The project report is an extremely important aspect of the project. It serves to show what you have achieved and should demonstrate that:

* You understand the wider context of computing by relating your choice of project, and the approach you take, to existing products or research.
* You can apply the theoretical and practical techniques taught in the course to the problem you are addressing and that you understand their relevance to the wider world of computing.
* You are capable of objectively criticising your own work and making constructive suggestions for improvements or further work based on your experiences so far.
* As a computing professional, you can explain your thinking and working processes clearly and concisely to third parties who may not be experts in the field in which you are working.

Most of the project assessors will not have followed the project throughout and will only have a short time to listen to a presentation or see a demonstration. For this reason they will rely heavily on the report to judge the project. Also, if in the end your overall degree marks put you on a boundary between two degree classifications, the final outcome can be influenced significantly by the quality of your project. You should appreciate that the external examiners, who play a crucial role in the final recommendation, have only the report by which to judge your project performance.

Many students underestimate the importance of the report and make the mistake of thinking that top marks can be achieved simply for producing a good product. This is fundamentally not the case and many projects have been graded well below their potential because of an indifferent or poor write-up. In order to get the balance right you should consider that the aim of the project is to produce a good report and that software, hardware, theory etc. that you developed during the project are merely a means to this end. Don't make the mistake of leaving the write-up to the last minute. Ideally you should produce the bulk of the report as you go along and use the last week or two to bring it together into a coherent document.

The physical layout and formatting of the report is also important, and yet is very often neglected. A tidy, well laid out and consistently formatted document makes for easier reading and is suggestive of a careful and professional attitude towards its preparation. Many supervisors will advise you to use LaTeX as this solves most of the formatting problems for you. This is not a requirement, but be aware that if you use a system like MS Word you may have to invest more time to get the layout right. If your report is to contain a substantial number of mathematical formulae you are strongly advised to use LaTeX.

Remember that quantity does not automatically guarantee quality. A 150 page report is not twice as good as a 75-page one, nor a 10,000 line implementation twice as good as a 5,000 line one. Conciseness, clarity and elegance are invaluable qualities in report writing, just as they are in programming, and will be rewarded appropriately. Try to ensure that your report contains the following elements (the exact structure, chapter titles etc. is up to you):

**Title page**

This should include the project title and the name of the author of the report. You can also list the name of your supervisor if you wish.

**Abstract**

The abstract is a very brief summary of the report's contents. It should be about half a page long. Somebody unfamiliar with your project should have a good idea of what it's about having read the abstract alone and will know whether it will be of interest to them. Note that the abstract is a summary of the entire project including its conclusions. A common mistake is to provide only introductory elements in the abstract without saying what has been achieved.

**Acknowledgements**

It is usual to thank those individuals who have provided particularly useful assistance, technical or otherwise, during your project. Your supervisor will obviously be pleased to be acknowledged as he or she will have invested quite a lot of time overseeing your progress.

**Contents page**

This should list the main chapters and (sub)sections of your report. Choose self-explanatory chapter and section titles and use double spacing for clarity. If possible you should include page numbers indicating where each chapter/section begins. Try to avoid too many levels of subheading - three is sufficient.

**Introduction**

This is one of the most important components of the report. It should begin with a clear statement of what the project is about so that the nature and scope of the project can be understood by a lay reader. It should summarise everything you set out to achieve, provide a clear summary of the project's background, relevance and main contributions. It should explain the motivation for the project (i.e., why the problem is important) and identify the issues to be addressed (i.e., why the problem is difficult). The introduction should set the scene for the project and should provide the reader with a summary of the key things to look out for in the remainder of the report. When detailing the contributions it is helpful to provide pointers to the section(s) of the report that provide the relevant technical details. The introduction itself should be largely non-technical. It is sometimes useful to state the main objectives of the project as part of the introduction. However, avoid the temptation to list low-level objectives one after another in the introduction and then later, in the evaluation section (see below), say something like "All the objectives of the project have been met blah blah...". A project that meets all its objectives is, by definition, weak and unambitious. Concentrate instead on the big issues, e.g. the main questions (scientific or otherwise) that the project sets out to answer.

**Background**

The background section of the report should set the project into context by relating it to existing published work which you read at the start of the project when your approach and methods were being considered. There are usually many ways of solving a given problem, and you shouldn't just pick one at random. Describe and evaluate as many alternative approaches as possible. The published work may be in the form of research papers, articles, text books, technical manuals, or even existing software or hardware of which you have had hands-on experience. Your must acknowledge the sources of your inspiration. You are expected to have seen and thought about other people's ideas; your contribution will be putting them into practice in some other context. However, avoid plagiarism: if you take another person's work as your own and do not cite your sources of information/inspiration you are being dishonest; in other words you are cheating. When referring to other pieces of work, cite the sources where they are referred to or used, rather than just listing them at the end. Make sure you read and digest the [Department's plagiarism document](https://wiki.imperial.ac.uk/display/docteaching/Plagiarism).

In writing the Background chapter you must demonstrate your capability of *analysis*, *synthesis* and *critical judgement*. Analysis is shown by explaining how the proposed solution operates in your own words as well as its benefits and consequences. Synthesis is shown through the organisation of your Related Work section and through identifying and generalising common aspects across different solutions. Critical judgement is shown by discussing the limitations of the solutions proposed both in terms of their disadvantages and limits of applicability.

Typically you can look for Background work using different search engines including:

* [Google Scholar](http://scholar.google.com/)
* [IEEExplore](http://ieeexplore.ieee.org/)
* [ACM Digital Library](http://portal.acm.org/dl.cfm)
* [Citeseer](http://citeseer.ist.psu.edu/)
* [Science Direct](http://www.sciencedirect.com/)

**Note 1:** Often the terms *Background*, *Related Work* or *State of the Art* are used interchangeably.  
**Note 2:** Keyword search is wonderful, but you need the right *Keywords*.  
**Note 2:** IEEExplore, ACM Digital Library and Science Direct require you to be on the College network to download the PDF of papers. If at home, use VPN.

**Body of report**

The central part of the report usually consists of three of four chapters detailing the technical work undertaken during the project. The structure of these chapters is highly project dependent. They can reflect the chronological development of the project, e.g. design, implementation, experimentation, optimisation, evaluation etc. although this is not *always* the best approach. However you choose to structure this part of the report, you should make it clear how you arrived at your chosen approach in preference to the other alternatives documented in the background. If you have built a new piece of software you should describe and justify the design of your program at some high level, possibly using an approved graphical formalism such as UML. It should also document any interesting problems with, or features of, your implementation. Integration and testing are also important to discuss in some cases. You need to discuss the content of these sections thoroughly with your supervisor.

**Evaluation**

Be warned that many projects fall down through poor evaluation. Simply building a system and documenting its design and functionality is not enough to gain top marks. It is extremely important that you evaluate what you have done both in absolute terms and in comparison with existing techniques, software, hardware etc. This might involve quantitative evaluation, for example based on numerical results, performance etc. or something more qualitative such as expressibility, functionality, ease-of-use etc. At some point you should also evaluate the strengths and weaknesses of what you have done. Avoid statements like "The project has been a complete success and we have solved all the problems asssociated with blah...; - you will be shot down immediately! It is important to understand that there is no such thing as a perfect project. Even the very best pieces of work have their limitations and you are expected to provide a proper critical appraisal of what you have done.

**Conclusions and Future Work**

The project's conclusions should list the things which have been learnt as a result of the work you have done. For example, "The use of overloading in C++ provides a very elegant mechanism for transparent parallelisation of sequential programs", or "The overheads of linear-time n-body algorithms makes them computationally less efficient than O(n log n) algorithms for systems with less than 100000 particles". Avoid tedious personal reflections like "I learned a lot about C++ programming...", or "Simulating colliding galaxies can be real fun...". It is common to finish the report by listing ways in which the project can be taken further. This might, for example, be a plan for doing the project better if you had a chance to do it again, turning the project deliverables into a more polished end product, or extending the project into a programme for an MPhil or PhD.

**Bibliography**

This consists of a list of all the books, articles, manuals etc. used in the project and referred to in the report. You should provide enough information to allow the reader to find the source. In particular references must contain all the information regarding the publication of the paper and must be consistently formatted. Usually this means:

* For journals: Authors, Title, Journal, volume number, issue number, page number, publisher, month, year.
* For conferences: Authors, Title, Conference name, Place where held, publisher, page number, month, year.
* For technical reports: Authors, Title, institution, Technical report number, month, year.
* For web references: Authors, Title, Web-reference, date accessed.
* URLs are optional for published work but preferred.

A weakness of many reports is inadequate citation of a source of information. It's easy to get this right so there are no excuses. Each entry in the bibliography should list the author(s) and title of the piece of work and should give *full* details of where it can be found. For example:

[1](https://wiki.imperial.ac.uk/pages/createpage.action?spaceKey=docteaching&title=1&linkCreation=true&fromPageId=33653459) Bennett, A.J., Field, A.J. and Harrison, P.G., "Modelling and Validation of Shared Memory Coherency Protocols", Performance Evaluation, 1996, Vol. 27 & 28, 1996, pp. 541-562.

rather than just listing the source as "Performance Evaluation 1996".

Using a reference management programme can make your life simpler. See for example [Bibdesk](http://bibdesk.sourceforge.net/), [JabRef](http://jabref.sourceforge.net/), etc.

**Appendix**

The appendices contain information which is peripheral to the main body of the report. Information typically included are things like parts of the code, tables, proofs, test cases or any other material which would break up the theme of the text if it appeared in situ. You should try to bind all your material in a single volume if possible.

**User Guide**

For projects which result in a new piece of software you should provide a proper user guide providing easily understood instructions on how to use it. A particularly useful approach is to treat the user guide as a walk-through of a typical session, or set of sessions, which collectively display all the features of your system. Technical details of how the package works should be in the body of the report. Keep it concise and simple. The extensive use of diagrams illustrating the package in action prove particularly helpful. The user guide is sometimes included as a chapter in the main body of the report, but is often better in an appendix.

**Program Listings**

Complete program listings should NOT be part of the report except in specific cases at the request of your supervisor. The project report(s) must be bound in a departmental folder and must include a standard title page produced by the project co-ordinator. More of this nearer the date.

You are strongly advised to spend some time looking at the reports of previous project students to get a feel for what's good and bad. In June 1999 we introduced a "Distinguished Project" classification, which is a formal recognition of outstanding projects for which no official prize was awarded. The complete list of prize winners and the other distinguished projects, along with links to the final reports, can be found [here](http://www3.imperial.ac.uk/computing/teaching/ug/ug-distinguished-projects).