

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики



**РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОГО
ЗБАГАЧЕННЯ ЗОБРАЖЕННЯ МЕТА ІНФОРМАЦІЄЮ ДЛЯ
ПОДАЛЬШОГО ВИКОРИСТАННЯ В ПОШУКОВИХ
СИСТЕМАХ**

**Текстова частина
магістерської роботи
за спеціальністю „Комп’ютерні науки” 121**

Керівник магістерської роботи
д.т.н., проф. М. М. Глибовець

(підпис)
“ ____ ” _____ 2021 р.

Виконав студент Д. М. Нікулін
“ ____ ” _____ 2021 р.

Київ 2021

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
 Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
 Зав.кафедри інформатики, к.ф.-м.н.
 _____ С. С. Гороховський
 (підпис)
 „____” _____ 2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
 на магістерську роботу

студенту 2 р.н. магістерської програми Комп'ютерні науки
Нікуліну Дмитру Миколайовичу

Розробити архітектуру системи збагачення зображення мета інформацією

Зміст текстової частини до магістерської роботи:

Зміст

Анотація

Вступ

1 Аналіз предметної області

2 Аналіз існуючих систем пошуку та обробки зображень та знаходження найбільш критичних проблем.

3 Розробка архітектури системи збагачення зображення мета інформацією

4 Розробка проекту системи

5 Інтеграція проекту системи з системами пошуку зображень

Висновки

Список літератури

Глосарій

Додатки

Дата видачі „____” _____ 2020 р.

Керівник

М.М. Глибовець, доктор технічних наук, професор

 (підпис)

Завдання отримав, Д.М. Нікулін

 (підпис)

Тема: Розробка системи автоматизованого збагачення зображення мета інформацією для подальшого використання в пошукових системах

Календарний план виконання роботи:

| № п/п | Назва етапу дипломного проекту (роботи) | Термін виконання етапу | Примітка |
|-------|---|------------------------|----------|
| 1. | Отримання завдання на дипломну роботу | 01.11.2020 | |
| 2. | Огляд технічної літератури за темою роботи | 15.11.2020 | |
| 3. | Провести аналіз предметної області | 29.11.2020 | |
| 3. | Виконати аналіз існуючих систем пошуку та обробки мета інформації та знайти найбільш критичні проблеми. | 27.12.2020 | |
| 4. | Розробити архітектуру системи збагачення зображення мета інформацією | 17.01.2021 | |
| 5. | Розробити проект системи та зробити інтеграцію з системою пошуку зображень | 14.02.2021 | |
| 6. | Зібрати та проаналізувати результати, зробивши висновки, щодо релевантності та ефективності обраного підходу. | | |
| 7. | Написання пояснювальної записки | 24.04.2021 | |
| 8. | Створення слайдів для доповіді та написання доповіді | 27.04.2021 | |
| 9. | Аналіз отриманих результатів з керівником, написання доповіді та попередній захист магістерської роботи | 30.04.2021 | |
| 10. | Корегування роботи за результатами попереднього захисту | 5.05.2021 | |
| 11. | Остаточне оформлення пояснювальної записки та слайдів | 10.05.2021 | |
| 12. | Захист магістерської роботи (проекту) | 18.06.2021 | |

Студент _____

Керівник _____

“ ”

Content

| | |
|---|-----------|
| Annotation | 6 |
| Introduction | 7 |
| 1 Overview of a subject..... | 8 |
| 2 Overview and analysis of private Multimedia IR systems..... | 12 |
| 2.1 The user needs and requirements for the private Multimedia IR systems | 13 |
| 2.2 Comparison of the private Multimedia IR systems..... | 16 |
| 2.3 Identification and classification of the issues in IR multimedia systems..... | 18 |
| 2.4 Overview of the solutions to address the issues found..... | 21 |
| 3 A proposed solution to enrich the photo metadata | 23 |
| 3.1 User journey to work with the proposed solution | 24 |
| 3.2 The functional and non-functional requirements to the system | 26 |
| 3.3 Context diagram of the System | 29 |
| 3.4 Functional decomposition of the System | 30 |
| 4 Project implementation | 32 |
| 4.1 The coverage of functionality by components | 33 |
| 4.2 The approaches and patterns to address non-functional requirements..... | 34 |
| 4.3 Integration with offline and online Multimedia IR systems | 36 |
| 4.4 The description of code with snippets | 40 |
| 4.4.1 The description of UI code | 40 |
| 4.4.2 The description of FlowController code | 42 |
| 4.4.3 The description of Metadata processor code | 45 |
| Conclusion..... | 48 |
| Glossary..... | 49 |
| Quality Attributes Glossary | 52 |

| | |
|---|-----------|
| References | 54 |
| Appendix A. Appearance of the UI screens | 56 |
| Appendix B. Collaboration/inheritance diagrams..... | 58 |
| Appendix C. PerformChecksForInputDir function implementation | 62 |
| Appendix D. MoveInputDirToOutputDirs function implementation..... | 64 |
| Appendix E. performDTChecks function implementation..... | 65 |

Annotation

This paper describes the IR Multimedia Systems, their subjects, issues, and the possible solution to address those issues. Firstly, it provides the overview and comparison of IR Multimedia Systems such as Google Photo, Mylio, and digiKam that are used as IR Multimedia Systems for private purposes. Further, it identifies the issues in the mentioned systems. Then, it provides the description for the solution in form of software architecture and proposes a definition of project to address those issues. It also describes the integration of an IR Multimedia System with the proposed solution. The paper ends up with a conclusion, a glossary and references list.

Keywords: IR Multimedia Systems, Photo album, Photo album metadata, Multimedia data, Exif.

Introduction

A purpose of investigation. To create the solution and software project to enrich the metadata in the private photo album with minimal human involvement.

The investigation task. To analyze IR Multimedia Systems that are present on the market and available for end user, and to find the issues related to their usage. To check the available products on the market that help the end user to address those issues. To compare the products found with the solution proposed. To check the approaches and the tools that can help in project implementation. To verify if one or two IR Multimedia Systems can easily be integrated with the proposed solution.

The object of investigation. IR Multimedia Systems, solutions to enrich the photo album metadata.

The object of investigation. The components to create the solutions to enrich the photo album metadata, API, the integration in-between the IR Multimedia Systems and solutions to enrich the photo album metadata.

Sources of research. Electronic versions of printed literature, software documentation, reference books to API links, electronic resources, including specialized forums and virtual conferences, source codes for programs and libraries, video instructions.

The scientific novelty of the results is based on finding the issues in IR Multimedia Systems that are not addressed by solutions available on the market (or covering not all problems in the IR Multimedia Systems), creating the solution to cover all important issues and providing integration in-between one or two IR Multimedia Systems and the proposed solution.

The value of obtained results is in the improved searchability of photo album that is driven by IR Multimedia Systems, decreasing the probability of human factor influence on the process of importing new photos into the existing photo album, and improving the consistency of metadata to be used later in search.

1 Overview of a subject

Today, in the world of high technology, high-speed Internet access gives everyone an opportunity to use software to search and to manage the information from any sources. People interact with digital data every day and, over the past 30 years, a continued growth contributes to the development of this type of information as multimedia data. Multimedia itself means data of more than one medium. It usually refers to data representing multiple types of media to capture information and experiences related to objects and events. Commonly used forms of data are numbers, alphanumeric characters, text, images, audio, and video. In common usage, people refer a data set as multimedia only when time-dependent data such as audio and video are involved [1].

The multimedia data is divided into three subclasses: multidimensional, dynamic, and static content.

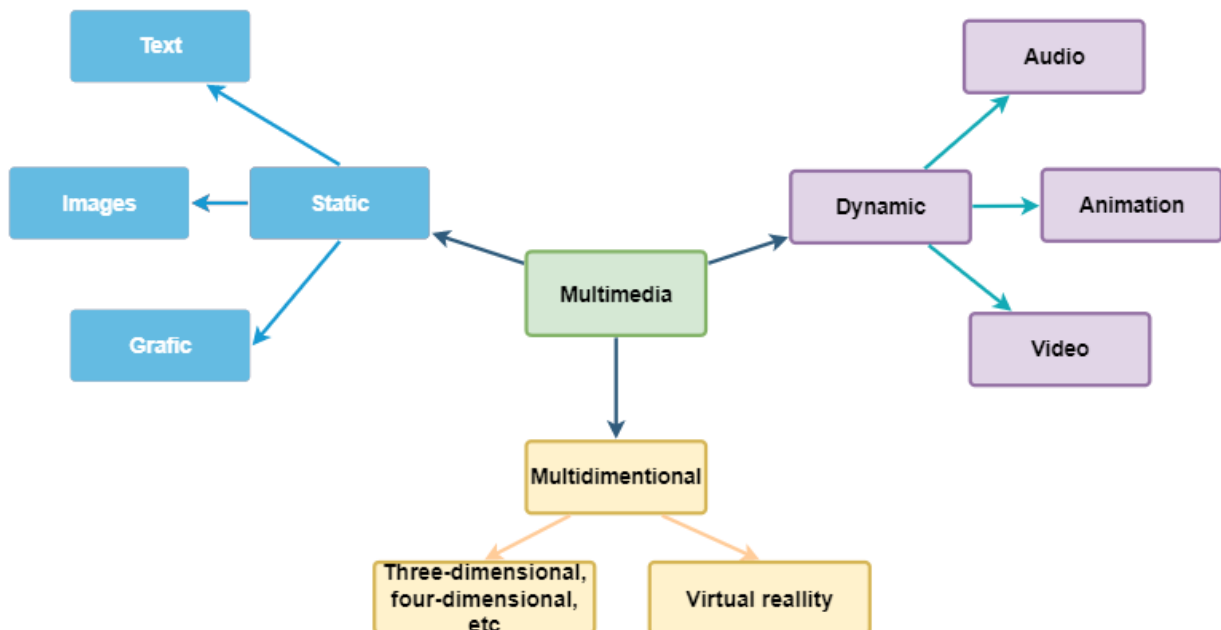


Figure 1. Multimedia data types

Multidimensional (also called spatial) types are:

- Three-dimensional, four-dimensional, five-dimensional, etc.;
- Virtual reality content.

Dynamic types are:

- Video content;

- Animation content;
- Audio content.

Static types are:

- Image or photo content;
- Graphic content;
- Text content.

The most compact (when it is stored) is the text type of information – it is the most important option for people to communicate. Meanwhile, basing on a fact of continued growth of the processor power calculation and extending the disk/memory space and speed, other media content types are also getting popular and widely used.

Since a lot of people own the phones with a camera, and there are numerous webcams, motion cameras, tablets with a camera, the global photo content amount is enormous. By prediction, the humanity is going to take 1,440,000,000,000 photos in 2021 – that's over 1.4 trillion [2].

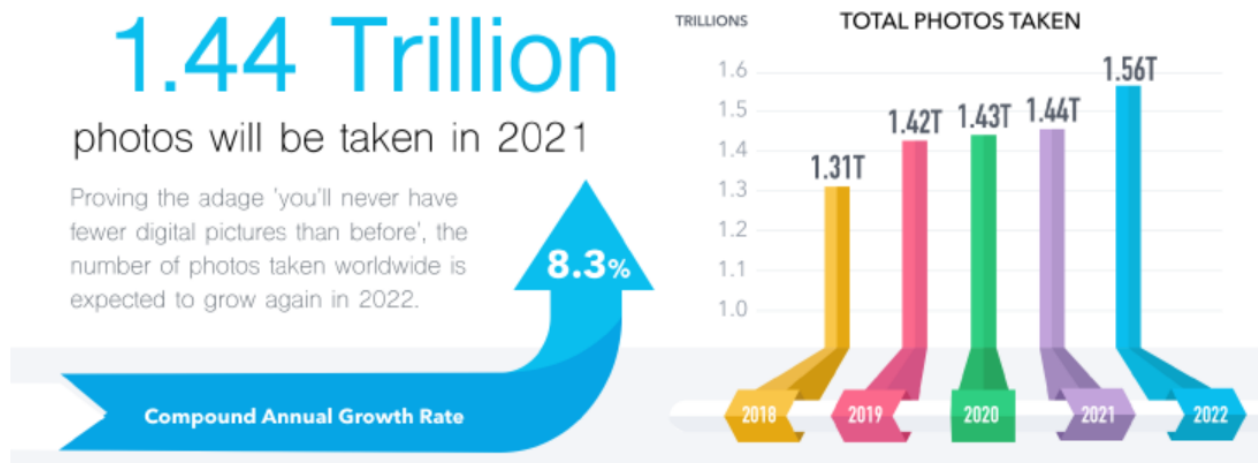


Figure 2. Number of photos to be taken in 2021 and expected growth for 2022 [2]

Due to enormous amount of multimedia content across the world and, often, large number of multimedia content in an average family (like photos), there is a demand to make the search among different types of media. The systems that implement this market request are referred to as Multimedia Information Retrieval (Multimedia IR) systems. In fact, in most cases those systems work with text content.

It is obvious that if such an average family does not have the Multimedia IR system to search for one item in three thousands of photos, it would almost be impossible to find that one item if searching manually. That is why ordinary people would be very interested in Multimedia IR systems and their opportunities they provide.

Moreover, not ordinary people only, but also the businesses might find Multimedia IR systems useful due to their professional needs. For example, there might be different professions who access the photos or images often in their everyday work activity, such as:

- photographers;
- journalists who need to create multimedia news content;
- hospital staff who might search for the medical photos;
- etc.

In this document, we are describing mostly Multimedia IR systems that are accessible for a usual end point user, and not for the corporate users. Those systems are hereinafter referred to in this paper as the Private Multimedia IR systems.

2 Overview and analysis of private Multimedia IR systems

First of all, we provide the description of user needs and requirements for the private Multimedia IR systems from end user point of view.

Then, we select several Multimedia IR systems to compare.

Basing on the comparison, we identify and prioritize the issues found.

Finally, at the end of this chapter we provide an overview of the available solutions on the market to address the issues found.

Let's briefly overview the Multimedia IR systems. According to the Wikipedia [3], Information retrieval (IR) is the process of obtaining information system resources that are relevant to an information need from a collection of those resources. Further, according to the Wikipedia [4] as well, Multimedia information retrieval (MMIR or MIR) is a research discipline of computer science that aims at extracting semantic information from multimedia data sources. Data sources include directly perceivable media such as audio, image and video, indirectly perceivable sources such as text, semantic descriptions, biosignals as well as not perceivable sources such as bioinformation, stock prices, etc. An image retrieval system is a computer system for browsing, searching and retrieving images from a large database of digital images. Most traditional and common methods of image retrieval utilize some method of adding information such as captioning, keywords, title or descriptions to the images so that retrieval can be performed over the annotation words [5]. So, when they talk about private Multimedia IR that works with images and photos they talk about the search based on the metadata, tags and images/photos content [6].

2.1 The user needs and requirements for the private Multimedia IR systems

From end user's perspective, when they are accessing the photo album, the main need is the ability to search through the photo album easily and quickly. In terms of searching ability, it shall also provide the indexing/cataloguing abilities and showing/browsing the search results.

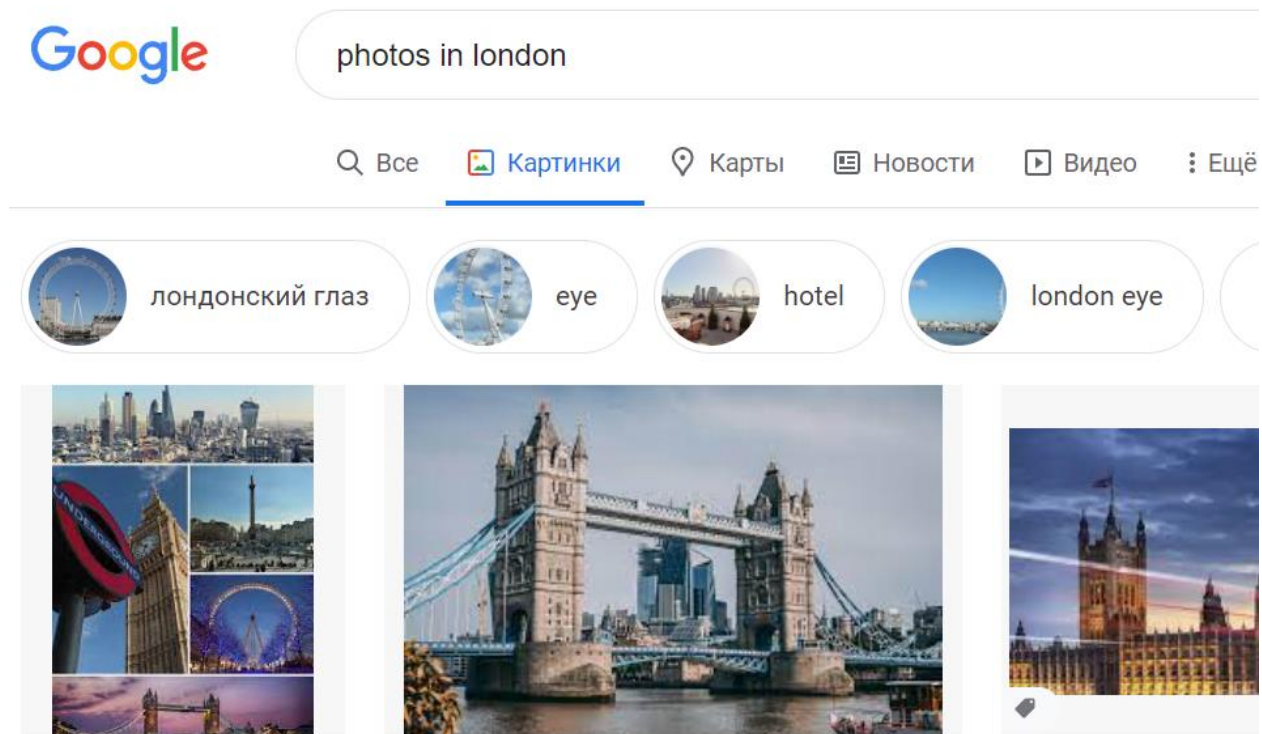


Figure 4. An example of Google response to request to find the photos in London

Let us overview the example of search request to the Multimedia IR system to understand search requests criteria and search requests attributes. Here are the examples of requests that people might use during the search for certain photos:

- The photos with me and my mother;
- All images I took when I was at home;
- The photos taken during the March of 2021;
- The weddings of my friends;
- The forest images;
- Photos of pets;
- Stuff taken by my mother;

- The photos with smiles;
- The photos with the highest rating;
- Photos that are similar to the provided one.

Basing on examples above, we can define the following criteria and attributes for the search:

- Device used to take the photos (like “Nikon E40”);
- The positions – GPS coordinates or textual description of places (e.g., “my work”, “my home”);
- Date and time when the photo was taken (e.g.: 1998/05/15 12:10:11);
- Image’s similarity (the system is configured to return the all photos with similarity level that is, for example, 0.90 to the given photo);
- The people’s emotions on the photos;
- The objects/subjects on photos (for example, “the children”, “my father”, “forests”, “rivers”);
- Mark/ratings (for example, rating that equals to 10);
- Event’s type (friends’ wedding, holiday in Spain).

For sure, the user might use many other photo criteria/attributes, such as Color representation, F-stop, Max aperture, etc. (see the figure below), but we are not going to focus on them in this document.

| Property | Value |
|-----------------------|-----------------------------|
| Color representation | sRGB |
| Compressed bits/pixel | 3 |
| Camera | |
| Camera maker | Canon |
| Camera model | Canon PowerShot G7 X Mar... |
| F-stop | f/1.8 |
| Exposure time | 1/50 sec. |
| ISO speed | ISO-800 |
| Exposure bias | 0 step |
| Focal length | 9 mm |
| Max aperture | 1.6875 |
| Metering mode | Pattern |
| Subject distance | |
| Flash mode | No flash, compulsory |
| Flash energy | |
| 35mm focal length | |
| Advanced photo | |
| Lens maker | |
| Lens model | |

Figure 5. Photo file properties/attributes

Despite the importance of search ability need, there are other important needs from end user's perspective, such as the following:

- the Multimedia IR system shall provide (or be integrated with other tools) an ability to keep the different types of multimedia data;
- they should be able to show any multimedia data types (video, audio, photo content) to user via UI;
- if possible, the Multimedia IR system shall provide editing functions for the user.

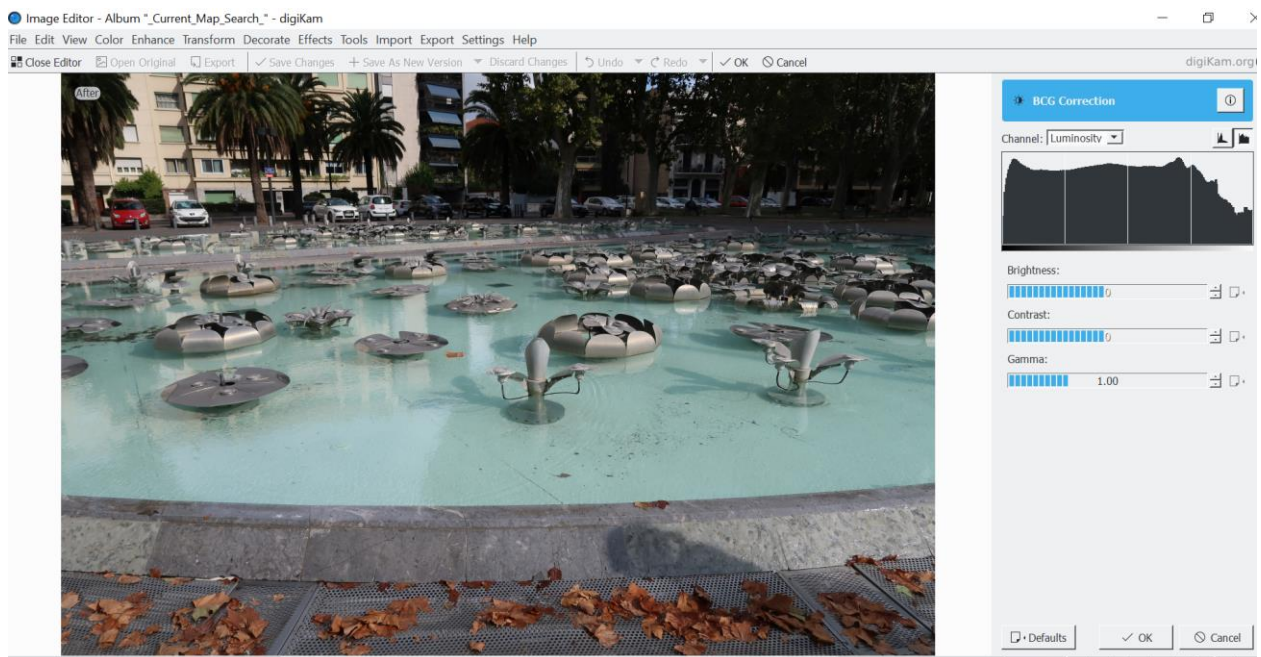


Figure 6. Photo editing function in digiKam [7]

2.2 Comparison of the private Multimedia IR systems

Let us overview and compare several multimedia IR systems by covering requirements and needs we identified in previous chapter. Since some users might feel free to select multimedia IR systems integrated with online storages (like Cloud ones), but others want to use offline multimedia IR systems, we need to choose one offline and one online system. In the same way, we need to add the proprietary and free systems to the comparison as well. Below is the list of them with brief descriptions:

1. Mylio – it is a free app that can be used to organize photos, videos, and other files. This program can be used on any Mac, iOS, Windows, and Android device. Mylio doesn't necessarily need the Internet to function; it can work automatically to organize the files based on the calendar app [8], [10].
2. Google Photo – a photo sharing and storage service developed by Google. The service automatically analyzes photos identifying various visual features and subjects. Users can search for anything in photos, with the service returning results in three major categories: People, Places, and Things [8], [9], [11].
3. digiKam – a free and open-source photo organizing software that can handle more than 100K images. The program has all the photo organizing functionality you will need like uploading, deleting and sorting images [7], [8].

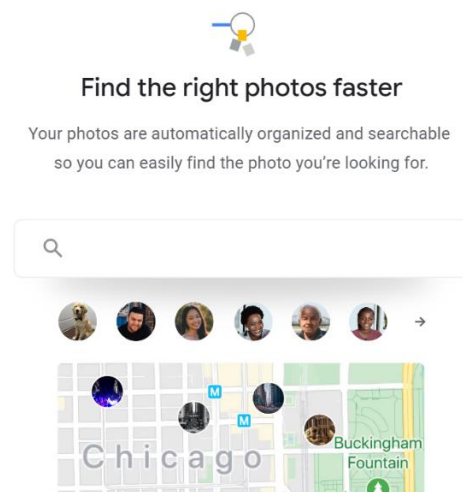


Figure 7. Google Photo welcome screen

Table 1. IR multimedia systems comparison

| Functionality | Mylio [8], [10]. | Google Photo [8], [9], [11] | digiKam [7], [8] |
|----------------------------|---|--------------------------------|---|
| Keeping the photos: | integrated with local and online storages | In the Cloud | integrated with local and online storages |
| Editing the photos | + | + | + |
| Browsing the photos | + | + | + |
| Date/time search | + | + | + |
| Location search | + | + | + |
| Search by people | + | + | + |
| Search by objects/subjects | - | + | - |
| Event search | - | + | - |
| Rating/marks search | + | + | + |
| Search by emotions | - | - | - |
| Similar photos search | - | - | + |
| Search by a photo author | - | - | + |
| Free/paid | Free/Paid | Free/Paid | Free |

2.3 Identification and classification of the issues in IR multimedia systems

This chapter dwells upon the identification and the classification of the issues basing on the comparison the private IR multimedia systems.

As we can see from the previous chapter (namely from the comparison table), all the IR Multimedia systems analyzed provide the following features for end user:

- wide functionality to edit the photos;
- keeping/browsing functions for the private photo album multimedia data.

From the user's point of view all mentioned functions are well-developed and do not require the improvements and substitution with any other software.

Now let's take a look at the searching functions (based on indexing and cataloguing) by a set of search parameter and photo attributes. The following set of searching functions are well-developed:

- search by date/time (using metadata);
- search by location (using metadata);
- search by people (using face recognition technology over multimedia data content);
- search by rating/marks (using metadata).

Still, the following set of searching functions needs to be developed to cover the users' needs:

- search by events;
- search by objects/subjects;
- search by emotions;
- similar photos search;
- search by a photo author.

Even having that set of searching functions implemented, the user might want to use other search attributes like Flash Mode or Flash Energy.

This paper and proposed solution do not cover the above searching functions that need to be developed.

A short usage of these Multimedia IR systems gives us the understanding that the analyzed Multimedia IR systems significantly rely on the multimedia

metadata (to be used for the search by date/time, location, etc.) and the photo content (to be used for the search by people, subjects, etc.) as a main information source. So, the presence and consistency of the metadata is very important for Multimedia IR systems to work and for other purposes [12].

The working experience with one of those systems shows us that the search results might not be full or even incorrect due to inconsistent or not existing multimedia metadata.

Let us provide two examples of incorrect search results due to the problems with metadata. In the first case, we tried to search for the photos with the existing Geo positions and to see them on a map. In Figure 8 we see the search results for that request that returned 4 photos taken in the Gulf of Guinea, near Ghana, marked by the red circle.

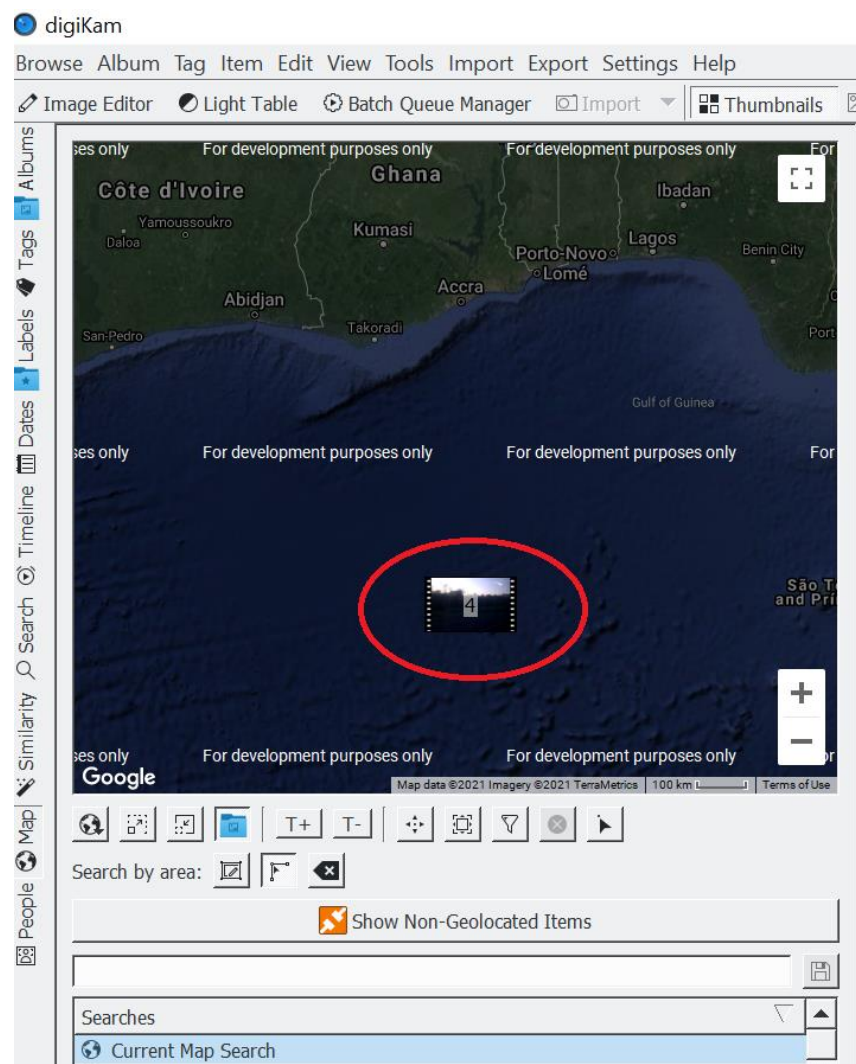


Figure 8. The incorrect search results due to incorrect metadata position

In fact, nobody has taken photos in the Gulf of Guinea, near Ghana. The system has provided such results because of incorrect metadata.

In the second case, we searched for the photos taken in 2011. In Figure 9 we see the search results for that request that returned 1 video, which is underlined with the red color.

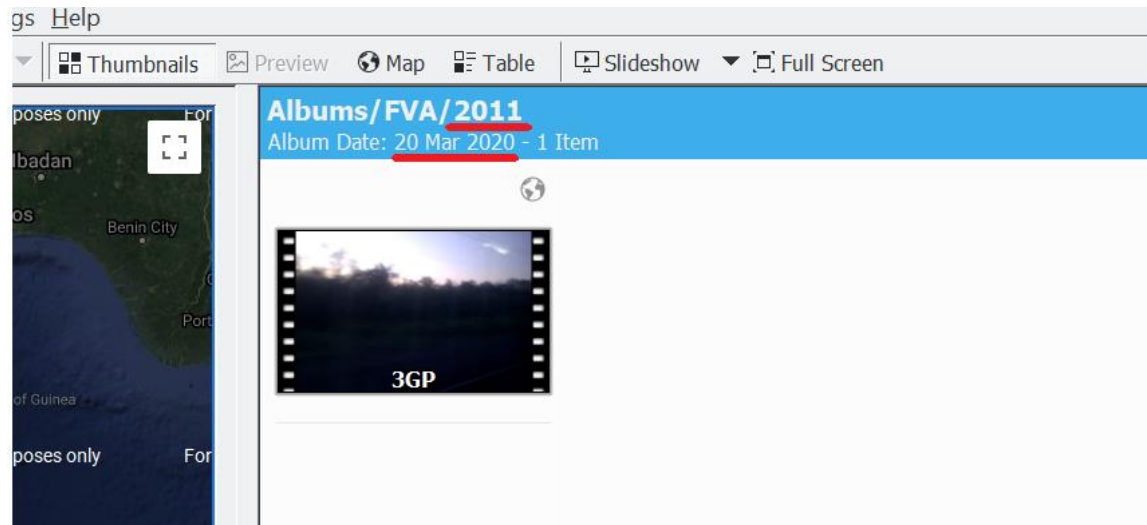


Figure 9. The incorrect search results due to incorrect date/time metadata

Actually, we got one video that, due to incorrect date/time metadata, got recognized dated as of March 2020.

The reasons for incorrect and absent metadata are rather different – from not setting proper data on the device that takes the photos to the privacy issues. This resource [13] describes the metadata importance in more details and there is information about other reasons of incorrect and absent metadata.

Therefore, the proposed solution shall focus on identification of inconsistent or not existing photo metadata and suggest the ways to solve it in a semi-automated or fully automated way.

2.4 Overview of the solutions to address the issues found

This chapter, basing on the previous one, will provide the possible solutions in terms of the following:

- identification of inconsistent or not existing photo metadata;
- suggesting the ways to solve the inconsistency and absence.

The most automated approach possible should be used because a user is going to work with a large number of items. A toolset available for a usual user and not for the enterprises shall be used as well.

Let us review several solutions that might identify and fix incorrect metadata.

The first one is based on use of Lightroom to solve and avoid metadata conflicts. The source [14] suggests to update metadata by using keywording strategies; it also suggests using batch updates and explains the metadata correction for time capture, etc. Despite the broad functionality, it still requires a big effort from a user and looks rather complicated for usual user.

The following options for us will be the simplest, but at the same time the least automated way to achieve our goal.



Figure 10. “Add Description” option is available on a camera once a photo has been taken [15]

The article [15] suggests adding the description to a photo in a predefined format, using UI on a camera to save any information required later. Such information might include date/time, location, etc. to be used in search later. It is

obvious, though, that to implement such option a photographer has to have a camera with that UI and each photo shall be updated with the valuable description if the photo is going to be saved.

The third option implements a way to fix the metadata using templates [16].

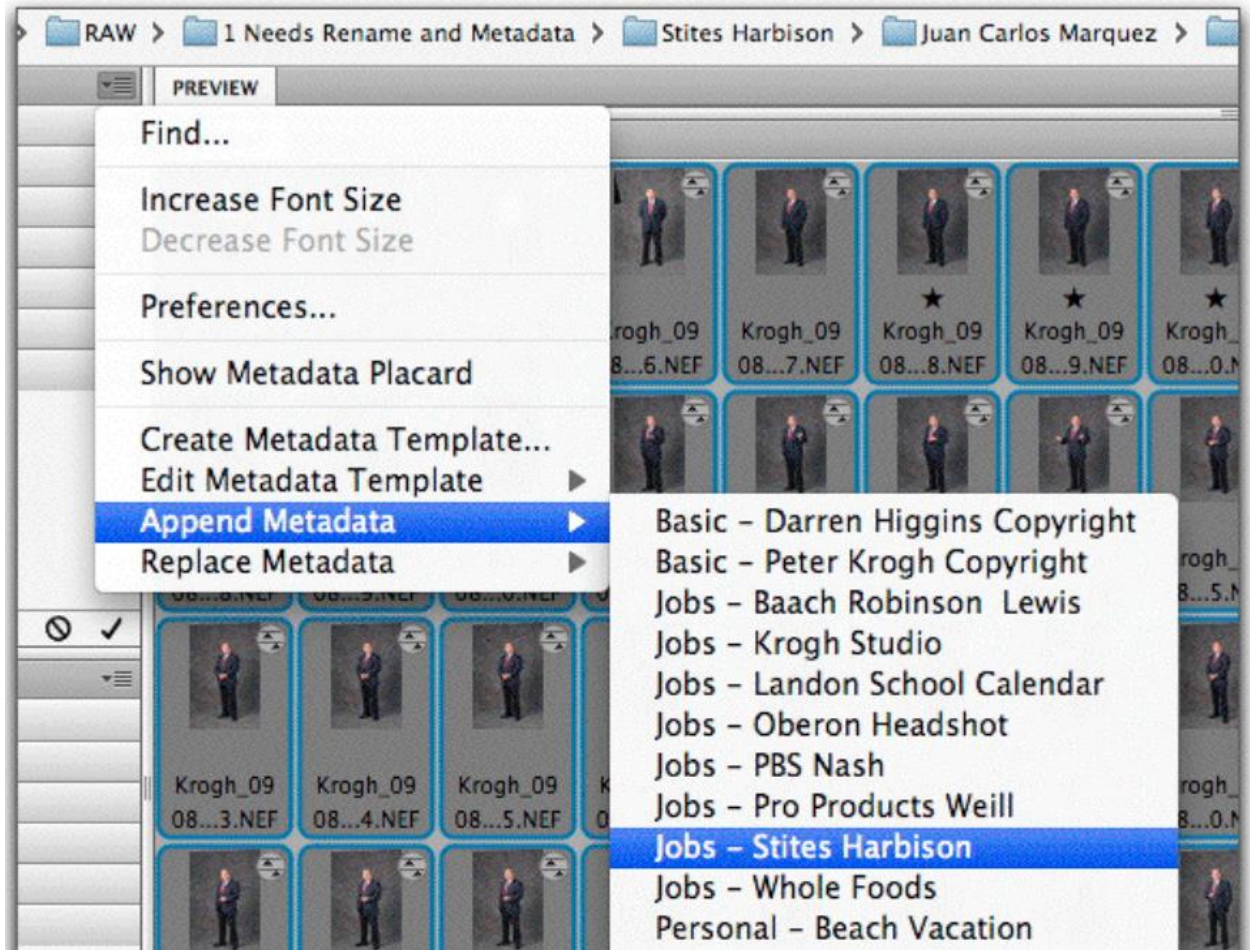


Figure 10. Updating the metadata using templates [16]

The article [16] tells us that updating the metadata using a template is the easiest way. Still, using this approach, a user will also need to find inconsistencies in the metadata and create the templates to apply them when necessary. It will also require a technical qualification to do that, which is not applicable to most users.

The last option described here is an algorithm described in a Google patent [17]. It seems to be a very efficient way to use; however, it is not implemented in the appropriate software and therefore is unavailable for usual users.

3 A proposed solution to enrich the photo metadata

Thus, basing on the previous chapter we found no solution available on the market that does fully cover the needs for:

- identification of inconsistent or not existing photo metadata;
- suggesting the ways to solve the inconsistency and absence.

The needs described above will be the main drivers to create the proposed solution.

First, this chapter describes the User Journey to understand the cadence of proposed solution in use and the interaction with the user.

Further on, it outlines the set of functional and non-functional requirements basing on chapter “Overview and analysis of private Multimedia IR systems”.

Then it provides a high-level technical design for the proposed solution in form of context diagram and its description.

At the end, it describes the functional decomposition of the proposed solution and provides a diagram for it as well.

Basing on information provided in this chapter, the project implementation will be suggested in scope of the next chapter.

3.1 User journey to work with the proposed solution

Let's consider the following flow to understand the cadence of the proposed solution in use and the interaction with the user:

1. First, the photo album owner inserts a device to a desktop or laptop to start downloading the multimedia data from it.
2. Then they prepare an input folder (the input folder might be created or be cleared at that time) on the desktop or laptop to copy the multimedia data into.
3. As soon as the previous step is done, the user starts copying or moving the multimedia data from the inserted device into the prepared folder using any tool.
4. Now the user can extract the device used to copy multimedia data from.
5. At this point, the user is ready to import the copied multimedia data to existing photo album and so now they need to start the proposed solution to check the multimedia metadata and to enrich it if necessary.
 - a. As soon as the user starts the solution, the welcome screen shall be displayed. Here, the user shall be able to optionally open the UI to configure the system before starting to use it.
 - b. On the next screen, the user shall be able to select the path to the folder where copied multimedia data is kept, with a button to start processing.
 - c. As soon as the user starts the processing, it shall start checking the multimedia metadata and enriching it in an automated way. The progress screen shall be shown there to notify the user about the current operation status.
 - d. In case the system is not able to enrich the multimedia metadata it shall ask a user to support it.
 - e. Once the processing is finished, the system shall ask the user if they want to send the enriched multimedia data to Multimedia IR systems.

- f. As a final step, it shall show ‘done’ status screen with a report of performed actions, e.g., what was checked, what was enriched and what was sent to Multimedia IR systems.
 - g. Now the user can close the proposed solution.
6. As soon as the multimedia data is checked and enriched, the user can start using the Multimedia IR systems to search the existing photo album and recently added multimedia data.

The screenshot shows the 'Advanced Search - digiKam' window. The title bar includes the application icon, the text 'Advanced Search - digiKam', and standard window controls (minimize, maximize, close). The window has a blue header bar with the text 'Find Items' and a sub-header 'Search your collection for Items meeting the following conditions'. A link 'Options >>' is visible in the top right of the header. Below the header, there is a search criteria section with a text input field labeled 'Find items that have associated all these words:'. The main content area is divided into several sections by horizontal lines: 'File, Album, Tags', 'Picture Properties', 'Audio/Video Properties', 'Caption, Comment, Title', and 'Photograph Information'. The 'File, Album, Tags' section contains search criteria for 'Album' (Search items located in, The album name contains, The album category is) and 'Tags' (Return items with tag, A tag of the item contains, item has no tags). The 'File Name' section contains a criterion for 'Return items whose file name contains'. The 'Photograph Information' section is currently empty. At the bottom of the window, there is a blue bar with four buttons: '+ Add Search Group', 'Reset', 'OK', and 'Cancel'. The 'Try' button is also visible on the right side of the bottom bar.

Figure 11. Example of search process configuration for digiKam [7]

3.2 The functional and non-functional requirements to the system

Basing on issues identified in “Overview and analysis of private Multimedia IR systems”, let us formulate the functional and non-functional requirements in the following two tables.

Table 2. Functional system requirements

| # | Requirement |
|--------|--|
| Req 1 | The system shall check the photo metadata for inconsistency in the date/time attribute. |
| Req 2 | The system shall check the photo metadata for absence of the date/time attribute. |
| Req 3 | The system shall provide the user with an ability to configure the checks for photo metadata inconsistency via UI. Also, the user shall be able to configure integrations with online and offline Multimedia IR systems, and change the log level. |
| Req 4 | The system shall enrich the photo album metadata in case of metadata absence in automated approach without the user’s involvement. |
| Req 5 | If it is not possible to enrich the metadata without the user’s involvement, the system shall ask h user via UI to provide the value for absent metadata. |
| Req 6 | The system shall fix the photo album metadata inconsistency in automated way without the user’s involvement for the date/time attribute. |
| Req 7 | If it is not possible to fix the metadata inconsistency without the user’s involvement, the system shall ask the user via UI to provide the value for consistent metadata. |
| Req 8 | The system shall provide a possibility to enrich the metadata for the date/time attribute. |
| Req 9 | The system shall provide a possibility to fix the inconsistent metadata for the date/time attribute. |
| Req 10 | The system shall be integrated with online Multimedia IR system - Google Photo |
| Req 11 | The system shall be integrated with offline Multimedia IR system - digiKam |

| | |
|--------|--|
| Req 12 | The system shall interact with the user during importing the new photos to an existing photo album |
|--------|--|

Table 3. Non-functional system requirements

| # | Requirement | Quality Attribute |
|--------|--|----------------------|
| Req 13 | The software shall be decomposed into well-structured modules according to functional decomposition. The code shall be developed using Google code style [18]. | Conceptual Integrity |
| Req 14 | The proposed solution shall have the ability to be integrated with online and offline Multimedia IR systems in the common way. | Interoperability |
| Req 15 | The system shall have the UI allowing users to quickly become familiar with it and be able to make good use of all their features and capabilities. The user guideline shall be available. | Learnability |
| Req 16 | The system shall be able to undergo changes with a high level of easiness. The changes here mean adding/changing the UI, adding/changing new modules and features. | Maintainability |
| Req 17 | The system shall be monitorable via UI and logging. The system shall support debugging as well. | Manageability |
| Req 18 | The same as for the whole system, the UI shall respond in less than 1 second. As there might be time-consuming jobs, it shall show the current progress for such operations to provide the user with a status. | Performance |
| Req 19 | The software shall support all wide known desktop/laptop operating systems with the latest update: Windows 10, MacOS Big Sur, Linux Ubuntu 20. No support of the web or mobile platform required. | Portability |

| | | |
|--------|--|----------------|
| Req 20 | If the system is going to fail during an operation, it shall report the operation status to user with error description and roll back all applied changes for this operation. | Reliability |
| Req 21 | The system components shall be usable in other functionality or other flows (to deduplicate of implementation, and to reduce the implementation time spent). | Reusability |
| Req 22 | During abnormal function, the software shall provide the information to find and to solve the issues via prompting user in UI and saving detailed information into the log file. The user shall be able to change a log level as configurable parameter. | Supportability |
| Req 23 | The system shall be intuitive, easy to localize (to support multiple time zones, languages, etc.) and globalize. | Usability |

Thus, the proposed system shall implement the above sets of functional and non-functional requirements.

3.3 Context diagram of the System

The context diagram below shows the solution bounds and its interactions with other objects and subjects.

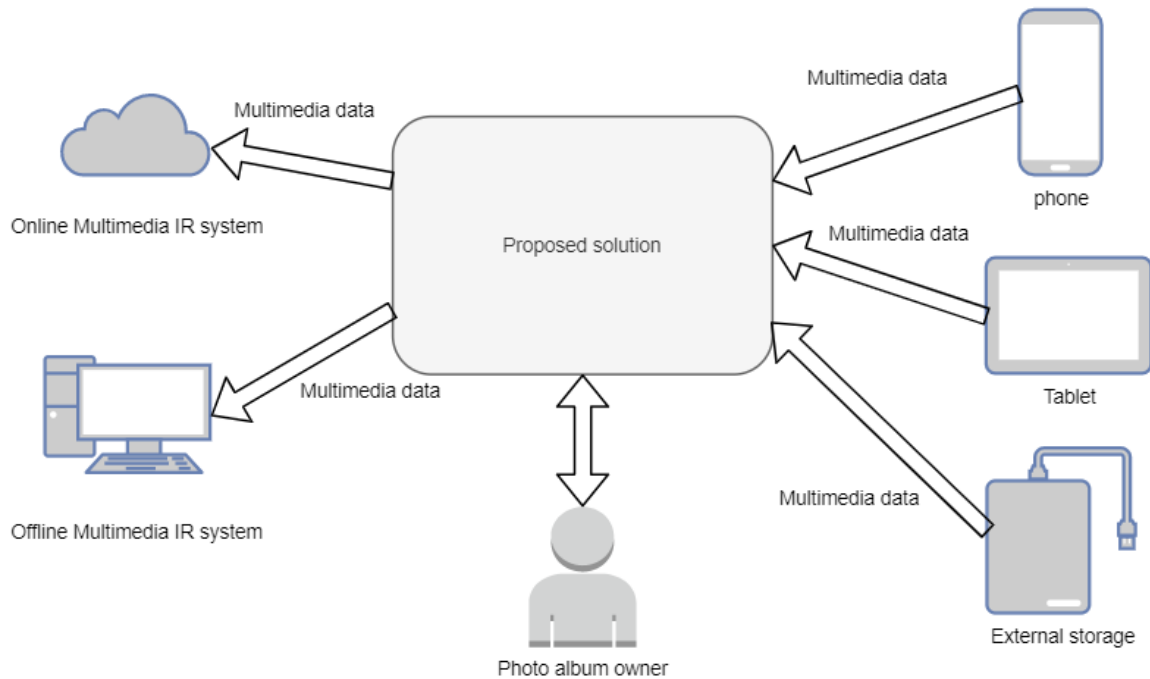


Figure 12. Solution context diagram

The photo album owner is the main and only user of the solution.

The solution is used during import of new multimedia content from external devices to an existing user photo album.

The external devices might be: the phones with a camera, the tablets with a camera, the photo/video cameras, external storages (e.g., hard drive or flash card) with multimedia data, etc.

Once the multimedia data from the external devices is processed, the solution interacts with integrated online and offline Multimedia systems.

3.4 Functional decomposition of the System

Below is a diagram to show functional decomposition of the system, the connections between the components, between the components and photo album owner and external systems.

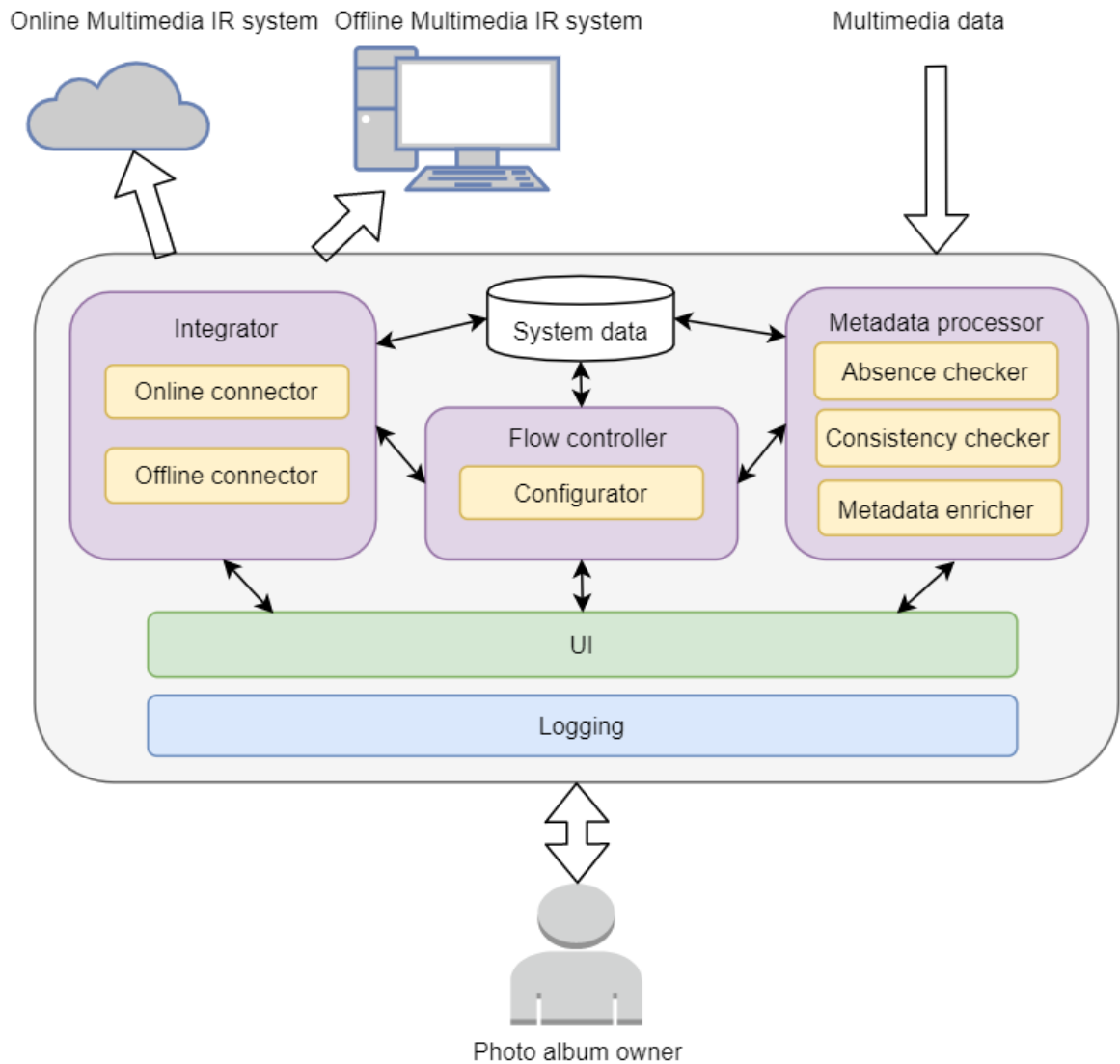


Figure 13. System functional decomposition

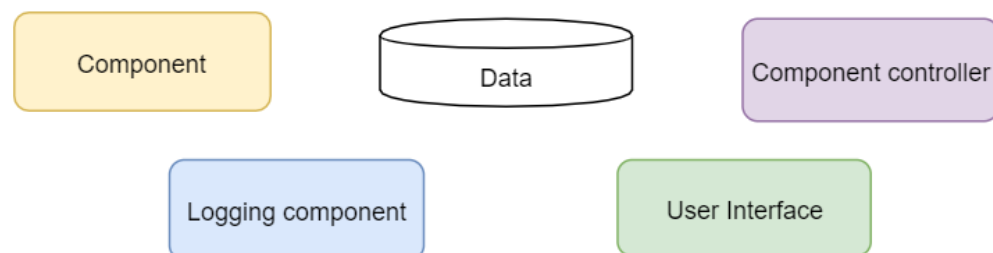


Figure 14. System functional decomposition agenda

There are five main components of the system:

- User Interface (UI);
- Integrator;
- Flow controller;
- Metadata processor;
- System data.

The photo album owner interacts with the system through the user interface (UI). Basing on flow and user needs, UI interacts with other main components described below.

The purpose of Flow Controller is to manage the action flows between UI, the Integrator and the Metadata processor. The flows can be configured by the Configurator component. The main function of the Configurator is to provide the settings to other components when requested and to save/load them to/from System data.

The function of the System data is based on keeping the configuration, user saved actions and the change logs for the metadata updates.

The Integrator purpose is to manage the integration points to the external systems like Online and Offline Multimedia IR systems. Thus, under the hood, there might be Online and Offline connectors that interact with external systems like Google Photo and digiKam.

The Metadata processor performs such actions as checking the multimedia Metadata for presence and consistency and enriching it. Under the hood, there are three components to perform the above mentioned actions:

- Absence checker;
- Consistency checker;
- Metadata enricher.

All the main components interact with the Logging component to log the important events during processing.

4 Project implementation

In previous chapter we identified the functional requirements and non-functional requirements as well. Also, we provided the context diagram to define the solution boundaries and showed there the integration with external systems like online and offline Multimedia IR systems. Interaction with the user has been defined there as well. Finally, we provided functional decomposition of the proposed solution and mapping of functional requirements on the functional components.

This chapter is focusing on solution implementation and provides the description of:

- the coverage of functionality by the implementation;
- the approaches and patterns to address non-functional requirements;
- implementation of integration with online and offline Multimedia IR systems;
- the code with the snippets.

4.1 The coverage of functionality by components

Basing on chapter “Functional decomposition of the System” we can match the functional components with the functional requirements to get traceability matrix shown below based on Table 1:

Table 3. Functional traceability matrix

| # | Component Name | Requirement IDs |
|----|---------------------|---|
| 1. | Consistency checker | Req 1 |
| 2. | Absence checker | Req 2 |
| 3. | UI | Req 3, Req 5, Req 7, Req 10, Req 11, Req 12 |
| 4. | Configurator | Req 3 |
| 5. | Metadata enricher | Req 4, Req 6, Req 8, Req 9 |
| 6. | Flow controller | Req 3, Req 5, Req 7, Req 10, Req 11, Req 12 |
| 7. | Online connector | Req 10 |
| 8. | Offline connector | Req 11 |

So, the above components are going to be implemented to cover functional requirements.

Additionally, the following components are going to be implemented:

- Integrator;
- Logging;
- System data;
- Metadata processor.

Let’s match them with non-functional requirements to cover presented in the table below:

Table 4. Non-functional traceability matrix

| # | Component Name | Requirement IDs |
|----|--------------------|--|
| 1. | Integrator | Req 13, Req 14, Req 16, Req 17, Req 20, Req 21 |
| 2. | Logging | Req 13, Req 17, Req 19, Req 20, Req 21, Req 22 |
| 3. | System data | Req 13, Req 16, Req 19, Req 20, Req 21, |
| 4. | Metadata processor | Req 13, Req 16, Req 17, Req 18, Req 19, Req 20, Req 21 |

4.2 The approaches and patterns to address non-functional requirements

The following approaches are going to be used to support Req 13:

- C++ code and python code shall be developed using Google code style [18].
- The code shall be checked by automatic linters like Pylint and CppLint.
- The “Return Code” and not “Exception” approach shall be used to provide a result to calling code.
- The whole code shall be decomposed into components according to the chapter “Functional decomposition of the System” according to that requirement.
- The Service Oriented Architecture (SOA) shall be applied to the system.

To support Req 16 and Req 14, the OOP, SOLID principles and MVC pattern are going to be used, so that the system is able to undergo changes with a high level of easiness. The changes might be:

- changes applied to the UI like adding new screen;
- adding new multimedia formats to support;
- new metadata search criteria to handle;
- new online or offline Multimedia IR systems to integrate, etc.

To cover Req 21, a common library is going to be created to keep the following functions used by several components:

- To work with configuration saved in the System Data;
- Logging functions;
- Common UI functions.

The code shall be documented using Doxygen to increase reusability and supportability (Req 21).

To address the Req 17 and Req 22, each component shall use UI and logging component:

- to save any important events to disk;
- to show the progress of processing;

- to show any important events to user;
- to provide the user with an ability to cancel the processing.

To address Req 15, the proposed solution shall be developed to have the UI that allow users to quickly become familiar with it. The working instruction also shall be provided to help the user to get familiar with the system.

To support Req 23, the Qt shall be used with such features as:

- QTranslator;
- QTextCodec;
- QLocale;
- QTextDecoder;
- QTextEncoder.

Also, the installation package must be provided for each supportable platform from Req 19 to increase the usability of product.

To support Req 18, any UI shall respond in less than 1 second. So, the native fast API to implement UI shall be developed based on Qt. In case the UI needs to operate with time-consuming jobs like multimedia data processing, it shall show current progress UI to provide a current operation status to the user.

To address Req 20, in case of failing, the system shall:

- show the status to user via UI;
- save the error message to the log;
- roll back all applied changes for this operation (applicable to the Metadata enricher).

The Qt library shall be used for the implementation to support portability requirement (according to Req 19) to work on all wide known desktop/laptop Operating systems with the latest update.

4.3 Integration with offline and online Multimedia IR systems

As it was mentioned before in the functional requirements, the integration with offline Multimedia IR system is shown through the integration with digiKam and integration with online Multimedia IR system is shown through the integration with Google Photo.

Because the Multimedia IR system performs the search by certain criteria basing on multimedia metadata, the proposed solution shall be working with the same multimedia metadata to be integrated with Multimedia IR system.

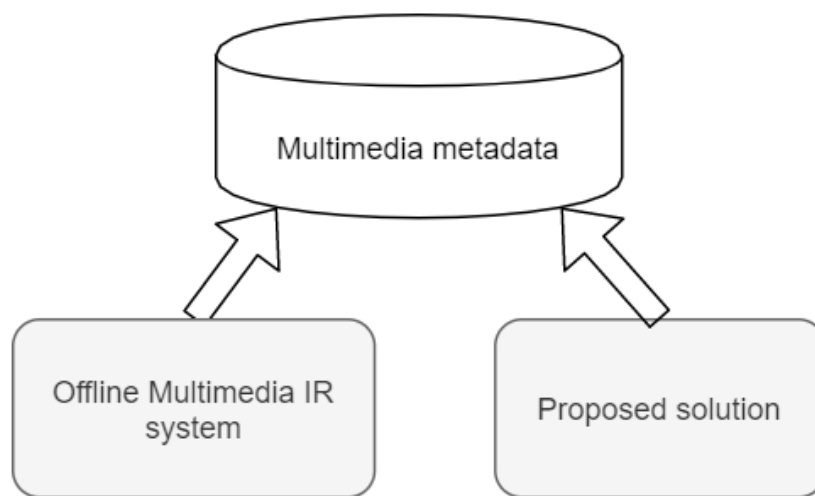


Figure 15. Integration between the proposed solution and offline Multimedia IR system

Thus, the proposed solution shall be installed on the same environment that Multimedia IR system is already working on. Moreover, the same photo album storage shall be used for offline or online Multimedia IR system and the proposed solution. Moreover, the user, which the proposed solution is going to run under, shall have the write access to storage of the photo album.

For digiKam, the Collection Setting shall be used to define how it is configured to search the multimedia data through. There the setting “Monitor the albums for external changes (requires restart)” shall be enabled. Taking into account these settings, the digiKam Multimedia IR system will pick up the external changes in the collection as soon as the proposed solution will put it in. Please see the Figure 16 for more details.

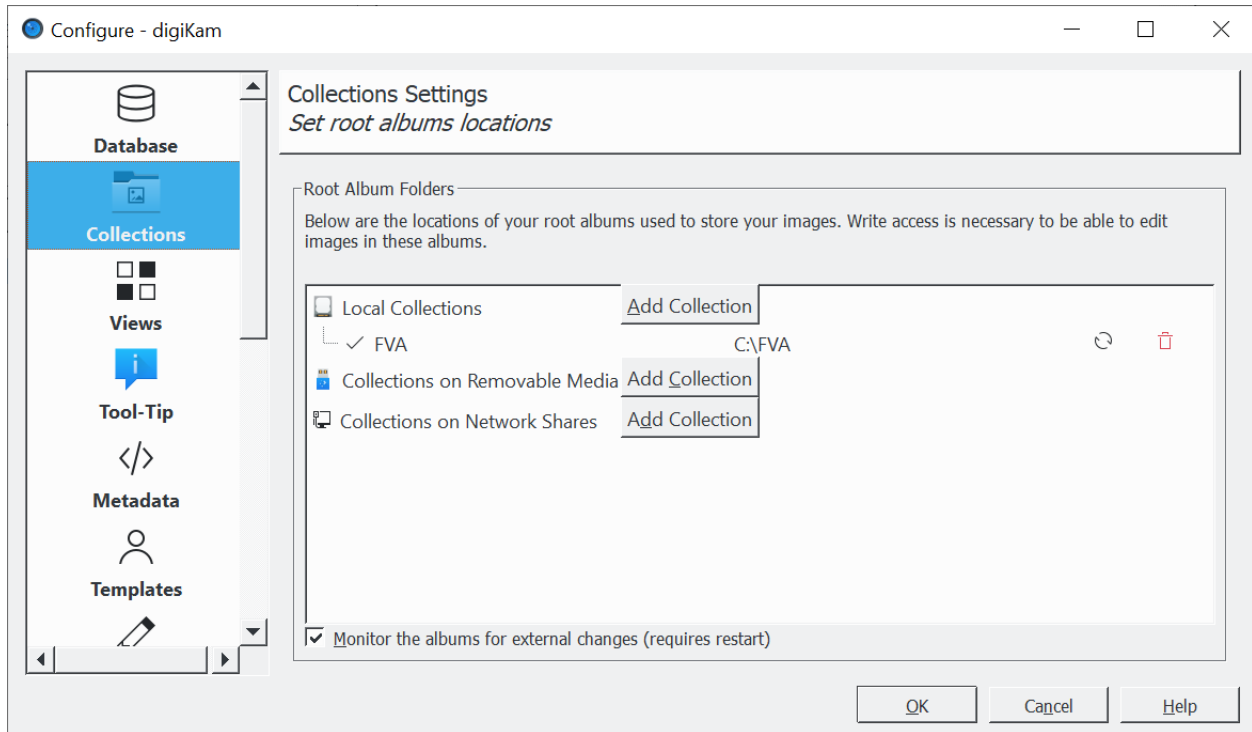


Figure 16. Collection setting of digiKam for integration with the proposed solution

Still, from the side of the proposed solution it is required to provide a possibility to configure the storage to put the multimedia data processed to be integrated with digiKam. So, the separate UI screen shall be created to ask the user what storage location digiKam should use for the incoming multimedia data.

For example, by the Local Collections settings the directory “C:\FVA” is selected on Figure 16. It means that when the user wants to integrate the multimedia data after processing by the proposed solution, the user shall be able to select the “C:\FVA” directory as a directory by UI screen to put the processed multimedia data in. As soon as it is done, the proposed solution will put multimedia data into selected directory and Multimedia IR system will be able to search through it.

In the same manner, the integration with Google Photo is going to be implemented. First of all, it shall be mentioned that Google Photo shall be configured to be integrated with the proposed solution. Thus, appropriate file types shall be set up as in the figure below:

Choose File Types

☒ Back up all files and folders

☐ Back up photos and videos

☒ Back up screenshots

☒ Back up RAW files

► [Advanced settings](#)





OK

Figure 17. Setting of Google Photo for file types used for searching

Also, the folders for synchronization shall be configured to be used later in the proposed solution. The example of configuration is shown below in Figure 16.

My Laptop

Choose folders to continuously back up to Google Photos and Google Drive

| | | |
|-------------------------------------|---|------------------|
| <input type="checkbox"/> |  | Desktop 14 KB |
| <input type="checkbox"/> |  | Documents 1.5 GB |
| <input checked="" type="checkbox"/> |  | Pictures 0 MB |
| <input checked="" type="checkbox"/> |  | fromDevices 0 MB |

[CHOOSE FOLDER](#)
Backing up all files and folders
[Change](#)

Uploading photos and videos in [High quality](#) [Learn more](#)

 **Google Drive** [Learn more](#)

Removing items synced between Google Drive and this computer

Ask before removing both copies ▼

 **Google Photos** [Learn more](#)

☒ Upload newly added photos and videos to Google Photos
Items that are deleted from this computer will not be removed from Google Photos

Figure 18. Setting of Google Photo for folders used for searching

For example, by the Folders settings the “Pictures” and “fromDevices” directories are selected in Figure 18. It means that when the user wants to integrate the multimedia data after processing by the proposed solution, the user shall be able to select the “fromDevices” or “Pictures” directory in UI screen, as a directory to put the processed multimedia data in. As soon as it is done, the proposed solution will put multimedia data into selected directory and Google Photo will be able to search through it as soon as it synchronizes the multimedia data.

It is assumed that the user, which the proposed solution is going to run under, has the write access to the Google Photo folder where multimedia data processed is going to be put in.

4.4 The description of code with snippets

This chapter is focusing on description of the most important parts of code and provides the code snippets.

Please see the appearance UI classes in the Appendix A and collaboration/inheritance diagrams for all classes in the Appendix B.

4.4.1 The description of UI code

The UI consists of the following classes:

FVAOrganizerWizard is child of QWizard and implements the flow with a User to interact. The flow consists of following steps:

1. Start Page - class FVAOrganizerStartPage;
2. Input Dir Page - class FVAOrganizerInputDirPage;
3. Output Dir Page - class FVAOrganizerOutputDirPage;
4. Done Page - class FVAOrganizerDonePage.

Each page/step is a separate class inherited from QWizardPage. Showing the elements and flow is based on class FvaConfiguration. All those classes implement "View" functions from MVC pattern.

FVAOrganizerStartPage is a child of QWizardPage and implements the next UI functions:

1. "Welcome words" to user in a QTextBrowser;
2. "Configuration" button (QPushButton) to start Configurator UI before System to use.

FVAOrganizerInputDirPage is a child of QWizardPage and implements the next UI functions:

1. QLineEdit for the input directory to get the Multimedia Data from;
2. QTextBrowser to output the logging events;
3. QPushButton to open QFileDialog to select input directory.

FVAOrganizerOutputDirPage is a child of QWizardPage and implements the next UI functions:

1. QLineEdit for the digiKam directory to put the Multimedia Data in;
2. QLineEdit for the GooglePhoto directory to put the Multimedia Data in;

3. QCheckBox to remove origin (input) directory to get the Multimedia Data from.

FVAOrganizerDonePage is a child of QWizardPage and implements the showing "Finish words" to user in a QTextBrowser.

As soon as a user has selected the input directory to check the multimedia data in and has pressed “Next” button in the Input Dir Page, the validatePage function is being called that is implemented as following:

```
bool    FVAOrganizerInputDirPage::validatePage ()
{
    // get the dir the user selected
    QString dir = inputDirLineEdit->text();

    // create the flow controller to proceed
    FVAFlowController flow;
    DeviceContext deviceContext;

    // call the flow controller PerformChecksForInputDir method to check the multimedia data and fix if needs
    FVA_EXIT_CODE exitCode = flow.PerformChecksForInputDir(dir, deviceContext,this);
    if (exitCode != FVA_NO_ERROR)
        return false;

    ((FVAOrganizerWizard*)wizard())->inputFolder(dir);
}
```

Figure 19. InputDirPage::validatePage function implementation

That code:

1. gets the directory the user selected;
2. create the flow controller to proceed;
3. call the flow controller PerformChecksForInputDir method to check the multimedia data and to fix it if needs.

As soon as a user has selected the output directory to integrate the multimedia data into offline or online Multimedia IR systems and has pressed “Next” button in the Output Dir Page, the validatePage function is being called that is implemented as following:

```

bool    FVAOrganizerOutputDirPage::validatePage ()
{
    // create the flow controller to proceed
    FVAFlowController flow;

    // prepare list of directories to pass later to the FVAFlowController
    STR_LIST dirList;

    // if googlePhotoLineEdit is not empty
    if (!googlePhotoLineEdit->text().isEmpty())
        // add this folder into list
        dirList.append(googlePhotoLineEdit->text());

    // if digiKamLineEdit is not empty
    if (!digiKamLineEdit->text().isEmpty())
        // add this folder into list
        dirList.append(digiKamLineEdit->text());

    // call the Flow Controller MoveInputDirToOutputDirs method to integrate the input data into Multimedia IR systems
    FVA_EXIT_CODE exitCode = flow.MoveInputDirToOutputDirs(
        ((FVAOrganizerWizard*)wizard())->inputFolder(),
        dirList,
        removeOriginDirCheckBox->isChecked());
}

```

Figure 20. OutputDirPage::validatePage function implementation

That code:

1. creates the flow controller to proceed;
2. prepares a list of directories to pass later to the FVAFlowController;
3. if googlePhotoLineEdit is not empty, adds this folder into list;
4. if digiKamLineEdit is not empty, adds this folder into list;
5. calls the Flow Controller MoveInputDirToOutputDirs method to integrate the input data into Multimedia IR systems.

4.4.2 The description of FlowController code

FVAFlowController is a main class to keep a logic of the operation flow and interaction with a user. It implements such functions as:

1. PerformChecksForInputDir - it performs the checks for input folder according to application configuration;
2. MoveInputDirToOutputDirs - it performs the moving input folder content to output folder with checks according to event configuration.

This class implements "Controller" functions from MVC pattern. Flow control is based on class FvaConfiguration.

Let's review PerformChecksForInputDir function of FVAFlowController:

```
FVA_EXIT_CODE FVAFlowController::PerformChecksForInputDir(const QString& dir, DeviceContext& deviceContext, QObject* obj)
{
    // create command-line-task context to keep common parameters for all commands
    CLTContext context;

    // set up the dir that user selected in UI
    context.dir = dir;

    // perform common checks
    FVA_EXIT_CODE res = performCommonChecks(context, m_cfg);

    // return to calling function if previous operation failed
    RET_RES_IF_RES_IS_ERROR

    // do we need to search by device?
    bool SearchByDevice = false;

    // ask configuration if we need to search by device
    FVA_EXIT_CODE exitCode = m_cfg.getParamAsBoolean("Search::Device", SearchByDevice);

    // show error message box and return to calling function if previous operation failed
    IF_ERROR_SHOW_MSG_BOX_AND_RET_EXITCODE("getParamAsBoolean(Search::Device)")
    if (SearchByDevice)
    {
        // perform device checks
        FVA_EXIT_CODE res = performDeviceChecks(deviceContext, context, m_cfg);

        // return to calling function if previous operation failed
        RET_RES_IF_RES_IS_ERROR
    }

    // do we need to search by date-time?
    bool SearchByDateTime = false;

    // ask configuration if we need to search by date-time
    exitCode = m_cfg.getParamAsBoolean("Search::DateTime", SearchByDateTime);

    // show error message box and return to calling function if previous operation failed
    IF_ERROR_SHOW_MSG_BOX_AND_RET_EXITCODE("getParamAsBoolean(Search::DateTime)")
    if (SearchByDateTime)
    {
        // perform date-time checks
        FVA_EXIT_CODE res = performDTChecks(context, m_cfg, obj);

        // return to calling function if previous operation failed
        RET_RES_IF_RES_IS_ERROR
    }
}
```

Figure 21. FVAFlowController::PerformChecksForInputDir function implementation

That code:

1. creates command-line-task context to keep common parameters for all commands;
2. sets up the directory that user selected in UI;
3. performs common checks;
4. returns to calling function if previous operation failed;
5. asks configuration if we need to search by device;
6. shows error message box and return to calling function if previous operation failed;
7. performs device checks;
8. returns to calling function if previous operation failed;

9. asks configuration if we need to search by date-time;
10. shows error message box and return to calling function if previous operation failed;
11. performs date-time checks;
12. returns to calling function if previous operation failed.

Full implementation of `FVAFlowController::PerformChecksForInputDir` with code description can be found in Appendix C.

Let's review `performDTChecks` function of `FVAFlowController`. That code:

1. prepares context to run `CheckDataTime` command;
2. runs `CheckDataTime` command in Data Processor;
3. checks if Data Processor said there is no exif date time;
4. asks configuration if we can fix "no exif date time" issue by picture modification time;
5. shows error message box and return to calling function if previous operation failed;
6. shows error to user if we can NOT fix "no exif date time" issue by picture modification time;
7. shows message to user if we can fix "no exif date time" issue by picture modification time;
8. runs command implemented in python to fix empty date-time issue;
9. shows error message box and returns to calling function if previous operation failed.

Full implementation of `FVAFlowController::performDTChecks` with code description can be found in Appendix E.

Let's review MoveInputDirToOutputDirs function of FVAFlowController:

```
FVA_EXIT_CODE FVAFlowController::MoveInputDirToOutputDirs(const QString& inputDir, const STR_LIST& outputDirs, bool remove)
{
    // get the size of folder list we received
    uint sizeProcessed = outputDirs.size();

    // for each folder in output list
    for (STR_LIST::const_iterator it = outputDirs.begin(); it != outputDirs.end(); ++it)
    {
        QString dirToMoveTo = *it;
        // check if we got 1 folder only to integrate the multimedia data changes into
        // and if we need to remove the input folder as well
        if (1 == sizeProcessed && removeInput)
        {
            // remove before rename if destination exists
            fvaRemoveDirIfEmpty(dirToMoveTo);

            // small optimization - do not copy to last folder if we need to remove the input one - we just rename it.
            QDir dir(dirToMoveTo);

            // move the input folder into output one that is what user selected in UI as Multimedia IR system folder
            if (!dir.rename(inputDir, dirToMoveTo))
            {
                // show message if folder moving has failed
                FVA_MESSAGE_BOX("Fva cmd MoveInputDirToOutputDirs could not rename the dir")
                return FVA_ERROR_CANNOT_MOVE_DIRS;
            }
        }
    }
}
```

Figure 22. FVAFlowController:: MoveInputDirToOutputDirs function implementation

That code:

1. gets the size of folder list we received;
2. checks if we got 1 folder only to integrate the multimedia data changes into and if we need to remove the input folder as well;
3. removes output folder before renaming if destination folder already exists;
4. moves the input folder into output one that is what user selected in UI as Multimedia IR system folder;
5. shows message if folder moving has failed.

Full implementation of FVAFlowController::MoveInputDirToOutputDirs with code description can be found in Appendix D.

4.4.3 The description of Metadata processor code

FVADataProcessor class creates the command line tasks and drives its execution per external requests, for example from the FVAFlowController side.

This class implements "Controller" functions from MVC pattern and Class Factory functions as well. Flow control is based on class FvaConfiguration.

So, it executes such commands as:

- CLTCheckFileFormat;
- CLTCheckDateTime;
- CLTFixEmptyDateTime.

Let's now overview what each command performs.

```
#include "CLTCheckFileFormat.h"
FVA_EXIT_CODE CLTCheckFileFormat::execute(const CLTContext& context)
{
    QString imageFilePrefix;
    Q_FOREACH(QFileInfo info, m_dir.entryInfoList(QDir::NoDotAndDotDot | QDir::System | QDir::Hidden |
    {
        if (info.isDir())
            continue;

        QString suffix = info.suffix().toUpper();
        if (FVA_FS_TYPE_UNKNOWN == fvaConvertFileExt2FileType(suffix))
        {
            LOG_QCRIT << "found not correct file format:" << info.absoluteFilePath();
            return FVA_ERROR_INCORRECT_FILE_FORMAT;
        }
    }
    return FVA_NO_ERROR;
}
```

Figure 23. CLTCheckFileFormat implementation

That code takes each file in a given folder and check its extension. If it finds unknown file type it returns error with code.

```
FVA_EXIT_CODE CLTCheckDateTime::execute(const CLTContext& context)
{
    QString imageFilePrefix;
    Q_FOREACH(QFileInfo info, m_dir.entryInfoList(QDir::NoDotAndDotDot | QDir::System | QDir::Hidden | QDir::AllDirs
    {
        if (info.isDir())
            continue;

        QString suffix = info.suffix().toUpper();
        if (FVA_FS_TYPE_IMG != fvaConvertFileExt2FileType(suffix))
            continue;

        QDateTime DateTime = fvaGetExifDateTimeOriginalFromFile(info.filePath(), m_fmctx.exifDateTime);

        if (!DateTime.isValid() || DateTime.isNull())
        {
            LOG_QCRIT << "found empty exif Date-Time:" << info.absoluteFilePath();
            return FVA_ERROR_NO_EXIF_DATE_TIME;
        }
    }
    return FVA_NO_ERROR;
}
```

Figure 24. CLTCheckDateTime implementation

That code takes each file in a given folder and check its extension. If it finds image file type, it tries to get its DateTimeOriginal from Exif. If time got is not valid or empty it returns error with code.

```

from exif import Image
import os
import sys
import datetime

for filename in os.listdir(sys.argv[1]):
    fullPath = sys.argv[1] + "/" + filename
    fiximage = Image(fullPath)
    if fiximage.has_exif:
        print(filename + " is good")
        print(f"{fiximage.datetime_original}\n")
    else:
        print(filename + " is NOT good")
        with open(fullPath, "rb") as fixfile:
            time = datetime.datetime.fromtimestamp(os.path.getmtime(fullPath))
            print(time)
            fiximage.datetime_original = str(time)
            fiximage.subsec_time_original = '000'

        with open(fullPath, 'wb') as fixedfile:
            fixedfile.write(fiximage.get_file())

```

Figure 25. CLTFixEmptyDateTime implementation

That code takes each file in a given folder and tries to get its Exif metadata. If it does not find its Exif metadata, it tries to put file Modification time into `datetime_original` of Exif.

Conclusion

In the first chapter we provided an overview of multimedia data with its definition and types. Also, we gave the description for Multimedia IR systems, their high-level functions and examples of usage. There is also an example of an average family photo album and the possible photo searching issue. That chapter ends up with a description of Multimedia IR systems usage for corporate segment.

The next chapter starts with a description of the user needs and requirements for the private Multimedia IR systems. Afterwards, it provides the overview and comparison of Google Photo, Mylio and digiKam products as IR Multimedia Systems. Further, it identifies and classifies the issues in the mentioned systems such as identification of inconsistent or not existing photo metadata and suggesting the ways to solve the inconsistency and absence. It finishes up with overview of solutions that address the issues found.

The next chapter describes the User Journey to understand the cadence of the proposed solution in use and the interaction with the user. Basing on issues identified, we formulated the functional and non-functional requirements for the proposed solution. Further, the context diagram is provided to show the proposed solution bounds and its interactions with other objects and subjects. The chapter finishes up with a description of functional decomposition of the system.

The last chapter covers the functionality by the system components and describes the approaches and patterns to address the non-functional requirements. Finally, it describes the integration of IR Multimedia Systems – Google Photo and digiKam – with the proposed solution.

As a result of the development, the architecture can be extended with new components as needed. For example, any formats of the multimedia data can be added as input, and other offline or online IR Multimedia Systems can be easily integrated. In addition, there is a significant potential for expanding the system to use the Cloud to support the flow in scalable way.

In general, the proposed solution confirms the relevance and effectiveness of this architecture, and can be treated as a prototype for further extension and used to service user requests.

Glossary

Table 5. Terms and definitions used

| Term | Definition |
|---------------|---|
| Aperture | is a hole or an opening through which light travels. More specifically, the aperture and focal length of an optical system determine the cone angle of a bundle of rays that come to a focus in the image plane. |
| Cpplint | is a C++ static code analysis tool which looks for programming errors, helps enforcing a coding standard, sniffs for code smells and offers simple refactoring suggestions. |
| digiKam | digiKam is an advanced open-source digital photo management application that runs on Linux, Windows, and MacOS. The application provides a comprehensive set of tools for importing, managing, editing, and sharing photos and raw files. |
| Doxygen | Doxygen is a documentation generator and static analysis tool for software source trees. When used as a documentation generator, Doxygen extracts information from specially-formatted comments within the code. |
| Exif | Exchangeable image file format (officially Exif, according to JEIDA/JEITA/CIPA specifications) is a standard that specifies the formats for images, sound, and ancillary tags used by digital cameras (including smartphones), scanners and other systems handling image and sound files recorded by digital cameras. |
| Geo positions | A geographic coordinate system (GCS) is a coordinate system associated with positions on Earth (geographic position) |
| Google | Google LLC is an American multinational technology company that specializes in Internet-related services and products, which include online advertising technologies, a search engine, cloud computing, software, and hardware. |
| Google Photo | Google Photos is a photo sharing and storage service developed by Google. The service automatically analyzes photos, identifying various visual features and subjects. Users can search for anything in |

| | |
|-----------------------------|---|
| | photos, with the service returning results from three major categories: People, Places, and Things. |
| IR Multimedia Systems | Multimedia information retrieval means the process of searching for and finding multimedia documents; the corresponding research field is concerned with building multimedia search engines. |
| multimedia data | Multimedia in principle means data of more than one medium. It usually refers to data representing multiple types of medium to capture information and experiences related to objects and events. Commonly used forms of data are numbers, alphanumeric, text, images, audio, and video. In common usage, people refer a data set as multimedia only when time-dependent data such as audio and video are involved. |
| Multimedia IR | Multimedia Information Retrieval (MIR) is an organic system made up of Text Retrieval (TR); Visual Retrieval (VR); Video Retrieval (VDR); and Audio Retrieval (AR) systems. So that each type of digital document may be analyzed and searched by the elements of language appropriate to its nature, search criteria must be extended. |
| metadata | Metadata is "data that provides information about other data". In other words, it is "data about data." Many distinct types of metadata exist, including descriptive metadata, structural metadata, administrative metadata, reference metadata and statistical metadata. |
| MVC | Model–view–controller (usually known as MVC) is a software design pattern ^[1] commonly used for developing user interfaces that divides the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user |
| OOP | Object-oriented programming (OOP) is a programming paradigm based on the concept of “objects”, which can contain data and code: data in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods). |

| | |
|-------------------------------|---|
| Operating systems | is system software that manages computer hardware, software resources, and provides common services for computer programs |
| Private Multimedia IR systems | Multimedia IR systems for not corporate users. |
| Pylint | is a Python static code analysis tool which looks for programming errors, helps enforcing a coding standard, sniffs for code smells and offers simple refactoring suggestions. |
| SOA | Service-oriented architecture (SOA) is an architectural style that supports service orientation. ^[1] By consequence, it is as well applied in the field of software design where services are provided to the other components by application components, through a communication protocol over a network. |
| SOLID | In object-oriented computer programming, SOLID is a mnemonic acronym for five design principles intended to make software designs more understandable, flexible, and maintainable. |
| Qt | is a widget toolkit for creating graphical user interfaces as well as cross-platform applications that run on various software and hardware platforms such as Linux, Windows, macOS, Android or embedded systems with little or no change in the underlying codebase while still being a native application with native capabilities and speed. |
| Quality Attribute | is a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders |
| UI | is the space where interactions between humans and machines occur. The goal of this interaction is to allow effective operation and control of the machine from the human end, whilst the machine simultaneously feeds back information that aids the operators' decision-making process |
| User Journey | A user journey is the experiences a person has when interacting with something, typically software. |

Quality Attributes Glossary

Table 5. Software Quality Attribute mentioned

| Quality Attribute | Definition |
|----------------------|--|
| Conceptual Integrity | defines the consistency and coherence of the overall software design. So, it shows the way the components (subsystems, modules) are designed, as well as such factors as the coding style and variable naming. |
| Interoperability | the ability of a software system systems to function well by the exchanging information with other software systems for example third parties. |
| Learnability | a quality of products and interfaces that allows users to quickly become familiar with them and able to make good use of all their features and capabilities |
| Maintainability | the ability of the system to undergo changes with a percent of easiness. So, the changes might impact any part of the system like components, features, the UI once adding or maintaining the software, during the fix for errors, the addressing new requirements. |
| Manageability | describes the easiness for people to manage software, during monitoring a system and for supporting/debugging purpose. |
| Performance | defines how the system can response on a request within a defined time interval. It might be latency or throughput. |
| Portability | the usability of the same software in different environments. The prerequisite for portability is the generalized abstraction between the application logic and system interfaces. When software with the same functionality is produced for several computing platforms, portability is the key issue for development cost reduction. |
| Reliability | shows a level of probability for a system not to fail and to perform during predefined time interval. |

| | |
|----------------|---|
| Reusability | the capability for components and subsystems to be usable in other functionality or other flows. This attribute aim is to deduplicate of implementation, and to reduce the implementation time spent as well. |
| Supportability | the software ability to provide the information to find and to solve the issues during abnormal function. |
| Usability | defines if the software meets the user needs by being intuitive, easy to localize and globalize |

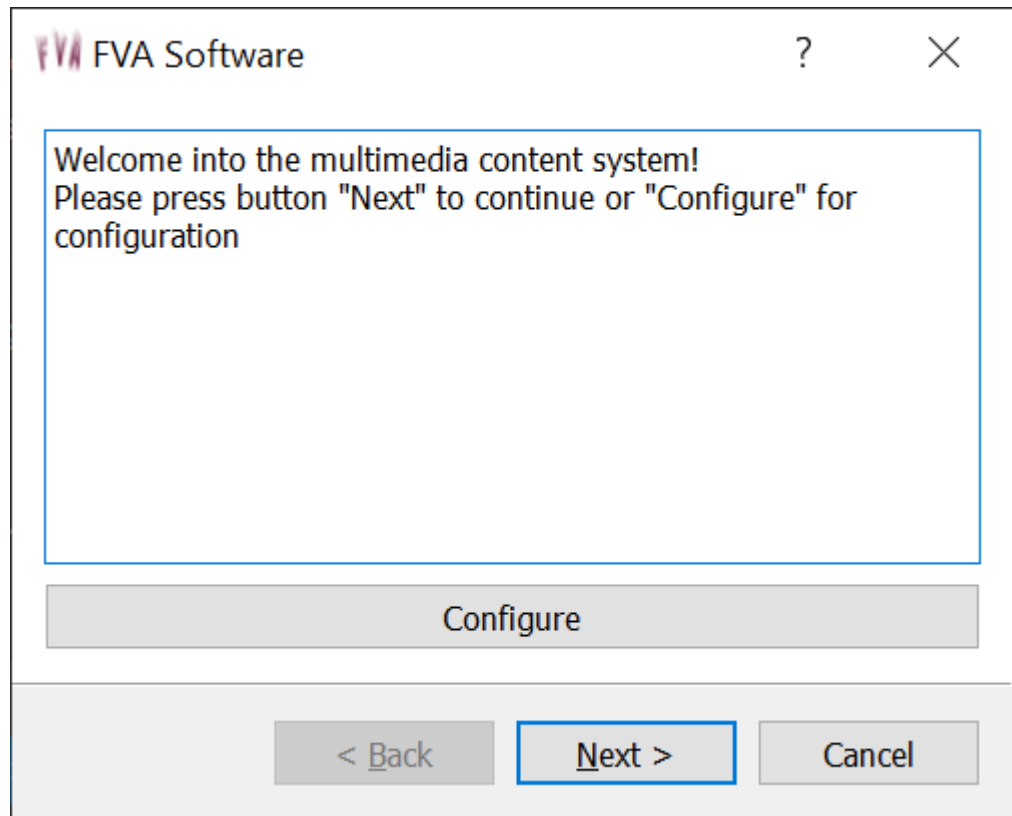
References

1. Multimedia Data [Ramesh Jain]. - [Web] – Online access: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_1008
2. How Many Photos Will Be Taken in 2021? - INSPIRATION/TECH TODAY. – March 11, 2021. – [Web] – Online access: <https://focus.myllo.com/tech-today/how-many-photos-will-be-taken-in-2021>
3. Information retrieval. – [Web] – Online access: https://en.wikipedia.org/wiki/Information_retrieval
4. Multimedia information retrieval. – [Web] – Online access: https://en.wikipedia.org/wiki/Multimedia_information_retrieval
5. Image retrieval. – [Web] – Online access: https://en.wikipedia.org/wiki/Image_retrieval
6. Multimedia Information Retrieval/[Stefan Rüger, The Open University], - 2009. – [Book] - Online access: <https://www.morganclaypool.com/doi/abs/10.2200/S00244ED1V01Y200912ICR010>
7. The digiKam Handbook [The digiKam developers team]. - [Web] – Online access: <https://docs.kde.org/trunk5/en/extragear-graphics/digikam/index.html>
8. Top 25 Photo Organizing Software and Apps of 2020[Boaz Eapen]. - Nov 06, 2019. - [Web] – Online access: <https://www.pixpa.com/blog/photo-organiser>
9. Google Photos. – [Web] – Online access: https://en.wikipedia.org/wiki/Google_Photos
10. Tour Mylio. – [Web] – Online access: <https://mylio.com/tour/>
11. A Beginner's Guide To Google Photos[Sue Waters]. – July 08, 2020. – [Web] – Online access: <https://www.theedublogger.com/google-photos-guide/#:~:text=Google%20Photos%20is%20a%20photo%20sharing%20and%20storage%20service%20devel->

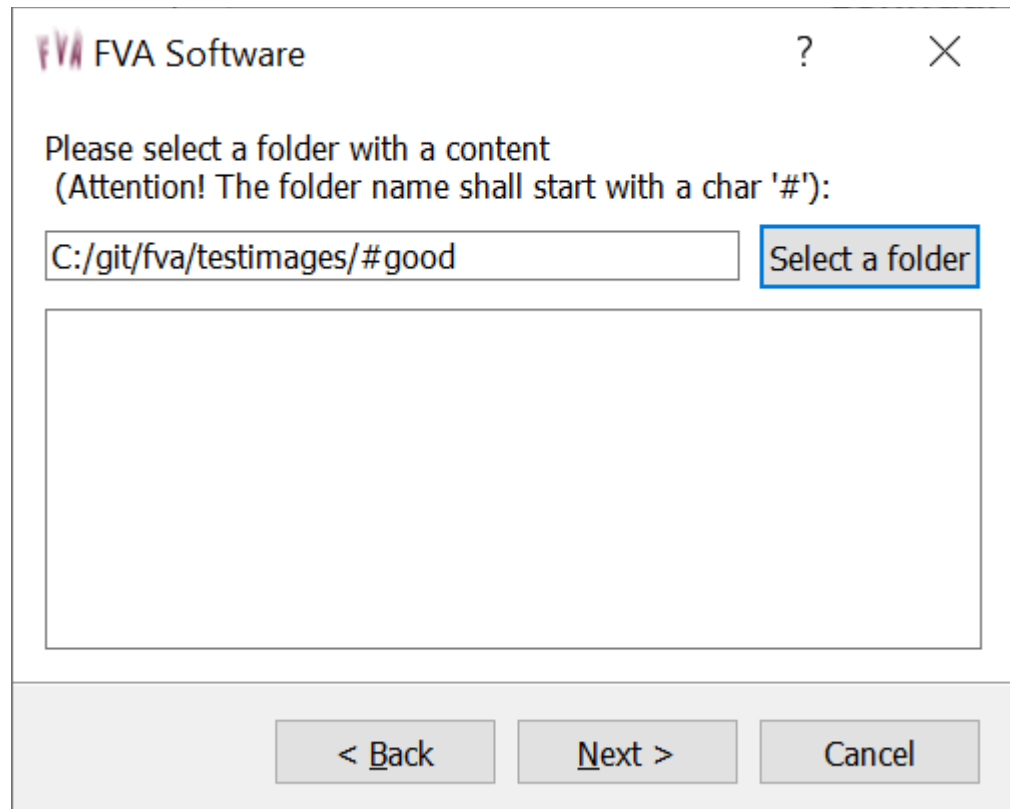
oped%20by%20Google.&text=Google%20Photos%20stores%20your%20photos,all%20your%20photos%20and%20videos.

12. Three Cheers for Embedded Metadata[Lynda Schmitz Funrig]. -February 28, 2012. – [Web] – Online access: <https://siarchives.si.edu/blog/three-cheers-embedded-metadata>
13. Best Practices and Tools to Create Archival Image Metadata[Elena Toffalori]. - June 28, 2016. – [Web] – Online access: <https://digitalarch.org/blog/2017/4/7/ykag6k2fvln7g1j02923n0c7zdrryg>
14. Resolving and avoiding metadata conflicts. – [Web] – Online access: <https://www.lynda.com/Lightroom-tutorials/Resolving-avoiding-metadata-conflicts/447237/485677-4.html>
15. Adding Descriptions to Digital Photos[Mike Ashenfelder]. – October 28, 2011. – [Web] – Online access: <https://blogs.loc.gov/thesignal/2011/10/mission-possible-an-easy-way-to-add-descriptions-to-digital-photos/>
16. Metadata Templates[Peter Krogh]. – [Web] – Online access: <https://dpbest-flow.org/metadata/metadata-templates>
17. Metadata generation for image files. – [Web] – Online access: <https://patents.google.com/patent/US8473525>
18. Google code style. – [Web] – Online access: <https://google.github.io/styleguide/>

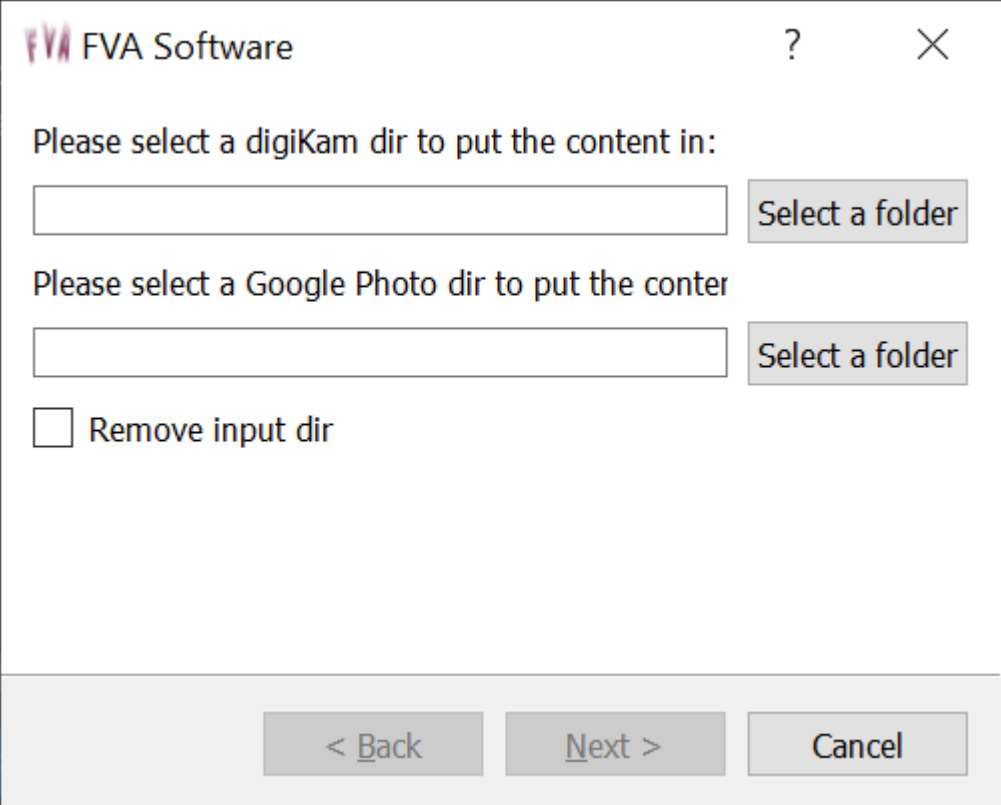
Appendix A. Appearance of the UI screens



Appearance of FVAOrganizerStartPage



Appearance of FVAOrganizerInputDirPage



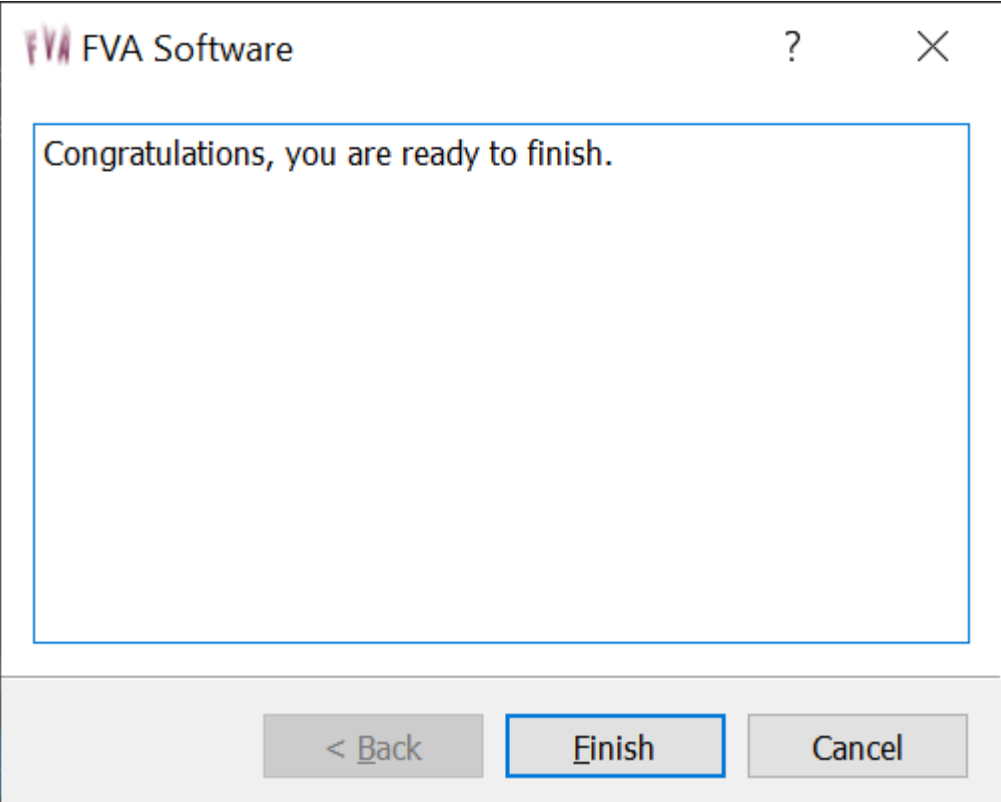
FVA Software

Please select a digiKam dir to put the content in:

Please select a Google Photo dir to put the conter

☐ Remove input dir

Appearance of FVAOrganizerOutputDirPage

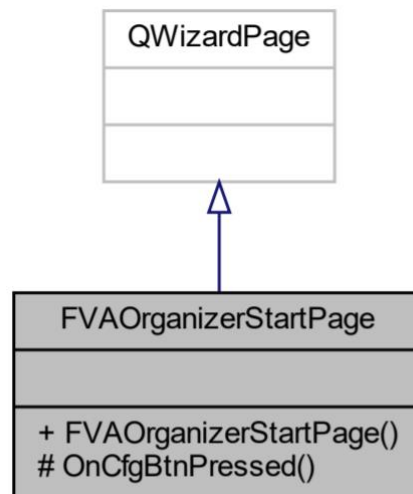


FVA Software

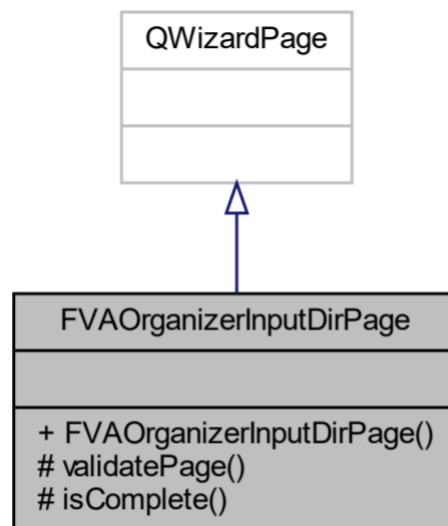
Congratulations, you are ready to finish.

Appearance of FVAOrganizerDonePage.

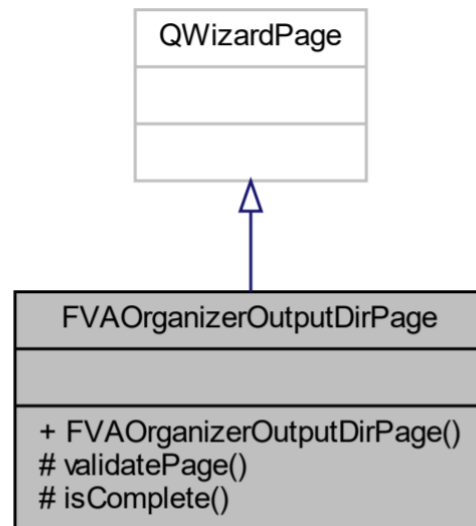
Appendix B. Collaboration/inheritance diagrams



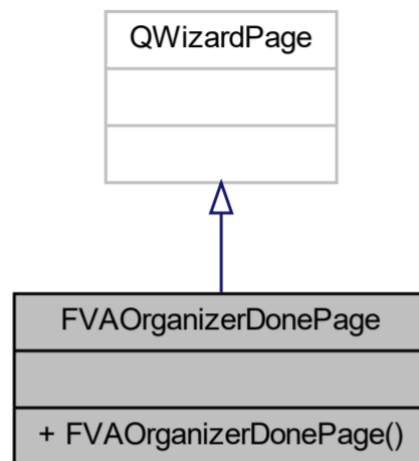
Collaboration/inheritance diagram of FVAOrganizerStartPage



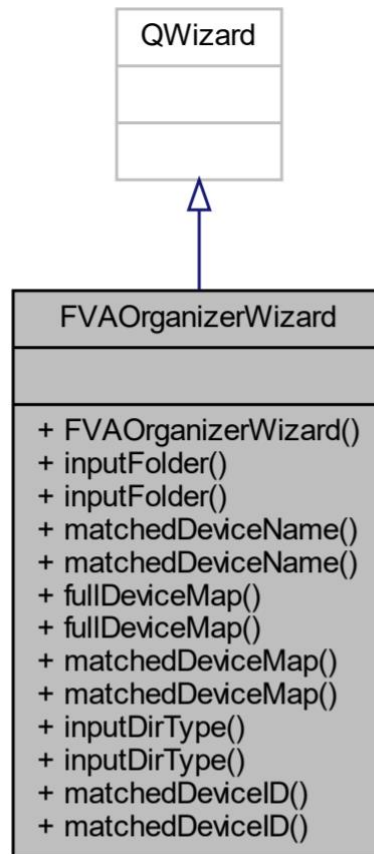
Collaboration/inheritance diagram of FVAOrganizerInputDirPage



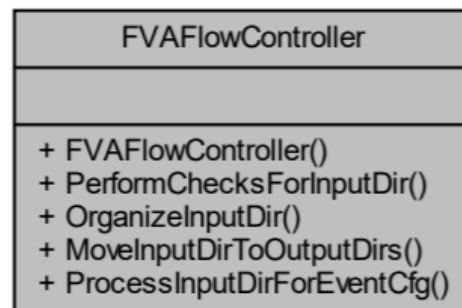
Collaboration/inheritance diagram of FVAOrganizerOutputDirPage



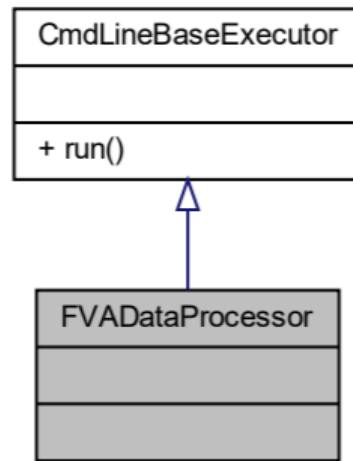
Collaboration/inheritance diagram of FVAOrganizerDonePage



Collaboration/inheritance diagram of FVAOrganizer



Collaboration/inheritance diagram of FVAFlowController



Collaboration/inheritance diagram of FVADDataProcessor

Appendix C. PerformChecksForInputDir function implementation

```
FVA_EXIT_CODE FVAFlowController::PerformChecksForInputDir(const QString& dir, DeviceContext& deviceContext, QObject* obj)
```

```
{
    // create command-line-task context to keep common parameters for all commands
    CLTContext context;

    // set up the dir that user selected in UI
    context.dir = dir;

    // perform common checks
    FVA_EXIT_CODE res = performCommonChecks(context, m_cfg);

    // return to calling function if previous operation failed
    RET_RES_IF_RES_IS_ERROR

    // do we need to search by device?
    bool SearchByDevice = false;

    // ask configuration if we need to search by device
    FVA_EXIT_CODE exitCode = m_cfg.getParamAsBoolean("Search::Device", SearchByDevice);

    // show error message box and return to calling function if previous operation failed
    IF_ERROR_SHOW_MSG_BOX_AND_RET_EXITCODE("getParamAsBoolean(Search::Device)")
    if (SearchByDevice)
    {
        // perform device checks
        FVA_EXIT_CODE res = performDeviceChecks(deviceContext, context, m_cfg);

        // return to calling function if previous operation failed
        RET_RES_IF_RES_IS_ERROR
    }

    // do we need to search by date-time?
    bool SearchByDateTime = false;

    // ask configuration if we need to search by date-time
    exitCode = m_cfg.getParamAsBoolean("Search::DateTime", SearchByDateTime);

    // show error message box and return to calling function if previous operation failed
    IF_ERROR_SHOW_MSG_BOX_AND_RET_EXITCODE("getParamAsBoolean(Search::DateTime)")
    if (SearchByDateTime)
```

```

{
    // perform date-time checks
    FVA_EXIT_CODE res = performDTChecks(context, m_cfg, obj);

    // return to calling function if previous operation failed
    RET_RES_IF_RES_IS_ERROR
}

// do we need to search by location?
bool SearchByLocation = false;

// ask configuration if we need to search by location
exitCode = m_cfg.getParamAsBoolean("Search::Location", SearchByLocation);

// show error message box and return to calling function if previous operation failed
IF_ERROR_SHOW_MSG_BOX_AND_RET_EXITCODE("getParamAsBoolean(Search::Location)")
if (SearchByLocation)
{
    // perform location checks
    FVA_EXIT_CODE res = performLocationChecks(context, m_cfg);

    // return to calling function if previous operation failed
    RET_RES_IF_RES_IS_ERROR
}

// do we need to check photo orientation?
bool needCheckOrientation = false;

// ask configuration if we need to check orientation
exitCode = m_cfg.getParamAsBoolean("Common::CheckOrientation", needCheckOrientation);

// show error message box and return to calling function if previous operation failed
IF_ERROR_SHOW_MSG_BOX_AND_RET_EXITCODE("getParamAsBoolean with Common::CheckOri-
entation")

// do we need to check photo orientation?
if (needCheckOrientation)
{
    // perform orientation checks
    performOrientationChecks(dir,obj);

    return FVA_NO_ERROR;
}

```

Appendix D. MoveInputDirToOutputDirs function implementation

FVA_EXIT_CODE FVAFlowController::MoveInputDirToOutputDirs(const QString& inputDir, const STR_LIST& outputDirs, bool removeInput)

```
{
    // get the size of folder list we received
    uint sizeProcessed = outputDirs.size();

    // for each folder in output list
    for (STR_LIST::const_iterator it = outputDirs.begin(); it != outputDirs.end(); ++it)
    {
        QString dirToMoveTo = *it;
        // check if we got 1 folder only to integrate the multimedia data changes into
        // and if we need to remove the input folder as well
        if (1 == sizeProcessed && removeInput)
        {
            // remove before rename if destination exists
            fvaRemoveDirIfEmpty(dirToMoveTo);

            // small optimization - do not copy to last folder
            // if we need to remove the input one - we just rename it.
            QDir dir(dirToMoveTo);

            // move the input folder into output one that is
            // what user selected in UI as Multimedia IR system folder
            if (!dir.rename(inputDir, dirToMoveTo))
            {
                // show message if folder moving has failed
                FVA_MESSAGE_BOX("Fva cmd MoveInputDirToOutputDirs could not rename the dir")
                return FVA_ERROR_CANT_MOVE_DIR;
            }
        }
        else
        {
            return FVA_ERROR_NOT_IMPLEMENTED;
        }
    }
    return FVA_NO_ERROR;
}
```


Appendix E. performDTChecks function implementation

```

FVA_EXIT_CODE FVAFlowController::performDTChecks(CLTContext& context, const FvaConfiguration& cfg,
QObject* obj)
{
    // prepare context to run CheckDateTime command
    context.cmdType = "CLTCheckDateTime";

    // run CheckDateTime command in Data Processor
    FVA_EXIT_CODE exitCode = m_dataProcessor.run(context, cfg);

    // lets check if Data Processor said there is no exif date time
    if (FVA_ERROR_NO_EXIF_DATE_TIME == exitCode)
    {
        // let's ask configuration if we can fix "no exif date time" issue by picture modification time
        bool fixPicsByModifTime = false;
        exitCode = cfg.getParamAsBoolean("Rename::picsByModifTime", fixPicsByModifTime);

        // show error message box and return to calling function if previous operation failed
        IF_CLT_ERROR_SHOW_MSG_BOX_AND_RET_EXITCODE("cfg.getParamAsBoolean")

        // if we can NOT fix "no exif date time" issue by picture modification time
        if (false == fixPicsByModifTime)
        {
            // show error to user so they are aware what happened
            FVA_MESSAGE_BOX("Found empty date-time metadata, automated fixing is not possible")
            return FVA_ERROR_NOT_IMPLEMENTED;
        }
        else
        {
            // show message to user so they are aware what happened
            FVA_MESSAGE_BOX("Found empty date-time metadata, that will be fixed automatically")
        }

        // run command implemented in python to fix empty date-time issue
        exitCode = runPythonCMD("CLTFixEmptyDateTime.py", obj, cfg, context.dir);

        // show error message box and return to calling function if previous operation failed
        IF_CLT_ERROR_SHOW_MSG_BOX_AND_RET_EXITCODE("CLTFixEmptyDateTime")
    }
    else
        // show error message box and return to calling function if previous operation failed

```

```
        IF_CLT_ERROR_SHOW_MSG_BOX_AND_RET_EXITCODE("CLTCheckDateTime")

    return FVA_NO_ERROR;
}
```