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
Reviewer's Name

Warsaw, January 21, 2026

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Nazwa katedry
Nazwa specjalizacji

Your Name — s#####

Twój starannie dobrany i ekspresywny tytuł pracy dyplomowej

Praca inżynierska / Praca magisterska napisana pod kierunkiem:

Supervisor's Name
Reviewer's Name

Warszawa, 21 stycznia 2026

Abstract

The abstract should be between 400 and 1500 characters, including spaces. **Thesis written in English must also include abstract and keywords translation 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.

Streszczenie

Tłumaczenie streszczenia.

Słowa kluczowe

Tłumaczenie słów kluczowych.

Table of Contents

1 Introduction

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

A good introduction is usually written quite late in the thesis' development, because throughout your work process 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 reader to the new topic. Avoid "gluing" higher-level and lower-level headings. 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.

What business or academic value has or will your work produce?

1.2 Conventions

Explain the conventions used in your work. For example, describe 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.

Another example would be a description of your very own notation you developed in your pioneering work.

An example of a convention would be that in this document, chapters that contain the † symbol are "this-document-specific", and should *not* appear in your thesis. The rest of them (without †) should be included in your document.

1.2.1 Glossary of terms

If you are using niche terms (or ones that you either invented or adapted in a non-standard way), this is the place to explain their meaning. In other words, here you provide definitions.

1.3 Results

Briefly introduce the results of your work. What was accomplished and what was not.

1.4 Document structure

Explain how is your document structured by briefly summarizing all the chapters. This section 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 are not titles, so Do Not 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.

1.6.1 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.

1.6.2 Form and style †

It is preferable to use infinitives to describe 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

On rare occasions, the author may wish to introduce a personal opinion. In such 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.

Contractions (*aren't*, *can't*, *wouldn't*) should be avoided. They are considered informal. If you cannot force yourself to not use them, use a `'[^ s]` regex to locate them in your project and replace them one by one.

1.6.3 Quoting †

In **1.6.2 Form and style †**, the colored sections were *quoted*. Styles of such quotations are automatically handled by using the appropriate `#quote` function.

1.6.4 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.

For example, you should not be using colors like they were used in **1.6.2 Form and style †**.

1.6.5 Emphasis †

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

Do <u>not</u> underline the text. Pretty much at all. [1]

In addition to that, try to avoid combining said styles. ***It may be surprising, but this style, instead of engaging the reader's attention, tends to make them weary and inclined to skip this section altogether. Do yourself a favor and answer this question—which part stands out more: This one?***

Or this one?

1.6.6 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 it 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.

1.6.7 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.

1.6.8 Customizing the style †

Want to change the font? Maybe the font size? Letter spacing? Margins? Table styles? Fine, but consult your supervisor first. Some things are subject to personal taste, while others are a non-negotiable standard. Between those, there lies a grey area of things that *may appear "cool"*, but could possibly be detrimental to good typography.

1.6.9 Bibliography details †

Bibliography items must be correctly referenced. Use the appropriate template and a `.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 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.

---

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

2 Context

The context of the project refers to the broader background and environment in which the research is conducted or the project is implemented. It helps the reader understand why the given topic was undertaken and which external factors may influence the development and functioning of the system being designed.

Explain how your work will influence the target industry.

2.1 Related works and alternatives

Introduce other applications and tools that could be compared to or used with your work.

3 Functional requirements

Functional requirements, in the context of an application, define the specific features and actions the application should perform in response to particular user requests or under certain conditions. These are clearly defined specifications describing the system's behavior, such as enabling login, adding products to a shopping cart, or processing payments.

In short, functional requirements define *what* the application should do, not *how* it should do it. They form the basis for designing, coding, and testing the application.

This chapter should be written **from the client's perspective**, i.e., how the user can interact with the application. It may be divided into subsections or presented as a bullet-point list. It is important that the descriptions are comprehensive and include all necessary information.

In addition to that, formulating those requirements helps your supervisor correctly assess the scope of your work.

4 Non-functional requirements

Non-functional requirements, in the context of an application, relate to aspects concerning the system's quality and characteristics rather than its direct functions. They address areas such as performance, security, availability, scalability, and the user interface. In practice, they define how well the application performs its functions. Examples include system response time, the maximum number of concurrent users, and compliance with specific standards.

Non-functional requirements are essential for ensuring user satisfaction and the application's success on the market.

5 Use cases

Use cases allow for:

- quick estimates regarding project's complexity and scope;
- easy and reliable way to inspect the details of the user-application interaction.

Since even very small applications may consist of a great number of use cases, it is recommended to provide many smaller sub-diagrams rather than one, big, complete diagram.

Some applications (e.g., CLI tools or programming libraries) are less suitable for describing their features using use cases. Those scenarios should be handled individually. For example, a CLI application may have a **Usage** or **Commands and grammar** chapter, where you comprehensively explain how a given tool can be used. Sort of like a `--help` flag.

5.1 Selected use case scenarios

For some use cases, it is worth to introduce detailed scenarios. It is usually the case for complicated use cases, or use cases that are the main features of the applications.

Do *not* include use case diagrams or use case scenarios as raster images (screenshots / `.png` / `.jpeg`, …). Use vector graphics or a dedicated Typst tool that renders them.

6 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.

6.1 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.

6.2 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.

6.2.1 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 are a programmer—apply and respect good programming practices.

This also applies to enumerations or lists (itemizations). To introduce an enumeration, use `+`:

Listing 1: Example of enumeration creation.

```
1  + first item
2    + first subitem
3    + second subitem
4  + second item
5  + third item
```

Listing 1, when rendered, looks as follows:

1. first item
    1. first subitem[3]

---

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

2. second subitem
2. second item
3. third item

Using + over manual numbering has a huge advantage—it makes it so that adding a new element in the middle does not require changing the numbering of trailing items.

Use - for unordered lists (itemizations):

Listing 2: Example of itemization creation.

```
1  - first item
2    - first subitem
3    - second subitem
4  - second item
5  - third item
```

Rendered:

- first item
  - first subitem[4]
  - second subitem
- second item
- third item

6.3 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.

6.3.1 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 3,

Listing 3: Example of Kotlin code rendered as a code block.

```
1  fun main() {
2      val text = getStringOrNull()
3
4      println(text?.length ?: "nothing")
5  }
```

code from Listing 4 was used. Notice that listings' labels (with captions) are placed above them.

Listing 4: Code used to create Listing 3.

```
1  #figure(
2      caption: [Example of Kotlin code rendered as a code block.]
3  )[
4  ```Kotlin
5  fun main() {
6      val text = getStringOrNull()
7
8      println(text?.length ?: "nothing")
```

---

[3]To some, using the exact same numbers for a sublist does not look elegant, or even good. Fortunately, it is very easy to use a `#set` rule to change change this appearance. See: https://typst.app/docs/reference/model/enum#parameters-numbering

[4]By default, this template already changes the default look of nested itemizations.

```
 9  }
10  ```
11  ]<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 3).

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

6.3.2 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 5. The rendered table is Table 1.

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

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: $\frac{a}{2} + \frac{b_1+b_2+c}{12} + \frac{d_1+d_2+d_3}{12}$.	

Listing 5: Example commands used to input a table.

```
 1  #figure(
 2      caption: "Components and calculations of your final grade issued on the
        diploma."
 3  )[
 4      #table(
 5          columns: (auto, 1fr),
 6          inset: 8pt,
 7          table.header([*Symbol*], [*Meaning*]),
 8          align: (center, left),
 9          [$a$],[Your GPA.],
10          [$b_1$],[Thesis' grade by supervisor.],
11          [$b_2$],[Thesis' grade by aux. supervisor.],
12          [$c$],[Grade for thesis presentation.],
13          [$d_1$],[Grade for answers to question \#1 from diploma exam.],
14          [$d_2$],[Grade for answers to question \#2 from diploma exam.],
15          [$d_3$],[Grade for answers to question \#3 from diploma exam.],
16          table.cell(colspan: 2)[Final grade: $a/2+(b_1+b_2+c)/12+(d_1+d_2+d_3)/12$.]
17      )
18  ]<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.

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

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

6.3.3 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: Typst's logo, but smaller.

Commands used to produce the above output are presented in Listing 6. Again, like with tables, images have labels placed below them.

Listing 6: Example commands used to input images.

```
1  #figure(
2      caption: "Typst's logo."
3  )[
4      #image("assets/Typst.png")
5  ] <img-raw-Typst-logo>
6
7  #figure(
8      caption: [Typst's logo, but smaller.]
9  )[
10     #image("assets/Typst.png", width: 80%)
11 ] <img-smaller-Typst-logo>
```

Always try to include images that are in vector format—rather than raster—so that automatic scaling is better applied.

6.3.4 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 is rendered like this:

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

and referenced like this: Equation 2.

6.3.5 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 Figures

List of Tables