FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Aplikace pro administraci testů manuální zručnosti
Student:	Nikita Nikolchuk
Vedoucí:	Ing. Tomáš Vondra, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství 2021
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Na Klinice rehabilitačního lékařství 1.LF UK a VFN v Praze probíhá výzkum zaměřený na standardizované testy hodnotící funkci horních končetin. Jde o Box and Block Test, Nine Hole Peg Test a Purdue Pegboard Test. Při výzkumu i v klinické praxi by velmi pomohla aplikace, která by naváděla terapeuta při administraci testů a zaznamenávala výsledky. Funkčnost by měla zahrnovat přehrávání zvukových instrukcí testovanému, zobrazení instrukcí pro testujícího, možnost případně subtest přerušit či opakovat při událostech popsaných v manuálu (např. upadnutí součástky na zem), nahrávání videa z připojené kamery a spouštění modelu strojového vidění k automatickému vyhodnocení testu, který byl vyvinut v rámci jiného projektu. Po testu se exportují výsledky daného pacienta včetně možnosti srovnání s normami.

1. Analyzujte detailně požadavky se zadavatelem a dále s ním spolupracujte při designu aplikace.
2. Navrhněte vhodné technologie a softwarovou architekturu aplikace.
3. Aplikaci implementujte.
4. Otestujte její použitelnost s terapeutem a případné připomínky zapracujte.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 13. února 2024 v Praze.

Bakalářská práce

APLIKACE PRO ADMINISTRACI TESTŮ MANUÁLNÍ ZRUČNOSTI

Nikita Nikolchuk

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Tomáš Vondra, Ph.D.
16. května 2024

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat vedoucímu panu doktoru Tomáši Vondrovi za jeho pomoc při tvorbě této práce a paní magistře Kateřině Vondrové za konzultace ohledně potřeb ergoterapeutů. Také bych chtěl poděkovat své rodině za jejich trpělivost a podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 16. května 2024

Abstrakt

Táto práce se zabývá vývojem aplikace pro administraci testů manuální zručnosti. Vývoj je založen na použití českých verzí manuálů vybraných standardizovaných testů jemné motoriky, s cílem vytvořit program, který by pomáhal ergoterapeutům v jejich klinické praxi a výzkumu. Výsledkem je samostatná nativní desktopová aplikace pro operační systém Windows. Užitečnost praktické části práce byla úspěšně ukázana a cílová skupina uživatelů aplikaci kladně ohodnotila.

Klíčová slova desktopová aplikace, ergoterapie, WPF, C#

Abstract

This thesis deals with the development of an application for manual dexterity tests. The development is based on the use of Czech versions of manuals for the selected standardized tests of fine motor skills, with the aim of creating a program that would assist ergotherapists in their clinical practice and research. The result is a stand-alone native desktop application for the Windows operating system. The usefulness of the practical part of the work was successfully demonstrated and the application was positively evaluated by the target user group.

Keywords desktop application, ergotherapy, WPF, C#

Seznam zkratek

BBT	Box and Block Test
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
FP	Funkční Požadavek
FFMPEG	Fast Forward Moving Picture Experts Group
FXML	FX Markup Language
HTML	HyperText Markup Language
LF	Lékařská Fakulta
MAUI	Multiplatform Application UI
MVVM	Model-View-ViewModel
NHPT	Nine Hole Peg Test
NP	Nefunkční Požadavek
PDF	Portable Document Format
PPT	Purdue Pegboard Test
PU	Případ Užití
QML	Qt Modelling Language
SD	Standard Deviation
SDS	Standard Deviation Score
UI	User Interface
UK	Univerzita Karlova
UML	Unified Modeling Language
UWP	Universal Windows Platform
UX	User Experience
VFN	Všeobecná Fakultní Nemocnice
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language
XML	Extensible Markup Language
XLSX	Microsoft Excel Spreadsheet

Úvod

Ergoterapie je medicínským oborem, který pomáhá osobám s nejrůznějšími typy zdravotních postižení zlepšovat jejich každodenní funkční schopnosti. Pro odborníky pracující v tomto oboru je důležité mít možnost objektivně vyhodnocovat stav pacientů pro určení léčby a sledování jejího průběhu. Tento problém řeší standardizované testy, které terapeuti mohou provádět podle konkrétních instrukcí.

Problematika této práce zahrnuje tři běžně používané testy hodnotící manuální zručnost: Nine Hole Peg Test (nebo také „Devítikolíkový Test"), Purdue Pegboard Test a Box and Block Test. Detaily jejich provádění budou popsány dále v teoretické části práce. Standardizace vybraných testů je aktuálním problémem v daném oboru. Před třemi lety byly poprvé vydány české verze manuálů [1, 2, 3] a probíhá výzkum zaměřený na vytvoření příslušných českých norem, které nahradí americké normy. Provádění těchto testů je časově náročné a pracné: ergoterapeuti musí zapisovat spoustu hodnot a poznámek, vypočítávat výsledky, vyhledávat různá pravidla atd. Z uvedených výše důvodů by byl vhodný program pro zjednodušení tohoto procesu založený na nově standardizovaných instrukcích.

Cílem této práce je vyvinout aplikaci, která strukturuje všechna používaná data a bude šetřit čas lékařů prostřednictvím částečné automatizace testování. Bude to také přínosné pro výzkumníky, kteří tyto data budou moci použít, a pro samotné pacienty, kteří budou mít přehled výsledků testů od začátku své léčby.

Prvním postupovým cílem je analýzovat problémovou doménu a specifikovat představu zadavatele o výsledném programu. Dalším cílem je strukturovat všechna podstatná data, návrhnout vzhled aplikace a zvolit vhodný technologický základ pro její tvorbu. Navazujícím cílem je realizovat samotnou aplikaci a popsat kostru programu spolu s řešením klíčových technických problémů. V neposlední řadě je cílem otestovat výsledný program prostřednictvím použití specializovaných nástrojů a provedením uživatelského testování s případnými úpravami aplikace. Závěrem práce bude uvedení splněných požadavků a možností dalšího vývoje aplikace.

Kapitola 1
Analýza

Tato kapitola popisuje aplikace z uživatelského pohledu a určuje potřebné funkcionality. Dále jsou uvedeny všechny uživatelské požadavky, současný stav provádění testů manuální zručnosti, případy užití navržené aplikace i analýza dostupných technologií. Analýza stanoví základní strukturu programu, která bude použitá pro jeho softwarový návrh.

1.1 Popis požadované aplikace

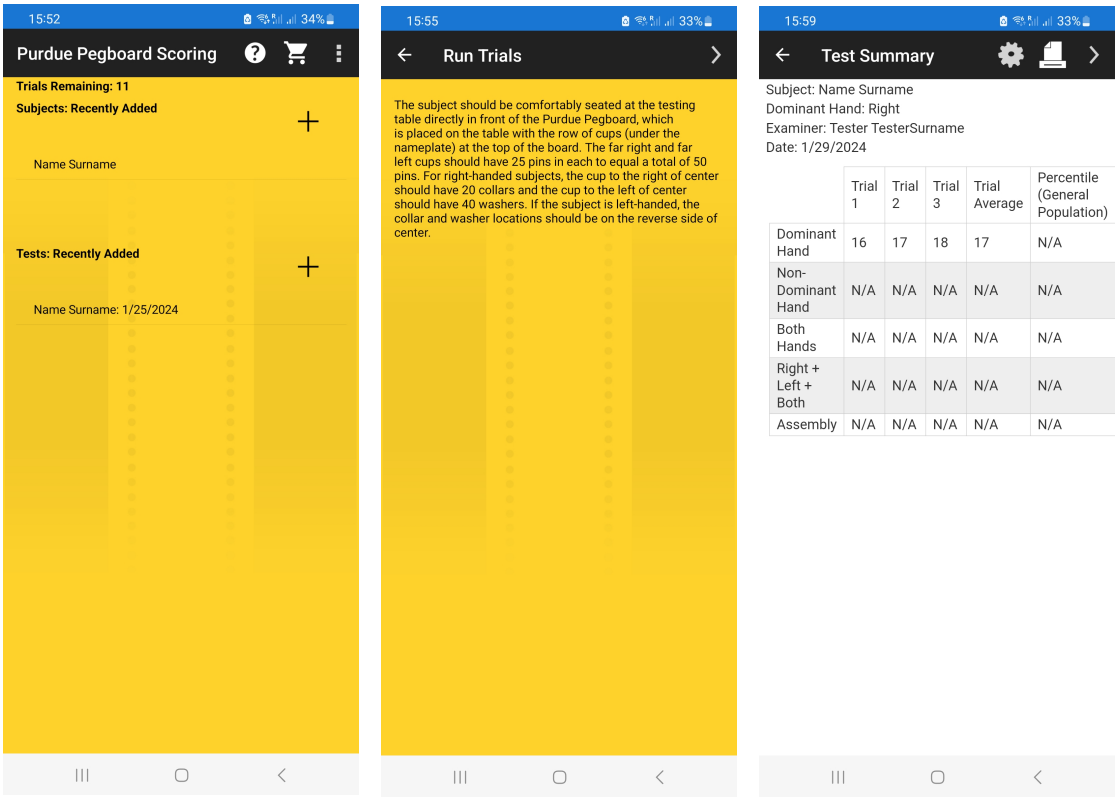
Desktopová aplikace bude sloužit ergoterapeutům v klinické praxi k usnadnění jednotného provádění vybraných standardizovaných testů hodnotících zručnost. Jedná se o Nine Hole Peg Test (dále jen „NHPT“), Purdue Pegboard Test (dále jen „PPT“) a Box and Block Test (dále jen „BBT“). Jsou to testy, které se v praxi nejčastěji používají pro hodnocení jemné motoriky. Aplikace bude vést uživatele standardizovaným provedením každého testu přesně podle manuálu, umožní zaznamenat výsledky testování, napsat poznámky a případně uložit videozáznam z testování, bude vytvářet databázi otestovaných osob pro dané pracoviště.

1.2 Analýza konkurenčních řešení

V současné době na trhu zdravotnických programů neexistuje žádná aplikace odpovídající popisu uvedenému v předchozí podkapitole. Ergoterapeuti se postačují s tím, že provádějí všechny potřebné kroky manuálně a cílem této bakalářské práce je tento proces usnadnit a částečně zautomatizovat. Nicméně existuje pár mobilních aplikací pro administraci jednotlivých testů, které jsou popsané v této podkapitole.

1.2.1 Purdue Pegboard Scoring App

Purdue Pegboard Scoring App [4] od Lafayette Instrument Company je mobilní aplikace v angličtině pro Android a iOS. Používá se pro bodování PPT a je uvedená v České rozšířené verze manuálu pro Purdue Pegboard Test [2].



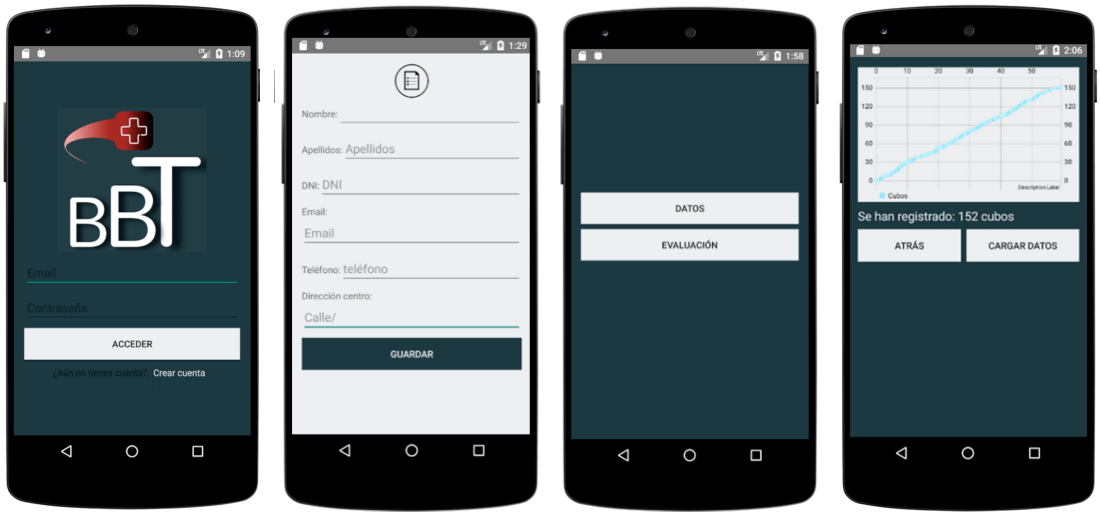
(a) Úvodní obrazovka **(b)** Instrukce pro testujícího **(c)** Tabulka s výsledky

Obrázek 1.1 Purdue Pegboard Scoring App [4]

Aplikace umožňuje lokálně ukládat data o testujících i pacientech a provádět testy. Během testování se ukazují instrukce pro testujícího i pacienta a odpočítává se čas. Po skončení testu se zobrazuje tabulka s výsledky a tlačítko umožňující tisk nebo export do PDF souboru. Také je možné prohlížet grafy výsledků jednotlivých pacientů, ale bez možnosti exportu dat do formátu vhodného pro další zpracování. Počet pokusů pro jednotlivé části PPT je omezený a za každý další je třeba připlatit. Celkově aplikace je nepoužitelná v praxi z důvodu výrazně omezené funkčnosti a zavedení neopodstatněné cenové politiky.

1.2.2 Wizard App

Wizard App je aplikace ve španělštině pro Android vyvinuta na Univerzitě Karla III. v Madridu [5], která slouží k bodování BBT.



(a) Úvodní obrazovka **(b)** Přihlášení **(c)** Nastavení **(d)** Graf výsledků

Obrázek 1.2 Wizard App [5]

Aplikace byla vytvořena v rámci výzkumu o automatizaci BBT pro odpovídající systém počítání kostek na základě senzorů přiblížení. Program ukazuje textové a video instrukce k provádění testu, automaticky počítá výsledky a zobrazuje naměřená data. Pro ukládání výsledků se používá lokální úložiště synchronizované s Firebase [6] databázi. Celkově testující uživatele kladně ohodnotili použití systému a zejména využití cloudového úložiště. Přesto se jedná pouze o prototyp s důrazem na automatizaci počítání kostek a samotna implementace nebyla publikována. Nicméně tento výzkum ukázal, že implementace podobného administračního systému je oceněná zdravotníky a může mít skutečný užitek. Aplikace není připravená k praktickému použití a neodpovídá potřebám českých ergoterapeutů, takže není konkurenční.

1.3 Analýza požadavků

Tato podkapitola se zabývá **systematickou analýzou požadavků**, která převádí uživatelské požadavky z nezpracované formy do strukturovaného seznamu technických požadavků [7]. Analýza požadavků probíhala formou konzultací se zadavatelem, který pracuje v oboru ergoterapie a má představu o potřebách koncových uživatelů. Byl sepsán dokument obsahující všechny požadavky, následně probíhala jejich specifikace.

1.3.1 Funkční požadavky

Funkční požadavky objasňují co se musí udělat a identifikuje nutné úkony, aktivity a akce, které musí být vykonány. Analýza funkčních požadavků se používá jako základ takzvané toplevel funkce systému pro funkční analýzu [8]. Ke každému požadavku v této sekci je přiřazen odhad složitosti a priorita. Požadavky s nižší prioritou nejsou závazné a budou implementovány v závislosti na skutečné časové náročnosti implementace ostatních částí aplikace.

FP1: Úvodní obrazovka

Úvodní obrazovka aplikace musí obsahovat tlačítka umožňující provedení testu, zobrazení uložených výsledků a nastavení aplikaci. Také by v úvodu měli být odkazy na:

- Webovou stránku s PDF manuály ke stažení [9].
- Online kurzy pro členy České asociace ergoterapeutů s videomanuály [10].

Priorita: vysoká
Složitost: střední

FP2: Zobrazení a přehrávání instrukcí

Během provedení testů aplikace musí zobrazovat a přehrávat několik typů instrukcí:

- Zeleně psané instrukce pro pacienta, zároveň předem namluvené zvukové instrukce.
- Černě nebo žlutě psané instrukce pro testujícího.
- Červeně psaný text, který má testující říct ústně.

Varianta instrukcí záleží na typu testu, pohlaví pacienta a jeho/její dominantní horní končetině podle psaní.

Priorita: vysoká
Složitost: střední

FP3: Předčasné ukončení

Musí být možnost kdykoliv uložit rozpracovanou část testování a ukončit aplikaci.

Priorita: vysoká
Složitost: střední

FP4: Poznámky k pokusům

Po každém pokusu se musí zobrazovat pole pro poznámky.

Priorita: vysoká
Složitost: střední

FP5: Anulování pokusu

Musí existovat možnost označit, že pokus nebyl úspěšně dokončen a musel být anulován. V takovém případě aplikace nabídne prostor pro zapsání poznámky a zobrazí text, který má testující osoba sdělit ústně. Celkem každý pokus může být zahájen až třikrát.

Priorita: vysoká
Složitost: střední

FP6: Normy testů

Aplikace musí používat americké normy pro hodnocení výsledků testů [11, 12, 13]. České normy prozatím neexistují a předpokládaný termín dokončení potřebného sběru dat pro jejich vytvoření je konec 2025 roku. Proto návrh programu musí umožnit snadnou aktualizaci norem.

Priorita: vysoká
Složitost: střední

FP7: Zobrazení výsledků

Po dokončení testování aplikace musí nabídnout možnost ukázat pro daný test:

- Samotné výsledky testované osoby.
- Výsledky testované osoby v porovnání s normou antropometricky.
- Výsledky testované osoby v porovnání s normou běžným způsobem.
- Žádné výsledky.

Aplikace musí umožňovat zobrazení výsledků předchozího testování pro dodatečné porovnání. Bez ohledu na výběr, aplikace také musí zobrazit text určený pro kopírování do lékařské dokumentace.

Priorita: vysoká
Složitost: střední

FP8: Ukládání výsledků

Aplikace musí využívat SharePoint nebo libovolnou výbranou složku k uložení všech dat, což umožní uživatelům snadný přístup k nim. Použití SharePoint nebo síťového disku také zaručí synchronizace dat, která tím pádem nebude ztěžovat vývoj a použití aplikace.

Priorita: vysoká
Složitost: vysoká

FP9: Struktura úložiště

Je klíčové umožnit uživatelům získat přístup k výsledkům testů i mimo aplikaci. Aplikace musí ukládat výsledky testů do jedné složky „VYSLEDKY“ s podsložkami pro jednotlivé lidé. Každá podsložka musí obsahovat:

- Informace o pacientovi: rodné číslo, jméno a příjmení, pohlaví, rok narození, dominantní horní končetina, horní končetina(y) s patologií.
- Tři tabulky s výsledky testů. Potřebné údaje z testů jsou: jméno testujícího, datum testování, čas zahájení testování, poznámky a naměřené hodnoty. Části testů a počty pokusů jsou popsány v manuálech [1, 2, 3]. Každá hodnota musí být uvedená ve srovnání s normou antropometricky a běžným způsobem.
- Textové soubory s výsledky každého testu, určené pro kopírování jejich obsahu do lékařské dokumentaci.
- Videonahrávky všech testů s názvy ve formátu: „PŘÍJMENÍ_JMÉNO_zkrátkaTestu_rok_měsíc_den“, případně končící na „_a“, „_b“ atd. pro více souborů s videem ze stejného testu.

Formát souboru (příp. souborů) s informaci o pacientovi a naměřenými daty musí být podporován programem Microsoft Excel (např. XLSX nebo CSV).

Priorita: vysoká
Složitost: vysoká

FP10: Grafické možnosti nastavení

Aplikace musí umožnit změnu velikosti textu alespoň na tři úrovně a změnu barvy textu na bílou s černým pozadím.

Priorita: střední
Složitost: nízká

FP11: Anonymní testování

Aplikace musí umožňovat anonymní testování. Toto může být implementováno buď jako testování bez ukládání dat, nebo s využitím fiktivního profilu.

Priorita: střední
Složitost: střední

FP12: Pravidla k vyhodnocení

Po každém pokusu aplikace musí uvádět situace popsané v sekcích o řešení situací vzniklých během testování z manuálů [1, 2, 3] s možnosti zaškrtnutí. Pravidla by se měla zobrazovat jak pro úspěšné, tak pro anulované pokusy. Zvolená možnost by se automaticky měla zaznamenat do poznámek daného pokusu.

Priorita: nízká
Složitost: střední

FP13: Propojení kamery s počítačem

Aplikace musí umožňovat použití připojené kamery pro automatické nahrání testů a uložení záznamu.

Priorita: nízká
Složitost: střední

FP14: Integrace modelu strojového vidění

Aplikace musí používat prototyp modelu strojového vidění pro kontrolu výsledků BBT testů. Vyhodnocování se v současné době provádí v reálném čase s pacientem, s kontrolou druhým hodnotitelem podle videa. Integrace tohoto modelu odstranila by potřebu druhého hodnotitele.

Priorita: nízká
Složitost: vysoká

1.3.2 Nefunkční požadavky

Nefunkční požadavky se týkají vlastností systému, které přesahují funkce určené pro uživatele, jako je použitelnost, bezpečnost, spolehlivost a výkon [14].

NP1: Operační systém

Aplikace musí být spustitelná na operačním systému Windows verze 10 nebo novější.

NP2: Intuitivní uživatelské rozhraní

Aplikace musí být hodně intuitivní, aby ji zvládli používat i starší ergoterapeuti nebo obecně lidé, kteří mohou mít problémy s ovládáním počítačů.

NP3: Jednoduchá instalace

Aplikace by měla být navržena tak, aby průměrný uživatel mohl provést instalaci a prvotní konfiguraci bez pomoci IT odborníků. Tedy potřeba nastavení komplikovaných parametrů a nasazení pomocných servisů, databázi atp. je nevhodná.

1.4 Analýza případů užití

Analýza případů užití se zabývá popisem interakcí mezi tzv. aktéry a systémem. Případy užití uvedené dále zahrnují chování aplikace definované v funkčních požadavcích z předchozí podkapitoly. Tohle dává přehled průběhu hlavních procesů aplikace z uživatelského pohledu rozdělených do logických částí pro jejich další návrh.

1.4.1 Případy užití

Případy užití popisují plánované scénáře použití aplikace. Složitější scénáře mají nepovinné nebo opakující se části vyčleněné do samostatných případů užití.

PU1: Přihlášení

Po prvním spuštění aplikace se zobrazí okno pro nastavení úložiště s možností použití SharePoint nebo lokálního úložiště. Pokud je informace o tom již uložena, uživatel si vybere svůj účet (příp. ho přidá). V případě použití cloudového úložiště zadá své heslo s možnosti jej zapamatovat pro další přihlášení. Také autorizace cloudového úložiště může vyžadovat přesměrovaní na odpovídající webovou stránku.

PU2: Přidání účtu uživatele

Uživatel zadá své údaje pro autorizaci v SharePoint: uživatelské jméno (příp. email) a heslo. Zobrazované jméno z uživatelského účtu (obvykle křestní jméno a příjmení) bude uvedeno v zaznamenaných výsledcích testování. Vytvořený účet se přidá k seznamu lokálně uložených uživatelských účtu, které se zobrazují na přihlašovací obrazovce. V závislosti na typu autorizace cloudového úložiště může být uživatel přesměrován na odpovídající webovou stránku.

PU3: Odhlášení

Uživatel přejde na úvodní obrazovku aplikace, klikne na tlačítko pro odhlášení a následně bude vrácen na přihlašovací obrazovku.

PU4: Odstranění účtu uživatele

Uživatel přejde na úvodní obrazovku aplikace a klikne na tlačítko pro nastavení aplikace. Poté klikne na tlačítko pro odstranění lokálně uloženého účtu, potvrdí akci a bude vrácen na přihlašovací obrazovku.

PU5: Nastávení připojení

Uživatel vloží odkaz na SharePoint stránku pracoviště, cestu ke složce pro ukládání dat a zadá případné heslo pro vybranou složku.

PU6: Nastavení aplikace

Uživatel přejde na úvodní obrazovku aplikace a klikne na tlačítko pro nastavení. Tam si může upravit grafické možnosti, nastavení kamery nebo připojení ke cloudovému úložišti a nakonec potvrdí provedené změny.

PU7: Zobrazení manuálů

Uživatel přejde na úvodní obrazovku aplikace a klikne na odkaz pro PDF manuály nebo videomanuály. Následně bude přesměrován na odpovídající webovou stránku.

PU8: Provedení testování

Uživatel přejde na úvodní obrazovku aplikace a klikne na tlačítko pro provedení testování. Dále vybere účet pacienta (příp. ho vytvoří) nebo si vybere anonymního pacienta a spustí testování, které bude rozděleno do různých částí s několika pokusy. Během testování se budou zobrazovat a přehrávat instrukce, a uživatel bude zaznamenávat naměřené hodnoty a poznámky. Po každém pokusu uživatel si zapíše poznámky s možnosti zaškrtnutí situace uvedené v odpovídajícím manuálu. Následně bude mít možnost pokračovat, anulovat pokus nebo ukončit testování. V případně provedení BBT aplikace umožní automatickou kontrolu výsledků. Na konci testování uživatel nahraje manuálně natáčené záznamy nebo aplikace to udělá automaticky pomocí připojené kamery.

PU9: Vytvoření účtu pacienta

Uživatel zadá všechny údaje uvedené v FP9. Tím se vytvoří složka s daty přidaného pacienta.

PU10: Anulování pokusu

Uživatel anuluje poslední pokus v rámci části testování. Pokud byl počet anulací menší, než 3, aplikace umožní opakovat daný pokus. V opačném případě přejde na další pokus nebo ukončí testování.

PU11: Kontrola výsledků BBT

Uživatel použije model strojového vidění pro kontrolu počítání kostek z BBT.

PU12: Zobrazení tabulky výsledků

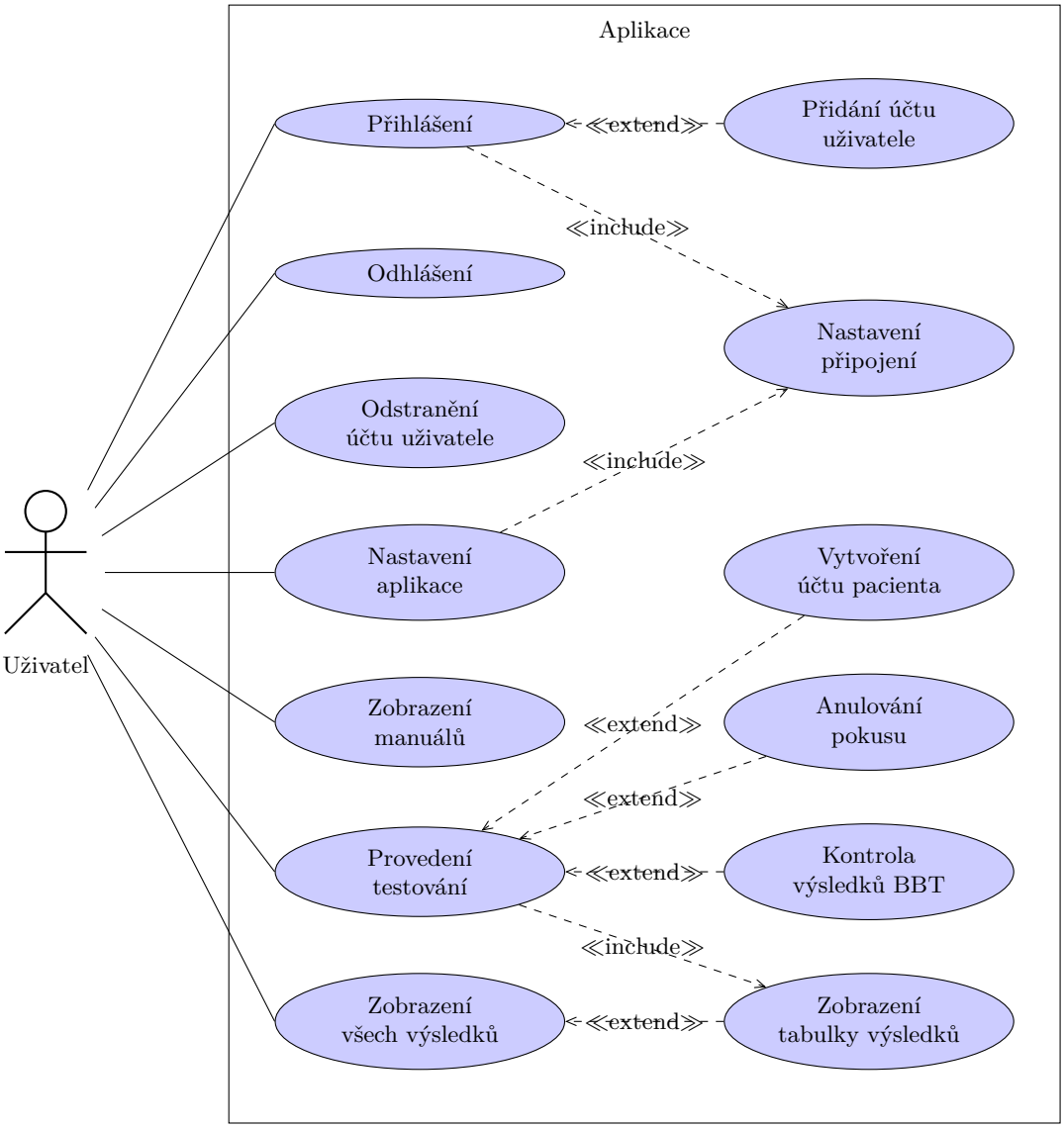
Uživatel vybere jednou z možností zobrazení výsledků testování popsaných v FP7. Aplikace zobrazí tabulku s naměřenými hodnotami a případnými porovnání s normami a předchozím testováním. Zároveň se zobrazí text určený pro lékařskou dokumentaci, který uživatel zkopíruje.

PU13: Zobrazení všech výsledků

Uživatel přejde na úvodní obrazovku a klikne na tlačítko pro zobrazení výsledků. V závislosti na typu úložiště bude přesměrován na odpovídající webovou stránku nebo adresář v souborovém systému.

1.4.2 Diagram případů užití

Tento diagram, vytvořený podle specifikace UML [15], ukazuje všechny plánované případy užití aplikace s předchozí sekce. Nepředpokládá se použití aplikace pacientem, administrátorem pracoviště a dalšími osobami, a proto v diagramu je zobrazen pouze jeden aktér, uživatel (ergoterapeut). Avšak tento diagram znázorňuje nepovinné a společné části scénářů. Důležité je také poznamenat, že data se kterými pracuje aplikace, mohou být použita výzkumníky, ale pro tyto účely budou přístupná mimo aplikaci v souladu s FP9.



Obrázek 1.3 Diagram případů užití

1.5 Volba technologií

V současné době není výběr nástrojů pro tvorbu desktopových aplikací jednoznačný. Existuje mnoho různých řešení, která používají odlišné technologie. V této kapitole jsou popsány některé

z nejpopulárnějších variant spolu s jejich výhodami a nevýhodami. Některé z nich nejsou pro tuto práci podstatné, ale jsou zahrnuty, aby poskytly úplnější přehled o dostupných možnostech.

1.5.1 JavaScript a Electron

Přestože účelem práce je vytváření programu pro operační systém Windows, jednou z možností implementace je webová aplikace zabalená do spustitelného souboru. Jeden z nástrojů pro tento účel je **Electron** [16]. Jeho použití spočívá v otevření webové stránky ve vlastním prohlížeči, což umožňuje vytvářet aplikace bez nutnosti nasazení plnohodnotného webového serveru. Je možné posoudit, že tohle je velice populární způsob tvorby desktopových aplikací, na základě toho, že byl použit pro vývoj spousty všeobecně známých programů. Patří mezi ně např. tyto vývojářské a komunikační nástroje: Visual Studio Code, Postman, Figma, Github Desktop, Microsoft Teams, Slack, Discord [17].

Výhody:

- Web technologie: HTML, CSS a JavaScript dávají velmi flexibilní možnosti stylizace.
- Multiplatformnost: Stejný kód s minimálními úpravami lze použit jak pro desktopové platformy (Windows, MacOS, Linux), tak pro webové aplikace.

Nevýhody:

- Výpočetní náročnost: Zobrazení webové stránky vyžaduje výrazně více prostředků počítače, než analogická nativní řešení.
- Omezenost prostředí: Vývoj v tzv. *sandbox* (česky „pískoviště“) prostředí může přinášet zbytečnou komplexitu a nižší interoperabilitu s počítačovým systémem.

1.5.2 JavaFX

JavaFX [18] je Java framework pro tvorbu uživatelského rozhraní, který přišel jako náhrada za starší Swing framework [19]. JavaFX používá vlastní jazyk pro definice uživatelského rozhraní, FXML, založený na XML. Stylizace komponent je možná pomocí CSS, avšak se specifickými pro tento framework parametry, což znamená, že možnosti jsou omezenější než u běžného webu.

Výhody:

- Multiplatformnost: JavaFX vývoj je založen na použití nezávislého na platformě kódu.
- „Scene builder“ (česky „stavitel scén“): Užitečný nástroj pro vizuální tvorbu UI komponent.

Nevýhody:

- Stárnutí nástrojů: Vzhled a pocit JavaFX může nesplňovat moderní očekávaní UI/UX.
- Aktuálnost: Klesá popularita JavaFX a s tím i podpora komunity [20].

1.5.3 C++ a Qt

Qt [21] je framework určený pro tvorbu uživatelského rozhraní na desktopových, mobilních a vestavěných platformách. Primárně je určen pro C++, ale podporuje také Python s možnosti použití jejich deklarativního jazyka, QML, pro definici uživatelského rozhraní. Přestože byl tento framework vyvinut téměř před třiceti lety, stále je široce používán. Mezi globální společnosti, které používají Qt, patří AMD, Intel, Mercedes-Benz, Bosh, LG, Panasonic a Ubuntu [20].

Výhody:

- Multiplatformnost s nativním pocitem: Qt aplikace mohou věrně napodobovat vzhled a vysoký výkon nativních aplikací.
- Nástroje: Qt poskytuje široký výběr nástrojů pro návrh, vývoj a lokalizaci aplikací.

Nevýhody:

- Složitost: Vývoj uživatelského rozhraní v C++ je obecně považován za složitější než v jiných programovacích jazycích.
- Licencování: Qt Group zavedla licenční poplatky pro komerční použití Qt.

1.5.4 C# a .NET

Podobně jako varianta popsaná v sekci 1.5.1, existuje mnoho C# frameworků pro desktopové aplikace používající stejný základ, v tomto případě .NET framework. Tato kategorie zahrnuje i multiplatformní řešení, avšak většina aplikací si zachovává nativní vzhled dané platformy. Jsou běžně používány např. pro vývoj softwaru od Microsoft a dalších společností, které vyvíjejí aplikace primárně pro operační systém Windows.

Výhody:

- Silná komunita a podpora Microsoftu: Microsoft investuje hodně prostředků v rozvoj nástrojů pro svoji platformu.
- Integrace: Bezproblémová integrace s různými služby od Microsoft jako např. SharePoint.

Nevýhody:

- Orientace na Windows: I když jsou některé frameworky multiplatformní, důraz je kladen na vývoj pro Windows.

1.5.5 Zvolené technologie

Ve velkých IT společnostech závisí volba technologií pro vývoj desktopových aplikací na zkušenostech vývojářského týmu, dostupnosti na různých platformách, výkonu a dalších faktorech. Tato práce má relativně malý rozsah a je určená pro hodně specifickou skupinu osob/pracovišť a jeden operační systém, takže je větší volnost ve výběru. I když je možné uvažovat o rozvoji aplikace pro jiné platformy, podle zadavatele se v praxi téměř nepoužívají jiné operační systémy. Mobilní aplikace by byla vhodná, avšak její návrh a vývoj by se významně lišily, a proto není zvažována.

Pro implementační část práce byl zvolen **Windows Presentation Foundation** [22] framework (dále jen „WPF“) pro .NET. Je to jeden z mnoha podobných frameworků od Microsoft a nástupce již archaických Windows Forms. Ačkoliv WPF by měl být nahrazen technologiemi jako UWP, .NET MAUI a nakonec WinUI 3, stále si udržuje popularitu. Je to sice starší framework, ale „stabilnější“: novější UWP už nedostává podporu pro poslední verze .NET [23] a WinUI 3 je ve fázi velmi aktivního vývoje [24]. Pro definici uživatelského rozhraní používá Extensible Application Markup Language (dále jen „XAML“) založený na XML pro pohodlné oddělení UI a business logiky. Stylizace je dostatečně flexibilní i pro moderní aplikace, ačkoliv může být občas komplikovaná. Další technologie jako např. knihovna pro práci s CSV soubory, budou popsány v příslušných částech následující kapitoly.

Kapitola 2
Návrh

Tato kapitola se zaměřuje na návrh aplikace, navazující na obsah předchozí kapitoly. Je zde prezentována struktura dat a plánovaný vzhled aplikace. Na konci kapitoly je popsán zvolený architektonický vzor.

2.1 Formát exportovaných dat

Vzhledem k tomu, že primární scénář použití aplikace spočívá v postupném ukládání dat pro jejich následný export, je důležité specifikovat všechny potřebné typy dat a jejich formát. To je relevantní také pro zobrazení výsledků v aplikaci, což bude popsáno v další sekci. V tomto kontextu se pro označení dominantní nebo nedominantní horní končetiny bude používat standardní lékařská zkrátka „HK“. Jak je uvedeno v FP7 a FP9, aplikace musí prezentovat naměřené hodnoty ve srovnání s normami, jak antropometrickým, tak běžným způsobem. Antropometrické porovnání spočívá v vypočítání skóre směrodatné odchylky (označované také jako „SD-skóre“ nebo „SDS“) pomocí následujícího vzorce [25]:

$$SDS = \frac{x_i - X}{SD}$$

- x_i - naměřená hodnota.
- X - průměrná hodnota znaku u referenčního souboru.
- SD - směrodatná odchylka referenčního souboru.

Druhý způsob porovnání je založen na Gaussové křivce a interpretuje získané SD-skóre podle následující tabulky [26]:

Tabulka 2.1 Statistické vyhodnocení výsledků

Slovní interpretace dle normy	Intervál
Významně nadprůměrná odchylka od normy	$[+2; +\infty)$
Vysoce nadprůměrné	$[+1,5; +2)$
Nadprůměrné	$[+0,75; +1,5)$
Průměrné	$(-0,75; +0,75)$
Podprůměrné	$(-1,5; -0,75]$
Vysoce podprůměrné	$(-2; -1,5]$
Významně podprůměrná odchylka od normy	$(-\infty; -2]$

Pro NHPT se získané hodnoty dodatečně násobí -1, protože v tomto případě jsou menší hodnoty lepší, než větší. V souladu s FP6 se jako reference budou používat americké testové normy.

2.1.1 Informace o pacientovi

Všechny pacienti budou mít samostatné adresáře s názvy ve formátu „PŘÍJMENÍ_JMÉNO_ID“ pro jejich jednoznačnou identifikaci. U některých pacientů mohou být křestní jméno a příjmení složené s více částí, což je běžná situace např. pro pacienty ze Španělska. V takovém případě se části jména budou oddělovat spojovníky. Ve většině případů bude „ID“ označovat rodné číslo, ale v praxi se také používají různé interní identifikátory, např. číslo pasu nebo fiktivní rodné číslo pro cizince.

Dále je popsán formát tabulky obsahující údaje o pacientovi. Hodnocení výsledků probíhá na základě pohlaví, dominantní ruky podle psání a věku, který bude vypočítán s data narození.

Tabulka 2.2 Formát tabulky s informací o pacientovi

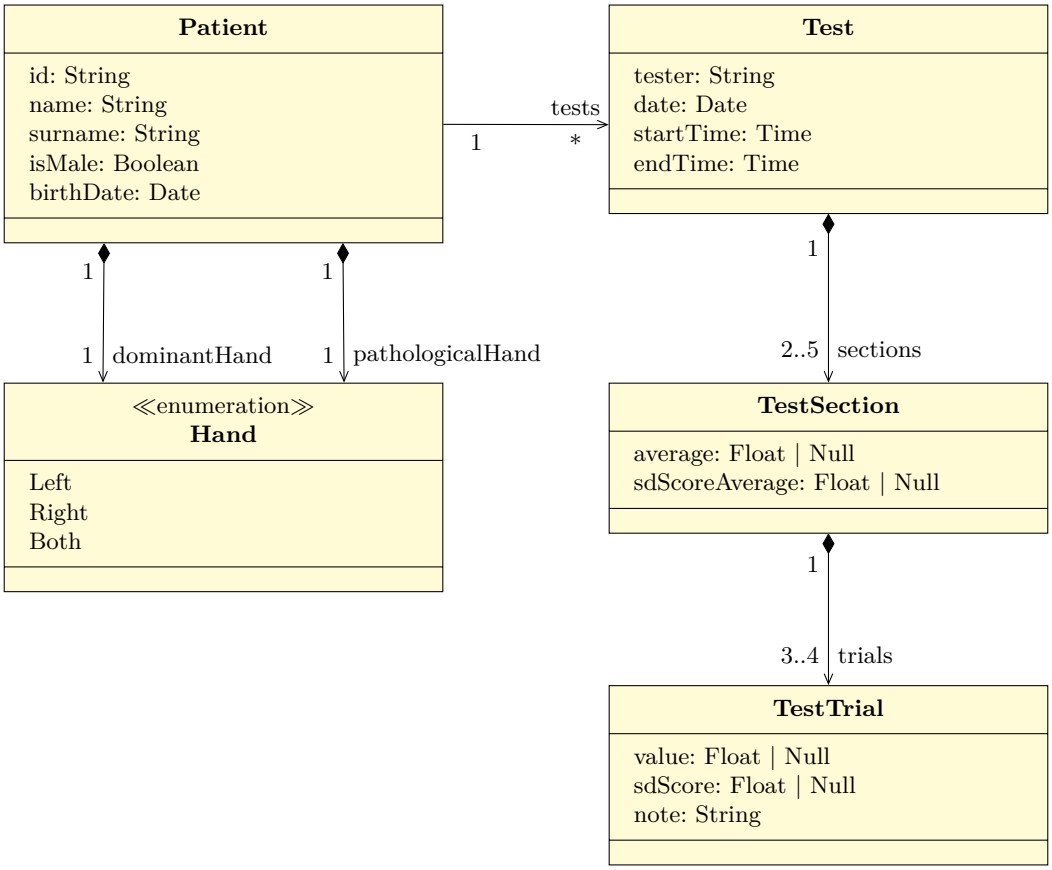
Pohlavi	Datum_narozeni	Dominantni_HK	HK_s_patologii
m/ž	dd.mm.rrrr	levá/pravá	levá/pravá/obě

2.1.2 Formát dat z testů

Konkrétní formát jednotlivých hodnot sbíraných během každého testování je uveden v příloze A. Naměřené hodnoty zahrnují časy v sekundách pro NHPT, počty součástek pro PPT (které se liší v různých částech testu) a počty kostek pro BBT. Pro PPT se navíc přidávají součty hodnot z prvních tří části testu. Číselné hodnoty budou zapisovány jako desetinná čísla s přesnosti na dvě desetinná místa a bez doplnění nulami, což znamená, že budou prezentovány jako celá čísla, pokud neobsahují desetinnou část. Některé hodnoty, teoreticky neomezené, prakticky nabývají relativně malých hodnot: od -2 do 2 pro SD-skóre a desítky sekund pro naměřené hodnoty z NHPT. V případě třetího anulování části testu nebo jeho předčasného ukončení nebudou odpovídající hodnoty v tabulce vyplněny, tj. budou obsahovat prázdné řetězce. Průměry a součty budou vypočítány, pokud bude k dispozici alespoň jedna z potřebných hodnot. V adresáři každého pacienta budou dva podadresáře: jeden s videozáznamy a druhý s textovými soubory určenými pro lékařskou dokumentaci. Názvy souborů budou nasledovat formát specifikovaný v FP9.

2.2 Doménový model

Na základě popisu dat z předchozí sekce byl sestaven **doménový model**, který specifikuje primární struktury dat z programátorského hlediska. Níže je prezentován diagram tohoto modelu:



Obrázek 2.1 Doménový model

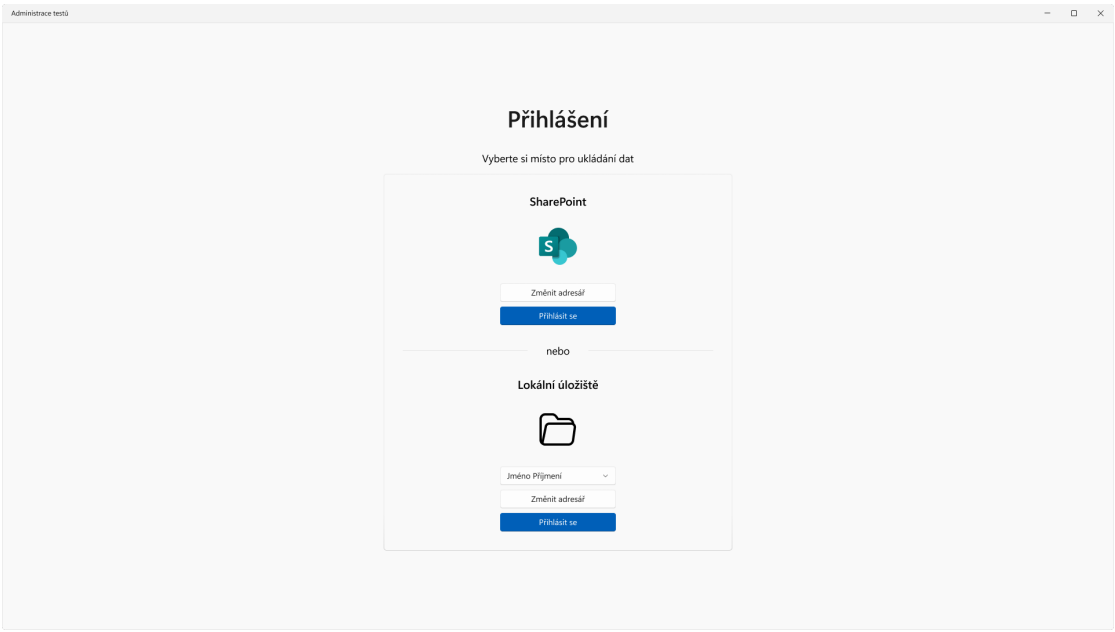
Data ze všech tři typu testů lze sjednotit do jedné datové struktury s proměnlivým počtem sekci a pokusů. Vzhledem k možnosti, že uživatel nesplní část testu, musí být všechny číselné hodnoty schopné přijímat hodnotu *Null* nebo její ekvivalent. Slovní interpretace výsledku dle normy je v podstatě způsob zápisu SD-skóre, takže není součástí navržených datových struktur. *Tester* lze oddělit do samostatné datové struktury obsahující další informace; avšak závisí to na implementaci a může být dostatečný pouhý řetězec s jménem a příjmením. Testy budou doprovázeny videozáznamy s netriviálním umístěním, což bude řešeno zvlášť, ať už jako bajtové proudy při natáčení videa nebo jako soubory při manuálním nahrávání. Program také bude pracovat s velkým množstvím textových a zvukových instrukcí, které nejsou na diagramu znázorněny, protože jejich obsah je konstantní.

2.3 Návrh uživatelského rozhraní

Na základě analýzy z předchozí kapitoly byly vytvořeny **wireframy** (česky „drátěné modely“), které definují obsah obrazovek aplikace. Tyto modely jsou podrobné a měly by odpovídat koncovému vzhledu aplikace. Návrh uživatelského rozhraní většinou odpovídá popisu případů užití z sekce 1.4, ale byla provedena řada úprav pro praktičtější implementaci. Návrh vychází z moderních doporučení Microsoft pro tvorbu Windows aplikaci [27]. Tato sekce podrobně popisuje tři obrazovky spolu s aktualizovanými scénáři použití aplikace. Další wireframy, které nevyžadují detailní popis, jsou uvedeny v příloze B.

2.3.1 Přihlášení

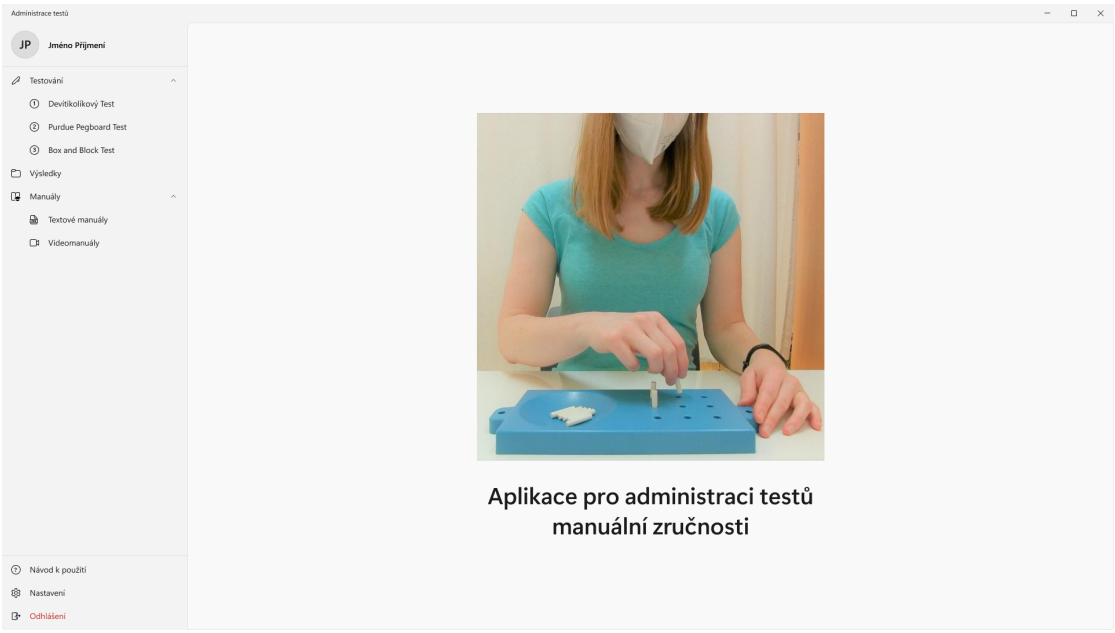
Po spuštění aplikace se uživateli zobrazí přihlašovací obrazovka, která nabízí výběr mezi dvěma typy úložišť. Obě varianty umožňují nastavení adresáře pro export dat, přičemž uživatel může toto nastavení změnit prostřednictvím malého dialogového okna. Pro nastavení SharePoint bude vyžadován webový odkaz na adresář. Přihlášení proběhne na webových stránkách pracoviště nebo v novém okně aplikace, v závislosti na implementaci. Lokální export dat bude využívat absolutní cestu k adresáři v lokálním souborovém systému. Správa lokálních účtu bude založena na jednoduchém seznamu jmén pracovníků s možnosti přidání a odstranění.



Obrázek 2.2 Návrh přihlašovací obrazovky aplikace

2.3.2 Úvodní obrazovka

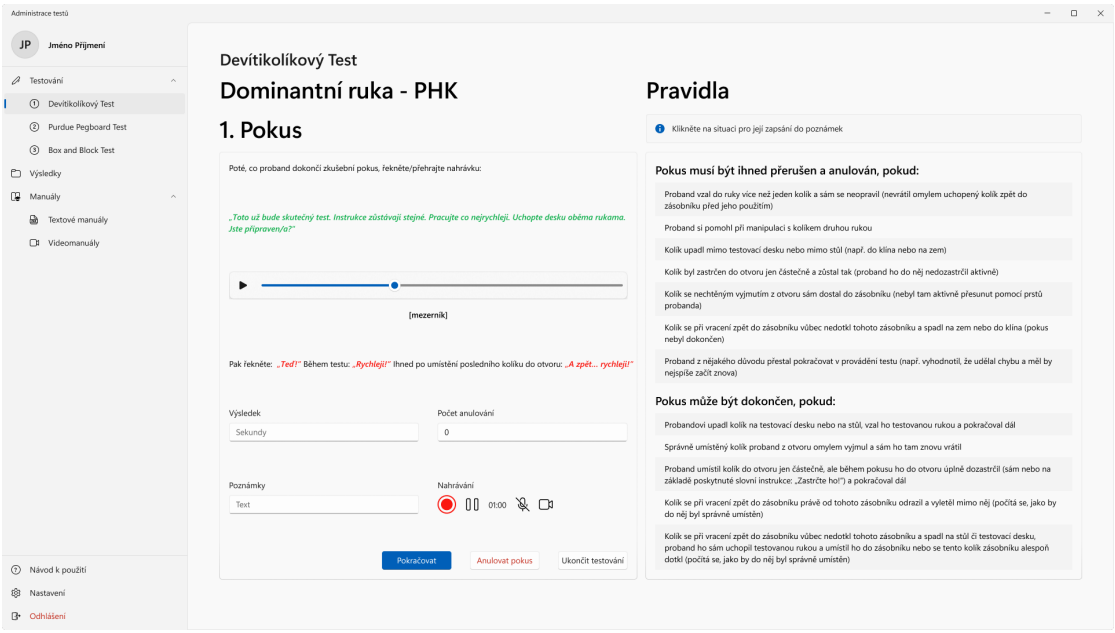
Po přihlášení uživatele se zobrazí úvodní obrazovka aplikace, která obsahuje levý navigační panel. Tento panel zůstává přístupný po celou dobu používání programu pro přihlášené uživatele. Tlačítka pro zobrazení výsledků, manuálů nebo návodu k použití budou pouze odkazovat na externí zdroje. Ostatní varianty jsou *blokující*, tj. budou vyžadovat uživatelské potvrzení o přechodu na jinou stránku v případě neuložených změn.



Obrázek 2.3 Návrh úvodní obrazovky aplikace

2.3.3 Provedení testování

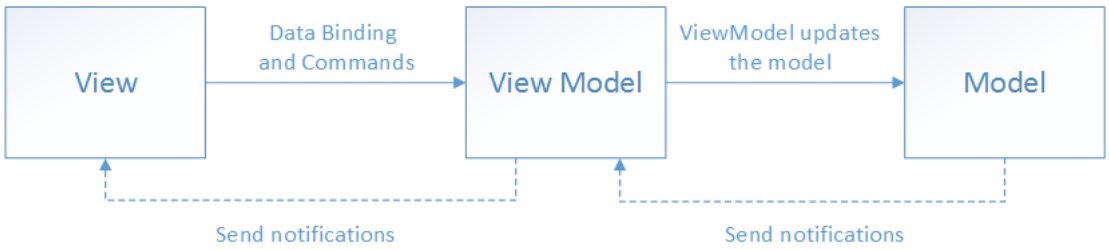
Po výběru typu testu a testovaného pacienta se zobrazí testovací obrazovka, obsahující jednu z mnoha testovacích stránek. Tyto stránky, umístěné v levé části aplikace, budou mít instrukce zvýrazněné různými barvy a doplněné odpovídajícími zvukovými instrukcemi, které lze spustit pomocí mezerníku. Anulování pokusu nezpůsobí přesměrování uživatele na předchozí stránku, ale pouze zvýší počet anulování. Po třech anulováních se textové pole pro zápis naměřené hodnoty zablokuje. Aplikace také umožní nahrávání videa z připojené kamery, které se automaticky ukončí po uložení poslední naměřené hodnoty. Aby se ušetřilo místo na obrazovce, video z kamery bude skryté a dostupné pro zobrazení pomocí odpovídajícího tlačítka. Dále je představen příklad takové stránky pro Devítikolíkový Test.



Obrázek 2.4 Návrh testovací obrazovky aplikace

2.4 Volba architektury aplikace

Pro tvorbu aplikace byla zvolena **Model-View-ViewModel** architektura, která je doporučená pro WPF framework [28]. Je podobná běžně známé architektuře Model-View-Controller, ale na rozdíl od ní, View třídy nekomunikují přímo s Modely obsahující business logiku, pouze pomocí zprostředkovatele.



Obrázek 2.5 Model MVVM [29]

Dále jsou popsané jednotlivé vrstvy:

- Model: Model třídy pracují s daty aplikace, představují doménový model aplikace a zahrnují business a validační logiku. Obvykle se používají ve spojení se službami, které zajišťují přístup k datům.
- View (česky „zobrazení"): View třídy zodpovídají za definování struktury, rozložení a vzhledu toho, co uživatel vidí na obrazovce. Jejich kód by měl definovat pouze vizuální chování bez zahrnutí business logiky.
- ViewModel (česky „model zobrazení"): ViewModel třídy implementují vlastnosti a příkazy, ke kterým může View vytvořit vazbu dat, a upozorňují View na změnu stavu prostřednictvím

události. ViewModel třídy jsou zodpovědné za koordinaci interakcí View se všemi Model třídami, které jsou požadovány [29].

Tato architektura odděluje business logiku od UI logiky, což zjednodušuje znovupoužití komponent aplikace a jejich testování.

Kapitola 3
Implementace

Tato kapitola se zabývá popisem realizace návrhu z předchozí kapitoly. Je zde uvedena celková struktura programu s detailnějším popisem problematických částí implementace. Kód byl napsán v souladu s principy MVVM architektury a další tři podkapitoly popisují jednotlivé vrstvy této architektury.

3.1 Model vrstva

V této podkapitole je popsána vrstva aplikace zaměřená na business logiku, která se nezabývá reprezentaci dat v grafickém uživatelském rozhraní.

3.1.1 Vytváření Test objektů

Jak již bylo zmíněno v sekci 2.2, vybrané testy mají podobnou strukturu a byly sjednoceny do jedné `Test` třídy. Třída `TestBuilder`, která implementuje Builder návrhový vzor, se zabývá vytvářením `Test` objektů a umožňuje postupné přidávání hodnot. Nastavení pacienta, testujícího a času zahájení je triviální, zatímco přidání a následné počítaní potřebných testových hodnot už triviální není. Metoda `AddValue` ve třídě `TestBuilder` používá rozhraní `ITestSectionBuilder` pro výpočet SD-skóre, jak je znázorněno v následujícím kódu:

Výpis kódu 3.1 Metoda pro přidání naměřené hodnoty

```
public ITestBuilder AddValue(float? value, string note)
{
    ...

    var trial = sectionBuilder.BuildTrial(
        value, note, CurrentSection, _patient
    );
    _trials.Last().Add(trial);

    if (!IsFinished && CurrentTrial == sectionBuilder.TrialCount)
    {
        _trials.Add([]);
    }

    return this;
}
```

Implementace rozhraní `ITestSectionBuilder` se vybírá v závislosti na typu testu pomocí třídy `TestBuilderFactory`.

Třídy implementující `ITestSectionBuilder` definují typ testu, počet subtestů, počet pokusů v každém subtestů a metody pro vytváření `TestTrial` a následně `TestSection` objektů. Samotné výpočty jsou přiděleny třídě `TestCalculator`, která využívá `ITestNormProvider` pro získání sad norem. Jedním z uživatelských požadavků (viz FP6) byla možnost změnit normy po dokončení odpovídajícího výzkumu. Pro splnění tohoto stačí upravit odpovídající konstanty:

Výpis kódu 3.2 Definice norem v kódu

```
public class NhptTestNormProvider : ITestNormProvider
{
    private static readonly SortedDictionary<int, TestNorm>
        MaleDominantNorms = new()
        {
            [20] = new TestNorm(1.9f, 16.1f),
            [25] = new TestNorm(1.6f, 16.7f),
            [30] = new TestNorm(2.5f, 17.7f),
            [35] = new TestNorm(2.4f, 17.9f),
            [40] = new TestNorm(2.2f, 17.7f),
            [45] = new TestNorm(2.3f, 18.8f),
            [50] = new TestNorm(1.8f, 19.2f),
            [55] = new TestNorm(2.6f, 19.2f),
            [60] = new TestNorm(2.6f, 20.3f),
            [65] = new TestNorm(2.9f, 20.7f),
            [70] = new TestNorm(3.3f, 22.0f),
            [75] = new TestNorm(4.0f, 22.9f)
        };

    ...
}
```

Aktuálně normy pro PPT nejsou rozdělené podle věku a pohlaví, což znamená, že odpovídající proměnné obsahují mnoho hodnot, které se opakují. Možnost výběru norem mohla být implementována pomocí konfiguračních souborů, ale protože se očekává pouze jedna změna po vytvoření českých norem, jsou normy definovány přímo v kódu.

3.1.2 CSV konverze

Pro zápis dat so CSV souborů se využívá knihovna CsvHelper [30]. V kódu jsou definovány datové třídy, jejichž atributy odpovídají sloupcům CSV tabulek. Dále je příklad definice několika vlastností takové třídy:

Výpis kódu 3.3 Příklad definice struktury CSV souboru

```
[Delimiter(Delimiter)]
[CultureInfo(Culture)]
public class PatientCsvRecord
{
    ...

    [Index(3)]
    [Name("Pohlavi")]
    [BooleanTrueValues("m")]
    [BooleanFalseValues("ž")]
    public required bool IsMale { get; init; }

    [Index(4)]
    [Name("Datum_narozeni")]
    [Format(DateFormat)]
    public required DateOnly BirthDate { get; init; }

    [Index(5)]
    [Name("Dominantni_HK")]
    [TypeConverter(typeof(HandCsvStringConverter))]
    public required Hand DominantHand { get; init; }

    ...
}
```

Je vidět, že tohle umožňuje jednoduché nastavení formátovaní různých typů hodnot. Může se zdát, že postupné zvětšování indexů je zbytečné, avšak tohle zajišťuje správné pořadí sloupců. Důvodem je, že .NET negarantuje zachování pořadí členů třídy při použití reflexe. Tyto třídy jsou využívány společně s odpovídajícími konvertory, které je převádějí na typy používané v ostatních částech programu a zpět. Hlavní nevýhodou tohoto řešení je duplikace kódu, způsobená velkým počtem sloupců v tabulkách s výsledky testů. První verze tohoto řešení používala dynamickou generaci CSV sloupců a jejich názvů pomocí `ClassMap<T>` tříd, což tento problém řešilo, ale finální varianta je výrazně jednodušší a lépe čitelná.

3.1.3 Import a export dat

Pro import a export `Patient` a `Test` objektů se využívá `TestStorage` třída, která implementuje Facade návrhový vzor a sjednocuje několik objektů používaných pro tyto účely. Například export

výsledku testu zahrnuje použití odpovídajících metod pro export CSV souborů, dokumentačních textu a videí. Exportéry dokumentačních textu a videí jsou triviální: první vkládá potřebné hodnoty do textové šablony a druhý kopíruje zadané soubory do určeného adresáře. Import a export dat z CSV souborů navazuje na konvertory z předchozí sekce:

Výpis kódu 3.4 Export výsledku testu do CSV souboru

```
private void _exportTest(Patient patient, Test test, string filePath)
{
    var fileExisted = File.Exists(filePath);
    using var stream = File.Open(filePath, FileMode.Append);
    using var writer = new StreamWriter(stream, new UTF8Encoding(true));

    var config = _getConfig(test.Type);
    config.HasHeaderRecord = !fileExisted;
    using var csvWriter = new CsvWriter(writer, config);

    _writeRecord(csvWriter, patient, test);
}
```

Pomocné metody vybírají konkrétní konvertor a typ CSV záznamu podle typu testu. CSV soubory se otevírají pomocí režimu, který nedovoluje přepisování existujících dat, ale i bez toho pouze přidává další řádky. Tento přístup zaručuje, že výsledky testů se neztratí, např. po manuální změně formátu souboru.

3.1.4 Připojení k SharePoint

Během vývoje bylo zjištěno, že nejjednodušším a zároveň nejpraktičtějším řešením problému připojení k SharePoint je vyhnout se jeho přímé integraci. Ačkoli použití SharePoint API nebo příslušných knihoven představuje sofistikované řešení, pro účely této práce se ukázalo jako zbytečné. Takový přístup vyžaduje registraci aplikace v systému pracoviště, složité nastavení na straně uživatele a komplexní testování, které lze provést pouze s existující instanci tohoto systému.

SharePoint standardně nabízí možnost „synchronizace“ [31], která po několika kliknutích zpřístupní vybraný adresář v lokálním souborovém systému a umožní s ním provádět všechny potřebné manipulace. Toto řešení plně vyhovuje uživatelským požadavkům a je pro tuto práci zcela dostatečné.

Přestože bylo toto řešení zvoleno pro aktuální fázi vývoje, rozšíření programu o podporu dalších typů cloudových úložišť představuje jednu z možnosti pro jeho budoucí rozvoj.

3.1.5 Přehrávání instrukcí

Správu audiopřehrávačů zajišťuje `AudioInstructionService` třída, která je také zodpovědná za vytváření `MediaPlayer` objektů z příslušných MP3 souborů. Tato služba umožňuje definování funkcí pro „globální“ kontrolu stavu aktuálního audiopřehrávače.

Výpis kódu 3.5 Služba pro přehrávání zvukových instrukcí

```
public class AudioInstructionService
{
    ...

    public void SetPlayerActions(
        Action? onResume, Action? onPause, Action? onStop
    )
    {
        _onResume = onResume;
        _onPause = onPause;
        _onStop = onStop;
    }

    public void Resume()
    {
        _isPlaying = true;
        _onResume?.Invoke();
    }

    ...
}
```

Toto nastavení poskytuje možnost reagovat na uživatelské akce, jako je změna testovací stránky, stisknutí mezerníku nebo výběr jiné audio instrukce. Namísto přímé manipulace s `MediaPlayer` objekty, třída používá delegáty (`Action`), což umožňuje např. změnu vizuálního stavu souvisejících komponent uživatelského rozhraní.

3.1.6 Nahrávání videozáznamu

Nahrávání videozáznamu je komplexní část implementace, kterou řídí `VideoRecorderService` třída. Tato třída využívá pro nahrávání videa knihovnu OpenCvSharp [32] a zajišťuje kontrolu nad uživatelsky nastavenou kamerou a mikrofonem. Metoda `StartCamera` spouští asynchronní operaci zápisů snímků, která pokračuje až do volání metody `StopCamera`. Během této operace je pravidelně vyvolávána událost o dostupnosti nového snímku, který lze zobrazit v aplikaci:

Výpis kódu 3.6 Získání snímku z připojené kamery

```
using var mat = _capture.RetrieveMat();
if (mat.Empty())
{
    continue;
}

var bitmapSource = mat.ToBitmapSource();
bitmapSource.Freeze();
NewFrameAvailable?.Invoke(bitmapSource);
```

Následuje opakování tohoto procesu s minimálními intervály. Pokud je nahrávání aktivní, snímky

se zapisují do objektu `VideoWriter` a tím pádem i do dočasného MP4 souboru:

Výpis kódu 3.7 Zápis videa do souboru

```
if (!IsRecording || IsPaused)
{
    continue;
}

_writer?.Write(mat);
```

Pro nahrávání zvuku byla zvolena knihovna NAudio [33], jelikož knihovna OpenCvSharp nepodporuje záznam zvuku a nebyla nalezena lepší alternativa. Objekt `WaveIn` se inicializuje s nahráváním a reaguje na události dostupnosti zvukových dat, která se zapisují do dočasného souboru, pokud je nahrávání aktivní:

Výpis kódu 3.8 Zápis audia do souboru

```
private void _writeAudio(object? sender, WaveInEventArgs e)
{
    if (_waveFileWriter is null || !IsRecording || IsPaused)
    {
        return;
    }

    _waveFileWriter.Write(e.Buffer, 0, e.BytesRecorded);
    _waveFileWriter.Flush();
}
```

Kombinace audio a video souborů do jednoho souboru se provádí prostřednictvím nástroje FFMPEG, který je integrován pomocí příslušné knihovny [34] a zvlášť uložených spustitelných souborů:

Výpis kódu 3.9 Kombinování video a audio souborů

```
FFMpeg.ReplaceAudio(
    TempVideoFilePath, TempAudioFilePath, TempRecordingFilePath
);
```

Po dokončení nahrávání lze tento soubor přesunout do určeného adresáře pacienta pomocí třídy `VideoExporter`, a při dalším nahrávání se tento soubor automaticky odstraní.

3.1.7 Konfigurace

Konfiguraci aplikaci zajišťuje třída `ConfigurationService`. Tato služba ukládá „data.json“ soubor s konfiguračními daty do stejného adresáře, kde se nachází spustitelný soubor aplikace. Tato data jsou načítána při spuštění aplikace a příslušný soubor je aktualizován vždy, když dochází ke změně některého z uživatelských nastavení. Níže je uveden příklad takového souboru:

Výpis kódu 3.10 Příklad obsahu konfiguračního souboru

```
{
  "LocalTestDataPath": "D:\\Vysledky",
  "LocalUsers": [
    "Jan Nov\u00E1k",
    "Marie Svobodov\u00E1"
  ],
  "CurrentUser": "Jan Nov\u00E1k",
  "ApplicationTheme": "Light",
  "FontSize": "14",
  "CameraId": "0",
  "MicrophoneId": "0"
}
```

3.2 ViewModel vrstva

V této podkapitole je popsána vrstva aplikace, která využívá třídy z předchozí podkapitoly. Tato vrstva zpřístupňuje potřebná data a příkazy pro jejich ovládání, čímž umožňuje interakci s uživatelem.

3.2.1 Dependency Injection

Základem aplikace je použití použití Dependency Injection, které je realizováno prostřednictvím příslušné .NET knihovny od Microsoft. Tohle propojuje mezi sebou třídy s business logikou a vkládá je do ViewModel tříd. Většina Model tříd a některé ViewModel třídy jsou vytvářeny přímo tímto mechanismem. Pro složitější třídy, jejichž závislosti se vyřizují za běhu programu, je využit Factory návrhový vzor. Tohle jsou například třídy, které závisí na zvoleném typu testu a pacientovi. Tento vzor navíc minimalizuje nežádoucí závislosti v třídách, které tyto objekty vytvářejí.

3.2.2 Navigace

Aplikace využívá „ViewModel-first“ přístup, což znamená, že životní cyklus ViewModel objektů je nezávislý na View objektech. ViewModel objekty zveřejňují další ViewModel objekty, které reprezentují vnitřní komponenty aktuálního uživatelského rozhraní. Následující příklad demonstruje, jak `TestingViewModel` řídí navigaci mezi svými stránkami:

Výpis kódu 3.11 Implementace navigace v rámci TestingViewModel

```
private void _onOpenPatientChoice() =>
    CurrentViewModel = new PatientChoiceViewModel(
        _testStorage, _onStartTesting, _onOpenAddPatient
    );

private void _onOpenAddPatient() =>
    CurrentViewModel = new NewPatientViewModel(
        _testStorage, _onOpenPatientChoice
    );

private void _onStartTesting(Patient patient) =>
    CurrentViewModel = _testConductionViewModelFactory.Create(
        patient, _testType, _onShowResults
    );

private void _onShowResults(Patient patient, Test test) =>
    CurrentViewModel = _resultsViewModelFactory.Create(
        patient, test, _onSaveTest
    );

private void _onSaveTest(
    Patient patient, Test test, List<string> videoFilePaths
)
{
    _testStorage.AddTest(patient, test, videoFilePaths);
    _onOpenPatientChoice();
}
```

V tomto případě interní ViewModel objekty přijímají funkce pro změnu stavu vnějšího objektu po dokončení své činnosti. Ačkoli to není jediný způsob řešení, vybraný přístup usnadňuje použití Dependency Injection a testování. Alternativně by mohly View objekty využívat příkazy navigační služby s parametry definujícími cílové stránky pro přechod, což by ale mohlo komplikovat správu závislosti.

Jedinou výjimkou z tohoto pravidla je zobrazení dialogových oken, které jsou předávány jako parametry příslušných metod a využívají stejný ViewModel objekt, který je vytváří. Tento přístup je zvolen, protože zobrazení dialogového okna se konceptuálně liší od výměny komponent v existujícím okně. Níže je uveden příklad kódu, který zobrazuje dialogové okno:

Výpis kódu 3.12 Příklad zobrazení dialogového okna

```
private async void _onOpenCameraDialog(ContentDialog? content)
{
    if (content is null)
    {
        throw new ArgumentException("Content is null");
    }

    content.DataContext = this;
    await _contentDialogService.ShowAsync(content, CancellationToken.None);
}
```

3.2.3 Provedení testování

Mezi hlavní funkce programu patří, mimo jiné, zobrazení textových instrukcí, přehrávaní zvukových instrukcí, poskytování pravidel pro řešení situací vznikajících během testování, nahrávání videozáznamů, zápis naměřených hodnot, anulování pokusů a předčasné ukončení testování. Tyto úkoly jsou spravovány pomocí třídy `TestConductionViewModel`, která pro většinu těchto funkcí využívá zmíněné výše třídy.

Anulování pokusů je implementováno jako zvýšení počtů anulovaných pokusů a blokování zápisů naměřených hodnot po dosažení maximálního počtu povolených pokusů. Příkaz pro předčasné ukončení testu volá metodu `Build` třídy `TestBuilder`, která automaticky doplňuje zbytek testu prázdnými hodnotami. Jednou z výzev v této části implementace bylo zobrazení instrukcí pro dlouhé textové manuály. Stránky s instrukcemi, zobrazující informace pro jednotlivé pokusy, mají variabilní počet textových bloků, zvukových nahrávek a různé formátování textu, což znemožňuje jednoduchou výměnu textu. Celkově aplikace zobrazuje 31 stránku, které jsou rozděleny do 13 typů s jednotnou strukturou.

Výber pacienta pro testování probíhá na stránce zobrazené na obrázku B.2. V souladu s požadavkem FP11 se při výběru prázdného adresáře pro výsledky automaticky přidává anonymní pacient.

3.3 View vrstva

Daná podkapitola popisuje vrstvu, která je zodpovědná za definování rozložení grafického uživatelského rozhraní a za nastavení vazeb mezi vlastnostmi ViewModel tříd a vizuálními komponentami. Pro definici uživatelského rozhraní jsou využívány XAML soubory. Každá XAML komponenta je spojena s tzv. „Code-Behind“ třídou v C#. Ve výsledném programu jsou tyto třídy skoro prázdné, s výjimkou pár řádků kódu, a využívají vlastnosti příslušných ViewModel tříd, čímž se zabraňuje fragmentaci aplikační logiky.

Přestože jsou „Code-Behind“ třídy minimalizované, mohou být využity pro realizaci složitějších vizualizaci, které nejsou realizovatelné pouze pomocí XAML, nebo pro tvorbu znovupoužitelných komponent, které nemají odpovídající ViewModel třídy. Dále je uveden příklad takové třídy s netriviální UI logikou:

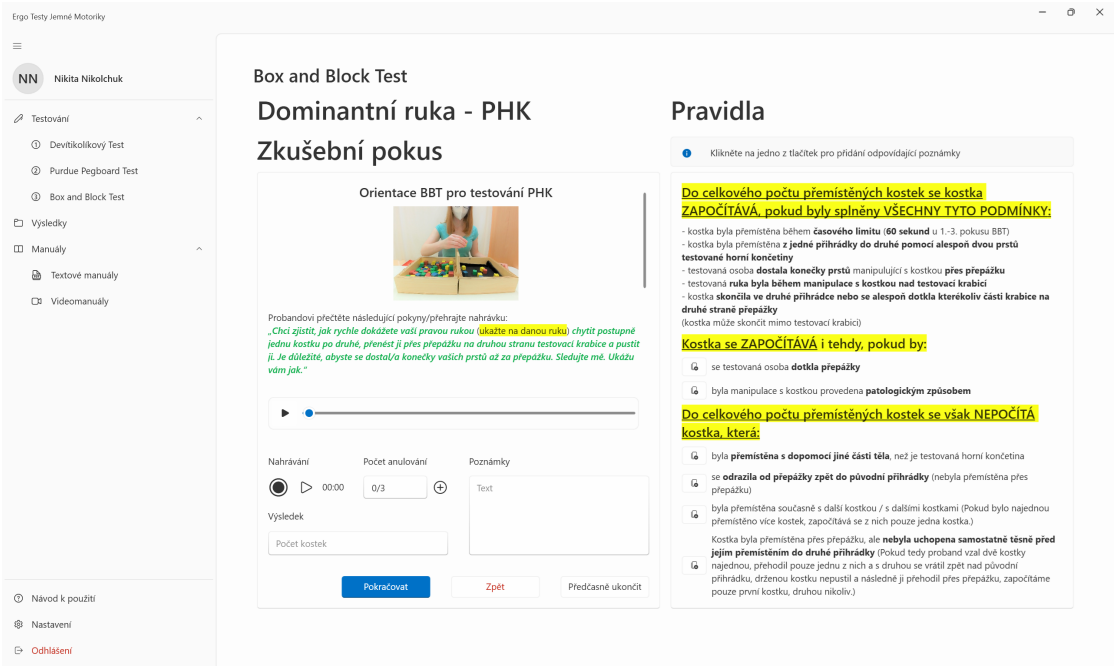
Výpis kódu 3.13 Zobrazení audiopřehrávače

```
public partial class InstructionPlayer
{
    public InstructionPlayer()
    {
        InitializeComponent();
        DataContextChanged += _onDataContextChanged;
    }

    private void _onDataContextChanged(
        object _, DependencyPropertyChangedEventArgs e
    )
    {
        if (e.NewValue is InstructionPlayerViewModel viewModel)
        {
            viewModel.OnPlayStateChanged += BringIntoView;
        }
    }
}
```

V tomto příkladu audiopřehrávač reaguje na změnu stavu odpovídajícího ViewModel objektu a zobrazuje se posouváním plochy s instrukcemi. Tohle umožňuje přechod k audiopřehrávači na konci stránky po stisknutí mezerníku.

Vzhled programu je celkem podobný jeho návrhu. Níže je představen snímek obrazovky obsahující instrukce pro BBT:



Obrázek 3.1 Výsledný vzhled aplikace

3.3.1 WPF UI

Pro tvorbu uživatelského rozhraní byla zvolena knihovna WPF UI [35], která obsahuje předefinované standardní styly WPF komponent, nabízí řadu vlastních komponent, různé barvy a několik dalších užitečných funkcí. Tato knihovna vizuálně napodobuje styl Windows 11, což bylo využito k dosažení moderního vzhledu aplikace, jak byl navržen v předchozí kapitole.

Obrázek 3.2 WPF UI knihovna [35]

3.3.2 Datové šablony

V sekci 3.2.2 bylo popsáno jak je řešen výběr ViewModel objektů pro zobrazení. Tento proces je realizován pomocí datových šablon, které automaticky nahrazují vybrané ViewModel objekty odpovídajícími komponentami uživatelského rozhraní. Tyto šablony jsou definovány globálně pro celý program a zároveň nastavují `DataContext` pro nově vytvořené komponenty, což umožňuje vytváření vazeb mezi daty a uživatelským rozhraním.

Výpis kódu 3.14 Příklad definice XAML datové šablony

```
<DataTemplate DataType="{x:Type viewModels:LoginScreenViewModel}">
    <views:LoginScreen />
</DataTemplate>
```

3.3.3 Grafické možnosti nastavení

V souladu s FP10, aplikace umožňuje uživatelům nastavit velikost písma a volbu mezi světlým a tmavým režimem. Globální základní velikost písma je stanovena na 14 bodů. Všechny textové styly využívají `AdditionConverter` a pomocné třídy pro dynamickou vazbu, které k této základní velikosti přidávají další hodnoty. Textové styly jsou pojmenovány podle stylů z knihovny WPF

UI, čímž dochází k jejich přepisu. Velikost písma je možné nastavit při spuštění aplikace pomocí `ConfigurationService` nebo později změnou v globálním objektu `Application`.

Změna režimu aplikace je zprostředkována pomocí `ApplicationThemeManager` z knihovny WPF UI. Standardní komponenty, jako jsou tlačítka a textová pole, používají tento mechanismus implicitně, zatímco ostatní komponenty mají explicitně definované dynamické barvy. Příklad nastavení dynamických barev lze vidět v následujícím XAML kódu:

Výpis kódu 3.15 Příklad nastavení dynamických barev

```
<Border
    Padding="32,16"
    CornerRadius="4"
    Background="{DynamicResource ControlSolidFillColorDefaultBrush}"
    BorderBrush="{DynamicResource ControlElevationBorderBrush}"
    BorderThickness="1">

    ...

</Border>
```

Kapitola 4
Testování

Tato kapitola je věnována testování výsledné aplikace. V první podkapitole je popsáno, jak bylo řešeno automatické testování aplikace. Druhá podkapitola udává hodnocení pilotní verze programu cílovými uživateli.

4.1 Vývojářské testování

Pro automatické testování aplikace byly využity knihovny xUnit [36] a Moq [37]. Všechny testy mají stejnou strukturu: definici vstupných a očekávaných dat s případným nastavením `Mock` objektů pro simulaci závislostí třídy, následně použití testované metody a na konci kontrolu výsledků. Většina testů má atribut `[Fact]`, který pouze označuje jednotkový test. Ostatní testy mají atribut `[Theory]`, který umožňuje definování seznamu parametrů pro opakované spuštění testované funkce. Níže je příklad takové metody pro testování počítání SD-skóre:

Výpis kódu 4.1 Testování počítání SD-skóre

```
[Theory]
[InlineData(19, 10, false, null)]
[InlineData(20, 10, false, 1.0f)]
[InlineData(23, 10, false, 1.0f)]
[InlineData(25, 10, false, 2.0f)]
[InlineData(25, 10, true, -2.0f)]
public void SdScore_ReturnsExpectedSdScore(
    int age, float value, bool isInverted, float? expectedSdScore
)
{
    ...
}
```

Testy pokrývají malou část kódu a tento problém musí být vyřešen v případě budoucího rozvoje programu. Avšak testování UI logiky a některých částí aplikace, jako je služba pro nahrávání videí, je velmi komplexní, pokud vůbec možné, a pravděpodobně se automaticky nebude provádět i když na tomto programu bude někdo dále pracovat.

4.2 Uživatelské testování

Tato podkapitola podrobně popisuje provedené uživatelské testování, včetně přípravy, průběhu a vyhodnocení zpětné vazby získané od uživatelů. Uživatelské testování bylo realizováno ve spolupráci se zadavatelem v prostorách Kliniky rehabilitačního lékařství 1. LF UK a VFN v Praze. Celkem se testování zúčastnilo pět osob, z toho tři ergoterapeutky a dvě studentky oboru ergoterapie. Všichni účastnici mají praktické zkušenosti s prováděním vybraných testů buď v rámci své klinické praxe, nebo v rámci studia.

4.2.1 Testovací scénáře

Provedení jednotlivých testů manuální zručnosti typicky vyžaduje účast dvou osob: jedné v roli testujícího a druhé v roli pacienta. Během plánovaní uživatelského testování bylo dohodnuto, že tři ergoterapeutky budou primárně vystupovat v rolí testujících a dvě studentky v roli pacientů. Z tohoto důvodu bylo vytvořeno tři testovací scénáře.

Většina průběhu použití aplikace je stejná: začíná přihlášením, následuje výběr testu, samotné testování, zobrazení výsledků a ukončení aplikace nebo přechod k dalšímu testování. Testovací scénáře byly rozděleny podle typu testu a představují různé variace tohoto standardního postupu.

Samotné instrukce pro uživatele jsou k dispozici v příloze C.

4.2.2 Průběh uživatelského testování

Výsledná aplikace je distribuována ve formě ZIP souboru, který obsahuje spustitelný soubor se všemi jeho závislosti a je připraven k okamžitému spuštění bez nutnosti instalace. Program byl úspěšně spuštěn na pracovním počítači kliniky a nevyskytly se při tom žádné závažné problémy. Uživatele obdrželi instrukce popsané v předchozí sekci a postupně se střídaly. Během testování byly zaznamenávány poznámky týkající se částí aplikace, kde se uživatele zasekli. Také v různých místech to uživatele přerušovali, aby poskytnout zpětnou vazbu o tom, co se jim na aplikaci líbilo nebo nelíbilo. Byla tam i diskuze mezi uživateli o tom, co by chtěli vidět ve finální verzi programu.

4.2.3 Zpětná vazba

Kromě zpětné vazby získané během osobních setkání, uživatele poskytli také dodatečnou v písemné formě. Níže je uvedeno shrnutí hlavních pozitivních a negativních připomínek.

Co se líbilo:

- Příjemné uživatelské prostředí.
- Vhodně umístěné instrukce, zvýraznění textu stejné jako v manuálech.
- Možnost generování dat pro práci i mimo aplikaci.
- Možnost nahrávání videozáznamu během testování.
- Možnost si nastavit velikost písma, zapnout tmavý režim.

Co je potřeba upravit:

- Přidat nutnost potvrzení některých akcí.
- Umožnit vracení zpět.
- Zvětšit textové pole pro poznámky.

- Změnit kopírování poznámek, aby se využíval dvojklik nebo pomocná tlačítka.
- Zlepšit intuitivnost přihlašování, posouvání plochy s instrukcemi a anulování pokusů.
- Doplnit chybějící nápisy, obrázky a zvýraznění textu, opravit pár překlepů.
- Optimalizovat rozložení uživatelského rozhraní pro menší monitory.

Celkově aplikace splňuje požadavky ergoterapeutů a nebyly zaznamenány žádné závažné problémy. Testování se ukázalo být velmi přínosné, poskytlo cenné informace o aspektech programu, které vyžadují dopracování.

Po uživatelském testování byly vyřešeny téměř všechny zmíněné problémy. Intuitivnost použití některých částí aplikace byla zlepšena. Je možné, že stále není dostatečně dobrá, ale jedná se o subjektivní kritérium. Byl přidán i dříve plánovaný návod k použití, který popisuje neočividné části uživatelského rozhraní. Například je tam informace o tom, že nahrávání videozáznamu vytváří pouze jeden soubor, i když může být přerušováno.

Další částečně vyřešený problém je adaptace uživatelského rozhraní pro monitory různých rozměru. Rozměr textu a velké části UI komponent nezáleží na používaném zařízení, což je problematické pro menší monitory, které nemohou zobrazit všechno najednou. Tento problém byl původně řešen a následně dopracován tak, že velké komponenty, jako jsou plochy s instrukcemi, pravidly a tabulkami, mají obsah, který je možné posouvat. Samotné tyto komponenty a mnoho dalších mají dynamický rozměr. Dokonce je možné i schovat levý navigační panel pro zvětšení celkové dostupné plochy. Nicméně rozměr mnoha komponent je fixní nebo dynamický s nějakou minimální hodnotou, a tohle může způsobovat nepatrné problémy.

Kapitola 5
Závěr

Výsledkem této práce je aplikace, která je plně funkční a jejíž užitečnost byla potvrzena během uživatelského testování. Skoro všechny uživatelské požadavky byly splněny. Jediný nesplněný požadavek je méně prioritní integrace modelu strojového vidění pro automatickou kontrolu výsledků BBT. Tento model, i když funkční, není v současné době připraven pro praktické použití. Jeho integrace by vyžadovala dělení videozáznamu na segmenty odpovídající jednotlivým pokusům a manuální spuštění pro každý pokus, což je komplikované z vývojářského hlediska a nepraktické z uživatelského pohledu.

Práce byla přínosná tím, že poskytla zkušenost vývoje většího projektu, který zahrnoval jeho důkladné plánování. Také to bylo užitečné seznámení s novými technologií.

Mezi možnosti budoucího rozvoje aplikace patří integrace výše zmíněného modelu strojového po jeho zdokonálení. Další možnosti zahrnují propojení s různými typy cloudových úložišť nebo dokonce s lékařskými systémy. Rozšíření aplikace do dalších jazyků by také bylo možné, avšak komplikované, neboť kromě překladu textu by vyžadovalo i adaptace logiky aplikace vzhledem k odlišným strukturám manuálů pro testy používané v jiných státech. Také návrh aplikace by mohl být využit alespoň částečně pro vývoj podobné mobilní aplikace.

Podlě dohody se zadavatelem, výsledná aplikace bude dostupná ke stažení na webových stránkách Kliniky rehabilitačního lékařství VFN a 1. LF UK.

Příloha A

Formát exportovaných tabulek

Tabulka A.1 Formát exportovaných dat pro NHPT

Sloupec	Formát
Testujici	Jméno Příjmení
Datum	dd.mm.rrrr
Cas_zahajeni	hh:mm
Cas_ukonceni	hh:mm
Dom_zkus	$x, x \in \mathbb{R} \wedge x \geq 0$
Dom_1_pokus	$x, x \in \mathbb{R} \wedge x \geq 0$
Dom_2_pokus	$x, x \in \mathbb{R} \wedge x \geq 0$
Dom_3_pokus	$x, x \in \mathbb{R} \wedge x \geq 0$
Dom_prumer	$x, x \in \mathbb{R} \wedge x \geq 0$
Nedom_zkus	$x, x \in \mathbb{R} \wedge x \geq 0$
Nedom_1_pokus	$x, x \in \mathbb{R} \wedge x \geq 0$
Nedom_2_pokus	$x, x \in \mathbb{R} \wedge x \geq 0$
Nedom_3_pokus	$x, x \in \mathbb{R} \wedge x \geq 0$
Nedom_prumer	$x, x \in \mathbb{R} \wedge x \geq 0$
Dom_zkus_SDS	$x, x \in \mathbb{R}$
Dom_1_pokus_SDS	$x, x \in \mathbb{R}$
Dom_2_pokus_SDS	$x, x \in \mathbb{R}$
Dom_3_pokus_SDS	$x, x \in \mathbb{R}$
Dom_prumer_SDS	$x, x \in \mathbb{R}$
Nedom_zkus_SDS	$x, x \in \mathbb{R}$
Nedom_1_pokus_SDS	$x, x \in \mathbb{R}$
Nedom_2_pokus_SDS	$x, x \in \mathbb{R}$
Nedom_3_pokus_SDS	$x, x \in \mathbb{R}$
Nedom_prumer_SDS	$x, x \in \mathbb{R}$
Dom_zkus_porovnani	Slovní interpretace dle normy
Dom_1_pokus_porovnani	Slovní interpretace dle normy
Dom_2_pokus_porovnani	Slovní interpretace dle normy
Dom_3_pokus_porovnani	Slovní interpretace dle normy
Dom_prumer_porovnani	Slovní interpretace dle normy
Nedom_zkus_porovnani	Slovní interpretace dle normy
Nedom_1_pokus_porovnani	Slovní interpretace dle normy
Nedom_2_pokus_porovnani	Slovní interpretace dle normy
Nedom_3_pokus_porovnani	Slovní interpretace dle normy
Nedom_prumer_porovnani	Slovní interpretace dle normy
Poznamky	Text

Tabulka A.2 Formát exportovaných dat pro PPT

Sloupec	Formát
Testujici	Jméno Příjmení
Datum	dd.mm.rrrr
Cas_zahajeni	hh:mm
Cas_ukonceni	hh:mm
Dom_1_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 25]$
Dom_2_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 25]$
Dom_3_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 25]$
Dom_prumer	$x, x \in \mathbb{R} \wedge x \in [0, 25]$
Nedom_1_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 25]$
Nedom_2_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 25]$
Nedom_3_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 25]$
Nedom_prumer	$x, x \in \mathbb{R} \wedge x \in [0, 25]$
Obe_1_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 25]$
Obe_2_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 25]$
Obe_3_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 25]$
Obe_prumer	$x, x \in \mathbb{R} \wedge x \in [0, 25]$
Soucet_1_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 75]$
Soucet_2_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 75]$
Soucet_3_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 75]$
Soucet_prumer	$x, x \in \mathbb{R} \wedge x \in [0, 75]$
Kompletovani_1_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 25]$
Kompletovani_2_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 25]$
Kompletovani_3_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 25]$
Kompletovani_prumer	$x, x \in \mathbb{R} \wedge x \in [0, 25]$
Dom_1_pokus_SDS	$x, x \in \mathbb{R}$
Dom_2_pokus_SDS	$x, x \in \mathbb{R}$
Dom_3_pokus_SDS	$x, x \in \mathbb{R}$
Dom_prumer_SDS	$x, x \in \mathbb{R}$
Nedom_1_pokus_SDS	$x, x \in \mathbb{R}$
Nedom_2_pokus_SDS	$x, x \in \mathbb{R}$
Nedom_3_pokus_SDS	$x, x \in \mathbb{R}$
Nedom_prumer_SDS	$x, x \in \mathbb{R}$

Tabulka A.3 Formát exportovaných dat pro PPT (pokračování)

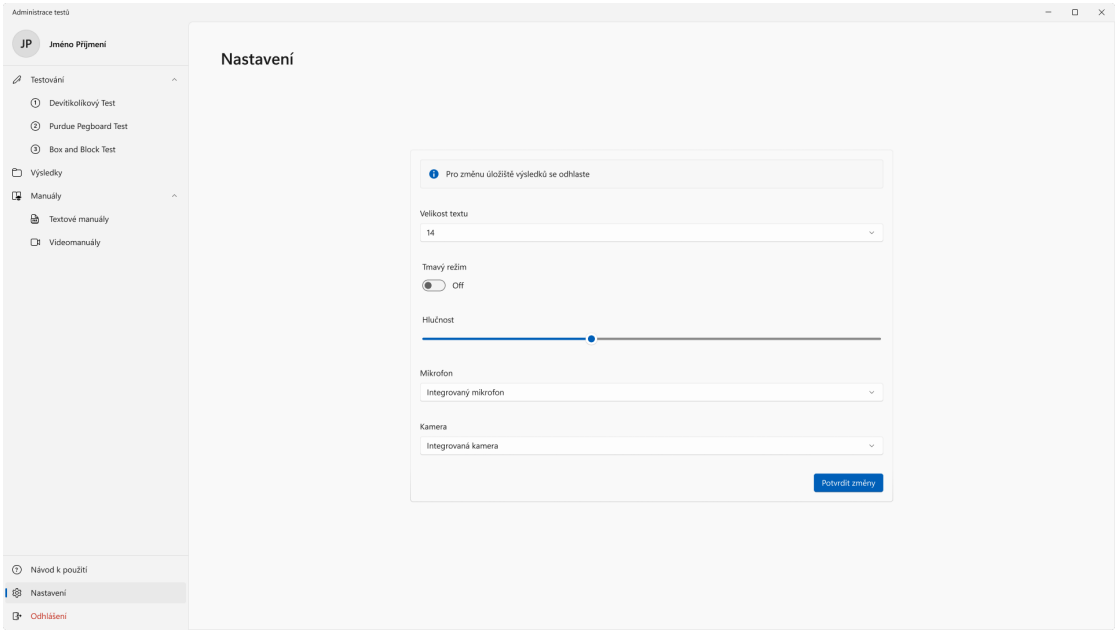
Sloupec	Formát
Obe_1_pokus_SDS	$x, x \in \mathbb{R}$
Obe_2_pokus_SDS	$x, x \in \mathbb{R}$
Obe_3_pokus_SDS	$x, x \in \mathbb{R}$
Obe_prumer_SDS	$x, x \in \mathbb{R}$
Soucet_1_pokus_SDS	$x, x \in \mathbb{R}$
Soucet_2_pokus_SDS	$x, x \in \mathbb{R}$
Soucet_3_pokus_SDS	$x, x \in \mathbb{R}$
Soucet_prumer_SDS	$x, x \in \mathbb{R}$
Kompletovani_1_pokus_SDS	$x, x \in \mathbb{R}$
Kompletovani_2_pokus_SDS	$x, x \in \mathbb{R}$
Kompletovani_3_pokus_SDS	$x, x \in \mathbb{R}$
Kompletovani_prumer_SDS	$x, x \in \mathbb{R}$
Dom_1_pokus_porovnani	Slovní interpretace dle normy
Dom_2_pokus_porovnani	Slovní interpretace dle normy
Dom_3_pokus_porovnani	Slovní interpretace dle normy
Dom_prumer_porovnani	Slovní interpretace dle normy
Nedom_1_pokus_porovnani	Slovní interpretace dle normy
Nedom_2_pokus_porovnani	Slovní interpretace dle normy
Nedom_3_pokus_porovnani	Slovní interpretace dle normy
Nedom_prumer_porovnani	Slovní interpretace dle normy
Obe_1_pokus_porovnani	Slovní interpretace dle normy
Obe_2_pokus_porovnani	Slovní interpretace dle normy
Obe_3_pokus_porovnani	Slovní interpretace dle normy
Obe_prumer_porovnani	Slovní interpretace dle normy
Soucet_1_pokus_porovnani	Slovní interpretace dle normy
Soucet_2_pokus_porovnani	Slovní interpretace dle normy
Soucet_3_pokus_porovnani	Slovní interpretace dle normy
Soucet_prumer_porovnani	Slovní interpretace dle normy
Kompletovani_1_pokus_porovnani	Slovní interpretace dle normy
Kompletovani_2_pokus_porovnani	Slovní interpretace dle normy
Kompletovani_3_pokus_porovnani	Slovní interpretace dle normy
Kompletovani_prumer_porovnani	Slovní interpretace dle normy
Poznamky	Text

Tabulka A.4 Formát exportovaných dat pro BBT

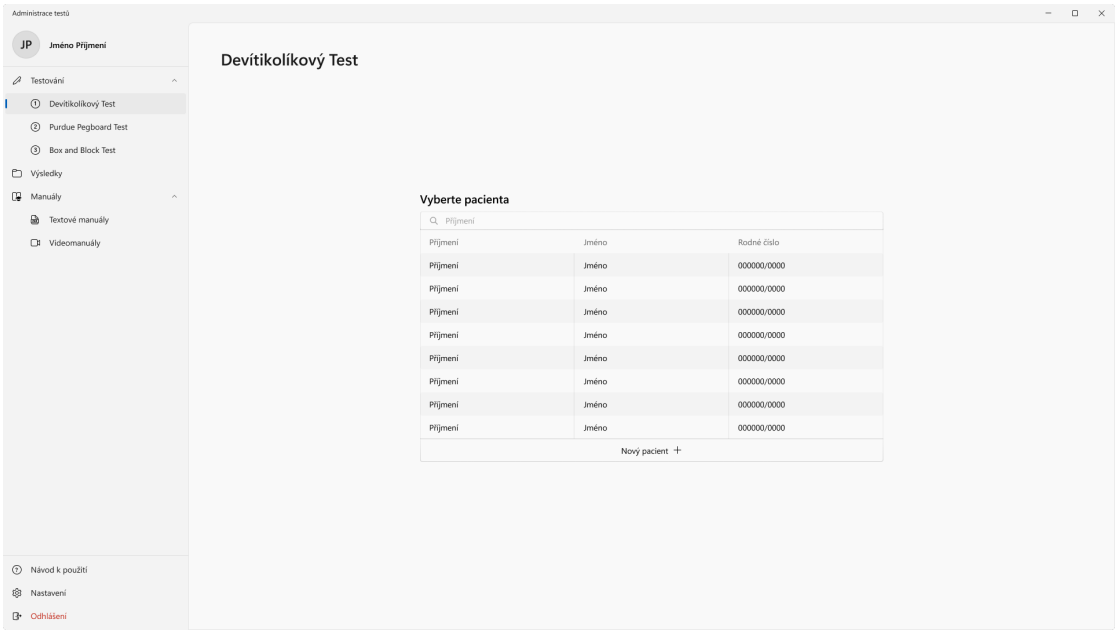
Sloupec	Formát
Testujici	Jméno Příjmení
Datum	dd.mm.rrrr
Cas_zahajeni	hh:mm
Cas_ukonceni	hh:mm
Dom_zkus	$x, x \in \mathbb{Z} \wedge x \in [0, 150]$
Dom_1_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 150]$
Dom_2_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 150]$
Dom_3_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 150]$
Dom_prumer	$x, x \in \mathbb{R} \wedge x \in [0, 150]$
Nedom_zkus	$x, x \in \mathbb{Z} \wedge x \in [0, 150]$
Nedom_1_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 150]$
Nedom_2_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 150]$
Nedom_3_pokus	$x, x \in \mathbb{Z} \wedge x \in [0, 150]$
Nedom_prumer	$x, x \in \mathbb{R} \wedge x \in [0, 150]$
Dom_zkus_SDS	$x, x \in \mathbb{R}$
Dom_1_pokus_SDS	$x, x \in \mathbb{R}$
Dom_2_pokus_SDS	$x, x \in \mathbb{R}$
Dom_3_pokus_SDS	$x, x \in \mathbb{R}$
Dom_prumer_SDS	$x, x \in \mathbb{R}$
Nedom_zkus_SDS	$x, x \in \mathbb{R}$
Nedom_1_pokus_SDS	$x, x \in \mathbb{R}$
Nedom_2_pokus_SDS	$x, x \in \mathbb{R}$
Nedom_3_pokus_SDS	$x, x \in \mathbb{R}$
Nedom_prumer_SDS	$x, x \in \mathbb{R}$
Dom_zkus_porovnani	Slovní interpretace dle normy
Dom_1_pokus_porovnani	Slovní interpretace dle normy
Dom_2_pokus_porovnani	Slovní interpretace dle normy
Dom_3_pokus_porovnani	Slovní interpretace dle normy
Dom_prumer_porovnani	Slovní interpretace dle normy
Nedom_zkus_porovnani	Slovní interpretace dle normy
Nedom_1_pokus_porovnani	Slovní interpretace dle normy
Nedom_2_pokus_porovnani	Slovní interpretace dle normy
Nedom_3_pokus_porovnani	Slovní interpretace dle normy
Nedom_prumer_porovnani	Slovní interpretace dle normy
Poznamky	Text

Příloha B

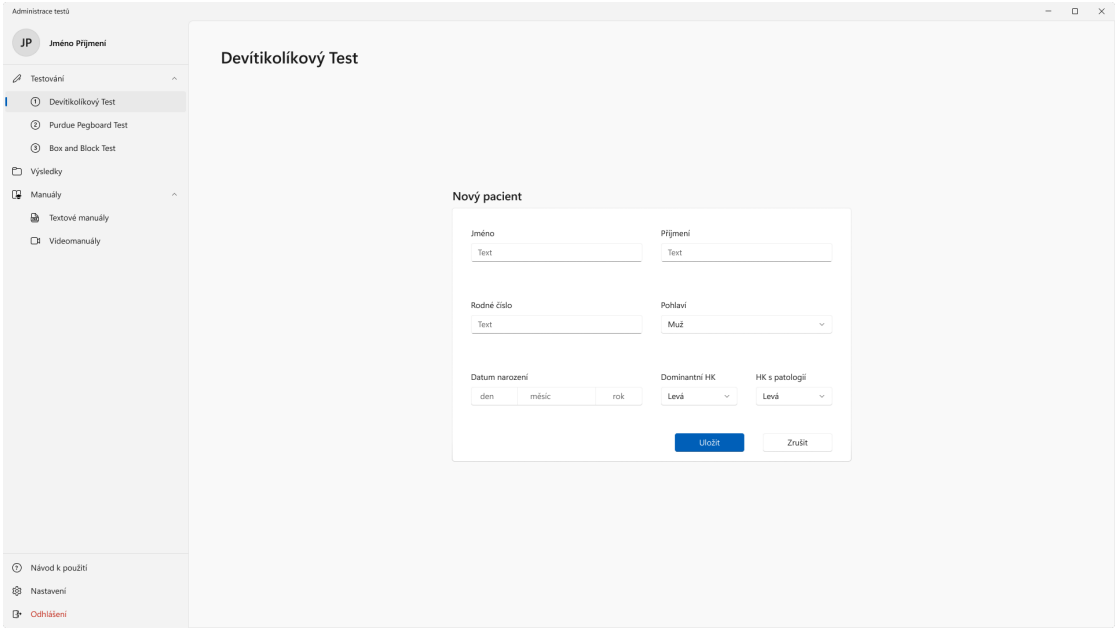
Návrh uživatelského rozhraní



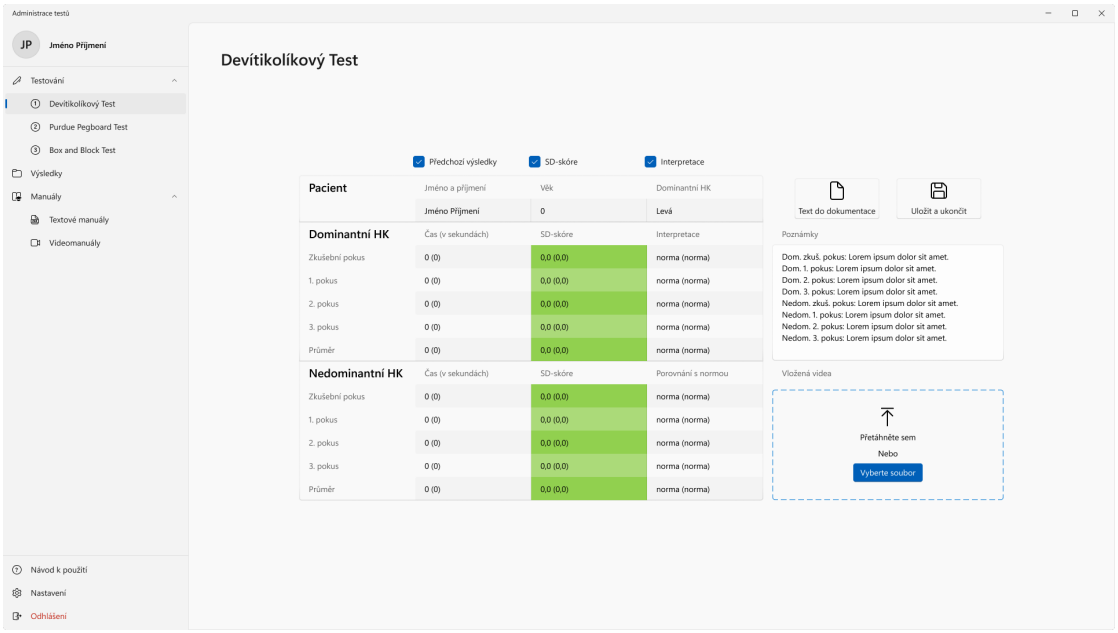
Obrázek B.1 Návrh obrazovky pro nastavení aplikace



Obrázek B.2 Návrh obrazovky pro výběr pacienta



Obrázek B.3 Návrh obrazovky pro přidání pacienta



Obrázek B.4 Návrh obrazovky pro zobrazení výsledků NHPT

Příloha C

Testovací scénáře

Testovací scénář č. 1

1. Přidejte uživatelský účet a vyberte ho.
2. Vyberte adresář pro uložení dat.
3. Vyberte Devítikolíkový test.
4. Vyberte anonymního pacienta.
5. Testování pacienta:
   a. Otevřete okno s přenosem z kamery.
   b. Spusťte nahrávání a udělejte alespoň jednu pauzu.
   c. Zapište alespoň jednu naměřenou hodnotu.
   d. Zapište alespoň jednu vlastní poznámku a jednu ze standardních.
   e. Anulujte alespoň jeden pokus třikrát.
   f. Během přehrávání alespoň jedné zvukové instrukce ji pozastavte a spusťte od začátku.
6. Zobrazení výsledků:
   a. Zobrazte si SD-skóre a porovnání s normou.
   b. Zkopírujte text do dokumentace a vložte ho někam.
   c. Ukončete testování a uložte výsledky.
   d. Otevřete adresář s výsledky přes aplikaci a otevřete videozáznam.
7. Zobrazení předchozích výsledků:
   a. Vyberte opět anonymního pacienta.
   b. Ukončete testování bez zapsání žádných hodnot.
   c. Zobrazte si SD-skóre ze skutečného (předchozího) testu.
8. Odhlaste se bez uložení výsledků.

Testovací scénář č. 2

1. Odstraňte předchozí uživatelský účet.
2. Přidejte vlastní uživatelský účet a vyberte ho.
3. Nastavte si tmavý režim.
4. Vyberte Purdue Pegboard Test.
5. Přidejte nového pacienta.
6. Použijte vyhledávání podle příjmení a nechte pouze nového pacienta v seznamu.
7. Vyberte tohoto pacienta.
8. Testování pacienta:
   a. Spusťte nahrávání bez použití této aplikace a rozdělte ho na alespoň dva videa.
   b. Zapište alespoň jednu naměřenou hodnotu.
   c. Zapište alespoň jednu poznámku ze standardních.
9. Zobrazení výsledků:
   a. Zobrazte si naměřené a vypočítané hodnoty ze všech subtestů.
   b. Přidejte nahrávky a libovolný další soubor.
   c. Odstraňte nepotřebný soubor.
   d. Ukončete testování a uložte výsledky.
   e. Otevřete adresář s výsledky přes aplikaci.
   f. Zobrazte si text do dokumentace.
   g. Zobrazte si videozáznamy.
   h. Použijte Excel pro zobrazení výsledků.
10. Odhlaste se.

Testovací scénář č. 3

1. Přidejte vlastní uživatelský účet a vyberte ho.
2. Změňte adresář pro uložení výsledků.
3. Nastavte si světlý režim.
4. Nastavte si velikost písma na 16.
5. Schovejte levý navigační panel.
6. Otevřete stránku s manuály.
7. Vyberte Box and Block Test.
8. Přidejte nového pacienta a vyberte ho.
9. Testování pacienta:
   a. Spusťte nahrávání v aplikaci, po několika pokusech ho ukončete a spusťte nahrávání mimo aplikaci.
   b. Zapište alespoň jednu naměřenou hodnotu.
   c. Zapište alespoň jednu poznámku.
   d. Ukončete testování bez zobrazení stránky pro poslední pokus.
10. Zobrazení výsledků:
    a. Přidejte videozáznam(y).
    b. Ukončete testování a uložte výsledky.
    c. Otevřete adresář s výsledky přes aplikaci.
    d. Otevřete videozáznamy.

Bibliografie

1. RYBÁŘOVÁ, Kateřina; SÝKOROVÁ, Jitka; NOVÁKOVÁ, Olga; KOL. KLINIKA REHABILITAČNÍHO LÉKAŘSTVÍ 1. LF UK A VFN V PRAZE. *Česká rozšířená verze manuálu pro Nine Hole Peg Test (NHPT)*. Praha: Rehalb, 2021. ISBN 978-80-906738-2-3.
2. RYBÁŘOVÁ, Kateřina; SÝKOROVÁ, Jitka; RODOVÁ, Zuzana; KOL. KLINIKA REHABILITAČNÍHO LÉKAŘSTVÍ 1. LF UK A VFN V PRAZE. *Česká rozšířená verze manuálu pro Purdue Pegboard Test (PPT): Model 32020A*. Praha: Rehalb, 2021. ISBN 978-80-906738-8-5.
3. RYBÁŘOVÁ, Kateřina; SÝKOROVÁ, Jitka; MARKOVCOVÁ, Lucie; KOL. KLINIKA REHABILITAČNÍHO LÉKAŘSTVÍ 1. LF UK A VFN V PRAZE. *Česká rozšířená verze manuálu pro Box and Block Test (BBT)*. Praha: Rehalb, 2021. ISBN 978-80-906738-5-4.
4. LAFAYETTE INSTRUMENT COMPANY. *Purdue Pegboard Scoring App* [soft.]. 2015. [cit. 2024-01-28]. Dostupné z: `https://lafayetteevaluation.com/products/purdue-pegboard-scoring-app/`.
5. OÑA, E. D.; BALAGUER, C.; JARDÓN, A. Automatic Cube Counting System for the Box and Blocks Test Using Proximity Sensors: Development and Validation. *Electronics*. 2023, roč. 12, č. 4. Dostupné z DOI: `10.3390/electronics12040914`.
6. GOOGLE. *Firebase* [soft.]. 2010. [cit. 2024-02-08]. Dostupné z: `https://firebase.google.com/`.
7. LAPLANTE, Phillip A. *Requirements Engineering for Software and Systems (Applied Software Engineering Series)*. 3. vyd. Boca Raton: Auerbach Publications, 2017. ISBN 978-1138196117.
8. LIGHTSEY, Bob. *Systems Engineering Fundamentals*. Fort Belvoir: Defense Acquisition University Press, 2001. ISBN 9780160732904.
9. *Publikační činnost* [online]. 1. lékařská fakulta Univerzity Karlovy, 2022 [cit. 2024-02-04]. Dostupné z: `https://rehabilitace.lf1.cuni.cz/publikacni-cinnost-uvod`.
10. *Moodle pro další vzdělavání na 1. LF UK* [online]. 1. lékařská fakulta Univerzity Karlovy, 2024 [cit. 2024-02-04]. Dostupné z: `https://kurzy.lf1.cuni.cz/`.
11. MATHIOWETZ, Virgil; WEBER, Karen; KASHMAN, Nanct; ET AL. Adult Norms For The Nine Hole Peg Test Of Finger Dexterity. *The Occupational Therapy Journal of Research* [online]. 1985, roč. 5, č. 1, s. 24–38 [cit. 2024-02-04]. Dostupné z DOI: `10.1177/153944928500500102`.
12. TIFFIN, Joseph; ASHER, E. J. The Purdue Pegboard: norms and studies of reliability and validity. *Journal of Applied Psychology* [online]. 1948, roč. 32, č. 3, s. 234–247 [cit. 2024-02-04]. ISSN 1939-1854. Dostupné z DOI: `10.1037/h0061266`.

13. MATHIOWETZ, Virgil; VOLLAND, Gloria; KASHMAN, Nancy; ET AL. Adult Norms for the Box and Block Test of Manual Dexterity. *American Journal of Occupational Therapy* [online]. 1985, roč. 39, č. 6, s. 386–391 [cit. 2024-02-04]. ISSN 0272-9490. Dostupné z DOI: 10.5014/ajot.39.6.386.

14. DAVIES, Rachel. Non-Functional Requirements: Do User Stories Really Help? *Methods & Tools* [online]. 2010 [cit. 2024-02-05]. Dostupné z: https://www.methodsandtools.com/archive/archive.php?id=113.

15. RUMBAUGH, James; JACOBSON, Ivar; BOOCH, Grady. *Unified Modeling Language Reference Manual*. 2. vyd. London: Pearson Higher Education, 2004. ISBN 0321245628.

16. OPENJS FOUNDATION. *Electron* [soft.]. 2013. [cit. 2024-04-11]. Dostupné z: https://www.electronjs.org/.

17. *Showcase* [online]. OpenJS Foundation, 2023 [cit. 2024-04-11]. Dostupné z: https://www.electronjs.org/apps/.

18. ORACLE CORPORATION. *JavaFX* [soft.]. 2008. [cit. 2024-04-11]. Dostupné z: https://openjfx.io/.

19. *Trail: Creating a GUI With Swing* [online]. Oracle Corporation, 2024 [cit. 2024-04-11]. Dostupné z: https://docs.oracle.com/javase/tutorial/uiswing/.

20. SCYTHE STUDIO. 4 Best frameworks for cross-platform desktop app development [online]. 2022 [cit. 2024-04-14]. Dostupné z: https://scythe-studio.com/en/blog/4-best-frameworks-for-cross-platform-desktop-app-development.

21. QT GROUP. *Qt Framework* [soft.]. 1995. [cit. 2024-04-14]. Dostupné z: https://www.qt.io/product/framework/.

22. MICROSOFT. *Windows Presentation Foundation* [soft.]. 2006. [cit. 2024-04-14]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/?view=netdesktop-8.0/.

23. *Windows Developer FAQ* [online]. Microsoft, 2024 [cit. 2024-04-14]. Dostupné z: https://learn.microsoft.com/en-us/windows/apps/get-started/windows-developer-faq/.

24. WHITE, Steven; ET. AL. An overview of Windows development options. *Microsoft Learn* [online]. 2023 [cit. 2024-04-14]. Dostupné z: https://learn.microsoft.com/en-us/windows/apps/get-started/.

25. HENDL, Jan. *Přehled statistických metod: analýza a metaanalýza dat*. 4., rozš. vyd. Praha: Portál, 2012. ISBN 978-80-262-0200-4.

26. RIEGEROVÁ, Jarmila; PŘIDALOVÁ, Miroslava; ULBRICHOVÁ, Marie. *Aplikace fyzické antropologie v tělesné výchově a sportu: (příručka funkční antropologie)*. 3. vyd. Olomouc: Hanex, 2006. ISBN 80-85783-52-5.

27. MICROSOFT. Design and code Windows apps. *Microsoft Learn* [online]. 2022 [cit. 2024-04-05]. Dostupné z: https://learn.microsoft.com/en-us/windows/apps/design/.

28. MESCIUS INC. WPF Development Best Practices for 2024. *Medium* [online]. 2024 [cit. 2024-04-15]. Dostupné z: https://medium.com/mesciusinc/wpf-development-best-practices-for-2024-9e5062c71350.

29. MICHAEL, Stonis; WARREN, Genevieve; JAIN, Tarun; PINE, David. Model-View-ViewModel (MVVM). *Learn Microsoft* [online]. 2023 [cit. 2024-04-15]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/architecture/maui/mvvm.

30. CLOSE, Josh. *CsvHelper* [soft.]. 2009. [cit. 2024-05-11]. Dostupné z: https://joshclose.github.io/CsvHelper/.

31. *Sync SharePoint files and folders* [online]. Microsoft, 2024 [cit. 2024-05-11]. Dostupné z: https://support.microsoft.com/en-us/office/sync-sharepoint-files-and-folders-87a96948-4dd7-43e4-aca1-53f3e18bea9b.
32. SHIMAT. *OpenCvSharp* [soft.]. 2024. [cit. 2024-05-11]. Dostupné z: https://github.com/shimat/opencvsharp/.
33. NAUDIO. *NAudio* [soft.]. 2024. [cit. 2024-05-11]. Dostupné z: https://github.com/naudio/NAudio/.
34. ROSENBJERG, Malte. *FFMpegCore* [soft.]. 2024. [cit. 2024-05-11]. Dostupné z: https://github.com/rosenbjerg/FFMpegCore/.
35. LEPO. *WPF UI* [soft.]. 2021. [cit. 2024-05-11]. Dostupné z: https://wpfui.lepo.co/index.html/.
36. FOUNDATION, .NET. *xUnit.net* [soft.]. 2024. [cit. 2024-05-12]. Dostupné z: https://xunit.net/.
37. CAZZULINO, Daniel. *Moq* [soft.]. 2024. [cit. 2024-05-16]. Dostupné z: https://github.com/devlooped/moq/.

Obsah příloh

- `readme.txt` ..... stručný popis obsahu média
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `nikolnik-thesis.pdf` ..... text práce ve formátu PDF