POLISH-JAPANESE ACADEMY OF INFORMATION TECHNOLOGY

Faculty of Computer Science

Name of your Specialization's Department
Name of your Specialization

Your Name — s#####

Your Carefully Selected and Expressive Thesis Title

Master's degree / Bachelor's degree thesis written under the supervision of:

Supervisor's Name
Auxiliary Supervisor's Name

Warsaw, November 2025

Abstract

Shortly describe what kind of problem has been inspected in the thesis, and how it has been solved. The abstract should be between 400 and 1500 characters, including spaces. **Thesis written in English must also include abstract and keyword translations to Polish**. Abstract should usually be written **towards the end** of your thesis work, since that is the time when you best know what (and how) exactly has been accomplished. **Pay extra attention and spend some extra time when developing an abstract.** This is due to the fact that most people will be glancing over your abstract in order to determine whether it is worth it for them to delve deeper into your work. This is the place where you need to attractively explain what can be found in this thesis. Do not introduce additional paragraphs in the abstract. The rest of this document describes general rules for writing theses documentations in PJAIT.

Keywords

Keywords · can · be · both · single- or multiple-word phrases · At · least · 3 · keywords · are · necessary · Threat · them · as · tags · Your · thesis · must · be · searchable · using · them · Separate them by using the `#sym.dot.op` syntax.

Table of Contents

1 Introduction

In **1 Introduction** you should explain the motivation behind your work, broad context of the work, and your inspirations. Explain the problem and how the path that you took to deal with it.

A good introduction is usually written quite late in the thesis' development, because throughout your work some important details may be subject to change.

Between chapters and sections (or subsections) always introduce few lines worth of a paragraph to introduce the topic. Avoid "gluing" higher-level and lower-level sections. Always separate them using some text.

1.1 Goals

Introduce the goals of the work. What kind of research and/or tool this work is supposed to produce.

1.2 Conventions

Explain the conventions used in your work. For example, how do you deal with inline code or with code listings that contain lines which are too long to be rendered as a single one in this document.

1.3 Results

Briefly introduce the results of your work.

1.4 Document structure

Explain how is your document structured by briefly summarizing all the chapters. This sections is usually filled in last.

1.5 Example: Chapters, sections, and subsections

Together they are called *headings*. Use appropriate commands to introduce them. Use enumerations or itemizations when it makes sense—it elevates the readability of your work. For example, here is how we introduce specific types of sections:

1. Top-level—*chapters*. Use `= Chapter Name` to introduce them.
2. Lower-level—*sections*. Use `== Section Name` to introduce them. For example, this very paragraph is a part of a section. Namely—**1.5 Example: Chapters, sections, and subsections**.
3. Lowest-level—*subsections*. Use `=== Subsection Name` to introduce them.

Headings aren't titles, so Don't Use Title Case.

1.6 Styling

Here we will briefly explain the appropriate styling for theses. For starters, let us explain the difference between different dashes:

- Hyphens (-) are inputted by using a single `-` character. They are used to form interconnected words such as *low-level* or *first-person*.
- En dashes (–) are inputted by using `--`. They are very rarely used. Mainly for specifying ranges, e.g.:
  - pages 10–26
  - we are open 6:00 a.m.–9:00 p.m.
  - saying such things was illegal in 1670–1680
- Em dashes (—) are inputted by using `---`. They sometimes replace commas or changes of text direction, e.g.:
  - This is very difficult—we are going to need to step down.
  - My good friend—John Doe—turned out very successful.

1.6.1 Form

It is best to use infinitives for describing the work done. For example:

> Sample Typst template has been prepared to make it easier for PJAIT's students to write their theses documentations.
>
> — Some source

First-person pronouns should be avoided:

> I created a simple Typst template to make it easier for PJAIT's students to write their theses documentations.
>
> — Some other source

In rare occasions, it might be the case that the author of the work wants to introduce a personal opinion. In those cases, a third-person form should be used:

> Author of this Typst template is certain that this document will be of much help for PJAIT's students.

1.6.2 Quoting

In **1.6.1 Form**, the colored sections were *quoted*. Styles of such quotations are automatically handled by using the appropriate `#quote` function.

1.6.3 Colors

The usage of colored text should usually be avoided (except for code listings and inline code) unless there is a very strong reason to make an exception. Consistency matters and it is usually better to use `_the standard tool for emphasis_` rather than colors.

1.6.4 Emphasis

Since this document uses (and thus so should your thesis) serif font, use *italics* for gentle emphasis and **bold** for heavier emphasis.

Do not underline the text. Pretty much at all. [1]

1.6.5 Blocks of text

Do not introduce overly lengthy paragraphs. If it exceeds a third of a page, break it down. Use enumerations, itemizations, figures, inline code, code listings, etc., which make the text more attractive. Do not overdo it though.

If a given chapter or (sub)section takes more than 3 pages, it is appropriate to divide into couple of lower-level sections. Each chapter (top-level) starts on a new page (preferably oddly-numbered[1]). This does not concern lower-level sections. An exception can be made if a significant number of figures, listings, etc. are present.

1.6.6 Typos

Pay attention to the quality of your text—both in terms of substantive value and form of the text. Use correct punctuation and a spell checker to detect typos.

1.6.7 Bibliography details

Bibliography items must be correctly referenced. Use the appropriate template and `.bib` file. Pay attention to how those files are structured. Use appropriate types, and supply necessary details.

Use a standardized package for bibliography and cite when appropriate. [2] Each bibliography position must be referenced in the text. It is unacceptable to list works in the bibliography that are

[1] So that when you physically read the printed thesis, new chapter is always on the right-hand side. This is handled automatically.

not mentioned in your work. You must show how exactly the given cited position was impactful for this work.

Notice that both the first citation and the chapter references are clickable and automatically generated. We use `@thingToReferenceOrCite` syntax for that.

2 Content

Be direct. The thesis should be *as short as possible, but not shorter than that*. Meaning that if you can be more concise without sacrificing readability or quality, you should be.

Here you should explain in detail the context of your work, state-of-the-art of similar tools, what is different about your approach, etc. If it gets lengthy, you can divide it into multiple chapters.

2.1 How much to cite?

It is impossible to answer this question precisely. A rule of thumb presents itself as follows:

- Copy-pasting an entire page of academic work without any citations is a **crime** (copyright).
- Each important aspect of your work that is not directly created by you, should be backed by citations.
- In the section dedicated to alternative sources / state-of-the-art, it is permitted to introduce grouped citations, e.g., by introducing a short summary and then providing many references at the same time.
- It is permitted to omit the citation of internet webpages if authors of referenced texts are unknown. In such situations you should reference the source broadly, e.g., "On IBM's website it can be seen that [...]."

Theses that were prepared and submitted without the supervisor's active overseeing of the process of creating them will be rejected.

2.2 How to structure my work?

It is best to split your chapters[2] into separate files. Notice how this project consists of `contents` directory with `introduction.typ` and `content.typ` inside, which are then `#included` in the `main.typ` file. This makes it very easy to change the ordering, separate different topics, and collaborate.

2.2.1 Manual numbering (of pretty much anything)

Do **not** include any numbering in the names of the files. E.g., notice that the files are named `introduction.typ` and `content.typ`, **not** `1-introduction.typ` and `2-content.typ`. You do **not** want to create such dependencies. Typst does the heavy-duty job of automatic numbering and ordering. Do not interfere with it without a good reason. What will happen if you spend a lot of time to number your files, and then realize that you need to introduce a new file/chapter in the middle? You're a programmer—apply and respect good programming practices.

2.3 How to introduce other media content?

Typst provides a wide array of options to deal with inputting things like:

- code listings
- tables
- images
- equations

Let us inspect couple of examples.

2.3.1 Code listings

For inline code use `` `backticks` `` (the character above your Tab key), and to introduce code listings use ```` ```triple backtics``` ```` (preferably with specified language). For example, to render Listing 1,

Listing 1: Example of Kotlin code rendered as a code block.

```
fun main() {
    val text = getStringOrNull()

    println(text?.length ?: "nothing")
```

[2] Chapters. Not sections, not subsections. Entire chapters.

```
}
```

code from Listing 2 was used. Notice that listings' labels (with captions) are placed above them.

Listing 2: Code used to create Listing 1.

````
#figure(
    caption: [Example of Kotlin code rendered as a code block.]
)[
```Kotlin
fun main() {
    val text = getStringOrNull()

    println(text?.length ?: "nothing")
}
```
]<listing-Example-of-Kotlin-code-rendered-as-a-code-block>
````

Referencing listings is done using `@thisFamiliarSyntax`. It searches for labels, which in this are put after the listings (e.g., `@listing-Example-of-Kotlin-code-rendered-as-a-code-block` creates the following reference: Listing 1).

Currently, this template uses the Codly package [3], [4], but it's acceptable to present and suggest alternatives (or at least help out with nicer rendering...).

2.3.2 Tables

Tables are very easy to make. To make a table that demonstrates how your final grade issued on a diploma is calculated, refer to Listing 3. The rendered table is Table 1.

Symbol	**Meaning**
a	Your GPA.
b_1	Thesis' grade by supervisor.
b_2	Thesis' grade by aux. supervisor.
c	Grade for thesis presentation.
d_1	Grade for answers to question #1 from diploma exam.
d_2	Grade for answers to question #2 from diploma exam.
d_3	Grade for answers to question #3 from diploma exam.
Final grade: $\frac{a}{2} + \frac{b_1+b_2+c}{12} + \frac{d_1+d_2+d_3}{12}$.	

Table 1: Components and calculations of your final grade issued on the diploma.

Listing 3: Example commands used to input a table.

```
#figure(
    caption: "Components and calculations of your final grade issued on
    the diploma."
)[
    #table(
        columns: (auto, 1fr),
        inset: 8pt,
        table.header([*Symbol*], [*Meaning*]),
        align: (center, left),
```

```
        [$a$],[Your GPA.],
        [$b_1$],[Thesis' grade by supervisor.],
        [$b_2$],[Thesis' grade by aux. supervisor.],
        [$c$],[Grade for thesis presentation.],
        [$d_1$],[Grade for answers to question \#1 from diploma exam.],
        [$d_2$],[Grade for answers to question \#2 from diploma exam.],
        [$d_3$],[Grade for answers to question \#3 from diploma exam.],
        table.cell(colspan: 2)[Final grade:
        $a/2+(b_1+b_2+c)/12+(d_1+d_2+d_3)/12$.]
    )
]<tab-Final-grade-calculations>
```

It is worth noting that the calculated grade according to Table 1 may be fractional. How does it map to the actual, discrete set of possible grades? Consult Table 2, which, for showcase purposes, uses slightly different style.

Scored grade range	Resulting grade
$[2.00; 3.00)$	2.0
$[3.00; 3.40)$	3.0
$[3.40; 3.80)$	3.5
$[3.80; 4.20)$	4.0
$[4.20; 4.50)$	4.5
$[4.50; 5.00)$	5.0

Table 2: Mapping of scored grade to the actual grade issued on the diploma.

2.3.3 Images (figures)

Including images is very easy. Figure 1 is included as-is, while Figure 2's width was set to 50% (height was scaled automatically).

Figure 1: Typst's logo.

Figure 2: Smaller Typst's logo.

Commands used to produce the above output are presented in Listing 4. Again, like with tables, images have labels placed below them.

Listing 4: Example commands used to input images.

```
#figure(
    caption: "Typst's logo."
)[
    #image("assets/Typst.png")
] <img-raw-Typst-logo>

#figure(
    caption: "Smaller Typst's logo."
)[
    #image("assets/Typst.png", width: 80%)
] <img-smaller-Typst-logo>
```

2.3.4 Equations

Use LaTeX-like syntax. For inline equations, wrap them in a pair of dollar signs (`$`). For example, this: `$a  x^2+b  x+c=0$` will produce: $ax^2 + bx + c = 0$. For numbered equations that are displayed separately, introduce additional space between the dollar sign and the equation. For example, typing `$ a x^2+b x+c=0 $` (notice the additional space after the first and before the last `$`—`$ ` and ` $`) will produce:

$$ax^2 + bx + c = 0 \tag{1}$$

Of course, they are numbered (thus should be referencable), so instead of `$ a x^2+b x+c=0 $` one should type `$ a x^2+b x+c=0 $ <eq-Quadratic-example>` so that it's rendered like this:

$$ax^2 + bx + c = 0 \tag{2}$$

and referenced like this: Equation 2.

2.3.5 Figure functions

Notice that in case of listings, tables, and images, they were wrapped inside a `#figure` call. This makes them referencable, have captions, and be correctly labeled. Without the `#figure` function call, they are rendered as-is, without being correctly labeled. There are, of course, valid use cases for that—the university's logo displayed on the title page is an example of such construct. Professionally, images are labeled as *figures*, so make sure the distinction is understood.

Bibliography

[1] M. Butterick, "Practical Typography, 2nd Edition." Accessed: Nov. 11, 2025. [Online]. Available: https://practicaltypography.com/

[2] Typst, "Typst bibliography documentation." Accessed: Nov. 11, 2025. [Online]. Available: https://typst.app/docs/reference/model/bibliography/

[3] Dherse, "Codly 1.3.0 package." Accessed: Nov. 11, 2025. [Online]. Available: https://typst.app/universe/package/codly/

[4] Dherse, "Codly 1.3.0 documentation." Accessed: Nov. 11, 2025. [Online]. Available: https://raw.githubusercontent.com/Dherse/codly/main/docs.pdf

List of Listings

List of Images

List of Tables