BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

APPLICATION FOR INTEGRATION OF ANALYTICAL TOOLS FOR CRIMINAL POLICE

APLIKACE PRO INTEGRACI ANALYTICKÝCH NÁSTOROJŮ PRO KRIMINÁLNÍ POLICII

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR — **MATĚJ MUDRA**
AUTOR PRÁCE

SUPERVISOR — **Ing. ONDŘEJ KANICH, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Bachelor's Thesis Assignment

146121

Institut:	Department of Intelligent Systems (UITS)
Student:	**Mudra Matěj**
Programme:	Information Technology
Specialization:	Information Technology
Title:	**Application for Integration of Analytical Tools for Criminal Police**
Category:	User Interfaces
Academic year:	2022/23

Assignment:

1. Get familiar with desktop application development and user-friendly design for Windows with a focus on displaying and filtering visual data.
2. Design application focused on a visual representation of annotated and classified images and their filtering. Data will be provided by Mock API.
3. Implement designed application.
4. Prepare a set of tests to verify application functionality.

Literature:

- OULASVIRTA, Antti, et al. Combinatorial optimization of graphical user interface designs. *Proceedings of the IEEE*, 2020, 108.3: 434-464.
- TILAK, Geetali; BHAUMIK, Amiya. A Review on Security and Usability of Graphical User Interface Design.

Requirements for the semestral defence:
Points 1-2.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Kanich Ondřej, Ing., Ph.D.**
Head of Department:	Hanáček Petr, doc. Dr. Ing.
Beginning of work:	1.11.2022
Submission deadline:	10.5.2023
Approval date:	8.5.2023

Abstract

The purpose of this thesis is to design and create a programme capable of integrating a series of analytical tools to aid crime investigation involving large digital data processing. The integration process is dynamic and user configurable to allow for future expansion and flexibility of the analysis pipeline without the need to update the code base. The integration interface itself is designed with online wireframing and prototyping tool Figma and implemented with UI framework Flutter using Dart programming language. This allows for easy development and testing across the Facis team members platfroms. The resulting application is tested by a subject matter expert who one of the assumed users of the final application and provided vital information about his field.

Abstrakt

Cílem této práce je navrhnout a vytvořit program schopný integrace série analytických nástrojů, které budou napomáhat prošetřování velkého množství dat. Integrační proces je dynamický a uživateli nastavitelný pro umožnění rozšíření nástrojů použitých pro analýzu dat bez nutnosti aktualizací samotného programu. Uživatelské prostředí je navrženo pomocí online nástroje sloužícího ke tvorbě prototypů Figma a implementováno pomocí frameworku Flutter, který využívá programovací jazyk Dart. Tato volba umožňuje vývoj a testování na všech platformách členů týmy Facis. Výsledná aplikace byla otestována odborníkem, který je jedním z budoucích uživatelů této aplikace.

Keywords

Dynamic user interface, Flutter, Desktop application, Figma, Analytical tools, HTTP, Dart, Client-Server architecture

Klíčová slova

Dynamické uživatelské prostředí, Flutter, Desktopová applikace, Figma, Analytické nástroje, HTTP, Dart, Client-Server architektura

Reference

MUDRA, Matěj. *Application for integration of analytical tools for criminal police.* Brno, 2023. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Ondřej Kanich, Ph.D.

Rozšířený abstrakt

Cílem této práce je navrhnout a vytvořit aplikaci, která bude schopná flexibilně integrovat široké spektrum nástrojů ke zpracovávání obrazových dat zajištěných během vyšetřování. Jednou z hlavních motivací tvorby této aplikace je jejich složitá rozšiřitelnost, kdy je potřeba je aktualizovat pro přidání nových nástrojů. Což je nejen časově náročné a vyžaduje to zásah administrátora. Je tedy potřeba zkombinovat poznatky z oboru vývoje uživatelských prostředí a integrace nástrojů využívajících příkazový řádek pro interakci s uživateli. Následně vytvořit aplikaci, která tento proces zjednoduší a umožní tak využívat tyto nástroje uživatelům s omezenou znalostí práce s příkazovým řádkem.

V současném stavu experti využívají mnoho nástrojů s různou funkcionalitou od prolomování šifrování přes katalogování obrazových dat. Časově nejnáročnější část analýzy je hledání závadných dat inkriminujících pachatele, ta se ale stále provádí ručně. To celkově tento proces zpomaluje a dělá práci vyšetřovatelů těžší. Ti ale již v současné době využívají mnoho nástrojů a přidáním dalšího by se nároky na ně ještě zvýšila. Vytvořená aplikace tedy musí být schopná integrovat nástroje, které funkcionalitu nejen přidají, ale zároveň nahradí některé aplikace, které jsou v současnosti používány.

Otázka, na kterou se touto prací snažím odpovědět je, jakým způsobem vytvořit aplikaci, jejíchž obsah není pevně definovaný a je ho třeba tvořit flexibilně a za běhu. Před vyřešením tohoto problému je nutné zjistit kdo bude aplikaci používat, jaké jsou schopnosti a motivace budoucích uživatelů, v jaké prostředí bude aplikace nasazena a jak vybrat technologie nutné pro vytvoření aplikace, která bude splňovat požadavky uživatelů. Pro seznámení se s budoucím uživatelem jsem využil čas strávený na schůzkách projektu Facis s vyšetřovatelem spolupracujícím na projektu. Metod pro vizualizaci návrhů jako je například tvorbu skic a maket aplikace pomocí nástrojů jako je Adobe Photoshop nebo online nástroje Figma, které byly použity během schůzek jako nástroje pro sbírání zpětné vazby. Pro výběr technologií použitých pro tvorbu samotné desktopové aplikace bylo třeba sestavit seznam těch, které tuto platformu podporují a porovnat několik jejich klíčových aspektů od velikosti komunity indikující dlouhodobou podporu po jejich výkon.

Technologie vybraná pro implementaci aplikace je Flutter, což je otevřený software vytvořený společností Google, který se z původních dvou mobilních platforem postupně dostal i na platformy desktopové jako je Windows, macOS, a dokonce i Linux. Flutter je považován za stabilní pro operační systém Windows od začátku roku 2022. Toto ovlivnilo výběr desktopové technologie, protože Flutter nabízí kombinaci rychlosti, stability, podpory a flexibility jako žádná z ostatních technologií. Po výběru nástroje pro vytvoření aplikace bylo možno přistoupit k samotné problematice flexibilní integrace nástrojů společně s jejich konfigurací, třídění a filtrování výsledků.

Pro dosažení tohoto cíle bylo nutno vytvořit popis nástroje, který je možné sterilizovat do formátu JSON, ve kterém je pak možné nástroj ze serveru odeslat klientské aplikaci, kde se z těchto dat vytvoří třída reprezentující nástroj, ze které je následně možné vytvořit interaktivní prvek. Pomocí toho může následně uživatel interagovat s nástroji a zpracovávat jejich výsledky. Jelikož tyto popisy nástrojů budou tvořit jejich tvůrci je možné do nich vložit rozumné výchozí hodnoty a výrazně tak zlepšit uživatelskou přívětivost.

Výsledné řešení bylo otestováno jak uživateli s různými zkušenostmi s prací s aplikacemi pro zpracování obrazových dat, tak zmíněným vyšetřovatelem. Z pohledu uživatelů nebyli jasné některé volby v uživatelském prostředí, které byli motivovány požadavky ze strany policie. Všichni uživatelé však byli schopni dokončit úlohy stanovené testovacím scénářem. Samotný vyšetřovatel aplikaci zhodnotil kladně s doporučením ovládacích prvků, které by dále přiblížili aplikaci těm, které jsou v jeho oboru hojně využívány.

Application for integration of analytical tools for criminal police

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mr. Ing. Ondřej Kanich, Ph.D. and prof. Ing. Martin Drahanský, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Matěj Mudra
May 9, 2023

Acknowledgements

I could not have undertaken this journey without Ing. Ondřej Kanich, Ph.D. and Prof. Ing. Martin Drahanský, Ph.D. the guidance, consultation, insightful tips, information and feedback they gave me on my work. Thanks should also go to Ing. Bc. Martin Spurný for sharing his insights regarding his field of expertise as well as to all the members of the Facis team for their feedback on the application. I would also like to thank Ing. Jan Pluskal, Ph.D. for his level-headed approach to all my questions.

Contents

Chapter 1

Introduction

Analysis of image data stored on confiscated devices is a large and time-intensive part of the law enforcement agencies job. This process involves a lot of forensic tools used for various purposes, from getting through the encryption of a storage to finding and potentially extracting images or videos from files that are not of a typical video or image format. After this data retrieval phase comes another, much more time consuming task of manually going through the images and assessing them on individual bases. The inspection, classification and organisation of the retrieved images and other audio visual materials and flagging of content that might be used as an evidence of illegal activity.

The goal of this work is to design a proof-of-concept application that will allow testing and presentation of underlying system being developed in parallel with this application. This application is going to integrate wide spectrum of tools allowing one to incrementally add more tools during their development according to the needs of the law enforcement with the intended goal of covering the whole process of analysis of audio visual data from extracting them from the storage to analysing them using various technologies from simple scripts to neural networks. This should speed up the whole process by implementing partial automation of the analysis.

In the beginning of this work, in chapter 2 an overview of the system and site and some of the challenges linked to the nature of this project that presented themselves in the discovery phase will be given. Next, a chapter 3 focuses on requirement analysis and defining the needs and wants of the criminal police. After that a chapter 4 will focus on a process of creating an user interface, terms and methods linked to it. The chapter 5 will describe the design of the application and the thought process behind the tool integration endeavour and propose a way of implementing it. The next chapter 6 will focus on the choice of technology used to create the application, the Mock API and show how will the resulting application look like. And then chapter 7 will present testing methodology and summarise the results of user testing.

Chapter 2

System overview

This chapter gives a high-level overview of the system integrated in this work. It is crucial to grasp the scope of the whole system because to efficiently integrate diverse tools, it is necessary to understand what is being integrated first. The purpose of this tool is to present the analysed data securely and efficiently. Security is of high importance mainly because of the sensitive manner in which the investigated data need to be handled. Some limitations and complications have occurred as a result of the material which will be handled in the future by this application. Others have manifested simply due to the physical location of the architecture on which the application will run. It is essential to explore, explain, and work around or with all of them to ensure the best experience possible. All the information about the system is derived from internal project meetings and interviews with (IT) experienced forensic who is the source of most of the police requirements.

2.1 System components

Components can be divided into several categories depending on their purpose in the architecture. The component described in this work is the graphical user interface, further referenced as GUI, which enables the user to operate tools and retrieve data processed by them. The tools themselves will be described in a further section. Bridging the gap between tools and GUI is the work of the orchestrator. Its main purpose is to receive GUI messages and orchestrate the analysis pipeline which is designed and implemented by a colleague working on the Facis [28] project. For this work, the orchestrator and tools themselves are replaced by a Mock API which is made in a way to mirror the production back-end. This API is described by OpenAPI [41] specification. Such configuration is going to look like the on in the figure 2.1.

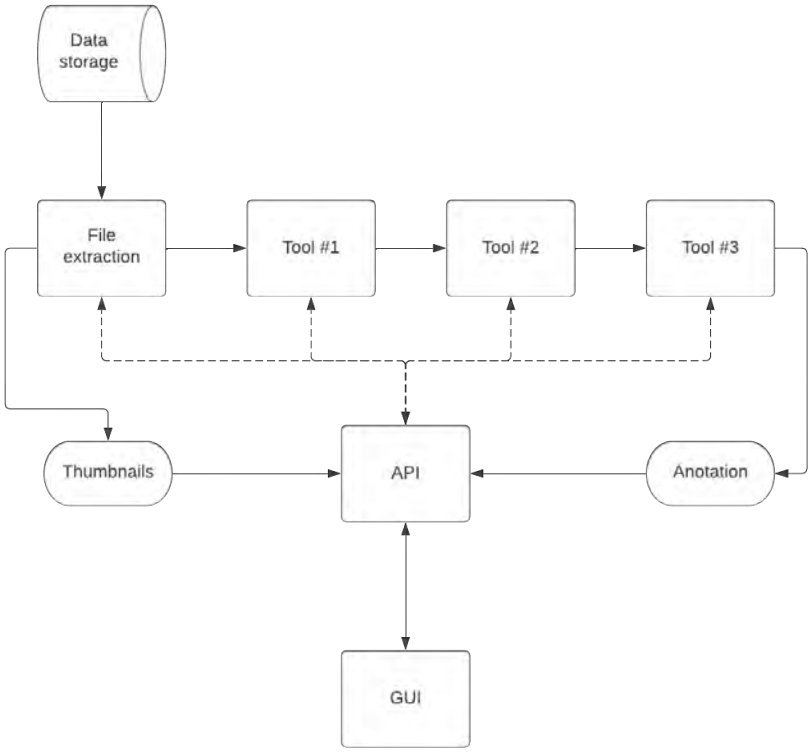


Figure 2.1: Conceptual model of Facis architecture.

2.2 Site related obstacles

Security is a big concern, as only certain authorised personnel will be granted access. Even among these trusted individuals, pipelines must be strictly separate. The separation will also be physical; the system itself is running on a Linux server in a server room. Connection is established over the local network. The client runs on a PC in a different room, but within the same building as the server. The room with a client PC will be located at a police station; even though only certain personnel will be allowed to enter the room, there will still be a need for roles with different permissions and access rights. The resulting placement of the server and client could resemble the floor map in figure 2.2. These roles will be described further in the following chapters.

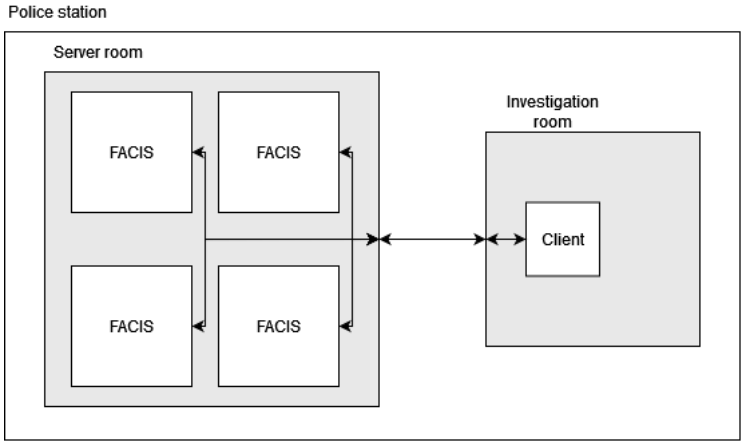


Figure 2.2: Visualisation of physical configuration of client and server machines.

2.3 Underlying infrastructure

The specification of the hardware on which the application will be deployed is, for the most part, dependent on what the Facis [28] team will agree on and, of course, some budgetary limitations will apply. The back-end is not the focus of my work, but it was agreed that the target operating system is Linux. For the client, Windows version 10 [11] or 11 [64] was chosen. The target platform will most certainly not change for the analysis phase, however, it has not been specified which OS and what hardware will be PC used for viewing mode equipped with. According to the forensic expert, the client PC could be equipped with 128 GB of RAM and the newest available consumer grade CPU as well as GPU. The application itself does not have to do any analysis of the source material; therefore, it is not expected to be very resource demanding.

2.4 Tools for integration

A pipeline for image analysis is composed of numerous tools with specific purposes, most of which are focused on finding objects, people or their parts in an image. There are a few tools which are extracting images from files and split videos into images. All of these tools require predetermined inputs adjustable by the user. The number of tools will with most certainty expand in the future, but the way they are interacting with the system, especially the orchestrator has to stay the same to streamline the implementation. Following tools were picked to simulate tools integrated in the final applications. While not all tools run in the pipeline will have outputs that are interesting or possible to visualise in the current state, the goal of these demonstrational tools is to give users something to interact with that is visible on the first sight and can therefore be evaluated easily.

2.4.1 Crowd counting

The first tool picked for the testing and demonstration is crowd counting tool. Its output is a single integer indicating the number of people found in the image and the certainty of the results accuracy. This tool with prepared data can simulate integration of tools such as *CrowdTrek* [62] which uses convolutional neural network.

2.4.2 Age estimate

The second tool chosen is age estimation tool. This tool was inspired by the work described in paper by Hiba and Keller [48], the output of this tool is an age estimation with given certainty. There are multiple tools that could inspire the demonstration representative but thanks to the high flexibility this application, the tools can be swapped out. For the testing purposes, ages have been generated randomly.

2.4.3 Face recognition

The last tool I chose for the demonstration is a face recognition tool inspired by face recognition software such as BioID [10]. It was chosen to showcase tool with more complex and non-numeric output. This being the name of the person in the picture in this case simple sorting and filtering is not possible and therefore more complex filtering and sorting tools could be created.

Chapter 3

Requirement analysis

Analysing user requirements can prove complicated when users do not have a clear vision of what the application should do and how it should look like. It is even more complicated if users actively do not share information and vision. Due to the circumstances, the development of this work was necessary to predict the requirements earlier than the actual reference materials were delivered. Adopting this strategy proved to be advantageous as the materials and the complete vision were shared just days before the deadline for this thesis.

3.1 Information gathering process

Most of the information was collected during the internal meetings of the Facis project [28]. Unfortunately, the GUI is not the only thing that needs to be dealt with and discussed during sessions, and therefore most of the notes recorded reactions throughout the iterative prototyping process. This, together with the user interface design being the most time-consuming part of the application [17] adds to the length of the individual iteration of design. During the literature review for this work, I have found that this issue is already well known and is deconstructed in an article by Daniel Florian et al. [17]. The similarity of this application and Lightroom is especially in the kind of data Lightroom works with and the workflow it allows the user to easily process thousands of pictures together with user-defined presets and some sensible defaults. With all these similarities, the information gathering strategy was a great inspiration and helped establish criteria and general standards by sharing the process from the perspective of industry professionals [58]. A big factor that partly negated the slow design process was the fact that the expert was present in person at most meetings and was very helpful in the process of collecting information.

3.2 Currently used software and processes

To better understand the needs of users, we examine the software and workflows that users are already familiar with. There are many types of software referenced by the expert, but a good example is EnCase Forensic [68] which is used by multiple government and private agencies; it allows users to decrypt different types of storage, organise files, and create reports. New versions of this software implement machine learning and artificial intelligence features, which can be based on the information on their website [68] that is used to detect multiple things like weapons, drugs, nudity, or sexually explicit content in

images. The feature frequently used and highlighted by the consulting expert was the file tree panel, which can be browsed and used to find images hidden in the folder structure with the option to jump to them in the main image browser. This process can also be reversed with the option to reveal the location of the files by clicking on it.

3.3 User experience personas

Creating a persona or multiple of them representing users, their motivation, needs, and skills goes a long way toward relating to the people who will use the final product. Personas can be represented by a poster that highlights their personal characteristics, profession, and skills and general mindset of the target audience [52]. This allows for developers to understand the userbase on a more personal level and thus tailor the user experience better for given group users represented by the persona.

3.3.1 Types of personas

There are three basic types of personas. The main difference between them is the amount of research needed to make them and consequently the time it takes to create them. The best method will differ depending on time, access to users, and information about them. These types of persona are:

- Proto personas
- Qualitative personas
- Statistical or mixed-method personas

Proto personas require the least time spent on research and preparation, they are created rapidly from the information already known about the userbase [52]. The time saved is valuable in an agile setting, and skipping research that consumes time and resources can result in faster initial results. However, there is the drawback that guesses and estimations might not be as on point as the developers would have hoped.

Qualitative personas are the exact opposite of the proto personas. The final representation of the end consumers is based on data obtained by qualitative research carried out with a small fraction of the userbase [52]. These data can be obtained by conducting interviews or field observations.

The third type is a mix of both previous types; the persona can be created by combining a few interviews to clear up misconceptions that developers might have about users. Alternatively, data can be obtained through quantitative research using questionnaires [52]. The amount of time that the quantitative research would take might be similar or even greater than in the case of qualitative research. This alternative is best used when the data about the users is already collected.

In my case, the first type was a clear choice because I had access to the expert who will use the application in the real world. The inability to contact other potential users narrowed down the choices to the point where I had no other choice.

3.3.2 Persona representing this project

Based on the information collected above, the following poster depicted in figure 3.1 representing the user experience persona was created with the use of Adobe Photoshop.

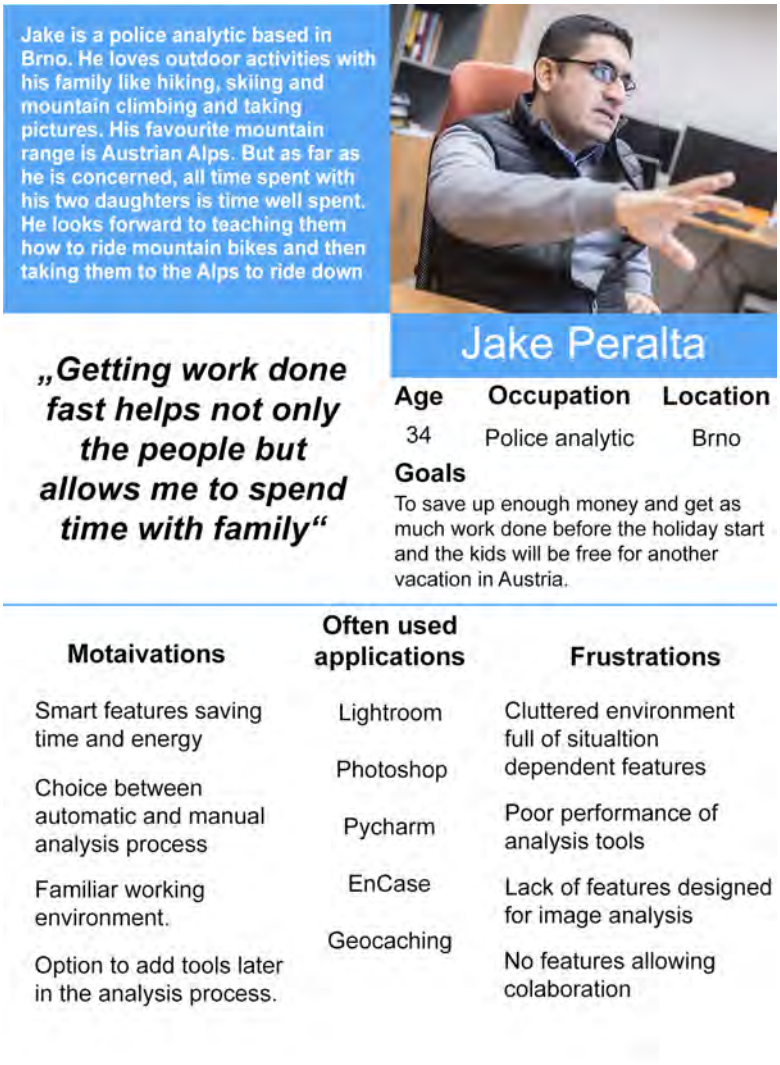


Figure 3.1: User experience persona Jake Peralta.

3.4 Requested improvements over existing software

During my interview with the expert and brief conversations during regular team meetings, a few suggestions came up that described features and functions already in place and preferred by the experts. These are mentioned further. Getting useful information from a potential user can be difficult because users do not have a clear vision and lists of features and use cases prepared in mind. In this case, however, the forensic expert had time to prepare notes and requests that he gathered during meetings and demonstrations of my designs. Unfortunately, because of the reasons mentioned in the beginning of this chapter,

many of these suggestions did not make it to the final design. The experts worked with EnCase Forensic on a daily basis and presented the following characteristics.

3.4.1 File tree browser

The file tree browser is implemented in EnCase, but it is not unique to this programme. For example, most modern integrated development environments such as Visual Studio, IntelliJ or Xcode do have file tree browser similar to the one present in figure 3.2 in the EnCase programme. The intent behind all of these implementations is similar, that is, to contextualise one file and find others similar to one currently in focus. The expert mentioned that in many cases, one evidence file leads to a larger cluster of evidence within the folder structure. This helps to speed up the analysis process by manually narrowing the scope of the investigation.

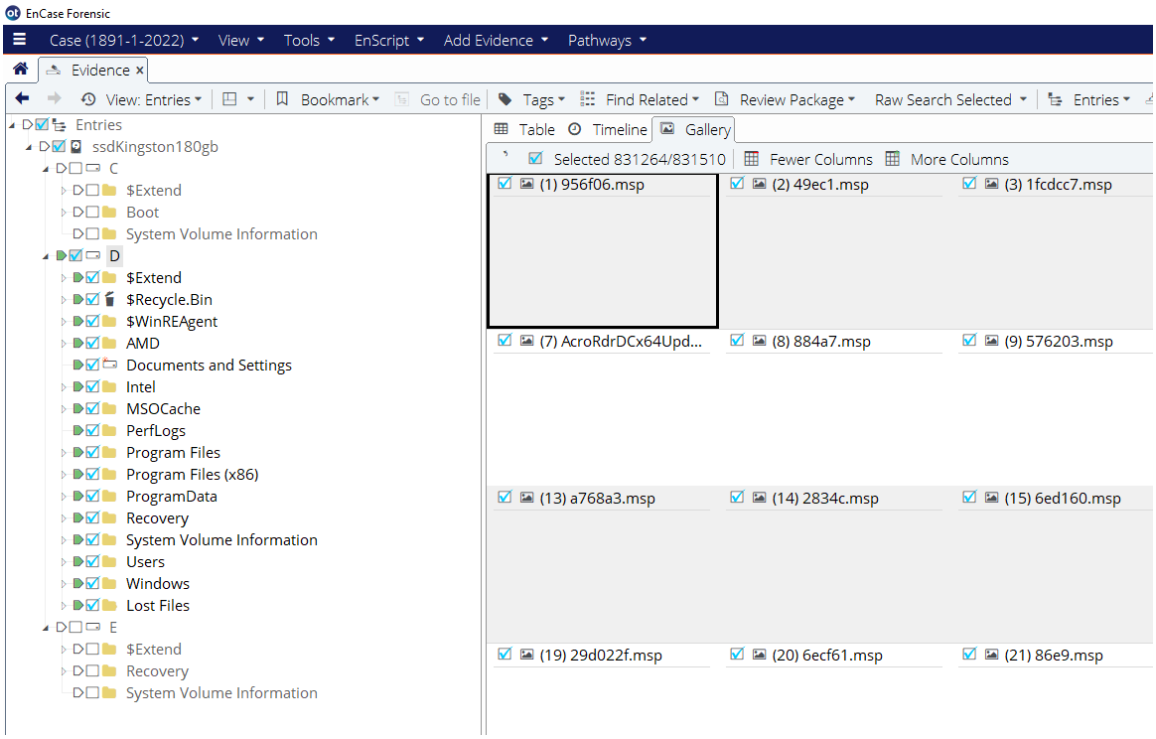


Figure 3.2: EnCase file tree inspector [68].

3.4.2 Collaborative features

Not a singular feature, but a set of features formulated in the meeting was the ability to collaborate on a case. Criminals are regular employees of the police department, and they also have days off and vacations. To avoid the investigation from going wrong, multiple people must work on one case. This requirement is not out of the ordinary; what is not common is to implement features that serve the purpose of assigning other analysts to the case in a way that works based on the input of other user IDs without any suggestions. The reason behind this unusual and not very user-friendly design is the anonymisation of the users.

3.4.3 Object annotation

According to Oracle's documentation, "Objects are key to understanding object-orientated technology. Look around right now and you'll find many examples of real-world objects: your dog, your desk, your television set, your bicycle." [3] This definition perfectly captures the type of object referenced in this context. The main purpose of an investigation is to find out what happened; it is important for analysts to have the option to add personal notes to as many things as they can. For example, the names of the images selected for analysis are complicated and long, and they scarcely provide information about the type of device from which they were extracted. Therefore, a custom note that describes it or references case files with further information is very important. This can be seen in the EnCase application in figure 3.3.

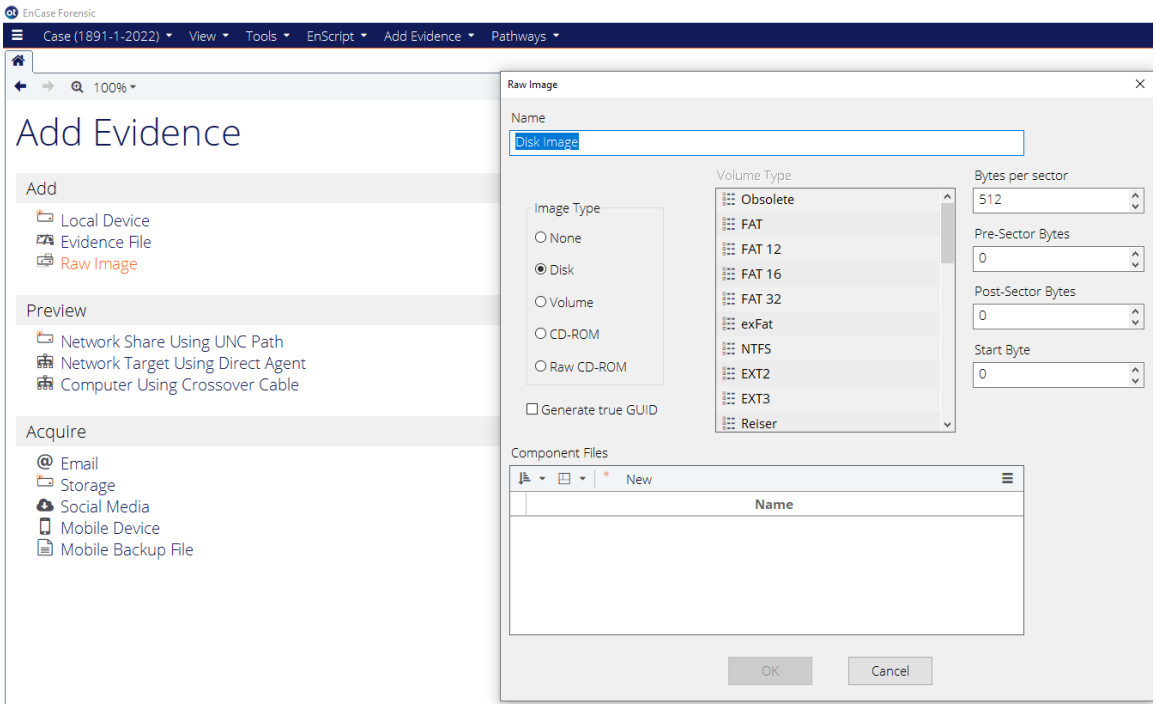


Figure 3.3: EnCase disk annotation feature [68].

3.4.4 Redundancy

Another pitch which could be shifted more toward backend is data redundancy, meaning assign multiple drives to be used during the analysis. While the concern over safety and integrity of analysed data is always valid, this is something that is going to be better managed by the server application itself in a smarter way than mirroring data between two or more disks.

3.5 Summary of requirements

During almost a year, this work is ongoing, some shared by team members of Brno University of Technology, some by team of Czech technical university in Prague, and some by the forensic experts. As stated previously, the crucial meeting with the police consultant

did not occur soon enough to implement all of the suggested features. The following points provide ideas and concepts to which the final design and implementation should adhere.

- Flexible,
- fast,
- expandable,
- allowing collaboration,
- secure,
- focused on analysis, sorting and filtering of visual data,
- similar to already used tools.

Chapter 4

User experience

This section describes the information gathering and interface-creating part that is needed for the best user experience. In order to gather sufficient information, multiple strategies need to be implemented at once. Some of the detail of this work cannot be shared, personas are a good way to avoid information leak and still get point across; these will be put together using information described in this chapter later on. Another thing that allows greater clarity in the process of transferring thoughts on figurative paper, in this case into code, is to use modern visualisation and mocking tools such as Figma [51], Lucidchart [82] or Adobe Photoshop [2].

4.1 Definition and basic terms

To design a user-friendly application, it is important to outline what the user experience is, what it is made of, and what language and terms are used to define it and tailor it to the needs of the users.

4.1.1 User experience

The definition of user experience is conflicted and not unified among the human-computer interaction community. There are multiple factors that cause this division [60].

- The term is used in a wide range of fields.
- Fast adoption of the term.
- It is associated with a range of fuzzy, dynamic, and nonspecific concepts.
- The concepts it is aggregating are not new.

The term user experience is used by people in many different fields, from researchers to salesmen. This causes the vagueness and nonspecificity because designers creating mock user interface while having developers making the application in mind will have different priorities and approach than salesmen presenting or pitching the application or its concept in the most desirable way possible. [60]

The very fast adoption of the term did not help either because the term is very catching and vague enough to use safely while not clearly defining clear boundaries or expectations that the project should meet. [60]

This ties to the next reason, which is the involvement of subjective variables such as aesthetic of the interface or the emotional state of the user. As it is impossible to objectively evaluate the emotions users are experiencing while using the application, the only thing that can be done is to poll the user experience after interacting with multiple versions of the application and evaluate the change between them. This still leaves a lot to be desired but gives the developers heading which will point them towards a better outcome. [47]

Contributing to confusion in regards to what user experience actually is the fact that the parts that together compose the term are not new in terms of human-computer interaction research. The aspects existed already and the term user experience ties them together; this is why different definitions seem familiar but are not known verbatim.

Although it is hard to define what user experience means, it would not be productive to ignore the term altogether. To help with the construction of the definition, the definitions of UX by other people researching this topic will provide a clearer image [60].

> Alben: All aspects of how people use an interactive product: the way it feels in their hands, how well they understand how it works, how they feel about it while using it, how well it serves their purposes, and how well it fits into the entire context in which they are using it.
>
> Nielsen-Norman Group: All aspects of the end-user's interaction with the company, its services, and its products.
>
> Mäkelä & Fulton Suri: A result of motivated action in a certain context.

The user interface was described in User experience, a research agenda by Hassenzahl and Tractinsky [47] as the term user experience is about technology which fulfils more than the needs of the user, but also acknowledges the existence of dynamic and complex encounter between it and its user. This definition covers all the previous definitions and addresses the drawbacks mentioned above.

4.1.2 The user interface

User interface and user experience are often mistaken for one another. Unlike user experience, the definition of user interface is clear and simple [90]. It is the point where users interact with the design or the point of human-computer interaction[40]. Although these two definitions do not appear to be the same in wording, they are of meaning. The user interface is the point where the user interacts with a computer; hence the term interface [46]. For the purpose of this work, only graphical user interfaces or GUIs will be expanded upon.

When making UI, designers are concerned with the surface and feeling their design gives away, as opposed to the experience the applications leave with the user after using it. A good user interface should visually guide the user towards their desired goal without being obtrusive or standing in the way. There is a saying that a well-thought-out user interface is "invisible" [46].

4.1.3 Design language

Similarly to the natural language, the design language is a tool that allows communication between two parties [7]. For the natural language, parties are mainly people, for designers, they are ideas of people creating an application an users. Tools used to communicate

thoughts instead of words and gestures are fonts, colours, layout, spacing, and many others [95]. Everyone using technology today is at least somewhat familiar with some design language. For example, Windows users should understand the Fluent design language, which was developed by Microsoft [63]. Because of the subtle nature of this way of communication, the differences between design languages are not obvious at first glance, but, for example, a Windows user will understand the design language that Apple uses in their MacOS and may feel confused or lost [59].

The design language is often used to maintain consistency throughout the application or across ecosystems, helping users adopt new products faster and strengthening brand recognition [5]. This will not only help the adoption rate, but also make users feel welcome and safe while exploring something completely new.

4.1.4 Usability

Usability is a term introduced as a replacement for the term „user friendly“ which was criticised for its vagueness [9]. It is a metric measuring how well a specific user can use a product or an application to effectively, efficiently, and satisfactorily fulfil predetermined goals [94]. Usability is part of user experience, it specifically covers how easy it is for the user to accomplish specific task with the product [79]. What usability consists of varies slightly from one definition to another, but most of the definitions have these points in common as listed by interaction design foundation [39]:

1. Effectiveness - Supports users to complete actions accurately.
2. Efficiency - Users can perform tasks quickly using the easiest process.
3. Engagement - Users find it pleasant to use and appropriate for its industry/topic.
4. Error tolerance - It supports a range of user actions and only shows an error in genuine erroneous situations. You achieve this by finding out the number, type, and severity of common errors users make, as well as how easily users can recover from those errors.
5. Ease of learning - New users can accomplish goals easily and even more easily on future visits.

There are a couple of good reasons why we care about usability for publicly accessible products and internal tools. For websites and widely distributed products, usability plays an important role in user retention, which is one of the primary indicators of a successful product. Although user retention is not a problem of internal tools, usability determines user productivity.[66]

4.2 Application layout

The layout of this application is partly inspired by Adobe Photoshop Lightroom Classic [1]. This inspiration was chosen for two main reasons.

- It is has a lot of widgets used to annotate an image in non-destructive way which is similar to this project,
- experts and other team members collectively described use cases that are similar to Lightroom use-cases,

- adobe software are the tool to choose when performing batch image data manipulation [80].

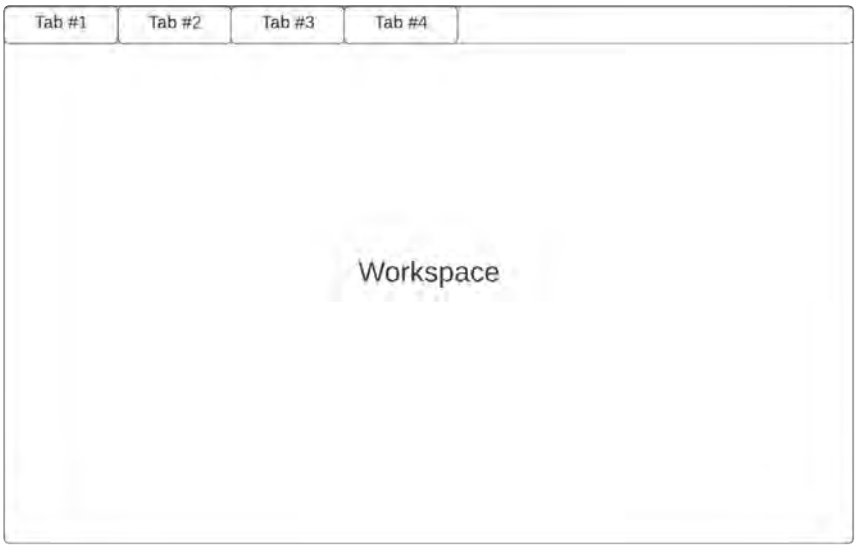


Figure 4.1: Many workspaces pattern wireframe.

As mentioned above, the app will integrate multiple tools into one application. There is no optimal or correct number of integrated tools. Currently, there are 12 tools in the integration plans. Tidwell [93] mention that to interact with multiple concurrent tasks at once, it is best to use the many workspace pattern. This pattern is best known for its usage in all the most popular web browsers, for example, Firefox as can be seen in figure 4.2. This pattern takes the shape of a window that has multiple rearrangeable tabs at the top, which are used to quickly access individual windows. These windows then present the web page or, in our case, the outputs and controls of one tool. Structuring the layout in this way helps the user navigate and be more comfortable to use. It was proven that common user interface design patterns improve navigation in cases where the user is familiar with the pattern [97]. Considering that just in the year 2021 over 3.2 billion people used Google Chrome alone, there is a very significant chance that the user of this application will be familiar with the chosen layout [96]. The wireframe of such pattern seen in figure 4.1.

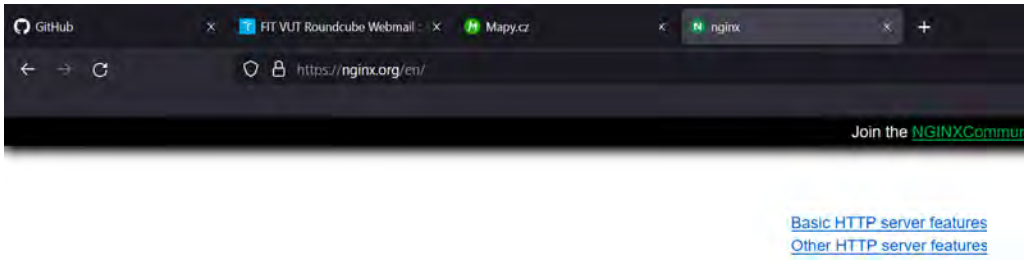


Figure 4.2: Many workspaces pattern implemented in Firefox.

After the initial layout is described, it is time to lay out the individual workspaces. As mentioned above, the tool choice is neither solid nor is the form of the individual tools.

Therefore, a system is needed to dynamically shape itself according to the needs of the user, who is the only person who knows which tools will be used to analyse data from the particular case.

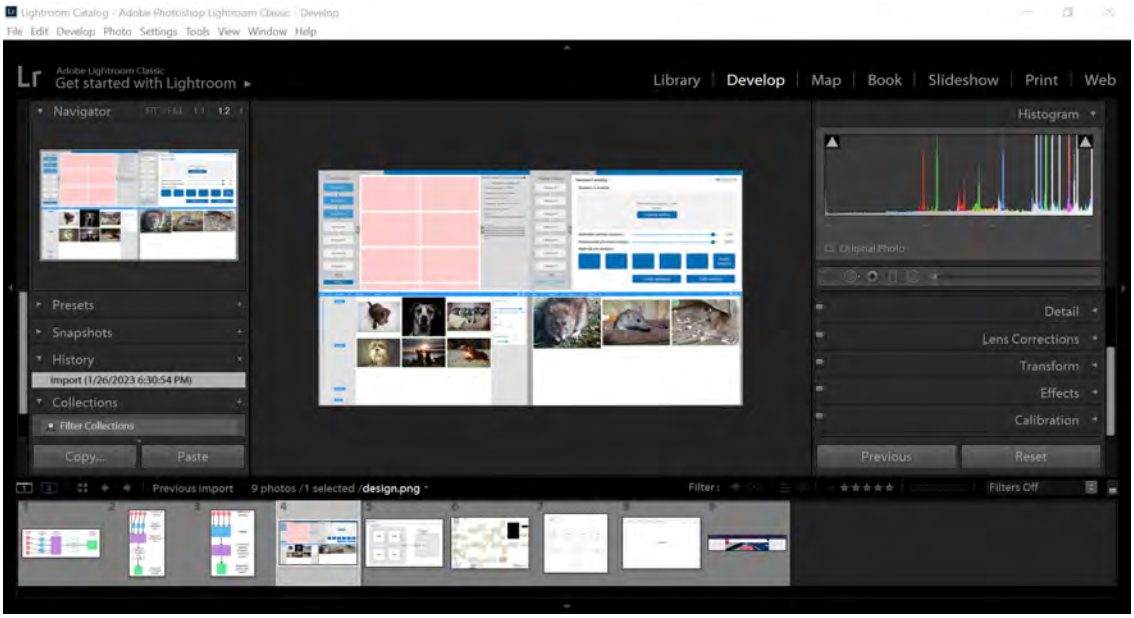


Figure 4.3: Collapsible panels pattern implemented in Adobe Lightroom [1].

Inspiration can in this case be drawn from Lightroom, which integrated whole range of tools used for image manipulation, which can be selectively expanded to reduce the amount of visual clutter the user needs to navigate through in order to perform desired actions. These tool tabs are placed on the expandable side bars on the left, right, and bottom of the application which can be seen in figure 4.3. This is a practical implementation of the collapsable panels pattern [93]. Three collapsable panels are not very common in web applications where they are used extensively, when, for example, the online navigation application Mapy.cz uses this pattern for the search bar or route planning with these functions contained in only one panel. But Lightroom is an application for advanced users, so a more complex user interface is to be expected. As users of the Facis application are also expected to be advanced users, adopting two collapsable side panels will increase the complexity of the interface by a bit but will still be manageable. Another point in favour of this approach is, that a lot of specialty tools use collapsable panels so this pattern is almost certainly already known to users [61].

4.3 Prototyping

Prototyping is a crucial part of developing the user interface. The ability to create a design that at least resembles the final product goes a long way in getting the point across during the meetings, which helps the client to express their opinions and concerns regarding this design. In recent years, many revolutionary applications have been developed for designing, wireframing, and prototyping applications. Despite the progress in many cases, less complex tools, sometimes even physical tools, can be used to increase the productivity and efficiency of this process [87].

4.3.1 Wireframes

Wireframes are an integral part of the user interface development [88]. They played an important role in this particular project due to the unclear vision and flexible boundaries the team was presented with. To start prototyping, a vague description of the project was needed. After that, I was able to draw the first frames, which were discussed with the rest of the team.

Figure 4.4: Wireframes with panels expanded and contracted.

The very first wireframe iteration was on paper, as shown by figure 4.4. This enabled a more open conversation and a hands-on approach that promoted conversation. The benefits of this hands-on approach are described by Sutipitakwong [88]. The biggest drawback of digital wireframes is the need to be familiar with the design tools. This is not a problem when dealing with experienced designers, but when dealing with people outside of creative fields, this will be a problem. This results in the client not being able to directly express his wishes and even drawing or modify the design on the spot in the cases of on-paper wireframe.

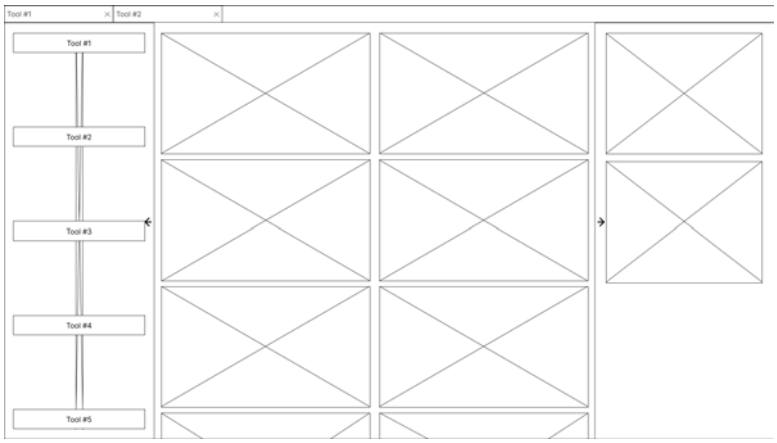


Figure 4.5: Digital wireframe created with Figma

To clearly present the wireframes and further discuss aspects of the interface, digital wireframes were created using the Figma online design and prototyping tool[51]. This design reflects the changes agreed upon when discussing the paper version of the design. This conversation resulted in the addition of the progress bar, which indicates which of the tools are already done and which are yet to be finished or have not even started yet. It can be seen on the left side under the tool buttons in the collapsable panel of the picture 4.5.

4.3.2 Prototyping

After the initial development phase, both I and the entire team had a unified vision for where the project is heading and how the user interface part will look like. To finalise the design phase to move to implementation-related topics, an interactive mock-up was designed. This was accomplished with online tool Figma[51]. This mock-up enables interactive exploration of the application within a limited scope.

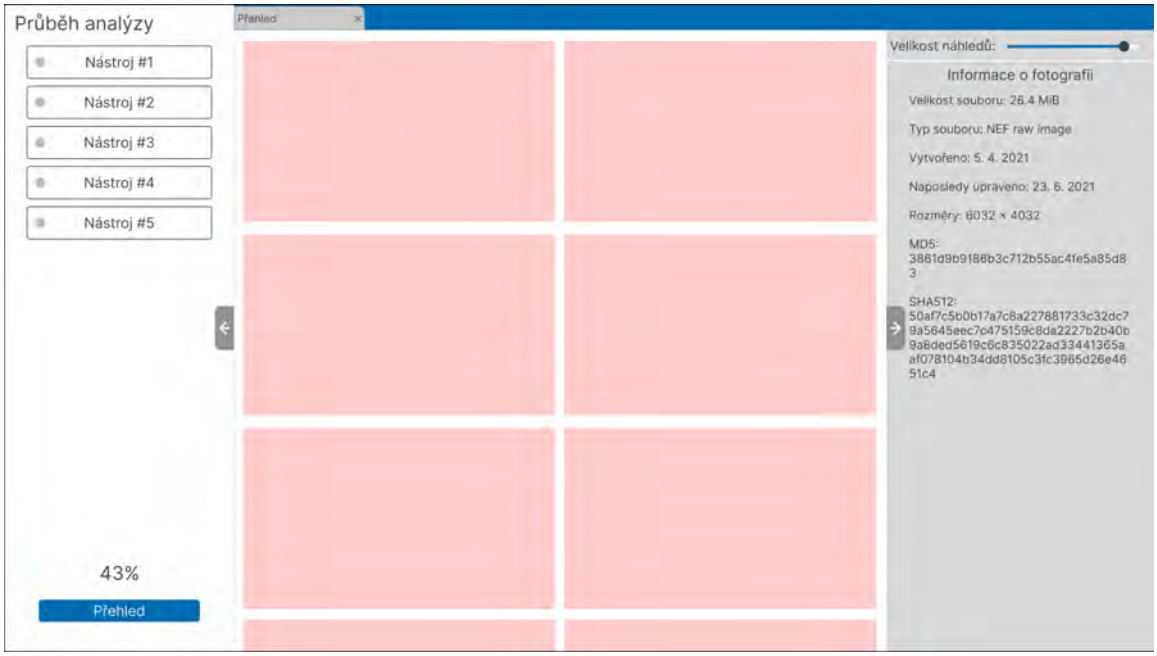


Figure 4.6: Prototype created with Figma.

The most noticeable change in the final prototype is the look of the tool state indicator, which was changed from the underlying loading bar to indicator dots similar to the indicators used to display the state of the pipeline tools by GitLab [43]. It can be seen in the figure 4.6

4.4 Usability evaluation

Now, when the concept of usability is outlined, it is important to specify how it can be measured and which techniques and methods can be used. These methods should be established in advance because testing the application in its finished state brings less of an advantage than using these methods during design and development.

4.4.1 Evaluation factors

The factors which usability can be evaluated by are mentioned in previous section, I have outlined five of them, but for the purpose of testing, it is not necessary to evaluate each of them separately because some of them need to work together and the effect they have on each other is so significant that it is almost impossible which one is at fault [77][50].

Effectiveness and Efficiency

Both of these factors together have an effect on the time in which the user can accomplish the task they set out to do. While having processes in place to make the work done in the application as efficient as possible is good, it does not reduce the time spent on one task if the user makes a lot of mistakes by using powerful functions in an undesired way. For example, submitting new configuration of a tool which is going to be run as a part of a larger pipeline which contains a mistake has more of an impact on the time it takes to complete the analysis a if then clearer but slower process of altering the configuration [6].

Not making mistakes is a good thing, but being overly protective is going to slow more advanced users down and potentially become very annoying. This is a trade-off, which needs to be considered based on the type of user the application focusses on.

Engagement

Engagement is a very important metric when evaluating commercial products that are intended for wide audiences and that rely on the reach of new customers to generate revenue. This is not the case however with internal tools, it is good to keep users engaged, but it is not worth sacrificing more complicated features that are going to raise the productivity in the long term in the sake of engagement [6].

Error tolerance and ease of learning

Both error tolerance and ease of learning have effect on the resulting learning curve; applications with steep learning curve have typically higher stake actions implemented, but also lead to bigger reward when user uses them correctly. However, applications and tools with gradual learning curves are more forgiving, but are usually not as powerful. The optimal scenario is to have a little bit from both worlds, but in this case, to achieve such a state would require exponentially more work. It is better to instead focus on the userbase and its skill set, for internal tools created with power users in mind, steeper learning curve is not a problem and they can benefit from the learnt experiences in the long term [6].

4.4.2 Evaluation methods

With the evaluation factors established, the methods verifying the set out goals need to be specified as well, in the following sections, commonly used tools for assessing the fulfilment of these goals.

Testing scenarios

Testing scenarios are a form of user testing in which users are given a scenario to complete. In contrast to a simple checklist, which would give step-by-step instructions on what the user should do, scenarios give general goal and or describe said goal. This helps in assessing whether the user is having trouble understanding the layout of the application and gives more accurate outputs in terms of time spent and errors made along the way [49].

Moderated and unmoderated testing

For in-person testing, there are two basic approaches. Moderated testing, during which the user is observed by a moderator who can ask follow-up questions if the user reacts to something they did or that interests or annoyed them while using the application [50].

The second approach is unmoderated, which is more natural, and users are more relaxed. Not being distracted from the task at hand gives more accurate results in terms of raw metrics, but valuable personal feedback can be missed. The user might say „Hm, interesting.“, but naturally will not elaborate further and the reason behind this reaction is not going to make it into the feedback.[49]

Time measurements

The time to complete scenario or task is a metric that can be an indicator of the difficulty level the action is at or how engaged the user is while using the application. The time spent will vary on the type of user as well, for example, users familiar with complex software like development environments or video editing tools are more likely to complete the test faster. Therefore, it is important to compare the measurements with this in mind.[50]

Clicks to complete action

The amount of clicks it takes to get from one screen to another indicates similarly to the time spent the difficulty and engagement level of the user. While exploring the application, users will make mistakes which will increase the count, but it also might indicate convoluted navigation. Counting clicks is one way to find this out; however, it is better to discover this during the development by drawing screen map with annotated number of clicks it takes for the user to get to the certain screen.

Errors to complete task

When evaluating the number of errors the user made during the testing is metric that could be interpreted multiple ways. If the application implements safe exploration pattern which allows users to make mistakes without punishing them significantly, errors have less of an impact and user is therefore not afraid to make them. High error in this case can indicated complicated or misguiding interface or desire of the user to explore application and its functions. To distinguish between these to scenarios, interview with the user needs to be conducted to decide which of the options occurred.[50]

Chapter 5

Application design

This chapter is going to present design choices, what lead to them and how I plan to achieve the requirements expected by users. Details are going to be given in several key areas such as client - server communication, tool integration, and security.

5.1 Client - server communication

To exchange information between the client application and the server, the Hypertext Transfer Protocol, better known as HTTP is used together with the JavaScript Object Notation (JSON) format for serialising data. This combination has become standard due to its simplicity and flexibility, it is also the recommended way of sending data over the network according to client application developers [38].

5.1.1 API specification

In order to write client application for a backend running on a server, it is important to know what kind of requests can the application send and what should it expect to receive. Defining this communication is API, which is a shortcut for *Application Programming Interface*, with this interface it is possible to generate part of both the client and server alike. For this reason, I have created OpenAPI specification, which proposed several requests divided into categories depending on their use. This was later transformed by a colleague creating the backend part of the application; the technology used for the backend has the possibility to create OpenAPI specification which in documents the request in a way they were actually implemented.

The following sections describe the categories of requests, why they were chosen this way, and how they will affect the client implementation.

Authentication endpoints

Authentication endpoints are all of type *POST* allowing the use of parameters in path and also to send body in which data in JSON format can be stored. One for singing in, which requires username and password to get authentication token. And one which will log the user out and invalidate its token. This ensures that the session created by the user will not remain open for anyone to use after the user leaves.

Case endpoints

Case endpoints allow creation, indexing, and modification of cases. It is possible to request all cases assigned to a user, this will provide case only the *IDs* of the workers and resources assigned to this case. Creating a new case and editing existing one is also possible. It is also possible to assign and remove users from a case including assigning and changing the user designated as case lead.

Orchestrator endpoints

Orchestrator endpoints allow client to interact with orchestrator and pipeline it creates, meaning it is possible to use endpoints to run the tool prepared in the pipeline, check their status and the status of the pipeline as a whole.

Resources endpoints

Resource endpoints serve the purpose of indexing available resources so that they could be assigned to cases and tools could work with them. It is possible to get the name of the available resources and their path.

Results endpoints

Endpoints serving results require *IDs* of the case, tool and base image from which they were created. This results in the result of the analysis by all tools in the pipeline being returned to the client. The number of individual results is not restricted. Name of the previews linked to the results of a tool within a case can also be requested, these are later used in conjunction with case and tool ID to create URL to retrieve the preview image with.

Tools endpoints

Endpoints providing access to tools allow client to create, index, and run or apply the settings for sorting or filtering for available tools. Individual configuration of tools are not case specific and therefore their settings can be listed without limitation of user permissions. To run the tool, however, it needs to already be present in the pipeline. That can be achieved by other tool endpoints which serve the purpose of adding and removing tools to and from the pipeline.

User endpoints

User endpoints serve primarily administrative purposes, the main one being listing and creating of users by the administrator or the user themselves. The secondary use of these endpoints is to get user details in order for them to be displayed in the UI.

5.2 Flexible tool integration

One of the primary goals of this application is to accommodate many different tools in the most flexible way possible. To do so, the widgets representing the tools could not have been prepared in advance because of the restriction this would place on the developers behind the tools themselves.

Another goal is to present the data produced by the tools so that the results would be easily understandable at first glance, but also had enough detail so that evaluation of the results would not lead to unnecessary errors.

The approach I chose for tool integration was created with flexibility and expandability in mind. It utilises JSON configuration files created by the developers of said tools, which are then transformed into widgets representing the tools in the user interface.

5.2.1 Setting, filtering and sorting

In order to use the provided tools effectively and as easily as possible, a proper interface must be created to accommodate these needs. The primary focus of this application is image analysis; to carry out this task successfully, sufficient controls need to be created in three different areas.

Setting up the tools is the first step in getting the results, as the granularity with which the tools can be controlled decides the quality of the results produced by them. Inexperienced users are going to need help with the settings, this has a great impact on usability, so employing a sensible defaults pattern with default settings suggested by the creators of the tools is very important [93].

Sorting the results is very helpful when large amount of data is being analysed, the tools planned to be integrated in this project so far are limited in the variables the results can be sorted by, therefore, the sensible default set by the developers of said tools will most likely remain unchanged, however, this does not mean that the sorting interface should not be included because the lack of such a fundamental feature many users are used to could have negative impact on the user experience [61].

Filtering is a key feature which can greatly improve the effectiveness and ease of use for the users. The filtering options will be created by the tool developers, which allows the queries to be tool-specific and as granular as possible.

Widget types

The types of widgets used to represent the tools were derived from the types of entries present in the currently most frequently used form, the HTML Form. I have also added input types present in the application used by field experts to better accommodate advanced users. Each type of widget has a description and a unique way of representing input.

The option widget is an alternative to radio input type, the output is the same (one of the options), but instead of the options being represented by radio buttons, they are represented by drop-down options, this was done to reduce space taken by the widget.

The range widget is inspired by the control element from Adobe Lightroom, specifically the slider used for majority of the inputs in development section of the application. This widget allows users to select values on a defined range, it is best for numerical ranges divisible into small number of chunks.

The checkbox widget has the same function as its HTML counterpart.

The text widget is the most flexible of all the widget types, it has the same function as the HTML text input, with the difference that this widget does not have the maximum character length set by default.

JSON representation

To construct the tool widgets, they need to be represented in some way. I chose JSON as it is readable by both computers and to some extent humans, it is also the most common serialisation format that can be easily serialised and deserialized by Flutter and most of front end technologies.

The format of the JSON representation was heavily influenced by the ability of the OpenAPI generator to create models usable without changing the generated code which would render the generator useless.

Each tool has its unique ID and three lists of parameters discussed above. All of these lists can be filled with 0 to N parameters; they each have name, description, type, and value. Then each parameter uses special fields.

- The option widget uses the option field which is a list of values of any type, which are then converted to string.
- The range widget uses the start field together with length fields that defines the value range, there is also an increment field that determines how many sections will the range be divided in.
- Text and the checkbox widget both use just the value field with the difference being the type, the checkbox widget only accepts true or false values compatible with this case-insensitive regex *(t(rue)*|f(alse)*)*.

To demonstrate how such a JSON is going to look like, listing 1 is an example defining a tool with two parameters in the *setting* section. In a similar way, parameters could be added to *filter* and *sort* section of the configuration.

```
{
  "config_id": 0,
  "filter": [...],
  "setting": [
    {
      "description": "Parameter description",
      "name": "Parm1",
      "type": 3,
      "value": "text-parametr-example-value"
    },
    {
      "description": "Parameter description",
      "name": "Parm2",
      "type": 2,
      "value": "True"
    },
  ],
  "sort": [...]
}
```

Listing 1: Generic tool configuration JSON representation.

Configuration interpretation

Once the JSON specification of a tool is received by the client API, it is deserialized into *ToolConfig* class and all the parameters into *Parameter* classes. The class object is then further digested with tool widgets.

The interpretation of the configuration received from server takes place in the *ToolWidget* which uses the type of widget determined by the *type* field in the parameter. Each parameter type has its own widget, these widgets also include lambda function, which sets new value in the tool model which can then be submitted to the server.

5.2.2 Presentation of analysis results

Representation of results is one of the primary tasks to be achieved with this application. There was a way that would be flexible enough to visualise the results of different tools with different output formats. Some of the tools output one integer, some string, but there already are and might in the future be tools that use output values of different types.

Format of results

Because of the already-mentioned unsure nature of the output of the tools, a simple string was ruled out immediately. It was clear from the start that structure holding the value of the result is going to have at least two parameters, first representing what kind of value is it representing, second one holding the actual value, and third one which is going to add the accuracy of the result. The last one is needed because most of the tools are going to use machine learning, in which case the certainty of the result is crucial. For some tools, this would be enough, for example, tools giving age estimates or number of people in image. Basing the design of the structure, which has such an important role in the system and would most certainly need to be a step in a direction of maintainability. The solution to this problem I came up with is not very complicated, it is a list. The fact that all of the displayed values are going to be converted to string anyway makes the choice of data structure representing composite results easier. Every result entry has the *ID* of the tool it came from written in it. This allows easy separation and linking of results to their tools. It is possible that in the future, some tools will need a more complex structure to represent their results, but that is beyond the scope of this project.

Visualization of results

Inspiration for the visualisation of the results itself came from an unexpected place. To be efficient but thorough, the user needs to find all the desired results in the shortest time possible. This is a process similar to online shopping, the goal is to obtain the best item for smallest price in shortest time possible.

There are two visual components working together, the first one being text describing the result in line with the value of the result, and trailing closely behind it is the confidence of the tool producing the result. The second one is a strip filled with colour indicating confidence using a a ratio between green and red section. This is redundant, but with the amount of data at play, it is much better to pay attention to colour changing as the user scrolls past the result, and it is possible to tell nearly instantly if the result produced by a tool from the pipeline is worth examining further.

5.3 Security

Security is an important part of most systems accessed by users, this is also true for this project. Even more so given the type of users and the setting at which this application will be used. As this project aims to be multi-user software, it was necessary to prepare user authentication and to separate data users are going to have access to. As this is a proof of concept application the security is considered more from user experience standpoint rather then focusing on encryption methods and safe client-server communication.

5.3.1 Authentication

One of the simplest ways to implement authentication in any modern application is to use a third-party solution. This, however, comes with a few problems which prove to be show stoppers for this application.

The biggest obstacle that prevents the use of tools such as OAuth2 [71] and other similar ones is the type of end user for whom this application is intended. These tools authenticate users through service providers; this is a no go for internal tool operating with sensitive and possibly classified data. Therefore, an in-house solution was needed to mitigate the possibility of unauthorised user retrieving data from the system.

The solution picked for this project was inspired by the aforementioned OAuth2 which uses tokens. One of the most commonly used tokens is JWT tokens that stand for *JSON Web Token* and are an open industry standard RFC 7519 method to represent claims securely between two parties [54].

In the first step, the user provides log-in credentials which are sent off to the server that validates them. The authentication then determines if the user credentials are correct, and the system issues token, which is sent to the client. The data is then stored in the client for future use when accessing protected endpoints.

The token contains payload which provides information about the user (their ID) as well as information about the time when the token was created and its expiration time. The expiration time can then be used to automatically log out the user after the token expires; this reduces the chance of someone unauthorised interacting with the application and the data it contains. An example payload of this token can be seen in listing 2.

```
{
    "id": "0f08dab4-1e1a-42e3-8f9e-be56a1e2c873",
    "name": "John Doe",
    "iat": 1680698513,
    "exp": 1680698873
}
```

Listing 2: JWT token payload example

5.3.2 User roles and permissions

For the system to be manageable in a secure way, user roles must be implemented. Each of these roles has a different set of permissions and actions that they are going to be able to use and trigger. There are three roles; each of them is well known from information

systems used on a daily basis. The role in possession of the most powerful privileges is administrator, then with fewer permissions the role of case lead and finally the analytic.

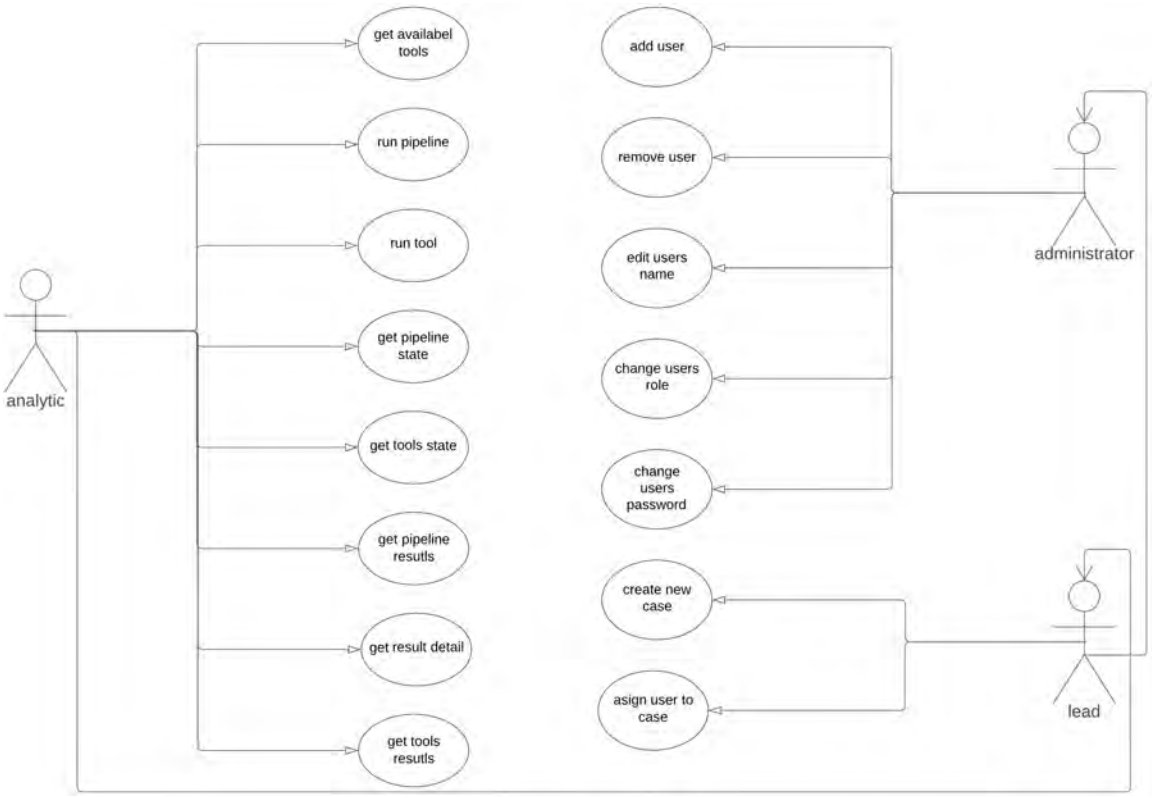


Figure 5.1: User case diagram

The figure 5.1 shows a use case diagram with roles and their usecases. The role of administrator was originally supposed to be completely detached from the case analysis part of the program, the forensic expert however specifically asked for the administrator role to be able to use the tools as well. This ended up being easier to implement because as can be seen in the graph, the roles are inheriting their permissions.

Administrator

Administrator is a role different from the other two, besides the impact of the privileges, the user with it is not going to have the ability to inspect the results of an ongoing case and neither are they going to have the ability to run the analysis pipeline. Its purpose is to add and remove users with the option to edit their roles, reset their passwords, and do other administrative tasks. The choice to deny the administrator interacting with the tools and data regarding individual cases has been made with the intention of the administrator not having to come in contact with sensitive material which would unnecessary elevate the risk of data leak.

Case lead

The role of case lead is one with some administrative power added on top, mainly in regards to creating new cases and assigning analytics to them. Users with this assigned role can

also view the results and interact with the case pipeline to a full extent. Case lead can also archive the case or remove any users from it; they however do not have access to the setting screen which controls environment settings such as the URL of the machine running the server for this application.

Analytic

Analytic is the role with least permissions, users with it are assigned to cases which grants them permissions to browse through results, user filter, sorting and run tools with desired settings.

Chapter 6

Implementation

This chapters focuses on laying out technologies and their usage while developing this application. The dynamic nature of this project motivated the programming language Dart together with the UI development kit Flutter. As the basic functionality of both Dart and Flutter was not sufficient to cover all use cases, several packages were used thanks to the great expandability and modularity of the chosen technologies.

6.1 Used tools and libraries

In this section, a deeper look at both the programming language itself and the UI kit built using it will be given, as well as the used packages.

6.1.1 Programming language

The programming language chosen for this project is Dart [20]. This language is an open source project developed by Google. Dart is an object-orientated language, uses a garbage collector and can be compiled into machine code. The primary motivation behind the development of this language was to offer the most productive programming language for multiplatform development. Dart is particularly suited for client development as it implements many quality of life features like the possibility to hot reload the application or tools to format, analyse and test the code. The language itself is type safe and uses static type checking; this ensures that the variable's value matches with its static type. There, however, is an option to use dynamic variables with the use of dynamic type. It is a good practise to avoid the usage of dynamic if possible or limit it's use to development for experimenting.[20]

Dart also implements sound null safety which was added in version 3, it is voluntary in version 2 and its subversions and can be enabled by a setting. Null safety guards errors that would otherwise be caused by accessing the variable set to null. Null safety support has its basis in three core design principles.[22]

- **Non-nullable by default.** Unless explicitly declared that Dart should allow the variable to hold the null value. The default of non-null was chosen after research, which showed that this was by far the most common choice in APIs.
- **Incrementally adoptable.** The developer has the option of adopting null safety when they choose to. Null-safe and non-null-safe code can be used in the same project together. Migration tools are also provided to make the adoption easier.

- **Fully sound.** Null safety in Dart is good; this enables compiler optimisations. If the type system determines that a variable is not null and is not nullable, then it will never be null and so runtime check are not necessary when executing the code. Many languages implement null safety, but very few implement sound null safety. An example of language that implements null sound safety is Swift.

Together, these principals prevent null related mistakes, which tend to be common in languages not implementing null safety. By knowing exactly when the variable can hold null value and when it cannot, developers can focus on writing code with checks preventing runtime errors being done by the compiler before the programmes even starts. In other words, the code should be safe by default.[34]

Dart, similarly to most modern programming languages, comes with a lot of libraries [19]. The libraries developed directly by the dart team are called *core libraries*. These libraries can be separated into three categories [18].

- **Multi-platform libraries**: Libraries are supported by all platforms adding general functionality like asynchronous programming (dart:async) or functions to convert between different data representations (dart:convert).
- **Native platform libraries**: Libraries working on Dart native platform which allows leveraging the full potential of Dart native platform, this includes Just In Time code compilation and Ahead of the Time compilation. The most notable package in this category is dart:io, as it allows Dart to interact with files, sockets, HTTP, and other I/O support for non-web applications. Another useful library is dart:isolate which enables concurrent programming using isolates, these are independent workers which are similar to threads but do not share memory and can communicate only with the use of messages.
- **Web platform libraries**: Libraries working on the Dart Web platform. Dart web is in contrast with the other categories compiled into JavaScript; this allows the Dart code to run in browser, for example, Google's own Chrome.

Beyond the already mentioned libraries, there is a lot more created by both the Dart team and thanks to the open source nature of the project, the community.

6.1.2 UI development technology

The choice of technology for the implementation of UI was crucial in order to develop a comprehensive desktop application in a short amount of time. After researching available desktop front technologies, Flutter was chosen as the best option for this project. In the following section, all considered options will be listed with reasoning on why they were either used or eliminated. In the case of the chosen option, a deeper dive will highlight some of the advantages and capabilities of this alternative.

Expectations and requirements

To develop an application successfully, there need to be requirements and expectations that the technology has to meet.

The first thing that comes to mind is what the technologies origin is, whether it is completely community driven or a for profit company's project. In recent years there is

a new alternative gaining traction with projects being started by companies but made open source, so the community has as say in the direction of the project.

Careful consideration should also be given to the limits of the technology. It should not limit the vision and goals of the application and therefore hinder the development process. This ranges from lack of functional capabilities to the devices that the technology supports.

Popularity plays an important role in the amount of support developers can get from the community, as well as the timeframe in which the application can be maintained with reasonable development cost. This, however, can go both ways, while older technology will be certainly more mature than a new one, older code bases are harder to maintain. That can result in the support being dropped completely or, at least, slower fixes and rollouts of new features. Some of the factors contributing to the popularity of the technology is its purpose and target community, niche technology aimed at a small community of specialists has a lower chance of being actively maintained compared to more mainstream and general purpose technologies.

Ability of the application to take the desired shape is an important part of conveying desired information to the end user. There are three fundamental ways UI technologies interact with the design languages of the platforms they are intended to run on [73].

Native technologies follow the design language of their specific platform, and application created using them will visually blend seamlessly into the rest of the platforms environment. They will provide components written in the native language of the platform using the same components as the system itself. In this case, it is easier to make a platform compliant application, but creating a unique look is generally harder to achieve. An example of such technology is WFP.NET for Windows or SwiftUI for Apple platforms [25].

Alternative technologies do not adhere to the design language of the platform on which the application is running. This is common for multi-platform technologies, as it does not make sense to limit the visual appearance to follow the design language when no single target platform is specified [30]. On the other hand, creating unique UI or new design language even tends to easier. To achieve full control over the look of the application there needs to be a way to render components without the use of platform-specific API. This is achieved by rendering components using a separate render engine. For example in case of Flutter the engine is called Skia which uses different graphics libraries based on which platform is the application running at. Similarly to Flutter, Qt uses Qt Quick Scene Graph which also takes advantage of multiple graphics APIs like OpenGL ES, OpenGL, Vulkan or Metal [12].

The cost of technology also plays a big role in the decision making process. Before the development process starts, it is a good practise to set up a budget be it of time, money, or in the optimal scenario both. The price for technology in both time and money might, in the long run, determine the success of the development endeavour. As increasing the cost of development hurts adoption of the technology, most of the technologies mentioned in the next segment are completely free or require payment only when commercial licence is required [12].

Qt

Qt is a cross-platform framework for developing user interfaces, aims to streamline application development with the use of pre-made widgets. This is something that is considered standard for today's UI frameworks, but at the time of release in 1995, this feature was seen as great innovation [53]. Qt comes with its own set of tools, for example Qt Creator,

which is a cross-platform development environment for C++ and QML [81]. The Qt modelling language, or QML for short, is a declarative language created by the Qt team with readability in mind that allows developers to describe the visual components themselves as well as the relations between them [13]. Qt performance is very good because the use of C++ and the mature nature of the project [31]. Qt is available under multiple licences, for example GPL or LGPL. This has a limited scope of usability because usage under these licences is only permissible if the developed project is open source, for student/academic purposes, or a hobby project [12].

GTK

GTK is a free and open-source widget toolkit that supports cross-platform development of graphical user interfaces. The reason behind creating GTK was to use it for the development of GIMP which is a free and open open source image manipulation programme. This is also where the name comes from, GTK is an abbreviation of *GIMP Tool Kit* and the authors did so that the origins of the projects were remembered [91]. The small team behind the projects in its beginning was Peter Mattis, Spencer Kimball, and Josh Macdonald. Now the primary maintainers are members of the GTK team with a large community of developers helping them push the project further [92].

In terms of developer numbers and active community, GTK is doing good with some notable projects being built with this software. The best known is the one that started the development of GTK itself, GIMP. Then a lot of GNOME tools followed as the The GNOME Project is the biggest contributor to GTK. For example, GNOME Shell, GNOME Panel, GParted, or Inkscape [14].

The languages used with GTK are C/C++ and Rust, while the former is well known and the chance a developer will be completely new to it is small, it is more complex than modern languages made with front-end technologies in mind. A similar situation arises when it comes to Rust. GTK supports both Linux and Windows, but its performance on Windows is worse than native or even cross-platform technologies [45].

WPF

Windows Presentation Foundation offers native Windows experience with applications that developers can create using it. WPF is a free and open-source project by Microsoft released in 2006 as a part of .NET Framework 3.0 [55]. It uses the combination of the extended application markup language for defining the placement of widgets in UI and C# for Code-behind, which allows clean separation of the visual representation of the application and the presentation logic. Using XAML is not the only way to defy what the application should look like [44]. Microsoft also provides Blend for Visual Studio Code which allows developers to create an application without writing XAML code, instead a menus with individual widgets that can be dragged around and arranged at will.

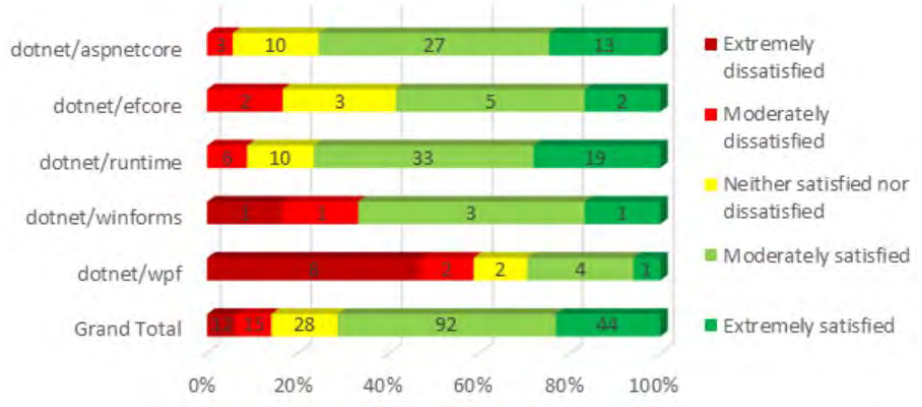


Figure 6.1: Overall, how do you rate your experience with the repo you selected? by Microsoft [75]

The community around WPF is sizable but rarely positive about the technology; claims about pull requests taking too long from submission to merge or even comment from developers are discouraging for the developers. This can be seen after reviewing the results of a Microsoft developer survey in which dissatisfaction with WPF can be clearly observed from the results of a developer survey show in figure 6.1 in article by David Ramel [75].

WinUI 3.0 UWP

Universal Windows Platform is a concept by Microsoft which aims to unify application development for all devices and platforms running the Microsoft-made operating system [72]. The advantage with this approach is that a single code base will cover all Microsoft platforms, which leads to an obvious downside when the whole user base is using these platforms. WinUI brings *Fluent* design language to UWP application development [29]. UWP with and without WinUI use similarly to WPF combination of XAML and another language, for example, C# for presentation logic [74].

UPW has been criticised for bringing yet another technology into the GUI development space with uncertain future. It has also generated backlash for building a *walled garden* or *platform-within-a-platform* within Windows 10 with which the UWP is released [78].

React Native

React Native is an open source project released by Facebook in 2015 and started out as technology for mobile applications, but also allows developers to create applications for desktop [57]. According to platforms it can run on, besides iOS and Android, React Native also supports Windows, macOS and Linux development [30]. React, also known as ReactJS is a Javascript library and so is React Native [27]. Javascript being one of the most popular languages today according to GitHub data is a valuable head start and will help developers familiarise themselves with this React faster [42]. The way it achieves native look is by using platform native components and tying them together using the React Native Bridge. This means that while the view is provided by components belonging to the platform on which the application is running, the login is handled by Javascript code. The performance of React Native is fast for non-native technology, but due to the slower Javascript code, other

multiplatform technologies can achieve faster response to user interaction, for example, Flutter.

Big community is always a bonus and in this case React Native is among the most popular multiplatform technologies for desktop application development. Even the size of the community however sometimes wasn't enough for packages lose maintainers and got abandoned.[85]

Flutter

Flutter is open-source, cross-platform framework created and maintained by Google. The original release of the framework was in 2017, since then Flutter grew in terms of both developer numbers and the platforms it supports [85]. In the beginning, Flutter was focused solely on mobile development for Android and iOS, since then it moved to wearable platforms, web and desktop. The fast growth of community was in part thanks to the amount of platforms it can run on, but more importantly thanks to the active support, flexibility and performance [69]. In fact the performance is gained by the application being compiled to binary code for each platform helps Flutter to beat other cross-platform techonologies such as Elector or React Native [86]. Besides Google which uses Flutter for their Android applications like Wallet or Google play, other companies have chose this framework for their applications. For example BMW - My BMW app, eBay - eBay Motors or Etsy and Canonical, the developers behind projects like Ubuntu desktop chose Flutter as the default choice for future mobile and desktop applications [56][24].

The programming language used by Flutter is Dart, which is expanded on in the previous section. Another advantage of Flutter is the flexibility it offers, the default design language it is using is Material 2 with opt-in preview of Material 3. There is also existing Cupertino style with others created by the community such as Fluent for Windows, this wide range of styles is possible thanks to the way Flutter is rendering widgets. That is done by the Skia rendering engine; this means that the developer controls every individual pixes of the application. This allows for creating fully custom look, even though using pre-made widgets and their combinations with custom styling is by far the fastest way to develop an application in Flutter.

6.1.3 Packages

Packages are a simple and effective way of expanding the capabilities of both Dart and Flutter. Some of the packages add new widgets or whole styles like fluent_ui and others enable completely new functionality like http which allows sending HTTP requests [8][21].

State management packages

Provider is a state management package that wraps *InheritedWidget* and makes it easier to use. The way it simplifies development is by implementing several quality of live improving features, such as lazy loading, common way to consume *InheritedWidget*, and more [70]. It also increases scalability and reduces boilerplate code. The provider package is recommended by the Flutter developers as a state management package with several other alternatives, with the most notable being the bloc package which uses the provider package and adds more advanced features. Good state management is crucial for any complex application, as maintaining a consistent state of the application is a key part of good user experience [35].

Get_it package is a simple service locator which can partly replace other state management options like *InheritedWidget* or the provider package. The main advantage of get_it over already mentioned provider package is the ability to access service without the need for *BuildContext* [37]. It is also allows developers to easily switch between different implementations of one abstract class which make development and testing easier. The price for better maintenance and improved unit tests is more code, because having one abstract class with more implementation of this class leads to multiple documents needing to be maintained at the same time [65].

Styling

Fluent icons package brings a list of icons based on Fluent UI System Icons pack which is modern set of icons used in Fluent design language created by Microsoft. This package allows the created application to closer resemble native Windows look [4][32].

Network communication

Package called http is a multiplatform future-based library for making HTTP requests. This package is developed by Dart team and offers first party support for all platforms [21].

Dio is a more advanced alternative to the http package maintained by flutter.cn. This package offers an HTTP client with global configuration, interceptors, request cancellation, file uploading/downloading, and other features. The Dio package is expanded upon by other complementary plugins such as *dio_cookie_manager* or *dio_smart_retry.* Although applications with complex network communication needs will benefit from added features, it also increases the overall amount of work needed to get the package running and integrate it properly [36][33].

FACIS API is and in house developed API for the communication with server providing analysis data and images. This package is generated with OpenAPI generator from the OpenAPI specification generated by the server [16].

Testing

Test package provides the ability to write and run custom tests. The use of this package makes testing cross-platform applications much easier due to its ability to run tests on each platform separately or at once covering all desired platforms. Tests can also be run straight on the Dart VM which will not load the full test runner and will miss some features, this can be used to run singular test to verify if some specific change resulted in bug or not. It is also possible to split test suits to the so-called shards and run them separately. Shuffling the tests is also an option that can verify not only if the components of the application are working properly, but also if the application contains some dangerous dependencies, which might cause an error in case that specific actions are executed in given order.

This package also has the ability to generate code coverage of the tests using LCOV which is a GNU tool which provides information about what parts of the application are executed. There is also an option to restrict the tests to particular platform[23].

Others

Dartz is a community package created by Björn Sperber which adds new types and concepts from function languages such as Scala or Haskell to Dart [84]. This package is especially

useful for a very robust error handling which is always welcomed in all types of applications. The ability to inform user about error in a user friendly way and recover from the error state if it is not too severe is a big improvement of user experience and standard in all modern applications [76][84].

6.2 Mock API

It is common practise to develop the backend of an application first, and this is the case especially true for heavily data driven applications such as this one [83]. In this case, however, it was decided that the front end of the application should be developed simultaneously. This approach has the benefit of shorter feedback loops which help to tailor the application better. The need to start development resulted in the need to create Mock API.

6.2.1 Flexible API creation

As this API is only a mock of the production one, it is not effective to create a completely new and separate one. The big advantage of the technology chosen by the backend team working on server for this application is the ability to generate OpenAPI specification. That can be then used to generate mock server in a similar way that the client used in this application was created.

The constant hand written part of the API is in memory storage of data which is seeded in a way that allows testing and presentation of the application to both developers and future users. This also helps to speed up the development because of the ability to implement the API as if it was already created.

The technology chosen for the mock server was Flask. The main reason behind this decision was that the options provided by the generator were technologies that I am not familiar with and are unnecessary for the purpose they were supposed to serve. Flask is a bare-bones web framework written with Python, its simplicity is an advantage for creating simple simple applications, just like this one.

For serving the images, nginx [89] which is a HTTP and reverse proxy server, meaning it can serve data based on the URL (Uniform Resource Locator) [15] which it receives from the client. This means it can be used to get images from server as well as redirect the requests to other servers. By combining the Flask API with nginx in separate containers using Docker development environment can be created in very short time[26].

6.2.2 Unified integration

The fact that both the mock and production server share the same OpenAPI specification means that the client generated from it is the same for both of them, which massively reduces labour needed for switching between them, resulting in faster integration.

To make the switching between mock and production API even faster and easier, services implementing both clients were made. Together with dependency injection implemented with *getIt* package, it is possible to switch between the APIs with one flag [65].

6.3 Results

In this section the resulting application will be presented with screens and complimentary commentary describing the screens current state choices that lead to the resulting applica-

tion. The primary focus will be on the parts of the application that implement the most important features, notably, sorting, filtering and pipeline configuration of individual tools. These pictures are place in full resolution in the section A.

6.3.1 Overview of screens

The figure A.1 depicts the screen which serves the purpose of choosing which case to open as well as the option to add a new case and the new case screen itself which allows users to create new case and add users to it under the condition they know the ID of the user they want to add. It is also possible to create new user if the current one has sufficient permissions to do so. The user that is assigned as a case lead can be also changed by clicking at the users icon.

Next couple of images from figure A.4 to A.6 shows two out of three steps of analysis setup process which was divided into three steps for the user to focus on the steps individually rather then try to set everything up in one pass. The image of the two shows the step in which the user can select disk they want to analyse and add notes to them, for example the type of device the disk is from or where was it found. The second image shows the menu in which the user can add tools to the pipeline and tweak their parameters in the process.

The last set of images in figures A.7 and A.8 displays the main screen and its sub-screen. In these screens it is possible to configure the tools, run them in the pipeline, sort and filter their results. All the tool widgets are created from a configuration stored on the server and can have as much controls as the tool creator wishes. The same applies for the sorting and filtering options. On the left side of the screen, there are state indicators of the tools present in the pipeline, this gives a semaphore like state report about the phase of the analysis of the individual tools. By clicking on the tool indicator a new tab opens in which only the results of the specific tool can be found.

6.3.2 Security

From a security standpoint a couple of standard features were implemented such as automatic logout. The time after which is the user logged out depends on expiration date in JSON response received during login. Particularly the field “exp” in token presented in listing 2 indicates the time and date at which the token ceases to be valid. This is used in the *AuthenticationProvider* class by a login function which after successful login starts asynchronous *autoLogout* function which is non-blocking and waits after the token is not valid anymore and then notifies listeners with the help of the provider package [70] that the user should not be allowed to further use the application. User is then presented with login screen.

6.3.3 Integration of client APIs

Using a mock API is great for testing, but when it is time to use the server the team members of the Facis project are working on a different API client needs to be implemented. Due to the shared timeline of this thesis and the project, a solution that would allow easy exchange of these two APIs was needed. The solution to this problem was the implementation of service injection with the use of getIt package [65]. This meant writing more code, but provided great benefits during the development. The increased amount of code is mainly

the result of every service needing at least two files and initialization. For demonstration purposes, description of tool service will be given.

It was needed to first create an abstract class for the service, in this case *ToolService* which defines names and return types of the methods, but because it is abstract, it does not contain any implementation. Then classes *ToolServiceMock* and *ToolServiceFACIS* were created, both implement similar API, with the difference being, that the mock service implements locally running mock API while the other one implements production API running on a remote server.

After writing both implementations, a global variable was created which serves as service locator meaning the services can be accessed through it. Now it is possible to register the services is a couple different ways, for the purpose of this thesis however, all of the services were registered as singletons as can be seen in listing 3. This means all of the services are available as long as the application is running.

```
1  // This is our global ServiceLocator
2  GetIt getIt = GetIt.instance;
3
4  // In case of producition server
5  getIt.registerSingleton<ToolService>(ToolServiceFACIS(),
6      signalsReady: true);
7
8  // In case of mock server
9  getIt.registerSingleton<ToolService>(ToolServiceMock(),
10     signalsReady: true);
```

Listing 3: Usage of services with getIt package.

6.3.4 Tool widgets creation

The most important part of this work is undoubtedly the process of dynamical creation of widgets representing the tools which will be described in following text. This process starts by JSON configuration stored on a server and ends by functional widget which can be used to interact with the tools.

In the begging of the tools journey, it is stored in a configuration file or in a database on a server. After using method from *ToolProvider* class which requests the tool configuration from the server using above mentioned service and http package [21], an object of type *ToolConfig* is created by deserializing the JSON response that can look like the one shown in listing 1. This object is stored in provider in a variable with special data type *Either<Failure, List<ToolConfig>>* from dartz package to allow error handling [76]. A tool stored in the provider is then accessed with the use of *fold* function which allows for handling the error or accessing the value. In the event of success, it is passed to builder functions, the first one beging *getToolWidgetList* which has the option to set state after the tools are built to update the interface or just build the tools. The return type of this function is *Map<int, List<ExpansionPanel>>*, the map type was chosen because the tool widgets need to be built for the setting, sort and filter section separately. The widgets themselves are built by *buildTools* which is a second builder function which outputs the map with lists of *ExpansionPanel*s, these have header, which contains the name of the tool

and body, in which the actual *ToolWidget* is placed. This widget gets lambda function with which can it update the *ToolConfig* object based on which it was built. And also the list of *Parameter* object from the tool configuration, these contain a type value used for determining which type of widget is used to represent the value. There are currently four of these widgets, slider, text field, drop down and check box. How the list of tools built by this process looks like can be seen in the figure A.9 and how are they used in therm of the layout shows the figure A.7 for the previews screen and figure A.8 shows the individual tool opened in a tab from where it can be modified and submitted.

Chapter 7

User testing and evaluation

Testing the final application is an important step in the development process. Having specific numbers and feedback from users, be it in a simulated setting, gives perspective that could not be gained otherwise, helps steer future development of mistakes of the past, and brings up not only ideas for the future, but also points to things done right. To obtain accurate results, a clear methodology must be established, and, similarly to the development process, fitting tools must be chosen to carry out the testing. User testing was chosen specifically because of the proof-of-concept nature of the application because unit or integration tests are not important for the success of the application as user testing is.

7.1 Testing scenario

This section is going to describe the implementation of testing scenario based on information described in the usability evaluation section in a way that covers there mentioned metrics important for further evaluation. The reason for using the test scenario instead of the checklist is to allow users to explore freely and express an opinion or at least get an expression that can be conveyed by the questionnaire and follow-up questions[50].

7.1.1 Creating testing scenario

In the end, there are two scenarios describing two slightly different sets of tasks within the application. The choice to add another scenario was based on the way the map of screens I put together after finishing the initial skeleton of the application looked like.

This map in figure 7.1 shows the paths the user can take to reach the individual goal screens, which is presumed to be the goal with this application, to inspect the results produced by the selected tools. There are two general paths one can take to reach the results. The first is longer, which is caused by the analysis setup screen having multiple steps within itself spreading the analysis parameters to multiple logical blocks, this path can be one step shorter or longer depending on whether the user wants to create a new case or use existing one. For the first script, the longest path was chosen, as it is most likely the way users will interact with the application. The second path is much shorter; its purpose is to get the user to the results as fast as possible, and, therefore, is a good second candidate for testing.

The created scenarios purposely do not tell users which screens to go to complete the goals set in order to simulate a more natural way of interaction. Users are given a hypothetical scenario, which they should try to finish according to their own abilities and

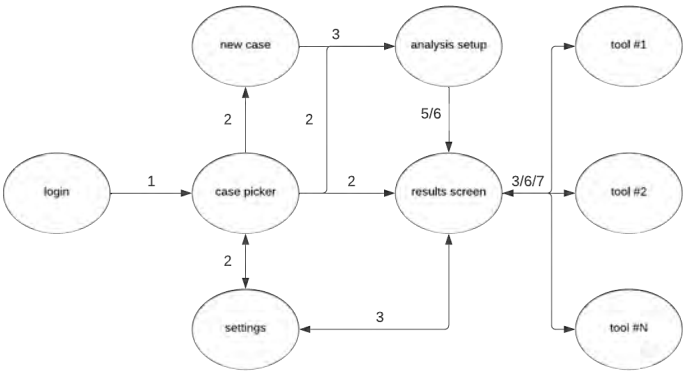


Figure 7.1: Map of screens with number of clicks needed to get to them

knowledge. With the second scenario, users should already be at least partially familiar with the application simulating reoccurring interaction which should mimic the usage of the second path in real world. Both of these scenarios were used in a moderated form of user testing.

7.1.2 Testing scenarios for the FACIS project

Based on the information in the above text, the following scenarios were created.

Scenario 1

To complete this scenario, you must log into the FACIS application with the username and password listed below. Create a new case and one new user. Then analyse disk test_disk1 with the help of all available tools.

Log in credentials:

1. *Username:* test-user-id
2. *Password:* TestPassword42

Try to complete the whole scenario from start to finish, you have as much time as you want. If something is unclear before starting the scenario, test moderator will answer all your questions.

Scenario 2

To complete this scenario, you must log into the FACIS application with the username and password listed below. Open a pre-existing case. The goal of this scenario is to find profiles that fit the further listed description.

Log in credentials:

1. *Username:* test-user-id
2. *Password:* TestPassword42

Find the profiles described in the following:

1. Profile names of the oldest person whose age estimate has certainty higher than 90%
2. Profiles name of the youngest person whose age estimate has certainty higher than 90%
3. Two profile names whose age estimate precision is between 0% and 20%

Try to complete the whole scenario from start to finish, you have as much time as you want. If something is unclear before starting the scenario, test moderator will answer all your questions.

7.2 Questionnaire

The questionnaire was not used in a traditional fashion to ask the user about the experience, suggestions and failures that they encountered and accumulated during the testing, but instead to ask the moderator to take note of the metrics mentioned in figure 4.4.

7.2.1 Creating questionnaire

In the case of moderated testing, the questionnaire can be used by the moderator, not by the user, with whom the testing is concluded. This simplifies many things because the meaning of the questions is less likely to be wrongly interpreted or not completed in a meaningful way.

There are, however, a couple of questions that need to be asked by the user directly; these were placed in the post-testing thoughts section and users can express their feelings and criticisms in it.

The rest of the questionnaire sections were created in a way that benchmarks each users error rate, speed, and number of clicks it took them to finish the intended goal of the page for every screen they need to go through.

User feedback was collected in three ways; the first was by selecting one of three or four choices according to which part of the scenario was the hardest. The second one was for them to rate the clarity of the interface; the word clarity was chosen over wording like user-friendly or other terms used by user interface designers because it is very abstract or completely foreign to most of the users. This rating scale ranged from 1 meaning that everything was clear to 5 meaning that the user was completely lost. The last way the feedback was gathered was by open question, where the user could freely express what improvements they would like to see.

7.2.2 Expected outputs

The outputs expected from the questionnaire are both user feedback directly expressed during the testing and numerical values of metrics mentioned in section 4.4. For each of these metrics, a one or two questions were created for each section, some require comparison between the difference between the two testing scenarios mentioned above.

Effectiveness and efficiency can be measured by the time spent on individual screens as well as the whole test which could then be compared with other users. Another data point that offers more information in this case is the number of clicks required to navigate the individual screens.

Engagement is in this case not a very important metric, as users will essentially need to use the application. It is, however, interesting and is an indicator of a well-implemented safe exploration pattern put in use[93]. Engagement is a tricky metric to create data for, but in this case, a higher number of clicks per page with time that is not significantly different from the average could be an indicator of higher user engagement.

Error tolerance and ease of learning can be measured by tracking the number of errors and their effect on the resulting time and success rate of users. Whether users find the application easy to learn and understand can be evaluated based on the difference between performance of the users reached in the long and short testing scenario as in the short one, the users should already be somewhat familiar with the layout.

7.2.3 Questionnaire for the FACIS project

There ended up being two questionnaires created, one for each scenario. For this, Google Forms was used, as it offers great features, such as the ability to host the form online and collect data directly to file stored in the cloud on Google disk. This made things much easier, as testing was done on multiple devices using a phone, laptop, or desktop to complete the answers. The questionnaire created for this project have one section for each page the use is going to visit and then final evaluation section in which the user can express their thoughts about the testing. Screenshots of both of these forms for scenario 1 from figure C.1 to C.3 and for scenario 2 from C.4 to C.6 are placed in section C.

7.3 Result evaluation

The data obtained from the tests were collected during 10 sessions with ten different people whose skills in areas like programming, photo editing, and forensic work ranged from none to professional in some of these areas. This should be a large enough testing group to discover most of the issues [67]. However, none of them had the same set of skills as the forensic experts who might operate the application in the future. The sessions were moderated and the data was entered by the moderator, not by the users themselves, in order to divert their focus as least as possible.

7.3.1 Gathered data analysis

The data was filled out on a laptop placed near the desktop on which the mock back-end and front-end ran with the moderator operating it. The data was automatically sent to two tables stored on Google Disk, from which they were downloaded into Excel to analyse them and create the following graphs. The data tables can be inspected in section B.

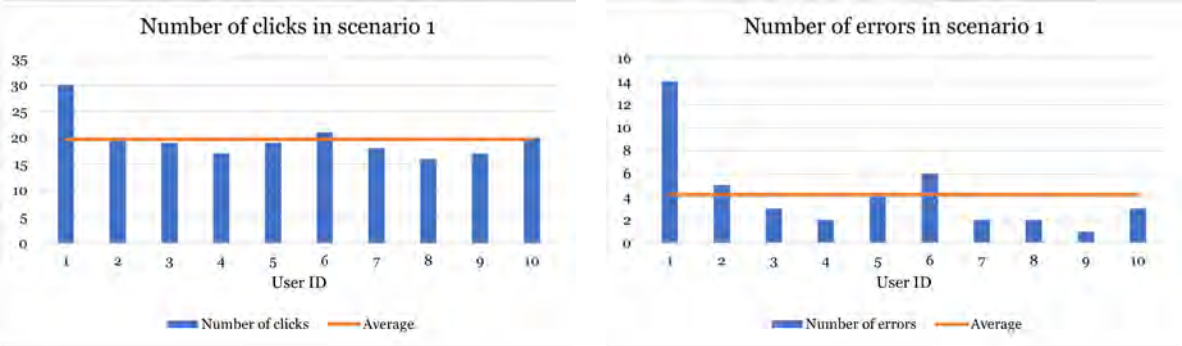


Figure 7.2: Number of clicks and errors in scenario 1

Graphs in figure 7.2 show the number of times the user clicked within the application. The number of errors then shows how many times users clicked on something that directly hindered their progress in the scenario, for example, one user clicked the back button while creating a new case resulting in cancelling the case creation. The only user who really struggled is user 1 who essentially panicked during testing, almost giving up, after a while they managed to finish the scenario. The screens where they made the most mistakes were the new case screen and the analysis setup screen. The former was particularly confusing due to some design choices requested by the police mentioned in section 3.4 that reduced user friendliness, such as not showing available users.

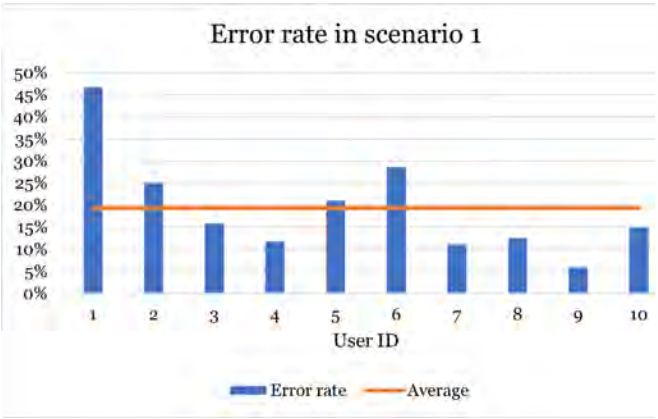
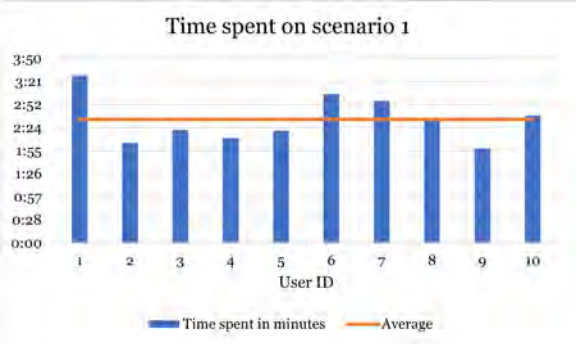


Figure 7.3: Error rate in scenario 1

The graph in figure 7.3 shows the relationship between the number of clicks and the errors during the test. There is no clearly given percentage that would be considered an acceptable error rate because it is highly dependent on the specification and requirements. However, there are numbers on the task completion rate with anything above 78% considered above average[77]. In this case, the completion rate of the task despite the high error rate is 100%.



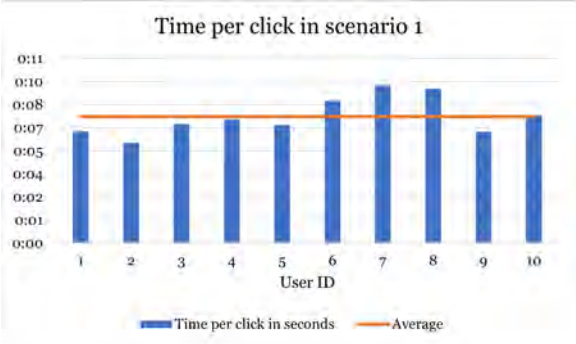
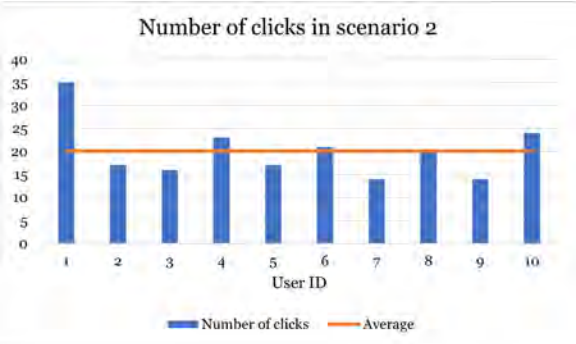


Figure 7.4: Time spent and time per click in scenario 1

The time in which the users completed the scenario and the time between clicks for each user are visualised by the graphs in figure 7.4. This is the time it took them to log in, create a new case with a new user, and configure the analysis.



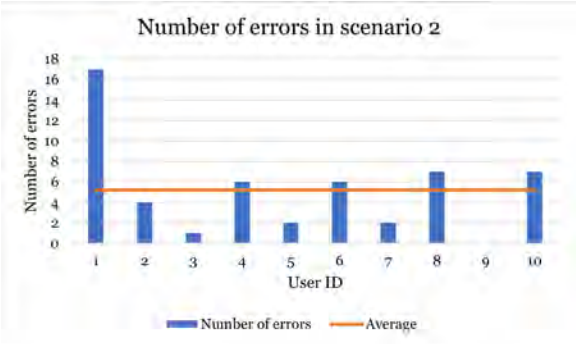


Figure 7.5: Number of clicks and errors in scenario 2

The graphs in figure 7.5 represent the same metrics as in the first scenario. There we can see that the second scenario required on average more clicks to be completed, 19.7 in the first scenario, and 20.1 with on average more errors, 4.2 in the first one, and 5.2 in the second one. This means that while the number of clicks increased by 2%, the number of errors increased by almost 24% compared to the data shown in figure 7.5. With the second scenarios focussing on operating the tools, it is clear that focussing on great flexibility might have led to a steeper learning curve.

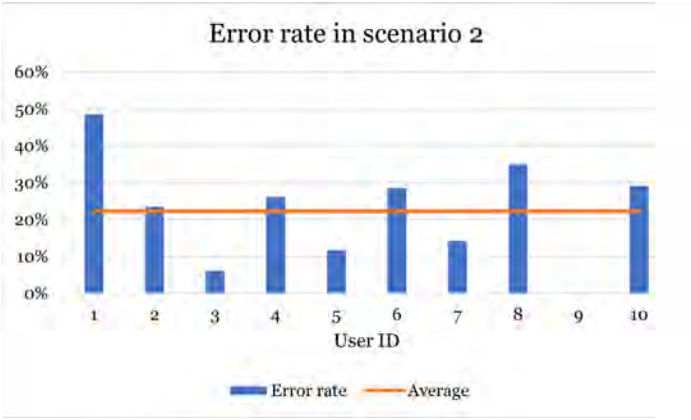


Figure 7.6: Error rate in scenario 2

The graph shown in figure 7.6 further paints the picture of the increased difficulty compared to the first scenario, which mainly focused on the creation of cases and the setup of analysis. The average error rate increased slightly in the second scenario due to the tool controls being a new concept the users discovered as opposed to the widgets in previous scenario which were closer to traditional user interface elements they interact with on daily basis. However, in this scenario, the difference between the user error rate is much more pronounced, some of the users have around or under the 10% error rate and others have almost or over the 30% error rate.

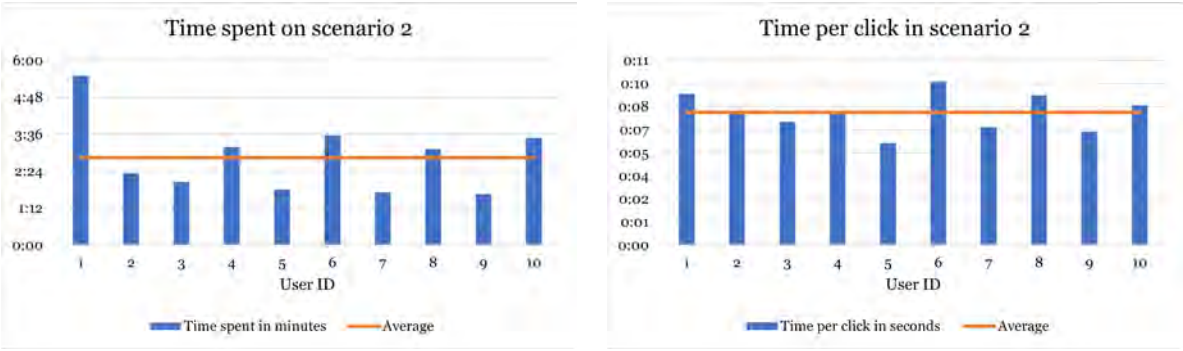


Figure 7.7: Time spent and time per click in scenario 2

As well as the number of clicks, the time to finish the second scenario is longer than in the case of the first one. This is a result of not only more clicks needed, but also a larger delay between the clicks; in the first scenario, the time per click averaged around 7 seconds and 8 seconds in the second one. Despite users taking more time before interacting, the error rate was higher; this was expected, as interacting with the tools is more complex than creating the case and preparing the analysis. This can be observed in the graphs in figure 7.7.

7.3.2 User feedback evaluation

The feedback evaluated in this section was collected in the three ways mentioned above. The first of the feedback that can be evaluated in the clarity rating. The average clarity

score in both is 2.4 which is not excellent but is still above average. With the skill of the users who participated in the user testing, this score is expected.

The result of the question asking which part of the scenario was the hardest had a clear answer for both scenarios. In the case of the first, the users found that creating the case was the most difficult part. For the second scenario, most users considered that navigating between the tool screens is the biggest obstacle.

The final open question feedback was perhaps most insightful of all three. The user complaints were mainly focused on lack of available documentation or info buttons explaining further the action they were attempting to make. On the other hand a praise was given to implementation of basic shortcuts and navigation patterns in the application. The suggestions ranged from changing the way new case button is display to notes about the widget types that could be added to the existing four control types.

7.3.3 Experts feedback

The user who did not attend the testing scenarios, but did interact with the application on a separate meeting is the forensic expert. His feedback was very important mainly because of his knowledge of many analytical applications and because it were his notes this application is mainly base on.

The thing he pointed out was missing was the file tree widget which as he stated many his colleagues like, but also understood that it isn't implemented due to the time constrains. What he did like was the main screen containing the tools with states and tab based approach to separating the tools. Another feature he commented on was the amount of flexibility the application provides with the option to integrate new tools without any coding required.

7.4 Evaluation summary

After gathering all the feedback and metrics and evaluating them, a few conclusions can be drawn about this work and the resulting application. The strength of the application in terms of large flexibility resulted in a poorer user experience as a side effect; this was expected and only substantiated by the results of the testing. What was not predicted was the feedback reflecting on case creation, this however, can be easily resolved by a couple small changes including the once the users suggested. The learning curve can seem a bit steep in the beginning, but after using the application a few times, it is possible to predict the behaviour of widgets and features thanks to the repeating widgets and principles throughout the application.

Chapter 8

Conclusion

The goal of this thesis was to create an application which is going to integrate a wide spectrum of tools in a way that allows flexibility in terms of output and widgets representing the integrated tools, which would then be used by criminal police to speed up the process of analysing image data. I have examined tools and processes used for these tasks at present and gathered feedback and recommendations from the workers performing the analysis. This process was carried out as part of the Facis project that aims to resolve issues related to the analysis of large amounts of audiovisual data. Based on these findings and research of the design and creation of user-friendly desktop applications, I have designed and implemented the resulting application. The framework of choice for this work was due to Flutter's unique advantages.

The design of the application was discussed biweekly in project meetings with both team members developing the tools and representative of Czech police. This helped shape the project with the mix of experience from other developers, as well as the input of the member of future userbase. The importance of flexibility was reiterated by both groups at every step, this led to development of unorthodox techniques being used for the representation of the tools.

The created application allows for demonstration of the final system using a mock server or testing of already implemented tools deployed on the development server. The assessment by the police expert of the resulting application was largely positive, with features agreed on prior to the implementation of the feedback meeting and a detailed roadmap of features to implement in the future.

The creation of application presented in this thesis marks the fulfilment of goals outlined in this work. These goal being to design and implement a proof of concept application that will combine ability to perform detailed analysis of visual data with tools which can be integrated easily and without the need of any additional updates or code changes. With this core concept further effort can be directed into the visual side of the application by customizing the theme of the application. Up next is also multi language support which would allow for the application to be distributed on the international market.

Bibliography

[1] ADOBE. *Adobe Lightroom* [online]. 2023 [cit. 2023-25-01]. Available at: https://www.adobe.com/cz/products/photoshop-lightroom-classic.html.

[2] ADOBE. *Adobe Photoshop* [online]. 2023 [cit. 2023-25-01]. Available at: https://www.adobe.com/products/photoshop.html.

[3] AFFILIATES, O. and/or its. *What Is an Object?* [online]. 2023 [cit. 2023-03-31]. Available at: https://docs.oracle.com/javase/tutorial/java/concepts/object.html.

[4] ALBERTOBONACINA.COM. *Fluentui_icons* [online]. 2023 [cit. 2023-03-19]. Available at: https://pub.dev/packages/fluentui_icons.

[5] BABICH, N. *How to Develop a Design Language* [online]. 2023 [cit. 2023-03-19]. Available at: https://xd.adobe.com/ideas/principles/web-design/how-to-develop-design-language/.

[6] BADASHIAN, A. S., MAHDAVI, M., POURSHIRMOHAMMADI, A. and NEJAD, M. M. Fundamental Usability Guidelines for User Interface Design. In: *2008 International Conference on Computational Sciences and Its Applications*. 2008, p. 106–113. DOI: 10.1109/ICCSA.2008.45.

[7] BALDWIN, N. *What is a Design Language... really?* [online]. 2023 [cit. 2023-03-19]. Available at: https://medium.com/thinking-design/what-is-a-design-language-really-cd1ef87be793.

[8] BDLUKAA.DEV. *Fluent_ui* [online]. 2023 [cit. 2023-03-19]. Available at: https://pub.dev/packages/fluent_ui.

[9] BEVAN, N., KIRAKOWSKI, J. and MAISSEL, J. *What is Usability?* [online]. 2023 [cit. 2023-03-19]. Available at: https://usabilitynet.org/papers/whatis92.pdf.

[10] BIOID. *BioID* [online]. 2023 [cit. 2023-03-31]. Available at: https://www.bioid.com/.

[11] BOTT, E. and STINSON, C. *Windows 10 inside out*. Microsoft Press, 2019.

[12] COMPANY, T. Q. *Licensing* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.qt.io/licensing/.

[13] COMPANY, T. Q. *Qt Quick Scene Graph* [online]. 2023 [cit. 2023-03-19]. Available at: https://doc.qt.io/qt-6/qtquick-visualcanvas-scenegraph.html.

[14] CONTRIBUTORS. *List of GTK applications* [online]. 2023 [cit. 2023-03-19]. Available at: https://en.wikipedia.org/wiki/List_of_GTK_applications.

[15] CONTRIBUTORS, M. *What is a URL?* [online]. 2023 [cit. 2023-03-31]. Available at: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_URL.

[16] CONTRIBUTORS, O.-G. *OpenAPI Generator* [online]. 2023 [cit. 2023-03-19]. Available at: https://openapi-generator.tech/.

[17] DANIEL, F., YU, J., BENATALLAH, B., CASATI, F., MATERA, M. et al. Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. *IEEE Internet Computing*. 2007, vol. 11, no. 3, p. 59–66. DOI: 10.1109/MIC.2007.74. ISSN 1089-7801. Available at: http://ieeexplore.ieee.org/document/4196176/.

[18] DART.DEV. *Commonly used packages* [online]. 2023 [cit. 2023-03-19]. Available at: https://dart.dev/guides/libraries/useful-libraries.

[19] DART.DEV. *Core libraries* [online]. 2023 [cit. 2023-03-19]. Available at: https://dart.dev/guides/libraries.

[20] DART.DEV. *Dart overview* [online]. 2023 [cit. 2023-03-19]. Available at: https://dart.dev/overview.

[21] DART.DEV. *Http* [online]. 2023 [cit. 2023-03-19]. Available at: https://pub.dev/packages/http.

[22] DART.DEV. *Sound null safety* [online]. 2023 [cit. 2023-03-19]. Available at: https://dart.dev/null-safety.

[23] DART.DEV. *Test* [online]. 2023 [cit. 2023-03-19]. Available at: https://pub.dev/packages/test.

[24] DAVIES, R. *Flutter and Ubuntu so far* [online]. 2023 [cit. 2023-03-19]. Available at: https://ubuntu.com/blog/flutter-and-ubuntu-so-far.

[25] DE GEORGE, A. et al. *Průvodce pro desktop (WPF .NET)* [online]. 2023 [cit. 2023-03-19]. Available at: https://learn.microsoft.com/cs-cz/dotnet/desktop/wpf/overview/?view=netdesktop-7.0.

[26] DOCKER. *Use containers to Build, Share and Run your applications* [online]. 2023 [cit. 2023-03-31]. Available at: https://www.docker.com/resources/what-container/.

[27] DRAGOMIR, B. *React Native: Under the Hood* [online]. 2023 [cit. 2023-03-19]. Available at: https://betterprogramming.pub/react-native-under-the-hood-281df5f548f.

[28] DRAHANSKÝ, M. *FACIS* [online]. 2023 [cit. 2023-25-01]. Available at: https://www.fit.vut.cz/research/project/1539/.

[29] FAKALIEVA, R. *Building Modern & Performant Desktop Apps—Is WinUI 3.0 the Way to Go?* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.telerik.com/blogs/building-modern-performant-desktop-apps-winui-30-the-way-to-go.

[30] FANCHI, C. *Flutter vs. React Native: Which is Better in 2023* [online]. 2023 [cit. 2023-03-19]. Available at: https://backendless.com/flutter-vs-react-native-which-is-better-in-2023/.

[31] FIBOR, M. *4 Best frameworks for cross-platform desktop app development* [online]. 2023 [cit. 2023-03-19]. Available at: https://scythe-studio.com/en/blog/4-best-frameworks-for-cross-platform-desktop-app-development.

[32] FLUENTCI et al. *Fluent UI System Icons* [online]. 2023 [cit. 2023-03-19]. Available at: https://github.com/microsoft/fluentui-system-icons.

[33] FLUTTER. *Fetch data from the internet* [online]. 2023 [cit. 2023-03-19]. Available at: https://docs.flutter.dev/cookbook/networking/fetch-data.

[34] FLUTTER. *Null safety in Dart - Introduction* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.youtube.com/watch?v=iYhOU9AuaFs.

[35] FLUTTER. *Simple app state management* [online]. 2023 [cit. 2023-03-19]. Available at: https://docs.flutter.dev/development/data-and-backend/state-mgmt/simple.

[36] FLUTTER.CN. *Dio* [online]. 2023 [cit. 2023-03-19]. Available at: https://pub.dev/packages/schedulers.

[37] FLUTTERCOMMUNITY.DEV. *Get_it* [online]. 2023 [cit. 2023-03-19]. Available at: https://pub.dev/packages/get_it.

[38] FLUTTER.DEV. *Send data to the internet* [online]. 2023 [cit. 2023-03-31]. Available at: https://docs.flutter.dev/cookbook/networking/send-data.

[39] FOUNDATION, I. design. *Usability Elements For Exceptional Experiences* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.interaction-design.org/literature/topics/usability.

[40] FOUNDATION, I. design. *User Interface (UI) Design* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.interaction-design.org/literature/topics/ui-design.

[41] FOUNDATION, T. L. *OPENAPI INITIAVE* [online]. 2023 [cit. 2023-25-01]. Available at: https://www.openapis.org/.

[42] GITHUB. *The top programming languages* [online]. 2023 [cit. 2023-03-19]. Available at: https://octoverse.github.com/2022/top-programming-languages.

[43] GITLAB. *CI/CD pipelines* [online]. 2023 [cit. 2023-30-01]. Available at: https://docs.gitlab.com/ee/ci/pipelines/.

[44] HAIYING, Y. *ASP.NET Code-behind model overview* [online]. 2023 [cit. 2023-03-19]. Available at: https://learn.microsoft.com/en-us/troubleshoot/developer/webapps/aspnet/development/code-behind-model.

[45] HANNA, A. *Gtk or Qt or Flutter for developing Linux app ?* [online]. 2023 [cit. 2023-03-19]. Available at: https://valueinbrief.com/posts/gtk-qt-flutter-linux/.

[46] HANNAH, J. *What Is a User Interface, and What Are the Elements That Comprise One?* [online]. 2023 [cit. 2023-03-19]. Available at: https://careerfoundry.com/en/blog/ui-design/what-is-a-user-interface/.

[47] Hassenzahl, M. and Tractinsky, N. User experience - a research agenda. *Behaviour & Information Technology*. Taylor & Francis. 2006, vol. 25, no. 2, p. 91–97. DOI: 10.1080/01449290500330331. Available at: https://doi.org/10.1080/01449290500330331.

[48] Hiba, S. and Keller, Y. *Hierarchical Attention-based Age Estimation and Bias Estimation* [online]. 2023 [cit. 2023-03-31]. Available at: https://arxiv.org/pdf/2103.09882v1.pdf.

[49] Hindi, D. *How to Perform User Testing For Your App* [online]. 2023 [cit. 2023-03-31]. Available at: https://buildfire.com/how-to-perform-user-testing-for-your-app/.

[50] HubSpot. *User Testing: The Ultimate Guide* [online]. 2023 [cit. 2023-03-31]. Available at: https://blog.hubspot.com/service/user-testing.

[51] Ijaz, A. *Figma: The Next Generation of Figure Drawing* [online]. 2023 [cit. 2023-25-01]. Available at: https://medium.com/codex/figma-the-next-generation-of-figure-drawing-2199785181fa.

[52] Institute, U. D. *What are UX personas and what are they used for?* [online]. 2023 [cit. 2023-03-31]. Available at: https://www.opentext.com/products/encase-forensic#features.

[53] IONOS. *Qt* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.ionos.com/digitalguide/server/know-how/qt/.

[54] Jones, M. et al. *JWT* [online]. 2023 [cit. 2023-03-31]. Available at: https://datatracker.ietf.org/doc/html/rfc7519.

[55] Joshi, A. *Is WPF Dead in 2021? What Are The Alternatives* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.rdglobalinc.com/is-wpf-dead/.

[56] Khomutova, S. *Top 24 famous apps built with Flutter Framework* [online]. 2023 [cit. 2023-03-19]. Available at: https://apexive.com/post/top-apps-built-with-flutter-framework.

[57] Khoroshulia, S. *How React Native App Development Works Under the Hood* [online]. 2023 [cit. 2023-03-19]. Available at: https://mobidev.biz/blog/how-react-native-app-development-works.

[58] Kim, G. K. Early research strategies in context. *CHI '07 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: ACM. 2007-04-28, vol. 2007, no. 1, p. 1777–1782. DOI: 10.1145/1240866.1240899. Available at: https://dl.acm.org/doi/10.1145/1240866.1240899.

[59] Kuznetsov, G. *Visual Design Language: The Building Blocks Of Design* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.smashingmagazine.com/2020/03/visual-design-language-building-blocks/.

[60] Law, E., Roto, V., Vermeeren, A. P., Kort, J. and Hassenzahl, M. Towards a Shared Definition of User Experience. Association for Computing Machinery. 2008. DOI: 10.1145/1358628.1358693. Available at: https://doi.org/10.1145/1358628.1358693.

[61] Maldonado, C. A. and Jlesnick, M. L. Do Common User Interface Design Patterns Improve Navigation? *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. 2002, vol. 46, no. 14, p. 1315–1319. DOI: 10.1177/154193120204601416. ISSN 2169-5067. Available at: http://journals.sagepub.com/doi/10.1177/154193120204601416.

[62] MC, A. *Crowd Safety with AI/ML Object Detection* [online]. 2023 [cit. 2023-03-31]. Available at: https://www.ardentmc.com/resource/crowdtrek/.

[63] Microsoft. *Fluent UI* [online]. 2023 [cit. 2023-03-19]. Available at: https://developer.microsoft.com/en-us/fluentui#/.

[64] Microsoft. *Windows 11* [online]. 2023 [cit. 2023-25-01]. Available at: https://www.microsoft.com/en-us/windows/windows-11.

[65] Mwangi, B. *Dependency injection in Flutter using GetIt and Injectable* [online]. 2023 [cit. 2023-03-19]. Available at: https://blog.logrocket.com/dependency-injection-flutter-using-getit-injectable/.

[66] Nielse, J. *Usability 101: Introduction to Usability* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.nngroup.com/articles/usability-101-introduction-to-usability/.

[67] Nielsen, J. *How Many Test Users in a Usability Study?* [online]. 2023 [cit. 2023-03-31]. Available at: https://www.nngroup.com/articles/how-many-test-users/.

[68] OpenText. *EnCase features overview* [online]. 2023 [cit. 2023-03-31]. Available at: https://www.opentext.com/products/encase-forensic#features.

[69] Overflow, S. *Developer Survey Results 2019* [online]. 2023 [cit. 2023-03-19]. Available at: https://insights.stackoverflow.com/survey/2019.

[70] overflow.net dash. *Provider* [online]. 2023 [cit. 2023-03-19]. Available at: https://pub.dev/packages/provider.

[71] Parecki, A. *OAuth 2.0* [online]. 2023 [cit. 2023-03-31]. Available at: https://oauth.net/2/.

[72] point, T. *Windows 10 Development - UWP* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.tutorialspoint.com/windows10_development/windows10_developmen_uwp.htm.

[73] Pusz, L. *Flutter for enterprise – why it adds up* [online]. 2023 [cit. 2023-03-19]. Available at: https://iteo.com/blog/post/flutter-for-enterprise-why-it-adds-up/.

[74] Radich, Q. *What's a Universal Windows Platform (UWP) app?* [online]. 2023 [cit. 2023-03-19]. Available at: https://learn.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide.

[75] Ramel, D. *'Is WPF Dead?' Some Devs Claim 'Yes' as Microsoft Relegates Issues/PRs to the Community* [online]. 2023 [cit. 2023-03-19]. Available at: https://visualstudiomagazine.com/articles/2022/12/16/wpf-dead.aspx.

[76] Rohman, A. *Easy Way to Create a Good Error-Handling in Flutter With Dartz* [online]. 2023 [cit. 2023-03-19]. Available at: https://betterprogramming.pub/easiest-way-to-create-a-good-error-handling-in-flutter-with-dartz-44084d5341bb.

[77] Sauro, J. *10 Benchmarks for User Experience Metrics* [online]. 2023 [cit. 2023-03-31]. Available at: https://measuringu.com/ux-benchmarks/.

[78] Savov, V. *Microsoft is turning the PC into a walled garden and 'we must fight it,' says Tim Sweeney* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.theverge.com/2016/3/4/11160104/tim-sweeney-microsoft-walled-garden-criticism.

[79] Schroeter, E. *What is Usability? (And How to Do It)* [online]. 2023 [cit. 2023-03-19]. Available at: https://careerfoundry.com/en/blog/ux-design/what-is-usability/.

[80] Scott, L. *The best photo editing software in 2023: retouch, fix and enhance your images* [online]. 2023 [cit. 2023-26-01]. Available at: https://www.digitalcameraworld.com/buying-guides/the-best-photo-editing-software.

[81] Shapel, M. *Qt Framework and QML* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.sam-solutions.com/blog/qt-framework/.

[82] Software, L. *Lucidchart* [online]. 2023 [cit. 2023-25-01]. Available at: https://www.lucidchart.com/pages/.

[83] Software, X. *Frontend vs Backend Development: What Comes First When Building a Web App?* [online]. 2023 [cit. 2023-03-31]. Available at: https://xbsoftware.medium.com/frontend-vs-backend-development-what-comes-first-when-building-a-web-app-2939cca011e2.

[84] Sperber, B. *Dartz* [online]. 2023 [cit. 2023-03-19]. Available at: https://github.com/spebbe/dartz.

[85] Studio, F. *React Native vs Flutter: Which One is Better for 2023?* [online]. 2023 [cit. 2023-03-19]. Available at: https://fireart.studio/blog/flutter-vs-react-native-what-app-developers-should-know-about-cross-platform-mobile-development/.

[86] Studio, L. *Why Flutter application is a great idea for your business in 2023* [online]. 2023 [cit. 2023-03-19]. Available at: https://linkupst.com/blog/flutter-app-for-your-business/.

[87] Sutipitakwong, S. and Jamsri, P. Pros and Cons of Tangible and Digital Wireframes. In: *2020 IEEE Frontiers in Education Conference (FIE)*. 2020, p. 1–5. DOI: 10.1109/FIE44824.2020.9274234.

[88] Sutipitakwong, S. and Jamsri, P. Pros and Cons of Tangible and Digital Wireframes. *2020 IEEE Frontiers in Education Conference (FIE)*. IEEE. 2020-10-21, vol. 2020, no. 1, p. 1–5. DOI: 10.1109/FIE44824.2020.9274234. Available at: https://ieeexplore.ieee.org/document/9274234/.

[89] Sysoev, I. *Nginx* [online]. 2023 [cit. 2023-03-31]. Available at: https://nginx.org/en/.

[90] Team, I. E. *What Is User Interface (UI)?* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.indeed.com/career-advice/career-development/user-interface.

[91] TEAM, T. G. *GIMP* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.gimp.org/.

[92] TEAM, T. G. *GTK timeline* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.gtk.org/about/.

[93] TIDWELL, J., BREWER, C. and VALENCIA, A. *Designing interfaces: patterns for effective interaction design*. Third editionth ed. Beijing: O'Reilly, 2020. ISBN 978-149-2051-961.

[94] USABILITY.GOV. *Usability Evaluation Basics* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.usability.gov/what-and-why/usability-evaluation.html.

[95] UXPIN. *What Actually Constitutes Design Language?* [online]. 2023 [cit. 2023-03-19]. Available at: https://www.uxpin.com/studio/blog/design-language/.

[96] VAILSHERY, L. S. *User population of selected internet browsers worldwide from 2014 to 2021 (in millions)** [online]. 2023 [cit. 2023-26-01]. Available at: https://www.statista.com/statistics/543218/worldwide-internet-users-by-browser/.

[97] WELIE, M. van, VEER, G. C. van der and ELIËNS, A. Patterns as Tools for User Interface Design. *Tools for Working with Guidelines*. London: Springer London. 2001, vol. 2001, no. 1, p. 313–324. DOI: 10.1007/978-1-4471-0279-3_30. Available at: http://link.springer.com/10.1007/978-1-4471-0279-3_30.

Appendix A

Facis application images

Following images represent the state of application as it was finished at the end of this work.

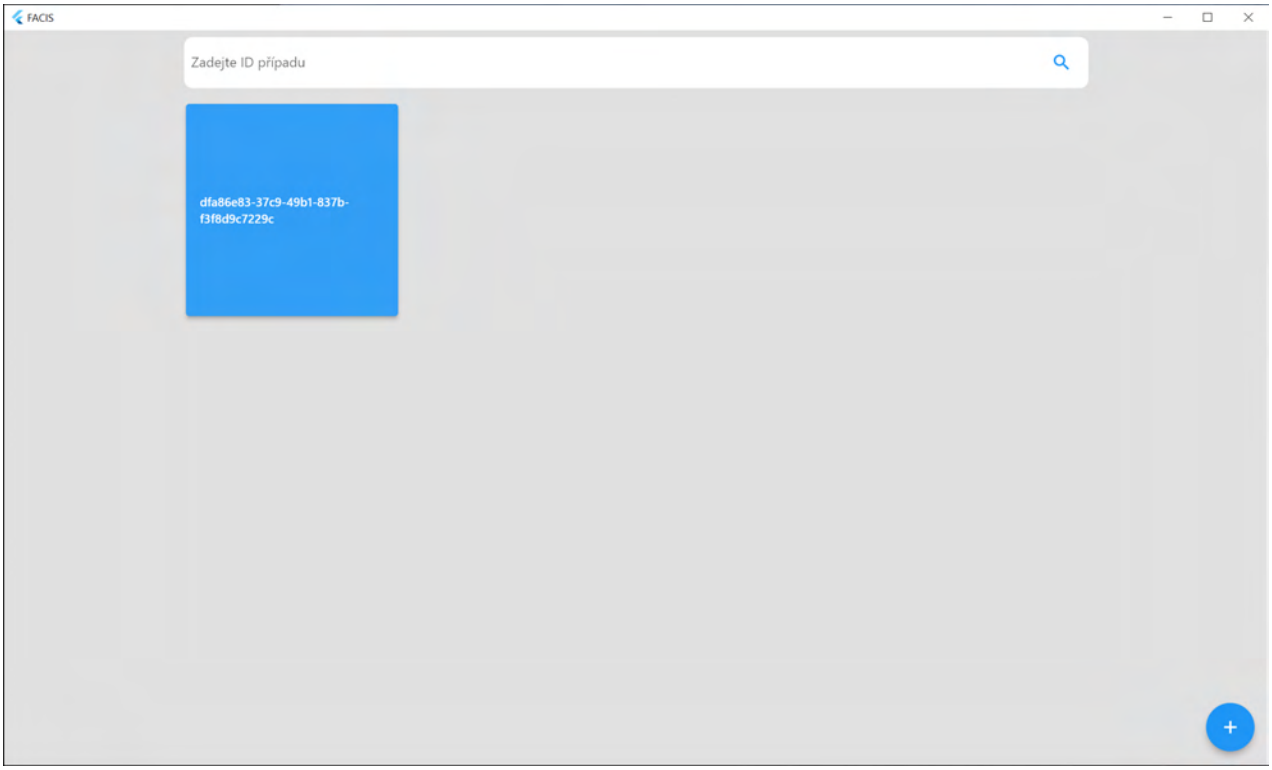


Figure A.1: Case picker screen.

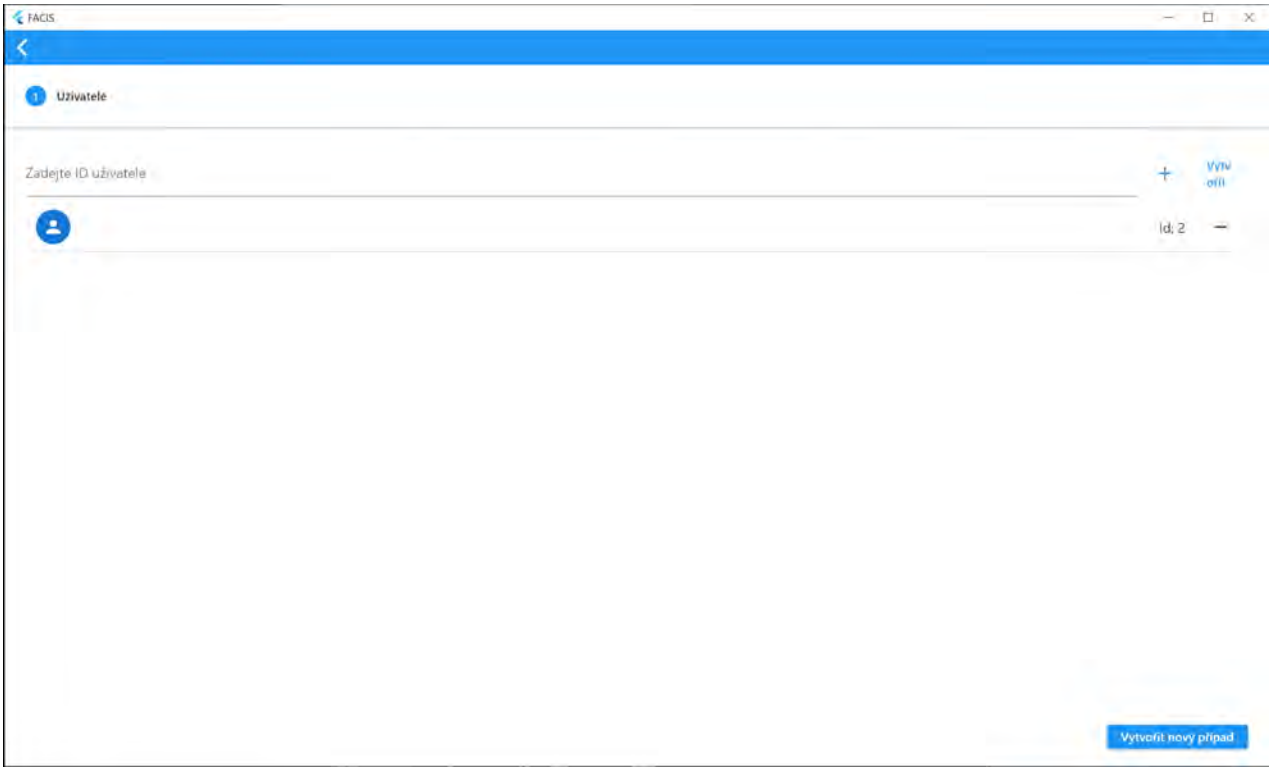


Figure A.2: New case screen part 1.

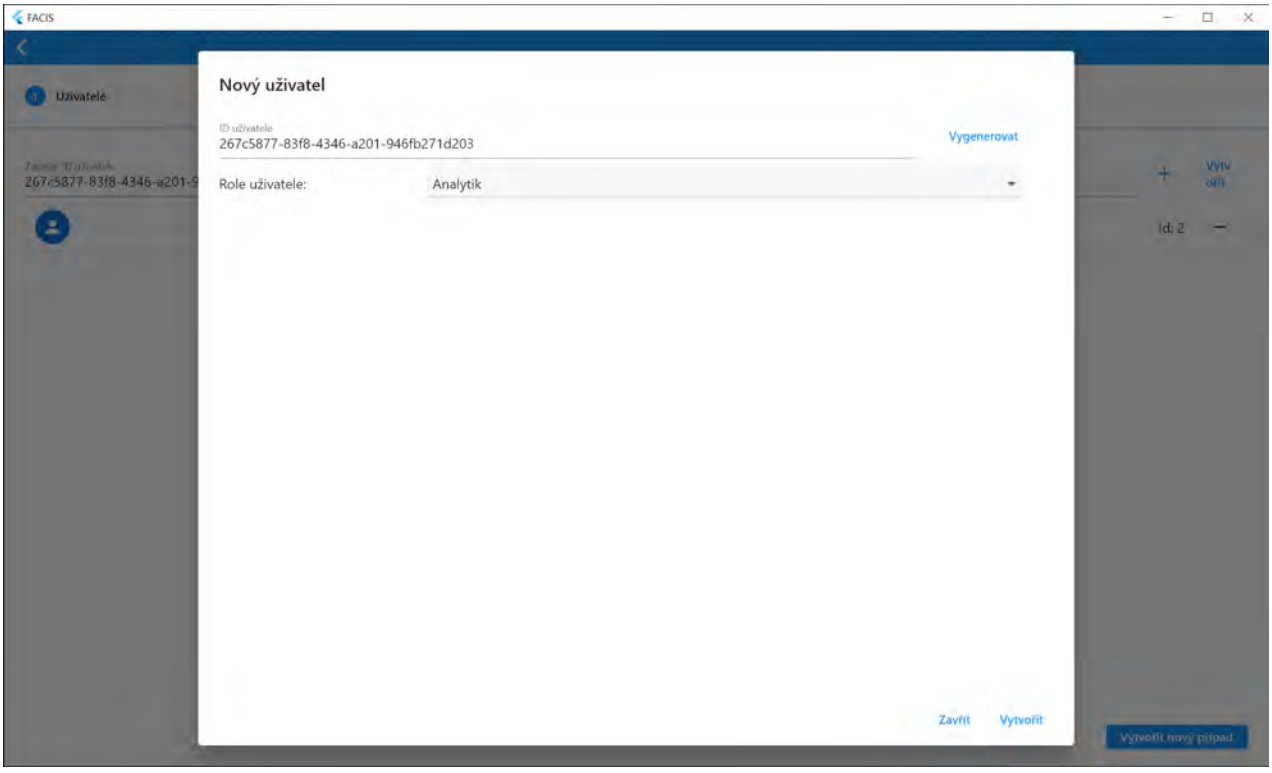


Figure A.3: New case screen part 2.

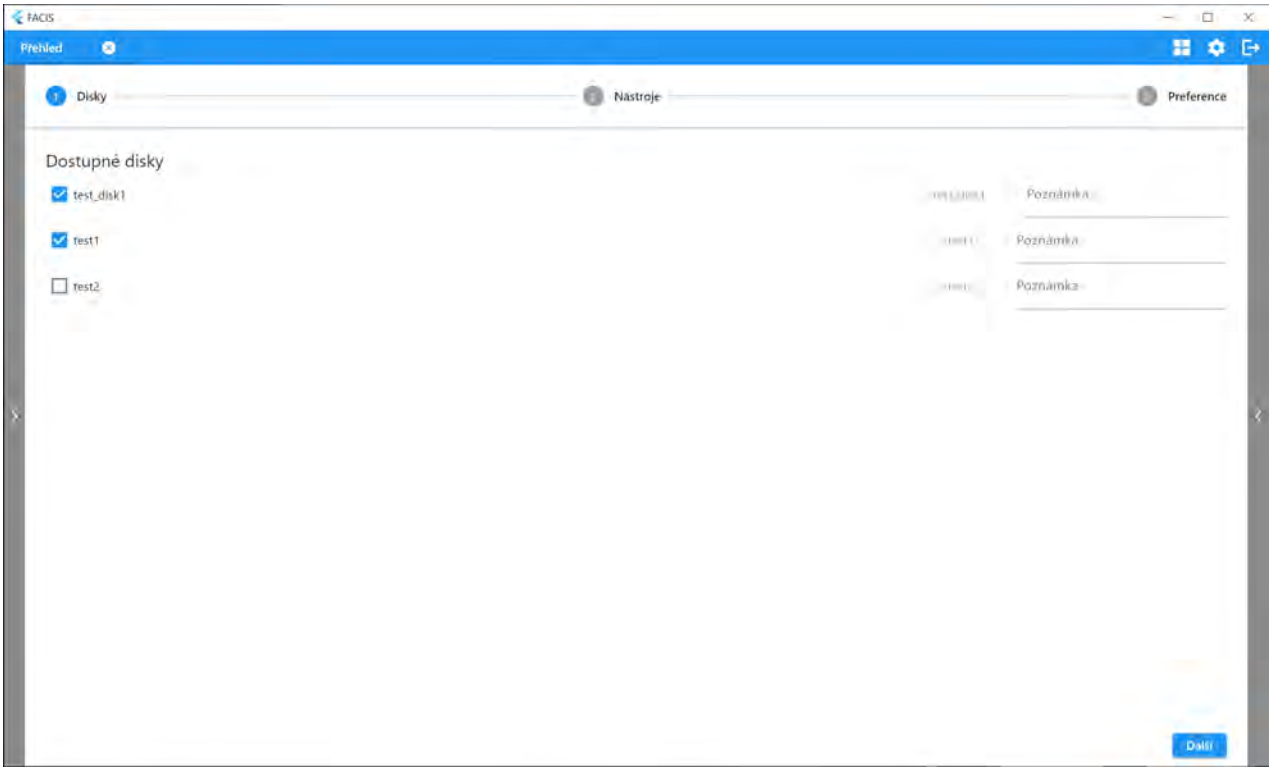


Figure A.4: Analysis setup part 1.

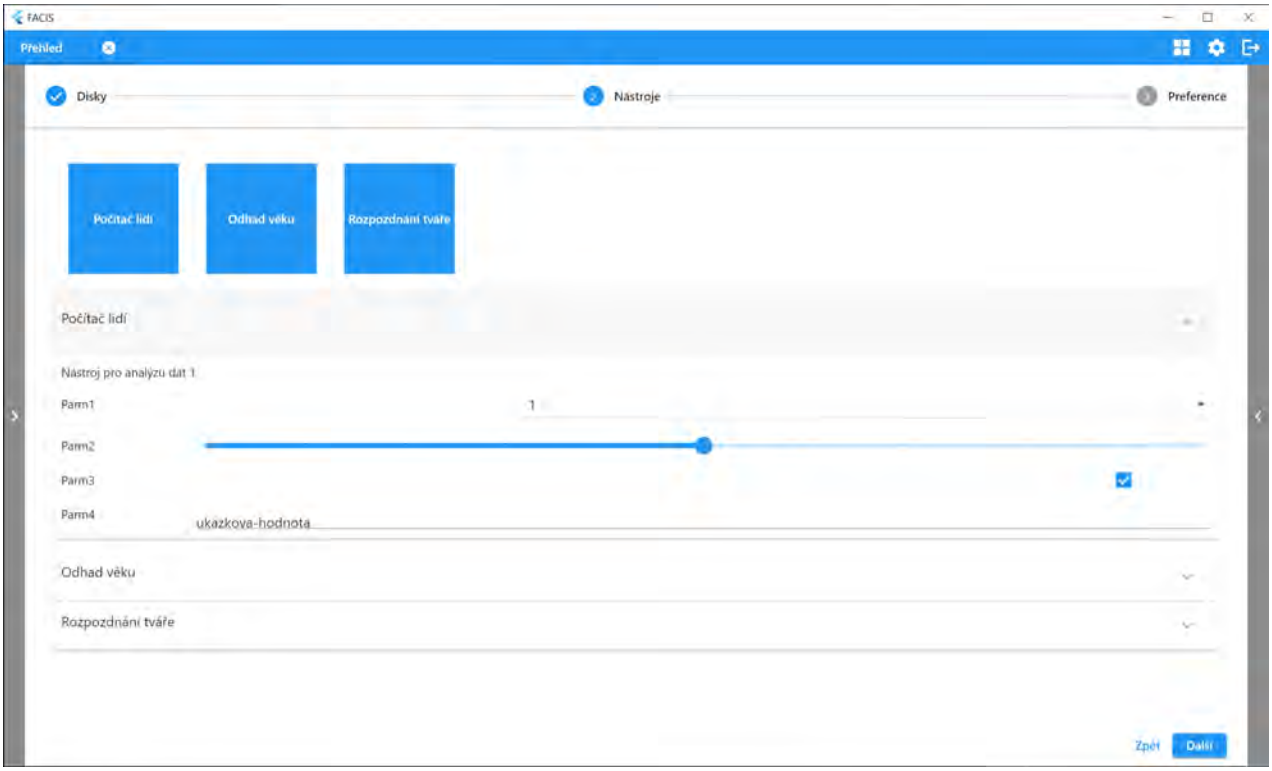


Figure A.5: Analysis setup part 2.

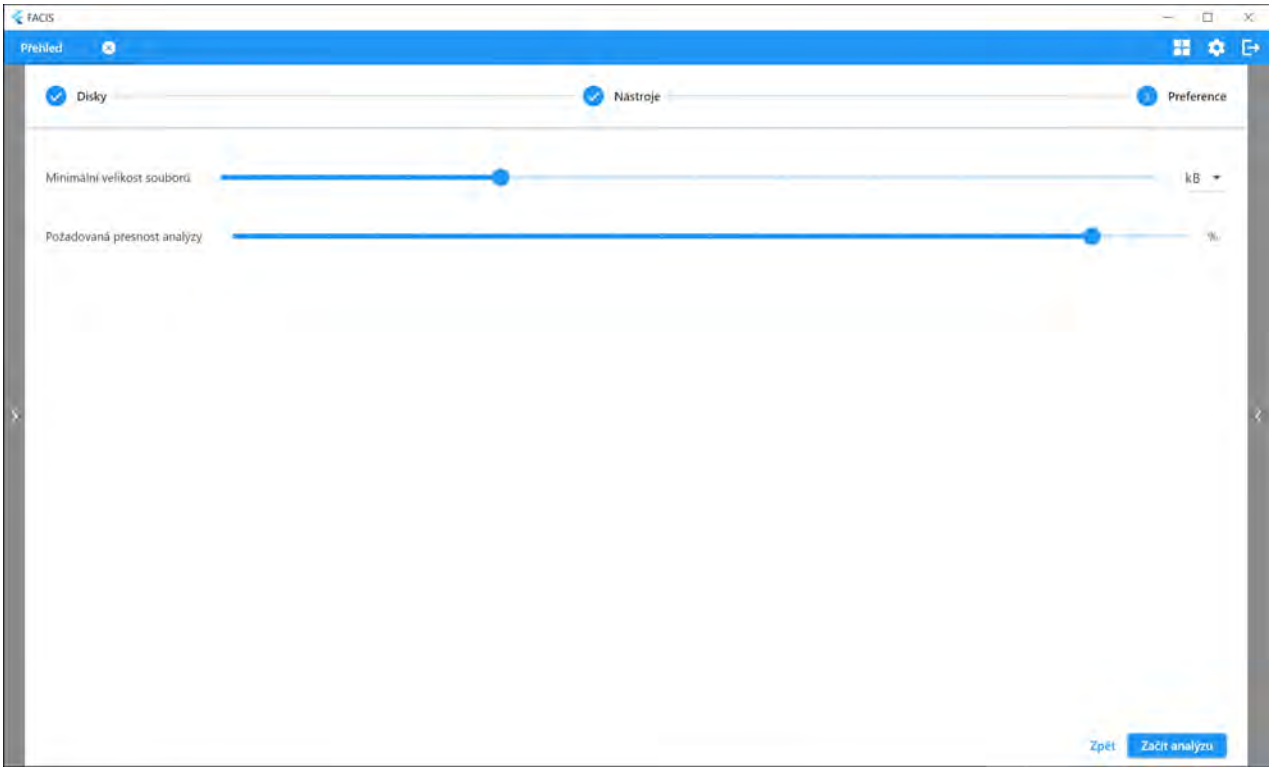


Figure A.6: Analysis setup part 3.

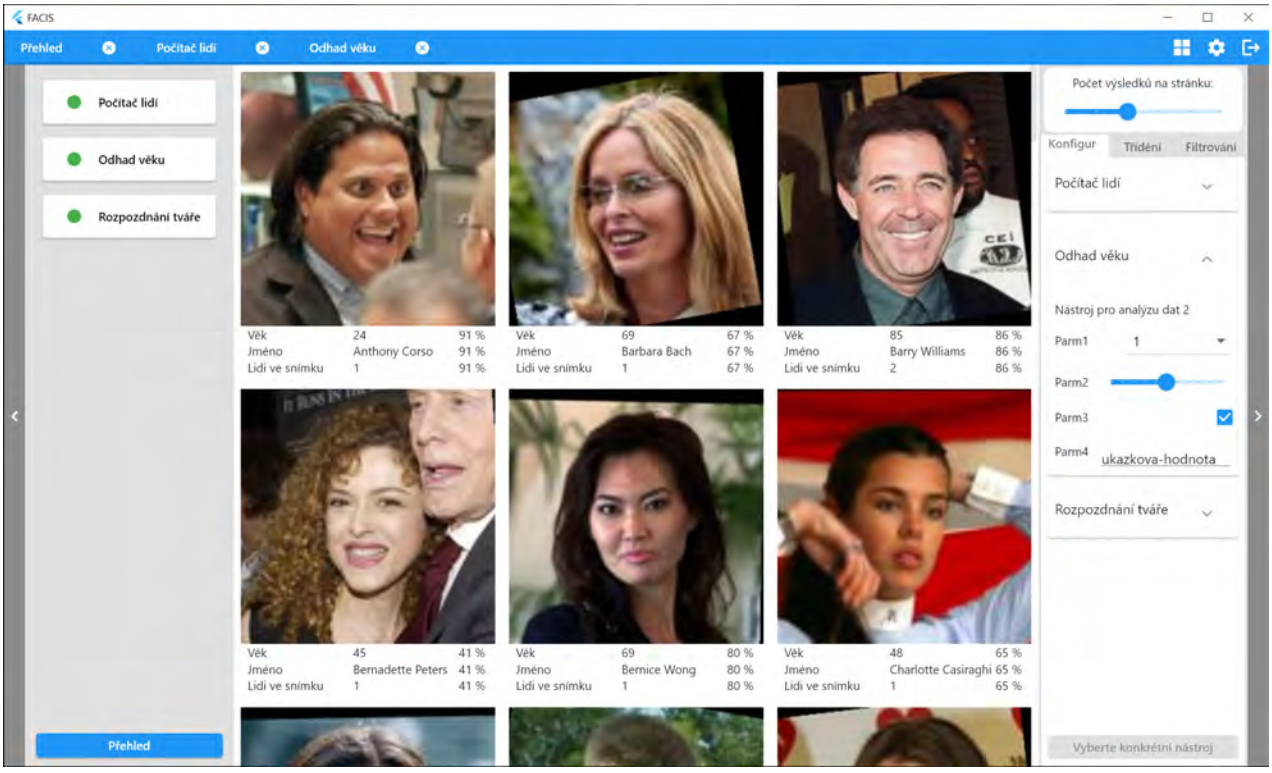


Figure A.7: Previews screen.

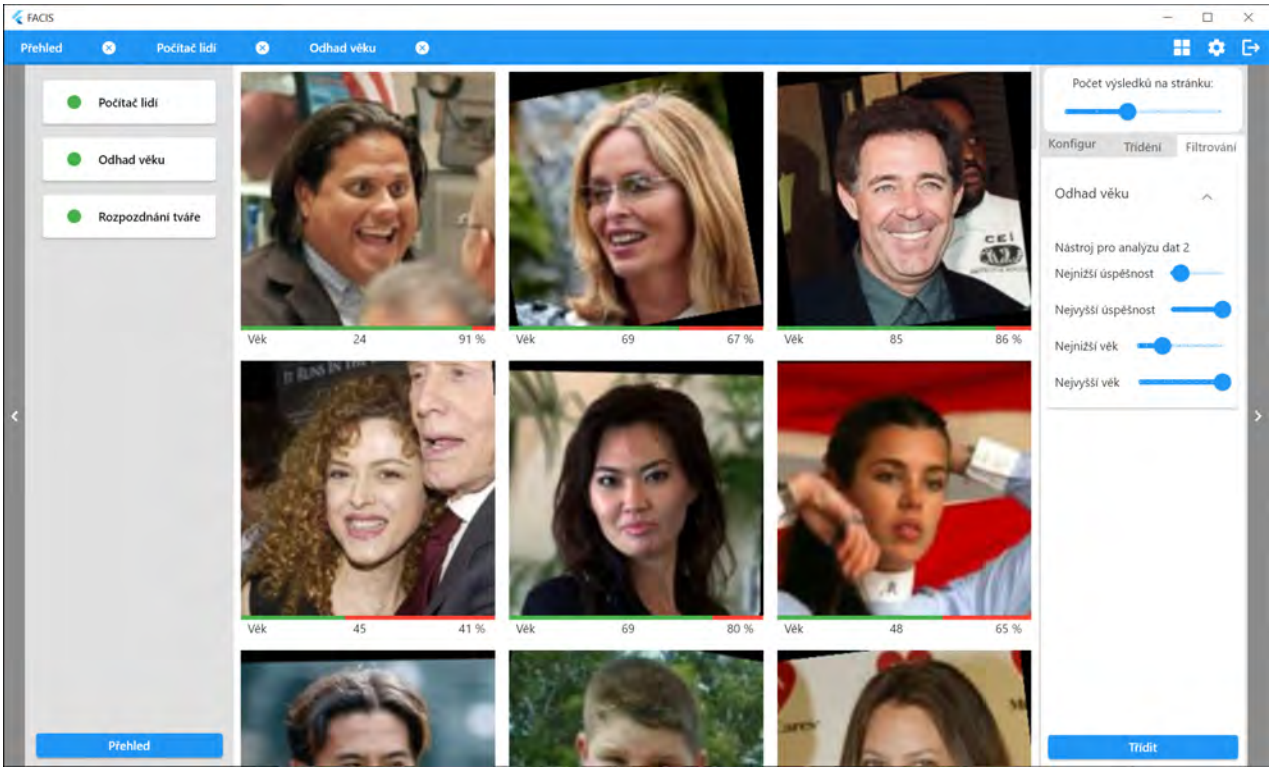


Figure A.8: Tools screen.

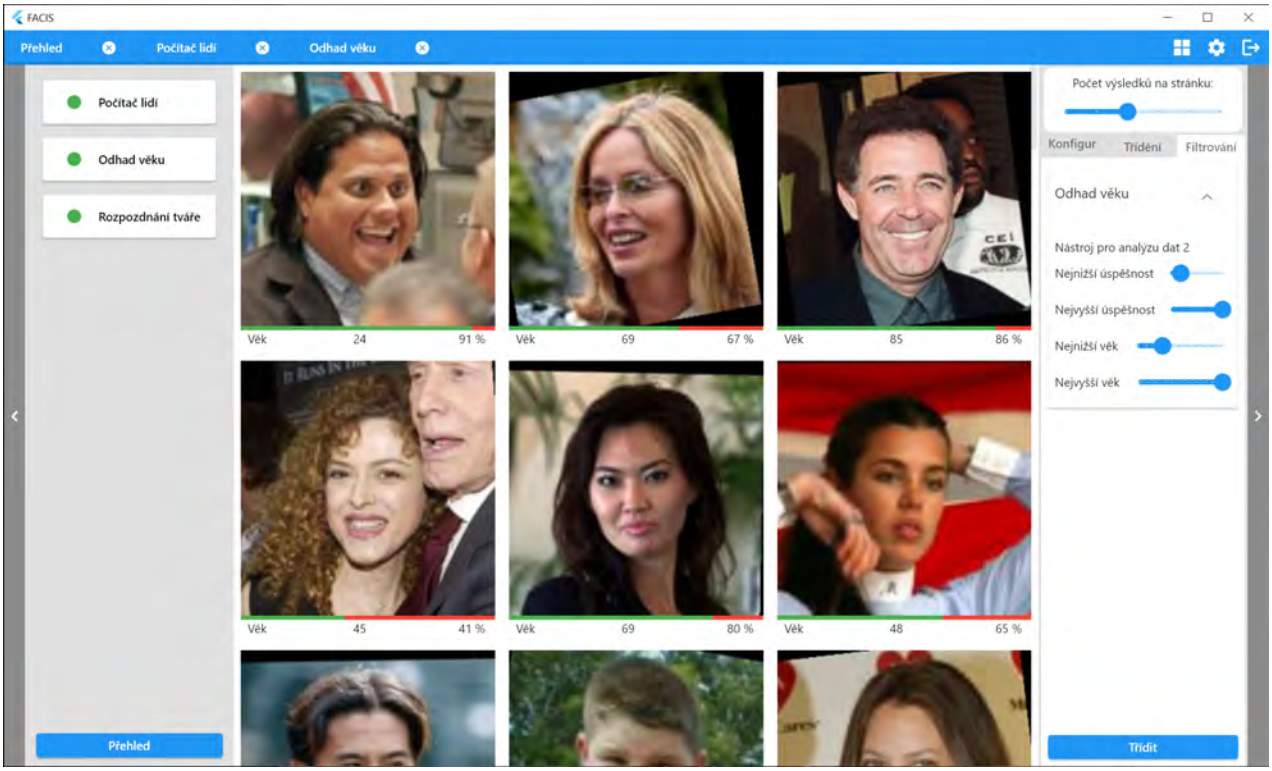


Figure A.9: Control widgets example.

Appendix B

Tables

Following table contain data which was gathered during user testing as an output of testing forms.

		Final assessment
Clarity of interface	What was the hardest part of this scenario?	What would you improve?
4	Understanding the scenario.	Drop downs have different directions of expanding.
2	Using tool inputs.	For filtering entry bar with two dots would be better than two separate entry bars.
2	Navigating to the tool screens.	It is unclear if I need to fill out all the values.
3	Navigating to the tool screens.	I would like to see what tab have I selected.
1	Navigating to the tool screens.	I would like to show the password during the login.
4	Understanding the scenario.	
1	Using tool inputs.	
3	Using tool inputs.	
1	Navigating to the tool screens.	
3	Navigating to the tool screens.	

Figure B.1: User testing table - scenario 1 assessment.

		Final assessment
Clarity rating	What was the hardest part of this scenario	What would you improve?
4	Creating new case.	New case button is too far off and small in the corner.
2	Setting up the analysis.	I wasn't sure what should I tweak.
3	Understanding the scenario.	Add existing user to case would be better if it had buttons more clearlly labeled.
1	Creating new case.	
2	Understanding the scenario.	I would change the way new case screen is opened.
4	Creating new case.	User creation creation screen is not very clear.
2	Setting up the analysis.	View password button would be nice
2	Creating new case.	
1	Setting up the analysis.	g tab to traverse page should work everywhere.
3	Creating new case.	I'd like Case ID format example

Figure B.2: User testing table - scenario 2 assessment.

		Login screen				Case picker screen				New case screen				Analysis setup screen				Previews screen			
user	date	errors	time	clicks	finished	errors	time	clicks	finished	errors	time	clicks	finished	errors	time	clicks	finished	errors	time	clicks	finished
1	28.4.2023	2	0:30	2	Yes	3	0:50	4	Yes	4	2:00	8	No	5	2:50	12	Yes	0	3:29	4	Yes
2	28.4.2023	1	0:22	3	Yes	1	0:29	2	Yes	1	1:05	5	Yes	2	1:57	9	Yes	0	2:05	1	Yes
3	29.4.2023	0	0:18	2	Yes	0	0:26	1	Yes	0	0:58	4	Yes	3	2:04	10	Yes	0	2:21	2	Yes
4	30.4.2023	1	0:21	3	Yes	0	0:28	1	Yes	1	1:02	5	Yes	0	1:57	7	Yes	0	2:11	1	Yes
5	30.4.2023	0	0:20	2	Yes	2	0:27	3	Yes	1	1:12	5	Yes	1	2:13	8	Yes	0	2:20	1	Yes
6	30.4.2023	2	0:29	3	Yes	0	0:40	1	Yes	1	1:27	5	Yes	3	2:41	9	Yes	0	3:06	3	Yes
7	1.5.2023	2	0:40	5	Yes	0	0:48	1	Yes	0	1:32	4	Yes	0	2:43	7	Yes	0	2:57	1	Yes
8	1.52023	0	0:19	2	Yes	1	0:37	2	Yes	1	1:22	4	Yes	0	2:21	7	Yes	0	2:34	1	Yes
9	1.5.2023	0	0:26	3	Yes	1	0:39	2	Yes	0	1:19	4	Yes	0	1:53	7	Yes	0	1:58	1	Yes
10	1.5.2023	0	0:20	2	Yes	0	0:33	1	Yes	2	1:17	6	Yes	1	2:21	8	Yes	0	2:39	3	Yes

Figure B.3: User testing table - scenario 1.

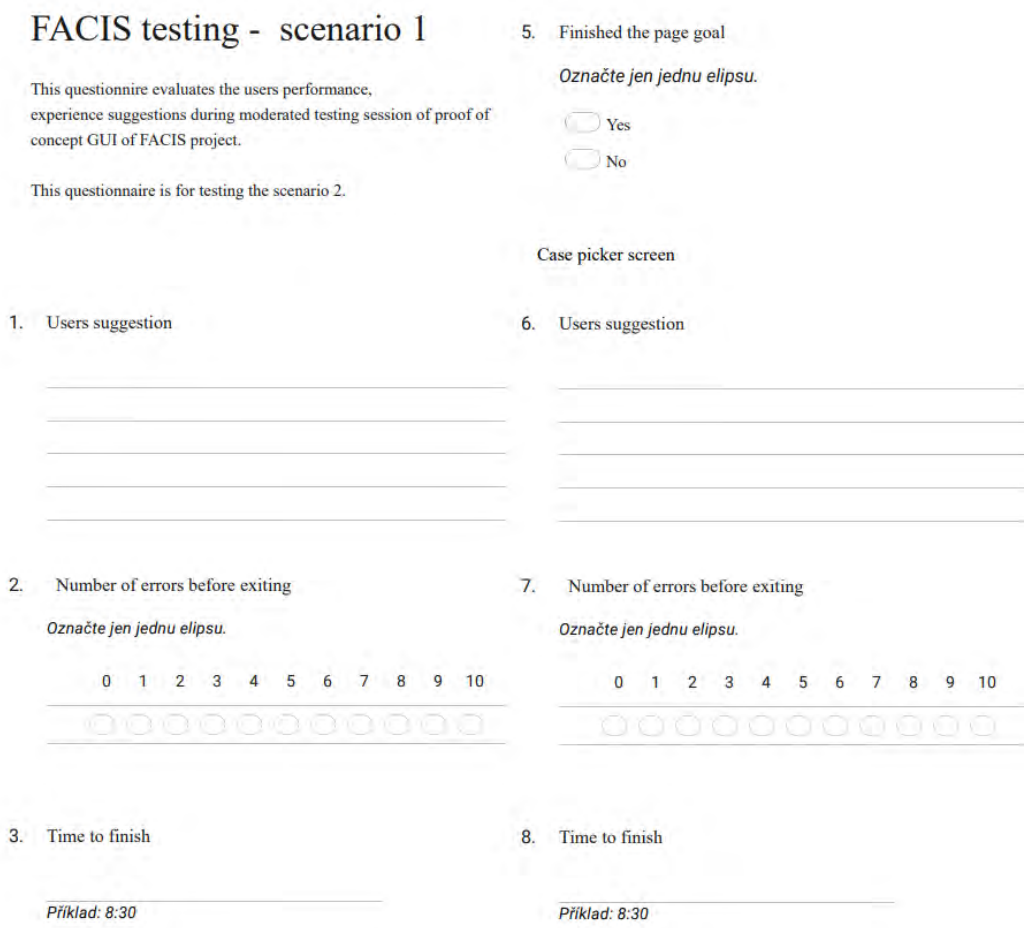
		Login screen				Case picker screen				Previews screen				Tool 1 screen					Tool 2 screen				Tool 3 screen				
user	date	errors	time	clicks	finished	errors	clicks	time	finished	errors	clicks	time	finished	Found name	errors	clicks	time	finished	Found name	errors	clicks	time	finished	Found name	errors	clicks	time	finished
1	28.4.2023	0	0:10	2	Yes	1	2	0:20	Yes	3	8	1:20	Yes	Correct	9	12	4:04	Yes	Correct	3	7	5:08	Yes	Correct	1	4	5:30	Yes
2	28.4.2023	1	0:18	2	Yes	0	1	0:21	Yes	0	2	0:28	Yes	Correct	2	5	1:35	Yes	Correct	0	4	1:58	Yes	Correct	1	3	2:20	Yes
3	29.4.2023	0	0:09	3	Yes	0	1	0:15	Yes	1	3	0:31	Yes	Correct	0	3	1:01	Yes	Correct	0	4	1:37	Yes	Correct	0	2	2:03	Yes
4	30.4.2023	0	0:21	2	Yes	0	1	0:32	Yes	1	5	1:05	Yes	Correct	2	6	2:11	Yes	Correct	2	6	2:46	Yes	Correct	1	3	3:11	Yes
5	30.4.2023	0	0:11	3	Yes	0	1	0:23	Yes	0	2	0:25	Yes	Correct	1	4	1:03	Yes	Correct	0	4	1:32	Yes	Correct	1	3	1:48	Yes
6	30.4.2023	0	0:14	2	Yes	0	1	0:25	Yes	2	4	0:34	Yes	Correct	0	3	1:49	Yes	Correct	2	6	2:51	Yes	Correct	2	5	3:34	Yes
7	1.5.2023	1	0:13	2	Yes	1	2	0:18	Yes	0	1	0:25	Yes	Correct	0	3	0:59	Yes	Correct	0	4	1:25	Yes	Correct	0	2	1:43	Yes
8	1.5.2023	1	0:17	2	Yes	0	1	0:24	Yes	1	3	0:33	Yes	Correct	2	5	1:52	Yes	Correct	2	6	2:39	Yes	Correct	1	3	3:07	Yes
9	1.5.2023	0	0:11	3	Yes	0	1	0:14	Yes	0	1	0:20	Yes	Correct	0	3	0:51	Yes	Correct	0	4	1:19	Yes	Correct	0	2	1:39	Yes
10	1.5.2023	1	0:20	2	Yes	0	1	0:31	Yes	1	6	1:12	Yes	Correct	2	6	2:23	Yes	Correct	1	5	2:55	Yes	Correct	2	4	3:29	Yes

Figure B.4: User testing table - scenario 2.

Appendix C

Testing forms

Following forms were used during user testing to take notes of users performance.



FACIS testing - scenario 1

This questionnire evaluates the users performance, experience suggestions during moderated testing session of proof of concept GUI of FACIS project.

This questionnaire is for testing the scenario 2.

Case picker screen

1. Users suggestion

2. Number of errors before exiting

Označte jen jednu elipsu.

0 1 2 3 4 5 6 7 8 9 10

3. Time to finish

Příklad: 8:30

5. Finished the page goal

Označte jen jednu elipsu.

Yes

No

6. Users suggestion

7. Number of errors before exiting

Označte jen jednu elipsu.

0 1 2 3 4 5 6 7 8 9 10

8. Time to finish

Příklad: 8:30

Figure C.1: Testing questionnaire for scenario 1 - part 1.

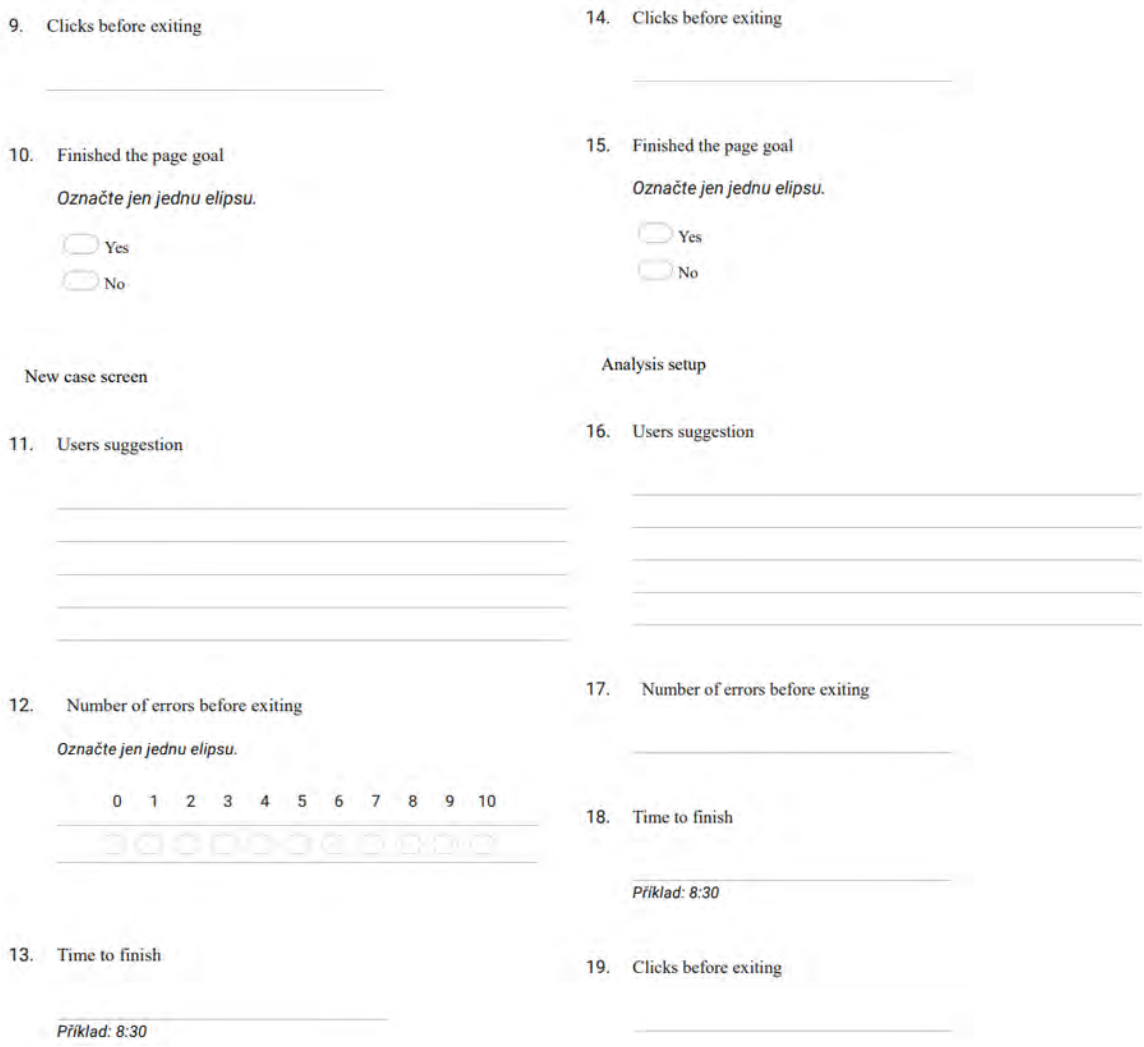


Figure C.2: Testing questionnaire for scenario 1 - part 2.

20. Finished the page goal

Označte jen jednu elipsu.

- Yes
- No

Results screen

21. Users suggestion

22. Number of errors before exiting

23. Time to finish

Příklad: 8:30

24. Clicks before exiting

25. Finished the page goal

Označte jen jednu elipsu.

- Yes
- No

Post testing notes

26. Clarity of interface

Označte jen jednu elipsu.

	1	2	3	4	5	
Ever						I was lost

27. What was the hardest part of this scenario

Označte jen jednu elipsu.

- Navigating to the tool screens.
- Setting up the analysis.
- Understanding the scenario.
- Creating new case.

28. What would you improve?

Figure C.3: Testing questionnaire for scenario 1 - part 3.

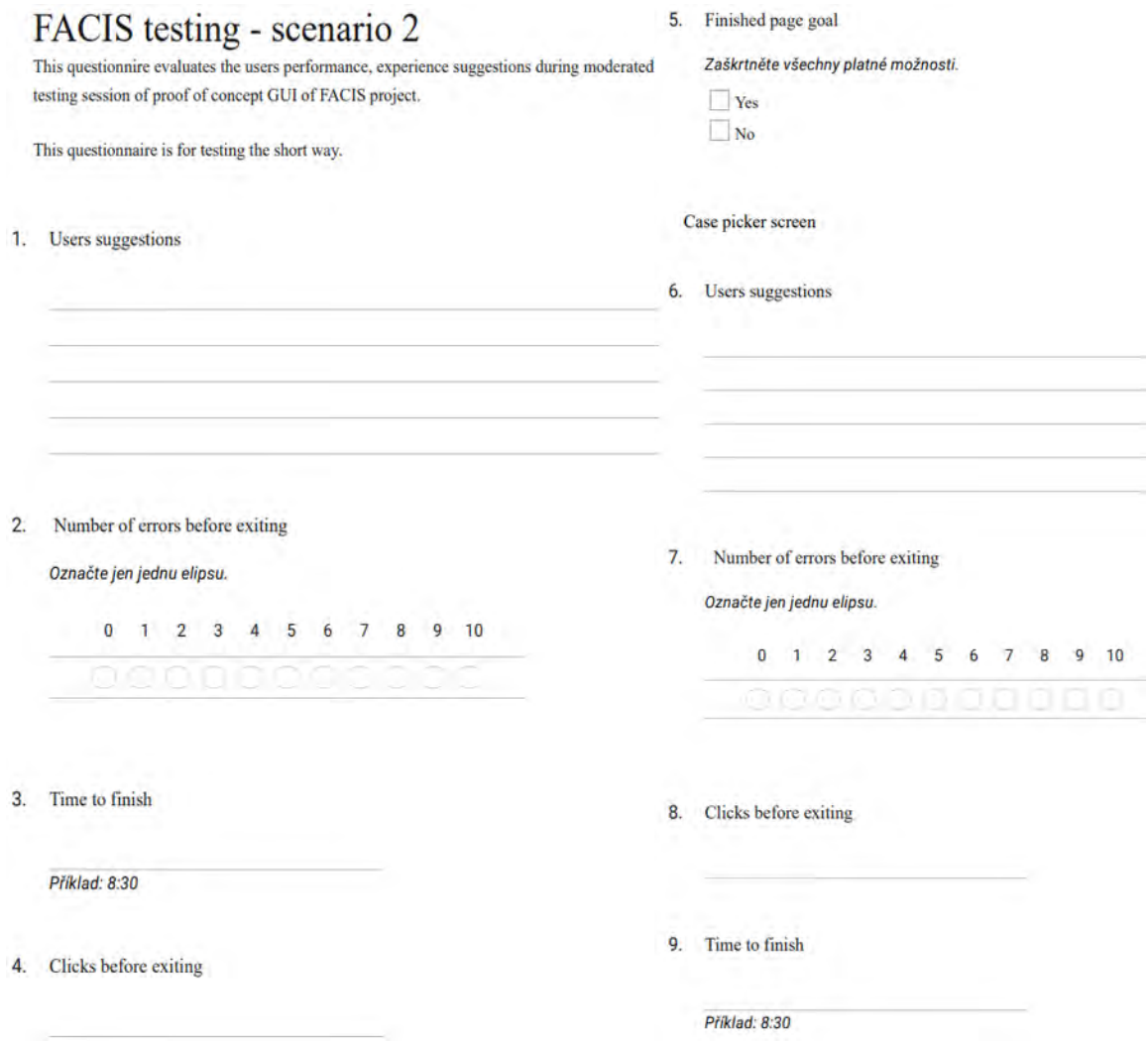


Figure C.4: Testing questionnaire for scenario 1 - part 2.1.

10. Finished page goal

Zaškrtněte všechny platné možnosti.

- [ ] Yes
- [ ] No

Results screen

11. Users suggestions

12. Number of errors before exiting

Označte jen jednu elipsu.

0 1 2 3 4 5 6 7 8 9 10

13. Clicks before exiting

14. Time to finish

Příklad: 8:30

15. Finished page goal

Zaškrtněte všechny platné možnosti.

- [ ] Yes
- [ ] No

Task 1

16. Found name

17. Users suggestions

18. Number of errors before exiting

Označte jen jednu elipsu.

0 1 2 3 4 5 6 7 8 9 10

19. Clicks before exiting

Figure C.5: Testing questionnaire for scenario 1 - part 2.2.

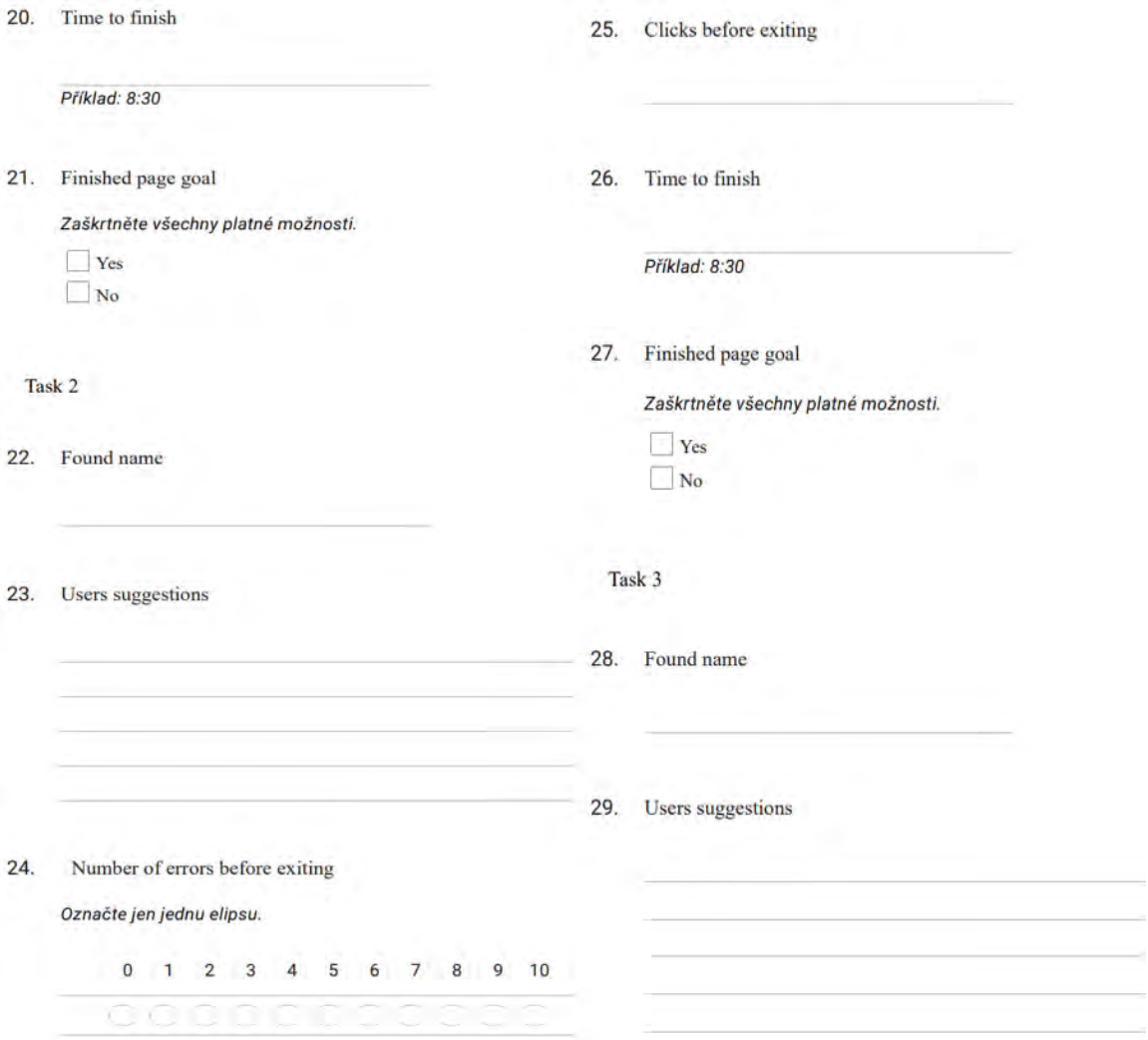


Figure C.6: Testing questionnaire for scenario 1 - part 2.3.