahrávky aj dokumenty sú ukladané výhradne v zariadení používateľa – dáta neopúšťajú jeho zariadenie. Aplikácia nevyžaduje registráciu ani prihlásenie.
- **Bez vlastného backendu:** Aplikácia nevyžaduje vlastnú serverovú infraštruktúru a môže byť nasadená ako statická webová stránka.
- **Responzívne rozhranie:** Aplikácia je použiteľná na mobilných zariadeniach bez potreby inštalácie natívnej aplikácie.

3.2 Architektúra systému

Aplikácia nemá vlastnú serverovú infraštruktúru, nevyžaduje backend ani databázu na strane servera. Jedinou externou závislosťou je služba automatického prepisu reči, ktorá spracúva audio na svojich serveroch a vracia výsledky do prehliadača. Všetky ostatné dáta zostávajú lokálne v zariadení používateľa.

Pre automatický prepis reči je v architektúre počítané s externou cloudovou službou; výber konkrétnej služby je súčasťou implementačnej fázy.

Komponenty

Aplikácia sa skladá zo štyroch hlavných komponentov:

- **Správa dokumentov** – vytváranie, načítavanie, ukladanie a odstraňovanie dokumentov. Predstavuje vstupný bod aplikácie.
- **Editor prepisu** – rich-text editor, v ktorom sa zobrazuje a upravuje prepis. Uchováva štruktúru textu vrátane formátovania, priradenia rečníkov a záložiek.
- **Prehrávanie audia** – načítanie a prehrávanie zvukového záznamu so synchronizáciou s textom prepisu.
- **Prepis reči** – komunikácia s externou prepísovou službou, príjem výsledkov a ich vkladanie do editora.

Komponenty sú navzájom previazané: prepis reči produkuje obsah, ktorý editor zobrazuje a ukladá; prehrávanie audia je synchronizované s pozíciou v editore; správa dokumentov zastrešuje perzistenciu dát zo všetkých ostatných komponentov.

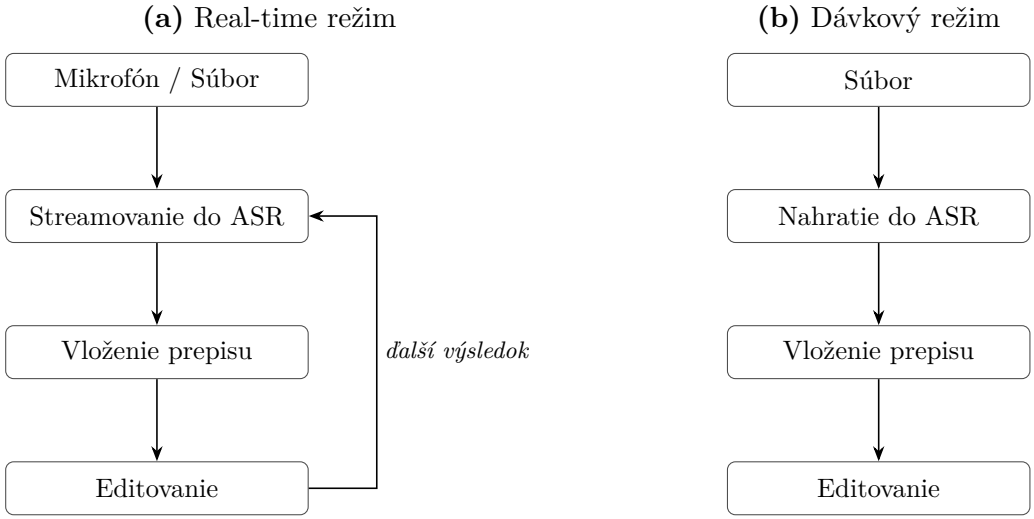
Tok dát

Aplikácia podporuje dva zdroje audia: nahrávanie z mikrofónu a načítanie zvukového súboru a dva režimy prepisu: real-time a dávkový.

V **real-time režime** je audio streamované na externú prepísovú službu, ktorá priebežne vracia čiastkové výsledky. Tie sú ihneď vkladané do editora a používateľ ich môže upravovať ešte počas prebiehajúceho prepisu. Tento režim je k dispozícii pri nahrávaní z mikrofónu aj pri spracovaní zvukového súboru. Pri nahrávaní z mikrofónu je možné nahrávanie opakovane spustiť a zastaviť. Každý úsek je doplnený na koniec dokumentu.

V **dávkovom režime**, ktorý je dostupný len pri spracovaní zvukového súboru, je celý súbor odovzdaný na spracovanie naraz a výsledok je vložený do editora až po dokončení celého prepisu.

V oboch prípadoch zostáva dokument po skončení prepisu naďalej k dispozícii na editáciu. Tok dát sa líši podľa zvoleného režimu, čo znázorňuje obrázok 3.1.



Obr. 3.1: Tok dát v aplikácii. Pri nahrávaní z mikrofónu (a) je možné nahrávanie opakovane spúšťať. Každý cyklus doplní prepis na koniec dokumentu. Real-time spracovanie súboru sleduje rovnaký tok bez možnosti opätovného spustenia nahrávania alebo pridania ďalšieho súboru do dokumentu. V dávkovom režime (b) sa čaká na kompletný prepis pred vložením do editora.

Ukladanie dát

Dokument je ukladaný automaticky po každej zmene, používateľ ho nemusí ukladať manuálne. Aplikácia pracuje s dvomi typovo odlišnými druhmi dát, ktoré kladú rozdielne požiadavky na úložisko.

Audio súbory môžu dosahovať veľkosti stovky megabajtov. Pre ich ukladanie je zvolené OPFS (Origin Private File System) — súborové úložisko prehliadača optimalizované pre binárne dáta, ktoré umožňuje priamy prístup k obsahu súboru na úrovni bajtov a priebežné dopĺňanie dát na koniec súboru počas nahrávania. OPFS je izolované od bežného súborového systému zariadenia a prístupné výlučne cez webové API prehliadača.

Dokumenty sú naopak relatívne malé štruktúrované záznamy, pri ktorých je dôležitá rýchlosť čítania a jednoduchosť dotazovania. Ukladajú sa do IndexedDB — transakčnej databázy zabudovanej v prehliadači, ktorá je navrhnutá pre efektívne ukladanie a načítavanie štruktúrovaných objektov. Každý dokument je uložený ako samostatný záznam identifikovaný unikátnym kľúčom, čo umožňuje rýchle vyhľadávanie bez prehľadávania celej kolekcie.

3.3 Dátový model

Aplikácia pracuje primárne s dokumentami, ktoré obsahujú prepis záznamu a metadáta. Dátový model je navrhnutý ako hierarchická štruktúra, ktorá umožňuje uchovávať pôvodný automaticky generovaný prepis aj jeho používateľom upravenú verziu.

Dokument sa delí na dve hlavné časti:

- **Originálny transkript:** uchováva dáta získané od prepisovej služby. Prepis je uložený jednak ako plný text, jednak ako zoznam segmentov (*utterances* – súvislých rečových úsekov jedného rečníka), pričom každý segment je ďalej členený na jednotlivé slová. Segmenty aj slová nesú časové značky udávajúce ich polohu v zázname. Táto štruktúra umožňuje presnú synchronizáciu textu s audiom.
- **Editovaný transkript:** predstavuje verziu prepisu upravenú používateľom. Je uložený ako strom uzlov rich-text editora, kde každý blokový uzol zodpovedá súvislému rečovému úseku jedného rečníka. Textové uzly nesú formátovacie atribúty a časové značky prevzaté z originálneho transkriptu, ktoré slúžia na synchronizáciu s audiom. Záložky sú reprezentované ako inline uzly vložené priamo do toku textu.

Rozdelenie na originálny a editovaný transkript je dôležité pre zachovanie pôvodných časových značiek prepisu, ktoré sú nevyhnutné pre synchronizáciu audia s textom. Štruktúra dátového modelu je znázornená vo výpise 3.1.

```
{
  id, názov, dátumy, zdroj vstupu
  originálny transkript: {
    segment[]: {
      rečník, začiatok, koniec
      slovo[]: { text, začiatok, koniec }
    }
  }
  editovaný transkript: {
    blok rečníka[]: {
      rečník
      slovo[]: { text, začiatok, koniec }
      záložka[]: { názov, text, začiatok, koniec }
    }
  }
}
```

Výpis 3.1: Dátový model dokumentu

3.4 Záver

Na základe definovaných požiadaviek bola zvolená architektúra bez vlastného backendu, aplikačná logika beží v prehliadači a dáta sú ukladané lokálne. Kľúčovým rozhodnutím v dátovom modeli je oddelenie originálneho prepisu od editovaného, čo zachováva pôvodný výstup prepísovej služby nezávisle od úprav používateľa. Na základe tohto návrhu sa nasledujúca kapitola venuje návrhu užívateľského rozhrania.

Kapitola 4

Užívateľské rozhranie

Táto kapitola sa zaoberá návrhom užívateľského rozhrania aplikácie. Návrh vychádza z požiadaviek definovaných v kapitole 3 a ich cieľom je vytvoriť jednoduché, konzistentné prostredie, ktoré umožní efektívnu prácu s prepisom hovoreného slova. Aplikácia je primárne určená na desktopové použitie, mobilná verzia musí byť napriek tomu plne funkčná a použiteľná. Rozhranie je navrhnuté podľa bežných konvencií webových aplikácií. Kľúčové prvky sú umiestnené na miestach, kde ich používateľ prirodzene očakáva, a dôraz je kladený na rýchlu orientáciu a okamžitú spätnú väzbu. Návrh prebiehal iteratívne s postupným ladením.

4.1 Rozloženie hlavnej stránky

Používatelia webové stránky spravidla nečítajú lineárne, ale skenujú ich obsah so zámerom čo najrýchlejšie dosiahnuť svoj cieľ [3]. V prípade editora prepisu reči je týmto cieľom rýchle spustenie procesu transkripcie a následná efektívna práca s vytvoreným prepisom.

Hlavná stránka je rozdelená na dve hlavné časti, bočný panel a pracovnú plochu. Tento návrh využíva princíp *Center Stage*, kde je najdôležitejší obsah, samotný dokument s prepisom, umiestnený do centrálnej a najväčšej časti obrazovky. Zatiaľ čo sekundárne nástroje a ovládacie panely sú umiestnené okolo neho. Takto je zaručené, že pozornosť používateľa je okamžite sústredená na hlavnú úlohu a ostatné prvky sú dostupné, ale neodvádzajú zbytočne pozornosť [6].

Bočný panel

Bočný panel slúži primárne na navigáciu v aplikácii. Obsahuje odkazy na vytvorenie nového prepisu a prístup k všetkým dokumentom používateľa. Panel je možné roztiahnuť a stiahnuť, ak by používateľ potreboval viac miesta. Je navrhnutý tak, aby bol vždy dostupný a umožňoval rýchly presun medzi jednotlivými pohľadmi bez narušenia pracovného toku používateľa.

Pracovná plocha

Pracovná plocha predstavuje centrálnu časť aplikácie a obsahuje všetky prvky potrebné na editáciu prepisu reči. Dominantným prvkom je samotný dokument, v ktorom sa zobrazuje automaticky generovaný prepis a kde používateľ vykonáva jeho úpravy. V prvotnom návrhu bol dokument umiestnený symetricky na stred obrazovky. Neskôr bol posunutý viac

k ľavému okraju, aby pravá časť obrazovky mohla byť využitá na paralelnú prácu s inými aplikáciami alebo materiálmi.

Okrem editora dokumentu pracovná plocha obsahuje aj doplnkové prvky, ktoré podporujú efektívnu prácu s prepisom:

- **Panel nástrojov pre úpravu textu:** umožňuje základné formátovanie textu ako tučné písmo alebo šikmé písmo.
- **Audio prehrávač:** slúži na prehrávanie pôvodného audiozáznamu. Pozícia prehrávania je synchronizovaná s pozíciou kurzora v texte. Počas prehrávania je aktuálna časť prepisu vizuálne zvýraznená.
- **Panel ovládacích prvkov:** obsahuje akcie ako stiahnutie dokumentu, kopírovanie textu, správu rečníkov a záložiek.

4.2 Vývoj užívateľského rozhrania

Hlavnou inšpiráciou pre návrh užívateľského rozhrania boli aplikácie *Beey* a *Rev*, ktoré pri analýze existujúcich riešení v kapitole 2 pôsobili najprehľadnejšie a najpohodlnejšie na používanie.

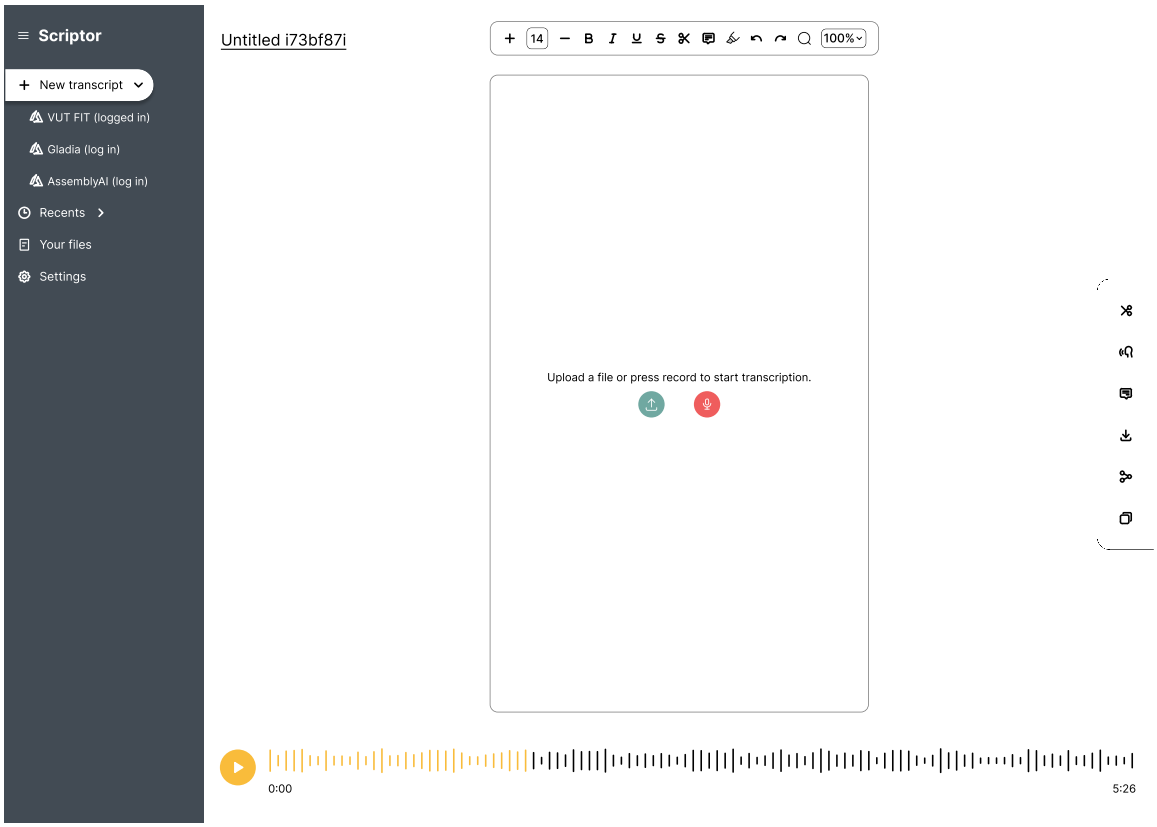
Z aplikácie *Beey* bolo prevzaté celkové rozloženie editora, spôsob zvýraznenia práve prehrávaného textu počas synchronizácie prepisu s audiom a klávesová skratka *Tab* na spustenie a pozastavenie prehrávania.

Z aplikácie *Rev* bol prevzatý spôsob farebného rozlíšenia jednotlivých rečníkov v prepise, princíp správy rečníkov priamo v rozhraní editora a malé tlačidlá prehrávania pri jednotlivých paragrafoch rečníkov, ktoré umožňujú rýchle prehranie konkrétneho úseku.

Ako vývojové prostredie pre návrh UI bol zvolený nástroj *Figma*, ktorý je štandardom pri návrhu webových rozhraní a umožňuje tvorbu interaktívnych prototypov s okamžitou vizuálnou spätnou väzbou.

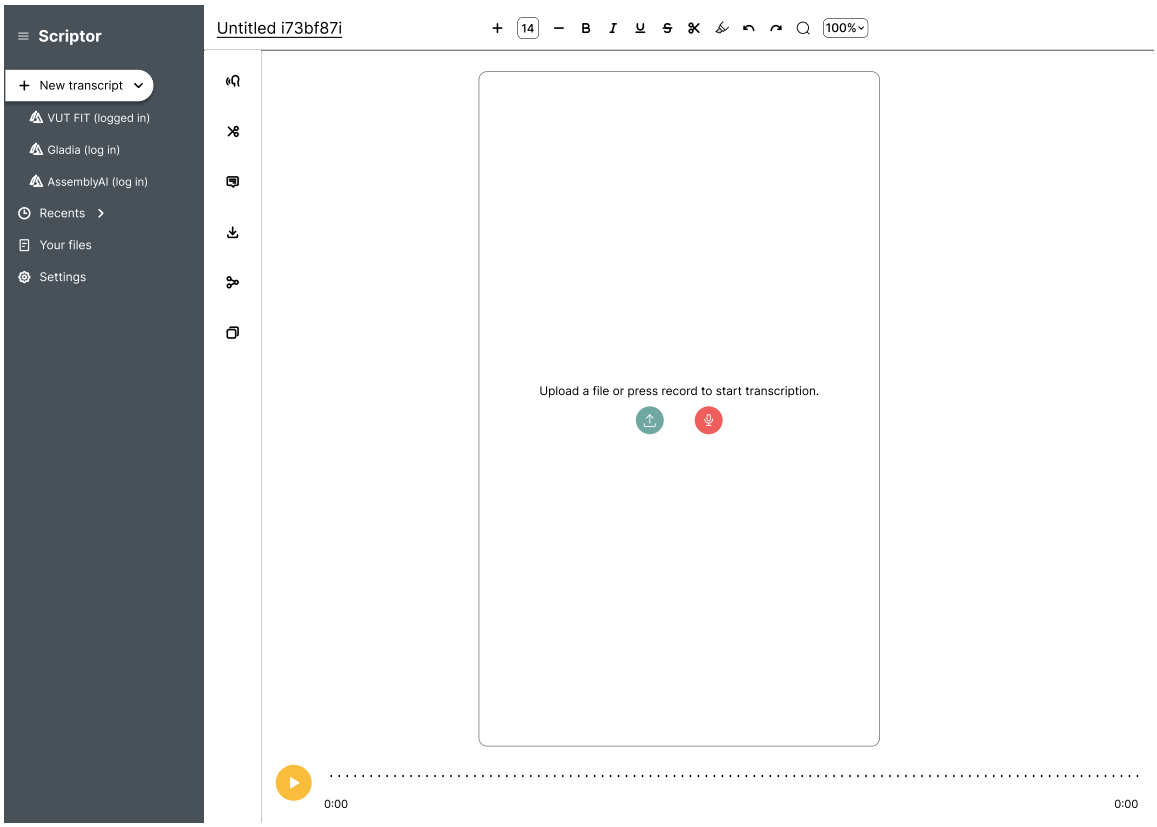
Desktopová verzia

Na obrázku 4.1 je zobrazená prvá iterácia návrhu užívateľského rozhrania. Rozloženie prvkov vychádza z bežných konvencií webových aplikácií a nástrojov na editáciu textu: bočný panel s navigáciou je umiestnený naľavo, čo zodpovedá štandardnému vzoru väčšiny webových aplikácií. Panel nástrojov pre formátovanie textu je umiestnený nad dokumentom, analogicky k najpoužívanejším textovým editorom ako *Microsoft Word* alebo *Google Docs*. Audio prehrávač sa nachádza v dolnej časti obrazovky, analogicky k rozšíreným mediálnym platformám ako YouTube alebo Spotify. Panel s ovládacími prvkami bol v tejto iterácii umiestnený napravo, inšpirovaný rozhraním aplikácie *Rev*.



Obr. 4.1: Prvá iterácia návrhu UI: bočný panel naľavo, panel nástrojov nad dokumentom, ovládací panel napravo a audio prehrávač v dolnej časti

V druhej iterácii návrhu, znázornenej na obrázku 4.2, bol panel s ovládacími prvkami presunutý. Keďže *F-pattern* model popisuje pohyb zraku zľava doprava a zhora nadol, prvky umiestnené vpravo dostávajú menej pozornosti [4]. Panel bol preto presunutý bližšie k ľavému okraju, kde je v súlade s prirodzeným skenovaním obrazovky dostupnejší.

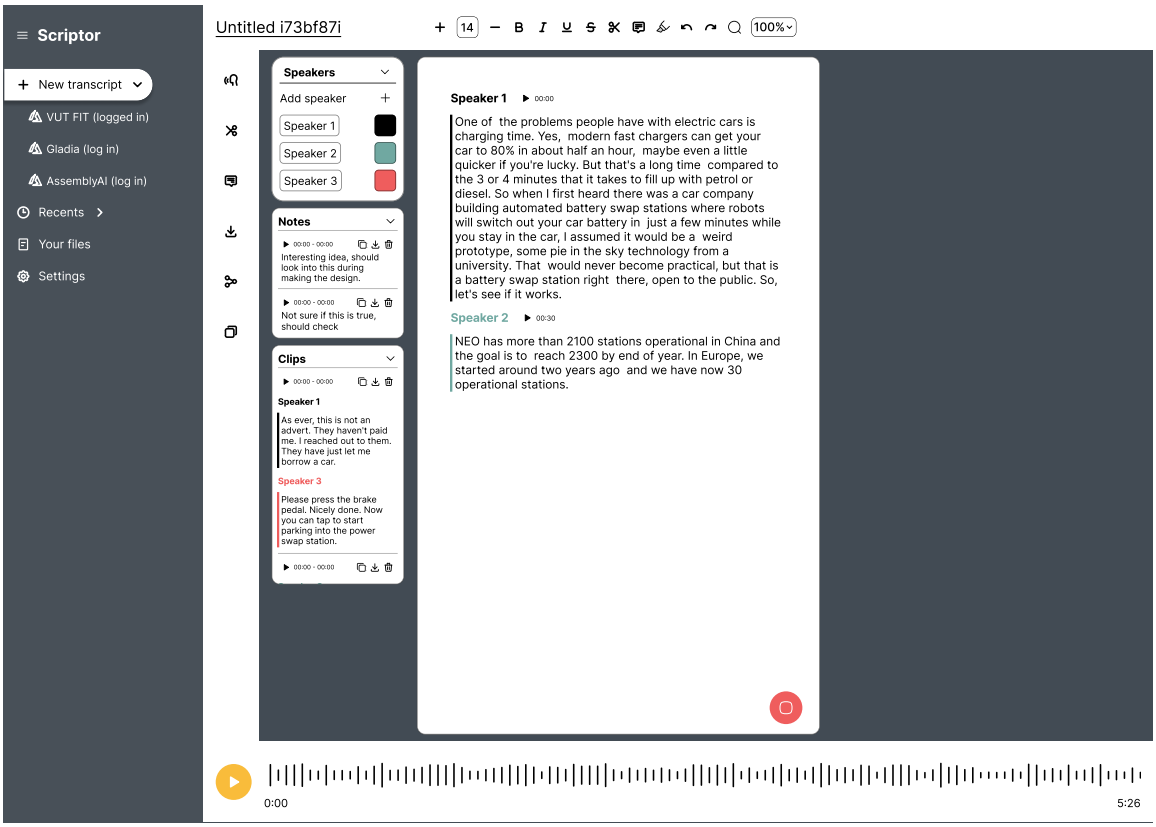


Obr. 4.2: Druhá iterácia návrhu UI: ovládací panel presunutý z pravej strany bližšie k ľavému okraju v súlade s *F-pattern* modelom skenovania

V tretej iterácii návrhu, zobrazenej na obrázku 4.3, boli realizované predovšetkým štylistické úpravy. Rozhranie používa tmavé pozadie, na ktorom vynikajú biele plochy dokumentu a panelov – tento kontrast dodáva rozhraniu vizuálnu hĺbku a zlepšuje čitateľnosť obsahu.

Zároveň bolo rozloženie pracovnej plochy viac orientované na ľavú stranu obrazovky. Hlavným dôvodom je praktická ergonómia: sústredenie ovládacích prvkov na ľavej strane skracuje vzdialenosť, ktorú musí myš prekonať pri bežných úkonoch, a zároveň uvoľňuje pravú časť obrazovky na paralelnú prácu s inými aplikáciami alebo zdrojovými materiálmi.

Panel s ovládacími prvkami je implementovaný ako *prepínací panel (angl. toggle bar)*, ktorý umožňuje používateľovi dynamicky zvoliť, ktoré moduly chce mať zobrazené na pracovnej ploche. Týmto spôsobom nie je pracovný tok používateľa narušený modálnymi oknami a rozhranie sa dokáže prispôsobiť individuálnym potrebám. Používateľ môže rečníkov spravovať buď priamo v dokumente, alebo vedľa v samostatnom module.



Obr. 4.3: Tretia iterácia návrhu UI: tmavé pozadie s bielymi plochami dokumentu a panelov pre lepší kontrast, rozloženie sústredené na ľavú stranu obrazovky

Mobilná verzia

Návrh mobilnej verzie rozhrania bol vypracovaný až po ukončení testovania použiteľnosti desktopovej verzie (kapitola 8). Vychádza zo stavu aplikácie po zapracovaní úprav plynúcich z testovania.

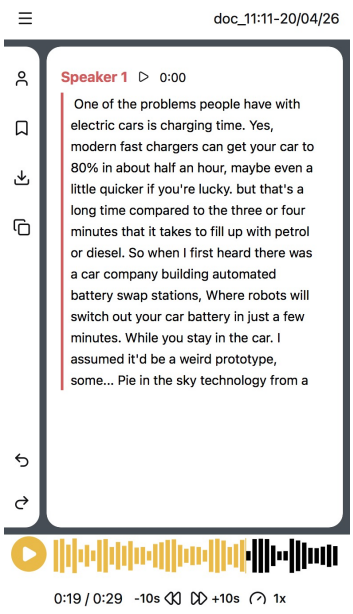
Na menších obrazovkách nie je možné zachovať široké rozloženie desktopovej verzie (obrázok 4.4). Preto je bočný panel skomprimovaný na vysúvací panel (obrázok 4.5), ktorý sa vysunie pri klepnutí na ikonu menu v záhlaví podobne ako v mobilnej verzii *Beey*. To umožňuje pohodlnú navigáciu bez trvalého zaberania priestoru pracovnej plochy.

Záhlavie muselo kvôli nedostatku priestoru tiež prejsť zmenou. Pri náhľade na prepis zobrazuje editovateľný názov dokumentu a ikonu menu. Pri editácii textu sa po otvorení klávesnice záhlavie nahradí panelom nástrojov na formátovanie (obrázok 4.6), čím má používateľ priamy prístup k základným možnostiam formátovania bez nutnosti zatvárať klávesnicu.

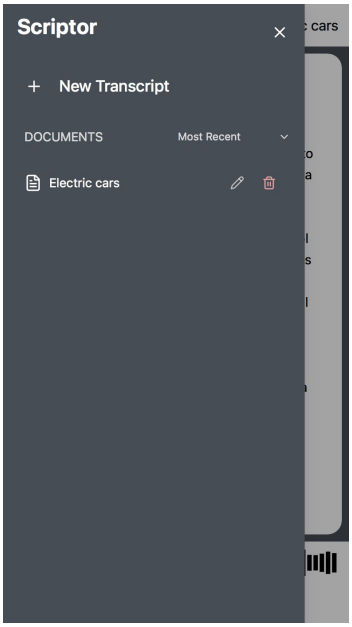
Panel nástrojov na ľavej strane obrazovky okrem tlačidiel na zobrazenie modulov obsahuje aj tlačidlá krok späť a krok vpred. Keďže ide o globálne operácie nad dokumentom, ktoré nie sú viazané iba na aktívnu editáciu, bolo by nepohodlné ich sprístupniť len v panely nástrojov na formátovanie.

Poloha prehrávača ostala nezmenená, nachádza sa dolnej časti obrazovky, analogicky k desktopovej verzii.

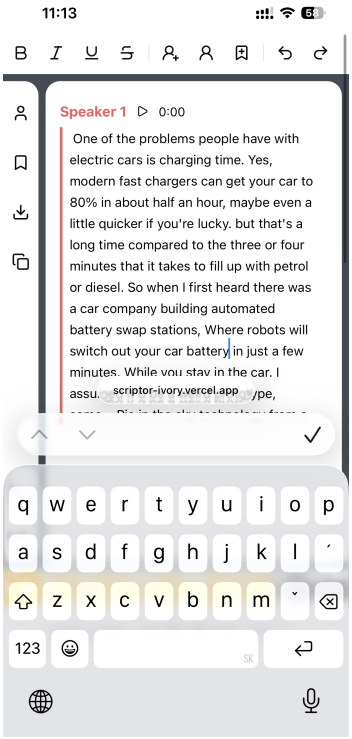
Všetky funkcionality desktopovej verzie boli zachované s výnimkou zmeny farby textu a zmeny farby pozadia textu. Tieto možnosti neboli do mobilného panela nástrojov zahrnuté pre nedostatok priestoru a ich menej zásadný charakter v porovnaní s ostatnými formátovacími voľbami.



Obr. 4.4: Mobilná verzia rozhrania editora.



Obr. 4.5: Bočný panel otvorený ako prekrývajúca zásuvka.



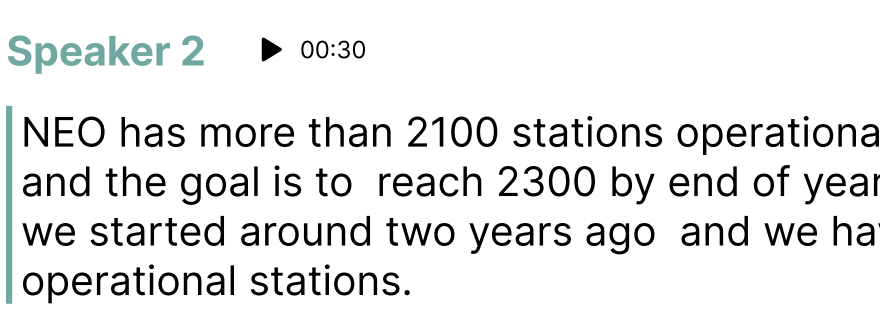
Obr. 4.6: Panel nástrojov na formátovanie pri otvorení klávesnice.

4.3 Funkčné komponenty

Táto sekcia opisuje kľúčové komponenty rozhrania, ktoré sú spoločné pre desktopovú aj mobilnú verziu aplikácie.

Vizuálne rozlíšenie rečníkov

Vizuálne rozlíšenie rečníkov farebným zvýraznením textových blokov bolo prevzaté z aplikácie *Rev* a bolo súčasťou návrhu už od prvej iterácie. Každému rečníkovi je priradená farba zvýrazňujúca príslušný textový blok, ktorý je na ľavej strane oddelený zvislou čiarou. Každý úsek textu obsahuje meno rečníka spolu s ikonou prehrávania s časovou značkou, čo umožňuje rýchle prehranie konkrétnej časti audia bez nutnosti manuálneho vyhľadávania v prehrávači (obrázok 4.7).



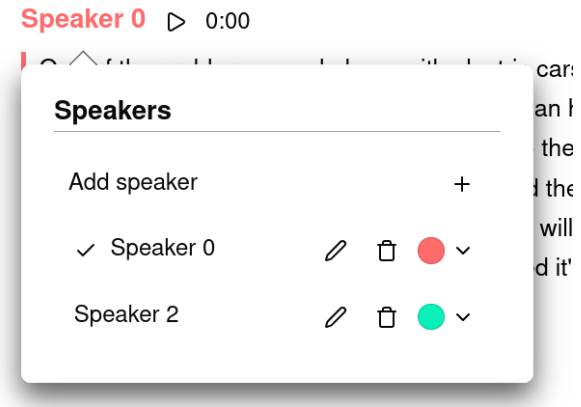
Obr. 4.7: Farebne zvýraznený blok textu jedného rečníka s ikonou prehrávania a časovou značkou začiatku úseku

Spôsob priraďovania rečníkov

Správa rečníkov priamo v rozhraní editora bola inšpirovaná aplikáciou *Rev*. Používateľ môže zmeniť priradenie jednotlivých rečníkov priamo v editore. Pri kliknutí na meno rečníka v dokumente sa zobrazí vyskakovacie okno (obrázok 4.8), ktoré poskytuje kompletnú správu rečníkov. Okno umožňuje existujúcich rečníkov premenovať, zmeniť im farbu alebo ich vymazať. Zároveň je možné pridať úplne nového rečníka. Samotné priradenie rečníka k danému úseku textu sa vykoná kliknutím na jeho meno v zozname.

Vďaka tomuto prístupu sa pozornosť používateľa udržuje v dokumente a nie je rozptýlená modálnymi oknami ako v *Beey*.

V prípade, že úsek danému rečníkovy nepatrí alebo mu patrí iba čiastočne, môže používateľ celý blok odstrániť klávesom *Backspace* alebo *Delete*, prípadne označiť konkrétnu časť textu a priradiť jej správneho rečníka.



Obr. 4.8: Vyskakovacie okno správy rečníkov zobrazené po kliknutí na meno rečníka; umožňuje premenovanie, zmenu farby a priradenie rečníka k danému úseku

Audio prehrávač

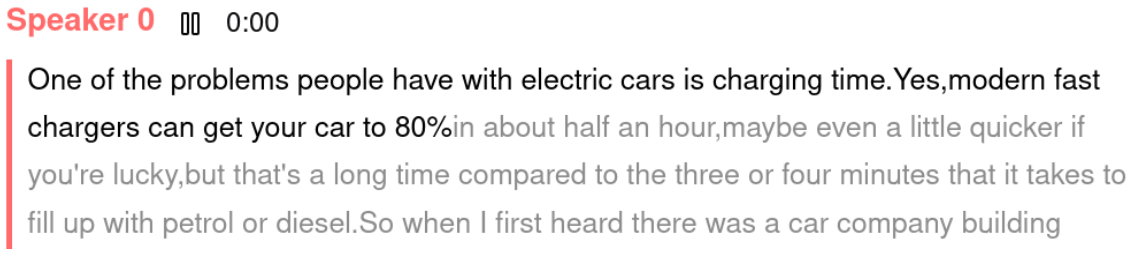
Prehrávač audia zobrazuje priebeh zvukového záznamu formou vizualizácie zvukových vĺn a poskytuje základné ovládacie prvky: spustenie a pozastavenie prehrávania, posun na konkrétne miesto v nahrávke a indikátor aktuálnej časovej pozície. Vizualizácia umožňuje používateľovi rýchlo identifikovať tichšie alebo hlasnejšie úseky nahrávky a navigovať v nej priamo kliknutím na priebeh signálu.



Obr. 4.9: Audio prehrávač s vizualizáciou zvukových vĺn; kliknutím na priebeh signálu možno skočiť na ľubovoľné miesto v nahrávke

Synchronizácia prepisu s nahrávkou

Aby sa používateľ v prepise dobre orientoval, aktuálna poloha v nahrávke je priebežne synchronizovaná s prepisom v editore. Slová, ktoré v nahrávke už zazneli, sú zobrazené čiernou farbou a zvyšok prepisu je zošednutý, táto funkcionalita bola prevzaná z editora *Beey*. Zvýraznenie sa aktualizuje plynule slovo po slove počas prehrávania, čím používateľ vždy vidí presne, ktoré miesto v texte práve znie.



Obr. 4.10: Synchronizácia prepisu s nahrávkou: odznené slová sú zvýraznené čiernou farbou, zvyšok prepisu je zošednutý

Záložky

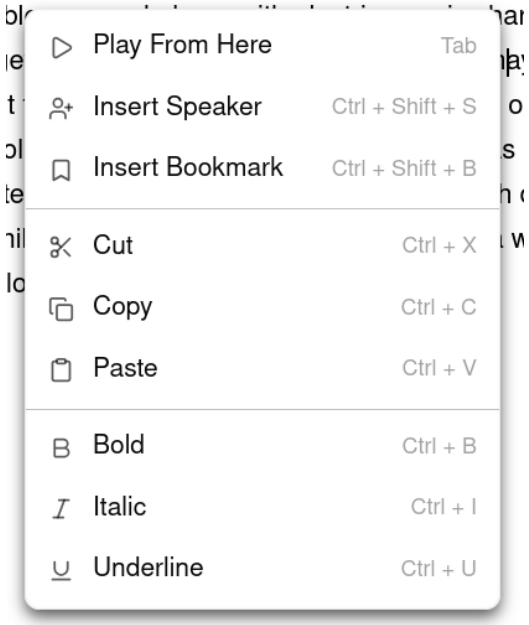
Záložka predstavuje označený bod priamo v tele prepisu, ku ktorému sa používateľ môže kedykoľvek rýchlo vrátiť. Vkladá sa na aktuálnu pozíciu kurzora a v texte je zobrazená ako ikona (obrázok 4.11), čím nijako nenarúša čitateľnosť prepisu. Záložky sú prístupné aj zo zoznamu v bočnom paneli, odkiaľ je možné priamo preskočiť na príslušné miesto v dokumente.



Obr. 4.11: Záložka zobrazená ako ikona priamo v tele prepisu bez narušenia čitateľnosti textu

Kontextové menu

Inšpiráciou pre kontextové menu bola aplikácia *Rev*, ktorá sprístupňuje akcie editora priamo v kontexte dokumentu. Dokument poskytuje vlastné kontextové menu (obrázok 4.12), ktoré sa zobrazí po kliknutí pravým tlačidlom myši na ľubovoľné miesto v prepise. Menu je rozdelené do troch skupín akcií. Prvá skupina obsahuje akcie špecifické pre editor prepisu: spustenie prehrávania od daného miesta, vloženie rečníka a vloženie záložky, všetky s uvedenými klávesovými skratkami. Druhá skupina pokrýva štandardné operácie so schránkou – vystrihnutie, kopírovanie a vkladanie. Tretia skupina sprístupňuje základné formátovanie textu.



Obr. 4.12: Kontextové menu rozdelené do troch skupín: akcie editora prepisu, operácie so schránkou a formátovanie textu

4.4 Záver

Rozhranie vznikalo iteratívne, pričom každá iterácia priniesla konkrétne zmeny – od presunu ovládacieho panela bližšie k ľavému okraju cez posunutie dokumentu mimo stredu až po finálne štylistické doladenie s tmavým pozadím. Výsledné rozhranie je postavené na princípe *Center Stage*: editor prepisu zaberá hlavnú časť pracovnej plochy a pomocné nástroje sú dostupné bez toho, aby narušovali pracovný tok. Rozmiestnenie prvkov zohľadňuje prirodzený pohyb zraku podľa *F-pattern* modelu a možnosť dynamicky skryť sekundárne panely nenarúša pracovný tok. Desktopová verzia bola následne adaptovaná pre mobilné zariadenia – bočný panel sa transformoval na vysúvací panel a panel nástrojov sa pri editácii zobrazuje v záhlaví pri vysunutí klávesnice pričom všetky kľúčové funkcie boli zachované. Takto vzniknutý návrh tvoril priamy podklad pre implementáciu opísanú v nasledujúcej kapitole.

Kapitola 5

API služby pre prepis reči

Táto kapitola vytvára prehľad rozhraní a služieb pre prepis reči (STT[1]), ktoré sú potenciálne vhodné pre integráciu do webovej aplikácie pracujúcej v reálnom čase. Pri výbere sa prihliadalo najmä na podporu real-time transkripcie a rozsah doplnkových funkcií. Pre každú zo služieb je popísaný spôsob práce s API a formát odpovede.

5.1 Gladia

Gladia[2] je poskytovateľ služieb umelej inteligencie so zameraním na prepis a analýzu zvuku. Ponúka (*Real-Time Speech-to-Text API*) pre živý prepis a (*Asynchronous Speech-to-Text API*) pre spracovanie nahratých súborov. Táto časť sa venuje (*Real-Time Speech-to-Text API*).

Funkcie a schopnosti

Služba garantuje ultranízku latenciu pod 300 milisekúnd, čo je kľúčové pre plynulú konverzáciu. Podporuje prepis vo vyše 100 jazykoch a disponuje pokročilými funkciami, ako sú

- **Automatická detekcia jazyka:** Služba dokáže automaticky detekovať jazyk buď spomedzi všetkých podporovaných jazykov, alebo medzi užívateľom vybranými kandidátmi.
- **Preklad:** Pomocou jediného volania API môže užívateľ preložiť svoj prepis do viacerých jazykov.
- **Multijazyčná podpora:** Umožňuje prepis nahrávok, v ktorých rečník striedavo prechádza medzi viacerými jazykmi počas jedného záznamu.
- **Vlastná slovná zásoba:** V prípade, že užívateľ očakáva vyšší výskyt niektorých slov, môže tieto termíny, ich výslovnosť a jazyk definovať a zlepšiť presnosť prepisu.
- **Okamžité výsledky:** Sú to okamžité reakcie spracované menším modelom s latenciou pod 100 ms. Následne príde finálny prepis, po spracovaní celého úseku.

[1] Speech-to-Text
[2] Domovská stránka služby Gladia: https://docs.gladia.io/chapters/introduction

Používanie služby

Pre prácu so službou je potrebný autentifikačný API kľúč. Gladia poskytuje oficiálne JavaScript/TypeScript a Python SDK[3], ktoré zjednodušuje integráciu do webových aplikácií. Dostupné je aj REST API.

Formát odpovede

Výsledky prichádzajú ako priebežné JSON správy. Každý prepísový úsek obsahuje časové ohraničenie, mieru istoty rozpoznania a identifikátor jazyka. Úsek je ďalej členený na jednotlivé slová, pričom každé slovo nesie vlastné časové značky a mieru istoty.

5.2 Azure AI

Azure AI Speech[4] je cloudová služba poskytovaná spoločnosťou Microsoft. Predstavuje sadu nástrojov pracujúcich s hlasom: prepis reči na text (*Speech-to-text*), syntéza reči (*Text-to-speech*) alebo preklad (*Speech translation*). V kontexte tejto práce sa sústreďuje pozornosť len na aspekt prepisu reči na text.

Samotná funkcia *speech-to-text* má tri módy:

- **Real-time prepis:** Prepis živého audia z mikrofónu alebo audio súboru.
- **Rýchly prepis:** Synchrónne spracovanie krátkych audio záznamov, pričom výsledok je k dispozícii rýchlejšie ako v reálnom čase.
- **Hromadný prepis:** Asynchrónna metóda určená na spracovanie veľkého množstva audia, vhodná pre tvorbu titulkov alebo analýzu hovorov call centra.

Funkcie a schopnosti

Služba podporuje viac ako 100 jazykov a disponuje nasledujúcimi funkciami:

- **Automatická detekcia jazyka:** Jazyk je možné explicitne definovať, použiť automatické rozpoznávanie alebo povoliť viacjazyčný prepis pre nahrávky obsahujúce reč v rôznych jazykoch.
- **Diarizácia rečníkov:** Služba dokáže v reálnom čase identifikovať a rozlíšiť jednotlivých rečníkov.
- **Zoznamy fráz:** Umožňujú poskytnúť modelu zoznam špecifických slov alebo fráz, čím sa zvyšuje presnosť rozpoznávania odbornej terminológie a vlastných mien.
- **Automatické formátovanie textu:** Služba automaticky aplikuje interpunkciu, normalizáciu textu (napr. prevod „štyri"na „4") a odstraňovanie výplňových slov z prepisu.

[3] Gladia SDK Documentation: https://docs.gladia.io/chapters/integrations/sdk

[4] Dokumentácia Azure AI Speech: https://learn.microsoft.com/en-us/azure/ai-services/speech-service/

Používanie služby

Azure AI Speech poskytuje rôzne nástroje ako Speech SDK, Speech CLI alebo REST API pre prístup k službe. Každý z týchto nástrojov je vhodný na iný úkon. Speech SDK je dostupné pre viacero programovacích jazykov vrátane JavaScriptu a TypeScriptu, čo umožňuje jeho integráciu do webových aplikácií. V rámci tejto práce je najrelevantnejšie Speech SDK.

Formát odpovede

Výsledky prichádzajú vo formáte JSON. Každý úsek prepisu obsahuje rozpoznaný text, časové ohraničenie a mieru istoty rozpoznania. Pri nastavení podrobného výstupného formátu sú dostupné aj časové značky na úrovni jednotlivých slov.

5.3 Web Speech API

Web Speech API[5] je rozhranie, ktoré webovým prehliadačom umožňuje rozpoznávanie reči (*SpeechRecognition*) a syntézu reči (*SpeechSynthesis*).

Funkcie a schopnosti

V porovnaní s ostatnými skúmanými službami ponúka API obmedzenú sadu funkcií:

- **Detekcia jazyka:** Jazyk je možné nastaviť explicitne pomocou parametra (`lang`), napríklad hodnotami `en-US` alebo `sk-SK`. Automatická detekcia jazyka nie je podporovaná.
- **Priebežné výsledky:** API podporuje tzv. *interim results*, teda nefinálne prepisy, ktoré sa priebežne aktualizujú pred vydaním finálneho výsledku.
- **Alternatívne prepisy:** Pre každý úsek reči možno získať viacero variantov prepisu zoradených podľa miery istoty.

Používanie služby

Použitie Web Speech API nevyžaduje registráciu ani API kľúč. Rozhranie je dostupné priamo v JavaScripte.

Rozpoznávanie reči štandardne prebieha na vzdialenom serveri prehliadača, nie lokálne v prehliadači. Novšie verzie prehliadačov postavených na jadre Chromium (Google Chrome, Microsoft Edge, Opera) navyše podporujú rozpoznávanie priamo na zariadení po jednorazovom stiahnutí jazykového balíčka, čo umožňuje fungovanie aj bez pripojenia k internetu.

Výraznou nevýhodou je obmedzená podpora naprieč prehliadačmi. Plnú podporu ponúkajú len prehliadače Google Chrome a Microsoft Edge. V ostatných prehliadačoch postavených na jadre Chromium alebo prehliadači Safari je podpora čiastočná, zatiaľ čo Mozilla Firefox toto API nepodporuje vôbec.

Formát odpovede

Výsledky rozpoznávania prichádzajú prostredníctvom udalostí. Každá udalosť obsahuje rozpoznaný text, mieru istoty rozpoznania a atribút rozlišujúci priebežné a finálne výsledky.

[5] Web Speech API Specification: https://wicg.github.io/speech-api/

5.4 Porovnanie

Ponuka rozhraní na prepis reči je rozsiahla. Okrem analyzovaných služieb existujú ďalší poskytovatelia ako Deepgram, AssemblyAI alebo Speechmatics. Napriek rôznym implementačným detailom zdieľajú cloudové STT API rovnaký základný princíp: zvuk je prenášaný cez WebSocket spojenie, výsledky prichádzajú vo formáte JSON a väčšina poskytovateľov ponúka priebežné aj finálne výsledky s časovými značkami na úrovni slov. Zásadné rozdiely teda nespočívajú v dizajne rozhrania, ale v presnosti modelov, latencii, pokrytí jazykov a cenovej politike.

Funkcia	**Gladia**	**Azure AI**	**Web Speech**	**Deepgram**	**AssemblyAI**	**Speechmatics**
Priebežné výsledky	áno	áno	áno	áno	nie	áno
Časové značky slov	áno	áno	nie	áno	áno	áno
Diarizácia rečníkov	lim*	áno	nie	áno	áno	áno
Viacjazyčný prepis	áno	áno	nie	áno	lim‡	lim§
Vlastná slovná zásoba	áno	áno	nie	áno	áno	áno
Preklad prepisu	áno	áno	nie	nie	nie	áno
Offline podpora	nie	nie	lim**	nie	nie	nie

Tabuľka 5.1: Porovnanie funkcionalít vybraných API služieb pre real-time prepis reči.

*Gladia umožňuje rozlíšenie rečníkov pri použití viackanálového WebSocket spojenia, nie však automatickú diarizáciu z jedného kanála.
**Novšie verzie prehliadačov postavených na jadre Chromium podporujú lokálne rozpoznávanie po jednorazovom stiahnutí jazykového balíčka.
‡AssemblyAI podporuje 6 jazykov v streaming režime (angličtina, španielčina, francúzština, nemčina, taliančina, portugalčina); funkcia je v štádiu beta.
§Speechmatics podporuje v real-time režime približne 10 jazykov; automatická detekcia jazyka je dostupná iba v dávkovom spracovaní.

Presnosť a výkon

Štandardnou metrikou pre hodnotenie presnosti automatického prepisu reči je miera chybovosti slov (WER)[6]. Pri porovnávaní WER naprieč poskytovateľmi je dôležité brať do úvahy podmienky merania: hodnoty namerané na čistom laboratórnom zvuku môžu byť výrazne nižšie ako hodnoty na reálnom zvuku obsahujúcom šum, prízvuky alebo odbornú terminológiu [1].

Nasledujúca tabuľka (tabuľka 5.2) uvádza orientačné výkonnostné parametre vybraných STT API. Hodnoty WER pochádzajú z nezávislého porovnania Artificial Analysis [1], ktoré hodnotí modely na troch datasetoch: proprietárnom konverzačnom korpuse[7] (50 %), prepísoch z európskeho parlamentu VoxPopuli (25 %) a prepísoch hovorov Earnings22 (25 %). Výslednú metriku autori označujú ako AA-WER v2.0. Keďže ide o dávkové vyhodnotenie, hodnoty nemusia priamo zodpovedať presnosti v real-time streamovanom režime. Latencia je uvádzaná podľa dokumentácie jednotlivých poskytovateľov.

[6]Word Error Rate – podiel chybne rozpoznaných slov k celkovému počtu slov v referenčnom prepise. Nižšia hodnota znamená vyššiu presnosť.
[7]Korpus je štruktúrovaná zbierka textov alebo nahrávok používaná na tréning alebo hodnotenie modelov. Konverzačný korpus obsahuje prirodzenú hovorovú reč.

Poskytovateľ / model	AA-WER	Latencia
AssemblyAI Universal-3 Pro	3,2 %	300–600 ms
Gladia Solaria-1	4,2 %	∼270 ms
Speechmatics Enhanced	4,3 %	<200 ms
Deepgram Nova-3	5,4 %	<300 ms
Azure AI Speech	–*	<300 ms
Web Speech API	–*	závisí od servera

Tabuľka 5.2: Orientačné výkonnostné parametre vybraných STT API. AA-WER podľa [1]; latencia podľa dokumentácie poskytovateľov.

*Tieto služby nie sú zahrnuté v benchmarku Artificial Analysis.

Napriek rozdielom vo výsledkoch AA-WER možno pozorovať, že väčšina cloudových STT API dosahuje pri real-time prepise latenciu pod 300 ms. Hlavnou diferenciáciou medzi poskytovateľmi zostáva presnosť na cieľovom jazyku a dostupnosť doplnkových funkcií. Cloudové služby ponúkajú bohatšiu funkčnosť vrátane časových značiek slov a diarizácie, no vyžadujú API kľúč a sieťové pripojenie. Web Speech API je jedinou analyzovanou možnosťou bez nutnosti registrácie, avšak za cenu výrazne obmedzenej sady funkcií a podpory prehliadačov.

5.5 Záver

Z analyzovaných služieb sa Web Speech API javí ako nevhodné riešenie pre potreby tejto práce. Hlavnými dôvodmi sú obmedzená podpora naprieč prehliadačmi a absencia časových značiek na úrovni jednotlivých slov, ktoré sú nevyhnutné pre synchronizáciu prepisu s audio záznamom.

Cloudové služby tieto nedostatky nemajú: poskytujú časové značky slov, diarizáciu rečníkov aj pokročilé funkcie ako vlastnú slovnú zásobu či viacjazyčný prepis. Spomedzi analyzovaných možností sa ako najvhodnejšia javí služba Gladia, ktorá kombinuje nízku latenciu, bohatú funkčnú sadu a dostupnosť oficiálneho JavaScript SDK uľahčujúceho integráciu do webových aplikácií.

Kapitola 6

Frontendové knižnice a frameworky

Moderné webové aplikácie kladú na frontendové technológie vysoké nároky: UI musí reagovať na rýchlo sa meniace dáta, spravovať komplexný stav a zároveň zostať výkonné. Pri editore prepisu reči v reálnom čase sú tieto požiadavky obzvlášť výrazné – priebežne prichádzajúce segmenty prepisu, prehrávač audia synchronizovaný s textom a úpravy zo strany používateľa sa navzájom ovplyvňujú a musia byť konzistentne udržiavané.

Táto kapitola poskytuje prehľad troch najrozšírenejších frontendových technológií – React, Angular a Vue.js – so zameraním na ich komponentový model, prístupy k správe stavu a dostupnosť knižníc tretích strán.

6.1 React

React[1] je JavaScriptová knižnica pre tvorbu UI vyvíjaná firmou Meta. Aktuálne patrí medzi najpoužívanejšie a najpopulárnejšie frontendové technológie.

Ako funguje React?

Aplikácie v Reacte sa skladajú z *komponentov* – znovu použiteľných a vnorenia schopných častí UI. Každá aplikácia musí obsahovať aspoň jeden koreňový komponent, od ktorého sa odvíja celá hierarchia.

Komponent je JavaScriptová funkcia obohatená o syntax JSX[2], ktorá umožňuje písať značky podobné HTML priamo v JavaScriptovom kóde. Takýto komponent môže predstavovať jednoduchý UI prvok (ako tlačidlo) až celú komplexnú stránku.

React organizuje UI pomocou stromovej štruktúry založenej na vzťahoch rodič–potomok, kde každý vnorený komponent je potomkom svojho nadradeného komponentu. Táto hierarchia, označovaná ako vykresľovací strom (*render tree*), je dôležitá pre správu stavov a optimalizáciu výkonu.

Pri každej zmene stavu React nevykonáva zmeny priamo v DOM prehliadača, ale pracuje s *virtuálnym DOM* – ľahkou in-memory kópiou skutočného DOM. React porovná novú verziu virtuálneho DOM so starou (tento proces sa nazýva *reconciliation*) a do skutočného DOM premietne iba minimálnu sadu zmien.

[1] Domovská stránka knižnice React: https://react.dev/learn

[2] JavaScript XML, rozšírenie pre syntax jazyka JavaScript, https://react.dev/learn/writing-markup-with-jsx

Stavový manažment

Niektoré časti UI sa menia v reakcii na užívateľský vstup alebo externé udalosti. Dáta, ktoré týmto zmenám podliehajú, sa nazývajú *stav (state)*. React poskytuje hook[3] `useState()` na deklaráciu stavových premenných a ich aktualizáciu. Pri zmene stavu React automaticky prekreslí všetky komponenty, ktoré na ňom závisia.

Okrem vizuálnych zmien UI často potrebujeme vykonávať aj vedľajšie operácie – *efekty (effects)*. Na takéto účely slúži hook `useEffect()`, ktorý umožňuje spúšťať kód ako odozvu na zmeny stavu alebo vykresľovanie komponentu, ako napríklad načítanie dát z API.

V prípade, že viaceré komponenty potrebujú prístup k rovnakým dátam, React ponúka funkciu *context* (kontext). Kontext umožňuje predávať dáta cez celý strom komponentov bez nutnosti explicitného posielania *props* (parametrov komponentu) na každú úroveň.

Výpis 6.1 ukazuje komponent s lokálnym stavom a jednoduchým JSX výstupom:

```
import { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>
        Increment
      </button>
    </div>
  );
}
```

Výpis 6.1: React komponent so stavom

React má vďaka svojej popularite najrozsiahlejší ekosystém knižníc tretích strán. Pre oblasť rich-text editingu sú dostupné knižnice ako Slate.js, Lexical (od Mety) alebo TipTap, ktoré sú primárne navrhnuté pre React alebo ho ako platformu plne podporujú.

6.2 Angular

Angular[4] je webový framework spravovaný firmou Google postavený na TypeScripte a HTML. Na rozdiel od knižníc ako React, Angular poskytuje úplné riešenie pre vývoj webových aplikácií vrátane nástrojov pre routing, správu stavu, formuláre, testovanie a deployment.

Ako funguje Angular?

Stavebnými blokmi Angularu sú znovu použiteľné komponenty organizované do hierarchickej štruktúry. Každý komponent pozostáva z troch častí: TypeScript triedy obsahujúcej logiku a vlastnosti, HTML šablóny definujúcej štruktúru a vzhľad, a CSS štýlov.

[3] Hook je špeciálna funkcia v Reacte, ktorá umožňuje komponentom manipulovať so stavom a pristupovať k ďalším funkciám Reactu.

[4] Domovská stránka frameworku Angular: https://angular.dev/overview

Angular šablóny rozširujú štandardné HTML o vlastnú syntaxu väzieb. Pomocou *property bindingu* (`[property]="value"`) sa hodnoty z TypeScript triedy premietajú do atribútov HTML elementov. *Event binding* (`(event)="handler()"`) naopak zachytáva udalosti z DOM a volá metódy komponentu. Pre obojsmernú väzbu, typickú pri formulárových prvkoch, Angular kombinuje oba prístupy do syntaxe `[(ngModel)]`. Štrukturálne direktívy `@if` a `@for` umožňujú podmienené vykresľovanie a iteráciu priamo v šablóne.

Kľúčovým konceptom Angularu je *dependency injection* (DI) – mechanizmus, ktorý umožňuje deklarovať závislosti komponentu (napríklad služby pre HTTP komunikáciu) a Angular ich automaticky poskytne pri jeho vytváraní. Vďaka tomu je možné závislosti ľahko vymeniť, čo uľahčuje testovanie.

Stavový manažment

Pre správu stavu sa v Angulari používajú *signály (signals)*. Signál je obal hodnoty, ktorý pri jej zmene upozorní všetkých odberateľov – komponenty a výpočty, ktoré danú hodnotu čítajú. Signál sa vytvára pomocou funkcie `signal()`, číta sa zavolaním ako funkcia (napríklad `count()`) a aktualizuje pomocou metód `set()` alebo `update()`.

Pri zmene hodnoty signálu Angular nevykresľuje celý komponent znova, ale iba tú časť šablóny, ktorá je na danom signále závislá. Tento prístup, označovaný ako *fine-grained reactivity*, prináša výkonnostné výhody oproti staršiemu mechanizmu detekcie zmien, ktorý prechádzal celý strom komponentov.

Výpis 6.2 ukazuje Angular komponent využívajúci signál a template binding:

```
@Component({
  selector: 'app-counter',
  template: `
    <p>Count: {{ count() }}</p>
    <button (click)="increment()">Increment</button>
  `
})
export class CounterComponent {
  count = signal(0);
  increment() { this.count.update(v => v + 1); }
}
```

Výpis 6.2: Angular komponent so signálom

6.3 Vue.js

Vue.js[5] je JavaScriptový framework určený na tvorbu užívateľských rozhraní. Vychádza zo štandardných webových technológií HTML, CSS a JavaScript a poskytuje deklaratívny, komponentovo orientovaný programovací model.

Ako funguje Vue.js?

Podobne ako React, aj Vue aplikácie sa skladajú z komponentov. Vue komponenty kombinujú šablónu, logiku a štýly do jedného súboru (*Single File Component*), čo zvyšuje prehľadnosť kódu. Šablóna používa HTML-like syntax rozšírenú o direktívy ako `v-if` pre

[5]Domovská stránka frameworku Vue.js: https://vuejs.org/guide/introduction.html

podmienené vykresľovanie a `v-for` pre iteráciu, a podporuje väzby medzi dátami a atribútmi elementov podobne ako Angular.

Rovnako ako React, aj Vue využíva virtuálny DOM. Pri zmene stavu Vue vypočíta diff medzi starou a novou virtuálnou reprezentáciou a do skutočného DOM premietne iba potrebné zmeny.

Stavový manažment

Reaktivita vo Vue je zabezpečená systémom sledovania závislostí. Vue pri vykresľovaní automaticky zaznamenáva, ktoré reaktívne hodnoty komponent číta, a pri ich zmene prekreslí iba dotknuté časti UI. Reaktívne hodnoty sa deklarujú pomocou funkcie `ref()` pre primitívne typy, alebo `reactive()` pre objekty. Pre zdieľanie stavu naprieč komponentmi Vue ponúka centralizovaný store prostredníctvom knižnice Pinia.

Výpis 6.3 ukazuje Vue Single File Component s reaktívnym stavom:

```
<script setup>
import { ref } from 'vue';
const count = ref(0);
</script>

<template>
  <p>Count: {{ count }}</p>
  <button @click="count++">Increment</button>
</template>
```

Výpis 6.3: Vue.js Single File Component so stavom

6.4 Knižnice UI komponentov

Doteraz sa kapitola zaoberala nástrojmi pre tvorbu a správu logiky UI. Samotné frameworky však neposkytujú hotové vizuálne prvky – tlačidlá, modálne okná a podobne. Na tento účel slúžia *knižnice UI komponentov*, ktoré ponúkajú sadu predpripravených komponentov. Ich použitie výrazne zrýchľuje vývoj a zaručuje konzistentný vzhľad rozhrania bez nutnosti budovať vlastné HTML prvky od nuly.

Každý z vyššie popísaných frameworkov má k dispozícii množstvo takýchto knižníc. Nasledujúci prehľad predstavuje najrozšírenejšie z nich.

Knižnice pre React

Material UI (MUI)[6] je jedna z najpopulárnejších knižníc pre React. Implementuje dizajnový systém Material Design od spoločnosti Google a ponúka rozsiahlu sadu komponentov od základných prvkov (tlačidlá, vstupy, tabuľky) až po komplexné štruktúry. Knižnica podporuje definíciu vlastných globálnych štýlov a štýlovanie priamo cez *prop* `sx`, ktorý prijíma CSS vlastnosti ako objekt. Je vhodná pre rozsiahle aplikácie.

Chakra UI[7] je knižnica zameraná na jednoduchosť a prístupnosť. Komponenty sú navrhnuté ako kompozičné stavebné bloky. Štýlovanie prebieha priamo cez *props* komponentov, čo znižuje množstvo pomocných CSS tried. Chakra UI sa hodí pre projekty, kde je dôraz kladený na čistý kód a rýchly vývoj.

[6] https://mui.com
[7] https://chakra-ui.com

Knižnice pre Angular

Angular Material[8] je oficiálna knižnica komponentov pre Angular, vyvíjaná priamo tímom Google. Rovnako ako MUI pre React, implementuje Material Design a je úzko integrovaná s ekosystémom Angularu vrátane podpory pre Angular Forms a CDK (*Component Dev Kit*).

PrimeNG[9] je rozsiahla knižnica komponentov pre Angular s viac ako 90 preddefinovanými prvkami. Na rozdiel od Angular Material nie je viazaná na konkrétny dizajnový systém, čo umožňuje väčšiu vizuálnu flexibilitu. Ponúka aj dátové komponenty ako tabuľky s radením a stránkovaním, stromové zobrazenia alebo grafové komponenty.

Knižnice pre Vue.js

Vuetify[10] je knižnica pre Vue.js inšpirovaná dizajnovým systémom Material Design. Ponúka komplexnú sadu komponentov a rozsiahle možnosti tematizácie. Je jednou z najdlhšie existujúcich a najpoužívanejších Vue knižníc.

PrimeVue[11] je Vue varianta knižnice PrimeNG. Podobne ako jej Angular protistrana, ponúka široké spektrum komponentov bez väzby na konkrétny dizajnový systém a kladie dôraz na flexibilitu štýlovania.

6.5 Knižnice pre rich-text editory

Bežný HTML element `<textarea>` neumožňuje pripojiť k jednotlivým slovám ľubovoľné metadáta ako napríklad časové značky. Na tento účel sú potrebné špecializované knižnice, ktoré reprezentujú obsah dokumentu ako štruktúrovaný dátový model, v ktorom každý uzol môže niesť vlastné atribúty.

ProseMirror[12] je nízkoúrovňový framework pre rich-text editory. Obsah dokumentu je reprezentovaný ako strom uzlov so striktne definovanou schémou. ProseMirror neposkytuje hotový editor, ale sadu primitív – správu stavu, aplikáciu transakcií a vykresľovanie – na ktorých je možné postaviť vlastné riešenie. Je frameworkovo nezávislý, čo ho robí flexibilným, no zároveň náročným na implementáciu.

TipTap[13] je editor postavený nad ProseMirrorom, ktorý poskytuje vyššiu úroveň abstrakcie. Rozširuje ho o systém rozšírení (*extensions*), vďaka ktorým je možné pridávať funkcie bez priamej práce s ProseMirror API. Podporuje React aj Vue.js, pre Angular je podpora obmedzená.

Slate.js[14] je knižnica pre React navrhnutá okolo vlastného dátového modelu – dokumentu ako stromu uzlov, kde každý uzol môže niesť ľubovoľné vlastné dáta. Tento prístup je vhodný najmä pre editory, kde je potrebné k textu pripojiť štruktúrované metadáta, napríklad časové ohraničenie jednotlivých slov pri prepise reči. Slate.js nepredpisuje schému dokumentu, čo dáva vývojárovi plnú kontrolu nad dátovým modelom.

[8] https://material.angular.io
[9] https://primeng.org
[10] https://vuetifyjs.com
[11] https://primevue.org
[12] https://prosemirror.net
[13] https://tiptap.dev
[14] https://docs.slatejs.org

Lexical[15] je knižnica od spoločnosti Meta zameraná na výkon a rozšíriteľnosť. Podobne ako Slate.js je primárne určená pre React. Jej architektúra oddeľuje stav editora od jeho zobrazenia, čo uľahčuje implementáciu kolaboratívneho editovania a historických operácií.

6.6 Knižnice pre vizualizáciu zvukových vĺn

Pre editor prepisu reči je vizualizácia zvuku dôležitou súčasťou rozhrania – umožňuje používateľovi zorientovať sa v nahrávke, identifikovať rečníkov a presne navigovať ku konkrétnym miestam v zázname.

Wavesurfer.js[16] je najrozšírenejšia knižnica pre interaktívnu vizualizáciu zvukových vĺn vo webovom prostredí. Vykresľuje priebeh signálu do HTML elementu `<canvas>` a poskytuje API na prehrávanie, vyhľadávanie a zoom. Funkcionalitu je možné rozšíriť pomocou pluginov – napríklad *Regions* pre označenie úsekov, *Timeline* pre zobrazenie časovej osi alebo *Minimap* pre prehľadovú mapu dlhých nahrávok. Knižnica je frameworkovo nezávislá; pre React je dostupný oficiálny wrapper `@wavesurfer/react`, ktorý sprístupňuje konfiguračné možnosti knižnice ako React props.

Peaks.js[17] je knižnica vyvinutá spoločnosťou BBC pre prácu s dlhými zvukovými nahrávkami. Na rozdiel od Wavesurfer.js je orientovaná na segmentovú anotáciu – umožňuje definovať úseky (*segments*) a body (*points*) v časovej osi a pracovať s nimi programaticky. Pre načítanie dlhých nahrávok využíva predgenerované dáta priebehov (*waveform data*) namiesto dekódovania celého audio súboru v prehliadači.

6.7 Záver

Všetky tri frameworky zdieľajú komponentový model a podporu TypeScriptu. React je minimalistická knižnica ponechávajúca výber ostatných nástrojov na vývojárovi, Angular uceleným frameworkom s pevnou štruktúrou a Vue zaujíma strednú pozíciu. Pre oblasť rich-text editingu je výber nerovnomerný: Slate.js a Lexical sú navrhnuté výhradne pre React, TipTap podporuje React aj Vue, pre Angular sú možnosti výrazne obmedzené.

Pre potreby tejto práce sa ako vhodné technológie javia React spolu s knižnicami Slate.js a Wavesurfer.js. Angular v tejto kombinácii nie je vhodný z dôvodu slabej podpory rich-text editovania, Vue je použiteľný, no ekosystém pre túto oblasť je v porovnaní s Reactom chudobnejší.

[15] https://lexical.dev
[16] https://wavesurfer.xyz
[17] https://github.com/bbc/peaks.js

Kapitola 7

Implementácia

Táto kapitola sa venuje výberu implementačných technológií a popisuje spôsob implementácie kľúčových častí aplikácie.

7.1 Voľba implementačných technológií

Keďže aplikácia nemá vlastný backend a všetka logika beží v prehliadači, bol pri výbere technológií kladený dôraz na ekosystém, dostupnosť knižníc a podporu prehliadačov.

Na implementáciu rich-text editora bol zvolený framework *Slate.js*, ktorý umožňuje definovať vlastnú štruktúru dokumentu a prispôsobiť správanie editora. Táto flexibilita je nevyhnutná pre prepojenie textu s časovými značkami audia, čo štandardné editory nepodporujú. Zároveň umožňuje definovať vlastné UI komponenty, čo je dôležité pre implementáciu vizuálneho odlíšenia rečníkov a ďalších interaktívnych prvkov.

Voľba Slate.js priamo podmienila výber knižnice *React*: ako bolo uvedené v kapitole 6, knižnica Slate.js je navrhnutá pre React. Ako stavebný rámec bol zvolený *Next.js*. Hoci ide primárne o framework s podporou vykresľovania na strane servera, v tomto projekte je využitý výhradne pre vývojové prostredie a statický export.

Na implementáciu užívateľského rozhrania bola použitá komponentová knižnica *Chakra UI*. Oproti alternatíve MUI, ktorá je vhodná pre rozsiahle podnikové aplikácie, sa Chakra UI vyznačuje jednoduchosťou a dôrazom na rýchly vývoj, čo lepšie zodpovedá rozsahu tohto projektu.

Na vizualizáciu a prehrávanie audiozáznamu bola použitá knižnica *WaveSurfer.js*. Alternatíva Peaks.js analyzovaná v kapitole 6 je orientovaná na segmentovú anotáciu dlhých nahrávok s predgenerovanými priebehmi zvuku – funkcie, ktoré aplikácia nevyžaduje. WaveSurfer.js je jednoduchšia na integráciu a ponúka natívnu podporu pre React.

Na nahrávanie audia z mikrofónu bola zvolená knižnica *RecordRTC*. Natívne rozhranie prehliadača *MediaRecorder API* nepodporuje formát WAV, produkuje výlučne kontajnery WebM alebo Ogg, ktorý Gladia vyžaduje pre streamovanie v reálnom čase. RecordRTC tento nedostatok rieši a zároveň umožňuje konfigurovateľnú vzorkovaciu frekvenciu a postupné odosielanie audio fragmentov počas nahrávania.

Na prepis reči bol integrovaný *Gladia SDK*. Z analýzy v kapitole 5 vyplýva, že Gladia dosahuje WER 4,2 % a latenciu približne 270 ms, čo sú konkurencieschopné hodnoty. Zároveň Gladia SDK pokrýva oba potrebné režimy, real-time streaming aj dávkový prepis. Integrácia je sústredená do dedikovaného hooku a pomocných súborov v adresári. *TranscriptionContext* tvorí abstrakčnú vrstvu, vďaka ktorej zvyšok aplikácie nezávisí od konkrétnej

STT služby. Náhrada za iného poskytovateľa by si vyžiadala implementáciu nového hooku s rovnakým rozhraním a aktualizáciu jazykových kódov, bez zásahu do UI vrstvy alebo ostatných kontextov.

Na ukladanie dát sú využité dve prehliadačové úložiská. Metadáta dokumentov a transkripčné dáta sú ukladané v *IndexedDB*, čo je klientská NoSQL databáza s asynchrónnym prístupom vhodná na ukladanie štruktúrovaných objektov. Audio súbory sú ukladané v *Origin Private File System (OPFS)*, súborovom systéme dostupnom len v rámci konkrétnej webovej aplikácie, ktorý umožňuje efektívnu prácu s binárnymi dátami bez nutnosti sieťového prenosu.

7.2 Kontexty aplikácie

Správa stavu aplikácie je rozdelená do siedmich samostatných kontextov využívajúcich *React Context API*. Každý kontext zodpovedá za presne vymedzenú časť aplikačnej logiky, čo zabraňuje zbytočným prekresleniam komponentov a uľahčuje orientáciu v kóde.

- **DocumentsContext** – spravuje zoznam všetkých dokumentov a poskytuje operácie na ich načítanie z IndexedDB.
- **EditorContext** – uchováva stav aktuálne otvoreného dokumentu vrátane obsahu Slate editora a sprístupňuje funkcie na jeho správu.
- **SpeakersContext** – udržiava odvodený zoznam rečníkov odčítaný priamo z obsahu Slate dokumentu.
- **BookmarksContext** – udržiava odvodený zoznam záložiek odčítaný priamo z obsahu Slate dokumentu.
- **PlaybackContext** – uchováva stav prehrávania a aktuálny čas prehrávania, ktorý slúži na synchronizáciu zvýraznenia textu.
- **AudioEngineContext** – poskytuje inštanciu *WaveSurfer.js* a funkcie na vyhľadávanie a spúšťanie prehrávania z ostatných komponentov.
- **TranscriptionContext** – riadi proces transkripcie, spúšťanie a zastavovanie prepisu z mikrofónu aj zo súboru.

Každý kontext je zapuzdrený vo vlastnom *provideri*, ktorý obaľuje stromovú štruktúru komponentov.

7.3 Rich-text editor

Rich-text editor je implementovaný pomocou frameworku *Slate.js*. Slate reprezentuje obsah dokumentu ako strom uzlov, kde každý uzol môže byť blokový element, inline element alebo textový uzol. Táto štruktúra umožňuje priradiť ku každému slovu v prepise jeho časovú pozíciu v audiozázname.

Dokument sa skladá z blokového elementu typu *transcript* a inline elementu typu *bookmark*. Element *transcript* reprezentuje jednu výpoveď rečníka – obsahuje identifikátor rečníka, jeho farbu a ako potomkov textové uzly s časovými značkami. Element *bookmark* je inline element vložený priamo do toku textu; obsahuje v tele používateľom vybraný text.

Na textových uzloch sú definované tieto značky: *bold*, *italic*, *underline* a *strikethrough* pre formátovanie. Okrem formátovacích značiek každý textový uzol uchováva aj metadáta transkripcie: časovú pozíciu začiatku a konca výpovede a pole slov s ich individuálnymi časovými značkami. Pre dynamické zvýraznenie prepisu počas prehrávania slúži dekorácia *playbackState* s hodnotami *past* a *future*, ktorá sa vypočítava funkciou `decorate` a nie je uložená v dokumente.

Spracovanie výsledkov transkripcie

Gladia API vracia výsledky prepisu ako sekvenciu výpovedí (*utterances*), pričom každá výpoveď nesie identifikátor rečníka, časové ohraničenie celého úseku a zoznam jednotlivých slov s ich časovými značkami.

Každá prichádzajúca výpoveď je rozložená na individuálne textové uzly – po jednom na každé slovo – pričom každý uzol preberá časové značky príslušného slova. Tieto uzly sa pridajú k poslednému *transcript* elementu v dokumente, alebo sa vytvorí nový element, ak posledný element neobsahuje časové metadáta, alebo by jeho dĺžka po pridaní prekročila stanovený limit (600 znakov). V dávkovom režime sa nový element vytvorí aj v prípade, že sa identifikátor rečníka líši od identifikátora posledného elementu. Posledná podmienka zaručuje čitateľnosť blokov a zmysluplnosť tlačidla prehratia sekcie. Celá operácia je vykonaná mimo histórie editora, čím sa zabraňuje neúmyselnému zvráteniu transkripcie krokom späť.

Rečníkom je pri prvom výskyte priradená farba z preddefinovanej palety na základe ich identifikátora.

Validácia pri vkladaní záložiek

Pri vkladaní záložky sú overené viaceré podmienky. Záložku nie je možné vložiť cez viaceré *transcript* elementy, teda cez úseky rôznych rečníkov, alebo ju rozdeliť vložením nového *transcript* elementu do jej tela. Dôvodom je, že záložka je inline element vnorený do konkrétneho elementu a nemôže presahovať hranicu bloku. Rovnako nie je možné záložku vložiť do rozsahu, ktorý už inú záložku obsahuje. Ak niektorá z podmienok nie je splnená, používateľ je o tom informovaný oznamom (*toast*).

7.4 Ukladanie dát

Keďže aplikácia neposiela žiadne dáta na server, všetky informácie sú ukladané lokálne v prehliadači. Na tento účel sú využité dve odlišné úložiská podľa formátu ukladaných dát.

Metadáta dokumentov, transkripčné dáta a stav Slate editora sú ukladané v *IndexedDB*. Zmeny v editore sú do databázy zapisované s oneskorením 3 sekundy po poslednej úprave, čím sa predchádza nadmernému počtu zápisov počas aktívneho písania.

Audio súbory sú ukladané v *OPFS*. Počas nahrávania z mikrofónu sú audio fragmenty vo formáte PCM priebežne zapisované do OPFS súboru. Každý fragment je pred zápisom zbavený WAV hlavičky, ktorá by pri spájaní fragmentov spôsobovala nevalidný súbor. Po skončení nahrávania je výsledný súbor obalený správnou WAV hlavičkou a prevedený na *Blob URL*, ktorú *WaveSurfer.js* použije na načítanie a vizualizáciu zvukovej stopy. Pri prepise zo súboru je nahraný súbor uložený do OPFS priamo.

Keďže obe úložiská sú trvalé, dáta zostávajú zachované aj po zatvorení aplikácie alebo prehliadača. Po opätovnom otvorení sa dokumenty načítajú z IndexedDB a príslušné audio

súbory z OPFS do stavu, v ktorom ich používateľ zanechal. Dáta sa stratia iba v prípade, že používateľ manuálne vymaže údaje prehliadača alebo samotný prehliadač zo zariadenia odstráni.

7.5 Prehrávanie audio záznamu

Na prehrávanie a vizualizáciu audia je použitá knižnica *WaveSurfer.js*. V aplikácii existujú dve inštancie tejto knižnice s odlišným účelom a životným cyklom.

Prvá inštancia zabezpečuje prehrávanie uloženého záznamu. Zobrazuje interaktívny priebeh zvukovej stopy a umožňuje vyhľadávanie kliknutím. Prostredníctvom pluginu *RegionsPlugin* vykresľuje záložky ako farebné vertikálne značky priamo vo vlnovom zobrazení. Táto inštancia vzniká pri otvorení dokumentu, keď je dostupná URL adresa zvukového súboru z OPFS, a zostáva aktívna po celý čas práce s dokumentom. V prípade prepisu zo súboru sa prehrávanie spustí automaticky, aby zvýraznenie textu prebiehalo súbežne s transkripciou.

Druhá inštancia slúži výhradne na vizualizáciu živého vstupného signálu z mikrofónu počas nahrávania. Zobrazuje priebeh zvuku v reálnom čase, ale neumožňuje vyhľadávanie ani prehrávanie. Táto inštancia je vytvorená pri spustení nahrávania a zničená po jeho ukončení; v ostatných stavoch aplikácie nie je viditeľná.

Aktuálna pozícia prehrávania je z inštancie *WaveSurfer.js* priebežne prenášaná do *PlaybackContext*. Aby nedochádzalo k zbytočnému prekresleniu, aktualizácia prebehne najviac každých 80 milisekúnd a len vtedy, ak sa pozícia zmenila o viac ako 80 milisekúnd.

7.6 Synchronizácia audia a textu

Synchronizácia medzi audiom a textom prebieha v dvoch smeroch.

Pri prehrávaní audia zvýrazňuje editor práve prehrávaný text. Každý textový uzol v Slate dokumente nesie časovú značku svojho začiatku pochádzajúcu z Gladia API. Mechanizmus *decorate* v Slate priebežne porovnáva aktuálny čas prehrávania s touto značkou a priradí uzlu stav *past* alebo *future*. Uzly pred aktuálnou pozíciou sú zobrazené čiernou farbou, uzly za ňou sú zošednuté.

V opačnom smere, keď používateľ premiestni kurzor do textu, komponent *CursorTimeSync* sleduje zmenu pozície kurzora a extrahuje časovú značku začiatku príslušného textového uzla. Prehrávač sa následne presunie na zodpovedajúcu pozíciu v audiozázname. Tento posun nastane iba vtedy, keď prehrávanie neprebieha, aby nedochádzalo k prerušeniu počas aktívneho počúvania.

Keďže používateľ môže text editovať, niektoré pôvodné slová s časovými značkami môžu chýbať alebo sa nezhodovať s aktuálnym obsahom textového uzla. V takom prípade mechanizmus zvýraznenia interpoluje pozíciu slova na základe jeho poradového čísla v rámci segmentu a celkového trvania výpovede. Pri presune kurzora do editovaného úseku sa ako náhradná pozícia použije časová značka konca daného textového uzla.

Treba podotknúť, že aplikácia nie je navrhnutá na tvorbu titulkov ani iných výstupov vyžadujúcich presné časovanie na úrovni každého slova. Jej primárnym účelom je uľahčiť orientáciu v prepise počas editácie, preto mierne nepresnosti v synchronizácii po editovaní textu nepredstavujú zásadné obmedzenie.

7.7 Dávkový prepis

Pôvodná verzia aplikácie podporovala pri spracovaní súboru výlučne real-time režim: Gladia SDK priebežne streamoval čiastkové výsledky prepisu, ktoré sa postupne vkladali do editora. Do aplikácie bola doplnená možnosť dávkového spracovania (*batch transcription*): aplikácia odovzdá celý súbor na prepis a počká na kompletný výsledok, ktorý vloží do editora naraz. Výhodou tohto režimu je automatické rozpoznávanie rečníkov, ktoré real-time režim nepodporuje. Výber medzi oboma režimami sa vykonáva pri nahrávaní súboru.

7.8 Export prepisu

Aplikácia umožňuje exportovať prepis do štyroch formátov. Všetky exporty pracujú priamo nad obsahom Slate dokumentu. Iterujú strom uzlov a zostavujú výstup podľa cieľového formátu.

Pri exporte do formátov **TXT** a **Markdown** sa pre každý *transcript* element vygeneruje hlavička v tvare `MenoRečníka [HH:MM:SS]` nasledovaná samotným textom výpovede. V prípade Markdown exportu sú navyše zachované formátovacie značky – tučné písmo sa premení na `**text**`, kurzíva na `*text*` a prečiarknutý text na `~~text~~`. Výsledný reťazec je uložený ako súbor `.txt`, respektíve `.md`.

Export do formátu **DOCX** využíva knižnicu *docx*. Každý *transcript* element sa zobrazí ako samostatný odsek s menom rečníka vo farbe priradenej danému rečníkovi a časovou značkou začiatku výpovede. Formátovacie atribúty textových uzlov (tučné, kurzíva, prečiarknuté) sú prevedené na zodpovedajúce vlastnosti objektu *TextRun*.

Export **audio záznamu** vo formáte WAV načíta surové PCM dáta uložené v OPFS, doplní im WAV hlavičku a výsledný súbor je možné stiahnuť.

7.9 Záver

Táto kapitola popísala implementáciu kľúčových častí aplikácie. Základom je knižnica *React* s frameworkom *Next.js*, pričom stav aplikácie je rozdelený do siedmich nezávislých kontextov využívajúcich *React Context API*.

Jadrom editora je *Slate.js*, ktorý umožňuje priradiť ku každému slovu v prepise jeho časovú polohu v audiozázname. Tento dátový model je základom pre obojsmernú synchronizáciu audia a textu – pri prehrávaní editor zvýrazňuje práve prehrávaný text na úrovni jednotlivých slov, a naopak, presun kurzora posúva prehrávač na zodpovedajúcu pozíciu.

Keďže aplikácia nepoužíva vlastný backend, všetky dáta sú ukladané priamo v prehliadači – transkripčné dáta v *IndexedDB* a audio súbory v *OPFS*. Hotový prepis je možné exportovať do formátov TXT, Markdown a DOCX, prípadne stiahnuť pôvodný audio záznam vo formáte WAV. Rozhranie je implementované ako responzívne a použiteľné aj na mobilných zariadeniach.

Kapitola 8

Testovanie použiteľnosti

Táto kapitola popisuje, akými spôsobmi bolo implementované riešenie testované. Keďže prívetivosť a intuitívne užívateľské rozhranie patrili medzi hlavné ciele práce, testovanie spočívalo v overení použiteľnosti aplikácie s cieľovými používateľmi.

8.1 Testovanie použiteľnosti desktopovej verzie

To, čo robí aplikáciu použiteľnou, je absencia frustrácie[5]. Výsledné riešenie by malo vykonať to, čo používateľ očakáva, tak aby pri tom nemusel váhať ani zbytočne rozmýšľať.

Testovanie použiteľnosti je proces, do ktorého sa zapájajú ľudia reprezentujúci cieľovú skupinu, aby sa posúdilo, nakoľko je systém pre nich intuitívny a ľahko použiteľný [5]. Zvolená metóda bola moderovaný test použiteľnosti, pri ktorom účastník nahlas komentuje svoje myšlienky a rozhodnutia počas plnenia úloh.

Výber účastníkov

Prvým krokom bola identifikácia cieľovej skupiny používateľov. Pri výbere bola overená doména kandidátov – napríklad to, čo práve robia a či by mohli aktívne využiť tento nástroj.

Výber pozostával z troch účastníkov, keďže toto množstvo by malo stačiť na odhalenie vážnych nedostatkov. Ďalší účastníci väčšinou odhaľujú tie isté problémy [2].

Priebeh testovania

Testovanie prvého účastníka prebehlo osobne, testovanie druhého a tretieho účastníka prebehlo vzdialene (*remote*) formou videohovoru. Aplikácia bola vo všetkých prípadoch spustená vo webovom prehliadači na počítači účastníka.

Samotný test prebiehal tak, že účastníkovi bola zadaná séria úloh, ktoré mal vykonať. Úlohy boli doplnené o kontext, napríklad stručný scenár opisujúci situáciu, v ktorej by sa skutočný používateľ mohol ocitnúť, aby bol výkon čo najprirodzenejší. Počas plnenia úloh bol účastník pozorovaný a boli mu kladené doplňujúce otázky.

Priebeh testovania bol zaznamenávaný, zachytená bola obrazovka aj zvuk, čo umožnilo spätné prehranie a detailnú analýzu momentov, ktoré sa mohli počas živého pozorovania prehliadnuť. Sledované boli aj neverbálne reakcie účastníka, výraz tváre, zaváhanie či opakované klikanie na nesprávne prvky mohli odhaliť problémy, ktoré účastník sám nezmienil.

Počas testu boli kladené otvorené otázky zamerané na aktuálne myslenie účastníka [2]. Napríklad čo práve očakával, že sa stane, alebo prečo zvolil konkrétnu akciu. Cieľom nebolo hodnotiť účastníka, ale porozumieť jeho mysleniu a odhaliť miesta, kde rozhranie nespĺňalo jeho očakávania.

Postup a zadanie úloh

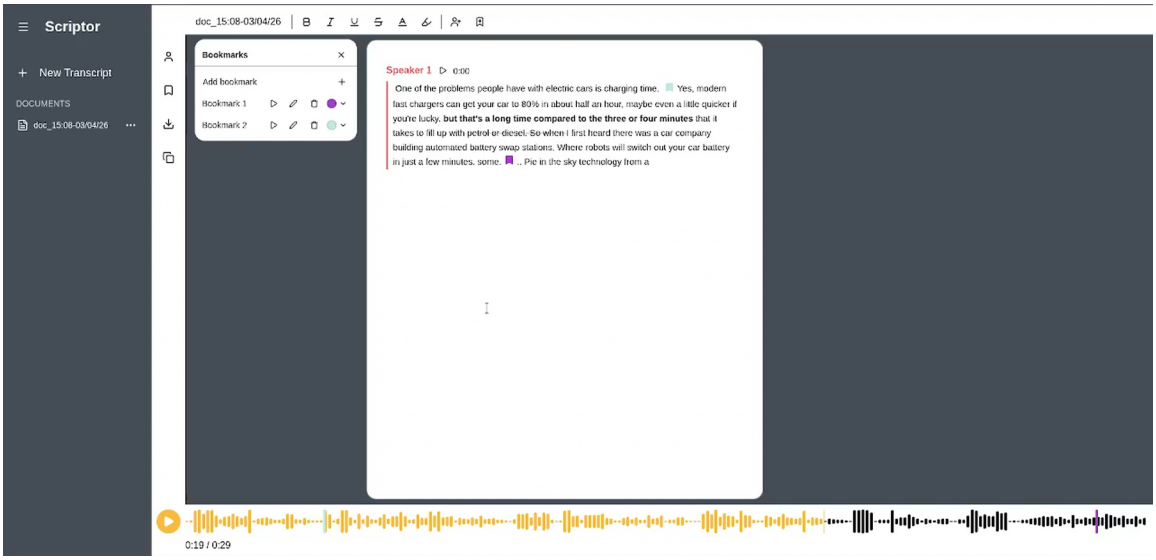
Účastníci boli pred samotným testovaním oboznámení, že zvuk aj obrazovka počítača sú zaznamenávané pre účely testovania. Boli požiadaní, aby nahlas hovorili, čo si myslia a prečo vykonali konkrétnu akciu. Každý účastník dostal okrem úloh aj stručný kontext zohľadňujúci jeho zamestnanie, aby bolo používanie aplikácie čo najprirodzenejšie. Aplikácia im nebola vopred predstavená.

Úlohy boli cielene zostavené tak, aby pokryli hlavné funkcionality aplikácie a overili, nakoľko je ich ovládanie intuitívne a ľahko použiteľné bez predchádzajúceho zaškolenia. Účastníkom boli zadané tieto úlohy:

1. *Pozrite sa na túto webovú stránku, neklikajte na nič. Čo si myslíte, na čo táto stránka slúži?*
2. *Skúste vytvoriť živý prepis zo súboru.*
3. *Narazili ste na zaujímavé miesto v prepise a chcete sa k nemu neskôr vrátiť.*
4. *Skúste vytvoriť nový živý prepis z mikrofónu.*
5. *Zabudli ste niečo povedať, doplňte to do prepisu.*
6. *V nahrávke sa striedalo viacero hovoriacich, rozlíšte ich v prepise.*
7. *Skúste si stiahnuť prepis, na formáte nezáleží..*
8. *Skúste prepnúť na iný dokument.*
9. *Premenujte dokument.*
10. *Vymažte dokument.*

Výsledky

Stav aplikácie v čase testovania zachytáva obrázok 8.1.



Obr. 8.1: Stav aplikácie počas testovania použiteľnosti.

Účastník 1 Prvá účastníčka bola 21-ročná študentka. Počítač používa každodenne a v oblasti digitálnych nástrojov sa cíti technicky zdatná. Úlohy jej boli zasadené do študentského kontextu: niektoré sa týkali záznamu z prednášky a iné sa vzťahovali na stretnutie organizácie ESN.

Na prvý pohľad pochopila, že ide o editor prepisu reči.

Zo začiatku bol problém s nahratím súboru, keďže pole drag&drop nezmenilo formát kurzora na pointer a účastníčke nedošlo, že pole je klikateľné.

Účastníčka pri práci s textom očakávala, že pri označení textu sa v blízkosti kurzora zobrazí plávajúce okno s nástrojmi na formátovanie textu. Poznamenala, že na takúto vlastnosť je zvyknutá z rôznych aplikácií.

Ocenila vzhľad a rozvrhnutie aplikácie, pozitívne hodnotila aj vizuálnu synchronizáciu prepisu s audiom.

Pri pridávaní ďalšieho rečníka očakávala možnosť vybrať si medzi vložením úplne nového rečníka a už existujúceho. Zároveň sa jej páčil spôsob správy rečníkov priamo v editore.

Pri vkladaní záložiek účastníčka intuitívne označila časť textu a tak vložila záložku, čo spôsobilo neočakávané správanie. Účastníčka očakávala, že sa označený text zvýrazní a bude v module záložiek. Koncept záložiek jej prišiel užitočný, ale nevyhovovalo jej aktuálne spracovanie.

Pri ovládaní prehrávania účastníčka postrádala možnosť zmeny rýchlosti prehrávania, skok o 15 sekúnd dopredu a dozadu. Tlačidlo prehrávania a iné ovládacie prvky očakávala vycentrované v strede obrazovky.

Účastníčke chýbali tlačidlá krok späť a krok vpred. Tieto tlačidlá boli síce súčasťou návrhu rozhrania, no v čase testovania ešte neboli implementované.

Tlačidlo „Append transcript“ pre opätovné spustenie živého nahrávania sa jej zdalo nejasné, namiesto neho navrhla označenie „Continue“.

Účastníčka si sťažovala, že pri kliknutí na modul v ľavom paneli celý dokument poskočí doprava, čo narušovalo jej pracovný tok.

Klávesové skratky účastníčka vôbec nepoužila, pri otázke prečo odpovedala, že všeobecne pri pohybe na webe nemá vo zvyku klávesové skratky používať.

Účastníčka si počas testovania neuvedomila, že prepis je možné editovať v reálnom čase. Poznamenala, že by bolo vhodné túto vlastnosť niekde napísať.

Účastník 2 Druhá účastníčka bola 22-ročná študentka a laborantka. Počítač používa každodenne a v oblasti digitálnych nástrojov sa cíti technicky zdatná. Úlohy jej boli zasadené do dvoch kontextov: prvý sa týkal záznamu z laboratórneho merania, kde potrebovala písomný prepis experimentu a chcela si označiť kľúčové momenty, druhý sa vzťahoval na záznam z workshopu, ktorého výstupy chcela zdokumentovať.

Opakovali sa viaceré problémy z testovania prvého účastníka. Ťažkosti s nahratím súboru, správanie záložiek, vkladanie rečníkov a chýbajúci plávajúci panel nástrojov pri výbere textu.

Zároveň sa opakovala pozitívna odozva na vizuálne odlíšenie rečníkov a možnosť ich úpravy, na celkové rozvrhnutie aplikácie a na synchronizáciu prepisu so zvukom. Záložky jej konceptuálne dávali zmysel, no aktuálne správanie nezodpovedalo jej očakávaniam. Ocenila tiež prítomnosť klávesových skratiek.

V zozname dokumentov jej chýbalo zoradenie podľa naposledy otvorených dokumentov, prípadne možnosť zoradiť ich podľa názvu.

Účastníčka si počas testovania neuvedomila, že prepis je možné editovať v reálnom čase počas nahrávania. Povedala, že by jej pomohlo, keby na to bolo pri vytváraní nového dokumentu upozornené.

Rovnako ako prvá účastníčka si sťažovala, že po kliknutí na modul v ľavom paneli celý dokument poskočí doprava, čo pôsobilo rušivo.

Ocenila by tiež, keby tlačidlá pre export a kopírovanie prepisu boli umiestnené v pravom hornom rohu dokumentu, kde ich očakáva na základe skúseností z iných aplikácií.

Účastník 3 Tretia účastníčka bola 46-ročná učiteľka francúzštiny na strednej škole. Počítač používa každodenne, pričom v niektorých oblastiach digitálnych nástrojov sa cíti sebaisto, v iných menej. Úlohy jej boli zasadené do školského kontextu: prvá sa týkala záznamu z ústnej skúšky, ku ktorému chcela vytvoriť písomný prepis a označiť si kľúčové momenty, druhá sa vzťahovala na prednášku o literárnom diele, z ktorej potrebovala textový záznam.

Z názvu aplikácie pochopila, čo je hlavná funkcionalita aplikácie.

Opäť sa opakovali podobné problémy. Problém pri nahrávaní súboru, správanie záložiek, správa rečníkov a nevedomosť, že text je možné editovať počas prebiehajúceho prepisu.

Ocenila vizuálne odlíšenie rečníkov a možnosť ich úpravy priamo v editore, ako aj vizuálnu synchronizáciu prehrávania s prepisom. Pozitívne hodnotila aj možnosť exportu prepisu vo formáte `DOCX`.

Pri spustení transkripcie si účastníčka neuvedomila, že prepis už prebieha, a nebola si istá, či môže začať hovoriť. Chýbal jej viditeľný indikátor stavu nahrávania.

Zmätok jej spôsobilo umiestnenie tlačidiel pre prehrávanie a opätovné spustenie živého prepisu. Očakávala, že budú bližšie pri sebe. Taktiež jej chýbali tlačidlá pre skok dozadu a dopredu v paneli prehrávania.

Taktiež jej chýbali tlačidlá krok späť a krok vpred, ktoré boli súčasťou návrhu, no v čase testovania ešte neboli implementované.

Pri hľadaní možnosti stiahnutia prepisu sa najprv obrátila na bočný panel so súbormi a klikla na menu s tromi bodkami, kde túto možnosť nenašla. Až potom si všimla modul na export.

Zhrnutie

Testovanie odhalilo viacero opakujúcich sa problémov, ako aj problémy špecifické pre jednotlivcov. Tabuľka 8.1 zachytáva najčastejšie zistené nedostatky a to, u ktorých účastníkov sa prejavili.

Tabuľka 8.1: Prehľad identifikovaných problémov použiteľnosti (Ú1–Ú3 = účastník 1–3).

Problém	**Ú1**	**Ú2**	**Ú3**
Záložky – nečakané správanie pri vkladaní	✓	✓	✓
Pole *drag and drop* pôsobí neklikateľne	✓	✓	✓
Editácia v reálnom čase nebola zrejmá	✓	✓	✓
Chýbajúci plávajúci panel nástrojov pri výbere textu	✓	✓	×
Chýbajúce tlačidlá krok späť a vpred	✓	×	✓
Ťažkosti s priraďovaním rečníkov (diarizácia)	✓	✓	×
Nejasné označenie tlačidla „Append transcript“	✓	×	×
Poskočenie dokumentu pri otvorení modulov	✓	✓	×
Nejasný indikátor stavu nahrávania	×	×	✓

Najzávažnejším opakujúcim sa problémom bola funkcionalita záložiek, ktorá nezodpovedala očakávaniam žiadneho z účastníkov, všetci predpokladali vizuálne zvýraznenie textu namiesto časovej značky. Všetci traja účastníci si počas testovania neuvedomili, že editácia prepisu je možná priamo počas nahrávania, čo naznačuje, že túto vlastnosť rozhranie dostatočne nekomunikuje. Rovnako všetci traja narazili na problém s poľom pre *drag and drop*, ktoré nepôsobí klikateľne. Dvaja z troch účastníkov pri označení textu očakávali plávajúci panel nástrojov na formátovanie, na ktorý sú zvyknutí z iných aplikácií. Rovnako dvaja účastníci si sťažovali na poskočenie dokumentu doprava pri otvorení modulu v ľavom paneli, čo narušovalo ich pracovný tok.

Problémy, ktoré sa vyskytli iba u jedného účastníka, súviseli prevažne s osobnými zvyklosťami a očakávaniami z iných nástrojov.

Napriek identifikovaným nedostatkom účastníci zaznamenali aj pozitívne stránky aplikácie. Všetci traja ocenili vizuálnu synchronizáciu polohy prehrávania s textom prepisu a vizuálne odlíšenie rečníkov vrátane možnosti ich priamej úpravy v editore. Dvaja z troch účastníkov kladne hodnotili celkové rozvrhnutie aplikácie.

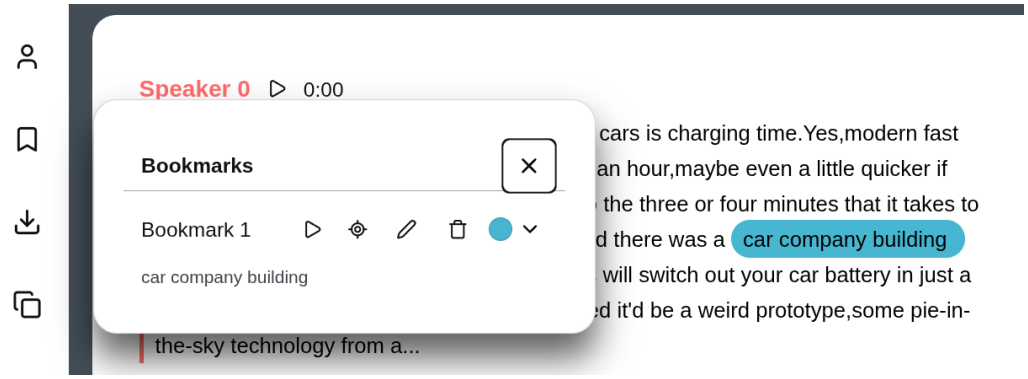
Vlastné kontextové menu editora počas testovania žiadny z účastníkov nevyužil. Neskôr bolo nahradené plávajúcim panelom nástrojov.

8.2 Úpravy po testovaní použiteľnosti desktopovej verzie

Na základe zistení z testovania použiteľnosti boli do aplikácie zapracované úpravy zamerané na odstránenie najčastejšie zaznamenaných problémov.

Záložky

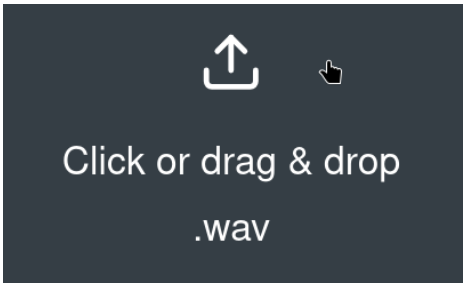
Všetci traja účastníci očakávali, že záložka vizuálne zvýrazní označený úsek textu. Pôvodné správanie vložilo záložku na aktuálnu pozíciu kurzora ako časovú značku, čo nezodpovedalo ich očakávaniu. Záložky boli upravené tak, že pri vložení sa označený textový rozsah v editore vizuálne zvýrazní a v module záložiek sa zobrazuje nielen časová značka, ale aj zodpovedajúci úsek textu a tlačidlo na rýchly presun na príslušnú sekciu textu (obrázok 8.2).



Obr. 8.2: Upravený modul záložiek so zvýraznením textu a odkazom na sekciu.

Pole pre nahratie súboru

Pole pre nahratie súboru metódou *drag and drop* nepôsobilo klikateľne, keďže pri prejazde kurzorom cez neho nedošlo k zmene jeho tvaru. Pole bolo upravené tak, že kurzor sa pri pohybe nad ním zmení na *pointer*, čím používateľovi signalizuje, že prvok je interaktívny a zároveň bola pridaná aj slovná indikácia (obrázok 8.3).

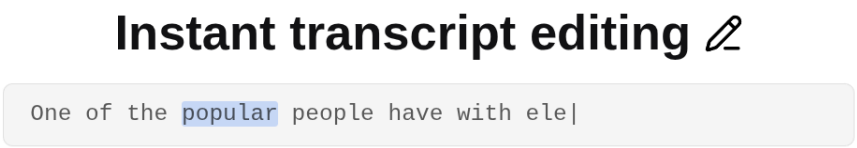


Obr. 8.3: Pole pre nahratie súboru so zmeneným kurzorom a slovnou indikáciou pre kliknutie.

Upozornenie na editáciu v reálnom čase

Žiaden z účastníkov si počas testovania spontánne neuvedomil, že prepis je možné upravovať priebežne počas nahrávania. Do počiatočného stavu editora pred spustením prepisu reči

bol doplnený text, ktorý na túto funkcionalitu nabáda spolu s animáciou editácie prepisu (obrázok 8.4).



Obr. 8.4: Indikácia funkcionality editácie v reálnom čase v počiatočnom stave editora pred spustením prepisu reči. Najskôr sa zobrazí text vyzývajúci používateľa na editáciu, po ktorom nasleduje animácia demonštrujúca priebežnú úpravu prepisu.

Vizuálne vyznačenie záložiek v prehrávači

Záložky sú okrem zvýraznenia v texte vizuálne vyznačené aj priamo v audio prehrávači (obrázok 8.5), čo umožňuje rýchlu orientáciu v kľúčových momentoch záznamu bez nutnosti otvárať modul záložiek.

Ovládacie prvky prehrávača

Prvá a tretia účastníčka postrádali možnosť skočiť v nahrávke o niekoľko sekúnd dozadu alebo dopredu. Do panela prehrávania boli doplnené tlačidlá pre skok o 10 sekúnd dopredu a dozadu. Zároveň bola pridaná možnosť zrýchliť alebo spomaliť nahrávku (obrázok 8.5).



Obr. 8.5: Upravený prehrávač s tlačidlami skoku a záložkami vyznačenými vo vlnovom zobrazení.

Tlačidlá krok späť a krok vpred

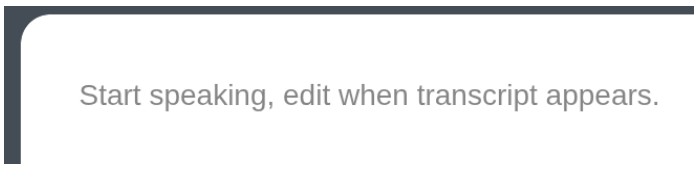
Dvaja účastníci postrádali tlačidlá krok späť a krok vpred. Tieto tlačidlá boli súčasťou návrhu panela nástrojov, no v čase testovania ešte neboli implementované. Po testovaní boli doplnené do panela nástrojov na ľavej strane obrazovky.

Označenie tlačidla pokračovania prepisu

Tlačidlo „Append transcript“ pre opätovné spustenie živého nahrávania bolo pre prvú účastníčku nejasné. Tlačidlo bolo premenované na „Continue“, čo lepšie vystihuje jeho funkciu.

Indikátory stavu nahrávania

Tretia účastníčka si počas testovania nebola istá, či môže začať hovoriť a či prepis skutočne prebieha. Do rozhrania boli preto doplnené vizuálne indikátory (obrázok 8.6). V editore sa zobrazí text, ktorý oznamuje, že užívateľ môže začať hovoriť a po zobrazení prepisu ho aj editovať. V prípade nahratia súboru sa zobrazí, že sa súbor spracováva a zobrazí v editore.



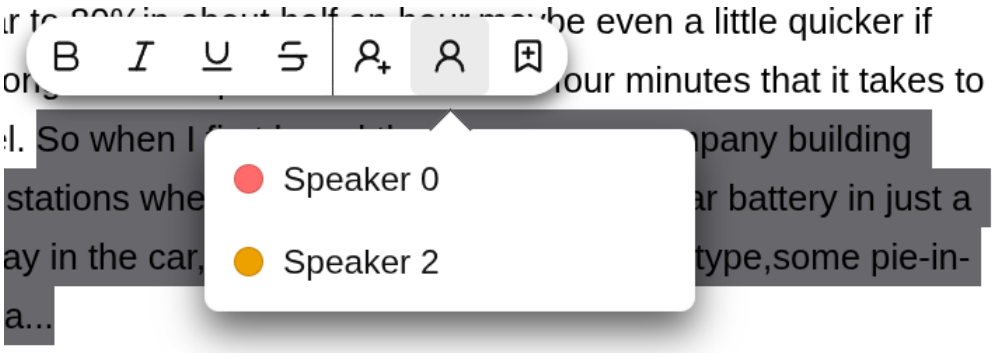
Obr. 8.6: Slovná indikácia v tele editora, ktorá upozorní používateľa, že môže rozprávať pri prepise z mikrofónu.

Plávajúci panel nástrojov

Dvaja účastníci pri označení textu očakávali zobrazenie plávajúceho panela s nástrojmi na formátovanie, na ktorý sú zvyknutí z iných textových editorov. Do editora bol doplnený plávajúci panel, ktorý sa zobrazí v blízkosti výberu pri označení časti textu a ponúka základné možnosti formátovania (obrázok 8.7). Zároveň bolo odstránené pôvodné vlastné kontextové menu, ktoré sa zobrazovalo po kliknutí pravým tlačidlom myši. Jeho súčasná prítomnosť spolu s plávajúcim panelom pôsobila rušivo — pri označení textu sa zobrazili obe prvky naraz, čo vytváralo vizuálny konflikt a zbytočne zahlcovalo rozhranie. Keďže plávajúci panel pokrýva rovnaké funkcie, kontextové menu bolo vypustené.

Vkladanie rečníkov

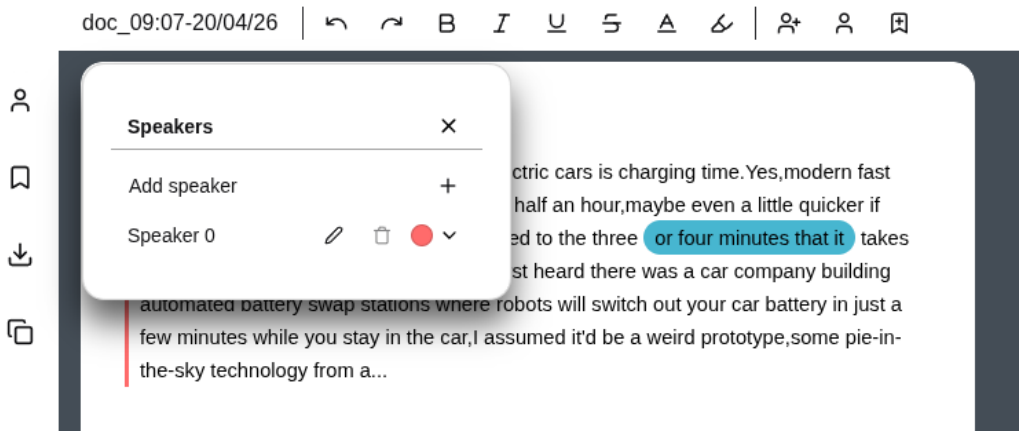
Pôvodná akcia vloženia rečníka do prepisu automaticky vložila už existujúceho rečníka v dokumente. Dvaja účastníci očakávali možnosť vložiť aj úplne nového rečníka. Akcia bola rozdelená na dve samostatné možnosti: vloženie nového rečníka a výber z už existujúcich rečníkov v aktuálnom dokumente (obrázok 8.7).



Obr. 8.7: Rozdelená akcia vloženia rečníka: nový rečník alebo výber z existujúcich.

Moduly ako prekrývajúce okná

Dvaja účastníci sa sťažovali, že po kliknutí na modul v ľavom paneli celý dokument poskočí doprava. Moduly boli prepracované tak, že sa zobrazujú ako plávajúce prekrývajúce okná (*popupy*), ktoré sa vrstvia nad dokumentom bez toho, aby ovplyvňovali jeho pozíciu v stránke (obrázok 8.8). Dokument tak pri používaní modulov nepremiestňuje.

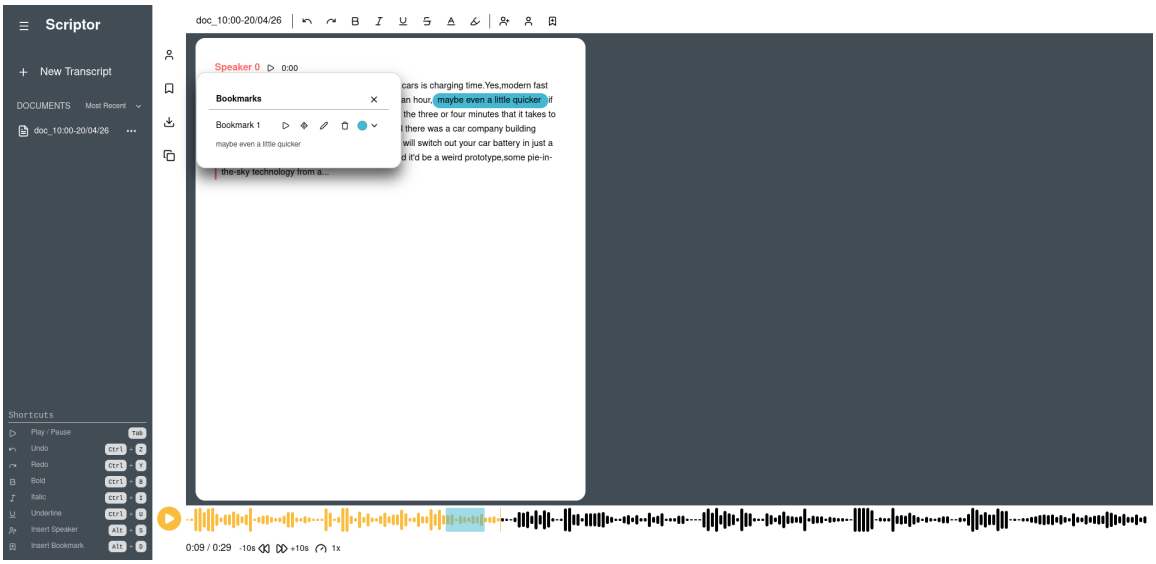


Obr. 8.8: Moduly zobrazené ako plávajúce okná prekrývajúce dokument.

Klávesové skratky v bočnom paneli

Keďže účastníci klávesové skratky počas testovania vôbec nevyužívali, do bočného panela boli doplnené stručné nápovedy (*quick hints*), ktoré zobrazujú dostupné klávesové skratky priamo v rozhraní (obrázok 8.9). Cieľom je sprístupniť skratky používateľom, ktorí ich existenciu nemusia aktívne vyhľadávať.

Na obrázku 8.9 je vidieť stav aplikácie po uplatení zmien.



Obr. 8.9: Stav aplikácie po zapracovaní úprav plynúcich z testovania použiteľnosti.

8.3 Testovanie použiteľnosti mobilnej verzie

Po testovaní použiteľnosti desktopovej verzie, oprave kritických problémov a doplnení synchrónneho prepisu (sekcia 7.7) bola navrhnutá mobilná verzia editora (kapitola 4). Tak ako desktopová verzia aj mobilná podstúpila testovaniu použiteľnosti.

Výber účastníkov

Prvý a druhý účastník boli tí istí ako pri testovaní desktopovej verzie – 21-ročná študentka a 22-ročná študentka a laborantka. Tretím účastníkom bol nový záujemca, ktorý sa nezúčastnil desktopového testovania. Cieľom bolo mať testera mobilnej verzie, ktorý ešte nemal žiadne skúsenosti s editorom.

Priebeh testovania

Testovanie prebehlo rovnakou metodológiou ako pri desktopovej verzii. Účastníci sa mali pokúsiť splniť rôzne úlohy a nahlas komentovať svoj postup myslenia. Aplikácia bola spustená v mobilnom prehliadači na smartfóne účastníka, aby bolo používanie aplikácie čo najprirodzenejšie. Prvý a tretí účastník používali iPhone 12 s prehliadačom Safari, druhý účastník Samsung Galaxy A53 s prehliadačom Google Chrome. Účastníci, ktorí sa zúčastnili aj desktopového testovania, na začiatku sedenia stručne zhodnotili zmeny zapracované do desktopovej verzie.

Postup a zadanie úloh

Účastníkom boli zadané rovnaké úlohy ako pri testovaní desktopovej verzie. Každý účastník dostal opäť stručný kontext zohľadňujúci jeho zamestnanie, aby bolo používanie aplikácie čo najprirodzenejšie.

Výsledky

Účastník 1 Na začiatku sedenia sa prvá účastníčka vyjadrila k zmenám zapracovaným do desktopovej verzie po prvom kole testovania. Úpravy hodnotila kladne a ocenila, že problémy, ktoré nahlásila pri prvom testovaní, boli zapracované. Nápovedy klávesových skratiek si pri prvom pohľade nevšimla, no keď na ne bola upozornená, poznamenala, že pri dlhodobejšom používaní by ich pravdepodobne využívala.

V mobilnej verzii narázila na niekoľko drobných nejasností. V module záložiek jej nebola na prvý pohľad zrejmá funkcia tlačidla skoku na záložku. Nevedela odhadnúť, čo kliknutie vyvolá. Po nahratí súboru považovala tlačidlá pre výber režimu transkripcie za ťažko rozoznateľné a nevedela intuitívne určiť význam jednotlivých tlačidiel a čo sa stane po kliknutí.

Vo všeobecnosti sa jej s aplikáciou pracovalo dobre a väčšinu úloh zvládla plynulo bez väčších ťažkostí.

Účastník 2 Na začiatku sedenia sa druhá účastníčka vyjadrila k zmenám zapracovaným do desktopovej verzie po prvom kole testovania. Hodnotila ich pozitívne. Ocenila najmä nápovedy klávesových skratiek v bočnom paneli a odlišný vizuálny štýl záložiek. Napriek tomu si všimla niekoľko pretrvávajúcich nedostatkov. Pri nahrávaní súboru už vedela, že môže na pole drag&drop kliknúť, ale keďže pole po kliknutí nezošedlo a kurzor mal naďalej

formu ukazovateľa, nadobudla dojem, že môže nahrať aj ďalší súbor. Rozlíšenie medzi režimami *batch* a real-time pri nahrávaní súboru jej nebolo jasné a plietlo sa jej s možnosťou živého nahrávania z mikrofónu. Na prvý pohľad jej nebol jasný význam tlačidla pre rýchly presun na záložku. Poznamenala tiež, že pravá strana obrazovky pôsobí prázdne. Editor zaberá len časť šírky a zvyšok priestoru sa javí ako nevyužitý.

Na mobilnej verzii jej prišlo neintuítívne, že po kliknutí na tlačidlo skoku na záložku v module záložiek sa tento modul automaticky nezavrie, čo prerušovalo jej pracovný tok a vyžadovalo manuálne zatváranie modulu. Lišta v ľavom okraji obrazovky určená na manipuláciu s modulmi jej kvôli zaobleným rohom na pravej strane navodila dojem, že ide o prvok určený na gesto *swipe* zľava doprava, čo nezodpovedá skutočnej funkcii prvku.

Účastník 3 Tretí účastník bol 30-ročný obchodný zástupca v telekomunikačnej firme. Počítač používa každodenne a v oblasti digitálnych nástrojov sa cíti technicky zdatný. Ako jediný z trojice sa nezúčastnil desktopového testovania. S aplikáciou sa stretol po prvýkrát, čo umožnilo pozorovať, ako nový používateľ bez akýchkoľvek predchádzajúcich skúseností pristupuje k mobilnej verzii.

Hneď na začiatku poznamenal, že niektoré ikony a interaktívne prvky mu prídu pri dotykovom ovládaní príliš malé. Toto sa prejavilo najmä pri manipulácii s ovládacími tlačidlami, kde musel opakovane korigovať klepnutia. Pri nahrávaní súboru mu nebolo jasné, aký je rozdiel medzi režimom dávkového spracovania (*batch*) a real-time prepisom. Pri tlačidlách chýbal popis, ktorý by mu umožnil správnu voľbu bez váhania. Taktiež mu na prvý pohľad nebolo zrejmé, na čo slúži tlačidlo rýchleho presunu na záložku v module záložiek.

Zhrnutie

Testovanie mobilnej verzie prebehlo s tromi účastníkmi, pričom dvaja mali predchádzajúce skúsenosti s desktopovou verziou a jeden sa s aplikáciou stretol po prvýkrát. Tabuľka 8.2 zachytáva zistené nedostatky a to, u ktorých účastníkov sa prejavili.

Tabuľka 8.2: Prehľad identifikovaných problémov použiteľnosti mobilnej verzie (Ú1–Ú3 = účastník 1–3).

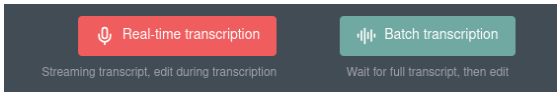
Problém	**Ú1**	**Ú2**	**Ú3**
Ťažkosti s voľbou medzi *batch* a real-time režimom	✓	✓	✓
Tlačidlo skoku na záložku nejasné	✓	✓	✓
Modul záložiek sa po skoku nezavrie	×	✓	×
Lišta pôsobí ako prvok pre gesto *swipe*	×	✓	×
Ikony a tlačidlá príliš malé	×	×	✓

Najčastejšie zaznamenaným problémom bola nejednoznačnosť voľby medzi režimami dávkového a real-time spracovania pri nahrávaní súboru, tento problém sa vyskytol u všetkých troch účastníkov. Tlačidlo skoku na záložku bolo nejasné pre všetkých troch účastníkov. Druhá účastníčka navyše zaznamenala, že modul záložiek sa po kliknutí na toto tlačidlo nezavrie. Problémy špecifické pre jednotlivých účastníkov súviseli najmä s vizuálnym dojmom rozhrania – dojem gesta *swipe* z bočnej lišty u druhej účastníčky a malé ikony u nového účastníka.

8.4 Úpravy po testovaní použiteľnosti mobilnej verzie

Rozlíšenie režimov transkripcie

Na vyriešenie tejto nejednoznačnosti boli do dialógového okna pre nahratie súboru doplnené stručné popisy pod každé tlačidlo a upravený ich vizuál. Tlačidlo pre živý prepis (*Live transcription*) obsahuje body „No waiting“ a „Edit in real-time“ a je označené bielym krúžkom, ktorý má indikovať živý prenos. Tlačidlo pre dávkový prepis (*Full transcription*) obsahuje body „Wait for full transcript“ a „Speaker recognition“, je farebne odlíšené modrou farbou a doplnené ikonou presýpacích hodín pre indikáciu čakania. Obrázky 8.10 a 8.11 zachytávajú stav dialógu pred a po tejto úprave.



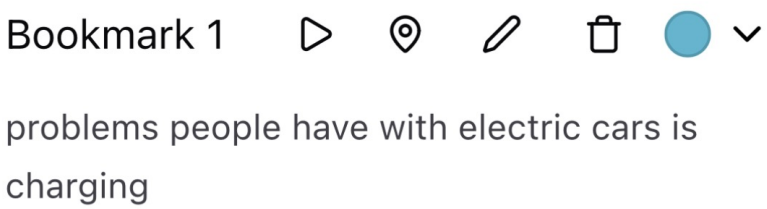
Obr. 8.10: Tlačidlá pre výber režimu prepisu súboru pred úpravou



Obr. 8.11: Tlačidlá pre výber režimu prepisu súboru po úprave

Vizuál a správanie tlačidla skoku na záložku

Ďalším problémom bola nejasnosť funkcie tlačidla v module záložiek, ktoré má umožniť rýchle zobrazenie záložky editore. V desktopovej verzii bol k tlačidlám v moduloch pridaný tooltip, ktorý vysvetľuje funkciu tlačidla, a zároveň bola zmenená ikona, ktorá lepšie vystihuje myšlienku lokácie (obrázok 8.12).



Obr. 8.12: Modul záložiek po testovaní použiteľnosti

8.5 Záver

Testovanie použiteľnosti prebehlo v dvoch kolách a umožnilo iteratívne zlepšovať rozhranie na základe reálnej spätnej väzby. Prvé kolo potvrdilo, že základný pracovný tok bol pre účastníkov zvládnuteľný bez predchádzajúceho zaškolenia, no odhalilo problémy ako neintuítívne fungovanie záložiek alebo nejasná možnosť editácie počas vkladania prepisu.

Druhé kolo overilo mobilnú verziu a zároveň čiastočne potvrdilo účinnosť úprav z prvého kola. Účastníci, ktorí poznali desktopovú verziu, hodnotili zmeny kladne. Nové problémy, predovšetkým nejednoznačnosť voľby medzi režimami prepisu zo súboru a nejasné tlačidlo skoku na záložku, poukazovali na to, že niektoré prvky rozhrania si vyžadujú explicitnejší popis bez ohľadu na kontext použitia.

Iteratívny prístup k testovaniu sa ukázal ako prínosný. Každé kolo odhalilo iný typ problémov. Prvé predovšetkým koncepčné nedostatky návrhu, druhé problémy špecifické pre mobilný kontext. Tento postup umožnil sústrediť úpravy tam, kde mali najväčší dopad, a overenie v druhom kole potvrdilo, že identifikované problémy boli riešené správnym smerom.

Kapitola 9

Záver

Väčšina existujúcich editorov prepisu reči funguje ako komplexné platformy závislé od vlastnej serverovej infraštruktúry, pričom editácia je spravidla možná až po dokončení celého prepisu. Cieľom tejto práce bolo navrhnúť a implementovať editor, ktorý tieto obmedzenia prekonáva: celá aplikácia beží v prehliadači, dáta zostávajú na zariadení používateľa a editácia prebieha súbežne s prepisom v reálnom čase.

Boli preskúmané existujúce editory prepisu reči so zameraním na ich UX a UI, analyzované dostupné API služby pre automatický prepis a frontendové nástroje na tvorbu UI. Analýza ukázala, že väčšina dostupných editorov neumožňuje editáciu počas prebiehajúceho prepisu a spolieha sa na vzdialenú infraštruktúru. Na základe prieskumu bola zvolená kombinácia knižníc React a Next.js, rich-text editora Slate.js, knižnice WaveSurfer.js pre vizualizáciu audia a Gladia SDK pre real-time prepis reči.

Výsledná aplikácia predstavuje funkčný nástroj pokrývajúci celý pracovný postup: od nahrávania z mikrofónu alebo načítania existujúceho súboru, cez automatický prepis v reálnom čase aj v dávkovom režime, až po editáciu s podporou rich textu, správu rečníkov a záložiek. Poskytuje obojsmernú synchronizáciu audia a textu na úrovni jednotlivých slov. Rozhranie je dostupné aj v mobilnej verzii a bolo navrhované iteratívne na základe spätnej väzby od používateľov. Prínosom práce je editor, ktorý na rozdiel od analyzovaných riešení nevyžaduje vlastnú serverovú infraštruktúru a umožňuje editovať prepis priamo počas jeho vkladania.

Testovanie použiteľnosti prebehlo v dvoch kolách a viedlo k sérii konkrétnych úprav: záložky boli prepracované na vizuálne zvýraznenie textu, moduly zmenené na plávajúce okná a doplnený plávajúci panel nástrojov pri výbere textu. Druhé kolo potvrdilo správnosť vykonaných zmien a odhalilo ďalšie, menej závažné nedostatky, ktoré boli následne tiež zapracované. Všetky ciele zadania práce boli splnené.

Hlavným limitom riešenia zostáva závislosť od externej služby poskytujúcej prepis reči. Plnohodnotné rozpoznávanie reči nie je v prostredí prehliadača bez sieťového volania reálne – lokálne modely sú výpočtovo náročné a bez internetového pripojenia prepis nefunguje. Automatická diarizácia rečníkov je podporovaná iba v dávkovom režime; pri prepise v reálnom čase musí rečníkov priradiť používateľ ručne. Do budúcnosti by bolo vhodné doplniť podporu titulkových formátov, umožniť výber poskytovateľa automatického prepisu reči a integrovať AI funkcie ako automatický súhrn prepisu.

Aplikácia je verejne dostupná na adrese https://scriptor.app/, kde je nasadená aj s funkčným API kľúčom pre službu Gladia, predbežne do konca augusta 2026. Pri lokálnom spustení si používateľ musí zaobstarať vlastný API kľúč na stránke Gladia. Zdrojový kód je dostupný na https://github.com/scriptor. Aplikácia bola vydaná pod licenciou MIT.

Literatúra

[1] Artificial Analysis. *Speech to Text AI Model & Provider Leaderboard* online. 2025. Dostupné z: https://artificialanalysis.ai/speech-to-text. [cit. 2026-03-28].

[2] Krug, S. *Rocket surgery made easy: the do-it-yourself guide to finding and fixing usability problems*. 1. vyd. Berkeley: New Riders, 2009. ISBN 9780321657299.

[3] Krug, S. *Don't make me think, revisited: a common sense approach to web usability*. 3. vyd. Berkeley: New Riders, 2014. ISBN 9780321965516.

[4] Nielsen, J. *F-Shaped Pattern For Reading Web Content* online. 2006. Dostupné z: https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content-discovered/. [cit. 2026-04-25].

[5] Rubin, J.; Chisnell, D. a Spool, J. *Handbook of usability testing: how to plan, design, and conduct effective tests*. 2. vyd. Indianapolis: Wiley Publishing, 2008. ISBN 9780470185483.

[6] Tidwell, J. *Designing Interfaces*. 2. vyd. Sebastopol: O'Reilly, 2010. ISBN 9781449379704.