Guidelines for Capstone Project Report Using LaTeX 2_ε

Joe Bloggs
Department of Computer Science
joe@bloggs.com

Jane Doe
Department of Information Systems
jane.doe@uct.ac.za

John Doe
Department of Computer Science
john.doe@uct.ac.za

This document outlines the requirements for the final report for the Computer Science Capstone Project Report and describes the use of the LaTeX 2_ε for that purpose. Please follow this precise template for your final report.

1. Introduction

The capstone project is done as the culmination of your three year study of Computer Science. It is the development of a real application that draws on all your knowledge of the field gained in the course of your training in the subject.

The project should be written up as a professional software engineering design and development project. We expect a report of about 3500-4000 words, written single spaced, with a font size of at least 11 pts. Use at least a 2.5 cm margin on all sides of the pages. Please use "styles" for formatting if you are using a word processing package or use LaTeX. Depending on how many diagrams you use (more is better) the report will be between 7 and 10 pages long. Note that while word users will struggle with numbered headings and lists, LaTeX has its own ideas about where "floats" (like tables and figures) will go, as usual search the internet for advice (e.g., http://www.andy-roberts.net/writing/latex/floats_figures_captions). Don't worry, word's specialty is loosing your figures in some between-page limbo; LaTeX will not loose them, just place them way after the spot where you want them. This document shows the format we expect and you can use it as a template. Your appendices (e.g., user manual, test results, which are needed) are not included in these limits.

You must had-in an Adobe Acrobat file for your report (i.e., pdf file). Not word, latex source, but PDF!

2. Approach

You should begin your write-up with an overview and then drill down into the details of what you produced. Your report should cover the following sections (Sections **??** – **??**).

Abstract

First you should have an executive summary (or abstract) just a single paragraph saying what the results of the project are (at most 200 words).

2.1. Introduction

Your introduction provides the context for the project and should contain the statement of the scope of the project (which may have changed since you first wrote it). Someone reading your introduction must have clear idea of what the system is intended for. If you think there is something special about the kind of problem you tackled that your reader needs to know up front then this is where you say it.

If you need any survey of other work (you probably don't) then put it towards the end of the introduction and give suitable references. A case where this is needed is if your project builds on someone else's project or some published algorithm.

Discuss your approach to solving the problem. Please give a short overview of the software engineering methods you used (e.g., traditional analysis followed by design and implementation – typically the case if you did an evolutionary prototype, or a more agile approach where you had a cyclical development process).

2.2. Requirements Captured

The next section deals with the analysis of your system. Cover the functional, non-functional and usability requirements. This is where you present your use case narratives and diagrams.

Discuss the major analysis artefacts that you produced. We will expect you to produce at least one overall description of the architecture used in your system as a diagram, either here or below (see Section **??**). You may also want to include an analysis class hierarchy diagram.

2.3. Design Overview

The next section is an overview of your design. The system design has to be justified in terms of the expected behaviour of the final product.

If you produced a design class diagram put it here.

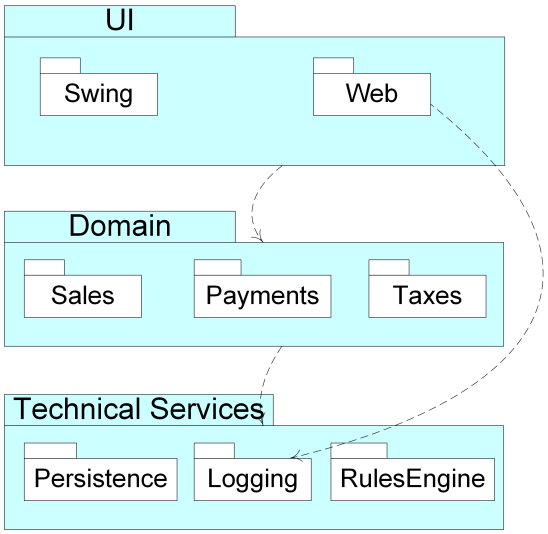

Figure 1: An architecture diagram. Caption to go below figure

You must present the overall architecture of the system together with an architecture diagram. You may choose what kind of diagram best suits your project but we would expect a layered architecture diagram (see Figure **??**) unless there is a good reason for some other kind of diagram. It need not be a formal UML diagram as long as it conveys all the necessary information clearly.

You should then (in subsections) cover the algorithms and the data organisation used and why they were considered the best.

2.4. Implementation

Now we get to the details.

- Describe your data structures and be sure to illustrate them with a diagram.
- If your user interface was a key feature describe how that was implemented.
- Discuss the function of the most significant methods in each class. This may well require flowcharts, or sequence diagrams, in some cases.
- Any special relationship between the classes (e.g. friends) and why they exist.
- A description of any special programming techniques or libraries used.

2.5. Program Validation and Verification

Tell us how you tested the system and why you believe it works. Describe all the steps taken to validate the correctness of the program.

If you had user tests then say what you did and what the results were. Describe why these test data were chosen (what test conditions the data was testing). Table **??** provides an example of the sorts of results we are looking for. The full detail of the test runs should be appended to the report.

Table 1: A table of tests. A table caption goes above the table.

Data Set and reason for its choice	**Test Cases**		
	Normal Functioning	*Extreme boundary cases*	*Invalid Data (program should not crash)*
Preliminary test (see Appendix 3)	Passed	n/a	Fell over

Follow your table of results with a discussions of them highlighting how useful and usable your system is for its intended purpose.

2.6. Conclusion

Your report must have a clear conclusion where you revisit the aims set out in the beginning and discuss how well you met them. Did you achieve the objective of creating a well-structured, modular, and robust system? Please summarize the design features and test results that show this.

2.7. User Manual

Your system must have a user manual. Append this to your report (make it Appendix A) or bind it separately if it is big. If your system is interactive and has a good user interface with context dependent help then this can be just a cheat sheet. Discuss the level at which your user manual is to be pitched with your client. If your system is to be extended then you might want to include a technical API manual.

3. Conclusion

This document has covered the major sections needed for your report. You will probably have each of the subsections 2.1–2.7 as major section in the report each with its own subsections.

A marking guide for the report will be provided later.

A. Code Legibility and Output

This is not strictly part of the report but is a requirement for the final hand-in.

- Each method should start wide a brief description of its function.
- Use indentation to display the structure within a method.
- Comments should be used extensively. They are best used to describe logical blocks of code rather than individual statements. Line-by-line comments have the drawbacks of not providing any overview and of decreasing readability.
- Meaningful identifiers should be chosen.
- Output should be pleasingly formatted and easy to read.

You do, of course, have the option to call in any of your favourite packages for setting maths, graphics, computer listings, etc.

References

[Kopka and Daly(2004)] Kopka, H. and Daly, P.W. (2004) *A Guide to LaTeX 2ε: Document Preparation for Beginners and Advanced Users* (4th edn). Addison-Wesley.

[Lamport(1994)] Lamport L. (1994) *LaTeX: A Document Preparation System* (2nd edn). Addison-Wesley.

[Mittelbach and Goossens(2004)] Mittelbach, F. and Goossens, M., (2004) *The LaTeX Companion* (2nd edn). Addison-Wesley.