YourBestPet

Evaluation

NESCOL |30082939

Graded unit 2 project

Radosław Rezler

2023

Table of content:

[Introduction 3](#_Toc135911941)

[Adherence to deadlines 3](#_Toc135911942)

[Division into stages 3](#_Toc135911943)

[Project goal 3](#_Toc135911944)

[Methodology/Resources and materials 4](#_Toc135911945)

[Internal documentation 4](#_Toc135911946)

[Technical documentation 4](#_Toc135911947)

[User documentation 4](#_Toc135911948)

[Data binding design 4](#_Toc135911949)

[Visual design 5](#_Toc135911950)

[Menu design 5](#_Toc135911951)

[Amount of quiz questions 5](#_Toc135911952)

[Colours and background 5](#_Toc135911953)

[Division into three sectors 5](#_Toc135911954)

[Disadvenages 6](#_Toc135911955)

[Functional requirements 6](#_Toc135911956)

[Non-functional requirements 6](#_Toc135911957)

[UML diagrams 6](#_Toc135911958)

[Logo and name 6](#_Toc135911959)

[Pseudocode/code 6](#_Toc135911960)

[MoSCoW 7](#_Toc135911961)

[Must 7](#_Toc135911962)

[Should 7](#_Toc135911963)

[Could 7](#_Toc135911964)

[Would 7](#_Toc135911965)

[Testing 7](#_Toc135911966)

[Recommendations for future development 7](#_Toc135911967)

[Better time management 7](#_Toc135911968)

[Creating prototype through planning stage 7](#_Toc135911969)

[More detailed checking of new solutions 8](#_Toc135911970)

[Meeting with client more often 8](#_Toc135911971)

[Object oriented programming 8](#_Toc135911972)

[Gained skills 8](#_Toc135911973)

[Tkinter 8](#_Toc135911974)

[Pillow 8](#_Toc135911975)

[PyInstaller 8](#_Toc135911976)

[Canva 9](#_Toc135911977)

[InstallForge 9](#_Toc135911978)

[Bing Image Creator 9](#_Toc135911979)

[GitHub Copilot 9](#_Toc135911980)

[Conclusion 9](#_Toc135911981)

# Introduction

In this document, I will discuss all the requirements that were constructed in the first phase of the graded unit exam. I will look at each of them individually and assess the compliance of the final product with the initial assumptions. Additionally, I will justify the correctness of all changes introduced and the strengths and weaknesses of my actions. Finally, I will summarize the results of my work and related thoughts/plans for the future.

# Adherence to deadlines

First of all, it should be noted that there were two main assumptions that had to be met in order to properly approach the exam and obtain the highest score. These are, in turn, dividing the creative process into specific stages and submitting each of the completed stages on a specific date. I fulfilled both of these assumptions one hundred percent. This document is the last stage of my project and will be submitted before the deadline expires. All previous stages were also completed before the final date and were submitted within the specified time frame. Additionally, it should be noted that the time in which I completed each stage was shorter than assumed both by my client (North East Scotland College) and myself in the initial documentation - there I attached TomsPlanner with all the dates I set for completing all stages of the project.

I believe that meeting all deadlines was crucial and I am satisfied with my results. I managed to complete each stage ahead of schedule, which gave me the opportunity for additional corrections, time to rest and a stress-free work flow.

# Division into stages

As for the consequences of adhering to the initial assumptions regarding the specific division of stages, I added a very detailed project plan in the initial documentation. As I mentioned earlier, the general division into initial documentation, developing stage, testing, creating a video and evaluation part were completed in a specific order and according to this division. However, each of these stages was additionally divided into smaller ones, presented in the aforementioned project plan. This division was mostly implemented, although some parts of it were modified. In the developing stage, I did not contact the client as many times as I initially assumed because there was no need (now I know it would be a good idea to do it differently, and I will speak about it later on). Additionally, this stage was described very generally and divided into a small number of sub-points, while it consisted of many additional elements such as changing some initial assumptions - creating a new logo and a new design for the visual part of the application, modifying many functions and the general outline of the code, working with my friends for additional analysis etc. These are important stages of production that I will include in my future projects to plan the entire process even more accurately. Additionally, the testing stage was also slightly modified by additional points such as summarizing and justifying specific testing techniques. This is due to adhering to marking guidelines in order to meet all requirements as best as possible. As for the evaluation part, here too I will not strictly adhere to my initial project plan but will work according to marking guidelines for the same reasons.

In summary, dividing the project into parts allowed me to identify the main components of the project and assign appropriate time frames and methods of implementation to each of them. This helped me a lot to find the key points that I have to meet and to monitor my progress and compliance with time on an ongoing basis. I consider these points to be advantages. The disadvantages include underestimating all stages, which made me have to add additional tasks between those already identified. Fortunately, my time management was at a high level and I gave myself extra time anticipating such problems. In the following parts of this document, however, I will mention that the time I ultimately had was just enough - I would not be able to add much more if there was such a need.

# Project goal

In accordance with the initial assumption, the general outline of the project, i.e. creating an application that facilitates the selection of a suitable pet, has been fulfilled. All functions mentioned in the initial documentation have been added and the program works correctly on the Windows operating system. Meeting all assumptions was crucial, so each of them was supervised.

I consider this to be a personal success. It allowed me to increase my self-confidence and significantly expanded my knowledge not only of using many new tools, but above all of managing all stages of such a large project.

# Methodology/Resources and materials

The project was created using all the techniques included in the initial documentation. However, it was expanded by adhering to the marking guideline. This means that I worked according to the assumptions, i.e. using my project plan, MoSCoW, TomsPlanner, UML diagrams, object-oriented programming etc., but I additionally expanded or modified some of these points to obtain the best possible score and make my program more efficient and easier to use. These changes are discussed in detail in the further part of this document. Additional educational and informational materials were also used, as well as python programs/libraries that were not mentioned in the initial documentation, including Canva, ChatGPT, Bing Image Creator, pyinstaller. This happened because I was not able to correctly predict all elements of the project. This influenced the search for additional tools needed for their implementation, such as creating a logo or creating an executable file. Since this is my first such large project, my lack of experience also played a role here. Not having worked with many of the tools used during this project before, I could not include them in the initial documentation because I was not aware of their existence and more specifically - about which tools I would choose to work with and about their main features.

Therefore, the advantages include working with many previously unknown programs and such a wide methodology of work needed to create a project. Thanks to this, I know which methods suit me best and which ones to use depending on the application. I also cannot say that the disadvantage is the lack of experience and the resulting difficulties because it forced me to explore new solutions and discover the applications of many previously unknown methods and new programs/libraries.

# Internal documentation

The code of the program has been commented in accordance with the initial assumptions. This was achieved by commenting on the code on an ongoing basis while creating the application.

This is undoubtedly an advantage of the entire project, as it personally allowed me to easily return to the code after a break and quickly find its various elements, and for other people it will enable quick understanding of the code and the principles of operation of all its elements.

# Technical documentation

The technical documentation contains all the elements required in the project assumptions and marking guidelines. Additionally, it guides the user through the installation process and the internal structure of folders.

That means it is exactly as I planned from the beginning. I am satisfied with the technical documentation and do not see any disadvantages in this regard. The documentation is exhaustive and at the same time clear and addresses all necessary and more complicated issues.

# User documentation

A user manual has been provided for use to facilitate work with the program. It is in line with the initial assumptions and marking guidelines.

Its advantage is its clarity and adaptation to the customer’s needs. It does not contain unnecessary information that could confuse less experienced end-users. The disadvantage, however, is its nice but simple graphic design.

# Data binding design

The project works exactly as presented in the planning stage. All data required for the application to work is stored inside the program (in the code or in the directory where the program is located) and is not editable. The user only interacts with the GUI of the application, which displays only the information intended for them. Additionally, the administrator of the application has the ability to change all parts of the program, which was also noted at the planning stage. However, I must note that the ability to secure all program files does not meet my initial expectations one hundred percent. Although the installer provided to the user creates a program folder in the program files directory and only an icon with a program shortcut is visible on the desktop, so the user has no direct influence on the operation of the program, but if they wanted to - they can go to the aforementioned directory and replace photos with others. However, I do not consider this a mistake and consider the initial assumptions to be met because many programs work in this way - the user cannot edit code or any other data and way of operating applications and a folder with photos is hidden by adding an installer to the program.

However, I classify this feature as a disadvantage because I am convinced that it is possible to create a program in such a way as to better protect all its data from editing. The advantages include compliance with initial assumptions and complexity of working with the program depending on who uses it (administrator or end user).

# Visual design

The general idea has not changed. The program still has a menu and specific functions that show specific data. The way they are presented has not changed significantly either. However, some changes have been made compared to the original project. Before they were introduced, the new project was discussed with the client for their acceptance and to make sure that it does not deviate too much from the initial assumptions. Below I will describe each of the changes along with their justification and the benefits that these changes have brought.

## Menu design

While creating the program, I came to the conclusion that the application window must be larger than I initially assumed. This is because the application presents a lot of information, sometimes long, which must have a lot of space to display correctly and be clear. Therefore, it makes no sense to create a separate menu containing only five options, as it would not look nice. Therefore, I decided to place the menu at the bottom of the application. This change has a positive impact not only on the visual aspect, but also on the ergonomics of the application. Now the user always has access to every option of the program and always sees which one is currently open.

## Amount of quiz questions

During the creation of the program, it turned out that the five questions that were initially assumed to find the best match of a dog/cat for the user are too many. This happened because at the planning stage I was not yet sure what attributes my classes would have and how exactly the algorithm would work. After taking these things into account, at the development stage I created a list of questions that excluded animals that were not suitable for the user and it turned out that the most optimal set would be four questions. It will not narrow down the results too much, but will also allow all the most important parameters to be taken into account. The number of questions needed to select an animal in general - such as dog/cat/rabbit/rat/fish remained unchanged. Now there is no excess of questions, which allows the user to make an easier choice.

## Colours and background

During the creation of the program, I changed the color scheme and removed the patterned background in favor of a solid colour. This allowed me to maintain a minimalist and more transparent style that does not interfere with reading the information selected for the user. The previous background simply interfered with the efficient reading of information and did not look as aesthetically pleasing as I thought. The reason for the incorrect initial estimation of these parameters was a very early stage of planning and the lack of a prototype version that could dispel any doubts.

## Division into three sectors

Since the menu has been added to the bottom of the application, in order to enhance the visual experience associated with using the application, I decided to also add another permanent element - a bar at the top of the screen with a logo. As a result, the program has been divided into three sections, each serving its specific and unchanging function. This increases the ergonomics of the application and its transparency. The user knows where to look for the menu, where to expect information, and constantly sees the logo and name of the application, which reinforces the significance of the brand.

## Disadvenages

Despite choosing a better background and colors, the program may not be legible for people with poor vision or other visual impairments. This was not on my list of priorities, but it would be good to have an additional function that changes the appearance of the application with these people in mind. Additionally, the program looks slightly blurry - this is a problem of the tkinter library, which is no longer the newest and does not work as well with modern monitors. The ability to fix this aspect would require a lot of changes in the code, which would have to check the resolution and scaling of the monitor every time it starts up and then manually assign all values anew depending on the parameters of these devices. However, such a solution would take a lot of additional time, which I no longer had, and I believe that the blur is so small that most users (including my testers) will not notice the difference.

# Functional requirements

The vast majority of functional requirements have remained unchanged. The differences include: three (not two) options lead to quizzes, the menu has five options (not four), and the order of options in the menu has been changed. As you can see, these changes mainly concern the appearance of the menu and its contents and were introduced for better transparency. This change was also previously discussed with the client. Now each quiz has a separate button - so it is accessible with just one click. Quizzes and other options in the menu have been arranged in a logical order to make it easier to use the application. The introduction of changes, as it was consulted with the client, is not a disadvantage.

# Non-functional requirements

All non-functional requirements have been met, in accordance with the initial assumptions. Some of them have been improved - usability, security, time management - which has been described and justified in earlier sections of this document. Meeting all requirements was made possible by their identification at the initial stage of the project and taking them into account during the creation stage, which is an advantage of such a methodology of operation.

# UML diagrams

In accordance with the changes mentioned earlier, all diagrams have been updated. These changes are not significant and refer to everything that has been discussed in this document. The fact that the future administrator of the application will receive updated diagrams is an advantage, as they help to understand the principles of operation of the program and its general idea.

# Logo and name

The name of the application has not changed. However, due to the prototype nature of the first version of the logo, and the fact that the use of previously unknown tools is being evaluated, I decided to create a new logo. The general idea remained unchanged - the logo contains the full name of the program, which makes it easier to remember. In addition, the new version is much more legible and looks more professional.

# Pseudocode/code

The pseudocode and the actual code have been significantly changed. Without even having a prototype version during the planning stage, I was not able to design the code in such detail. An additional complication was the addition of the tkinter library - an extensive part of the code responsible for the visual part of the program. In accordance with the requirements, I used object-oriented programming to create classes and dependencies between them, but I also defined many functions separately. Since the code has grown to over two thousand lines, I did everything I could to maintain its logical consistency and clear internal documentation. Nevertheless, this code differs significantly from the version given in the initial documentation. However, this documentation contains a warning about such a possibility.

Nevertheless, I consider this changes to be a disadvantage - I should have better allocated time for the planning stage to save some extra time for creating a beta version of the program. This would have allowed for better documentation. Additionally, I could have used only object-oriented programming - also for the visual part of the program, if I had more time for it. The advantage, however, is that the entire code has been completed in a way close to the initial idea and that the finished version is fully functional and has no bugs.

# MoSCoW

MoSCoW Generally speaking, all tasks set in the planning stage regarding the Must and Should sections have been met. Additionally, due to the size of the project and lack of extra time, I had to prioritize remaining ideas. Since those from Could and Would did not seem to significantly improve the application, and a new idea - creating an installer for the program seemed to bring more to the final product, I decided to implement it. Therefore, I can say that the general outline of MoSCoW was planned correctly at the planning stage and was extremely useful during the creation of this project.

In summary, among the advantages are meeting all Musts and Shoulds and choosing most priorities well. However, a disadvantage is that I did not foresee everything and added a feature completely not described earlier in the initial documentation.

## Must

All requirements from this section have been met. I changed the layout of the menu, but I did it to improve the application, not to simplify it. These changes have been described in previous parts of the document.

## Should

All additional aspects from this category have been implemented in the final version of the product, as they were previously thought out and constituted an important part of the finished product.

## Could

Due to the size of the project and the client’s requirements (prototype version), none of the points mentioned in the planning stage were implemented. Lack of time and a multitude of other school tasks is an additional reason for this state of affairs.

## Would

Just like in the Could section, no additional features were implemented.

# Testing

The testing phase was completed correctly - all parts of the code were thoroughly tested using many methods. As a result, my program meets all the quality standards set by the client, as well as those specified in the preliminary documentation. Additionally, it allowed me to develop good work practices for the future and to become even more familiar with testing methodologies.

# Recommendations for future development

Taking into account all the advantages and disadvantages of this project, I am able to draw conclusions and improve my future projects. Below I will present changes that I will take into account when working on new tasks.

## Better time management

During the creation of this project, I discovered that many things can go wrong, especially when implementing new, unknown solutions. Although I took such a possibility into account, I underestimated the amount of work that each change brings. The consequence of this was creating a program on time - I didn’t have to stress or chase deadlines, but I also didn’t have extra time that I might need in case of further problems. Better distribution of duties and working every day (which was not the case with this project) will greatly facilitate finding extra time, which will translate into a better final result and its flexibility in relation to many changes and errors that may occur during its creation.

## Creating prototype through planning stage

Some parts of the initial documentation were not fully applied when compared to the finished project. These are pseudocode, MoSCoW, and UML diagrams. For this reason, I believe that it is a good idea to allocate time during the planning stage to create a basic beta version of the program. The initial documentation does not have to fully match the final project, but such a solution will certainly help improve it, and in extreme cases eliminate problems that, if unforeseen, could significantly extend the project or even prevent its creation within a given deadline - working with completely new things can greatly underestimate one’s capabilities.

## More detailed checking of new solutions

I believe that before starting the implementation of the project, I should spend more time finding the right tools for its creation. This is because many elements of the program - in my case especially the interface - could be created more easily if other options than the tkinter library were used. Using it made me learn many useful things - but ultimately I did not work on the most efficient and up-to-date solution. This is another feature to take into account in the future - it is better to use programs that have a future in the context of later employment. This way I do not waste time and energy working on inefficient and non-future solutions.

## Meeting with client more often

I believe that more frequent consultations with the client would make it much easier to prioritize individual elements of the project. This would directly translate into better time management - because I would not waste time on unnecessary functions. Additionally, this way I would have a chance to get a better final grade, as my project would meet all of the client’s requirements. This will be an even more important stage during my professional career - more frequent, but not excessive - contacts with the client will show my commitment, professionalism and reassure the client about the progress of the project and meeting their requirements.

## Object oriented programming

Object-oriented programming is still a fresh topic for me - and although more efficient and transparent - I already know other solutions that also work. Because of this, in many places I did not use object-oriented programming by default, which ultimately made my code not look as professional as it could and did not meet industry standards. For this reason, in every subsequent project, if possible and I consider this solution to be the best - I will use object-oriented programming for whole code. Thanks to this, my code will be better organized, its functions logical and the whole more professional.

# Gained skills

## Tkinter

This is a python library on which its simplified version - guizero - was created. At school, I learned to work with guizero, but as it turned out during the creation of this project, guizero does not have all the necessary functions to create a more advanced interface. As a result, I had to work with the base tkinter library which has more functions and differs significantly in the way the code is written. However, this allowed me to create the program exactly as I wanted. In this way, I gained much greater creative power and also the certainty that from now on I can write programs for windows with a graphical interface that satisfies me. I learned most of the functions that this library provides and operate efficiently on its functions. An additional advantage is that I can connect my view model with the business model - so my graphical interface handles all the functions of the program that I want to put in it and displays their results correctly. This is an extremely useful skill, because even less experienced users can now use programs written by me.

## Pillow

I learned the basic functions of this library, so now I can upload photos in full resolution, not cropping them beforehand, and perform all their processing inside the program in the form of code, loading them, rescaling them and placing them in specific places.

## PyInstaller

This is a new library for me. Thanks to it, I can now say that I can create an executable file from any of my programs written in python. I can create both an executable file of a program with a graphical interface and one that only displays a console or even one that, when clicked on its icon, does not display anything but only runs a previously programmed procedure.

## Canva

By learning how to use this program, I learned how to create simple but professional and eye-pleasing logos. This will help me create any brand that comes to mind in the future. I can generate an image with high quality, without a background and suitable for programs/websites/documents. Since I have both a tablet and a phone with a stylus, I am able to create an image that combines with my logo so that everything is as close to my imagination as possible.

## InstallForge

Having the ability to use this program, I am able to generate a windows installation wizard, so well-known to users of this system. Thanks to this, as well as previously described skills, by combining everything into one whole, I learned how to create a graphical program with a nice, transparent interface, having an .exe extension which now additionally the user is able to install in the same way as all other programs on their computer. This reduces the possibility of errors to a minimum and allows any windows user to operate my programs. The installer has all the features of a professional product - it has its own icon and photos of the installed program, uninstall file, creates a directory in the folder chosen by me/the user and creates a shortcut on the desktop/start menu.

## Bing Image Creator

Among many available image generators on the market, I chose for myself one that gives the most possibilities without requiring additional funds. I learned how to write prompts properly in order to achieve the expected results. This will allow me to diversify all my future projects, including programs, websites and presentations.

## GitHub Copilot

I consider this skill to be one of the key ones considering what world we are entering. As it seems that entering the IT industry as a junior developer is now becoming more difficult due to all artificial intelligence helping in programming and soon may be able to replace junior positions, it is very important to keep up with the newly created market and use its features as my advantage. Only in this way will I be able to keep up with new technology and employer requirements regarding skills and time required to create their orders. During the graded unit project, I learned how to work with GitHub Copilot, an artificial intelligence that allows me to integrate its functions with my code writing program. This enabled efficient writing of repetitive parts of the code, as well as faster creation of internal documentation.

## Conclusion

During work on this project, I primarily learned how to work on a large, complex task that requires planning, time and resource management, creativity, professionalism and a range of necessary skills from me. These skills, which I also had to acquire during work, are described above and combined into a whole, allowed me to create a smoothly functioning program from scratch, in line with my vision (both in terms of functions and visual aspects) and user-friendly. Additionally, I now know how to create technical documentation and how to improve the entire process during my future projects, and I have improved my use of the python language by learning new shortcuts and methods for achieving the required results. I also feel much more confident in the IT market, believing in my skills and being motivated to create more big, meaningful projects.