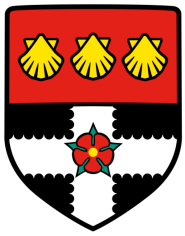

University of Reading

Department of Computer Science

Routers for LLM: A Framework for Model Selection and Tool Invocation

Ruben J. Lopes

Supervisor: Dr. Xiaomin Chen

A report submitted in partial fulfilment of the requirements of
the University of Reading for the degree of
Bachelor of Science in *Computer Science*

May 6, 2025

Declaration

I, Ruben J. Lopes of the Department of Computer Science, University of Reading, confirm that this is my own work and figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

I give consent to a copy of my report being shared with future students as an exemplar.

I give consent for my work to be made available more widely to members of UoR and public with interest in teaching, learning and research.

Ruben J. Lopes
May 6, 2025

Abstract

In the current landscape of large language models, users are confronted with a plethora of models and tools each offering a unique blend of specialisation and generality. This project proposes the development of a dynamic middleware "router" designed to automatically assign user queries to the most appropriate model or tool within a multi-agent system. By using zero-shot Natural Language Inference models, the router will evaluate incoming prompts against criteria such as task specificity and computational efficiency, and *route* the prompt to the most effective model and/or allow specific tools relevant that the model could use.

The proposed framework is underpinned by three core routing mechanisms:

- Firstly, it will direct queries to cost effective yet sufficiently capable models, a concept that builds on existing work in semantic routing **?**.
- Secondly, it incorporates a tool routing system that automatic invocation of specialised functions, thus streamlining user interaction and reducing inefficiencies currently inherent in systems like OpenAI's and Open Web UI. Furthermore this could also reduce inefficiencies in the recent reasoning models addressing the observed dichotomy between underthinking with complex prompts and overthinking with simpler queries when reasoning is manually toggled which can be costly and could cause hallucination.
- Thirdly, while the primary focus remains on model and tool routing, this work will preliminarily explore the potential application of the routing architecture as a security mechanism. Initial investigations will examine the theoretical feasibility of leveraging the router's natural language understanding capabilities to identify adversarial prompts. This includes a preliminary assessment of detection capabilities for prompt engineering attempts, potential jailbreaking patterns, and anomalous tool usage requests. However, given the rapidly evolving nature of LLM security threats and the complexity of implementing robust safeguards, comprehensive security features remain outside the core scope of this research. This aspect represents a promising direction for future work, particularly as the field of LLM security continues to mature.

By integrating these mechanisms, the research aims to pioneer a more efficient, modular, and secure distributed AI architecture. This architecture not only optimises resource allocation but also reinforces system integrity against emerging adversarial threats, thereby contributing novel insights into the development of next-generation LLM deployment strategies.

Acknowledgements

An acknowledgements section is optional. You may like to acknowledge the support and help of your supervisor(s), friends, or any other person(s), department(s), institute(s), etc. If you have been provided specific facility from department/school acknowledged so.

Contents

List of Figures

List of Tables

List of Abbreviations

SMPCS	School of Mathematical, Physical and Computational Sciences

Chapter 1

Introduction

In the past few years the landscape of large language models has expanded dramatically, with many domain specific as well as general-purpose agents emerging across domains such as healthcare (like Med-PaLM 2 and BioGPT), coding (like CodeLlama and GitHub Copilot), and research (like Claude Opus and GPT-4o). Organisations that provide inference as a service now face complex trade-offs between cost, latency, and capability: for example, GPT-4.5 can cost up to $75 per million tokens compared with just $0.15 for gemini-2.5-flash[1][2]. Although these models could be vastly different in terms of capability, the problem organisations face is determining *when* to deploy premium models versus more cost-effective alternatives for a given task / prompt. This suggests a need for intelligent routing systems that can analyse incoming prompts and direct them to the most appropriate model based on task complexity, required capabilities, and cost considerations.

Inspired by lower level (transformer-embedded) "router" such as the one employed by mistral for their Mixtral (MoE) model the goal of this was to allow for a more distributed, higher level prompt based routing between a verity of models with varying levels of cost and complexity.

1.1 Problem statement

The proliferation of large language models has created a complex ecosystem where selecting the optimal model for a given task has become increasingly challenging.

Existing multi-agent routing systems reveal several shortcomings. First, many current routers rely on **manual configuration**. For example, Both Open AI's as well as Open WebUI's chat interface require explicitly toggling of tools/selection of agents from users, and listing models to skip, on a per-chat basis. This manual toggling is brittle. Second, LLM-based routers can suffer from reasoning **inefficiencies**. Recent studies identify *"underthinking"* (prematurely abandoning good reasoning paths) and *"overthinking"* (generating excessive, unnecessary steps) in modern LLMs. For instance, Wang *et al.* (2025) find that top reasoning models often switch thoughts too quickly – an *"underthinking"* effect that hurts accuracy[3]. Conversely, Kumar *et al.* demonstrate how even simple queries can be made to "overthink" (spending many tokens on irrelevant chains of thought) without improving answers[4]. Both phenomena imply wasted tokens. Finally, prompt interpretation remains imperfect: ambiguous

[1] https://sanand0.github.io/llmpricing/
[2] https://artificialanalysis.ai/
[3] https://arxiv.org/pdf/2501.18585v1
[4] https://arxiv.org/pdf/2502.02542v1

or poorly phrased queries may be misrouted or require multiple LLM calls to resolve intent, leading to inefficiency.

Organisations and users face several key problems:

1. **Cost-Efficiency Trade-offs**: High-capability models like GPT-4o and Claude 3 Opus provide powerful capabilities but at significantly higher costs than simpler models. Without intelligent routing, organisations and users may unnecessarily infer to expensive models for tasks that could be adequately handled by more cost-effective alternatives.

2. **Selection Complexity**: With the dawn of function calling and Multi-modal Context Processing (MCP), most chat systems offer numerous specialised tools and functions, but determining which tools are appropriate for a given query often requires manual specification by users or developers.

3. **Computational Resource Allocation**: Indiscriminate routing of all queries to high-performance models can lead to inefficient resource allocation, increased latency, and higher operational costs for LLM providers and users.

1.2 Research Objectives

The premise of this research is to investigate whether pre-existing Natural Language Inference models such as Facebook's bart-large-mnli could be used as drop-in replacements to perform automated model selection and tool selection. Furthermore, we will examine the effectiveness of fine-tuning existing NLI models with specialised datasets designed for routing tasks.

The specific research objectives include:

- Creating a LLM Router library that can be deploy to existing systems with ease.
- Experimenting with Pretrained NLI models such as bart-large-mnli for both tool routing and model selection.
- Evaluating and assessing the accuracy the effectiveness using a set of prompts.
- Incorporate it with an existing Chatbot UI platform such as OpenWebUI.

Chapter 2

Literature Review

2.1 Large Language Models: Current Landscape

Large scale LLMs continue to grow in parameter count and capability, intensifying the trade off between performance and computational cost. Models such as OpenAI's GPT 4 and Google's Gemini 2.5 Pro deliver top tier results, but at significantly higher inference costs often 400 to 600 times more than comparable alternatives (**?**). With many state of the art models being closed source (only accessible through an API), a new wave of open weight and open source models has emerged. These models make it easier for individuals and companies to self host, potentially lowering operational costs. For organisations offering inference as a service, open models are particularly advantageous not only for cost efficiency, but also for addressing privacy and security concerns associated with sending user prompts to third party providers.

2.2 Multi-Agent Systems and Distributed AI Architecture

Multi-agent systems (MAS) have been a subject of research and development since the 1980s. While traditional MAS research established fundamental principles by using agent communication protocols such as KQML and FIPA-ACL, the emergence of Large Language Models has transformed how these systems operate in practice.

In December 2023, Mistral AI introduced Mixtral 8x7B, a model that employs a Sparse Mixture of Experts (MoE) architecture suggesting a promising approach which only activates a subset of a large model's "experts" per query. This gave them the edge over other models such as Llama 2 70B on most benchmarks where Inference was 6 times faster and even "matches or outperforms GPT 3.5 on most benchmarks" (**?**). While Mixtral applies routing at the model architecture level rather than through a separate system level orchestration, it demonstrated the potential for such a middle layer.

2.3 Semantic Routing Mechanisms

Several recent projects provide router-like middleware to manage multi-model access. OpenRouter.ai provides a unified API that hides model providers behind a single endpoint, dynamically routing requests across providers to optimise cost and availability. On the open-source side, RouteLLM formalises LLM routing as a machine learning problem, with results showing "cost reductions of over 85% on MT Bench while still achieving 95% of GPT-4's performance" (**?**). Another routing mechanism, Router Bench, shows promise with over 405,000 inference outcomes from representative LLMs (**?**).

On the tool routing side, landmark papers like Toolformer (**?**) demonstrate how LLMs can learn to invoke tools. At the interface level, OpenAI's Function Calling and "built-in tools" features have begun to infer tool usage directly from user prompts.

2.4 Routing Approaches

In evaluating alternatives, several decision making mechanisms currently used by LLM services are:

- **Rule-based routing:** This relies on predefined heuristic rules or configuration files (**?**). Each routing decision is directly traceable to an explicit rule (**?**). However, it often lacks contextual understanding.
- **Prompt-based routing:** This involves invoking a language model with a crafted system prompt. The model's response is passed to the relevant tool or agent.
- **Similarity Clustering based Routing:** This method leverages unsupervised clustering algorithms to group historical user queries (**?**).
- **NLI-based (zero-shot) routing:** This employs a pre-trained Natural Language Inference model for zero-shot intent classification.

2.5 Research Gap Analysis

As highlighted previously, multi-agent routing has been successfully implemented both as closed source (OpenRouter.ai) and in open source libraries such as RouteLLM. However, there remains significant opportunity for innovation in this space.

2.6 Example of in-text citation of references in LaTeX

The references in a report relate your content with the relevant sources, papers, and the works of others. To include references in a report, we *cite* them in the texts. In MS-Word, EndNote, or MS-Word references, or plain text as a list can be used. Similarly, in LaTeX, you can use the "thebibliography" environment, which is similar to the plain text as a list arrangement like the MS word. However, In LaTeX, the most convenient way is to use the BibTex, which takes the references in a particular format [see references.bib file of this template] and lists them in style [APA, Harvard, etc.] as we want with the help of proper packages.

These are the examples of how to *cite* external sources, seminal works, and research papers. In LaTeX, if you use "**BibTex**" you do not have to worry much since the proper use of a bibliographystyle package like "agsm for the Harvard style" and little rectification of the content in a BiBText source file [In this template, BibTex are stored in the "references.bib" file], we can conveniently generate a reference style.

Take a note of the commands \cite{} and \citep{}. The command \cite{} will write like "Author et al. (2019)" style for Harvard, APA and Chicago style. The command \citep{} will write like "(Author et al., 2019)." Depending on how you construct a sentence, you need to use them smartly. Check the examples of **in-text citation** of sources listed here [This Department recommends the **Harvard style** of referencing.]:

- **?** has written a comprehensive guide on writing in LaTeX [Example of \cite{}].

- If LaTeX is used efficiently and effectively, it helps in writing a very high-quality project report (**?**) [Example of \citep{}].
- A detailed APA, Harvard, and Chicago referencing style guide are available in (**?**).

This is an example of how to construct a numbered list in LaTeX, and it includes in-text, named and parenthetical citations:

1. **?** has written a comprehensive guide on writing in LaTeX.
2. If LaTeXis used efficiently and effectively, it helps in writing a very high-quality project report (**?**).

2.6.1 Reference Resources

You can find additional referencing resources from the University Library:

- https://libguides.reading.ac.uk/computer-science
- https://libguides.reading.ac.uk/citing-references/citationexamples

Chapter 3

Methodology

We mentioned in Chapter 1 that a project report's structure could follow a particular paradigm. Hence, the organization of a report (effectively the Table of Content of a report) can vary depending on the type of project you are doing. Check which of the given examples suit your project. Alternatively, follow your supervisor's advice.

3.1 Examples of the sections of a methodology chapter

A general report structure is summarised (suggested) in Table 3.1. Table 3.1 describes that, in general, a typical report structure has three main parts: (1) front matter, (2) main text, and (3) end matter. The structure of the front matter and end matter will remain the same for all the undergraduate final year project report. However, the main text varies as per the project's needs.

Table 3.1: Undergraduate report template structure

Frontmatter		Title Page Abstract Acknowledgements Table of Contents List of Figures List of Tables List of Abbreviations
Main text	Chapter 1	Introduction
	Chapter 2	Literature Review
	Chapter 3	Methodology
	Chapter 4	Results
	Chapter 5	Discussion and Analysis
	Chapter 6	Conclusions and Future Work
	Chapter 7	Refection
End matter		References Appendices (Optional) Index (Optional)

3.1.1 Example of a software/Web development main text structure

Notice that the "methodology" Chapter of Software/Web development in Table 3.2 takes a standard software engineering paradigm (approach). Alternatively, these suggested sections can be the chapters of their own. Also, notice that "Chapter 5" in Table 3.2 is "Testing and Validation" which is different from the general report template mentioned in Table 3.1. Check with your supervisor if in doubt.

Table 3.2: Example of a software engineering-type report structure

Chapter 1	Introduction	
Chapter 2	Literature Review	
Chapter 3	Methodology	
		Requirements specifications
		Analysis
		Design
		Implementations
Chapter 4	Testing and Validation	
Chapter 5	Results and Discussion	
Chapter 6	Conclusions and Future Work	
Chapter 7	Reflection	

3.1.2 Example of an algorithm analysis main text structure

Some project might involve the implementation of a state-of-the-art algorithm and its performance analysis and comparison with other algorithms. In that case, the suggestion in Table 3.3 may suit you the best.

Table 3.3: Example of an algorithm analysis type report structure

Chapter 1	Introduction	
Chapter 2	Literature Review	
Chapter 3	Methodology	
		Algorithms descriptions
		Implementations
		Experiments design
Chapter 4	Results	
Chapter 5	Discussion and Analysis	
Chapter 6	Conclusion and Future Work	
Chapter 7	Reflection	

3.1.3 Example of an application type main text structure

If you are applying some algorithms/tools/technologies on some problems/datasets/etc., you may use the methodology section prescribed in Table 3.4.

Table 3.4: Example of an application type report structure

Chapter 1	Introduction	
Chapter 2	Literature Review	
Chapter 3	Methodology	
		Problems (tasks) descriptions
		Algorithms/tools/technologies/etc. descriptions
		Implementations
		Experiments design and setup
Chapter 4	Results	
Chapter 5	Discussion and Analysis	
Chapter 6	Conclusion and Future Work	
Chapter 7	Reflection	

3.1.4 Example of a science lab-type main text structure

If you are doing a science lab experiment type of project, you may use the methodology section suggested in Table 3.5. In this kind of project, you may refer to the "Methodology" section as "Materials and Methods."

Table 3.5: Example of a science lab experiment-type report structure

Chapter 1	Introduction	
Chapter 2	Literature Review	
Chapter 3	Materials and Methods	
		Problems (tasks) description
		Materials
		Procedures
		Implementations
		Experiment set-up
Chapter 4	Results	
Chapter 5	Discussion and Analysis	
Chapter 6	Conclusion and Future Work	
Chapter 7	Reflection	

3.1.5 Legal Social and Ethical considerations

This section addresses ethical aspects of your project. This may include: informed consent, describing how participants will be informed about the study's purpose, procedures, risks, and benefits. You should detail the process used for obtaining consent and ensuring participants understand their rights.

- **Informed Consent**: If data was collected from participant, detail the process for obtaining consent and ensuring participants understand their rights.
- **Confidentiality and Privacy**: Explain measures taken to protect participants' data and maintain confidentiality. Discuss how data is stored, who will have access, and how anonymity will be preserved.

- **Risk Assessment**: Identify potential risks to participants and outline strategies to minimize them.
- **Vulnerable Populations**: If applicable, address how you will protect vulnerable groups (e.g., children, elderly, or marginalized communities) involved in your project.
- **Research Integrity**: Highlight your commitment to honesty and transparency in research. Discuss how you will avoid plagiarism, fabrication, and falsification of data.
- **Compliance with Regulations**: Mention relevant ethical guidelines and regulations that your project will adhere to.
- **Impact on Society**: Reflect on the broader implications of your project. Discuss how the outcomes may affect communities, stakeholders, or the environment, and how you plan to address any potential negative consequences.
- **Feedback Mechanisms**: Describe how you incorporate feedback from participants and stakeholders to improve the ethical conduct of the project throughout its duration.

3.2 Example of an Equation in LaTeX

Eq. 3.1 [note that this is an example of an equation's in-text citation] is an example of an equation in LaTeX. In Eq. (3.1), s is the mean of elements $x_i \in \mathbf{x}$:

$$s = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{3.1}$$

Have you noticed that all the variables of the equation are defined using the **in-text** maths command $.$, and Eq. (3.1) is treated as a part of the sentence with proper punctuation? Always treat an equation or expression as a part of the sentence.

3.3 Example of a Figure in LaTeX

Figure 3.1 is an example of a figure in LaTeX. For more details, check the link:
wikibooks.org/wiki/LaTeX/Floats,_Figures_and_Captions.
Keep your artwork (graphics, figures, illustrations) clean and readable. At least 300dpi is a good resolution of a PNG format artwork. However, an SVG format artwork saved as a PDF will produce the best quality graphics. There are numerous tools out there that can produce vector graphics and let you save that as an SVG file and/or as a PDF file. One example of such a tool is the "Flow algorithm software". Here is the link for that: flowgorithm.org.

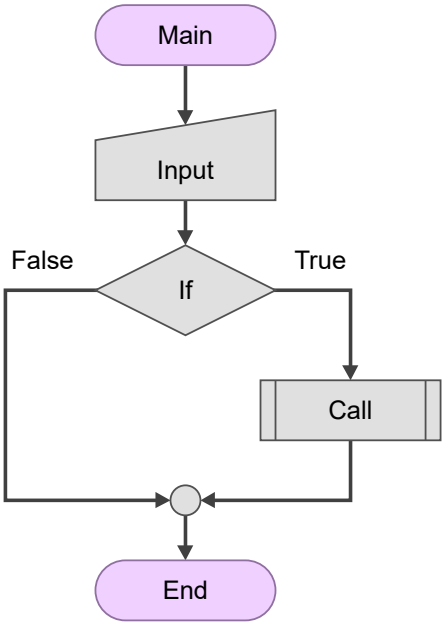

Figure 3.1: Example figure in LaTeX.

3.4 Example of an algorithm in LaTeX

Algorithm 1 is a good example of an algorithm in LaTeX.

Algorithm 1 Example caption: sum of all even numbers

Input: $\mathbf{x} = x_1, x_2, \ldots, x_N$
Output: $EvenSum$ (Sum of even numbers in $\mathbf{x}$)

```
1: function EvenSummation(x)
2:     EvenSum ← 0
3:     N ← length(x)
4:     for i ← 1 to N do
5:         if x_i mod 2 == 0 then          ▷ Check whether a number is even.
6:             EvenSum ← EvenSum + x_i
7:         end if
8:     end for
9:     return EvenSum
10: end function
```

3.5 Example of code snippet in LaTeX

Code Listing 3.1 is a good example of including a code snippet in a report. While using code snippets, take care of the following:

- do not paste your entire code (implementation) or everything you have coded. Add code snippets only.
- The algorithm shown in Algorithm 1 is usually preferred over code snippets in a technical/scientific report.
- Make sure the entire code snippet or algorithm stays on a single page and does not overflow to another page(s).

Here are three examples of code snippets for three different languages (Python, Java, and CPP) illustrated in Listings 3.1, 3.2, and 3.3 respectively.

```python
import numpy as np

x  = [0, 1, 2, 3, 4, 5] # assign values to an array
evenSum = evenSummation(x) # call a function

def evenSummation(x):
    evenSum = 0
    n = len(x)
    for i in range(n):
        if np.mod(x[i],2) == 0: # check if a number is even?
            evenSum = evenSum + x[i]
    return evenSum
```

Listing 3.1: Code snippet in LaTeX and this is a Python code example

Here we used the "\clearpage" command and forced-out the second listing example onto the next page.

```java
public class EvenSum{
    public static int evenSummation(int[] x){
        int evenSum = 0;
        int n = x.length;
        for(int i = 0; i < n; i++){
            if(x[i]%2 == 0){ // check if a number is even?
                evenSum = evenSum + x[i];
            }
        }
        return evenSum;
    }
    public static void main(String[] args){
        int[] x  = {0, 1, 2, 3, 4, 5}; // assign values to an array
        int evenSum = evenSummation(x);
        System.out.println(evenSum);
    }
}
```

Listing 3.2: Code snippet in LaTeX and this is a Java code example

```cpp
int evenSummation(int x[]){
    int evenSum = 0;
    int n = sizeof(x);
    for(int i = 0; i < n; i++){
        if(x[i]%2 == 0){ // check if a number is even?
            evenSum = evenSum + x[i];
      }
    }
    return evenSum;
}

int main(){
    int x[]  = {0, 1, 2, 3, 4, 5}; // assign values to an array
    int evenSum = evenSummation(x);
    cout<<evenSum;
    return 0;
}
```

Listing 3.3: Code snippet in LaTeX and this is a C/C++ code example

3.6 Example of in-text citation style

3.6.1 Example of the equations and illustrations placement and reference in the text

Make sure whenever you refer to the equations, tables, figures, algorithms, and listings for the first time, they also appear (placed) somewhere on the same page or in the following page(s). Always make sure to refer to the equations, tables and figures used in the report. Do not leave them without an **in-text citation**. You can refer to equations, tables and figures more them once.

3.6.2 Example of the equations and illustrations style

Write **Eq.** with an uppercase "Eq" for an equation before using an equation number with (\eqref{.}). Use "Table" to refer to a table, "Figure" to refer to a figure, "Algorithm" to refer to an algorithm and "Listing" to refer to listings (code snippets). Note that, we do not use

the articles "a," "an," and "the" before the words Eq., Figure, Table, and Listing, but you may use an article for referring the words figure, table, etc. in general.

For example, the sentence "A report structure is shown in **the** Table 3.1" should be written as "A report structure is shown **in** Table 3.1."

3.6.3 Tools for In-text Referencing

You will have noticed that there are linked references within the text to specific items in this document (e.g., equations, figures, tables, chapters, sections, etc.). This is enabled by a combination of `\label{}` and `\ref{}` commands. The former is typically "attached" to an object/section to be labeled, for instance: `\section{My Section}\label{sec:my}`. This label, `sec:my`, can then be used to create an in-text reference (with link) to the referenced object: `\ref{sec:my}`.

The in-text references to the preceding equation were written as: `Eq.~\eqref{eq:eq_example}`. Here, the author needed to explicitly write the `Eq.` text, include a tilde, `~`, to ensure that the text is not separated from the number at a line break, and used `eqref` to automate placement of parentheses around the number. Alternatively, we could use the cleverref system to reference this item with `\Cref{eq:eq_example}`, yielding: Equation (3.1). This makes the textual part (Equation) automatic along with spacing and other formatting. The capital `C` in that command specifies capitalisation of the word, whereas lowercase for a figure item would, `\cref{fig:chart_a}` would yield a lowercase abbreviated form: fig. 3.1.

3.7 Summary

Write a summary of this chapter.

Note: In the case of **software engineering** project a Chapter "**Testing and Validation**" should precede the "Results" chapter. See Section 3.1.1 for report organization of such project.

Chapter 4

Results

The results chapter tells a reader about your findings based on the methodology you have used to solve the investigated problem. For example:

- If your project aims to develop a software/web application, the results may be the developed software/system/performance of the system, etc., obtained using a relevant methodological approach in software engineering.
- If your project aims to implement an algorithm for its analysis, the results may be the performance of the algorithm obtained using a relevant experiment design.
- If your project aims to solve some problems/research questions over a collected dataset, the results may be the findings obtained using the applied tools/algorithms/etc.

Arrange your results and findings in a logical sequence.

4.1 A section

...

4.2 Example of a Table in LaTeX

Table 4.1 is an example of a table created using the package LaTeX"booktabs." do check the link: wikibooks.org/wiki/LaTeX/Tables for more details. A table should be clean and readable. Unnecessary horizontal lines and vertical lines in tables make them unreadable and messy. The example in Table 4.1 uses a minimum number of liens (only necessary ones). Make sure that the top rule and bottom rule (top and bottom horizontal lines) of a table are present.

Table 4.1: Example of a table in LaTeX

Bike		
Type	Color	Price (£)
Electric	black	700
Hybrid	blue	500
Road	blue	300
Mountain	red	300
Folding	black	500

4.3 Example of captions style

- The **caption of a Figure (artwork) goes below** the artwork (Figure/Graphics/illustration). See example artwork in Figure 3.1.
- The **caption of a Table goes above** the table. See the example in Table 4.1.
- The **caption of an Algorithm goes above** the algorithm. See the example in Algorithm 1.
- The **caption of a Listing goes below** the Listing (Code snippet). See example listing in Listing 3.1.

4.4 Summary

Write a summary of this chapter.

Chapter 5

Discussion and Analysis

Depending on the type of project you are doing, this chapter can be merged with "Results" Chapter as " Results and Discussion" as suggested by your supervisor.

In the case of software development and the standalone applications, describe the significance of the obtained results/performance of the system.

5.1 A section

Discussion and analysis chapter evaluates and analyses the results. It interprets the obtained results.

5.2 Significance of the findings

In this chapter, you should also try to discuss the significance of the results and key findings, in order to enhance the reader's understanding of the investigated problem

5.3 Limitations

Discuss the key limitations and potential implications or improvements of the findings.

5.4 Summary

Write a summary of this chapter.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Typically a conclusions chapter first summarizes the investigated problem and its aims and objectives. It summaries the critical/significant/major findings/results about the aims and objectives that have been obtained by applying the key methods/implementations/experiment set-ups. A conclusions chapter draws a picture/outline of your project's central and the most signification contributions and achievements.

A good conclusions summary could be approximately 300–500 words long, but this is just a recommendation.

A conclusions chapter followed by an abstract is the last things you write in your project report.

6.2 Future work

This section should refer to Chapter 4 where the author has reflected their criticality about their own solution. The future work is then sensibly proposed in this section.

Guidance on writing future work: While working on a project, you gain experience and learn the potential of your project and its future works. Discuss the future work of the project in technical terms. This has to be based on what has not been yet achieved in comparison to what you had initially planned and what you have learned from the project. Describe to a reader what future work(s) can be started from the things you have completed. This includes identifying what has not been achieved and what could be achieved.

A good future work summary could be approximately 300–500 words long, but this is just a recommendation.

Chapter 7

Reflection

Write a short paragraph on the substantial learning experience. This can include your decision-making approach in problem-solving.

Some hints: You obviously learned how to use different programming languages, write reports in LaTeXand use other technical tools. In this section, we are more interested in what you thought about the experience. Take some time to think and reflect on your individual project as an experience, rather than just a list of technical skills and knowledge. You may describe things you have learned from the research approach and strategy, the process of identifying and solving a problem, the process research inquiry, and the understanding of the impact of the project on your learning experience and future work.

Also think in terms of:

- what knowledge and skills you have developed
- what challenges you faced, but was not able to overcome
- what you could do this project differently if the same or similar problem would come
- rationalize the divisions from your initial planed aims and objectives.

A good reflective summary could be approximately 300–500 words long, but this is just a recommendation.

Note: The next chapter is "**References**," which will be automatically generated if you are using BibTeX referencing method. This template uses BibTeX referencing. Also, note that there is difference between "References" and "Bibliography." The list of "References" strictly only contain the list of articles, paper, and content you have cited (i.e., refereed) in the report. Whereas Bibliography is a list that contains the list of articles, paper, and content you have cited in the report plus the list of articles, paper, and content you have read in order to gain knowledge from. We recommend to use only the list of "References."

Appendix A

An Appendix Chapter (Optional)

Some lengthy tables, codes, raw data, length proofs, etc. which are **very important but not essential part** of the project report goes into an Appendix. An appendix is something a reader would consult if he/she needs extra information and a more comprehensive understating of the report. Also, note that you should use one appendix for one idea.

An appendix is optional. If you feel you do not need to include an appendix in your report, avoid including it. Sometime including irrelevant and unnecessary materials in the Appendices may unreasonably increase the total number of pages in your report and distract the reader.

Appendix B

An Appendix Chapter (Optional)

...