

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

UŽIVATELSKÉ ROZHRANÍ PRO SBĚR BIOMETRICKÝCH DATOVÝCH SAD

USER INTERFACE FOR COLLECTING BIOMETRIC DATASETS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MATĚJ NOVÁK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MAREK VAŠKO,**
SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

160596

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
Student: **Novák Matěj**
Program: Informační technologie
Název: **Uživatelské rozhraní pro sběr biometrických datových sad**
Kategorie: Uživatelská rozhraní
Akademický rok: 2024/25

Zadání:

1. Seznamte se s problematikou vývoje aplikací pro mobilní zařízení.
2. Prozkoumejte aktuální řešení sběru obrázkových datových sad pro využití v metodách strojového učení.
3. Prostudujte možnosti využívání vizuální podpory pro získávání datových sad snímků rukou.
4. Prototypujte dílčí prvky uživatelského rozhraní umožňujícího pořizování co nejlepších snímků rukou z více pohledů. Testujte a iterativně vylepšujte navržený prototyp.
5. Integrujte navržené rozhraní do aplikace, testujte a iterativně vylepšujte aplikaci ze zpětné vazby od uživatelů.
6. Proveďte uživatelské testování aplikace s cílovou skupinou a zhodnoťte její účinnost a využitelnost pro sběr datové sady.
7. Zhodnoťte dosažené výsledky a diskutujte o možnostech budoucího vývoje. Vytvořte krátké video prezentující tyto výsledky.

Literatura:
Podle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:
1, 2, 3, 4 a značně rozpracované cíle 5 a 6.

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

Vedoucí práce: **Vaško Marek, Ing.**
Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.
Datum zadání: 1.11.2024
Termín pro odevzdání: 14.5.2025
Datum schválení: 12.11.2024

Abstrakt

Tato bakalářská práce se zaměřuje na návrh a implementaci progresivní webové aplikace *Capture My Hand*, určené pro efektivní sběr obrazových dat lidské ruky. Cílem projektu bylo vytvořit multiplatformní řešení, které umožňuje intuitivní pořizování fotografií z mobilního zařízení bez nutnosti instalace. Práce se věnuje návrhu uživatelského rozhraní na základě zásad UX pro mobilní aplikace, výběru vhodných technologií a knihoven (React, FastAPI, Intro.js, Toastify) a iterativnímu testování s reálnými uživateli. Aplikace podporuje detekci ruky na obrazovce, validaci vstupních dat a poskytuje zpětnou vazbu prostřednictvím toast notifikací. Výsledkem je udržitelný nástroj pro strukturovaný sběr dat, vhodný pro využití širší veřejností.

Abstract

This bachelor thesis focuses on the design and implementation of a progressive web application called *Capture My Hand*, intended for efficient image data collection of the human hand. The goal of the project was to develop a multiplatform solution enabling intuitive photo capture from mobile devices without the need for installation. The thesis covers the design of the user interface based on mobile UX principles, the selection of appropriate technologies and libraries (React, FastAPI, Intro.js, Toastify), and iterative testing with real users. The application supports hand detection in the image, input data validation, and provides user feedback via toast notifications. The result is a sustainable tool for structured data collection, suitable for use by a broader public.

Klíčová slova

PWA, React, FastAPI, biometrie, uživatelské rozhraní, sběr dat, UX design, validace, notifikace

Keywords

PWA, React, FastAPI, biometrics, user interface, data collection, UX design, validation, notifications

Citace

NOVÁK, Matěj. *Uživatelské rozhraní pro sběr biometrických datových sad.* Brno, 2025. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Marek Vaško,

Uživatelské rozhraní pro sběr biometrických datových sad

Prohlášení

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

.....
Matěj Novák
13. května 2025

Poděkování

Děkuji vedoucímu práce panu Marku Vaškovi za kvalitní konzultace a vstřícný přístup v průběhu semestru.

Obsah

1 Úvod **4**

2 Průzkum trhu **5**
2.1 Aplikace pro pořizování fotografií 5
2.2 Aplikace pro asistované pořizování fotografií 7

3 Principy návrhu mobilních a PWA aplikací **9**
3.1 Intuitivnost 9
3.2 Zvyklosti 10
3.3 Vizuální hierarchie 10
3.4 Prostorová orientace 11
3.5 Behaviorální vzory 11
3.6 Progresivní odhalování 12
3.7 Mikrointerakce a zpětná vazba 12

4 Průzkum a výběr implementačních technologií **14**
4.1 JavaScriptové frameworky 14
4.2 Frameworky pro kaskádové styly 16
4.3 Výběr knihoven 18
4.4 Algoritmy pro hodnocení kvality snímku 20
4.5 MediaPipe 20

5 Grafické rozhraní **22**
5.1 Intro.js 23
5.2 Toastify 23
5.3 Pracovní postup uživatele 24

6 Implementace **25**
6.1 Vývojový design aplikace 25
6.2 Technologie a způsob fungování backendu 26
6.3 Stručný popis vybraných částí implementace 29

7 Testování **33**
7.1 Testování mobilních aplikací s uživateli 33
7.2 Jednotlivé iterace 35

8 Závěr **40**

Literatura **41**

A Konečný stav aplikace **43**

B Testovací protokol **48**

Seznam obrázků

2.1 Uživatelské rozhraní aplikace fotoaparát pro operační systém iOS 6
2.2 Uživatelské rozhraní aplikace fotoaparát pro operační systém iOS 7

5.1 Ukázky obrazovek uživatelského rozhraní aplikace Capture My Hand: pořizování fotografií, galerie, nápověda, sync a formulář pro zadání metadat. . . 22
5.2 Diagram znázornující interakci uživatele s aplikací Capture My Hand. . . . 24

6.1 Schéma zpracování snímku a rozhodnutí o jeho kvalitě 32

7.1 Nejčastěji hlášené problémy během testování aplikace Capture My Hand. . 35
7.2 Průměrné hodnocení jednotlivých aspektů aplikace Capture My Hand podle uživatelů. 36
7.3 Problémy zjištěné během druhé iterace testování. 37
7.4 Průměrné hodnocení aplikace po druhé iteraci testování. 38
7.5 Hodnocení aplikace po zavedení validace kamery a onboardingu ve formuláři. 39

A.1 Úvodní obrazovka aplikace, upozornění na absenci ruky a nízkou kvalitu snímku. 44
A.2 Použití Intro.js na úvodní obrazovce a ve formuláři. 45
A.3 Použití toast notifikací ve formuláři a po odeslání formuláře. 46
A.4 Toast notifikace při nevyplnění žádného z povinných polí, nápověda jak zachycovat snímky a správně vyplněný formulář. 47

B.1 Testovací protokol pro všechny 3 iterace. 49

Kapitola 1

Úvod

Digitální technologie dnes zásadním způsobem ovlivňují způsob, jakým sbíráme, analyzujeme a uchováváme vizuální informace. Zatímco oblast počítačového vidění a mobilních aplikací se rozvíjí mimořádně rychle, zůstává stále velký potenciál pro tvorbu specializovaných nástrojů určených pro strukturovaný sběr obrazových dat. Tato bakalářská práce reaguje na tuto výzvu návrhem a implementací progresivní webové aplikace (PWA), která slouží k pořizování fotografií hřbetu ruky a jejich následnému odesílání do centrální databáze. Projekt vznikl jako rozšíření a redesign původního řešení vytvořeného Ing. Markem Vaškem a klade důraz především na použitelnost, jednoduchost ovládání a technickou udržitelnost.

Aplikace *Capture My Hand* je určena pro prostředí, kde je vyžadováno rychlé a intuitivní pořízení dat bez zbytečné zátěže na straně uživatele. Využití technologie PWA umožňuje provoz bez nutnosti instalace z aplikačních obchodů, a zároveň zachovává klíčové výhody nativních aplikací, jako je práce offline nebo přímý přístup ke kameře zařízení. Díky použití knihovny React na straně frontendové logiky a frameworku FastAPI na straně serverové části vzniklo multiplatformní řešení kombinující flexibilitu webových aplikací s výkonností moderního REST API.

V rámci práce byl kladen důraz na dodržení zásad přístupného návrhu uživatelského rozhraní, jako je vizuální hierarchie, prostorová orientace či princip progresivního odhalování funkcí. Aplikace byla dále rozšířena o interaktivní prvky nápovědy (onboarding), validaci vstupů na základě detekce ruky v obraze a přehlednou zpětnou vazbu pro uživatele pomocí tzv. toast notifikací. Celý vývoj probíhal v iteracích, jejichž cílem bylo nejen technické zdokonalení aplikace, ale především zajištění vysoké míry intuitivnosti při jejím používání.

Tato práce dokumentuje celý proces – od výběru vhodných technologií přes návrh uživatelského rozhraní až po testování s reálnými uživateli. Výsledkem je stabilní a snadno rozšiřitelná aplikace, která může sloužit jako nástroj pro organizovaný sběr obrazových dat v různých kontextech, včetně výzkumu, zdravotnictví či biometrické identifikace.

Kapitola 2

Průzkum trhu

Před samotným návrhem aplikace je nezbytné provést důkladný průzkum trhu. Ten představuje klíčovou fázi vývoje, neboť umožňuje identifikovat stávající řešení, která jsou na trhu již dlouhodobě ověřena a úspěšně využívána. Zároveň poskytuje cenné poznatky o zavedených konvencích a uživatelských zvyklostech, jejichž respektování přispívá k vyšší míře srozumitelnosti a přijatelnosti navrhovaného uživatelského rozhraní.

2.1 Aplikace pro pořizování fotografií

Mobilní zařízení běžně obsahují předinstalované aplikace pro pořizování fotografií, které se vyznačují rychlým spuštěním, intuitivním ovládáním a přímým přístupem k fotoaparátu zařízení. Tyto nativní aplikace jsou navrženy s ohledem na ergonomii a minimalistické rozhraní, čímž podporují efektivní a pohodlné zachycení obrazu.

V následujících podkapitolách jsou představeny konkrétní příklady takových aplikací v prostředí systémů iOS a Android. Pozornost je věnována především rozmístění ovládacích prvků, stylu vizuální prezentace a způsobu, jakým jednotlivé systémy podporují kompozici a rychlé snímání.

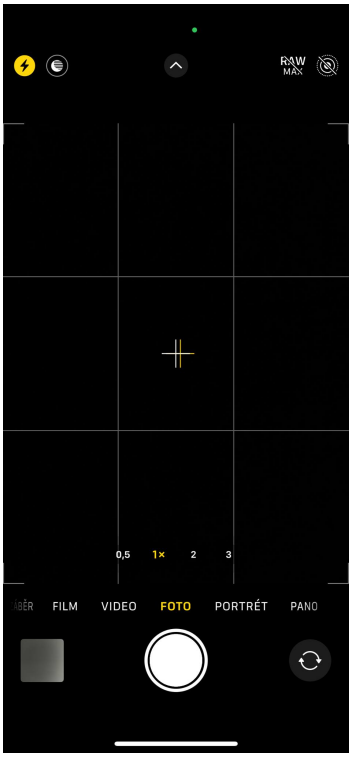
Nativní aplikace fotoaparát pro iOS

Nativní aplikace Fotoaparát v systému iOS je příkladem minimalistického a funkčně vyváženého uživatelského rozhraní, které klade důraz na rychlý přístup k nejdůležitějším ovládacím prvkům. Uživatelské prostředí je optimalizováno pro ovládání jednou rukou.

V dolní části obrazovky se nachází spoušťová tlačítka – hlavní kruhové tlačítko pro pořízení snímku je umístěno uprostřed, doprovázené menšími ikonami vlevo a vpravo pro přístup k náhledu galerie a přepínání mezi přední a zadní kamerou. Tyto prvky jsou vizuálně kontrastní a snadno dostupné palcem při běžném držení zařízení.

Horní okraj obrazovky je vyhrazen pro nastavení, která se dynamicky mění podle zvoleného režimu snímání. Mezi tyto ikony patří například nastavení blesku, časovače nebo filtrů.

Střední část obrazovky je vyplněna náhledem z kamery, který je ohraničen pouze minimálními prvky. Uživatel má tak zajištěn maximální vizuální prostor pro kompozici snímku. V dolní části náhledu je umístena karuselová nabídka režimů, kterou lze horizontálně posouvat. Tato nabídka je stylizována jako textové popisky s decentním zvýrazněním aktuálně vybraného režimu. Vzhled aplikace je ukázán na obrázku 2.1.



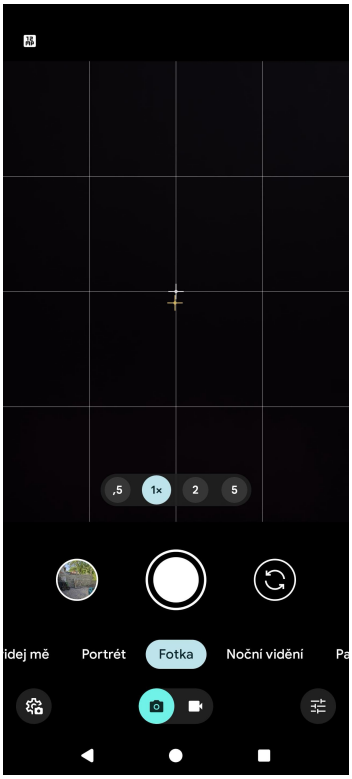
Obrázek 2.1: Uživatelské rozhraní aplikace fotoaparát pro operační systém iOS

Nativní aplikace fotoaparát pro Android

Nativní aplikace Fotoaparát v systému Android se může v detailech lišit podle výrobce zařízení, ale většina moderních verzí zachovává podobnou strukturu uživatelského rozhraní. Cílem je nabídnout uživateli přehledné prostředí s rychlým přístupem ke všem klíčovým funkcím při fotografování.

Ve spodní části obrazovky se nachází hlavní panel ovládání, kde je dominantní centrální kruhové tlačítko sloužící jako spoušť. Po levé a pravé straně tohoto tlačítka se zpravidla nachází ikony pro přístup do galerie a přepínání mezi kamerami. Horní část obrazovky je určena pro nastavení, jako je ovládání blesku, poměru stran nebo přístup k dalším možnostem v rozbalovací nabídce. Rozvržení těchto ikon je většinou horizontální a odpovídá aktuálně zvolenému režimu snímání.

Střed obrazovky je vyhrazen živému náhledu kamery s minimálním vizuálním rušením. Ve spodní části náhledu je často umístěna lišta s režimy, která se horizontálně posouvá a umožňuje rychlé přepínání mezi režimy. Vzhled této nabídky se liší podle výrobce, ale funkčně odpovídá iOS karuselu. Vzhled aplikace je ukázán na obrázku 2.2.



Obrázek 2.2: Uživatelské rozhraní aplikace fotoaparát pro operační systém iOS

Celkově je GUI navrženo tak, aby bylo intuitivní a snadno použitelné i pro nezkušené uživatele.

2.2 Aplikace pro asistované pořizování fotografií

Kromě běžných nativních aplikací pro fotografování existují také specializované nástroje, které uživatelům poskytují interaktivní asistenci při pořizování snímků. Tyto aplikace využívají různé technologie, jako je rozšířená realita nebo umělá inteligence, aby uživatelům pomohly dosáhnout požadované kompozice nebo stylu fotografie. Následující podsekce představuje aplikaci PicMe.

PicMe

PicMe je mobilní aplikace, která poskytuje interaktivní vizuální asistenci při pořizování fotografií s požadovanou kompozicí. Uživatel může obdržet požadavek na konkrétní typ snímku a aplikace mu následně poskytne vizuální vodítka, která mu pomohou správně zarámovat a pořídit fotografii podle požadavků. Tato funkcionalita je zvláště užitečná v situacích, kdy je důležité dodržet specifické kompoziční požadavky, například při dokumentaci nebo sběru dat [12].

Funkce

- **2.5D virtuální rám** – Zobrazuje trojrozměrný obrys požadované kompozice v reálném čase, což umožňuje fotografovi přesně zarovnat kameru s cílovým záběrem

- **Obraz v obraze** – Malý náhled referenčního snímku v rohu obrazovky slouží jako vodítko pro hrubé zarovnání
- **Zvýraznění hran** – Po dosažení přibližného zarovnání aplikace zvýrazní kontury cílového obrazu, což umožňuje jemné doladění kompozice

Kapitola 3

Principy návrhu mobilních a PWA aplikací

V této kapitole jsou představeny klíčové zásady, které ovlivňují návrh uživatelského rozhraní mobilních aplikací a progresivních webových aplikací. Vzhledem k tomu, že hlavním cílovým zařízením je mobilní telefon, ale aplikace je zároveň přístupná i skrze webový prohlížeč, je důležité reflektovat specifika obou platforem.

3.1 Intuitivnost

Intuitivní interakce představuje základ každého kvalitního návrhu mobilní aplikace. V praxi to znamená, že uživatel je schopen aplikaci používat bez nutnosti číst návody, absolvovat školení či využívat externí podporu. Úspěšný návrh vychází ze schopnosti uživatele ihned pochopit, co dané ovládací prvky znamenají a jaké akce od něj systém očekává. Cílem je minimalizovat kognitivní zátěž a vytvořit prostředí, ve kterém se uživatel cítí jistě a neztrácí čas hledáním funkcí.

Mobilní prostředí je přitom charakteristické několika omezeními – především malým displejem, častým používáním za pochodu a ovládáním prstem, nikoliv kurzorem. To má přímé důsledky pro návrh interakcí. Ovládací prvky, jako jsou tlačítka či vstupní pole, musí být dostatečně velké a mezi sebou dostatečně vzdálené, aby nedocházelo k neúmyslnému stisku. Reakce systému na uživatelské vstupy by měla být okamžitá a doprovázena jasnou vizuální zpětnou vazbou – například změnou barvy tlačítka, animací nebo krátkým zvukovým signálem.

Intuitivnost návrhu lze podpořit i pomocí známých vzorců chování. Například tlačítko zpět se očekává v levém horním rohu a potvrzovací akce vpravo dole. Pokud je uživatelská cesta navržena logicky a respektuje ustálené konvence, uživatel se snadno zorientuje i v novém prostředí. Vývojář by měl dbát na to, aby uživatel v každém okamžiku věděl, kde se nachází, co se od něj očekává a jaký bude výsledek jeho akce.

Dalším prvkem intuitivní interakce je prediktivní návrh, který vychází z kontextu a očekávání uživatele. Například aplikace pro správu financí může po detekci nové transakce automaticky zobrazit nabídku pro její zaúčtování. Tento přístup nejen zrychluje interakci, ale také zvyšuje spokojenost uživatele, protože systém „předvídá“ jeho potřeby.

Nezanedbatelným aspektem intuitivního rozhraní je i schopnost systému odpouštět chyby. Uživatelé by měli mít vždy možnost vrátit se zpět, zrušit akci nebo opravit nesprávný vstup. Absence těchto možností může vést ke frustraci a negativnímu vnímání

aplikace. Intuitivnost je tak úzce propojena s celkovou použitelností – jak ji definuje Bevan, použitelnost zahrnuje nejen efektivitu a účinnost, ale také spokojenost uživatele při dosahování jeho cílů [5].

3.2 Zvyklosti

Uživatelé si během práce s aplikací vytvářejí mentální představy o jejím fungování – označované jako „mentální modely“ [17]. Pokud aplikace odpovídá jejich očekáváním, stává se její používání přirozeným. Typickým příkladem je umístění tlačítka zpět vlevo nahoře nebo potvrzovací akce vpravo dole. Pokud se tyto konvence poruší, uživatelé se mohou cítit dezorientovaní a aplikaci opustit. To platí zejména v případě mobilních aplikací, kde se očekává jednotné chování napříč systémem i napříč aplikacemi.

Důležitou roli hrají tzv. „motorické zvyklosti“ – automatizované sekvence chování, které uživatelé používají bez přemýšlení. Pokud například místo potvrzení dojde k nechtěnému smazání, protože se tlačítka nacházela na opačných pozicích, vzniká chyba čistě z důvodu narušení očekávání. Podobné případy vedou ke ztrátě důvěry v aplikaci. Právě proto je konzistence v návrhu jedním z klíčových principů návrhu uživatelského rozhraní.

Mentální modely se také utvářejí skrze analogie s fyzickým světem – například „koš“ pro smazané soubory nebo „zvonek“ pro notifikace. Pokud aplikace nabídne podobné metafory, je snazší se v ní orientovat. Vývojář by měl mít na paměti, že každá odchylka od běžných vzorců vyžaduje novou mentální adaptaci.

3.3 Vizuální hierarchie

Vizuální složka rozhraní hraje v mobilních aplikacích zásadní roli, neboť prostor na obrazovce je omezený a uživatelé se spoléhají na rychlé vizuální rozpoznání důležitých prvků. Vytvoření správné vizuální hierarchie znamená uspořádat prvky na obrazovce tak, aby nejdůležitější informace a akce byly snadno rozpoznatelné a přirozeně vedly uživatelovu pozornost. Toho lze dosáhnout pomocí velikosti prvků, kontrastu, barvy, polohy, ale i bílého prostoru, který umožňuje prvkům „dýchat“ a nepůsobit přeplněně.

Na mobilních zařízeních se běžně uplatňuje takzvaný „F-vzor“ nebo „Z-vzor“ pro skenování obrazovky, a je proto vhodné umisťovat klíčové informace do levého horního a pravého dolního kvadrantu. Tlačítka pro hlavní akce, jako je potvrzení nebo odeslání, by měla být zvýrazněna barevně a vizuálně oddělena od méně důležitých prvků. Příliš velké množství barev, textů nebo ikon vede k vizuálnímu šumu, který nejenže snižuje čitelnost, ale také zatěžuje uživatelovu kognitivní kapacitu.

Vhodnou strategií, jak předejít zahlcení, je princip tzv. „progresivního odhalování“ (progressive disclosure), kdy jsou méně důležité informace nebo pokročilé funkce skryty a zobrazují se až na základě kontextu nebo akce uživatele. Tím se zachová čistota rozhraní a zároveň se neztrácí plná funkcionalita aplikace. Praktickým příkladem může být nastavení detailů notifikací, které se objeví až po klepnutí na hlavní přepínač. Minimalistický design by měl být funkcí, nikoliv estetickým cílem – jeho účelem je přehlednost a efektivita.

Z výzkumů vyplývá, že aplikace s jasnou vizuální strukturou a omezeným množstvím rušivých prvků vykazují vyšší míru konverzí a nižší míru chyb při používání [26].

3.4 Prostorová orientace

Prostorová paměť hraje významnou roli v tom, jak si uživatelé zapamatovávají rozložení prvků v aplikaci. Při každodenním používání se uživatelé nespoléhají pouze na názvy či ikony, ale často na to, kde se konkrétní prvek nachází – ať už na ploše mobilní aplikace, v menu, nebo v rámci nástrojového panelu. Uživatelé se tedy v rozhraní orientují nejen podle textového obsahu, ale podle jeho fyzické polohy, což je behaviorální jev s kořeny v kognitivní psychologii.

Typickým příkladem může být pozice tlačítek jako „Zpět“ či „Uložit“ – pokud jsou tato tlačítka náhle přesunuta, uživatelé mohou nechtěně spustit nesprávnou akci. Tento problém je zvláště patrný v případech, kdy se očekává rychlá interakce, např. při snímání fotografií nebo ukládání záznamu. Změna rozvržení tak může zásadně narušit plynulost práce a způsobit chyby, které jsou důsledkem nesouladu s naučenými návyky.

Zachování stabilního rozložení prvků je proto klíčové, obzvlášť u aplikací, které se často používají. Vhodné je také umožnit uživateli vlastní organizaci rozhraní – například přetažením ikon nebo volitelným umístěním nástrojových panelů. Tím se podpoří aktivní tvorba prostorové paměti, čímž se zvyšuje pravděpodobnost, že uživatel najde to, co hledá, bez zaváhání.

Tento princip má hluboký základ v experimentální psychologii – výzkum ukazuje, že prostorové uspořádání informací zlepšuje pracovní paměť a přístup k opakovaně používaným informacím [13].

3.5 Behaviorální vzory

Návrh mobilní aplikace by měl zohledňovat opakující se vzorce chování uživatelů, tedy tzv. „behaviorální vzory“. Tyto vzorce vycházejí z předchozích zkušeností uživatelů s jinými aplikacemi a platformami a vytvářejí určitá očekávání, která pokud jsou naplněna, vedou k intuitivnímu a efektivnímu používání. Příkladem je umístění hlavní navigace v dolní části obrazovky ve formě „bottom navigation bar“ nebo využití tzv. „hamburger menu“ pro skrytí sekundárních funkcí.

Důležitou roli hraje i kontext použití – uživatelé často aplikaci používají na cestách, v rychlosti nebo pouze jednou rukou. Návrh by měl reflektovat tento způsob interakce například tím, že hlavní ovládací prvky budou umístěny v dosahu palce a že se aplikace přizpůsobí různým rozlišením obrazovky. Tomuto přístupu se říká adaptivní návrh a jeho cílem je zajistit konzistentní uživatelský zážitek napříč zařízeními a situacemi.

Z praxe je známo, že uživatelé dávají přednost aplikacím, které se přizpůsobí jejich návykům – například zobrazí naposledy použitou funkci na prominentním místě nebo automaticky nabídnou relevantní obsah podle času a místa. Tento typ návrhu může být podpořen i analytickými nástroji, které sledují uživatelské chování a umožňují aplikaci dynamicky přizpůsobit své rozhraní podle nejčastějších vzorců používání.

Behaviorální design se tak stává mostem mezi technickým návrhem a psychologií uživatele. Pokud návrháři dokážou tyto vzorce správně identifikovat a promítnout je do struktury aplikace, výrazně tím zvyšují pravděpodobnost úspěchu svého produktu [27].

3.6 Progresivní odhalování

Jedním z častých omylů v návrhu aplikací je snaha zobrazit veškeré funkce a informace najednou, čímž však dochází k přetížení uživatelovy kognitivní kapacity. Uživatelé chtějí mít k dispozici bohatou funkcionalitu, avšak bez nutnosti se s ní konfrontovat okamžitě. Zde přichází na řadu princip skryté komplexity a progresivního odhalování, který umožňuje zobrazovat informace a nástroje pouze tehdy, když jsou skutečně potřebné. Celá tato sekce vychází z práce Saffera [22].

V mobilních aplikacích je tento princip zvlášť důležitý kvůli omezenému prostoru. Příkladem může být formulář, který nejprve zobrazí jen základní pole, a po kliknutí na volbu „Více možností“ odhalí rozšířené nastavení. Tímto způsobem se snižuje kognitivní zátěž a zvyšuje se uživatelská přívětivost. Také mobilní fotoaplikace často využívají tohoto principu – základní funkce jako focení nebo nahrávání jsou ihned dostupné, zatímco pokročilé nástroje jako úprava expozice či RAW export jsou skryty v podmenu.

Tento přístup navíc podporuje tzv. „objevování rozhraní“ – uživatel se cítí kompetentní, protože sám odhaluje další funkce a možnosti, čímž se zvyšuje jeho angažovanost a spokojenost. Návrháři by měli klást důraz na to, aby bylo vždy jasné, že rozšířené možnosti existují, ale jejich přístupnost nebyla nátlaková. Důležitým aspektem je vizuální design: rozhraní musí naznačovat interaktivitu (např. pomocí šipek, záložek či ikon), ale zároveň nepůsobit zahlceně.

Princip progresivního odhalování se opírá o poznatky kognitivní psychologie a návrhových teorií, které ukazují, že minimalizace jednorázového zatížení pracovní paměti vede k efektivnějšímu používání systému [24].

3.7 Mikrointerakce a zpětná vazba

Podle Saffera [22] představují mikrointerakce drobné, ale významné detaily interakce mezi uživatelem a aplikací. Typicky se jedná o krátké animace, zvuky, změny stavu tlačítek, přechody mezi obrazovkami či drobná upozornění, která informují uživatele o tom, že jeho akce byla zaznamenána nebo dokončena. Jejich hlavní funkcí je poskytovat okamžitou a přirozenou zpětnou vazbu, díky níž uživatel rozumí stavu systému a výsledku své akce.

V mobilních aplikacích je tento prvek obzvláště důležitý, neboť dotykové rozhraní neumožňuje takovou přesnost jako klasická myš. Například vizuální reakce tlačítka při dotyku (např. změna barvy, stínu nebo vibrace) informuje uživatele, že jeho gesto bylo zaznamenáno. Podobně může být průběh nahrávání doprovázen animací kruhového indikátoru, nebo po úspěšném odeslání formuláře se může zobrazit zelené zaškrtnutí. Takové drobnosti nejenže zvyšují důvěru uživatele v systém, ale také podporují vnímání kvality a profesionality celé aplikace.

Dobře navržené mikrointerakce posilují orientaci a zabraňují chybám – například při přetažení položky může uživatel vidět, kam bude přesunuta, dříve než akci dokončí. Také pomáhají vytvářet „osobnost“ aplikace, zejména pokud jsou navrženy konzistentně a v souladu s brandem. Důležité je ale zachovat jejich střídmost – přespříliš animací nebo zpětné vazby může vést k zahlcení a zpoždění.

Význam mikrointerakcí je podložen i výzkumy v oblasti uživatelského prožitku, které ukazují, že aplikace s promyšlenou zpětnou vazbou jsou vnímány jako rychlejší, spolehlivější a příjemnější k použití.

Z hlediska návrhu by mikrointerakce měly být součástí počáteční fáze designového procesu, nikoliv až dodatečný prvek. Jejich role v PWA aplikacích je rovněž zásadní – zajiš-

ťují konzistenci chování napříč platformami a podporují rychlé pochopení stavu aplikace i při proměnlivém síťovém připojení nebo různém výkonu zařízení.

Kapitola 4

Průzkum a výběr implementačních technologií

Cílem této kapitoly je systematicky představit a analyzovat technologie, které přicházely v úvahu při vývoji aplikace, a zdůvodnit výběr konkrétních nástrojů použitých v rámci implementace. V zhodnocení jsou zohledněna kritéria jako multiplatformnost, vývojová efektivita, dostupnost dokumentace a kompatibilita s dalšími komponentami systému. Pozornost je věnována zejména výběru JavaScriptových frameworků pro vývoj uživatelského rozhraní, stylovacích metod (CSS frameworky), podpůrným knihovnám pro interakci s uživatelem a nástrojům pro automatické formátování zdrojového kódu. V jednotlivých podkapitolách jsou stručně popsány dostupné alternativy, jejich výhody a nevýhody, a na závěr každé části je uveden odůvodněný výběr řešení použitého v této práci.

4.1 JavaScriptové frameworky

JavaScript je dnes základním jazykem pro tvorbu interaktivních webových i hybridních mobilních aplikací. V průběhu posledního desetiletí se kolem něj vyvinula řada výkonných frameworků a knihoven, které vývojářům umožňují rychle a efektivně budovat komplexní uživatelská rozhraní. Mezi nejvýznamnější frameworky patří React, Angular, Vue.js, Svelte a další. Tyto nástroje poskytují strukturu a sadu nástrojů pro správu komponent, stavů, routingu a optimalizaci výkonu [8].

Frameworky se navzájem liší způsobem deklarace komponent, přístupem ke správě dat a mírou abstrakce. Důležitým kritériem pro výběr frameworku je také velikost vývojářské komunity, dostupnost nástrojů třetích stran a kompatibilita s dalšími technologiemi jako je například TypeScript. Rozhodování mezi jednotlivými možnostmi je proto nutné založit na konkrétním účelu aplikace, cílové platformě, požadavcích na výkon a zkušenostech vývojářského týmu.

React Native

React Native je framework vyvinutý společností Meta (dříve Facebook), který umožňuje vytvářet nativní mobilní aplikace pomocí JavaScriptu a Reactu. Na rozdíl od klasického Reactu, který slouží pro vývoj webových aplikací, React Native kompiluje komponenty do nativních prvků dané platformy (Android a iOS). Tento přístup nabízí výhodu sdílení velké části kódu mezi platformami, což zrychluje vývoj a snižuje náklady [6].

React Native rovněž poskytuje přístup k nativním funkcím zařízení, včetně fotoaparátu, přes knihovny jako `react-native-camera` a modernější `react-native-vision-camera`. Ty umožňují snímání, záznam videa a detekci obličejů. Integrace těchto komponent vyžaduje určitou úroveň znalostí nativních modulů. Výsledkem je však plynulý a výkonný přístup k fotoaparátu, téměř srovnatelný s čistě nativním vývojem. To činí React Native vhodným pro aplikace, které vyžadují interakci s obrazovým vstupem, jako jsou skenovací, biometrické nebo dokumentační nástroje.

Výhody:

- Sdílení kódu napříč platformami (Android/iOS)
- Rychlý vývoj díky hot-reloadingu
- Bohatá knihovna komponent a integrace s nativními API
- Silná komunita a podpora velkých projektů

Nevýhody:

- Výkonnostní limity u graficky náročných aplikací
- Závislost na bridge mezi JavaScriptem a nativním kódem
- Komplexita při integraci vlastního nativního kódu
- Potenciální problémy při aktualizaci verzí a knihoven

PWA – progresivní webové aplikace

Progresivní webové aplikace představují moderní přístup ke tvorbě webových aplikací, které se chovají podobně jako aplikace nativní. Využívají standardní webové technologie a rozšiřují je o pokročilé funkce díky Service Workerům, manifestu aplikace a offline režimu. Hlavním cílem PWA je přinést nativní zážitek z aplikace uživatelům přímo v prohlížeči bez nutnosti instalace z obchodů [21].

V oblasti práce s fotoaparátem je PWA v posledních letech výrazně pokročilá díky rozhraní `MediaDevices.getUserMedia`, které umožňuje přímý přístup ke kameře zařízení prostřednictvím webového prohlížeče. Pomocí tohoto API lze snadno implementovat základní funkce jako pořízení snímku, náhled videa v reálném čase nebo přepínání mezi přední a zadní kamerou. I když zde nejsou dostupné všechny funkce jako u nativních řešení (např. RAW režimy, rozšířená detekce), pro účely základního snímání je PWA dostatečně výkonná a zároveň jednodušší na implementaci a nasazení.

Výhody:

- Není třeba instalace z obchodů
- Multiplatformní fungování na desktopu i mobilu
- Jednodušší aktualizace a nasazení
- Možnost práce offline

Nevýhody:

- Omezený přístup k některým nativním funkcím jako je třeba Bluetooth nebo NFC
- Závislost na schopnostech a omezeních prohlížeče
- Horší integrace s operačním systémem
- Občasná nestabilita v závislosti na platformě nebo verzi prohlížeče

Mezi výhody PWA patří snadná dostupnost, multiplatformnost (funkční na desktopu i mobilu), rychlé nasazení aktualizací a nižší náklady na vývoj a údržbu. Navíc podporují přidání na domovskou obrazovku a mohou běžet offline. Nevýhodou může být omezený přístup k některým nativním funkcím (např. Bluetooth, NFC), horší integrace s operačním systémem a občasná nestabilita kvůli závislosti na prohlížeči.

Další alternativní technologie

Mezi další často zvažované JavaScriptové frameworky pro vývoj frontendových řešení patří například Angular, který nabízí robustní architekturu a striktní typový systém díky integraci s TypeScriptem, nebo Vue.js, který je známý svou jednoduchostí a rychlostí učení. Svelte představuje moderní přístup s přímou kompilací do efektivního kódu bez běhového prostředí, což snižuje velikost aplikací.

Konečný výběr

Přestože všechny technologie mají své výhody, pro účely této práce – tj. vývoj lehké mobilní aplikace pro snímání hřbetu ruky a její provoz i bez nativní instalace – byla zvolena technologie PWA, která nejlépe odpovídá požadavkům na multiplatformnost, nízkou zátěž a flexibilní nasazení.

4.2 Frameworky pro kaskádové styly

Způsob definování vzhledu uživatelského rozhraní je zásadní součástí návrhu frontendové architektury. Použité technologie určují nejen vzhled, ale také rychlost vývoje, čitelnost kódu a možnosti údržby. V této části jsou představeny tři nejběžnější přístupy – klasické CSS, utility-first přístup reprezentovaný Tailwind CSS a komponentově orientovaný přístup využívaný v rámci Bootstrapu. Pro jejich porovnání a výběr vhodného řešení byly zohledněny jak praktické aspekty, tak i doporučení odborné literatury [9].

Klasické CSS

Přístup využívající klasické CSS je jedním z nejtradičnějších způsobů stylování webových rozhraní. Vývojář definuje vzhled komponent přímo pomocí kaskádových stylů, buď externě v samostatných souborech, nebo přímo v hlavičce dokumentu. Tento přístup poskytuje úplnou kontrolu nad vzhledem a nevyžaduje žádné externí knihovny či frameworky.

Výhody:

- Maximální flexibilita

- Žádná externí závislost
- Možnost optimalizace velikosti stylů

Nevýhody:

- Při větším projektu může být složité udržet konzistenci
- Vyžaduje manuální správu přepisů a specifity

Tailwind

Tailwind CSS představuje moderní stylovací framework založený na principu utility-first. Namísto psaní vlastních selektorů a pravidel vývojář využívá již připravené třídy, které se zapisují přímo do HTML. Tento přístup podporuje konzistenci a efektivitu při tvorbě uživatelských rozhraní – a to i v případě rozsáhlejších projektů.

Výhody:

- Utility-first přístup, kde se styly zapisují přímo do tříd v HTML
- Podporuje responzivitu, tmavý režim i komponentové návrhy
- Rychlý vývoj bez nutnosti psát vlastní CSS
- Konzistentní design skrze celé rozhraní

Nevýhody:

- Omezená expresivita při složitějším vizuálním designu
- Vyšší vstupní náročnost pro začátečníky

Bootstrap

Bootstrap je rozsáhlý CSS framework, který obsahuje předdefinované komponenty, mřížky a responzivní nástroje. Díky své jednoduchosti a komplexní dokumentaci patří mezi nejpoužívanější nástroje při tvorbě webových prototypů i produkčních aplikací.

Výhody:

- Rychlý začátek bez nutnosti tvořit vlastní komponenty
- Rozsáhlá dokumentace a komunita

Nevýhody:

- Výrazný „bootstrapový vzhled“ bez výrazné úpravy
- Méně flexibilní pro specifický design

Konečný výběr

Pro účely této práce, kde je požadována jednoduchost, kontrola a plná přizpůsobitelnost vzhledu, byla zvolena varianta nativního CSS. Publikace jako například *CSS in Depth* od K. J. Granta [9] potvrzují, že pro menší a středně velké projekty je ručně psané CSS efektivní a dobře udržovatelné řešení.

4.3 Výběr knihoven

Součástí návrhu uživatelského rozhraní je také zajištění snadné orientace uživatele při prvním použití aplikace. K tomu slouží knihovny pro interaktivní průvodce, které umožňují krok po kroku vysvětlit funkce aplikace, zvýraznit klíčové prvky a poskytovat vizuální nápovědu.

Intro.js

Intro.js [10] je jednoduchá a lehká knihovna pro vytváření interaktivních návodů, která umožňuje zvýraznit jednotlivé prvky na stránce a doplnit je o popis. Její implementace nevyžaduje závislost na žádném frameworku, což ji činí vhodnou také pro projekty založené čistě na JavaScriptu nebo PWA.

Výhody:

- Nízká velikost balíčku a snadná integrace
- Možnost přizpůsobit vzhled i pozice pomocí CSS

Nevýhody:

- Omezené možnosti pokročilé logiky nebo interakce

Shepherd.js

Shepherd.js [23] je robustnější alternativou ke knihovně Intro.js. Poskytuje rozsáhlé možnosti pro tvorbu interaktivních průvodců s podporou asynchronních kroků, přizpůsobení pozice i vzhledu jednotlivých kroků a integraci s moderními frontendovými frameworky. Díky větší přizpůsobitelnosti je vhodnější pro komplexní aplikace nebo dynamický obsah, který se mění v reálném čase.

Výhody:

- Podpora asynchronních kroků a složitějších toků
- Vysoká míra přizpůsobení vzhledu

Nevýhody:

- Vyšší komplexita implementace
- Vyžaduje více nastavení pro základní funkčnost

Konečný výběr

Na základě porovnání funkcionality, složitosti implementace a kompatibility s PWA byla zvolena knihovna Intro.js. Řešení splňuje požadavky na přívětivé a intuitivní vedení uživatele aplikací bez výrazné zátěže na výkon či složitou integraci. Intro.js umožňuje rychlou implementaci vizuálních nápověd přímo v prostředí aplikace.

Knihovna se osvědčila jako stabilní, dobře zdokumentovaná a podporovaná komunitou. Vzhledem k tomu, že aplikace nevyžaduje komplexní dynamický onboarding, nebylo nutné sahat po robustnějších alternativách, jako je Shepherd.js.

Toastify

Toastify.js [11] je jednoduchá a lehká JavaScriptová knihovna pro zobrazování toast notifikací. Nevyžaduje žádný framework a funguje čistě s vanilla JavaScriptem, což ji činí vhodnou i pro PWA aplikace bez závislosti na větším ekosystému.

Výhody:

- Jednoduchá implementace a API
- Možnost přizpůsobit pozici, délku trvání, barvu a obsah notifikace

Nevýhody:

- Omezené možnosti přizpůsobení animací a interakcí
- Nepodporuje složitější typy hlášení (např. fronty, potvrzovací dialogy)

Notyf

Notyf [4] je moderní knihovna pro toast notifikace, která nabízí čistý design a přístupnější API. Oproti Toastify umožňuje jednodušeji definovat typy zpráv (např. „success“ nebo „error“), využívá minimalistický vzhled a podporuje WCAG přístupnost.

Výhody:

- Předdefinované styly pro různé typy zpráv
- Přístupnost a podpora screenreaderů
- Možnost rozšíření o vlastní šablony a callbacky

Nevýhody:

- Větší velikost oproti Toastify
- Méně přizpůsobení bez zásahu do zdrojového kódu

Konečný výběr

Toastify.js byl v této práci zvolen kvůli své jednoduchosti, minimální velikosti a snadné integraci s komponentami React a PWA. Pro potřeby základních informačních hlášení je jeho funkcionalita plně dostačující.

4.4 Algoritmy pro hodnocení kvality snímku

BRISQUE a Laplacian jsou dva rozšířené přístupy pro posouzení kvality obrazových dat bez nutnosti referenčního snímku. V následujícíh sekcí jsou algoritmy vysvětleny podrobněji.

Brisque

Algoritmus BRISQUE [15] představuje metodu pro objektivní hodnocení kvality obrazu, která nevyžaduje referenční snímek. Je založen na analýze statistických vlastností obrazu v prostorové doméně, přičemž se vyhodnocuje, nakolik se daný snímek odchyluje od charakteristických rysů přirozených obrazů.

BRISQUE modeluje tzv. „natural scene statistics“, které jsou typické pro kvalitní fotografie bez rušivých jevů, jako je nadměrný šum, kompresní artefakty nebo rozmazání. Na základě těchto charakteristik algoritmus vypočítá příznakový vektor, který je předložen předtrénovanému regresnímu modelu. Výsledkem je číslo reprezentující subjektivně vnímanou kvalitu. Nižší hodnoty znamenají lepší obraz, vyšší hodnoty značí degradačni efekty.

V kontextu aplikací se BRISQUE uplatňuje jako filtr, který automaticky vyřazuje snímky, jejichž kvalita nedosahuje stanoveného prahu. To pomáhá získat homogenní a vizuálně spolehlivé vstupy pro další zpracování nebo analýzu.

Laplacian

Laplaciánová metoda [18] je jednoduchý a efektivní algoritmus pro posouzení ostrosti obrazů. Založený je na Laplaciánově operátoru, který detekuje hranice a změny intenzit v obrazovém poli.

Princip metody spočívá v tom, že ostré obrazy obsahují mnoho jemných detailů a struktur, což se projevuje vyšším rozptylem (variancí) Laplaciánovy transformace. Naopak rozmazané snímky mají malý počet řídkých účelných hran a jejich rozptyl je tedy nižší. Tento parametr se následně porovnává s nastavenou prahovou hodnotou.

Laplacián se často používá v reálných systémech, kde je potřeba rychle rozlišit vhodné a nevhodné obrazové vstupy. Vzhledem k jeho nenáročnosti je ideální pro systémy s omezeným výkonem nebo tam, kde je požadována okamžitá zpětná vazba uživateli.

4.5 MediaPipe

MediaPipe je open-source framework vyvinutý společností Google, který slouží k efektivní detekci a sledování tělových struktur v reálném čase. Je navržen pro snadnou integraci do mobilních i webových aplikací a podporuje multiplatformní nasazení.

V oblasti zpracování obrazu nabízí MediaPipe hotové moduly pro rozpoznávání obličejů, sledování očí, detekci lidských postav nebo právě rukou. V případě aplikace Capture My Hand je využit modul „Hand Tracking“, který zajišťuje detekci dlaně a výpočet tzv. landmarků – bodů odpovídajících kloubům a prstům.

MediaPipe tak poskytuje informace o poloze a orientaci ruky na snímku, což je klíčové pro ověření, zda je pořízená fotografie relevantní. Systém dokáže v reálném čase reagovat a upozornit uživatele, pokud ruka není zachycena kompletně nebo je ve špatném postavení.

Díky přesnosti, rychlosti a dostupnosti na různých platformách se MediaPipe stalo často voleným nástrojem v projektech zaměřených na interakci člověka s počítačem prostřednictvím kamerového záznamu [14].

Prettier

Prettier [19] je nástroj pro automatické formátování zdrojového kódu, který výrazně zvyšuje konzistenci a čitelnost při práci v týmu. Pomáhá eliminovat stylistické rozdíly mezi jednotlivými vývojáři tím, že při uložení automaticky přeformátuje kód podle předem definovaných pravidel. Podporuje širokou škálu jazyků, včetně JavaScriptu, TypeScriptu, HTML, CSS a JSON, a je široce využíván v moderních frontendových projektech.

Použití Prettieru v projektu přináší několik výhod. Předně zajišťuje jednotný styl napříč všemi soubory bez nutnosti ruční úpravy. To je klíčové zejména při spolupráci více vývojářů na jednom repozitáři. Dále eliminuje zbytečné code-review komentáře týkající se stylu a umožňuje zaměřit se na logiku a funkčnost kódu. Prettier je obvykle integrován do vývojového prostředí (např. VS Code) nebo jako součást CI procesu.

Konfigurace Prettieru je možná prostřednictvím `.prettierrc` souboru nebo příkazové řádky. Možné je definovat např. délku řádku, použití uvozovek, typ odsazení, zachování či odstranění středníků aj. V kombinaci s ESLint lze Prettier používat pro komplexní kontrolu jak formátu, tak i kvality kódu.

V této práci byl Prettier použit pro automatizované formátování kódu na frontendové části aplikace, čímž byla zajištěna konzistentní struktura napříč komponentami a lepší přehlednost celého projektu.

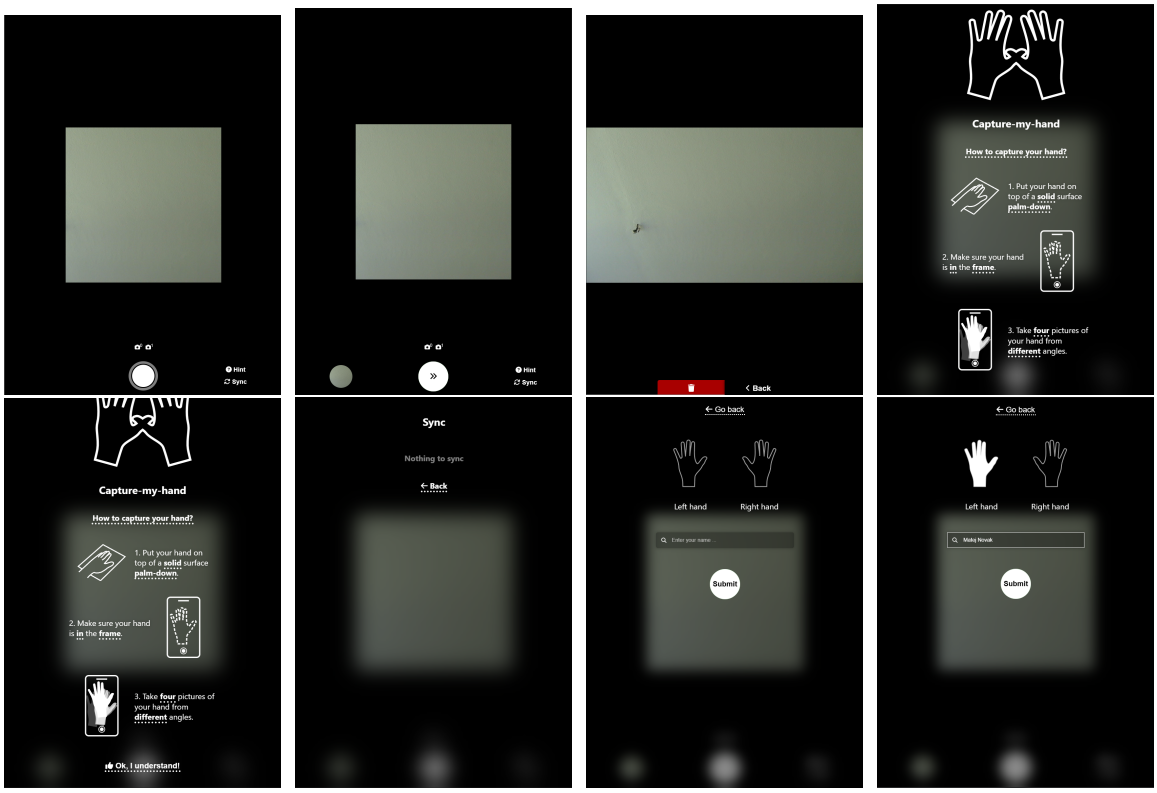
Kapitola 5

Grafické rozhraní

Cílem této práce byla úprava již existujícího řešení – aplikace Capture My Hand, kterou vytvořil Ing. Marek Vašek. Prvotním krokem bylo detailní seznámení se s vizuální identitou aplikace, včetně použitých barev a grafických prvků. Grafický styl je velmi minimalistický a opírá se o kontrast černé a bílé barvy. Pozadí aplikace je černé, zatímco texty, piktogramy, ikonky a ovládací prvky jsou zobrazeny bílým odstínem.

Celkově je rozhraní koncipováno velmi jednoduše. Skládá se z úvodní obrazovky a následujícího formuláře pro zadání metadat k pořízeným fotografiím.

Vzhled původní aplikace je demonstrován na obrázku 5.1.



Obrázek 5.1: Ukázky obrazovek uživatelského rozhraní aplikace Capture My Hand: pořizování fotografií, galerie, nápověda, sync a formulář pro zadání metadat.

Při návrhu byly uplatněny principy intuitivnosti a tzv. „self-explaining“ přístupu, tedy aby se uživatel v aplikaci zorientoval bez nutnosti externí nápovědy. Zásadní roli zde hraje první interakce uživatele s aplikací, která by měla být srozumitelná, přehledná a přívětivá.

Za účelem podpory uživatelské přívětivosti byla nasazena knihovna Intro.js, která slouží k vizuálnímu provedení uživatele rozhraním aplikace. Dále byla odstraněna původní záložka `Sync`, která zobrazovala stav odesílání fotografií, a tato funkcionalita byla nahrazena tzv. „toast“ notifikacemi za pomoci knihovny Toastify.

5.1 Intro.js

Knihovna Intro.js byla implementována jako nástroj pro interaktivní průvodce, jehož účelem je usnadnit uživateli první orientaci v aplikaci. Aktivace probíhá pomocí tlačítka označeného jako `Intro` na úvodní obrazovce. Po spuštění je uživatel proveden jednotlivými klíčovými prvky uživatelského rozhraní:

- kamera
- tlačítko spouště fotoaparátu
- informace o počtu aktuálně pořízených fotografií
- galerie s fotografiemi
- tlačítko `Guide` pro zobrazení nápovědy
- opakování celého průvodce dle potřeby

Podobný průvodce byl přidán také do formuláře pro vyplňování metadat. Uživatel je zde instruován, jaké pole je nutné vyplnit a jaké volby zvolit:

- výběr typu ruky (pravá/levá)
- pole pro zadání jména
- tlačítko pro potvrzení a odeslání formuláře

Díky knihovně Intro.js je aplikace pro nového uživatele přístupnější a příjemnější, což přispívá k pozitivnímu uživatelskému zážitku při prvním použití. Interaktivní nápověda je dostupná i později, například v případě, že se uživatel potřebuje k nějaké informaci vrátit.

5.2 Toastify

Knihovna Toastify byla implementována jako náhrada za původní zobrazení stavů v záložce `Sync`. Jedná se o elegantní formu notifikací, které se zobrazují v horní části obrazovky jako plovoucí informační prvky. Aktivují se automaticky na základě změny stavu při komunikaci s backendem aplikace.

Toastify notifikace byly také využity v kontextu formuláře pro zadávání metadat. Pokud uživatel nevyplní povinné položky, je o této skutečnosti informován právě touto formou. Povinnými prvky jsou:

- výběr typu ruky

- vyplnění jména uživatele

Po úspěšném nebo neúspěšném odeslání formuláře je uživateli zobrazena notifikace s informací o průběhu akce:

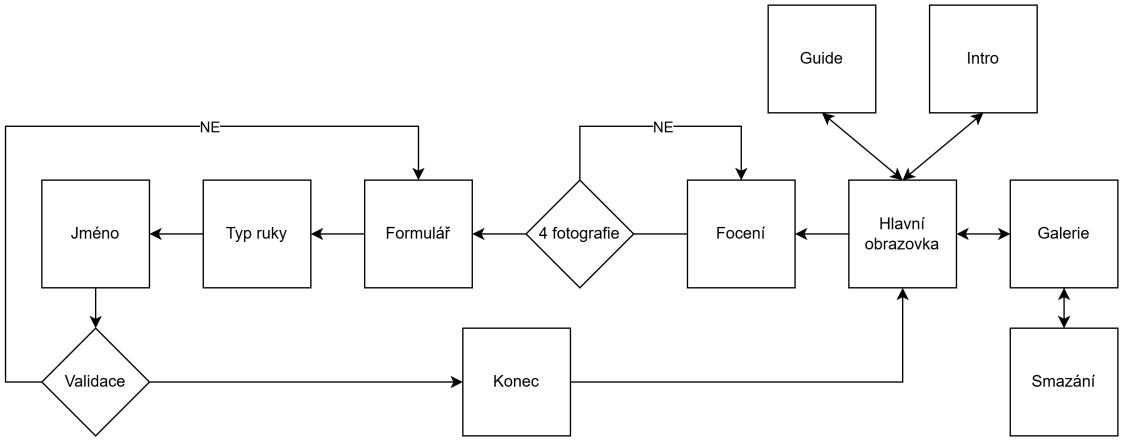
- Nahrání bylo úspěšné
- Nahrání se nezdařilo

Díky tomuto řešení byla zachována jednoduchost rozhraní a uživatel je informován rychle a přehledně, bez nutnosti přecházet do jiné záložky nebo sekce aplikace.

5.3 Pracovní postup uživatele

Pro znázornění postupu, který uživatel v aplikaci prochází, byl vytvořen následující diagram 5.2. Ten zachycuje jednotlivé kroky od spuštění aplikace, pořizování fotografií a jejich odeslání, až po kontrolu vstupů ve formuláři. Diagram ukazuje logickou posloupnost mezi obrazovkami a možnými odbočkami v případě nevyplněných nebo nesprávných vstupů.

Tento diagram slouží jako podpůrný materiál pro lepší pochopení logiky aplikace a poskytuje ucelený pohled na interakci uživatele s jednotlivými komponentami aplikace.



Obrázek 5.2: Diagram znázornující interakci uživatele s aplikací Capture My Hand.

Po úspěšném pořízení všech čtyř požadovaných fotografií je uživatel automaticky přesměrován na formulář pro zadání metadat. Zde musí zvolit typ ruky (pravá nebo levá) a vyplnit své jméno. Pokud některý z údajů chybí, aplikace zobrazí upozornění prostřednictvím toast notifikace a neumožní pokračování. Po správném vyplnění všech polí jsou data odeslána do systému. Proces zároveň počítá s možností návratu zpět nebo opakovaného pořízení snímků, čímž podporuje flexibilitu a redukuje riziko uživatelských chyb.

Kapitola 6

Implementace

Tato kapitola podrobně popisuje způsob technické realizace aplikace *Capture My Hand*, a to jak z hlediska návrhu architektury frontendové části, tak i z pohledu implementace backendu a přenosu dat mezi oběma vrstvami systému. Cílem je představit strukturu zdrojového kódu, použité technologie, knihovny a komponenty, stejně jako konkrétní technické řešení klíčových funkcionalit aplikace. Důraz je kladen na modularitu, přehlednost kódu, použití moderních vývojových přístupů a respektování zásad dobré softwarové praxe. Součástí kapitoly je rovněž vysvětlení zvolených přístupů ke zpracování obrazových dat, validaci vstupů, práci s kamerou v rámci webového prostředí a využití podpůrných knihoven pro zvýšení použitelnosti aplikace. Na závěr jsou stručně popsány vybrané komponenty a logika jejich propojení v rámci celého systému.

6.1 Vývojový design aplikace

Aplikace je navržena jako jednostránková webová aplikace (SPA) s využitím knihovny React. Uživatel pomocí aplikace pořídí fotografie hřbetu ruky, které jsou následně zasílány skrze jednoduchý formulář. Vývojový design klade důraz na modularitu, přehlednost a jednoduchou správu komponent.

Architektura komponent a správa stavu

Každá část aplikace je realizována jako samostatná funkcionální komponenta. Komponenty jako `Gallery`, `SubmissionForm`, `Hint` nebo `ButtonCaptureContinue` spolupracují prostřednictvím lokálního stavu. K přenosu informací mezi komponentami slouží vlastnosti (*props*) a sdílený stav mezi rodičovskými a dceřinými prvky. Díky tomu není potřeba centralizovaná knihovna jako Redux.

Soubor `App.js` slouží jako hlavní rozcestník celé aplikace – definuje tok aplikace a propojuje jednotlivé komponenty.

React Hooks

React Hooks [1] jsou funkce, které umožňují používat stav a další funkce Reactu v rámci funkcionálních komponent. Hlavní výhodou je eliminace potřeby třídových komponent, což zjednodušuje čitelnost a opakovatelnost kódu.

useState

Hook `useState` slouží k uchovávání lokálního stavu uvnitř komponenty. Umožňuje vytvářet proměnné, jejichž hodnota se může během života komponenty měnit – např. stav formuláře, aktuálně pořízená fotografie nebo indikátor nahrávání. V aplikaci *Capture My Hand* se používá v komponentách jako `Gallery` nebo `SubmissionForm`, kde uchovává data potřebná k řízení toku aplikace. Použití `useState` zvyšuje flexibilitu komponent a zajišťuje, že každá komponenta může pracovat se svými specifickými daty nezávisle na ostatních [3].

useEffect

Hook `useEffect` se používá pro vykonávání vedlejších efektů – tedy operací, které nesouvisí přímo s vykreslováním komponenty, ale souvisejí s jejími změnami. Příkladem je načítání dat, naslouchání událostem nebo práce s časovači. V aplikaci slouží `useEffect` např. ke kontrole platnosti formuláře nebo ke sledování změn v uživatelských interakcích. Každý `useEffect` lze omezit jen na konkrétní změny sledovaných hodnot, což zlepšuje výkon a předchází zbytečným přerenderováním [2].

Adresářová a souborová struktura projektu

Projekt nemá hlubší adresářovou strukturu – všechny soubory jsou umístěny přímo ve složce `src`. Tento přístup odpovídá jednoduchosti aplikace a malému rozsahu. Mezi klíčové soubory patří:

- `App.js` – centrální komponenta, která spojuje logiku aplikace.
- `Gallery.js`, `Hint.js`, `SubmissionForm.js` – komponenty pro jednotlivé obrazovky.
- `*.css` – každá komponenta má svůj vlastní soubor se styly.
- `serviceWorkerRegistration.js` – zajištění podpory pro režim PWA.

Tento způsob organizace je vhodný pro malé projekty. Pro větší projekty by bylo vhodné komponenty rozdělit do podsložek podle typu či účelu (např. složka `components`, `hooks`, `utils`).

6.2 Technologie a způsob fungování backendu

Backendová část aplikace je postavena na webovém frameworku FastAPI, který umožňuje rychlý vývoj moderního REST API v jazyce Python. Tento backend zajišťuje příjem dat z frontendu a jejich uložení či další zpracování.

REST API

Aplikace komunikuje s klientskou částí pomocí REST API. Jedná se o rozhraní založené na standardních HTTP metodách (`GET`, `POST`, `PUT`, `DELETE`), které umožňují operace s prostředky uloženými na serveru. V případě této aplikace jsou to zejména obrazová data (fotografie ruky), která jsou přijímána a uchovávána pro další zpracování nebo analýzu.

REST API dodržuje základní architektonické principy, jako je bezstavovost, oddělení klienta a serveru nebo jednotné rozhraní. FastAPI navíc umožňuje snadné připojení dokumentace pomocí Swaggeru [7].

Struktura backendu a adresáře

Projekt obsahuje hlavní soubor `app.py`, který slouží jako vstupní bod API serveru. Dále je zde složka `capture_my_hand_backend`, která uchovává inicializační skript a další konfigurační soubory jsou umístěny v rootu (např. `config.py`, `Dockerfile`, `compose.yaml`). Backend tak lze snadno kontejnerizovat pomocí Dockeru a nasadit jako samostatnou službu.

- `app.py` – hlavní definice endpointů a nastavení serveru.
- `config.py` – konfigurační proměnné (např. cesty, adresy).
- `capture_my_hand_backend/__init__.py` – základní inicializace aplikace jako balíčku.
- `Dockerfile, compose.yaml` – soubory pro kontejnerizaci a deployment.

Formát JSON

Formát JSON je lehký textový formát sloužící k výměně dat mezi klientem a serverem. Je založen na párech klíč–hodnota a podporuje vnořené struktury jako pole a objekty. V kontextu aplikace slouží JSON k přenosu dat mezi frontendem a backendem, zejména pro odesílání obrazových dat a metadat o uživateli.

```
{
"user_id": "abc123",
"image": "...",
"timestamp": "2025-04-30T15:34:00Z"
}
```

Výpis 6.1: Příklad jednoduchého JSON objektu přijatého backendem

JSON je široce podporovaný většinou programovacích jazyků a je ideální pro REST API, protože je čitelný pro lidi i stroje.

Swagger dokumentace (OpenAPI)

Swagger [25] je sada nástrojů postavených na specifikaci OpenAPI, která slouží k popisu a dokumentaci REST API. V rámci FastAPI je Swagger UI automaticky generován a přístupný na adrese `/docs`. Poskytuje přehled o všech dostupných endpointech, očekávaných parametrech, odpovědích a chybových stavech.

Hlavní výhody Swagger dokumentace:

- Automatická vizualizace struktury API bez nutnosti ruční dokumentace.
- Možnost testovat jednotlivé endpointy přímo z prohlížeče.
- Export specifikace ve formátu OpenAPI (JSON), který lze použít v dalších nástrojích.

Swagger dokumentace výrazně usnadňuje vývoj, ladění i integraci API s dalšími systémy.

Příklady OpenAPI specifikace (Swagger JSON)

Swagger dokumentace je založena na formátu OpenAPI ve struktuře JSON. Následující ukázky znázorňují části výstupu specifikace pro jednotlivé endpointy:

`/api/adduser/` – POST

```
"/api/adduser/": {
  "post": {
    "summary": "Add new user",
    "requestBody": {
      "content": {
        "application/json": {
          "schema": {
            "$ref": "#/components/schemas/UserData"
          }
        }
      },
      "required": true
    },
    "responses": {
      "200": { "description": "User created successfully" },
      "400": { "description": "Invalid input" }
    }
  }
}
```

Výpis 6.2: OpenAPI výpis pro /api/adduser/

`/api/uploadfiles/` – POST

```
"/api/uploadfiles/": {
  "post": {
    "summary": "Upload user hand images",
    "requestBody": {
      "content": {
        "multipart/form-data": {
          "schema": {
            "type": "object",
            "properties": {
              "files": {
                "type": "array",
                "items": { "type": "string", "format": "binary" }
              }
            }
          }
        }
      },
      "required": true
    },
    "responses": {
      "200": { "description": "Files uploaded" },
      "400": { "description": "Upload failed" }
    }
  }
}
```

Výpis 6.3: OpenAPI výpis pro /api/uploadfiles/

`/api/detect/` – POST

```
"/api/detect/": {
  "post": {
    "summary": "Detect hand landmarks",
    "requestBody": {
      "content": {
        "application/json": {
          "schema": {
            "$ref": "#/components/schemas/HandDetectionData"
          }
        }
      },
      "required": true
    },
    "responses": {
      "200": { "description": "Detection successful" },
      "503": { "description": "Model not available" }
    }
  }
}
```

Výpis 6.4: OpenAPI výpis pro /api/detect/

Postup zpracování fotografie

Validace výstupu z kamery je klíčovou součástí interakce uživatele s aplikací. Aplikace v reálném čase každých 300 ms pořizuje malé screenshoty z videozáznamu kamery a odesílá je na backendový endpoint `/api/detect`. Ten provádí serverovou detekci – typicky se jedná o kontrolu, zda je ve výřezu správně rozpoznána lidská ruka. Na základě odpovědi serveru se ve frontendové aplikaci dynamicky povoluje nebo zakazuje tlačítko pro pořízení fotografie. Tento mechanismus zajišťuje, že uživatel pořídí pouze snímek s dostatečnou kvalitou a správným obsahem.

Funkce `sendFrameToBackend` je volána právě s tímto časovým intervalem v rámci detekční smyčky. Vstupem této funkce je obraz ve formátu Base64, který je zabalen do objektu `FormData`. Následně jsou data odeslána metodou POST na uvedený endpoint. Pokud server vrátí pozitivní výsledek detekce, stav aplikace se aktualizuje a uživatel může aktivovat tlačítko fotoaparátu.

Uživatel pořídí celkem čtyři fotografie. Jakmile jsou všechny snímky zachyceny, aplikace přesměruje uživatele na formulářovou stránku, kde zadá doplňující metadata – své jméno a výběr levé či pravé ruky. Po potvrzení jsou všechny čtyři snímky spolu s metadaty odeslány metodou POST na endpoint `/api/uploadfiles`, kde dochází k jejich uložení a dalšímu zpracování na straně serveru.

Tento postup kombinuje důslednou kontrolu kvality snímků s intuitivním uživatelským rozhraním, čímž poskytuje efektivní a bezpečný proces sběru obrazových dat.

6.3 Stručný popis vybraných částí implementace

V této části jsou popsány vybrané komponenty uživatelského rozhraní aplikace, které zásadním způsobem ovlivňují její funkcionalitu a použitelnost. Popis zahrnuje průvodce Intro.js,

knihovnu pro notifikace Toastify, komponentu pro práci s kamerou `react-webcam` a formulářovou logiku pro odesílání dat na server. Uvedené části byly vybrány s ohledem na jejich význam pro celkový chod aplikace a uživatelský komfort.

Intro.js

Pro podporu uživatelské orientace v aplikaci je využita knihovna `Intro.js`, která umožňuje tvorbu interaktivních průvodců. Tato knihovna je použita v několika částech aplikace, zejména v hlavní komponentě a ve formuláři pro odeslání dat.

Na domovské obrazovce je průvodce dostupný po kliknutí na tlačítko s textem „Intro“. Tímto způsobem se spustí průvodce, který vysvětluje jednotlivé základní komponenty hlavní obrazovky – typicky náhled z kamery, ovládací tlačítka a další prvky rozhraní. Každý krok průvodce odkazuje na konkrétní prvek a pomocí textové bubliny vysvětluje jeho funkci. Díky tomu se uživatel snadno seznámí s prostředím aplikace ještě před samotným pořízením fotografií.

Druhé využití průvodce se nachází ve formulářové části aplikace, kde uživatel doplňuje metadata pořízených fotografií. Zde průvodce objasňuje, jak vyplnit jednotlivá pole – například jméno uživatele a výběr ruky (levá/pravá). Průvodce napomáhá uživateli k bezchybné a úplné specifikaci potřebných údajů před jejich odesláním na server.

Průvodce je konfigurován jako neinteraktivní pomocí parametru `disableInteraction: true`, což znamená, že uživatel během průvodce nemůže manipulovat s ostatními prvky rozhraní. Použitý styl je `introjs-modern`, který zajišťuje moderní vzhled nápovědních bublin.

Tímto způsobem poskytuje aplikace uživateli přehledný a srozumitelný návod, jak s aplikací pracovat, čímž se zvyšuje její použitelnost a minimalizuje riziko nesprávného použití jednotlivých funkcí.

Toastify

Pro informování uživatele o stavu operací (např. úspěšné odeslání dat, chyba při uploadu apod.) je v aplikaci využita knihovna `react-toastify`. Tato knihovna umožňuje zobrazování krátkých informačních zpráv – tzv. „toastů“ – v pravé dolní části obrazovky, a to bez nutnosti zásahu uživatele.

V projektu je `Toastify` využívána např. po úspěšném odeslání fotografií a metadat pomocí funkce `submitDataWithCallbacks`. V případě chyby během odesílání se zobrazí hlášení s možností opakovaného pokusu. Zpráva obsahuje i vlastní HTML strukturu s tlačítkem pro opětovné odeslání.

Navíc se knihovna používá při validaci formuláře pro přidávání metadat. Informuje uživatele, pokud nevyplní některé z polí, které jsou povinná.

Díky integraci knihovny `Toastify` je uživatel neustále informován o výsledku svých akcí bez rušivých dialogových oken, čímž se zvyšuje plynulost práce s aplikací a zároveň se podporuje lepší uživatelská zkušenost.

React-webcam

Pro zajištění přístupu ke kameře uživatele v prostředí webové aplikace je v projektu využita komponenta `webcam` z `react-webcam`. Tato komponenta poskytuje jednoduché rozhraní pro integraci kamery a umožňuje snadné pořizování fotografií přímo z webového prohlížeče.

Parametry a konfigurace

Parametr pro záznam zvuku je zakázán, protože aplikace nevyžaduje audiovýstup. Snímky jsou pořizovány ve formátu JPEG a kamera je konfigurována tak, aby bylo možné využít například zadní kameru mobilního zařízení. Konfigurace kamery je definována pomocí objektu, který určuje technické parametry jako je typ kamery nebo rozlišení.

Díky využití rozhraní `mediaDevices` je aplikace nastavena tak, aby automaticky zvolila hlavní zadní kameru zařízení. Toto řešení odpovídá požadavkům na přesnost snímků a eliminuje potřebu uživatelského výběru, čímž se zjednodušuje celé uživatelské rozhraní.

Komponenta využívá referenci, díky které lze kdykoliv získat aktuální snímek z videa. Tento snímek je následně uložen do stavu aplikace pro další použití.

Detekce ruky a kontrola kvality snímku

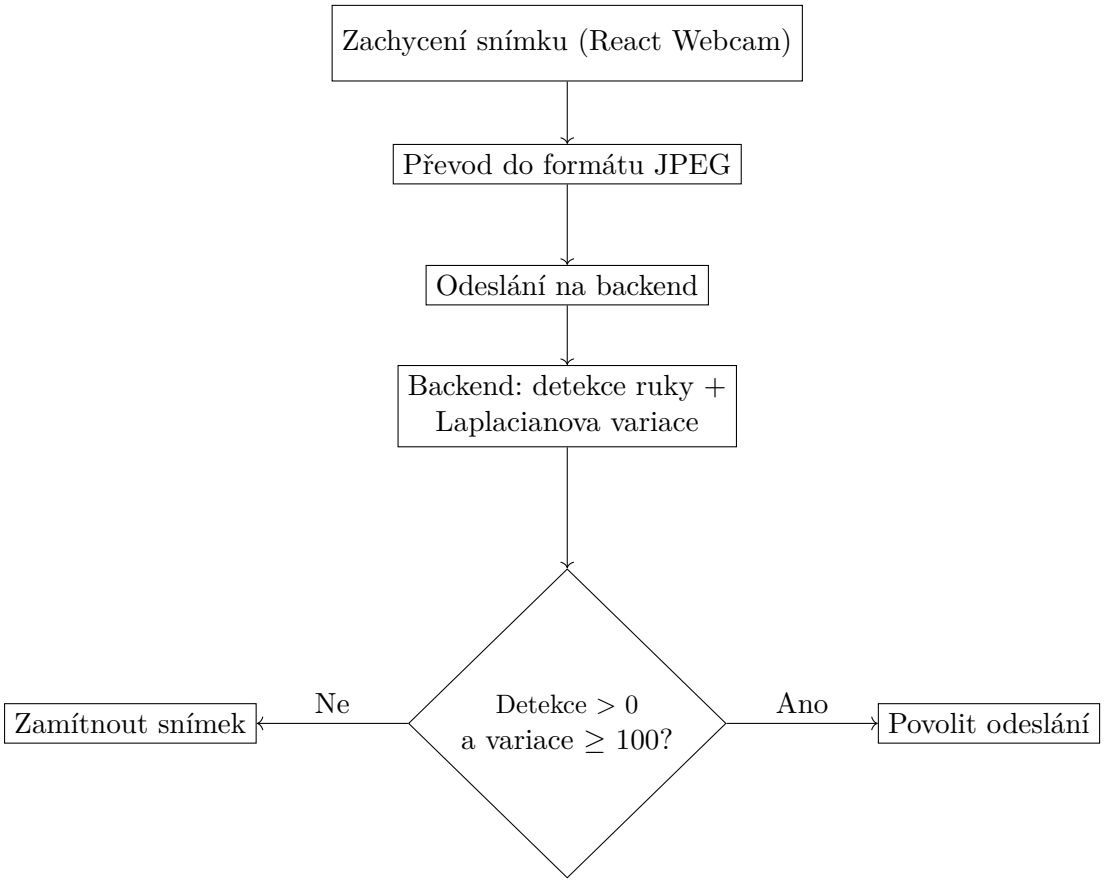
Po zachycení snímku z kamery je tento obraz převeden do formátu JPEG a odeslán na backend pomocí HTTP požadavku. Backendová část systému provádí dvě klíčové operace: detekci přítomnosti ruky v obraze a vyhodnocení kvality snímku pomocí Laplacianovy metody.

Detekce ruky je realizována prostřednictvím předem natrénovaného modelu, který analyzuje obraz a vrací pole detekovaných objektů. Pokud backend detekuje alespoň jeden objekt klasifikovaný jako ruka, je na frontend předána hodnota `response.data.detections` s nenulovou délkou. Tato informace slouží k vizuální zpětné vazbě uživateli – v případě, že aplikace v obraze nedetekuje žádnou ruku, je uživatel upozorněn informační hláškou na tuto skutečnost. Uživatel je tak informován o úspěšnosti detekce ještě před samotným odesláním finálních dat.

Kromě detekce objektu je zároveň vyhodnocována technická kvalita snímku pomocí tzv. Laplacianovy variance (`laplacian_variance`). Tento ukazatel vychází z analýzy změn intenzity jasu v obraze – nízká hodnota značí rozmazání či neostrost. Aplikace nastavuje práh kvality na hodnotu 100; pokud variace nedosahuje této hodnoty, je snímek označen jako nevyhovující. Uživateli je v takovém znemožněno pořídit snímek.

Tato kombinace detekce a kontroly kvality umožňuje aplikaci aktivně filtrovat nevhodné snímky a zvyšuje tak spolehlivost následného zpracování rukou ve fotografiích.

Na obrázku 6.1 je zobrazeno schéma o rozhodnutí kvality snímku.



Obrázek 6.1: Schéma zpracování snímku a rozhodnutí o jeho kvalitě

Zachycení fotografie

Zachycení fotografie je zajištěno funkcí, která volá metodu pro získání aktuálního náhledu z kamery. Pořízený snímek je uložen mezi již zachycené snímky a zobrazuje se uživateli v přehledu. Tento mechanismus umožňuje pořídit více fotografií za sebou a následně je zpracovat nebo odeslat na server společně s dalšími doplňujícími údaji z formuláře.

Formulář

Komponenta `SubmissionForm` zajišťuje odeslání fotografií a metadat pomocí funkce `submitDataWithCallbacks`. Tato funkce přijímá instanci API klienta, formulářová data, identifikátor synchronizační položky a aktualizační funkci stavu.

Před samotným odesláním dojde k resetování stavu konkrétní položky ve frontě – nastaví se nulový postup nahrávání a odstraní se indikace případného selhání. Během přenosu na server je pomocí posluchače průběžně aktualizován údaj o průběhu nahrávání.

Po úspěšném přenosu se příslušná položka označí jako synchronizovaná. V případě chyby je položka označena jako neúspěšná. Uživatel je následně informován pomocí hlášky z knihovny `react-toastify`, která zobrazí výzvu k opakování odeslání.

Součástí komponenty je rovněž převod obrazových dat z formátu base64 do formátu Blob, což je nutné pro korektní přenos přes HTTP protokol. Celý proces odesílání je tak robustní, flexibilní a přizpůsobený různým síťovým podmínkám.

Kapitola 7

Testování

Tato kapitola se zaměřuje na testování mobilní aplikace typu PWA (Progressive Web App), která byla vytvořena v rámci bakalářské práce. Cílem je zjistit míru intuitivnosti, efektivity a celkového dojmu z používání aplikace u různých věkových skupin, a to na základě konkrétních úkolů, hodnoticích škál a otevřené zpětné vazby dle připraveného testovacího protokolu. V průběhu vývoje byly provedeny opakované iterace testování, které sloužily k identifikaci a odstranění nedostatků, ale také k ověření nově přidaných funkcionalit. Testování bylo koncipováno tak, aby odpovídalo reálným podmínkám uživatelského provozu a poskytlo vývojovému týmu relevantní zpětnou vazbu. Kapitola nejprve představuje teoretická východiska testování mobilních aplikací, poté se věnuje samotnému průběhu testování a jeho jednotlivým fázím.

7.1 Testování mobilních aplikací s uživateli

Celá tato kapitola čerpá z přístupů a doporučení uvedených v odborných publikacích autorů Nielsena a Rubinové s Chisnellovou, které se zabývají metodikou testování použitelnosti softwarových rozhraní [16, 20]. Testování mobilních aplikací je nedílnou součástí jejich vývoje, přičemž zásadní roli hraje zejména uživatelské testování (usability testing). Tento typ testování se zaměřuje na praktické ověření, zda uživatelé dokáží aplikaci efektivně, účinně a spokojeně používat. Podle Nielsenovy definice je použitelnost složena z pěti komponent: snadnost naučení, efektivita používání, zapamatovatelnost, počet chyb a spokojenost uživatele.

Metody uživatelského testování

U mobilních a progresivních webových aplikací se k testování často používá tzv. task-based testing, kdy jsou uživatelé požádáni o splnění specifických úkolů. Během těchto testů lze sledovat nejen úspěšnost dokončení úkolu, ale i čas potřebný k jeho provedení, počet chyb nebo subjektivní hodnocení. V rámci task-based přístupu je důležité navrhnout scénáře realisticky odpovídající běžnému používání aplikace. Tyto scénáře by měly pokrývat hlavní funkce aplikace a zároveň být dostatečně jednoznačné, aby eliminovaly variabilitu způsobenou nepochopením instrukcí.

Alternativními metodami mohou být například A/B testování, které porovnává dvě různé verze rozhraní, nebo eye-tracking, jenž umožňuje analyzovat, kam uživatel směřuje svou pozornost. Metoda think-aloud je pak vhodná k zachycení uživatelských úvah během

plnění úkolů a poskytuje hlubší náhled na to, jak účastník testu rozumí jednotlivým krokům v aplikaci.

Moderované a nemoderované testování

Uživatelské testování může mít formu moderovaného testu (facilitated test), kde je tester přítomen a klade doplňující dotazy, nebo nemoderovaného testu, který probíhá bez interakce s testovanou osobou. Moderované testování je obvykle vhodnější v počátečních fázích vývoje, kdy je potřeba rychle odhalit zásadní nedostatky a detailně porozumět uživatelskému chování. Naopak nemoderované testy, realizované např. prostřednictvím online nástrojů, jsou efektivní při validaci řešení u většího počtu respondentů a v různých prostředích.

Vyhodnocování výsledků testování

Důležitou roli při interpretaci výsledků hraje kvalitativní i kvantitativní analýza. Kvantitativní metody poskytují objektivní data, jako je průměrná doba splnění úkolu, počet chyb nebo skóre spokojenosti. Tato data umožňují statistické porovnání mezi různými verzemi aplikace nebo skupinami uživatelů.

Kvalitativní analýza naproti tomu zahrnuje pozorování uživatelského chování, analýzu verbálních komentářů a otevřených odpovědí. Tento přístup odhaluje hlubší problémy s pochopením logiky aplikace, srozumitelností textů, či reakcí na chybové stavy. Kombinací obou metod lze získat komplexní obrázek o použitelnosti a návrhových nedostatcích aplikace.

Jak se testovalo

Testování probíhalo ve formě moderovaného testu, během něhož byly jednotlivé kroky testovacího protokolu průběžně konzultovány mezi zkoušejícím a účastníkem. Účastníci obdrželi obecně formulované úkoly bez podrobného návodu, čímž byla simulována situace spontánního používání. Tento přístup umožnil sledovat, do jaké míry je aplikace intuitivní i bez předchozí instruktáže. Tester během testování pozoroval chování uživatelů, zaznamenával jejich reakce a postřehy do otevřené části protokolu, čímž získal cenná kvalitativní data o jejich zkušenostech a případných obtížích.

Testování aplikace probíhalo podle předem stanoveného protokolu, jehož cílem bylo získat informace o použitelnosti aplikace napříč různými skupinami uživatelů. Testovací scénář se skládal ze tří základních úkolů:

- Seznámení s aplikací
- Pořízení fotografií hřbetu ruky
- Odeslání fotografií do databáze

Každý účastník testu nejprve vyplnil demografické údaje, jako je věk, pohlaví a předchozí zkušenosti s podobnými aplikacemi. Po splnění úkolů byl požádán o vyplnění hodnoticího dotazníku zaměřeného na vzhled aplikace, její navigaci, intuitivnost a celkový dojem. Hodnocení bylo provedeno na desetibodové škále, doplněné o prostor pro otevřenou zpětnou vazbu, kde mohli uživatelé uvést konkrétní připomínky či návrhy na zlepšení.

Testování probíhalo v přirozeném prostředí s cílem zachytit reálné chování uživatelů. Výsledky byly následně vyhodnoceny jak kvantitativně (průměrné skóre jednotlivých kategorií), tak kvalitativně (analýza komentářů a pozorování). Cílem tohoto přístupu bylo nejen

identifikovat technické či designové nedostatky, ale také zjistit, jak různí uživatelé aplikaci vnímají a používají v praxi.

Testovací protokol je zobrazen v příloze B na obrázku B.1.

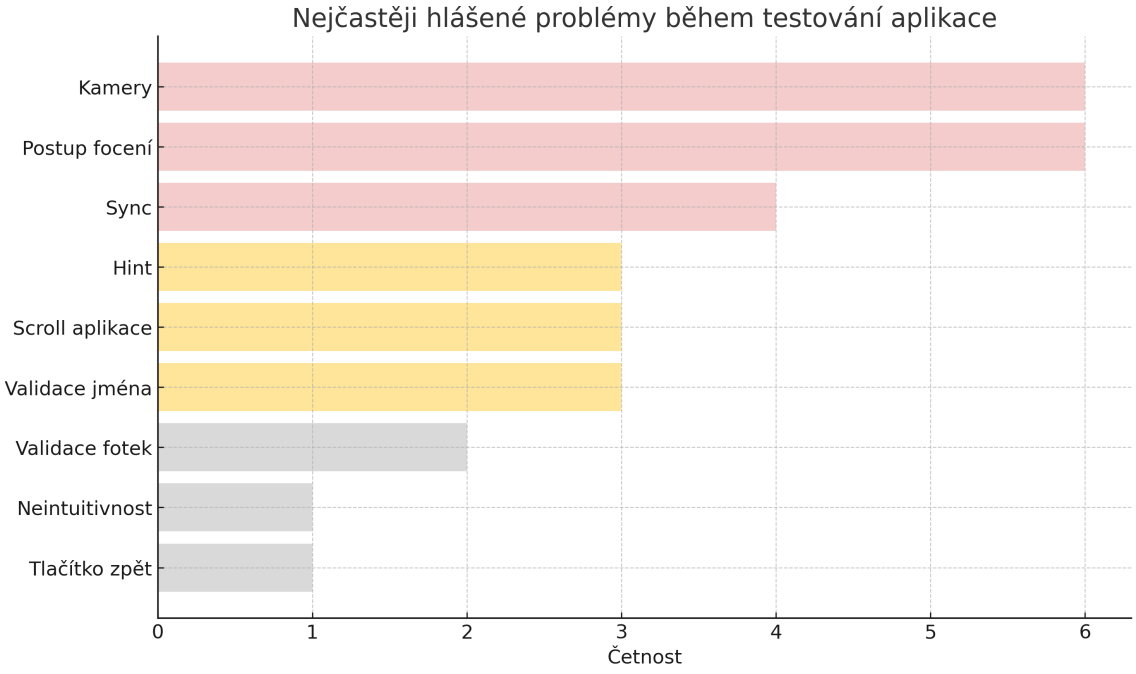
7.2 Jednotlivé iterace

V rámci testování byly realizovány celkem tři iterace, které navazovaly na jednotlivé fáze vývoje aplikace. První iterace se zaměřila na testování původní verze aplikace a identifikaci hlavních problémů z hlediska použitelnosti. Druhá iterace ověřovala úpravy provedené na základě zjištěných nedostatků. Třetí, závěrečná iterace se soustředila na validaci nově přidaných funkcionalit a celkovou stabilitu systému.

Testování původní aplikace

První iterace testování byla provedena s deseti subjekty různého věku, pohlaví a s rozdílnou mírou zkušeností v oblasti informačních technologií. Cílem této fáze bylo zmapovat hlavní problematické oblasti původní verze aplikace a zjistit, které prvky rozhraní jsou pro běžného uživatele nejvíce matoucí nebo nepřehledné. Výsledky byly zaznamenány v podobě četnostní analýzy, která odhalila nejčastěji zmiňované problémy během používání aplikace.

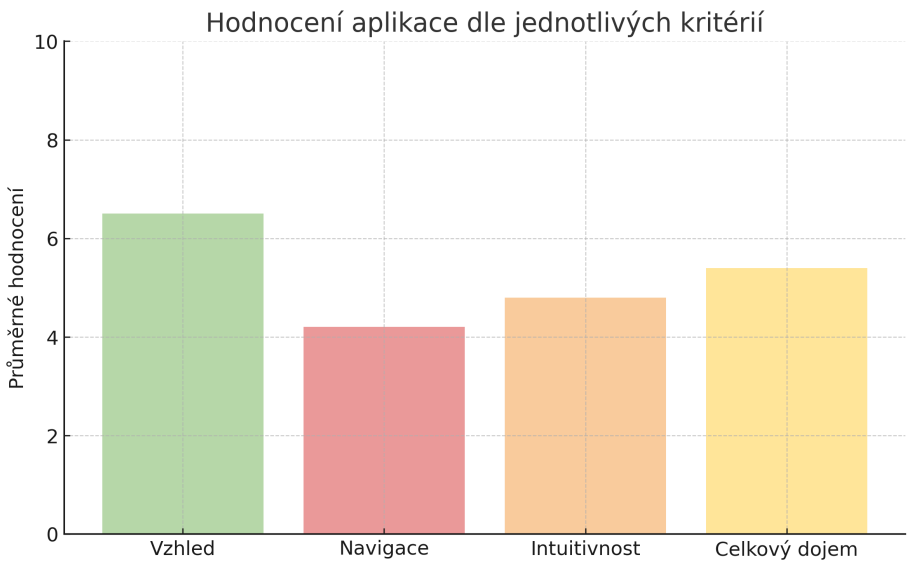
Následující graf na obrazku 7.1 znázorňuje nejčastější problémy identifikované během testování.



Obrázek 7.1: Nejčastěji hlášené problémy během testování aplikace Capture My Hand.

Nejvíce subjektů naráželo na problematiku výběru kamery a nejasnosti ohledně postupu při pořizování fotografií. Následovaly méně časté potíže týkající se nápovědy, synchronizace či validace vstupních údajů. Níže jsou tyto problémy popsány podrobněji:

- **Kamery** – Subjekty opakovaně narážely na nepřehledný způsob výběru kamery. Aplikace umožňovala zvolit mezi přední a zadní kamerou, přičemž v některých případech zařízení nabízelo více zadních kamer. Tato situace byla pro testované osoby matoucí a vedla k prodlevám a opakovaným pokusům při výběru správného záznamového zařízení.
- **Postup focení** – Aplikace obsahovala grafický indikátor (progress bar) okolo tlačítka spouště, který měl znázorňovat počet již pořízených fotografií. Subjekty si tohoto indikátoru často nevšimly, což vedlo k tomu, že si nebyly jisté, kolik snímků již pořídily, případně se během focení zcela zasekly.
- **Sync** – Záložka „Sync“, která sloužila ke kontrole chybových hlášení při nahrávání fotografií do databáze, nebyla během testování ani jedním subjektem aktivně využita. Většina si její přítomnosti všimla až při závěrečné diskuzi s testujícím, což svědčí o její nízké vizuální nebo funkční výraznosti.
- **Hint** – Tři ze subjektů nedokázaly samostatně najít tlačítko s nápovědou (hint), které mělo sloužit k vysvětlení správné pozice ruky při focení. V důsledku toho zprvu nevěděly, jakým způsobem ruku správně vyfotit, a hledaly pomoc jinde v rozhraní.
- **Scroll aplikace** – Další tři subjekty narazily na problém se scrollováním. V některých případech bylo tlačítko spouště částečně zakryto nebo zcela skryto pod systémovou lištou. Tento problém se opakoval v různých částech aplikace a byl způsoben nesprávným přizpůsobením rozhraní velikosti displeje.



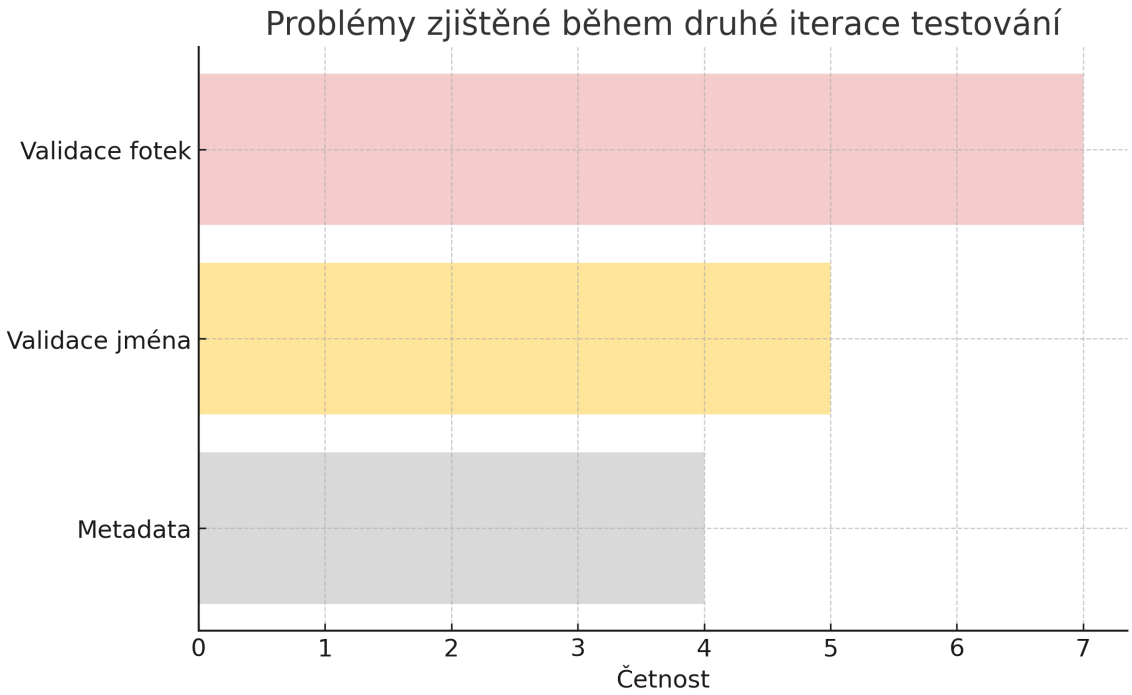
Obrázek 7.2: Průměrné hodnocení jednotlivých aspektů aplikace Capture My Hand podle uživatelů.

Z výše uvedeného grafu na obrázku 7.2 je patrné, že uživatelé hodnotili vizuální stránku aplikace relativně pozitivně, zatímco za problematické označili především navigaci a intuitivnost rozhraní. Tyto poznatky napomohly identifikovat oblasti, které si vyžadují přepracování z hlediska uživatelského komfortu a srozumitelnosti.

Testovaní opravy chyb

Na základě výsledků z předchozí fáze byly provedeny cílené úpravy aplikace. Zrušena byla možnost manuálního výběru kamery – nově se automaticky vybírá hlavní zadní kamera zařízení, čímž se eliminovalo časté zmatení uživatelů. Dále byla implementována knihovna Intro.js, která na úvodní obrazovce zajišťuje jednoduché seznámení s aplikací. Zásadní úpravou prošlo i rozvržení obrazovek – problém se scrollováním a překrýváním ovládacích prvků byl odstraněn ve všech zobrazeních. Vylepšena byla rovněž podoba tlačítek, aby byly lépe rozlišitelné od ostatního obsahu a snáze identifikovatelné jako interaktivní prvky.

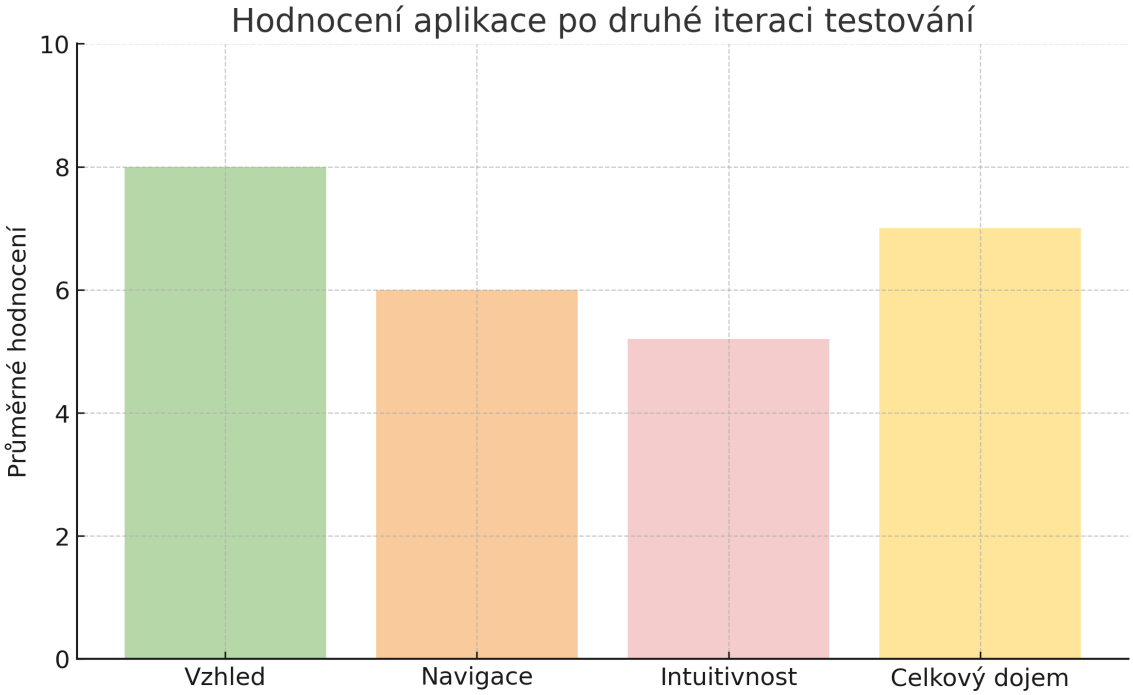
Testování druhé iterace proběhlo opět s deseti subjekty různého věku a IT zkušeností. Bylo použito původní testovací schéma a formulář. Graf na obrázku 7.3 níže shrnuje nejčastější problémy, které byly během této fáze zaznamenány:



Obrázek 7.3: Problémy zjištěné během druhé iterace testování.

Nejčastěji bylo hlášeno nefunkční či nejednoznačné chování validace fotografií (7 subjektů). Problémy přetrvávaly také v oblasti validace jména (5 subjektů), kde nebylo zřejmé, jaké znaky nebo formátování jsou povoleny. Nově se objevila i obtížnost při vyplňování metadat, kterou zmínily 4 osoby. Tento aspekt však nebyl u všech subjektů relevantní a jeho dopad byl méně výrazný.

Obecné zhodnocení aplikace bylo oproti prvnímu kolu výrazně pozitivnější. Subjekty vysoce hodnotily vzhled aplikace (průměrně 8 bodů z 10) a celkový dojem (7 bodů). Intuitivnost a navigace zaznamenaly rovněž mírné zlepšení, i když zde zůstává prostor pro další optimalizaci. Výsledky jsou zachyceny v grafu na obrázku 7.4:



Obrázek 7.4: Průměrné hodnocení aplikace po druhé iteraci testování.

Celkově druhá iterace potvrdila smysluplnost provedených úprav a nasměrovala pozornost na oblasti validace vstupních údajů, které si nadále zaslouží větší důraz při dalším vývoji.

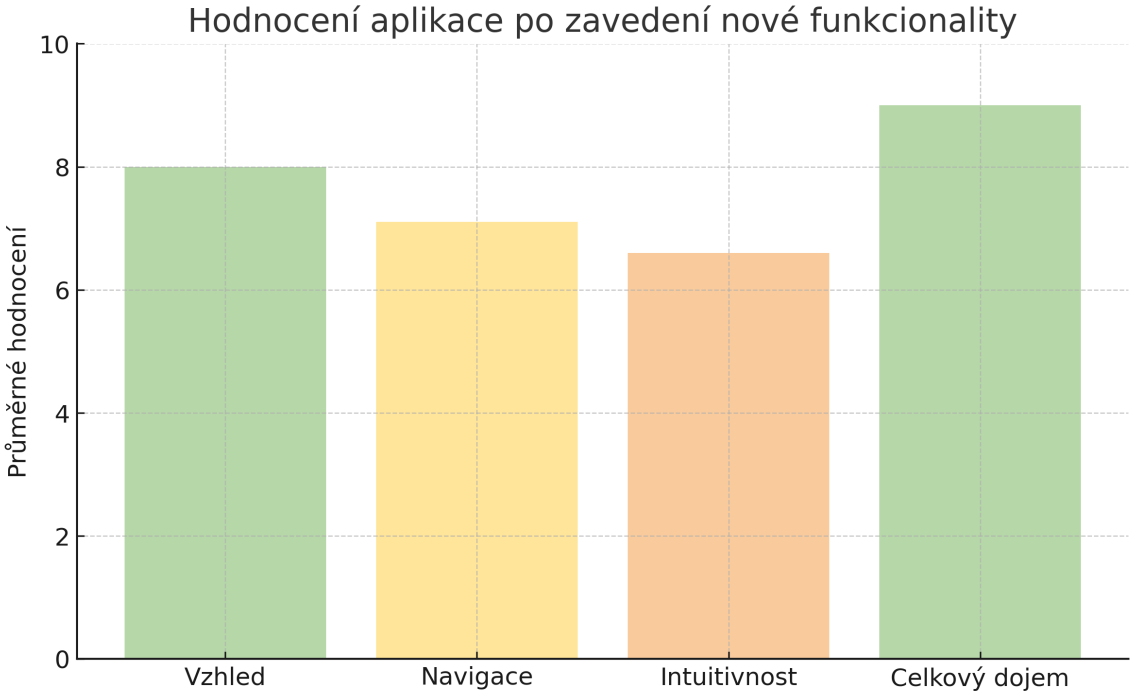
Závěrečné testování a validace nové funkcionality

Ve třetí a závěrečné iteraci vývoje aplikace byly implementovány dvě zásadní nové funkcionality. První z nich byla validace vstupu z kamery – systém v reálném čase vyhodnocuje kvalitu a obsah obrazu na základě metrik, které slouží k detekci přítomnosti ruky. Pokud detekce selže, aplikace znemožní pořízení snímku a je zobrazena informační hláška. Obdobný mechanismus je implementován také pro případy, kdy je kvalita snímku nedostatečná. Tento mechanismus má za cíl zvýšit kvalitu dat a minimalizovat chybné snímky, které neobsahují požadovaný objekt.

Druhou novinkou bylo rozšíření onboardingu pomocí knihovny Intro.js i do formuláře pro zadávání metadat. Uživatel je tak prostřednictvím krokového průvodce informován o správném způsobu vyplnění jednotlivých polí formuláře (např. jméno, poznámka, datum).

Testování této verze aplikace opět probíhalo s deseti subjekty. Hodnocení ukázalo další zlepšení v oblasti vzhledu (8 bodů), navigace (7,1 bodu) i intuitivnosti (6,6 bodu). Celkový dojem byl hodnocen velmi pozitivně s průměrným skóre 9 bodů. Vzhledem k charakteru nových funkcí byly při testování pozorovány zejména reakce na vizuální zpětnou vazbu při neúspěšné detekci a porozumění doprovodným instrukcím.

Grafické shrnutí hodnocení jednotlivých aspektů aplikace je uvedeno na obrázku 7.5.



Obrázek 7.5: Hodnocení aplikace po zavedení validace kamery a onboardingu ve formuláři.

Výsledky závěrečného testování potvrzují, že aplikace dospěla do stabilní a uživatelsky vstřícné podoby. Klíčové problémy z předchozích iterací byly eliminovány a nově přidané funkce byly většinou subjektů přijaty kladně. Dílčí výhrady k rychlosti detekce nebo délce onboardingu byly zaznamenány pouze ojediněle a neměly vliv na celkový pozitivní dojem.

Kapitola 8

Závěr

Cílem této bakalářské práce bylo navrhnout, implementovat a otestovat mobilní aplikaci typu PWA, která umožňuje jednoduchý a uživatelsky přívětivý způsob pořizování fotografií hřbetu ruky a jejich následné odeslání do vzdáleného úložiště. Aplikace je určena pro použití v kontextu sběru obrazových dat, kde je kladen důraz na srozumitelnost rozhraní, konzistenci výstupů a minimalizaci uživatelských chyb. V průběhu vývoje byly zohledněny principy návrhu uživatelského rozhraní pro mobilní zařízení, jako je intuitivnost, prostorová orientace, vizuální hierarchie či progresivní odhalování funkcí.

Zvolená technologie PWA umožnila vytvořit multiplatformní řešení dostupné jak na mobilních telefonech, tak i v běžných webových prohlížečích. Frontend aplikace byl implementován pomocí knihovny React, zatímco backend využíval framework FastAPI. Pro zajištění konzistentního chování a snadné správy komponent byl kód strukturován s důrazem na modularitu. Knihovny jako Intro.js a Toastify.js byly využity pro interaktivní onboarding a vizuální zpětnou vazbu, čímž se zvýšila použitelnost aplikace zejména při prvním kontaktu uživatele.

V rámci testování byly realizovány tři iterace, přičemž každá z nich reagovala na zjištěné nedostatky předchozí fáze. První testování identifikovalo zásadní problémy týkající se výběru kamery, navigace a nápovědy. Ve druhé iteraci byly tyto nedostatky odstraněny a rozhraní bylo upraveno tak, aby lépe odpovídalo očekáváním běžného uživatele. Závěrečná třetí iterace přinesla pokročilou validaci vstupu z kamery pomocí metrik pro detekci ruky v obraze, které umožnily blokovat pořízení snímků bez požadovaného obsahu. Dále byl rozšířen interaktivní průvodce i na část formuláře, čímž se snížila chybovost při vyplňování údajů.

Na základě kvantitativního i kvalitativního hodnocení lze konstatovat, že aplikace dosáhla vysoké míry použitelnosti. Subjekty testování pozitivně hodnotily zejména vzhled a celkový dojem, přičemž vývoj každé iterace prokazatelně vedl ke zlepšení vnímané intuitivnosti a navigace. Výsledné řešení je připraveno k nasazení v praxi, a to jak pro testovací, tak i produkční účely.

Do budoucna je možné aplikaci dále rozvíjet – například rozšířením zpětné vazby pro uživatele o interaktivní náhled detekce, přidáním více typů validací nebo vylepšením exportu nasbíraných dat. Významný potenciál má rovněž propojení s mobilními senzory a rozšířená analytika využívající strojového učení. Z pohledu vývoje se aplikace *Capture My Hand* osvědčila jako udržitelný a rozšiřitelný nástroj vhodný pro širší veřejnost.

Literatura

[1] Abramov, D. a Contributors, R. *Introducing Hooks – React*. 2019. Dostupné z: https://reactjs.org/docs/hooks-intro.html.

[2] Abramov, D. a Contributors, R. *Using the Effect Hook – React*. 2019. Dostupné z: https://reactjs.org/docs/hooks-effect.html.

[3] Abramov, D. a Contributors, R. *Using the State Hook – React*. 2019. Dostupné z: https://reactjs.org/docs/hooks-state.html.

[4] Alvarez, D. P. *Notyf: Minimalistic, Simple, Clean JS Notification Library*. 2024. Dostupné z: https://github.com/caroso1222/noty.

[5] Bevan, N. Measuring usability as quality of use. *Software Quality Journal*, 1995, sv. 4, č. 2, s. 115–130. ISSN 0963-9314.

[6] Dabit, N. *React Native in Action*. 1. vyd. Manning Publications, 2019. ISBN 978-1617294051.

[7] Fielding, R. *Architectural Styles and the Design of Network-based Software Architectures* Doctoral dissertation, University of California, Irvine. 2000. https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[8] Flanagan, D. *JavaScript: The Definitive Guide*. 7. vyd. O'Reilly Media, 2020. ISBN 978-1491952023.

[9] Grant, K. J. *CSS in Depth*. 1. vyd. Manning Publications, 2018. ISBN 978-1617293450.

[10] Intro.js Contributors. *Intro.js Documentation*. 2024. Dostupné z: https://introjs.com/docs.

[11] Khadra, F. *React-Toastify*. 2024. Dostupné z: https://github.com/fkhadra/react-toastify.

[12] Kim, M. a Lee, J. PicMe: Interactive Visual Guidance for Taking Requested Photo Composition. In: ACM. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, s. 1–12. ISBN 978-1-4503-5970-2.

[13] Liddell, T. M. a Baddeley, A. D. Spatial working memory. *The Quarterly Journal of Experimental Psychology*, 1982, sv. 34, č. 4, s. 403–421. ISSN 0272-4987.

[14] Lugaresi, C.; Tang, J.; Nash, H.; McGuire, M.; Chang, C.-L. et al. *MediaPipe: A Framework for Building Perception Pipelines* arXiv preprint arXiv:1906.08172. 2019. Dostupné z: https://arxiv.org/abs/1906.08172.

[15] Mittal, A.; Moorthy, A. K. a Bovik, A. C. No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing*, 2012, sv. 21, č. 12, s. 4695–4708. ISSN 1057-7149.

[16] Nielsen, J. *Usability Engineering*. 1. vyd. Morgan Kaufmann, 1993. ISBN 978-0125184069.

[17] Norman, D. A. *The Design of Everyday Things: Revised and Expanded Edition*. Revised and Expanded. Basic Books, 2013. ISBN 978-0465050659.

[18] Pech Pacheco, J. L.; Cristóbal, G.; Chamorro Martinez, J. a Fernández Valdivia, J. Diatom autofocusing in brightfield microscopy: a comparative study. In: *Proceedings 15th International Conference on Pattern Recognition (ICPR)*. IEEE, 2000, s. 314–318. ISBN 0-7695-0750-6.

[19] Prettier Contributors. *Prettier Documentation*. 2024. Dostupné z: https://prettier.io/docs/.

[20] Rubin, J. a Chisnell, D. *Handbook of Usability Testing: How to Plan, Design and Conduct Effective Tests*. 2. vyd. Wiley, 2008. ISBN 978-0470185483.

[21] Russell, A. a Osmani, A. *Learning Progressive Web Apps*. 1. vyd. Addison-Wesley Professional, 2021. ISBN 978-0136484226.

[22] Saffer, D. *Microinteractions: Designing with Details*. 1. vyd. O'Reilly Media, 2013. ISBN 978-1491945926.

[23] Shepherd.js Contributors. *Shepherd.js Installation Guide*. 2024. Dostupné z: https://docs.shepherdjs.dev/guides/install/.

[24] Shneiderman, B.; Plaisant, C.; Cohen, M.; Jacobs, S.; Elmqvist, N. et al. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. 6. vyd. Pearson, 2016. ISBN 978-0134380384.

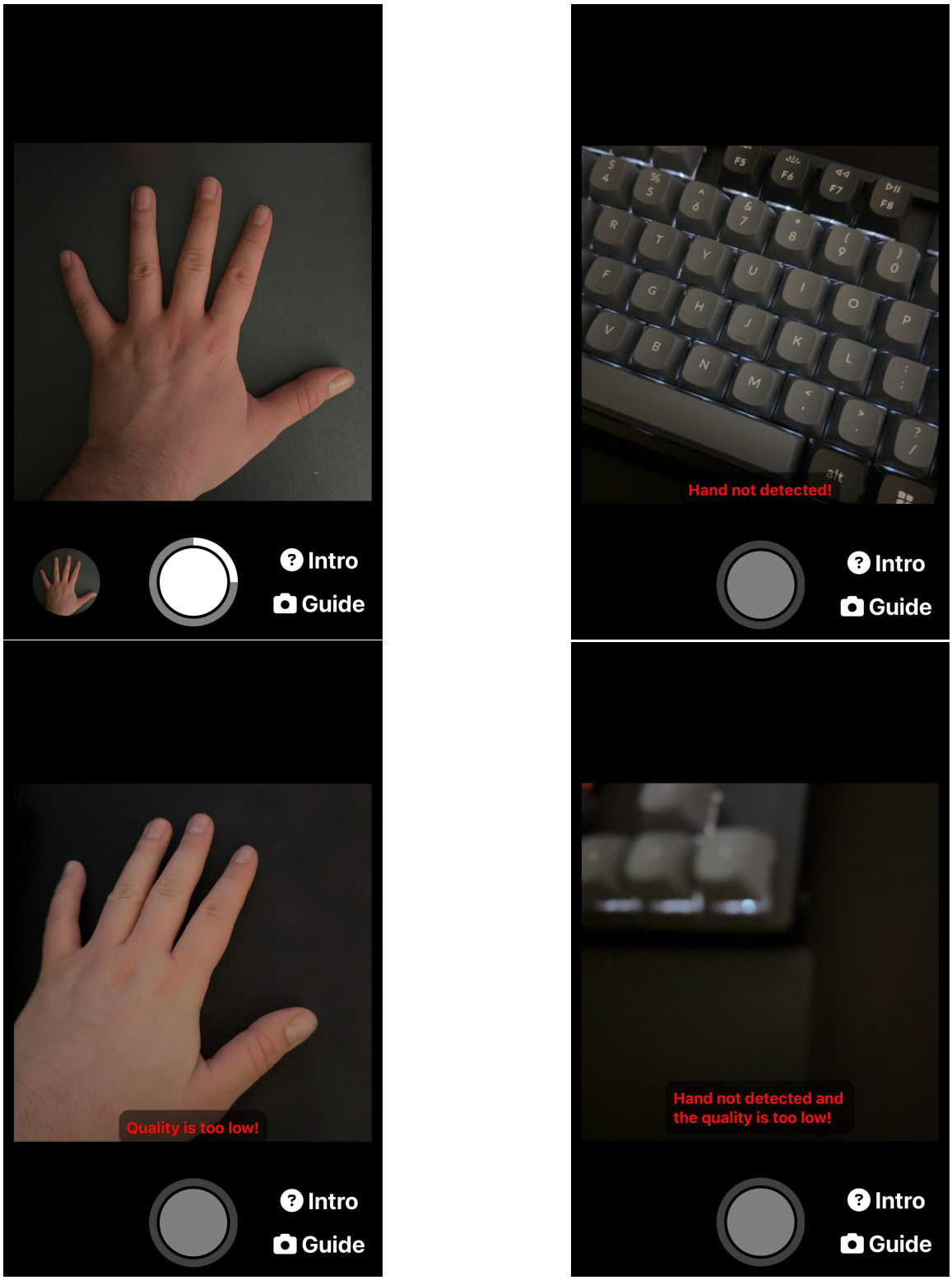
[25] Swagger Contributors. *OpenAPI Specification Version 3.0 – Introduction*. 2024. Dostupné z: https://swagger.io/docs/specification/v3_0/about/.

[26] Sweller, J. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 1988, sv. 12, č. 2, s. 257–285. ISSN 0364-0213.

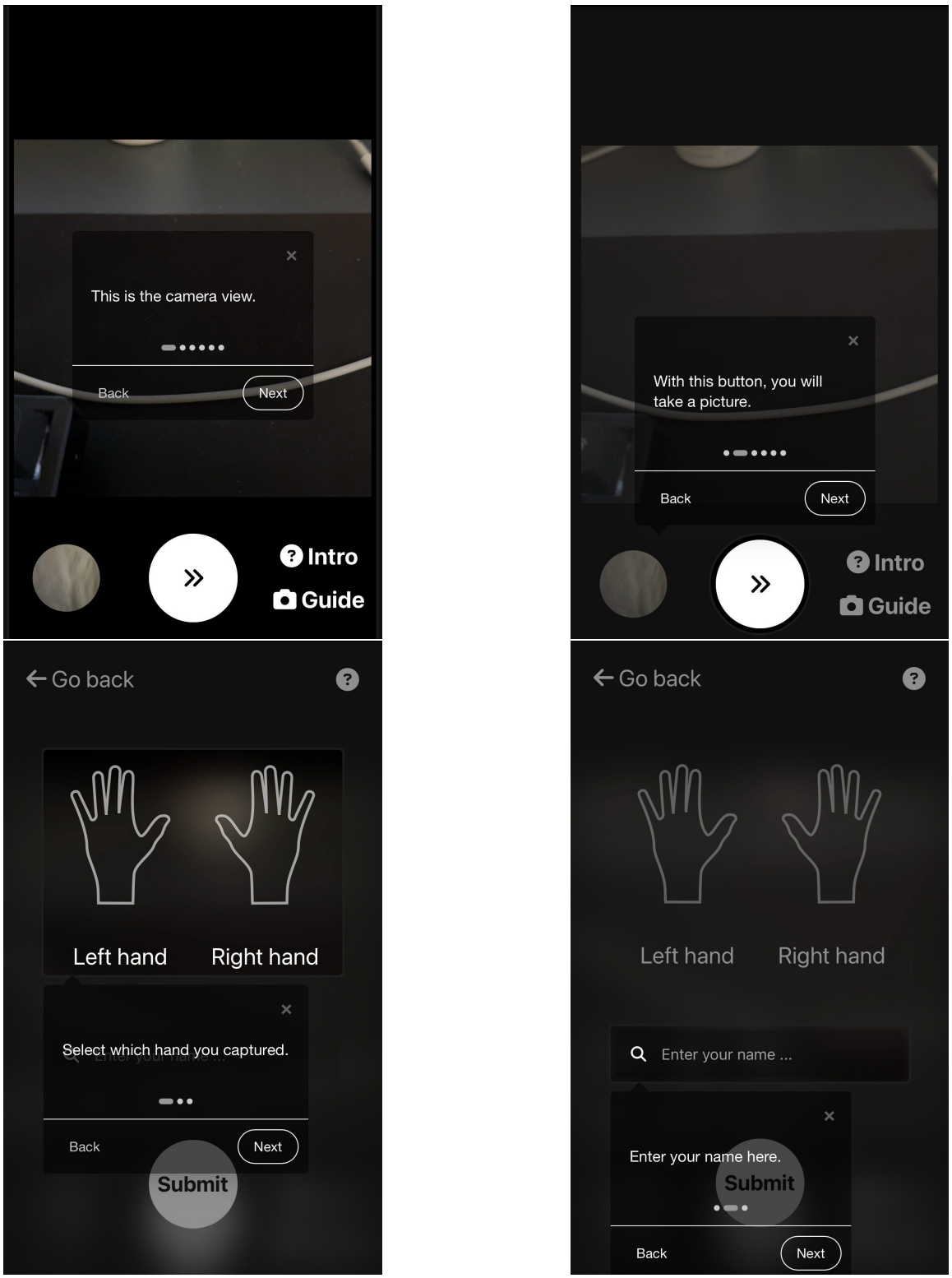
[27] Tidwell, J. *Designing Interfaces: Patterns for Effective Interaction Design*. 2. vyd. O'Reilly Media, 2011. ISBN 978-1492051961.

Příloha A

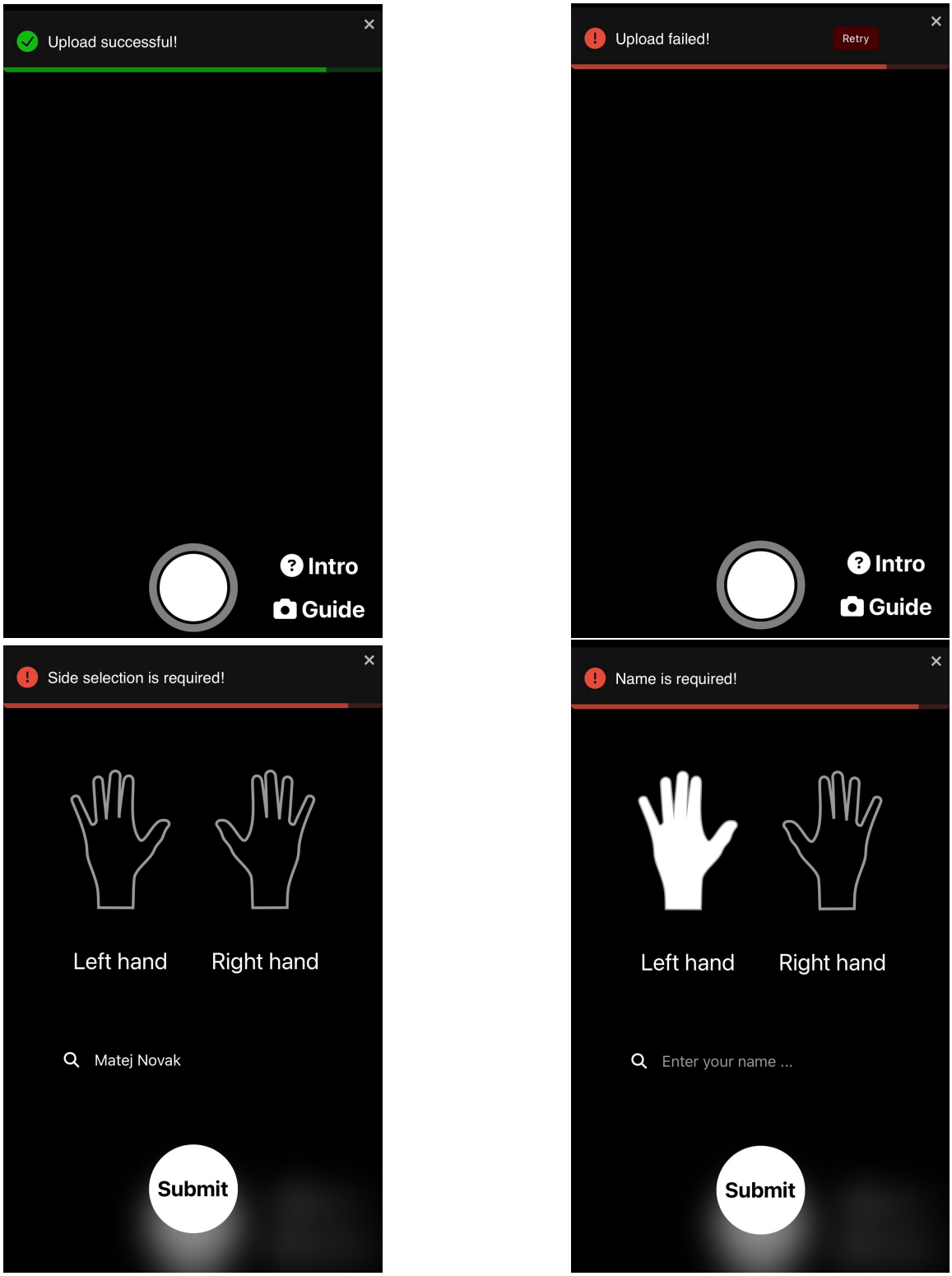
Konečný stav aplikace



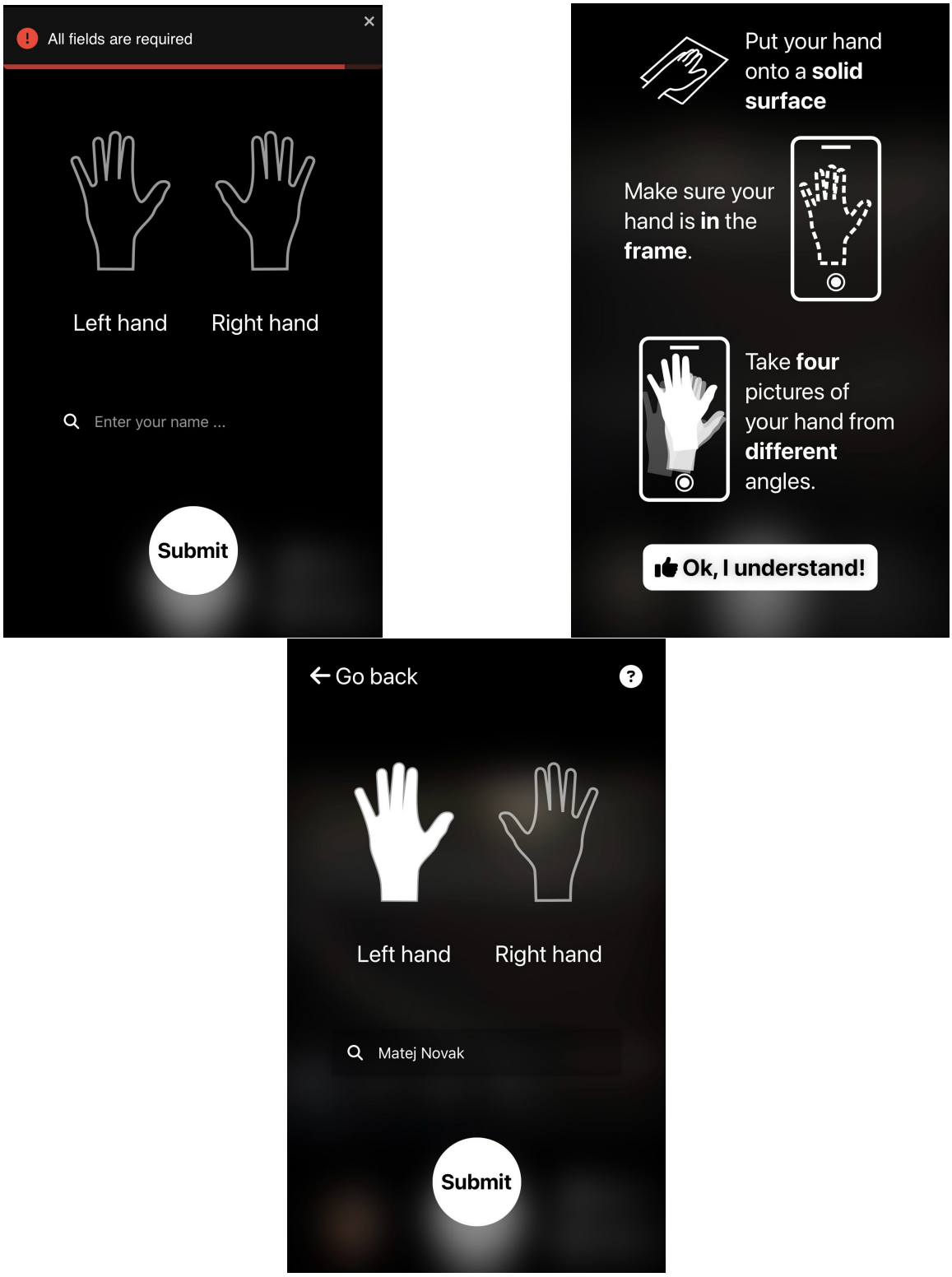
Obrázek A.1: Úvodní obrazovka aplikace, upozornění na absenci ruky a nízkou kvalitu snímku.



Obrázek A.2: Použití Intro.js na úvodní obrazovce a ve formuláři.



Obrázek A.3: Použití toast notifikací ve formuláři a po odeslání formuláře.



Obrázek A.4: Toast notifikace při nevyplnění žádného z povinných polí, nápověda jak zachycovat snímky a správně vyplněný formulář.

Příloha B

Testovací protokol

Testování aplikace pro sběr fotek hřbetu ruky

Testovaná aplikace je aplikace pro sběr fotek hřbetu ruky. Jedna se jednoduchou aplikaci skládající se pouze z fotoaparátu a následné zaslání do databáze.

Testováním by se mělo docílit celkového lepšího dojmu z používání aplikace a implementování připomínek testovaných osob. Dále zjištění do jaké míry je aplikace intuitivní jak pro mladé, tak i pro starší osoby.

Demografické Informace

Věk, pohlaví, zkušenosti s podobnými aplikacemi.

Úkoly k Provedení

Popis úkolů, které uživatelé budou vykonávat (např. spuštění aplikace, navigace, fotografování hřbetu ruky, nahrávání obrázku do databáze).

1. Seznámení se s aplikací
2. Fotka ruky
3. Odeslání fotky

Hodnocení Uživatelského Rozhraní

Dotazník hodnotící intuitivnost, design, navigaci, rychlost reakcí aplikace atd.

Vzhled

Nejhorší									Nejlepší
1	2	3	4	5	6	7	8	9	10

Navigace

Nejhorší									Nejlepší
1	2	3	4	5	6	7	8	9	10

Intuitivnost

Nejhorší									Nejlepší
1	2	3	4	5	6	7	8	9	10

Celkový dojem

Nejhorší									Nejlepší
1	2	3	4	5	6	7	8	9	10

Otevřená Zpětná Vazba

Prostor pro uživatelské komentáře a návrhy na vylepšení.

Obrázek B.1: Testovací protokol pro všechny 3 iterace.