Guidelines for Capstone Project Report

Edwin Blake
Department of Computer Science
edwin@cs.uct.ac.za

This document outlines the requirements for the final report for the Computer Science Capstone Project Report. It is also the precise template for your final report. Please follow this format exactly.

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 during 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. We mean you to use styles for everything, not just headings and lists but also for a different font. So use 'Emphasis' style for *italics* and do *not* use a **bold face**. No blank lines between paragraphs: use a style with built-in spacing between paragraphs (except to get figures and their captions to position properly).

Depending on how many diagrams you use (more is better) the report will be 7 to 10 pages long. This document shows the format we expect and you must please use it as a template (or the LaTeX version). 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*!

1.1 Special Issues with Microsoft Word

You have a choice in formatting your final report. You can use either MS Word, Libre (or Open) Office or LaTeX. *Your final handin in each case will be a pdf document.* All these tools have their strengths and their weaknesses: your first source of help is of course the internet.

If you intend using MS Word or LibreOffice you will have to master the use of styles. These are essential for creating longer documents. LibreOffice Writer also has similar styles to word but is more explicit about the difference between character styles, paragraph styles and list styles. It also has page styles that word lacks.

For general help have a look at the Word MVP site which has lots of useful advice: word.mvps.org/index.html and in particular their FAQ word.mvps.org/FAQs/index.htm. For example: if my reference to styles has you confused look at shaunakelly.com/word/styles/tipsonstyles.html.

Once you start using styles the biggest problem is how to get numbered headings and lists to behave. At that point you can either give up and use LaTeX or LibreOffice or read "How to create numbered headings or outline numbering" by Shauna Kelly (www.shaunakelly.com/word/numbering/outlinenumbering.html) with the memorable opening lines:

> Managing numbered headings and outline numbering in anything but the simplest of Microsoft Word documents can easily drive you crazy. You seem to go round and round in circles, and never end up with what you want. And just when you get close, it falls to pieces.
>
> You are not alone!

The summary of the advice is:

1. Don't use the pretty numbering or bullet buttons on the toolbar or the ribbon. They look inviting, but they're not what you need.
2. Don't use the toolbar numbering or bullet buttons. They're not what you need either.
3. Apply styles to your headings, preferably Word's built-in Heading styles.
4. Modify the styles so you can have the font, paragraph and other formatting to suit your needs.
5. Modify the numbering and indenting by modifying the numbering settings of the Heading styles. The styles will manage the numbering and the indents.

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:

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. 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 2.3). 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.

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 1) 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.

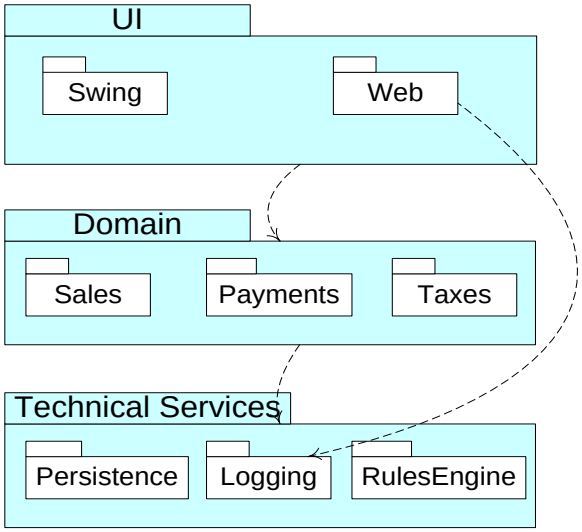

Figure 1: An architecture diagram. Caption to go below figure.
Note that LibreOffice handles this better than MS word

- 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 the Quality Management Plan for your project, that is, software testing plan. The plan should indicate the types of testing that was performed and detail how they were done. This must include the reasons on why the chosen testing protocol was considered effective.

Create a table that summarizes the testing plan (see Table 1).

Table 1: *Summary Testing Plan. A table caption goes above the table.*

Process	Technique
1. Class Testing: test methods and state behaviour of classes	Random, Partition and White-Box Tests
2. Integration Testing: test the interaction of sets of classes	Random and Behavioural Testing
3. Validation Testing: test whether customer requirements are satisfied	Use-case based black box and Acceptance tests
4. System Testing: test the behaviour of the system as part of a larger environment	Recovery, security, stress and performance tests

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 2 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 2: *Summary of tests carried out. 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.

Appendix 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.

References

If you need to give references ensure that the following information is included:

Journal article: Author’s surname, Author’s initial. (Year of publication) Title of paper. Title of journal, volume number, page numbers.

Book: Author’s surname, author’s initial. (Year of publication) Title of book. Publisher, publisher location.

Chapter: Author’s surname, author’s initial. (Year of publication) Title of chapter. In editors (eds), Title of book. Publisher, publisher location.

Conference: Author’s surname, author’s initial. (Year of publication) Title of paper. Title of Conference, conference location, conference date, page numbers. Publisher, publisher location.

Technical documents: Document number (Year of publication) Document title. Publisher, publisher location.

Internet source: Author’s surname, author’s initial. (Year of publication) Title of source. url (retrieved date).