Final Year Project Report Chapter Guide

This note covers in detail the *normal report structure*. You must also comply with previously taught guidelines on clear labelling of figures, graphs, use of tables instead of lists in text, plagiarism, etc.

The advice given here will apply to many projects. However each project is unique, and you should without fail get advice from your supervisor about how to structure your report, and its content, well before writing starts.

Where the advice given here and that of your supervisor differ, you should follow your supervisor's advice as they will be taking into account the individual nature of your project.

Your report should, without fail, contain the sections specified in the Project Guide. In addition, the structure and content of the internal chapters is *typically* as below.

Some variation in structure is possible for the internal Chapters, however omitting relevant content is not. For example, the project Evaluation Chapter could be a subsection of the Conclusions, or the Results. An analytical project would not have a User Guide unless the software used to run simulations is intended to be of use to others. That is often but not always the case.

Where linked material is distributed throughout the report it is very important to make explicit forward or backward references to the relevant sections. This enables a reader to understand your argument without the burden of repeated material.

Front Matter

The first page of your report should be a standard format Cover Page. The second page should be the mandatory Plagiarism Statement. Signature is not required – its presence will imply signature.

Between the Abstract and the Introduction there will normally be a Contents page (or more than one if required).

Acknowledgments

This section is optional, but customary.

Abstract

The abstract should (very concisely) summarise the topic, content and conclusions of the project. Abstracts can vary in length from 50 words up to at most 200 words. They are more concise than executive summaries. They do not need to be identical to any previously submitted abstract. Note that abstracts have no explicit marking in the final report scheme, so keep them short and clear.

Introduction

The introduction should set the scene and give a high-level problem statement/specification, so that after reading the introduction the reader understands roughly what the problem is and what you intend to do about it. Is the idea to write software, or develop an algorithm, or produce hardware, or something else?

You should then highlight and summarise the most interesting or important questions or problems that your project addresses, and the broader context in which those questions or problems are situated.

Finally, you must briefly introduce the structure of report (what you will cover in which chapters and how these relate to each other). You don't need to go into any detail, the aim is to make sure the reader has an idea about what will be discussed and in what order.

Background

You should provide enough background to the reader for them to understand what the project is all about, and what is the relevant prior work.

Examiners like to know that you have done the appropriate background research and it is important that you review either what has been done previously to tackle related problems, or perhaps what other products exist related to your deliverable. Clear references are important here, and much of this section will typically already have been written in your Interim Report. You may use feedback from that report to improve what you write in your Final Report, and should note that self-plagiarism between the two reports is not possible, so no citation is needed of your own earlier writing.

- What does the reader need to know in order to understand the rest of the report? What problem are you solving?
- Why is this problem interesting or worthwhile to solve?
- Who cares if you solve it?
- How does this relate to other work in this area?
- What work does it build on?
- For 'research-style' projects involving the design and analysis of specific algorithms there is a large amount of relevant background both of general theory, and very specific to the algorithm you investigate. Supervisors will help you to see what is most important here, but the general rule is that you must both provide overall context and note work close to what you do that influences your work or is in some way comparable to your work.

Requirements Capture

Projects with a deliverable that serves a specific function often have an initial phase in which expected use is investigated and a brief more detailed than the specification is constructed. This would include what is necessary, what is desirable, etc in the final deliverable. The results of requirements capture determine project objectives and are used to inform project evaluation.

Requirements capture is important in all projects with real-world deliverables, and is often a significant amount of work in software projects. Where requirements capture is less relevant (for example in an analytical 'research-style' project) this may be replaced by a detailed description of the project aims and objectives in the Introduction or the Background sections.

Analysis and Design

If your project involves designing a system, give a good high-level overview of your design.

In many projects, the final design is different from that originally envisaged. If the differences are interesting, write about them, and why the changes were made. Discoveries during the project may have changed the direction of work, or invalidated prior work, in which case you get credit for the design process, if it is principled, as well as the end product.

If your design was not implemented fully, describe which parts you did implement, and which you didn't. If the reason you didn't implement everything is interesting (eg it turned out to be difficult for unexpected reasons), write about it.

Note that the Project Report is written at the end of project work and must describe the project work, but need not do this chronologically. Often the best description of design, in retrospect, is far from the way in which you developed it. Where the evolution of ideas is interesting or relevant it can be described, as above, but otherwise the order in which things were done need not be documented.

The Examiners are just as interested in the engineering process you went through in performing your project work as the results you finally produced. So make sure your report identifies when design choices have to be made, what were the possibilities, and why you made the particular choices and decisions that you did. They are looking for principled rational arguments and for critical assessment.

Engineering involves trade-offs and the reasons for a design decision may be various, and may in some cases be out of your control. Explicit understanding of this, and the ability to communicate it, is important.

Implementation

In projects with a software element give code details (not a complete listing, but descriptions of key parts). Discuss the most important or interesting aspects, and anything that was surprising or difficult. It probably won't be possible or relevant to discuss everything – state clearly what you leave out.

Pasting verbatim code from your project into the report is rarely a good idea; usually it should be edited down to remove extraneous details and often annotated to help the reader understand what they are looking at. Good reasons for including code could be to illustrate algorithmic flow, or highlight an interesting optimisation, demonstrate interactions with a data-structure, or give an example of input for a tool that has been designed. If you cannot explain what message or point a code fragment is conveying, it probably should not be in the report, and if you cannot justify why a line of code in the fragment helps convey that point, it maybe it shouldn't be in the code fragment.

For similar reasons, pasting screen-shots is unlikely to be a good use of space, unless it serves a particular purpose. Screen-shots are sometimes used instead of drawing a picture (for example as a "cheap" way of showing wave-forms or state machines), or in order to capture the results of running a tool (which will often be text anyway), and tend to look unprofessional and lazy. They are also sometimes used as page filler, or as "proof" that a tool was launched and something compiled, which is not necessary: your marker's default position will be to believe your claims. Use a screen-shot if it is demonstrating a particular point, such as a failure mode in a tool, or a particular interaction that is difficult/impossible to highlight, but make sure you edit and annotate the figure to show and highlight the important parts.

When discussing the implementation, it is generally best to focus on the conceptual and logical design, and only then dive into details for interesting parts or to highlight important decisions. Well thought-out figures and diagrams are often much more effective at conveying your design than source code, whether that is a data-structure, a client-server interaction, a design-flow, or a circuit hierarchy. For example, a diagram of a block-diagram of digital circuit can be used to communicate to the reader most of the important high-level decisions you have made, then VHDL fragments (or another diagram) could be used to focus on specific parts and demonstrate any interesting local details.

Complete listings may be included as appendices of your report but this is not normally appropriate, unless the appendix is documentation, or describes an API. Software may be provided on a CD-ROM given to your supervisor or in a cloud repositories such as Github. No marks are awarded for the mass or page-count of a report, and reports which contain 50 pages of printed code are more likely to be seen as showing poor judgement.

Testing

Describe your Test Plan -- how the program or system was verified. Put the actual test results in an Appendix if they are repetitive but relevant. Detailed test data may be omitted from the report if not relevant, however an accurate summary of tests should be included in the Report itself. Sometimes non-working designs are described in project reports as though they work, when in reality they don't, or only partially work. Therefore a precise description of what works and *how this has been established* is important. Examiners may try to compile, use, or test deliverables themselves (even after your report is submitted), and your report should accurately reflect the state of the project.

This section is normally useful for software or hardware deliverables and less relevant in analytical projects.

Results

This covers an area different from the 'Testing' chapter, and relates to the understanding and analysis of the algorithm, program, or hardware behaviour. Where the deliverable is a product with easily quantified performance then the previous Chapter shows how functional correctness is established, while this Chapter shows qualitatively or

quantitatively how well the product works. The list below shows typical content, not all of this will be applicable to all projects.

- An empirical relationship between parameters and results may be investigated, typically through the use of appropriate graphs.
- Where theory predicts aspects of performance the results can be compared with expectation and differences noted and (if possible) explained.
- Semi-empirical models of behaviour may be developed, justified, and tested.
- The types of experiments/simulations that were carried out should be described. Why were certain experiments carried out but not others? What were the important parameters in the simulation and how did they affect the results? If there are very many graphs and tables associated with this chapter they may be put in the Appendix, but it is generally better to keep these close to the text they illustrate, as this is easier for the reader.

Evaluation

This Chapter (or possibly section of the conclusions) is distinct from your results. It must contain your critical evaluation of your work as compared to previous analysis, algorithms, products, and when related to your original objectives. To what extent have your original objectives been fulfilled? If they have changed, what is your rationale for this? What are the advantages, disadvantages of your approach compared with related work? How does the scope of your work differ from related work? Examiners expect your project report to show evidence of your ability to think as an engineer, and that includes the ability to critically reflect on your own work and evaluate its significance.

Material here will compare project outcomes with initial objectives and requirements captured. Usually your Interim Report will contain these. Where these have changed significantly over the course of the project this should be explained and reasons given. This section should not require examiners to read your Interim Report, and will not reference it. Changes between final and initial objectives should be explained in a self-contained manner.

Note that here you will reference and summarise, rather than repeat, your description of Requirements Capture earlier in the Final Report.

Conclusions and Further Work

How successful have you been? What have you achieved? How could the work be taken further given more time (perhaps by another student next year)? It is important here to identify positively what is worthwhile in your work. At the same time, honesty, and a clear description of the limits of your work, is equally important. It is often most appropriate to describe work you did not have time to complete as further work.

Your readers will not be clear where, in your long report, are your most significant achievements. In the conclusions you must summarise this, referring as necessary to other sections for more detail.

- What design choices did you have along the way, and why did you make the choices you made?
- What was the most difficult and/or clever part of the project?
- Why was it difficult?
- How did you overcome the difficulties?
- Did you discover or invent anything novel?
- What did you learn?

Note that “difficult” does not necessarily mean the thing that took you the longest amount of time. Note also that the conclusions must concisely summarise this material, and refer to other sections for the details.

User Guide

When the deliverable is software or hardware intended to be useful to others, this Chapter provides details on how to install/configure/use it. Otherwise it can be omitted.

Bibliography

Give publication details for all the citations you have made in the report. Internet URLs are allowed and in many cases necessary, however published material should if possible be given in standard form with publishing body, date, and author(s). Open URLs to published material, where these exist, may also be given for a reader's convenience.

Bibliographic format: the recommended format for simplicity (using latex and bibtex) is numeric, with hyperref allowing hyperlink references (which you will almost certainly need).

For example:

```
\usepackage[backend=biber,style=numeric,
sortcites,sorting=nty,backref,natbib,
hyperref]{biblatex}
```

Exact blibliographic format is not mandated but uniformity across the cohort is helpful: let me know if you think the above does not work for you.

Appendices

Please use appendices as necessary for material that is tangentially relevant, or necessary to preserve project value, but that you do not expect Examiners to read. Note that software projects with significant code should normally provide electronic versions of the code on USB stick, CDROM, or cloud repository. In that case the Appendix should state where the code may be found. **The Appendix should normally include, or refer to, all the technical details needed for another user to continue code development.**

Typical items that should not be in the main report, but should (possibly - since Appendices are not normally read this is a matter of judgement) be in appendices:

- Source code. Note that code (if long) should also be available in some electronic form, e.g. a github repository, CD, or USB stick. Software projects must provide access to the source code in some form. The Appendix will then reference this.
- Test data sets (again, for large volumes of data an electronic form would be more appropriate). The report itself will contain concise summaries of the test data in a human readable form.
- Raw results. Tables of results in unreadable form should not be put in the main report, but if needed may be put in an Appendix.
- Related material possibly but not directly relevant to the project work. E.g. manuals of test equipment used. (There is no requirement to include such, but in some cases, where they have some tangential relevance, it might be appropriate).