FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	A tool for improving focus on tasks
Student:	Dinh Tu Hoang
Supervisor:	Ing. Jiří Chludil
Study program:	Informatics
Branch / specialization:	Web and Software Engineering, specialization Software Engineering
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2024/2025

Instructions

The goal of this thesis is to create a program that improves focus and prevents procrastination during work.
This application implements the "Pomodoro technique". It allows users to block selected websites based on user-defined rules, such as blocking during working sessions and unblocking during breaks. It also includes a To-Do list, where tasks can be marked to focus on them. Lastly, gamification elements will be implemented, such as achievement, experience points, and coins.

1. Analyse existing solutions like Cold Turkey, and Habitica, and their interfaces.
2. Perform analysis of potential users (like students) and define user requirements.
3. Design a prototype of a user-friendly interface.
4. Implement a prototype of the desktop application and web extension.
5. Subject the solution to suitable tests: (unit test, integration test, End-to-End (E2E) test, user test)

Electronically approved by Ing. Michal Valenta, Ph.D. on 20 September 2023 in Prague.

Bachelor's thesis

A SYSTEM FOR PRODUCTIVITY IMPROVEMENT

Dinh Tu Hoang

Faculty of Information Technology
Katedra softwarového inženýrství
Supervisor: Ing. Jiří Chludil
January 10, 2024

Contents

List of Figures

List of code listings

I would like to thank the supervisor of the work, Ing. Jiří Chludil, for his support and professional advice during the writing of this thesis. Special thanks to my family and friends for their support and motivation during my studies and the writing of this thesis.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis. I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Czech Technical University in Prague has the right to conclude a licence agreement on the utilization of this thesis as a school work pursuant of Section 60 (1) of the Act.

In Prague on January 10, 2024 .....

Abstract

The aim of this thesis is to create a desktop application for the Windows operating system that improves concentration and prevents procrastination while working. The thesis includes an analysis of competing applications and potential users. The thesis also includes the design of the user interface and application. The application has been implemented using C# .NET with MVVM architecture. The resulting application from this thesis includes a Pomodoro timer, a to-do list and a feature to block websites or applications with gamification features. The main benefit of the application is to improve the user's concentration, which enables the user to work on the task more efficiently. The application has been user tested and the results show that 66.7 % of the users find the application intuitive and 85.7 % of the users would use the application.

Keywords Pomodoro timer, increase work productivity, avoiding procrastination, web blocking, to-do list, gamification elements, C#, WPF, MVVM

Abstrakt

Cílem této práce je vytvořit desktopovou aplikaci pro operační systém Windows, která zlepšuje soustředění a zabraňuje prokrastinaci při práci. Práce obsahuje analýzu konkurenčních aplikací a potenciálních uživatelů. Součástí práce je také návrh uživatelského rozhraní a aplikace. Aplikace byla implementována v jazyce C# .NET pomocí architektury MVVM. V práci vytvořená aplikace obsahuje Pomodoro časovač, seznam úkolů, funkce pro blokování webových stránek nebo aplikací a gamifikační prvky. Hlavním přínosem aplikace je zlepšení soustředění uživatele, což uživateli umožní efektivněji pracovat na úkolu. Aplikace je uživatelsky otestována a výsledky ukazují, že 66,7 % uživatelů považuje aplikaci za intuitivní a 85,7 % uživatelů by aplikaci používalo.

Klíčová slova Pomodoro časovač, produktivita práce, zabránění prokrastinace, blokování webů, seznam úkolů, gamifikační prvky, C#, WPF, MVVM

List of abbreviations

CTU	Czech Technical University
DTO	Data Transfer Object
EF	Entity Framework
FIT	Faculty of Information Technology
IP	Internet Protocol
MBF	Manual Blocking Function
MVVM	Model-View-ViewModel
PBS	Pomodoro Break Session
PS	Pomodoro Session
PT	Pomodoro Timer
PWS	Pomodoro Work Session
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language
XML	Extensible Markup Language

Introduction

Most people have always struggled with procrastination. When people procrastinate, they usually do something irrelevant instead of working on their tasks. As a result, people do not complete their tasks, which can be a cause of failure.

Today, the fight against procrastination can be even more difficult. Many people work on computers to improve their efficiency. The problem is that there are many sources of distraction on the computer. These sources can come from social media, messaging applications, video games, emails, and video platforms. Once people are distracted by these sources, they can spend hours and hours on these platforms. The main goal of this thesis is to create an application that improves concentration and prevents procrastination while working on the computer.

The author chose this topic because he also has a problem with procrastination, and the problem has not been solved satisfactorily. For example, many existing solutions can block distracting applications and websites, but they do not provide a to-do list to help the user see what needs to be done, or a Pomodoro timer that can help the user to improve their concentration.

The analysis part begins by clarifying what procrastination and the Pomodoro technique are. Then, continues with an analysis of the existing solution and potential users, functional and non-functional requirements. The next part is about design. It discusses use cases, available technologies, architectures, and the user interface. In the implementation part, readers can learn about selected technologies, project structures, and implementation of key features. The testing part introduces different types of tests that are used to test the application.

Goals

The goal of this thesis is to create a desktop application for the Windows operating system that improves concentration and prevents procrastination while working. It is useful for users who often use a computer for study or work purposes. The application blocks distracting applications and websites and uses the Pomodoro technique.

The first step is to analyse existing solutions and their interfaces. These solutions include Cold Turkey, Pomofocus, and Habitica. This is followed by an analysis of potential users and the definition of user requirements. After the analysis, a prototype of a user-friendly interface will be designed. The next goal is to implement a prototype of the desktop application and the web extension. The final step is to test the solution with unit tests and usability tests.

Chapter 1
Analysis

This chapter is about the analysis of existing solutions, potential users, functional and non-functional requirements. There are many existing solutions. It is necessary to find out what these solutions can do. What their advantages and disadvantages are and why these solutions are not enough. Requirements analysis also involves identifying potential users and conducting a survey to understand their needs. This is followed by functional and non-functional requirements.

In the first place, the questions of what are Pomodoro technique, To-do list, and procrastination need to be answered.

1.1 Procrastination

What is procrastination? "Procrastination is not laziness, but the inability to talk yourself into doing the activities we should or would like to do." [1]

When someone procrastinates, they cannot convince themselves to do the things they should or want to do. Instead, they do something irrelevant like watching TV shows, browsing social media, playing computer games, watching meaningless videos. Procrastination is followed by remorse, leading to feelings of helplessness. This keeps the cycle of procrastination alive... [1]

Procrastination is not laziness. A lazy person does not want to do anything and is satisfied with that. People who procrastinate want to do something, but cannot bring themselves to do it. Procrastination is not rest because in rest people gain energy. In procrastination, they lose their energy. The less energy people have, the more likely they are to put off their tasks and procrastinate. [1]

Procrastination causes people to waste time that could be invested in something more meaningful.

1.2 Pomodoro technique

One of the techniques that can help prevent procrastination is the Pomodoro technique.

What is the Pomodoro technique

The Pomodoro technique is a time management method developed by Francesco Cirillo. This technique aims to provide a simple tool/process to improve productivity. This technique can, according to [2], also:

- Increase and maintain motivation
- Reduce distractions
- Improve attention and concentration
- Improve work time estimation

How it works

A Pomodoro session (PS) consists of a Pomodoro work session (PWS) and a Pomodoro break session (PBS). A typical PS lasts for 30 minutes. 25 minutes is work time and 5 minutes is a break. After four PWS, there is a long break. The break can last between 15–30 minutes, depending on the user.

In general, PWS cannot be interrupted. It is 25 minutes of uninterrupted intensive work without distractions. A PS cannot be divided into smaller parts. If the user is interrupted by someone or something, the PWS is invalid, and the user must start a new one.

PWS is followed by PBS. This can be a long or short break, depending on how many PWS the user has completed. The 5 minute short break helps the user to absorb and process what they have learnt during the PWS. During this break, the user can drink some water, get up and walk around, do some stretching, or something else. The long break is an opportunity for the user to take a walk, clean the desk, wash the dishes, ...

In breaks, it is not recommended to perform mentally demanding activities. These activities can be like writing an email or making an important phone call. [2]

Based on the experience of the thesis writer, 25 minutes of PWS is not always a good option. Some tasks require deep concentration over a longer period of time, e.g. programming. During the break, it is also not recommended to browse social media, because the new information on these platforms can fatigue the brain.

1.3 Exists solutions

There are many solutions on the market for different platforms that can help users improve their concentration and reduce procrastination. These solutions can provide Pomodoro timers, to-do lists, website blocking, and gamification features. The problem is that in most cases, these solutions do not implement all of the listed features.

The following subsections provide an analysis of the selected solutions based on the author's experience. Analysis is primarily focused on solutions that support the Windows platform. This is because the solution of this thesis is for the Windows platform. At the time of writing, these applications are free or offer a free version.

1.3.1 Cold Turkey

Cold Turkey [3] is a desktop application that blocks websites, games, and applications available for Windows and macOS. The free version of the application only offers website blocking and website exceptions. The user can start blocking manually by clicking the "on" toggle. The user can also set a fixed time interval to block selected websites. The free version of the application also provides statistics on blocked websites and the time spent on each website.

The Cold Turkey interface is a little complicated, especially for new users. The section for adding or setting up site blockers is a little confusing. There are so many buttons that it is not obvious at first glance. Users have to try out what each button does to find out how to add sites to block and rules. Figure 1.1 shows a screenshot of the application for the "Blocks" section.

The paid version has more features. This version allows the user to block applications and games. The user can schedule the blocking by creating a blocking schedule. Pauses are also

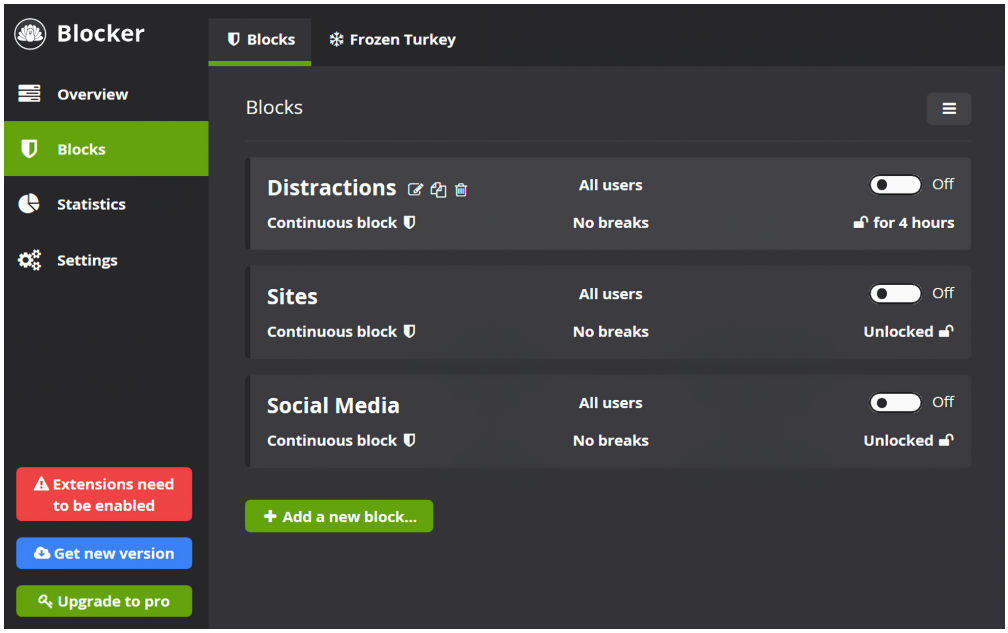

Figure 1.1 Cold Turkey screenshot – Blocks section

available, and the user can define their own rules. The Pomodoro timer is also available and all of the blocks are automatically unblocked during the break session.

Users can set rules to prevent them from turning off the blocking feature while the application is blocking applications and websites. There are many rules to choose from. Some are very strict. They can include typing random characters or even restarting the computer to unblock.

Cold Turkey uses web extension to block websites. The web extension communicate with the desktop application.

Advantages

- The application offers many blocking rules. Some of the rules are very strict to prevent the user from overriding the blocking.
- Other blocking profiles can be defined by the user.
  - For example, you can choose to block social media in the morning and games in the evening.

Disadvantages

- Most features are only available in the paid version, such as application blocking and the Pomodoro timer.
- Support for the Pomodoro timer is also very poor. The user can only set a work time and a break time, without a long break time. Figure 1.2 shows the Pomodoro timer settings.
- There are no timer presets, which means that users must manually change the timer if they need to.
- Viewing and controlling the timer is also limited, as the Pomodoro timer is only used here to define when to block or allow applications and websites.
- The to-do list is not supported here either.

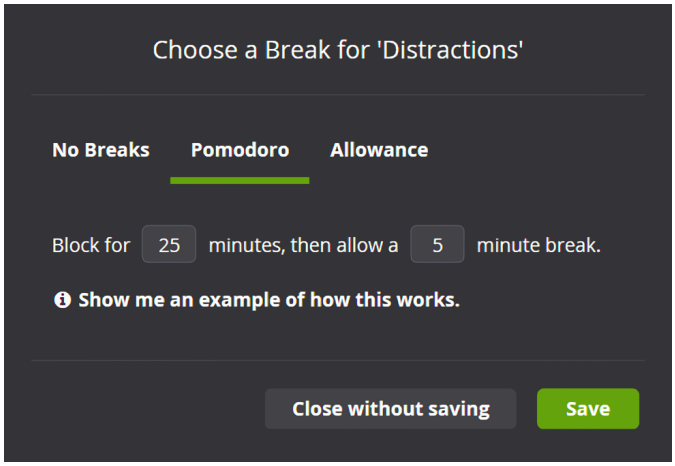

Figure 1.2 Cold Turkey screenshot – Pomodoro timer setting

1.3.2 Pomofocus

Pomofocus, as described in [4], is a customisable Pomodoro timer that works on desktop and mobile browsers. The user can personalise work/break time, and more. The aim of this application is to help users focus on any task they are working on, such as studying, writing, or coding.

Pomofocus keeps track of the user's total work time and displays it as a statistic. The application provides a to-do list with templates. Users can create tasks and save them as templates. A task can be highlighted in order to prioritise it.

The interface of Pomofocus is very user-friendly, because it is simple and minimalist. When users enter Pomofocus, all they have to do is click on the start button to launch the PWS. The addition of a new task is also easy to do. The only thing that can be a bit chaotic at first sight is the PT setting, as it offers many settings. Figure 1.3 shows a screenshot of the application.

The application has a free plan and a premium plan. The premium plan offers additional features such as unlimited to-do list templates, more advanced statistics, no ads, integration with third-party applications, and more.

Advantages

- The free version offers key features without major limitations.
- There are task templates so that users do not have to enter the same recurring task into the to-do list over and over again.
- A task can also be highlighted.
- The interface is simple and minimalistic.

Disadvantages

- Pomofocus does not offer timer presets, which means users must manually change the timer if necessary.
- There is no blocking feature, which means that users cannot block distracting websites/applications while in the working session.

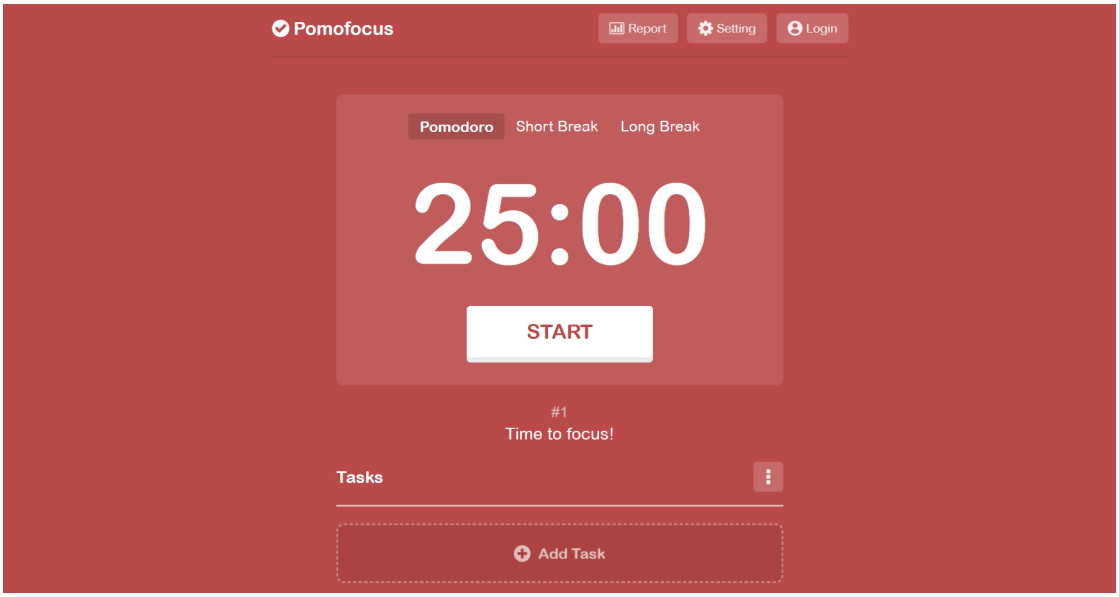

Figure 1.3 Pomofocus screenshot – Homepage

1.3.3 Habitica

Habitica, as described in [5], is a habit-building and productivity application with a to-do list. It uses retro RPG elements to gamify user tasks and goals. The application is available on the Web platform for desktop users. Habitica also supports mobile users. It can be downloaded on both Android and iOS. The application is popular with 4 millions users.

The gamification feature is really well developed to motivate and inspire users. There are in-game rewards, punishments, monster fights with friends, inventory, shops, health points, experience points, and more. Users earn coins when they complete a task or habit before the deadline. Users lose health points when they fail a task or habit. Coins can be used to buy items. These items can then be used to fight some monsters.

As mentioned above, Habitica has many gamification features. As a result, Habitica's interface contains a lot of information, buttons, and navigation. However, everything is logically categorised so that it is not too complicated for the user. Figure 1.4 shows a screenshot of the application.

Habitica is free to use. However, there are some fees in the application. Some of the items in the gamification system can only be obtained by paying real money. But, in general, users can use the habit tracker and the to-do list without any restrictions.

Advantages

- The gamification system is very well developed.
- This feature keeps users motivated to do their tasks and habits.

Disadvantages

- Pomodoro timers are not supported by Habitica.
- It does not offer the ability to block websites/applications to help users focus on their tasks.
- Instead of doing their tasks, users may procrastinate on gamification features.

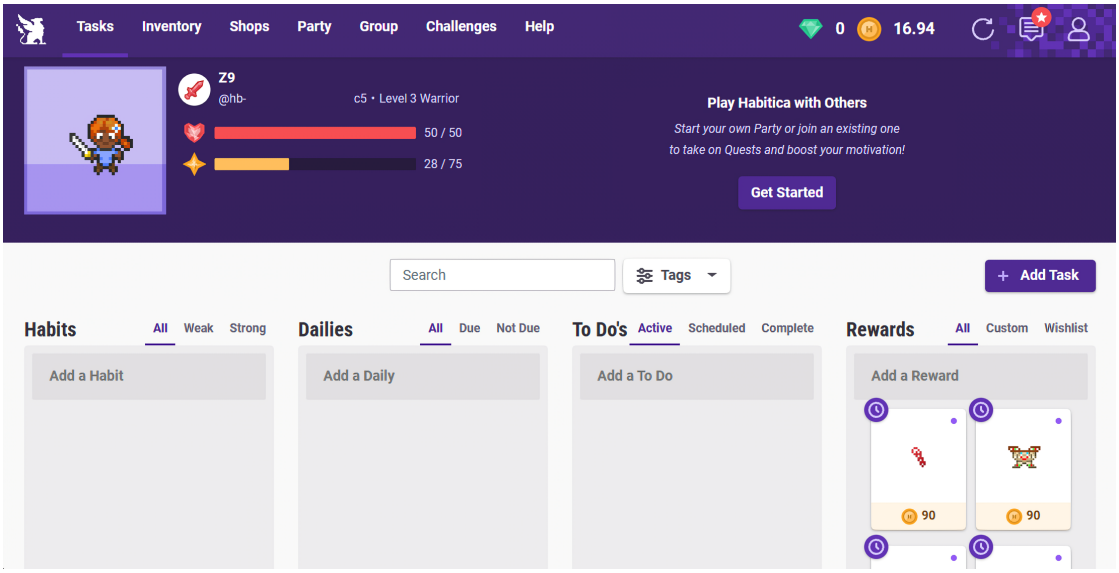

Figure 1.4 Habitica screenshot – Homepage

1.3.4 Other solutions

There are other solutions as well. Applications with blocking features include Freedom, StayFocusd, Unhook, and others. Other solutions with a timer feature can be the focus timer directly in the Windows Clock application.

Freedom can block websites and applications to protect users from distractions. Freedom allows users to manage application/website blocking across multiple devices in one place. Freedom is like Cold Turkey. There is no to-do list. The blocking rules are more limited than in Cold Turkey. Furthermore, Freedom is not free. It only offers a trial version. After that, users must pay a fee to continue using it. [6]

StayFocusd is a free productivity extension for Google Chrome that helps users stay focused on work by restricting the amount of time users can spend on time-wasting websites [7]. StayFocusd does not have a Pomodoro timer feature. There is no to-do list.

Windows Clock provides users with a feature called "Focus sessions" [8], which is similar to the Pomodoro Timer. This application has integration with Microsoft To Do. Microsoft To Do is a cross-platform task management application [9]. In the Windows Clock, a task can be highlighted for the current focus session. However, the application does not provide any application/website blocking features. The timer adjustments of "Focus Sessions" are limited.

Unhook is a web extension that can be used to block YouTube video suggestions, comments, and more [10]. This helps users not get distracted if they need to find some tutorials on YouTube while they are working/studying. This extension only blocks selected YouTube elements. It has no other features.

1.3.5 Conclusion

The aim of this work is to help users avoid procrastination, improve concentration on tasks, and motivate them to complete their tasks. Existing solutions offer decent functionality. However, none of them provide all of the required features together. These features are the Pomodoro

timer, to-do list, website blocker, and gamification elements. Some of these solutions only offer a trial version, after which users must pay a fee to continue using it without restrictions.

Cold Turkey it is a power blocker with many rules and customisations for users. To use all the features (including the Pomodoro Timer), users must purchase the Pro version. There is no to-do list. [3]

Pomofocus is a customisable Pomodoro timer with a to-do list. Pomofocus is very easy to use. Timer presets and blocking features are not offered in this application. [4]

Habitica is a habit-building and productivity application with a to-do list. Pomodoro timer and blocking features are not available in Habitica. The gamification system may distract some users instead of motivating them as it is very detailed. [5]

Apart from the statistics feature, there is usually nothing big to motivate users in applications without gamification elements. Some users may stop using the application. As a result, they may start procrastinating again. Gamification elements can solve this problem by rewarding the user after completing the task. The gamification elements should be simple enough to motivate users. If it becomes too robust, it could distract users instead of helping them. Statistics allow users to see their progress. This can motivate users to improve.

The Pomodoro timer reminds users to take breaks to recover so they can focus on the next work session [2]. The preset function of the Pomodoro timer is a great feature to have. The user does not have to change the timer setting every time. They can switch to other settings more quickly.

The to-do list helps users organise their tasks. Users get an overview of what needs to be done. The task focus feature is a good way to keep the user focused on an important task.

The website blocker prevents users from being distracted by sites such as social media or messaging sites. The Application Blocker prevents users from being distracted by games and other entertainment applications. The YouTube blocker blocks recommended YouTube videos on the YouTube homepage. This can be useful for users who need to find some YouTube tutorials without being distracted by entertainment videos.

In summary, the final product of this work aims to have these features:

- Pomodoro timer with timer presets
- To-do list with task highlight
- Gamification elements
- Statistics
- Websites blocker
- YouTube blocker
- Application blocker

1.4 User analysis

The analysis of existing solutions is followed by an analysis of potential users. The first part of this section is about potential users. The second part deals with the analysis of the users and the results. The analysis was carried out using questionnaires. A total of 54 people participated in the questionnaire.

1.4.1 Target group

Procrastination in general is a big problem for many people. The target group are users who need to use computer devices for study or work purposes. This target group has the highest potential to be distracted when using these devices. This distraction can lead to procrastination. The distraction can come from direct messaging applications, social media, email, or streaming platforms. Even without distractions, users can still open some of the platforms mentioned above and spend hours on them instead of working.

The author's friends and colleagues at work, and some other students in Faculty of Information Technology CTU in Prague (FIT) usually get distracted while working or studying for many reasons that lead to procrastination. This is why the author chose them as a focus group for the survey.

1.4.2 User questionnaire

As mentioned above, user surveys are conducted using questionnaire included in the enclosed medium. The questionnaire was anonymous.

The purpose of the questionnaire was to obtain as much information from the users as possible. However, the questionnaire had to be short and easy to answer. The questions were multiple choice, and users could also add their own answers. This can increase the success rate of completing the questionnaire.

The main aim of the questionnaire was to find out:

- What operating systems do they use?
- How long do they work on the computer?
- What are their main sources of distraction?
- How often do they procrastinate while working on the computer?
- What features do they need?
  - What to block (websites, applications, YouTube recommended videos)
  - When to block (manually, during Pomodoro work session)
  - Gamification and statistics

1.4.3 Results

A total of 54 people participated in the questionnaire. The questionnaire was published in the following way. For FIT students, it was published on social media platforms where there was a group of these students. In addition, the author sent the questionnaire to all his friends who use computers for study and work. Finally, the questionnaire was distributed personally by the author through his work and school.

Here are some of the key questions and answers from the user questionnaire:

- What operating system do you use?

  The purpose of this question is to find out which platforms users are using for the selection of platforms and programming languages for the solution.

  The answers were multiple choice: were multiple choice: Windows, macOS, Linux, Android, and iOS. The result is mixed. But the Windows platform is a significant part of it. About 85 % of the respondents use the Windows operating system. It is a good idea to make the solution multi-platform. However, this can be more complex, so the focus is on one specific platform. The target platform for the application will be Windows.

- How much time a day do you spend working/studying on computers?

  This question was designed to find out whether many users spend a lot of time working or studying on the computer. The more time they spend on the computer, the more likely they are to procrastinate.

  The result shows that all respondents work or study on the computer. 63 % spend more than 6 hours a day on the computer. 20.4 % spend between 3 and 6 hours.

- What are the most common sources of distraction when you are working?

  The purpose of this multiple choice question is to identify common sources of distraction that can lead to procrastination. Identifying these sources helps create a tool to block these sources if possible.

  72.2 % of the participants chose instant messaging applications (Messenger, Discord, ...). 55.6 % chose social media (Facebook, Instagram, Twitter, ...). Other sources of distraction include streaming platforms, email, and video games.

- How many hours a day do you spend on the above mentioned platforms instead of working?

  The aim of this question is to find out how much time participants spend on these platforms instead of working or studying. In other words, how much time participants procrastinate on these platforms.

  The result shows that all participants spent at least less than 1 hour on these platforms. 18.5 % spent more than 3 hours.

  According to these results, it is necessary to have a website blocker because a significant part of the distraction comes from web applications. An application blocker is nice to have. Only a small proportion of participants are distracted by video games. A notification blocker can be replaced by a do not disturb mode built in the system.

- What other features do you need?

  As there is a lot of information in one question, the question about features required by users was split into several simple questions. These questions aim to find out if users need other features such as a to-do list, gamification elements, statistics, application and notification blockers, and others.

  These results show that only 11.1 % of the users do not use a to-do list to manage their tasks. The rest use a to-do list or would like to use one in the future. The to-do list is a required feature because many users need it.

  Only 20.4 % of the users would like to add gamification elements. More than 50 % of the users need to see some statistics, such as how much time they have worked. Statistics can be seen as part of the gamification elements. According to these results, the gamification elements should be simple, since only 20.4 % of the users want them and others do not need them. It should also be possible to disable gamification features that may affect users who do not want to use them.

  More than 60 % of the users need the ability to block notifications and applications. About 47 % of the users require to hide YouTube thumbnails to avoid distracting videos. Approximately 53 % of the users want to change their Discord[1] status to "do not disturb" during PWS. As these features are not part of the thesis assignment, they have a lower priority and are not guaranteed in the application.

[1] "Discord is a voice, video and text communication service used by over a hundred million people to hang out and talk with their friends and communities." [11]

1.5 Author's experience from study in FIT CTU

During Team Software Project 1 (BI-SP1), Team Software Project 2 (BI-SP2), and other courses at the Faculty of Information Technology CTU in Prague, the author gained experience and knowledge. With these experiences, the author decided that it would be great to consider the following areas as well. These are security, configuration, and user-friendly interface.

It is appropriate to focus on securing applications to prevent unauthorised access and application exceptions or unwanted behaviour that can damage the system. Unauthorised access can result in the loss of sensitive user data, which can lead to financial loss, and perhaps more.

The use of configuration files is recommended. This allows application parameters, such as settings, to be changed in one place without modifying the source code, giving more flexibility. It can also be useful for testing.

Simple user interfaces can have a positive impact on the use of applications, and users will continue to use the application. However, a complex interface can have a negative impact because it is too difficult for them to navigate to the things they need. Some users may give up and stop using the application.

1.6 Functional requirement

This section discusses the functional requirements that were identified based on the assignment of the bachelor's thesis, the potential user questionnaire, and the analysis of existing solutions. For ease of understanding, the functional requirements are divided into sections: Pomodoro timer, to-do list, blocking management, gamification elements, general settings, and other requirements. Each of the functional requirements will have a description and its own priority.

High – The feature is required

Medium – Nice to have, but not important.

Low – The feature is not required.

Pomodoro timer (PT)

These functional requirements describe what the user can do with the Pomodoro timer and how it relates to blocking.

F1 Start Pomodoro timer PT can be initiated by the user when the work/break session has finished or has not yet started. The user can also resume PT when the PT is in a break session or has paused. This will skip the current break session and go to the next work session, or end the break timer and resume the work session.

priority: High

F2 Restart Pomodoro timer The user can usually restart the Pomodoro Timer in all cases. The user cannot restart the timer if the application has just started (because the timer has not yet started), or the user has just restarted the timer.

priority: High

F3 Pause Pomodoro timer If the PS has already started, the PT can be paused by the user. The timer can be paused indefinitely or for a specific period of time. If the user pauses the timer for a certain period of time, it is possible to unblock sites or applications that are blocked due to other settings. However, unblocking for a specific time will cost the user coins and health points if the user is using the gamification feature described in F17.

priority: High

F4 Configure Pomodoro time presets The user can configure up to 4 different Pomodoro timer settings. The length of the working time, break and long break can be set. It is also possible to set the automatic start of a break or a work session. Finally, the user can choose whether or not to use Pomodoro intervals. If the user does not use Pomodoro intervals, the intervals will not be displayed and the long break will not be used.

priority: High

To-do List

These functional requirements describe what the user can do with the task list. This can be creating tasks, deleting tasks, and other functions.

F5 Create/Remove tasks The user will be able to create new tasks and remove existing ones. The name of the task can be the same.

The edit function is not included because the only thing the user can edit is the task name. The user can do this by simply removing the old task and creating a new one.

priority: High

F6 Focus task When the user is working on a task or prioritising it in the current working session, they have the option to highlight it for emphasis. Tasks can be unhighlighted. Multiple tasks can be highlighted at the same time.

priority: High

F7 Finish task The user can mark the task as done. The task is then moved to the completed section.

priority: High

F8 Fail task A task can be moved to the failed list if the user has not completed it. The user can use this feature at his/her own discretion, as the tool is not able to determine whether he/she has actually failed the task or not.

priority: Medium

F9 Undo task A task can be moved back to the to-do list if it was accidentally moved to the completed or failed list. However, deleted tasks cannot be recovered.

priority: Medium

Blocking management

F10 Manual blocking Applications and websites can be manually blocked or unblocked if the user does not want to use the Pomodoro Timer. The user can also pause the blocker for a specified amount of time as described in F3, but cannot pause the timer indefinitely as there is no timer in manual blocking.

priority: Medium

F11 Web blocking Selected websites can be blocked by the user based on their settings. Websites will be blocked during a Pomodoro session or manually, as described in F10. All websites can be unblocked during a Pomodoro break session. The user can also use the ”Pause Blocker” described in F3 to unblock websites for a certain period of time.

priority: High

F12 Web exceptions If the user wants to allow selected websites and block the others, they can use this feature. All websites except the selected ones can be blocked or unblocked as described in F11

priority: Medium

F13 Add and remove websites The user can add or remove websites to be blocked or to be allowed.

priority: High

F14 YouTube blocking The user can partially block YouTube. The home page and the video suggestions sidebar while watching a video can be blocked according to the user's settings. YouTube blocking works together with web blocking described in F11.

priority: Low

F15 Application blocking Selected applications can be blocked by the user based on their settings. Applications will be blocked during a Pomodoro session or manually, as described in F10. All applications can be unblocked during a Pomodoro break session. The user can also use the "Pause Blocker" described in F3 to unblock websites for a certain period of time.

priority: Low

F16 Add and remove applications The user can add or remove applications to be blocked or to be allowed.

priority: Low

Gamification elements

F17 Coins and health points management The user can gain or lose health points and coins. If the user completes the task, the user will get coins. If the user fails the task, the user loses coins and health points.

If the user completes the task, the user receives coins. If the user fails the task, the user loses coins and health points. If the user completes the task, the user receives health points and coins. The amount of coins the user receives is based on the amount of time the user has worked. In general, the longer the time, the more coins the user will receive.

If the user needs to take a break from blocking, the user will lose health points and coins. The amount of coins depends on the length of the break. The number of health points is fixed.

Gamification features can be disabled. The user cannot gain or lose coins or health points. All other features work without restrictions. The user can see how many coins and health points are left.

priority: High

F18 Tracking working statistics Work time is statistically recorded every day in minutes. Only the last 30 days are recorded.

priority: Medium

General settings

F19 Gamification settings The user can change the gamification configuration. The user can choose to enable or disable coins or health points, or both.

priority: Medium

F20 Timer notifications setting The user can enable or disable the notification of the timer when it has finished.

priority: High

F21 Application theme setting The user can change the theme of the application to a dark mode or a light mode.

priority: Low

Other requirements

These functional requirements are a bonus, so they are not guaranteed to be implemented.

F22 Notification blocker The user can select applications and block their notification. The user can also block all notifications.

priority: Low

F23 Discord status The user can add a message to their Discord profile based on their current work or break session. Other Discord users will be able to see this message. The message can be changed based on the user settings. It can be something like ”I am currently working” if the user is in the work session.

priority: Low

1.7 Non-functional requirement

NF1 SQL (SQLite) To store data, the application must use SQLite as its database.

NF2 Web socket The desktop application (server side) must communicate with the web extension (client side) through the web socket to block and unblock websites.

NF3 Security The application should communicate locally with minimal or no connection to the Internet to prevent unauthorised access. The desktop application (server side) should not receive data from outside (via the web socket) to prevent the application from being controlled by others.

The application must not block or terminate critical system processes using the application block feature.

NF4 Reliability The application must catch as many exceptions as possible and report them to the user. When the application receives an exception, such as a wrong data input, it should not crash or cause data loss.

NF5 User-friendly UI design Design should be simple for the user. Navigation must require the lowest possible operation count.

It is a good thing that the solution partially meets some of Nielson’s heuristics. This makes the result more user-friendly. In general, Nielson’s 10 general principles is use to make the UI more user-friendly. [12]

NF6 Desktop application using C# with .NET 7 The application must be implemented using C# with .NET 7 for the Windows 10 (version 1607 or later) or Windows 11 operating system.

Chapter 2
Design

2.1 Use-Case

This section describes how the user interacts with the application to achieve their goal and how the application should respond. Use cases were defined mainly from functional requirements and partly from non-functional requirements. Use cases are grouped into categories for the sake of clarity.

The use cases for blocking notifications and changing Discord status are not covered here. This is because there are some issues with these features that will be discussed later.

There is only one actor for all use cases. It is the user that uses the application.

Manage Pomodoro timer

UC1 - Start the Pomodoro timer (PT)

Pre-condition:

- The manual blocking function (MBF) is stopped or not yet started.

Post-condition:

- MBF is disabled for user.

Basic Path

1. The use case begins when the user clicks the “Start” button.
2. The timer starts and the Pomodoro work session (PWS) begins.
3. When the PWS ends, the application enters the Pomodoro break session (PBS). There are 2 situations depending on the timer setting.
   a. If the user has set the timer to automatically start the break, the PBS will start immediately after the PWS ends.
   b. Otherwise, the user must press the “Start” button to enter the PBS.
4. After the PBS has finished, the application will return to the PWS.
   a. As before, depending on the timer setting, the user must press the “Start” button to proceed to the PWS.

UC2 - Restart the Pomodoro timer

Post-condition:

- The MBF is enabled for user.
- All websites and applications are unblocked.

Basic Path

1. The user clicks on the “Restart” button.
2. The timer stops.
3. The application enters the PWS and resets the timer time to the PWS time.

UC3 - Pause the Pomodoro timer

Pre-condition:

- The application is in PWS and the timer is counting down.

Basic Path

1. The user clicks on the “Pause” button.
2. The application will open a window with options to pause the timer, or to unblock everything and pause the timer for a period of time.
   a. If the user chooses to pause the timer, the application will pause the timer indefinitely.
   b. Otherwise, the application will pause the timer for a specified period of time.

UC4 - Change Pomodoro timer preset and setting

Pre-condition:

- The PT and MBF is stopped or not yet started.

Basic Path

1. The user clicks on the button with number from 1 to 4 to choose the timer preset.
   a. This use case ends here if the user does not want to change the timer presets.
2. The user clicks the button with the timer cogwheel icon.
3. The application displays the timer presets.
4. The user selects the preset and changes the settings.
   a. If the user wishes to keep the changes, they click the “Save” button.
   b. Otherwise, the user clicks the “Cancel” button.
5. The application returns to the PT windows.

Manage To-do List

UC5 - Create task

Basic Path

1. The use case begins when the user clicks on the “Enter task name” field and enters task name.
2. The user clicks the “Add” button.
3. The application adds a new task and displays it in the To-do list.

UC6 - Finish task

Pre-condition:

- The status of the task is “To do”.

Post-condition:

- The status of the task is “Done”.

Basic Path

1. The user clicks on the task’s checkbox to mark it as done.
2. The application moves the task to the “Done” list.
3. The user can view the completed task by clicking on the “Done” tab.

UC7 - Fail task

Pre-condition:

- The status of the task is “To do”.

Post-condition:

- The status of the task is “Failed”.

Basic Path

1. The user right-clicks on a task and selects the “Fail Task” option.
2. The application moves the task to the “Failed” list.
3. The user can view the failed task by clicking on the “Failed” tab.

UC8 - Undo task

Pre-condition:

- The status of the task is not “To do”.

Post-condition:

- The status of the task is “To do”.

Basic Path

1. The user clicks on the “Failed” or “Done” tab.
2. The user right-clicks on the selected task and selects “Undo” option.
3. The application moves the task back to the “To do” list.

UC9 - Delete task

Basic Path

1. The user right-clicks on the selected task and selects “Delete” option.
2. The application removes selected task.

Start and stop block

UC10 - Block

Pre-condition:

- Website or application block is enabled.
- PT and MBF are stopped or have not yet started.

Basic Path: Block websites and applications using the Pomodoro timer.

1. The user starts the PT as in UC1.
2. The application starts to block selected applications and websites, but unblocks when the application launches the PBS.

Alternate: Block websites and applications using the Manual blocking function.

1. The user switches from the Pomodoro Timer to the Manual blocking function by pressing the ”Manual Blocking” tab.
2. After switching, the user clicks on the “Start” button.
3. The application starts to block the selected applications and websites for an indefinite period of time.

UC11 - Unblock

Pre-condition:

- The PT or MBF is started.

Basic Path Stop blocking for an indefinite period of time

1. The user clicks the “Restart” button when using PT to block websites or applications. When using MBF to block, the user clicks on the “Stop” button.
2. The application stops blocking everything.

Alternate Stop blocking for a certain period of time

1. The user clicks on the “Pause” button.
2. The application will open a window with options to pause the timer, or to unblock everything and pause the timer for a period of time.
3. The user chooses the duration of the unblocking and clicks on the “Unblock” button.
4. The application stops blocking for a certain period of time according to the user’s previous selection.

Manage block settings

The following use cases describe important situations where the user wants to change blocking settings. These settings include adding or removing a website or application. Another website blocking setting is a mode that allows only the sites selected by the user, rather than blocking specific websites.

All use cases **pre-condictions** are: PT and MBF are stopped or have not yet started.

UC12 - Add or remove Web to block

Basic Path

1. The user clicks the “Block” tab and then the “Website to block” field.
2. If the user wants to add a website to block:
   a. The user enters a web URL in the field and clicks on the “Add” button.
   b. The application adds the web URL and displays it in the list of blocked websites.
3. If the user wants to remove a website from blocking:
   a. The user clicks the button with a minus sign on the selected URL.
   b. The application removes the web URL from the list.

UC13 - Change to exception mode

Basic Path

1. The user clicks the “Block” tab and the clicks on the “Use allow mode” toggle.
2. The application switches to “allowed mode” and displays a list of allowed websites.

UC14 - Add or remove app to block

Basic Path

1. The user clicks on the “Block” tab and then on “Application” tab.
2. If the user wants to add an application to block:
   a. In the application drop-down box, the user selects an application to block.
   b. The user clicks on the “Add” button.
   c. The application displays the application to block in the list.
3. If the user wants to remove an application to block:
   a. The user clicks the button with a minus sign on the selected application.
   b. The application removes the selected application from the list.

View Statistics and change settings

UC15 - View statistics

Basic Path

1. The use case starts when the user wants to see the statistics of his/her working time.
2. The user clicks on the “Statistics” tab.
3. The application displays working statistics.

UC16 - Change settings

Basic Path

1. The user clicks on the “Settings” tab.
2. The following situation may occur.
   a. The user changes the colour scheme. The application displays a new colour scheme based on the user’s previous selection.

b. The user changes the gamification settings. The application stops using the gamification features, but the gamification UI elements, such as “Coins” and “Health points”, still appear in the UI.

c. The user turns notifications on/off. The application sends a notification when the PWS/PBS ends, based on the user’s previous selection.

3. The user must click the “Save” button to save changes or the “Cancel” button to discard changes.

2.2 Selected technologies

The application will be a desktop application running on the Windows operating system. According to the questionnaire, up to 85.2 % of respondents use the Windows operating system.

For desktop applications, there are many technologies to choose from. However, after consulting with the supervisor, C# and .NET were chosen for development. The following text describes C# and .NET. Some information is current at the time of writing, such as the popularity of the technology. This information may change in the future.

C#

C# is a modern, object-oriented, type-safe, cross-platform programming language [13]. This language is familiar to JavaScript, Java or C++.

C# is very popular, being one of the top five programming languages on GitHub. It has a large community of over 5 million developers. There are many resources available for learning C#. These include Microsoft Learn, Stack Overflow, YouTube, .NET Live TV and more. [14]

This language is the most popular for .NET development [14].

.NET

What is .NET? It is a free, cross-platform, open-source developer platform that supports multiple programming languages C#, F#, and Visual Basic for building many types of applications. You can use .NET to build many different types of applications, such as native applications for Windows and macOS, web applications, games, and much more. [13]

.NET has a standard set of class libraries that provide implementations for many general purpose types and utility functionality. There is also a package manager for .NET called NuGet, which provides many popular packages from the community. .NET offers many advanced language features such as generics, LINQ[1] and asynchronous programming, along with extensive class libraries. [15]

.NET is fast, which means that applications have better response times and use less compute power. The .NET platform is trusted by thousands of companies and milions of developers. It is officially supported by Microsoft and Microsoft takes security very seriously and releases updates quickly when threats are discovered. If you want to know more about .NET and why you should choose .NET, you can read about it here [15] and here [16].

Descriptions of other selected technologies are included directly in the relevant sections dealing with specific topics. For example, technologies used for databases are described in a separate section on databases.

[1] Language Integrated Query

2.3 User Interface

This section contains information about the User Interface (UI). First, there is a discussion of the characteristics that the UI should have in order to be intuitive for users. This is followed by the design of the screens and finally the selection of a UI framework for implementation.

The UI should be designed to be simple and intuitive to use based on NF5. Too complicated UI can lead to a situation where the user gives up on using the application on the first attempt. To achieve a user-friendly UI, the UI design will follow some of Nielsen's principles:

- The design should actively inform the user about the statuses and errors. This allows the user to determine the next steps.
- Users often perform actions by mistake. They need a clearly marked emergency exit button to exit or stop unwanted action without having to go through an extended process. If users navigate to another screen to perform an action, they should be able to easily find the place to perform that action.
- The application should prevent errors. If an error occurs, the application should notify the user with error messages and actions to correct the error.
- The UI should be minimal and not contain irrelevant information. Any additional information that is not relevant to the current context (screen) can reduce the visibility of important elements. This can distract the user from taking the desired action. [12]

2.3.1 UI frameworks

There are many UI frameworks and UI libraries to build Windows desktop applications. These are, e.g.: Windows Presentation Foundation (WPF), Windows Forms, WinUI, Material Design In XAML and WPF UI.

WinUI offers a more modern and intuitive design [17]. WinUI is still an emerging technology. It has less resources and community support than WPF. It may be less stable.

WPF uses XAML for UI design. This separates the UI logic from the business logic. Windows Forms does not use a markup language for design, and it is more difficult to separate the UI logic from the business logic. Windows Forms has limited data binding[2] between UI and business logic. [19], [20]

WPF was chosen as the UI framework. Because WPF has been around for a long time, there is a large community and many resources. The author also has experience with WPF. Although WPF is older, it is still supported and updated by Microsoft [21]. WPF is compatible with other UI libraries such as WPF UI and Material Design In XAML. This makes the resulting application beautiful and modern.

The Material Design In XAML library was chosen for UI development with WPF. This library implements Google's Material Design principles and is compatible with WPF. Compared to WPF UI, this library has a longer history and more resources to learn from. [22]

2.3.2 Screen design and prototype

The application must have multiple screens, sections, and tabs to organise all functions logically and intuitively. Based on the use case above, it is ideal to split into four basic screens:

- **The home screen** contains the Pomodoro timer and to-do list to keep users informed of time remaining and current work tasks.

[2] "Data binding is the process that establishes a connection between the app UI and the data it displays. If the binding has the correct settings and the data provides the proper notifications, when the data changes its value, the elements that are bound to the data reflect changes automatically." [18]

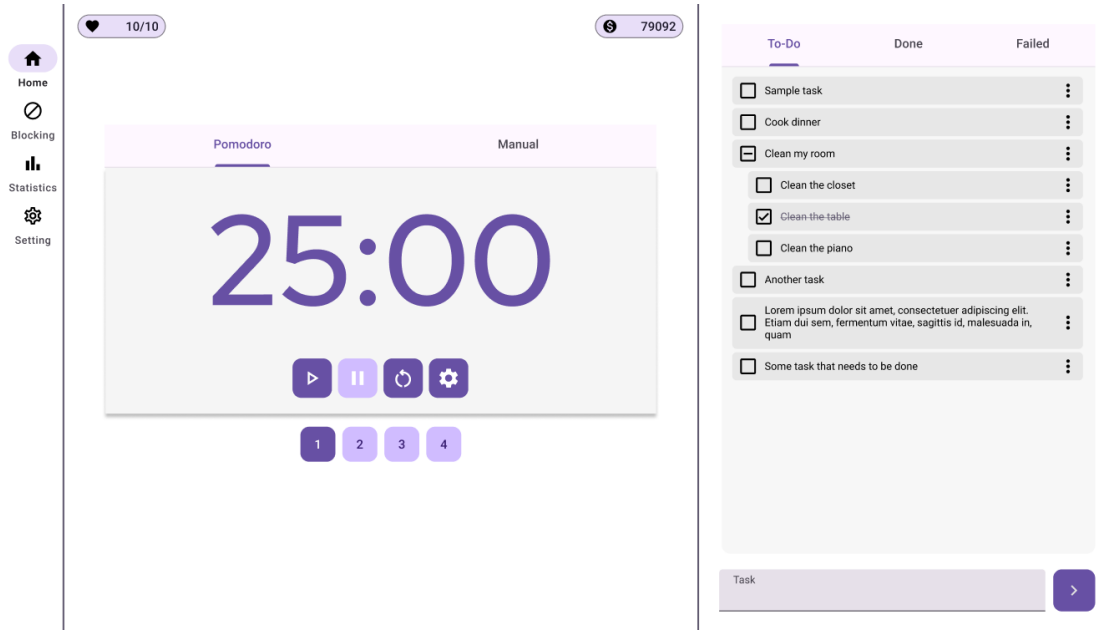

Figure 2.1 Prototype screenshot – Home screen

- **The blocking manager screen** includes all the settings to block websites and applications.
- **The statistics screen** displays working statistics.
- **The Settings screen** contains the remaining general settings.

These screens are organised into a main tab bar with icons that are always visible to users from the left side of the application. Users can easily switch between these four screens at any time with a single click on the tab.

The following images are prototypes that were used as design during the development. These prototypes closely resemble look of the final application. The prototypes was created using Figma [23]. The prototypes use the Material 3 Design Kit and the Material Design Icons package to resemble the target application as closely as possible.

Home screen

This is the default screen. Users will enter this screen after running the application. It contains the most commonly used and important components. These components are the Pomodoro Timer (PT) and the to-do list. They should be on the same screen to inform users of the current task to be completed and the time remaining on the timer.

Users spend most of their time interacting with the PT and to-do list, as they need to interact with them frequently, such as start/stop the timer or add/complete tasks. Other features, such as general settings, application and website blocking settings, can be set once and do not need to be changed frequently by the user.

This screen is divided into two separate sections. The first section on the left is for the PT and the second section on the right is for the to-do list. Navigating in one section does not affect the other. For example, if the user enters the PT setting, they will still be able to see the to-do list and navigate to the list of completed tasks. The prototype of the home screen can be seen in the figure 2.1.

Let's start with the task list section. This section contains three task lists divided into three tabs.

- The first one is the current to-do tasks, this is also the default list that is displayed to the user.
- The second contains the completed tasks.
- The last one contains the failed tasks.

The user can freely navigate between these three lists by clicking on the tab with the name of the list. The "Add New Task" function is at the bottom of the list and is always visible, regardless of the type of list.

The PT section contains the PT, manual blocking, and labels with gamification elements that provide information about health points and coins. The PT and the manual blocking are split into two tabs. Labels with gamification elements are always visible at the top, independently of these two tabs.

The PT tab is the default tab with these features:

- Start and stop the PT.
- Switch the PT time presets.
- Pause the PT or unlock everything for a fixed period of time.
  - This is displayed as a button with a pause icon. The "pause" button has two functions.
  - When the user clicks on it, a new pop-up window is displayed within the PT section.
  - In this window, the user can unblock everything for a fixed period of time or pause the timer indefinitely.
- Change the PT settings.
  - This function contains many information such as presets, work time, break time, and auto start.
  - After clicking on the "PT setting" button, the user is navigated to the PT setting window inside the PT section.

The manual block tab contains the following features:

- Start and stop the manual blocking.
- Pause the blocker for a fixed period of time.
  - After the user clicks on the "pause" button, a pop-up window will appear.
  - In this window, the user can set the unblocking time.

The following diagram in figure 2.2 shows how the user can navigate in the Pomodoro section of the home screen.

Blocking manager screen

The blocking settings are divided into a website blocking section and an application section. Each section is on a separate tab for ease of use. The default tab is the website blocking tab. The prototype of the blocking screen can be seen in the figure 2.3.

The website block screen contains these elements:

- Toggle buttons to use the website block, hide YouTube elements, and allow websites instead of blocking them.
- Two lists of websites, one to block and one to allow.

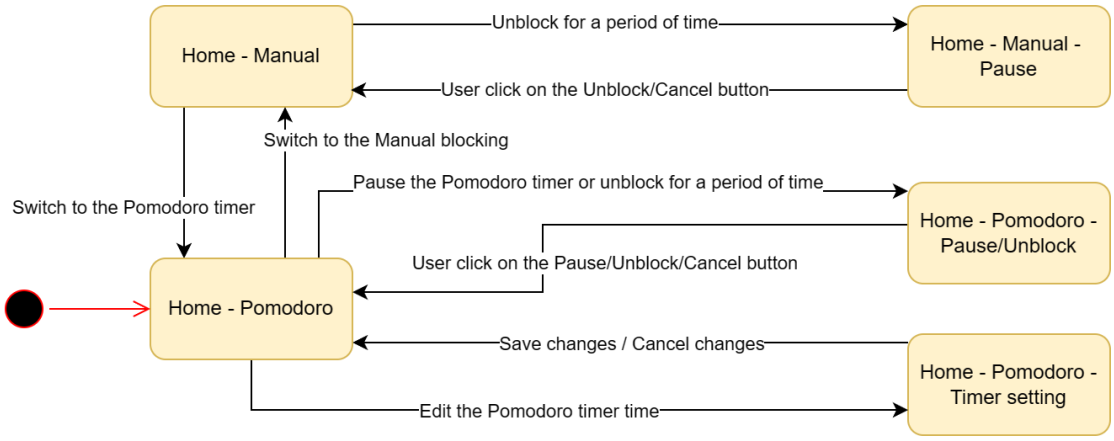

Figure 2.2 Navigation in the Pomodoro section

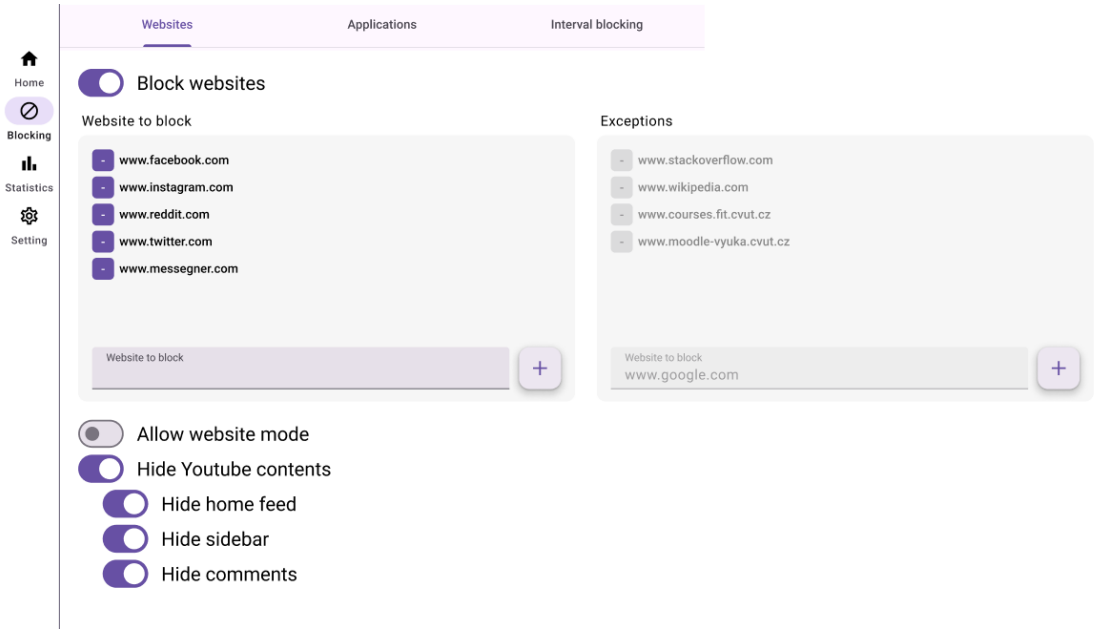

Figure 2.3 Prototype screenshot – Blocking manager screen

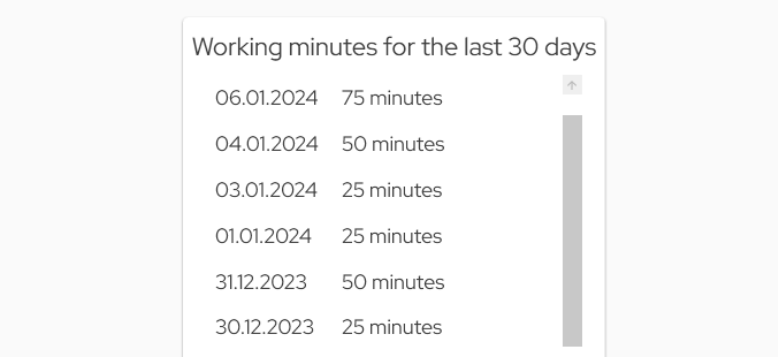

Figure 2.4 Prototype screenshot – Statistics screen

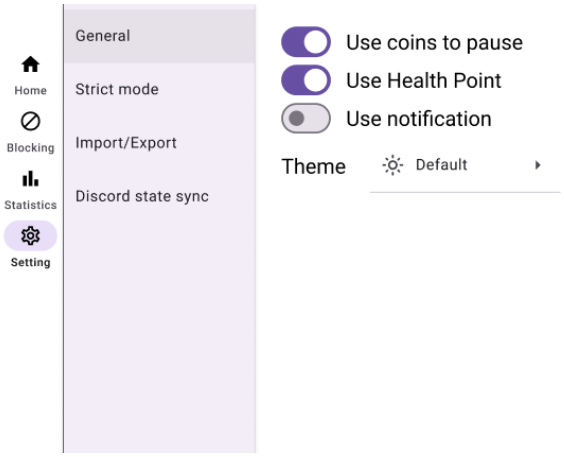

Figure 2.5 Prototype screenshot – Setting screen

- One of the lists is blurred depending on whether the allow mode is enabled or not.
- Blurring one of the lists can increase clarity for users as they can only interact with the list they need.
- Users will also notice if the “allow” mode is active or not.

The application blocking tab contains only a toggle to use application blocking and a component to add or remove applications to block.

Statistics screen

This screen displays only work time. The user cannot interact with this screen. The prototype of the statistics screen can be seen in the figure 2.4.

Setting screen

The setting screen can be divided to tabs that contains other types of settings. The default one is general setting that contains settings like change appearence. Other screens are notification and discord status settings. The prototype of the setting screen can be seen in the figure 2.5.

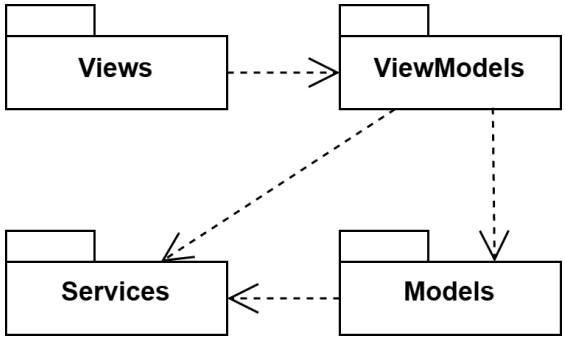

Figure 2.6 Package diagram

2.4 Architecture

The application will utilize the Model-View-ViewModel (MVVM) architectural pattern. As described above, WPF[3] was chosen for development, and WPF was designed to make it easy to build applications using the MVVM pattern. Other benefits of MVVM will be discussed later. For more information on why to choose MVVM for WPF please see [24].

MVVM

MVVM is an architectural pattern that helps to cleanly separate an application's business and presentation logic from its user interface. There are three main components in the MVVM: the model, the view, and the view model. Each of them serves a different purpose. [25]

To understand the responsibilities of each component, it is important to understand how they interact. At a high level, the view is dependent on the view model. The view model is dependent on the model. But the model is unaware of the view model, and the view model is unaware of the view. The view model isolates the view from the model, and allows the model to evolve independently of the view. [25]

Basically, the view takes care of everything related to displaying data and interacting with users. The view model handles user input and manages the flow of the application based on that input. The model is used to represent the data and contains the logic to work with it. Other logic, such as website/application blocking or timer countdowns, is placed separately in the service layer. The diagram in the figure 2.6 shows how the following layers depend on each other.

The benefits of using the MVVM pattern are as follows:

- Unit tests for the view model and the model can be created, without using the view.
- The user interface can be redesigned without changing the view model and model code, provided that the view is implemented entirely in XAML or C#. Therefore, a new version of the view should work with the existing view model.
- Designers and developers can work independently on their components. Designers can focus on the view. Developers can work on the view model and the model components. [25]

One of the most important aspects of WPF that makes MVVM a great pattern to use is the data binding infrastructure. By binding properties of a view to a view model, these components are now loosely coupled. There is no need to write a code in the view model that directly updates a view. [24]

[3]Windows Presentation Foundation as UI Framework, C# as programing language and .NET as developer platform.

The following sections discuss the responsibilities of each component in the MVVM pattern in the context of WPF.

View

The view is responsible for defining the structure, layout, and appearance of what the user sees on the screen. Ideally, each view is defined in XAML. The code-behind of the view should not contain business logic, but in some cases the code-behind may contain UI logic, such as animation, that is difficult to express in XAML. [25]

A view can have its own view model, or it can inherit the view model of its parent. A view gets data from its view model through data binding. The view models are responsible for defining logical state changes that affect some aspects of the view's display, such as whether a command is available. It is preferable to enable and disable UI elements by binding to view model properties, rather than enabling and disabling them in code-behind. [25], [26]

There are several ways to execute code on the view model in response to interactions on the view, such as a button click or item selection. If a control supports commands, the Command property of the control can be data-bound to an ICommand property on the view model. When the control's command is invoked, the code in the view model will be executed. [25]

ViewModel

The view model is an intermediary between the view and the model/services. This means that communication from the view to the model/services (or vice versa) goes through view models. It manages the flow of the application based on user input. The view model may use other services to perform some actions, such as blocking websites.

The view model implements properties and commands to which the view can data bind to. It notifies the view of any state changes through change notification events. The view model is responsible for coordinating the view's interactions with any model classes that are required. [25]

Each view model provides data from a model in a form that can be easily consumed by the view. To achieve this, the view model sometimes performs data conversion. Placing this data conversion in the view model is a good idea because it provides properties that the view can bind to. [25]

Model

Model classes are non-visual classes that encapsulate the application's data. Therefore, the model can be thought of as representing the application's domain model, which typically includes a data model along with business and validation logic. [25]

Model classes are typically used in conjunction with services or repositories that encapsulate data access and caching. Other services, such as website/application blocking, are no longer associated with the model but with the view model. [25]

2.5 Database storage

This section focuses on selected database technology and describes the database model and its attributes.

Selected database technology

The application needs to store data such as settings, to-do tasks, information about blocked sites or applications, and statistics. Some data, such as preferences, only require a table to store the data. In general, the relationships between the data are simple. In theory, a text file would

suffice. This means that the database system can be simple. The database can be stored as a file directly in the application. This is why SQLite was chosen.

SQLite

SQLite is a library that implements a small, fast, self-contained[4], high-reliability, full-featured SQL database engine. SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables and views is contained in a single disk file. This also simplifies the backup and restore process. [28]

Database model

As mentioned above, the application needs to store settings such as general settings, Pomodoro timer settings, tasks for the to-do list, information about blocked sites or applications, and work statistics. The relationships between the data are minimal because they are mostly unrelated. As there is only one user, some of the table records are fixed. For example, one table record is sufficient to store general settings, and four records are sufficient to store four Pomodoro timer presets. The database model can be seen in figure 2.7.

Here is a description of the tables and their attributes.

GeneralSettings stores the general settings described below. There will always be one record of this table as it represents the general settings for the whole application.

- `Theme` - Represents current application color scheme.
- `UseNotification` - Enable or disable the pop-up notification at the end of the Pomodoro timer.
- `UseCoins` - Enable or disable the coin gamification feature. Disable means that the user will not be able to gain or lose coins.
- `UseHealthPoints` - Same as `UseCoins`, but for health points.
- `DcWorkText` - Text displayed in Discord while the user is in a Pomodoro session (PWS).
- `DcBreakText` - Same as `DcWorkText`, but for the Pomodoro break session (PBS).
- `DcIdleText` - Text displayed in Discord when the user is not in PWS or PBS.

PomodoroPresetSettings represents the Pomodoro timer (PT) settings. There are four presets. Therefore, this table always has four records.

- `WorkTime` - This attribute is the work time for PT in seconds.
- `ShortBreakTime` - Same as `WorkTime`, but this is the break time after a work session.
- `LongBreakTime` - Same as `ShortBreakTime` but this is a longer break. `LongBreakTime` replaces `ShortBreakTime` after every four work sessions.
- `UseIntervals` - This enable or disable long break.
- `AutoStartBreak` - This enables or disables the auto start break at the end of the PT work time.
- `AutoStartWork` - Same as `AutoStartBreak` but for work time.

PlayerStats This table stores data related to the gamification elements. There will always be one record in this table. This is because it only stores information, such as remaining coins and health points, that can be stored in one record.

[4]Means that SQLite has very few dependencies [27].

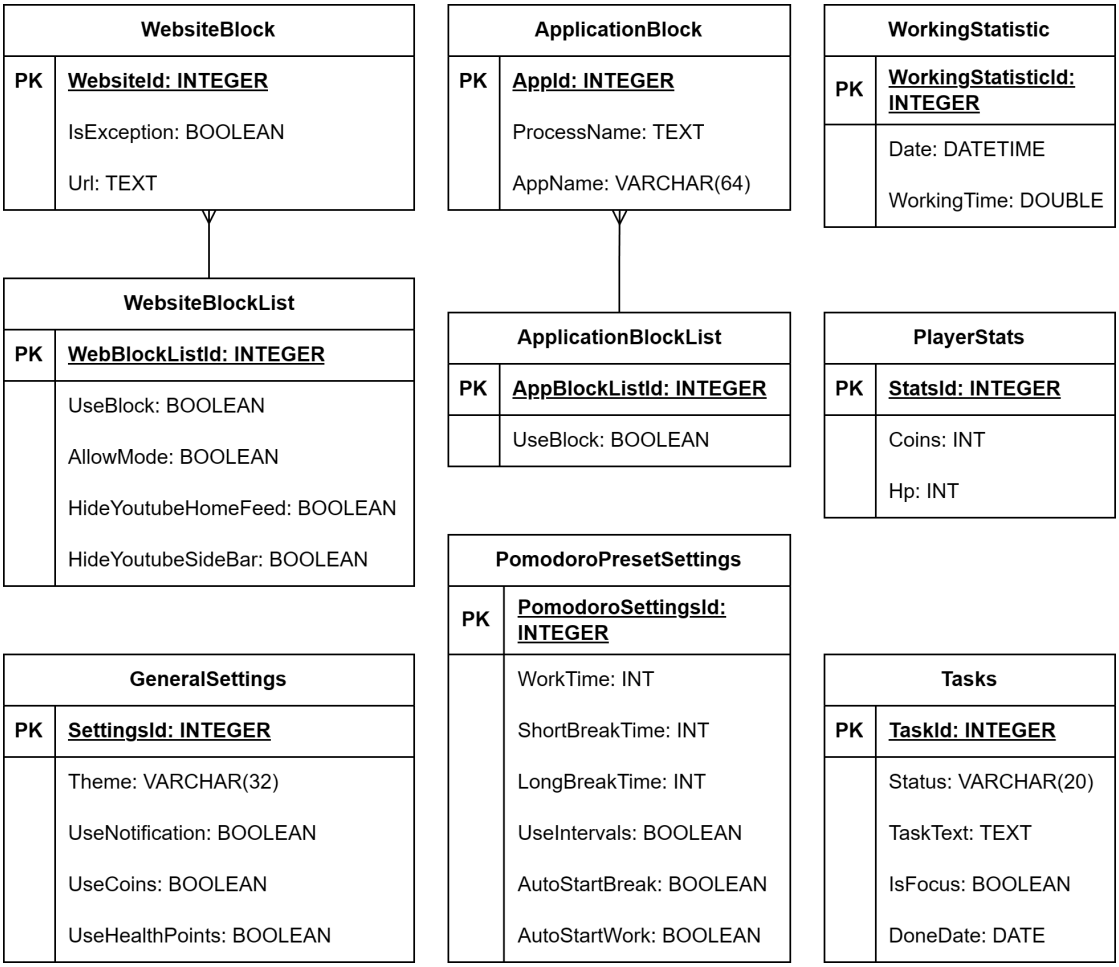

Figure 2.7 Database model

- `Coins` - User's current coins.
- `Hp` - User's current health points.

WebsiteBlock stores the URL of the website to be blocked or allowed.

- `IsException` - Indicates whether the current website is allowed (`true`) or blocked (`false`).
- `Url` - Website URL.

WebsiteBlockList This table represents a list of websites to blocked or allowed. There will always be one record in this table, since there is only one list of sites to block or allow.

- `UseBlock` - Enable or disable website blocking.
- `AllowMode` - Toggle between blocking and allowing websites.
- `HideYoutubeHomeFeed` - Hide or show YouTube recommended videos on the home page.
- `HideYoutubeSideBar` - Hide or show recommended YouTube videos in the sidebar of the current video.

ApplicationBlock represents an application to be blocked.

- `ProcessName` - Name of the process to be blocked.
- `AppName` - Name of the blocked application in a readable format.

ApplicationBlockList This table represents a list of applications to be blocked. There will always be one record in this table, since there is only one list of applications to block.

- `UseBlock` - Enable or disable application blocking.

Tasks This table stores all tasks for the to-do list.

- `Status` - Indicate whether the task is to be done or has been done/failed.
- `TaskText` - Task description
- `IsFocus` - Highlight the task if `true`.
- `DoneDate` - Date and time when the task was completed or failed to sort the tasks in the list.

WorkingStatistics This table stores statistics on working time.

- `Date` - The day of the recorded working time.
- `WorkingTime` - Total number of minutes worked in a day.

For better extensibility, the database can be designed differently. This database design is included in the enclosed medium. In this case, it is possible to have, e.g. multiple users, multiple setup profiles, multiple application/website block lists, and to-do lists. However, due to the scope of the thesis, a simpler database, as described above, will be used.

2.6 Website blocking

There are several ways to block a specific website. These ways are blocking via the Hosts file or via the web extension.

The Hosts file is used by the operating system to map human-friendly hostnames to numerical Internet Protocol (IP) addresses which identify and locate a host in an IP network [29]. This means that the Hosts file allows the user to redirect a domain name to a custom IP (for example to 0.0.0.0). As a result, the user will no longer be able to access that domain unless it is removed from the Hosts file. [30].

However, blocking via the Hosts file requires restarting the browser to take effect. This means that the application would have to restart all browsers, which could result in data loss. Due to the negative effects, a second approach (using the web extension) was chosen.

Web extension

The web extension in this thesis primarily targets Google Chrome. Extensions for Google Chrome are also supported by other browsers such as Edge and Opera. The web extension will be developed using JavaScript.

The blocking will work as follows. The web extension connects to the application via a WebSocket. When the application needs to block a web page, it sends a message to the web extension. The web extension then blocks access by creating a layer over the web page to be blocked, preventing the user from interacting with the page. In the case of unblocking, the web extension simply removes this layer.

This solution has a downside. It requires the web extension to be installed in the browser, and not all browsers support the same extensions.

Communication

The communication between the desktop application and the web extension will be implemented using WebSocket. The WebSocket is an advanced technology that makes it possible to open a two-way interactive communication session between the user's browser (the web extension) and a server (the desktop application) [31]. This means that the application can send messages to the web extension to block the website if the extension is connected to the application. If the extension is not connected, it will always attempt to connect to the application in a fixed period of time (e.g. 5 seconds).

2.7 Other functions

Application Blocking

To block applications or games, this application can force the target application or game to close. This solution was chosen because it is simple. However, it is sometimes inconvenient because it closes the blocked applications, which can cause data loss.

There are also several things to keep in mind. The application must not block critical system processes, only the user processes. It must not block system applications such as Windows Explorer, Command Prompt, or Task Manager.

Notification blocking

After research, the author cannot find an easy way to block notifications directly from the application. There is a way to disable notifications via the Windows registry as described in [32]. However due to the complexity and scope of this thesis, it was decided not to include this functionality.

Discord status sync

After research, the author cannot find a way to change the user status on Discord from within the application. For example, when the user enters the Pomodoro work session (PWS), Discord changes the status to "do not disturb" or when the user enters the Pomodoro break session (PBS), Discord changes the status back to "available".

However, Discord has "Rich Presence" which allows the application/game to display custom text in the "Now Playing" section of a Discord user's profile via the Discord GameSDK. Other users will be able to see this text. [33]

Based on the current timer session, there are three types of text that can appear on a Discord user's profile. Users can set a custom text while in PWS, PBS, or neither. For example, when the user is in PWS the text could be "I am working, please don't text me.". If the user is not in PWS or PBS, the text could be "I am not here." or just a blank text.

Chapter 3
Implementation

3.1 Development tools and libraries

To develop the application, it is necessary to use tools. These tools are an IDE[1] and a version control system. IDE simplifies the task of writing code. A version control system serves as a backup for the code. The following tools were used during the development of the applications:

Visual Studio 2022 is the comprehensive IDE for .NET and C# developers on Windows. It can be used to edit, debug, and build code. It also provides a graphical designer, a NuGet Package Manager for installing libraries, and many other features. [34]

The application is developed mainly using Visual Studio 2022.

Visual Studio Code is a lightweight source code editor that runs on Windows, macOS, and Linux. It comes with built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other languages and runtimes. [34]

The web extension to block websites is developed using Visual Studio Code.

ReSharper is an extension for the Visual Studio for .NET developers. It helps Visual Studio users to analyse the code quality, eliminate errors and code smells, and much more. [35]

Git is the version control system used in this thesis along with GitLab.

The following libraries are used to simplify development:

Material Design In XAML is a fully open source and one of the most popular GUI[2] libraries for WPF. It styles the majority of standard WPF controls using Google's Material Design principles. [22]

Mvvm Community Toolkit The Mvvm Community Toolkit is a modern and fast .NET MVVM library with helpers that make writing code faster. One of such helpers include `ObservableObject`, a base class for objects that implement the `INotifyPropertyChanged` interface.

`ObservableObject` can be used to reduce the code like in the code listing 1. [36]

Configuration Manager is a library that supports the use of configuration files. [37]

[1] Integrated Development Environment
[2] Graphical User Interface

Entity Framework Core (EF Core) serves as an object-relational mapper, allowing .NET developers to work with a database using .NET objects. It eliminates the need for most of the data access code that is typically written. EF Core can access many different databases through plug-in libraries called database providers. [38]

The application in this thesis will use SQLite as its database storage. Therefore, it requires the `Microsoft.EntityFrameworkCore.Sqlite`, which is a SQLite database provider for EF Core. [39]

```
// isFocus property
private bool _isFocused;
public bool IsFocused
{
    get => _isFocused;
    set
    {
        _isFocused = value;
        OnPropertyChanged(nameof(IsFocused));
    }
}

// isFocus property with MVVM toolkit
[ObservableProperty] private bool _isFocused;
```

Code listing 1 MVVM Toolkit – Code reduction

3.2 Project structure

This section focuses on the project structure and its components. It can also be used as a developer's manual. The main components of the project with a brief description are listed below.

- `ImprovementFocusTool`
  - `Commands` ..... Command classes to connect business logic with the UI
  - `DbContexts` ..... Classes for managing DB operations and mapping entities
  - `DTOs` ..... Simple objects for transferring data between DB layer and DB storage
  - `Fonts` ..... Application fonts
  - `Migrations` ..... Set of changes applied to the database
  - `Models` ..... Objects for transferring data between layers in the application
  - `Resources` ..... Assets such as audio files and images
  - `Services` ..... Classes with logic operation such as blocking websites
  - `Stores` ..... Holds data/states and allows others to subscribe to changes
  - `ViewModels` ..... Handles UI logics
  - `Views` ..... UI components
  - `App.config` ..... Stores application configurations
  - `App.xaml` ..... The application entry point
  - `MainWindow.xaml` ..... Default application window
  - ...
- `ImprovementFocusTool.Tests` ..... Project with unit tests

As described above, the application uses the MVVM pattern. The structure of the application is divided into four main components: the view, the view model, the model and the service. A more detailed description, such as the interaction between these components and their responsibilities, is given below.

Views

The view takes care of everything related to displaying data and interacting with the user. A view can be split into smaller views/components called "User Controls", which helps to make the code more readable. However, it is not always best practice to split into smaller components, as each component may need its own view model and creating sub-components can be complicated. This can complicate the project structure and take longer to develop. For this reason, some views contain more functionality to reduce complexity.

`MainWindows.xaml` is the main window. It contains all the UI components and presents them to the user as a whole. All UI components (views) are located in the Views folder. Views is divided into five main parts. Each of these can be made up of several components, which are located in the `UserControl` folder.

- `HomeView.xaml` is the default view and is divided into two sub-components.
  - `TimerHomeControl.xaml` has two parts. The Pomodoro timer (PT) and the Manual blocking. Both have control buttons, such as Start, Stop, and Pause. In addition, the PT has a function to change or set the timer presets. As the preset setting function contains many controls, it is divided into the sub-component `PomodoroProfileSettingsControl.xaml`.
  - `ToDoHomeControl.xaml` contains the to-do list, the completed list, the failed list of tasks, and a text box with a button to add new tasks.
- `BlockingView.xaml` is divided into two sub-components.
  - `AppBlockingControl.xaml` contains a toggle to enable/disable application blocking and a control to add/remove applications.
  - `WebBlockingControl.xaml` has toggles to enable/disable website blocking and hide/show YouTube elements. It also has a control to add/remove websites and a toggle to toggle between blocking and allowing websites.
- `StatisticsView.xaml` is very simple, it just contains a table showing the work statistics for the last 30 days. The number of days can be changed by configuration, which will be described in the configuration section.
- `SettingView.xaml` contains toggles to enable/disable timer notifications and gamification elements. It also has a combobox to change the theme of the application. These settings are located in `GeneralSettingsControl.xaml`. This allows the settings view to be extensible.
- `GlobalErrorMessage.xaml` is for displaying error messages at the top of the application. For example, if the user enters an invalid value in the configuration file, an error message will be displayed at the start of the application.

ViewModels

As described above, the view model is an intermediary between the view and the model / services and manages the flow of the application.

The view model implements properties and commands to which the view can data bind to. When the view model wants to update the data on the view, its properties must raise

the `PropertyChanged event`. By implementing the `INotifyPropertyChanged`, and raising the `PropertyChanged`, view models satistify this requirement. [25]

For collections, the `ObservableCollection<T>` is provided. This collection implements collection changed notification, which means that developers do not need to implement the `INotifyCollectionChanged` interface on collections. [25]

When a control in a view wants to perform an action, it calls the commands of the view model, which are properties that implement the `ICommand` interface. This makes view models more portable, as they are not directly dependent on events provided by the platform's UI framework [25]. For more information on how commands work, see the Commands section.

Each view is associated with its own view model. If a view contains a collection, such as a to-do list with tasks, there is also a separate view model for the item in the collection. The structure of the `ViewModel` folder is divided into the following.

- `HomeViewModels` contains view models related to the PT, the manual blocking the and the to-do list.
- `SettingViewModels` contains view models related to the general settings and the Pomodoro timer settings.
- `BlockingViewModels` contains view models related to the website blocking and the application blocking.
- `StatisticsViewModels` contains view models related to the work statistics.
- The rest of the view models are not categorised and are located in the `ViewModel` folder.

Commands

Commands are classes that implement the `ICommand` interface. Commands, as described in [25], provide a convenient way to represent actions that can be bound to controls in the user interface. They encapsulate the code that implements the action, helping to decouple it from its visual representation in the view.

The `Commands` folder has the following structure.

- `TimerCommands` contains commands related to the PT and the manual blocking, such as commands to start, stop, change presets for the PT.
- `TaskCommands` contains to-do list commands such as add, remove, complete, and delete a task.
- `SettingCommands` contains commands to save, cancel, and load settings.
- `BlockingCommands` contains commands to block websites or applications, such as adding or removing a website/application to block. There are only commands to enable or disable the blocking feature. The start/stop blocking websites/applications is triggered when the PT or the manual blocking is started.

Models

The model is used to represent the data. It contains the logic to work with the data, such as updating the data using data services.

The structure of this folder is similar to the `Commands` folder. There are timer models that represent the PT values. There are also task models, setting models, blocking models, and statistics models. The statistics models represent the work statistics.

In addition to the model, there are other data structures to simplify the model, such as the `ThemeSetting` enum to represent the colour scheme of the application.

DTOs

DTOs[3] are objects that are similar to the models used to transfer data between the database layer and the database storage. However, these objects have simple data types because they represent a database table. For example, a setting model contains a theme setting that has a `ThemeSetting` data type, while the setting DTO represents the theme setting as text.

Services

Services contain business logic that is used to perform an action. The following services are included in the application:

- **Database services** contain logic related to the database, such as adding, deleting and updating tasks in the to-do list.

  To insert or update data, database services accept models, transform them into DTO objects, and use database contexts to work with these data. When data are retrieved from a database, database contexts return DTO objects and database services transform them back into models.

- `AppConfigService.cs` loads the configuration data and validates those data.
- `TimerManager.cs` contains timer logic, such as timer countdown and timer start or stop.
- `AppBlockingService.cs` contains the logic to block applications.
- `WebBlockingService.cs` contains the logic to block websites.
- `NotificationService.cs` contains the logic that triggers the notification when the timer expires.
- `RewardManager.cs` calculates the rewards for the user in coins according to the user's work time.

Stores

Stores have logic to hold data/states and allow other objects to change states or be notified of changes. If an object wants to be notified of changes, it simply subscribes to a specific store event. There are six stores:

- `BlockingStore.cs` stores information about blocked websites and applications.
- `ErrorMessageStore.cs` stores error messages and notifies subscribers on new error messages.
- `PlayerStatStore.cs` stores information about gamification elements such as coins and health points.
- `StatisticsStore.cs` stores working statistics.
- `TaskStore.cs` stores tasks and notifies subscribers on task changes, such as adding a new task or removing a task.
- `TimerTimeStore.cs` stores the PT settings and notifies subscribers when PT setting changes.

[3] Data transfer objects

DbContexts

The `DbContexts` has classes for managing database operations and mapping entities. It also has a class for creating migrations at design time[4].

Migrations

Migrations are a set of changes made to the database. A migration is invoked at the start of the application to apply the changes to the database. However, in this case, it is used to prevent unexpected actions by the user. For example, if the user accidentally removes the database file, the migration will create a new database file at the start of the application.

Configurations

The application uses the Configuration Manager library to support configuration. The configuration is an XML file named `App.config`. This file contains many configuration parameters such as port (for the web extension to connect to), maximum coins, maximum statistics records, etc.

The `AppConfigService.cs` loads and validates the configuration data. If an error occurs, such as an invalid configuration parameter value or the configuration file could not be found, it uses the default values defined directly in the `AppConfigService.cs` file.

3.3 Implementation of key functions

This section briefly describes how the following functionality has been implemented.

Pomodoro timer - At the beginning, the PT service loads the settings from the database and creates a countdown timer event. When the user starts the PT, the PT service starts the countdown timer event. If the user enables websites/applications blocking, it also starts the blocking.

When the timer expires, the PT service unblocks the websites/applications and adds work statistics. If the gamification feature is enabled, the PT service will also add coins and health points for the user. Note that "the PT service adds/blocks ..." means that the PT service uses other services or stores to perform this action.

To-do List - Tasks in the to-do list contain information about the task status (todo, done, failed). The application loads all tasks and then categorises them into the to-do list, done list, and failed list.

When a task is completed/failed, it is moved to the done/failed list. Basically, the status and completion date of the task are updated. When the task undo action is triggered, the task status is set to todo, and the completion date is removed, resulting in the task being moved to the todo list. When the task is deleted, the task is completely removed from the database.

If the user has the gamification feature enabled, the user will also gain or lose coins (or health points) based on whether they have completed or failed the task.

Website blocking - When the application starts, the website blocking service starts the WebSocket server for extensions to connect to. The server is started on the port defined in the configuration file.

When blocking, the blocking service sends a list of blocked websites and settings, such as YouTube blocking, to all connected extensions. Based on the information received from the

[4]The migration can be created without starting the application

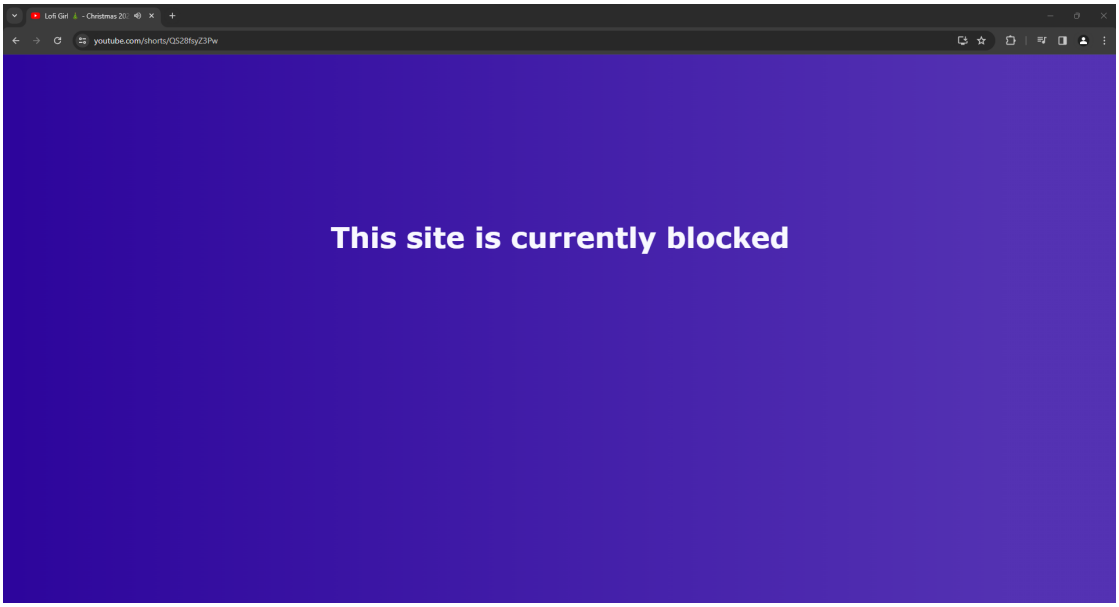

Figure 3.1 Website block screenshot

server, the web extension blocks websites. It blocks the website by creating a layer that covers the website and prevents the user from interacting with it. This layer can be seen in the figure 3.1. To unblock, the extension simply removes this layer.

YouTube blocking - If YouTube blocking is enabled, the web extension will hide videos recommended by YouTube. Note that the search bar is still available, so the user can search for videos to watch.

Based on the URL, the web extension knows if the user is on YouTube or not. If "Hide YouTube home feed" is enabled, the extension will hide the `page-manager` element that contains recommended videos on the YouTube home page. If "Hide YouTube sidebar" is enabled, the extension will hide the `related` element that contains recommended videos on the video sidebar. Figure 3.2 shows a YouTube video without blocking and figure 3.3 shows a YouTube video with sidebar blocking.

Application blocking - The application blocking service blocks user-defined applications by stopping their process when they are running. It has a timer tick event that scans and blocks these processes at a fixed interval defined in the configuration file.

To prevent users from blocking important system processes such as Task Manager, the blocking service maintains a list of important system processes. If the process is on this list, the service will not block it.

Statistics - When the PSW finishes, the PT service calls the update function via the statistics store. The statistics store then adds or updates the working time to the database via other services and notifies the statistics view model of the changes. The changes are then displayed to the user in the statistics section.

Discord status - The Discord GameSDK contains too many warnings and is unstable. For this reason, the Discord status feature described above has been removed.

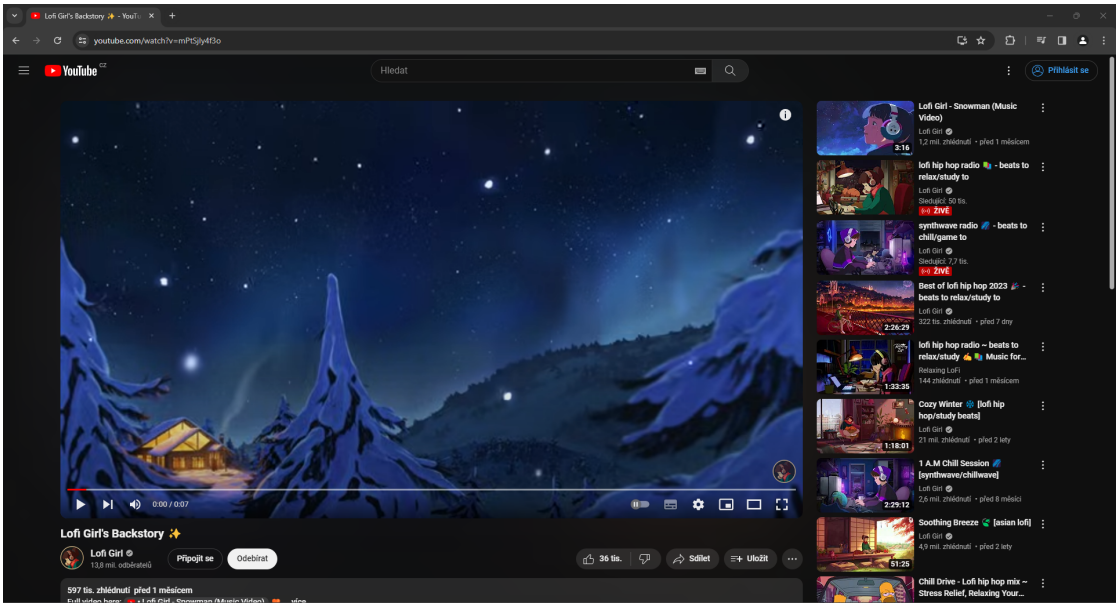

Figure 3.2 YouTube screenshot – A video on YouTube without blocking

Figure 3.3 YouTube screenshot – A video on YouTube with sidebar blocking

3.4 User manual

The complete user manual can be found in the enclosed medium, as it contains many images. This section only briefly describes the information contained in the manual.

The manual will answer the following questions:

- How to change the PT preset?
- How to pause the blocking during manual blocking or PT?
- How to change the PT work time and pause time?
- How to start manual blocking?
- How does the to-do list work (focus task, undo task, fail task)?
- How does the gamification feature work?
- How to block or allow websites?
- How to hide YouTube elements?
- How to block applications?
- How do the statistics work?
- How to change the colour scheme of the application?
- How to turn off the PT notification?

Chapter 4
Testing

4.1 Unit testing

Unit testing is important because it can find many problems and bugs in the application. Another advantage of unit testing is that it is fast and repeatable. This makes it possible to run these tests every time, when new features are added. The tests check that everything works as it should.

In this project, not all components were optimally designed for testing. As a result, only some services, stores, and models were tested. Other components, such as view models and commands, were not tested. However, since the application is tested by users as a whole, these components have been partially tested by the users.

The test project is located in the same project solution as the main project. The main components of the test project are listed below. The structure of the test project is similar to the structure of the main project. However, as described above, not all components can be tested.

```
Desktop
├── ImprovementFocusTool
└── ImprovementFocusTool.Tests
    ├── Models
    ├── Services
    ├── Stores
    └── ...
```

4.2 Usability testing

Usability testing, as described in [40], is a popular UX[1] research methodology. In a usability testing session, a researcher asks a participant to perform tasks, usually using one or more specific user interfaces. As the participant completes each task, the researcher observes the participant's behaviour and listens for feedback.

Why is usability testing important? As described in [40], usability testing can identify problems in the design of the product or service. It helps to learn about the target user's behaviour and preferences. It also helps to discover opportunities to improve the design.

[1] User Experience

4.2.1 Testing process

The tests were scheduled to take place during the FIT CTU Open Day on 25. 11. 2023. This day is a good opportunity to test the application, as there will be many students who are potential users. A test scenario and a user experience questionnaire have been prepared for the test. The testing process should be fast in order to test as many people as possible. Therefore, the test scenario and the user experience questionnaire are very short.

The whole testing process was managed by the author, who helped out at the Software Engineering booth. If someone was interested in studying software engineering or had some questions, the author answered their questions. Then the author asked them if they could do a short usability test of the author's application, which is the result of his bachelor thesis. Users were allowed to try and test the application according to the prepared test scenario. However, not all users tested the application as described in the test scenario. Some users even tested other cases than described in the scenario, such as blocking applications and changing the colour scheme of the application in the settings section.

During testing, the author listened and recorded feedback. After testing, users were asked to complete a short questionnaire, which is in the enclosed medium. The questionnaire shows that 21 people completed the questionnaire. However, not everyone completed the questionnaire after testing, so the number of users who tested the application is slightly higher than 21.

Test scenario

The objective of this scenario is to see if users can use the core functionality of the applications. The scenario verifies that users are able to start the PT, change the PT time, work with the to-do list, and block websites. The scenario is as follows.

1. Set the Pomodoro timer for 10 minutes and start the timer.
2. Add two tasks and complete them.
3. Try to undo the completion action.
4. Block the website `https://www.facebook.com/` and check that it is actually blocked.
5. Remove the website `https://www.facebook.com/` from the blocked site and check that it is accessible again.

User experience questionnaire

The purpose of this questionnaire is to check whether the application is easy to use and whether users would use it or not. The questions in the questionnaire are as follows.

1. Do you find the application easy to use?
2. Would you use this application for work or study?

The results can be found in the enclosed medium.

4.2.2 Results

The results of the tests are positive. The result of the questionnaire shows that 66.7 % of the respondents found that the application is intuitive to use. 23.8 % were able to navigate in the application, but some elements were unclear to them. About 85 % of the respondents probably would use the application.

However, the application is not perfect. There are some comments from users and bugs in the application. The comments are about which elements of the application were not intuitive and where the application could be improved. All comments and bugs are listed below.

- It would be useful to add a feature to change the PT directly by clicking on the timer text. Some users did not realise at first glance that the change timer setting was in the settings button just below the timer. They thought they could change the timer directly by overwriting the timer text.
- Changing the PT time can only be done by dragging the slider. The solution is to add a function to change the time using the text box. This allows the user to enter the time directly using a keyboard.
- Many users were unaware that applications and websites are only blocked when the PT or manual blocking is started. Most users look for the “Block” button in the blocking section, which only contains settings related to blocking applications/websites. It would be useful to inform the user that they need to start PT or manual blocking to start blocking applications and websites.
- Many users were unaware that the ”Allow website blocking” toggle is used to enable website blocking when the PT or manual blocking is started. Some users did not realise that the toggle “Use allowed mode” was used to toggle between blocking selected websites or allowing selected websites. Allowing selected websites means that selected websites are allowed and others are blocked.
- Sometimes the YouTube block feature does not work correctly. In some cases, YouTube content does not hide as expected, and the user has to reload the page to see changes.

Chapter 5
Conclusion

The aim of this thesis is to create a desktop application for the Windows operating system that improves concentration and prevents procrastination while working.

To create this application, an analysis of existing solutions and their interfaces was carried out. An analysis of potential users was also carried out, and functional and non-functional requirements were defined.

After the analysis, a prototype of a user-friendly interface was designed using Figma. The architecture and database of the application were also designed.

Once the design was complete, the prototype of the desktop application and the web extension were developed. JavaScript was chosen to develop the web extension and C# with .NET was chosen to develop the desktop application. SQLite was used for database storage. The desktop application was implemented using the MVVM architecture.

Finally, the application was usability tested. The results were positive and there were comments from the users for improvement.

Future expansions

The application is already in a usable state. It contains the PT, the to-do list, the block function, etc. However, there is still room for improvement in the application. The following features can be added to the application in the future.

- **Multiplatform** - Multiplatform support for other platforms including Linux, Android, and iOS.
- **Subtasks** - The addition of subtasks allows users to have a better overview of their to-do list.
- **Integration** - Integration with other to-do list applications such as Microsoft To Do and Google Tasks. This allows users to manage their tasks in one place.
- **Interval blocking** - Block applications/websites based on a specified time interval.

Bibliography

1. LUDWIG, Petr. *Konec prokrastinace*. Jan Melvil Publishing, 2013. ISBN 978-80-87270-51-6.
2. CIRILLO, Francesco. *Pomodoro Technique – The Acclaimed Time-Management System That Has Transformed How We Work*. Maven Publishing, 2019. ISBN 978-80-7555-069-9.
3. COLD TURKEY SOFTWARE, INC. *Cold Turkey Blocker* [online]. 2023. [visited on 2023-10-28]. Available from: `https://getcoldturkey.com/`.
4. UZU, Yuya. *Pomofocus* [online]. 2023. [visited on 2023-10-28]. Available from: `https://pomofocus.io/`.
5. HABITRPG, INC. *Habitica* [online]. 2023. [visited on 2023-10-29]. Available from: `https://habitica.com/static/home`.
6. FREEDOM.TO. *Freedom: Internet, App and Website Blocker* [online]. 2023. [visited on 2023-10-29]. Available from: `https://freedom.to/`.
7. *StayFocusd: Block Distracting Websites* [online]. 2023. [visited on 2023-10-29]. Available from: `https://www.stayfocusd.com/`.
8. MICROSOFT. *Windows Clock* [online]. 2023. [visited on 2023-10-29]. Available from: `https://apps.microsoft.com/detail/windows-clock/9WZDNCRFJ3PR?hl=en-gb&gl=US`.
9. MICROSOFT. *Microsoft To Do: Lists, Tasks & Reminders* [online]. 2023. [visited on 2023-10-29]. Available from: `https://www.microsoft.com/cs-cz/microsoft-365/microsoft-to-do-list-app?rtc=1`.
10. *Unhook YouTube* [online]. 2023. [visited on 2023-10-29]. Available from: `https://unhook.app/`.
11. DISCORD INC. *Discord* [online]. 2023. [visited on 2023-11-20]. Available from: `https://discord.com/company`.
12. JAKOB, Nielsen. *10 Usability Heuristics for User Interface Design* [online]. 2020. [visited on 2023-11-06]. Available from: `https://www.nngroup.com/articles/ten-usability-heuristics/`.
13. MICROSOFT. *What is .NET? Introduction and overview* [online]. 2023. [visited on 2023-12-04]. Available from: `https://learn.microsoft.com/en-us/dotnet/core/introduction`.
14. MICROSOFT. *C#* [online]. 2023. [visited on 2023-12-11]. Available from: `https://dotnet.microsoft.com/en-us/languages/csharp`.
15. MICROSOFT. *Why Choose .NET?* [Online]. 2023. [visited on 2023-12-11]. Available from: `https://dotnet.microsoft.com/en-us/platform/why-choose-dotnet`.
16. MICROSOFT. *What is .NET, and why should you choose it?* [Online]. 2023. [visited on 2023-12-10]. Available from: `https://devblogs.microsoft.com/dotnet/why-dotnet/`.

17. MICROSOFT. *An overview of Windows development options* [online]. 2023. [visited on 2023-12-26]. Available from: https://learn.microsoft.com/en-us/windows/apps/get-started/?tabs=winappsdk-winui%2Cnet-maui#windows-platforms.
18. MICROSOFT. *Data binding overview (WPF .NET)* [online]. 2023. [visited on 2023-12-31]. Available from: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/data/?view=netdesktop-8.0#what-is-data-binding.
19. *Difference between WPF and WinForms* [online]. 2022. [visited on 2023-12-26]. Available from: https://www.geeksforgeeks.org/difference-between-wpf-and-winforms/.
20. CHATGPT. *WPF vs WinForms* [online]. 2023. [visited on 2023-12-26]. Available from: https://chat.openai.com/share/5206ce5c-1ba0-443c-845e-6003cc812f99.
21. MICROSOFT. *What's new in .NET 8* [online]. 2023. [visited on 2023-12-26]. Available from: https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-8#windows-presentation-foundation.
22. *Material Design In XAML* [online]. 2023. [visited on 2023-12-26]. Available from: http://materialdesigninxaml.net/.
23. INC., Figma. *Figma* [online]. 2024. [visited on 2024-01-10]. Available from: https://www.figma.com/.
24. MICROSOFT. *Patterns - WPF Apps With The Model-View-ViewModel Design Pattern* [online]. 2009. [visited on 2023-12-28]. Available from: https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern#why-wpf-developers-love-mvvm.
25. MICROSOFT. *Model-View-ViewModel (MVVM)* [online]. 2022. [visited on 2023-12-30]. Available from: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm#the-mvvm-pattern.
26. MICROSOFT. *The MVVM Pattern* [online]. 2012. [visited on 2023-12-31]. Available from: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/hh848246(v=pandp.10)?redirectedfrom=MSDN.
27. *SQLite is a Self Contained System* [online]. 2023. [visited on 2024-01-01]. Available from: https://www.sqlite.org/selfcontained.html.
28. *About SQLite* [online]. 2023. [visited on 2024-01-01]. Available from: https://www.sqlite.org/about.html.
29. MICROSOFT. *How to reset the Hosts file back to the default* [online]. 2022. [visited on 2024-01-02]. Available from: https://support.microsoft.com/en-us/topic/how-to-reset-the-hosts-file-back-to-the-default-c2a43f9d-e176-c6f3-e4ef-3500277a6dae.
30. *A Guide to Hosts File and Using it to Block Websites on Windows, Linux, and Mac OS* [online]. 2022. [visited on 2024-01-02]. Available from: https://medium.com/@prcooltechzone/a-guide-to-hosts-file-and-using-it-to-block-websites-on-windows-linux-and-mac-os-4edc44fbd915.
31. MOZILLA CORPORATION. *WebSocket* [online]. 2024. [visited on 2024-01-02]. Available from: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket.
32. ANANDK@TWC. *How to turn off App Notifications using Registry Editor in Windows 11/10* [online]. 2022. [visited on 2024-01-02]. Available from: https://www.thewindowsclub.com/how-to-turn-off-app-notifications-using-registry-editor-in-windows-10.
33. DISCORD INC. *Introducing Rich Presence* [online]. 2023. [visited on 2024-01-02]. Available from: https://discord.com/developers/docs/rich-presence/how-to.
34. MICROSOFT. *Visual Studio 2022* [online]. 2024. [visited on 2024-01-05]. Available from: https://visualstudio.microsoft.com.

35. JETBRAINS S.R.O. *ReSharper* [online]. 2024. [visited on 2024-01-05]. Available from: https://www.jetbrains.com/resharper/.

36. MICROSOFT. *Introduction to the MVVM Toolkit* [online]. 2023. [visited on 2024-01-06]. Available from: https://learn.microsoft.com/en-us/dotnet/communitytoolkit/mvvm/.

37. MICROSOFT. *ConfigurationManager Class* [online]. 2023. [visited on 2024-01-06]. Available from: https://learn.microsoft.com/cs-cz/dotnet/api/system.configuration.configurationmanager?view=dotnet-plat-ext-8.0.

38. MICROSOFT. *Entity Framework Core* [online]. 2021. [visited on 2024-01-06]. Available from: https://learn.microsoft.com/en-us/ef/core/.

39. MICROSOFT. *Database Providers* [online]. 2023. [visited on 2024-01-06]. Available from: https://learn.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli.

40. MORAN, Kate. *Usability Testing 101* [online]. 2023. [visited on 2024-01-09]. Available from: https://www.nngroup.com/articles/usability-testing-101/.

Content of enclosed CD

- `readme.txt` ..... a brief description of the content of the media
- `exe` ..... directory with executable form of implementation
- `src`
  - `impl` ..... implementation source codes
    - `Desktop` ..... source code of the desktop application
    - `WebExtension` ..... source code of the web extension
  - `thesis.zip` ..... source code of the thesis in format LaTeX
- `text`
  - `thesis.pdf` ..... the thesis in PDF
- `resources`
  - `Extended_Database_Diagram.pdf`
  - `Productivity_Improvement_Program_Questionnaire.pdf`
  - `User_Experience_Questionnaire.pdf`
  - `User_Manual.pdf`