COMP0029 and COMP0138
Undergraduate Final Year
Individual Project Information, 2023-24

1 Introduction

This document describes in detail how final year projects are structured and covers many of the things you need to know about or to do. Please read all of this document carefully! Links to all documents and forms mentioned in this document are either circulated per email (year prior to the project) or available directly from the Moodle page (in the project year).

1.1 Timeline

The key activities are summarised here and described in detail in the rest of the document.

In the academic year prior to your project:

1. **Beginning of T2:** The projects are available in research areas of the department's Research Groups. Make also use of the list of supervisors and their research interests (circulated). Spend time looking at the web pages for those groups and browse research papers published by members of those groups. Reach out to academics with aligned research interests directly to arrange a project, this is, by far, the best way to end up with a project you want.
2. **Beginning of T2:** Consider whether you would prefer an **external project with industry**, the public sector or other agency. This can be arranged through Industry Exchange Network (UCL IXN) that runs in the department (contact d.mohamedally@ucl.ac.uk). IXN network will contact you directly if you express interest in IXN projects on your preferences form (see point 3). All external projects must have a supervisor from the department who will be allocated by the IXN (based on topic expertise), but once allocated (see point 4), you must contact them and confirm their role as an academic supervisor for an IXN project.
3. **End of Term 2:** Complete the online questionnaire (circulated per email) to give us your preference of research topics using ACM classification. This information will be used to match you to a supervisor if you have not found one via the direct approach (preferable), or to an academic supervisor for an IXN project. If you want an IXN project, please be sure to select the IXN project option on the form.
4. **June:** The name of your allocated academic supervisor will be circulated. Some areas are more popular than others and not everyone will obtain their first or even second choice, so be proactive and arrange the project yourself (see point 1).
5. **June:** Make contact with your allocated supervisor before the summer break and ask their guidance on any background reading, tools and technology you should research in preparation for the project.

In your final year:

For the exact deadlines for each milestone refer to the Moodle page (which goes live at the beginning of Term 1 of your final year). Here we use the weeks of each term numbered: 1-5, reading week, 6-10; **End of the week means by Friday 4pm of that week**.

- **From beginning of Term 1 till end of February:** main project development work. You must meet the following milestones:
  - **By the end of the reading week in Term 1** submit a Project Plan.
  - **By the end of the 6th week in Term 1** complete an ethics assessment of your project with your supervisor. Beware, this might involve additional steps and hence time, depending on the nature of the project (e.g., data protection, ethics application).
  - B**y end of the 2nd week in Term 2** submit an Interim Report.
- **March and April:** refining, testing, data analysis and writing of your Main Project Report
  - By the end of last week of **Term 2 submit a video review (o**ptional but recommended!**).**
- **By the end of the 1st week of Term 3** submit your Project Report online via the project submission link on the Moodle page. Note this is a provisional date and is subject to change by a few days (refer to Moodle page for all up-to-date deadlines). Also note that we do not know when specific exams will be timetabled, and you may well have one or more exams close to the submission date. Ideally, aim to have your project completed and ready for submission 2 weeks before the start of Term 3. This will allow time for your academic supervisor to go through it and give feedback.

2 What makes a Project

2.1 Topic

The topic of your project will be arrived at by discussion with your allocated supervisor and your external client if you are taking an industry project. Often it will be within the supervisor's research area and may contribute to their research activity.

A good project will have depth and challenge, making use of the Computer Science material you have been learning about. Projects also provide a great opportunity to learn about new ideas and concepts not covered in the taught modules you take. You can try out different programming languages, experiment with cutting-edge technologies or explore a new area of Computer Science. Don't limit yourself to what you already know!

There is a wide range of possible topics for projects. The main requirement is that all projects need to be focussed on computation and Computer Science in one form or another, making them relevant to your degree programme.

All projects are required to include research into the problems being investigated, analysis of the issues, identification of potential solutions, the design and implementation of the chosen solution, and a thorough evaluation of the results.

The majority of projects each year aim to design and implement a piece of software, so programming will be a significant part of many projects. Within this area, however, there is a lot of flexibility. The software may be a user application developed against a detailed set of requirements or for an external supervisor, but equally it may be a tool needed to carry out experiments (for example, to measure user responses for human-computer interaction experiments). Another variation is that the software can be a proof of concept that a design or algorithm can be implemented and have the desired performance requirements.

Projects in areas of Computer Science that do not involve a significant amount of programming are just as valid. For example, you may want to work on a more theoretical problem perhaps involving a lot of mathematical analysis. Providing your supervisor deems the project to be suitable, then it can be done. If you undertake such a project it should demonstrate relevant skills in areas such as mathematics, analysis, synthesis, design, critical assessment and evaluation.

2.1.1 MEng Projects

The challenge and depth of a MEng project is expected to reflect the additional year of study that a final year MEng student has. This needs to be taken into account when talking to your supervisor or making a project proposal. You should identify one or more significant problems that require a good amount of investigation to identify potential solutions, and the solutions should be cutting edge, making good use of computer science knowledge and theory, or recent developments in relevant areas.

While an MEng project is not required to be a research project, in the sense that it must uncover new facts or processes, the project should reflect a deeper understanding of the area being investigated that is expected for study at Master's level. In addition, there is an expectation of greater independence in carrying out the project, and more depth in the interaction with your project supervisor. The best MEng projects can actually be research projects, leading to potential publications in collaboration with the supervisor.

2.2 Supervisors

All projects require a supervisor who is a member of staff in the department of Computer Science. Most members of the academic staff, and some members of the research staff, can be project supervisors. Supervisors will be allocated to students as previously explained.

A project may have more than one supervisor within the department, typically to provide additional expert knowledge or to act as a client for the work being done. One arrangement is that the academic staff supervisor delegates the regular supervision to one of their research staff or research students, while still overseeing the project.

2.3 External projects

Any project in collaboration with someone outside the department can have an external supervisor. An external supervisor is someone not registered as a Computer Science Examiner here at UCL; essentially anyone from outside the department, whether a UCL academic or not. This means that the external supervisor does not do assessment of your project but can provide support and feedback in the same way as the internal supervisor.

Every project with an external supervisor *must* also have an 'internal supervisor', who is an examiner within the department. In general, you cannot count on your external supervisor being able to provide you with help in areas such as technical support in programming, or how to write a good project report. They will be an expert in the domain of application of the project, but not necessarily expert in Computer Science matters. Hence, check how much technical support an external supervisor can offer. If it will be limited, make sure that your internal supervisor (or even your industry mentor for an IXN project) is knowledgeable in the necessary areas.

External supervisors will be expected to provide 'domain knowledge', anything you need to know to tackle the problem that is not covered on our modules. In the past this has ranged from the physiology of the human ear to the tourist industry in Northern Cyprus. The external will be the source of the project idea and may in some cases take on the role of a 'client', whose requirements you may need to survey in some detail before even beginning to think about coding up a solution. Your external supervisor doesn't need to be a computer scientist, and probably won't know what a UCL CS project report is supposed to look like in detail. Getting your work into the required shape is the responsibility of your internal supervisor.

You should, in addition to your weekly meetings with your internal supervisor, be in regular contact with your external supervisor too. It is also helpful if your internal and external supervisors can be in occasional contact – phone or email contact is fine.

Where there is any difference of opinion between your internal and external supervisors, you must be guided by that of your internal supervisor. That person is more fully aware of the academic requirements that you must meet, and of the rules and procedures of the department. If your external supervisor does not

easily accept this, get your internal supervisor to contact them and resolve the situation.

An external supervisor will not be a registered Computer Science Examiner, meaning that they are not able to be either a first or second examiner for a project. However, their input to the marking process can still valuable, and the first examiner (the internal supervisor) can take into account the external supervisor's comments and feedback before deciding on a mark. Another member of the CS department, the second examiner, will then provide a further independent mark.

Finally, although we welcome the involvement of external supervisors as a source of interesting project ideas, we can't guarantee their availability or that projects in specific areas will be available. We can't, for example, offer up new projects in response to requests like "can you find me a project where I'm working with a bank?", if we have not already been contacted by a potential external supervisor in that area.

2.3.1 Legal Issues, Intellectual Property Rights, and NDAs

The department now has a Strategic Alliances Team (cs.strategicalliancesteam@ucl.ac.uk), which manages Intellectual Property (IP) issues and the formal interactions with external clients. This means that if you take up a project with an external client, the Strategic Alliances team should be informed and will guide you through the process of setting up your project properly. The team deals with much of what needs to be done, making it very straightforward for the student, so don't be put off taking a project if the client needs a contract or similar arrangements. All IXN projects will be managed by the team from the start.

For most internal projects you own the IP on what you create by default, but for IXN and potentially other projects with external organisations, the IP will be either owned by the client or transferred to the client as part of the contract. If there is a close connection with funded research work or some other substantial input from your supervisor or another UCL source, IP may also be an issue. For such projects the Strategic Alliances team has a standard contract template that can be used to set up contracts between UCL and the client, and between UCL and the student, placing UCL and the resources of the UCL legal team in the middle. The contract will properly define issues such as who owns the IP or what the deliverables should be, and make sure that everyone is properly legally protected. Should the IP created turn out to have significant value for the external organisation, the contract requires them to report back to UCL, and negotiate over the use of the IP and how the value generated is distributed.

Sometimes an external supervisor or client may be concerned about how widely information about a project, or the results produced, are made available. You should make sure that your external supervisor or client knows that you have to write a detailed report on your work in order to get a decent mark, and that they

cannot demand arbitrary changes to your report or delay submission of the report. The report will be read by the various examiners, so there cannot be unreasonable restrictions on their access to the report for assessment, but you can request that the report is not made available to anyone else.

However, not all details of a project have to go into the final report. Those that are most sensitive as far as your external supervisor is concerned may not be very important from our point of view. For example, in the case of a project with a bank on using neural nets for credit rating purposes, all that was required was that the origins of the vector components input to the neural network (relating in some way to an individual's credit worthiness) not be discussed. Since from the point of view of a neural net (and its trainer) these are just a string of numbers, this didn't really affect the write-up at all, and instantly converted the report from very sensitive in the eyes of the external supervisor to totally innocuous. Similarly, data sets used for data analysis or testing algorithms do not have to be submitted, as usually it is their structure and range of values that are significant in the report, not the full data.

If the contents of your report are felt to be sensitive you have the option of adding a restricted access disclaimer to your final report, stating that it can be seen by supervisors and examiners only (see section 4.5.3).

You should *not* under any circumstances sign any form of Non-Disclosure Agreement (NDA) if asked by the client. It is strictly against both the department's or UCL's policy to allow NDAs for project work and no such restrictions should be placed on projects. If your client asks about an NDA then refer them straight away to the Strategic Alliances team. Neither the department or examiners will observe any NDAs, and examiners will not agree to any NDA in order to mark a project.

2.4 Ethical Considerations and Ethical Approval

Any project that involves any form of study or data collection with people or animals, or involves any data that can be used to identify people (on its own or in conjunction with other data like social media), or where the results of the work have potential moral implications (e.g., the use of face recognition) must have ethical approval before the study/experiment/data collection is carried out. Please note this does not rule out other required activities related to your project, such as literature review, analysis, design and coding.

Your project plan (See Section 3.5) will need to include a statement of whether you believe there are ethical issues with your project, and an account of what those issues might be (or if there are not, what you have considered and why such issues do not apply to your project). Your supervisor will review this statement with you and will submit an application for ethical approval on your behalf where they believe it is needed. Applications for ethical approval are submitted to the

departmental ethics committee and approval may then be granted by the Head of Department, or in some cases (which will take longer), by the UCL Research Ethics Committee.

When writing your project proposal, you will therefore need to assess if there are likely to be ethical issues with your project. To make this assessment you will need to take the department's ethics training course available in Moodle: https://moodle.ucl.ac.uk/course/view.php?id=16849 (please check our Moodle page for enrolment details and password). As part of that you must complete the UCL data protection training for students (this may help to reduce the registration requirements for your project).

Many projects will not need ethical approval as they involve only design, coding or mathematical work and don't have any of the aspects mentioned in the first paragraph of this section. Increasingly, though, projects are making use of data sets obtained from a wide variety of sources and this does need to be checked carefully and possibly approved before you start using them. If there are potential ethical issues a straightforward response might be to alter the project goals to remove the need for ethical approval, for example by adding a goal to create synthetic data sets to avoid having to obtain and use real data.

Also note that use of data containing any directly or indirectly personally identifiable information about real people is covered by the General Data Protection Regulation (GDPR) and the UK Data Protection Act 2018, and this states that all such information must be kept with an adequate degree of protection and security, and used only on a legitimate and identified legal basis. For that reason, we very strongly recommend that you use fake or manufactured data wherever possible. If this is not possible, then it is essential that you satisfy the legal requirements for data use in accordance with UCL policy, as well as satisfy the ethical considerations. You may need to register your project with the UCL data protection office (even if ethics approval is not required). Again, this process should be overseen by your supervisor (and the departmental ethics committee can advise them on how to do this), and you will need to provide details of the data you wish to hold and the security measures you propose to put in place before you use or capture the data (or receive it from a project partner). Once the project is completed and marked, you should have arrangements in place to delete the data properly and confirm that is has been deleted.

You must not carry out any work requiring ethical approval (e.g., interviews with people or analysis with participant or other identifying data) until that approval is granted. Note also that the approval you receive may be granted on condition that some changes are made to the work you are planning to do on the project.

2.5 Project Resources

The great majority of projects can make use of readily available resources:

- Lab computers or your own machine
- Standard operating systems and software
- Open-Source software
- On-line and UCL Science Library resources
- Small items of hardware such as Arduino, Raspberry Pi, Kinect, smartphone or tablet
- Cloud based virtual computing services, either via services run in the CS department or on services such as Microsoft Azure

These resources are either provided to you as a UCL student (hence available for free) or, in the case of computer hardware (desktops, laptops, tablets or smartphones), you probably already own. Some hardware can be borrowed from the departmental pool of load equipment, check with the Computer Science Help Desk on the 4th floor of the Malet Place building (see https://tsg.cs.ucl.ac.uk/contact-us/).

The web, of course, has a great deal of material available but don't ignore the UCL Science library as it gives you access to the full digital libraries of societies such as the ACM and IEEE. For research purposes, especially if you are doing a research-oriented project, this will give you access to a very wide range of research and academic publications. In addition, you can access the e-book service via the Science Library, which will give you access to a wide range of CS textbooks.

For software design and development, there are many high-quality Open Source tools and other software available, particularly for languages like Java, Groovy, Ruby, Python, C, C++ and UML. This software is available for free download and many projects make use of this resource.

Do not pirate (i.e., steal) software or use any software that you don't have a valid license to use. Always respect copyright and licenses. Being found to have used pirated or unlicensed software can have serious implications for your project.

A wide variety of licenses are used with Open-Source software (such software may be free to download but will still be licensed). The licenses may have no impact on your project but if you are planning to make your software available to other people, make it publicly available as an Open-Source project, or you are developing software for an external supervisor, you do need to take notice of which license is used and what its consequences are. In particular, a license like the GNU Public Licence (GPL)[1] requires source code to be made available to users

[1] http://www.gnu.org/licenses/gpl.html

of the software (notably the GPL3 variant), which can have important implications for the code you write.

2.5.1 Additional Resources

If your project needs non-free or specialised resources then it is the responsibility of your supervisor (internal or external) to provide those resources. Make sure at the start of the project that such resources will be available and that you will have proper access to them. Also make sure that documentation, training and safety procedures are properly available if needed. Beware of delays in getting access to resources, especially if they are critical to your project.

There is no 'project budget' available for buying software or hardware if your supervisor cannot provide it or items are not available in the departmental loan pool. Also we do not expect you to spend your own money to pay for such resources. If the resources you need for your project idea are not available then you will need to modify the project or choose a different one.

2.6 Backing up Your Work

You are reminded that if you are working on your own computer, it is YOUR responsibility to back-up your work and deal with hardware problems; unexpected software or hardware problems cannot be used as excuses for missing the project submission deadline.

You should make emergency plans for any breakdowns, perform regular backups (preferably by copying your files onto your CS file store at UCL) and not use pirated or dubious software. If you use Microsoft Windows make sure you use good virus checking software. On-line backup services, such as Dropbox or OneDrive, provide a good way of keeping copies of your work, and many are free for several gigabytes of data or more. You have full access to Microsoft Office 365 via your UCL computer account, which includes One Drive and related software.

Be aware that you may not be permitted to use or back up personally identifiable data used in your project anywhere other than at UCL or on a UCL contracted service like OneDrive for Business (personal OneDrive, Dropbox, GitHub, bitbucket etc are not suitable for such data).

You should also make use of version control, especially for programming work, so that your source code and other documents are properly managed. You can set up your own version control server or use cloud-based services such as GitHub or Bitbucket. GitHub offers free Student Developer Packs with free unlimited public or private repositories, see https://education.github.com/pack.

If you don't want to use a commercial service, CS has a managed GitLab service available[2], allowing you to keep your remote source code repositories on a CS server. This may be preferable if you or an external client have concerns about keeping source code on a commercial service.

2.7 Security Issues

The UCL firewall imposes strict limits on external access to machines within the department. While you can access departmental machines via secure mechanisms such as ssh and VPN, these are intended for personal use only, not to provide more general access from outside UCL. Normally it is not permitted to set up a server, such as a web or database server, on a departmental machine and access it from outside UCL, especially to run any sort of publicly available service, and certainly not any sort of commercial service. There may be ways to circumvent the firewall, doing so would be considered a serious security violation, so please don't try!

If you need to run a server or cloud-based service then ask for access to a either a virtual machine hosted in Computer Science, or ask your internal supervisor about access to cloud services. Virtual machines are very suitable for running a wide range of services such as web applications, database servers, and other online services you might be implementing.

Be aware that if you are going to use personally identifiable data in your project (see Ethics above), you may not be permitted to store or use it anywhere other than on a UCL server (including not downloading it to your own computer), and you may need to plan your project's technical requirements accordingly (e.g. ensure you can run your code through an ssh terminal on a UCL server).

3 During the Project

3.1 Preliminary Activities

Over the summer you should do research and background reading as part of your general preparation. Activities might include learning a new programming language or development framework, reading relevant books or papers, and generally building up your knowledge of your proposed subject area. However, your project supervisor will not start normal supervision of the project until the start of the academic year (don't worry your supervisor will be supervising MSc projects over the summer, so will not be short of projects to keep them occupied!).

[2] The CS GitLab service is at https://gitlab.cs.ucl.ac.uk

3.2 Tutorials

As early as possible in October you should book regular *weekly* project tutorials with your supervisor. This applies even if your project is in part being supervised outside the department and you are also seeing your external supervisor regularly. Tutorials should be held every week during term time, with the exception of reading weeks. Outside of term time you should make suitable arrangements if you need to see your supervisor.

Note that it is your responsibility to arrange and attend tutorials, and to be properly prepared for each tutorial. Make sure the day and time of each tutorial is in both your own diary *and* your supervisor's diary. It is also a good idea to send a *meeting agenda*, describing what you want to cover in the meeting and particular places where you would need input. This will allow your supervisor to catch up with where you are, prepare for the tutorial and make better use of time.

At the start of each tutorial you should report on what you have done in the last week, then move on to discuss any problems or issues, your plans for the next week and so on. Your supervisor will give feedback (criticism and praise!), help you plan your work and check that the project is running properly. It is also a good idea to show examples of your work, demonstrate your program code running or highlight results produced, so that your supervisor can build a clear picture of what you have produced. Be honest with your supervisor, especially if your progress is slower than you expect or you are having trouble understanding things.

Always remember: this is *your* project. You should be the one in charge and driving it forward. Your supervisor is there to give you help, advice and feedback but not to do your work for you! Don't expect to be given a list of instructions on what to do each week; make your own decisions.

3.3 Establishing the goals of your project

This is the single most important aspect of the initial phase of your project. The goals of your project must be sufficiently clear to you before you move on to the main phase of work. Discuss and agree the goals with your supervisor(s):

- What is your project fundamentally about?
- What are you intending to design/build/investigate?
- What do you intend to deliver as the project results?
- What would constitute, in your own and your supervisor's eyes, a 100% satisfactory solution?
- In the worst case what is the minimum that needs to be completed to achieve a pass?

- What are your personal aims that you hope to achieve?

After you have thought about these issues, it should be possible for you to reduce your project description and goals to one page of A4. If you can't summarise your goals in this way then you need to do some further thinking and planning. Make sure that your supervisor is in agreement with your aims as finally established.

It is strongly advised that you keep notes of your progress in the form of a logbook, detailing the decisions you make as the project advances. You will find this invaluable when it comes time to write your project report.

3.3.1 Finalise Requirements

If you are working with an external supervisor, or developing software for someone, it is important not to let the requirements keep changing or the goals to drift, so that you start to lose control. Define a cut-off point (e.g., around the Christmas break) where you no longer expect any significant changes, and make that clear to anyone you are working with. This is to avoid the scope of the project getting out of control and delays caused by people changing their minds.

In practice, this may not be such a straightforward strategy, though. As your project progresses you may find some ideas don't work, there are new ideas to investigate, or the requirements don't really become clear until some significant work has been done. In that case, do a short review to decide the best strategy to follow to manage the change, and get advice from your internal supervisor.

3.4 A Typical Project Schedule

A typical project schedule is as follows:

- Summer till late September: Check-in with your supervisor and set up project tutorials. Undertake the preliminary work on the project. This includes detailed background reading, analysis and design work, coding small test programs and simple prototypes. Also identifying and learning how to use tools, programming languages and so on.
- Up to late-October: Complete your preliminary work and ensure that the aims and goals of your project are absolutely clear, documented and approved by your supervisor. Also complete the ethical assessment of the project with your supervisor.
- Late-October to early February: The real work! The bulk of the detailed requirements modelling, analysis, design and implementation (serious coding, not just small tests and prototypes) or equivalent work relevant to your project.
  - Mid-October: Project Plan to be submitted. See Section 3.5.

  - Mid-January: Interim Report to be submitted. See Section 3.6.

- Early to mid-February (Reading Week), finishing off the main design and implementation or equivalent work; planning the final project report. You should also put together a chapter and section outline for your report and get your supervisor to approve it. This is also a good time to create a storyboard for your video preview submission, in which you figure out how to explain what your project is about in a clear way. As a next step, write a first draft of your introduction chapter, elaborating on your project description.

- Mid-February (Reading Week) to end of Term 2 (Video preview Deadline): Your project should be well underway. Implementation should be mostly finished, and you should focus on validation. This can include a range of activities, from performance tests, simulations or user studies, but also the data processing and analysis required in order to show the degree to which your project meets its goals. Reflect these in your video preview, explaining how you will validate the success of your project (although you might not be able to report your results yet, as testing/analysis might still be underway).

- Second half of Term 2: Focus on writing your main project report. At this point you should largely stop any implementation or equivalent work and concentrate on writing. Your ability to present and explain your design decisions and results is just as important as artefacts such as code, and they might take just as long to produce. Also, keep in mind that most of the marking will come from whatever markers can assess through your report. If something was hard and made you sweat, you have to show it in the report. Finally, writing might help you identify aspects in your project that are not properly tested or validated. Writing early will help you identify these last critical aspects that you need to implement/test. Read through the guidelines for the report structure given in this document and make sure that you keep to the format requirements in relation to page limit, font size, and so on.

  Don't underestimate how long it will take to complete the writing and detailed editing of your report.

- The end of Term 2 would be an appropriate time to give your supervisor a demonstration of the final working system or a detailed explanation of the results. Your video preview can also act as an excellent source to help you explain your system. You are very strongly advised to try to have a complete first draft before the end of the Term 2 so that you have time over the Easter break for concentrating on your exam revision.

- The start of Term 3: Make this your target for having a complete final version of your report, to give you time to deal with any last minute issues. You are also encouraged to submit this version of the report to your supervisor 2 weeks before the final deadline, for final comments and feedback.
- The submission deadline: This is the date and time by which you must submit your project report (see Moodle for the date and detailed information about submission). The date is unlikely to change but if a change is necessary it will be announced by email or Moodle forum. You are required to check your UCL email regularly (at least once every day, more regularly as deadlines and exams approach); not reading your email is not an acceptable excuse for missing a deadline.

It is very easy to over-estimate what you can achieve in the time available. Also your other modules and coursework will be competing for your time. Monitor your progress carefully and focus on the elements that will have the most value to your project. By far the most common lament supervisors hear is along the lines of "I've only done half of what I expected by this time". The chances are that you will say something like this too, it is quite normal!

Many projects are best structured by taking an iterative or incremental approach in the main development phase. Having determined what you are trying to design and build, start by constructing a very basic but working version of it. Then step-by-step add one feature or requirement at a time, so that you go from one working version to the next. Keep going while there is time available, reviewing your progress after each step and select the next most important feature to add. It is unlikely you will have time for every feature you would like to have, so make sure you focus on adding those that gives most value to your project. Some features may have to be dropped or implemented in a basic form to demonstrate their feasibility but no more.

3.5 The Project Plan

You are expected to produce a project plan as your first project deliverable, to confirm that you have started on a viable project and know how to proceed. This is not assessed as part of your final project mark, but nevertheless must be completed on time and to an acceptable standard. Your supervisor should check the plan and once they have approved it as valid, they should complete the required form (Link on Moodle) uploading a copy of your plan. You must also include an unedited copy of this document in the appendices of your main project report.

How detailed the plan needs to be should be determined by your supervisor. This is not intended to be a difficult task, somewhere between one and three pages of text should normally be sufficient.

Your plan must include at least the information outlined below; your supervisor may wish you to include further material.

- Your name
- Project title (see Section 3.5.1)
- Supervisor's name
- External supervisor's name (where relevant)
- Aims and objectives (see Section 3.5.2)
- Expected outcomes/deliverables (see Section 3.5.3)
- Work plan (see Section 3.5.4)

3.5.1 Project Title

Make your title clear and concise. Avoid the use of abbreviations or acronyms where possible, except for those in common usage (such as 'HCI' for 'human computer interaction').

Don't worry if you want to change your project title as the work develops. Provided that it does not reflect a major change in topic, in which case you should consult with your supervisor as to the advisability of submitting an entirely new Project Plan, just use the new title when the time comes to prepare your Interim and Final Reports.

3.5.2 Aims and Objectives

Aims and objectives should be clearly distinguished.

Aims represent your project's overall purpose and what you personally want to achieve; objectives are specific and measurable achievements on the way to completion, the means by which you hope to accomplish your aims. For example:

> **Aim:** *"To learn about amino acid sequences and then determine whether the representation of amino acid sequence in a Fourier-transformed format can aid neural network prediction of protein fold types from sequence data alone."*
>
> **Objectives:**
>
> ***1.*** *Review the 'protein folding problem' and previous techniques for predicting protein folds from amino acid (primary) sequence, in particular those using artificial neural networks.*
>
> ***2.*** *Develop software tools that can pre-process primary sequence data using Fourier transform methods in order to produce fixed-length data vectors suitable as inputs for neural network training.*

3. *Develop an appropriate neural network model that can use the above data vectors as input and as its output classify amino acid sequences into one of a number of fold types.*

4. *Evaluate the success of the method using Fourier pre-processing compared with previous neural network based fold type predictors, and with other forms of structure predictor (for example those using logic programming).*

3.5.3 Deliverables

List the outputs or artefacts to be created during your project, which you feel represent significant and useful pieces of work.

You might include for example:

- A literature survey that summarises previous work in your specialist area in a way that makes in accessible to a more general reader.
- Results obtained from gathering available data, questionnaires and so on, with statistical analysis and discussion of their significance.
- A new model or algorithm developed (by yourself or in collaboration with your supervisor) to solve your project's underlying problem.

Everyone should also normally include:

- A design specification for a software application or equivalent specification for relevant deliverables.
- A fully documented and functional piece of software or equivalent.
- A strategy for testing and evaluating your software or equivalent.

3.5.4 Work Plan

The aim of the plan is to demonstrate that you have a clear idea of how to proceed with your project, and that you have a realistic picture of the work involved.

Start by reviewing your project's objectives and break them down into appropriate sub-tasks (though it is probably not worthwhile trying to break down the work so finely that you include activities expected to take one week or less).

Clarify the order in which these tasks must be performed, and estimate the time that will be taken by each piece of work, and the particular time period over which you expect to be working on it.

Sequence your planned activities – remembering also to include the preparation of your Interim Report (due in early February) and Final Report (with a deadline in Term 3) – so that your aim is to finish your Main Report by the end of Term 2; this should take into account 'slippage' time and ideally get your project out of the way before you embark on the bulk of your revision for the May examinations.

Present the plan in the following way, adding detail relevant to your specific project. Please note this is just an example (e.g. the project plan, video preview and interim report are not included). Your plan should add items specific to your project, according to the tasks required and methodology followed:

- Project start to end of October (4 weeks) Literature search and review.
- Mid-October to mid-November (4 weeks) Analysis and modelling.
- November (4 weeks) System design, coding small test programs and simple prototypes.
- End November to mid-January (8 weeks) System implementation.
- Mid-January to mid-February (4 weeks) System testing and evaluation.
- Mid-February to end of March (6 weeks) Work on Final Report

The plan above presents a relatively linear sequence of activities. In practice it is often a good strategy to take a more iterative approach, where you break down the work into a sequence of iterations based on the requirements you identify. Each iteration then realises a small sub-set of the requirements, chosen in order of priority. At the end of each iteration you can review and update your requirements based on your progress so far and what you learnt during the last iteration.

Each iteration will involve a combination of analysis, design, implementation and testing. Early iterations develop a basic working version, with successive iterations building up more complete versions. An iteration should last 1-2 weeks at the most.

An iterative plan outline would have a structure like this:

- Project start to end of October (4 weeks). Literature search and review. Start defining your requirements.
- Mid-October to mid-November (4 weeks). Refine your requirements and start the initial iteration(s).
- November (16 weeks) to mid February. Work through the iterations. Include here a list of primary features you plan to add.
- Mid-February to end of March (6 weeks). Work on Final Report

Specific milestones should be added as appropriate to your project. For example, the dates where you intend to have a first working version or a complete working version.

3.5.5 Ethics review

You must include a statement of whether you believe your project will require ethical approval (See section 2.4) and a review of the ethical issues raised by the planned work (or those that were considered but not seen as relevant). The review should follow the Guidance video 'Assessing a project for ethics' on the departmental ethics training Moodle. Your supervisor will review your statement and assessment and will make an application for ethics approval on your behalf as needed.

3.6 The Interim Report

The purpose of this report is to confirm that you have made substantial progress and know what is required to finish your project properly. A current status section should give a good summary of what has been done so far (at least 3-4 paragraphs). The remaining work section should list the work needed to complete your project along with the time needed and milestones/deadlines you are setting yourself.

This is not assessed as part of your final project mark, but nevertheless must be completed on time and to an acceptable standard. Your supervisor should check the report and once they have approved it as valid, complete the associated form on Moodle.

Aim to be clear and concise; try to limit your Interim Report to around two or three pages of A4. The report must include at least the information outlined below. Your supervisor may wish you to include further material, depending on the progress of your project.

- Name
- Project title as given in your October Project Plan
- Current project title
- Your internal supervisor's name
- Your external supervisor's name (where appropriate)
- Progress made to date
- Remaining work to be done before the final report deadline

Failure to submit an Interim Report and demonstrating that an appropriate amount of work has been done will be seen as evidence that your project is at very serious risk of failure. Expect action to be taken to correct the situation. There will still be time to try and remedy the situation but you will have to commit to working very hard to catch up.

Don’t forget that you should have a project tutorial with your supervisor at least once per week and you should always keep your supervisor up to date with progress. If you are having problems talk to your supervisor about them earlier rather than later, or talk to the Projects Organiser.

3.7 The Video preview [Optional]

You are encouraged to create a short video (ideally 2-3 minutes long, maximum 6 minutes) explaining your project, a task that serves a variety of purposes.

First, you should consider the video as a tool to help you explain your work. Writing a Project Report might be an intimidating task at first. Even if you start writing early, facing an empty page is always hard, and it you can easily get stuck if you try to progress through each section, covering it in depth and detail.

The video should cover all aspects that one needs to know about your project (check *What to write about* in Section 4.1). However, having to cover all of that in 2-3 minutes will force you to be brief in each point, fleshing out what is really important for your project. You will also identify key images and diagrams that you need to explain your work (please note, you do not need create a “movie”, your video is very likely to be a few PowerPoint slides, with you explaining them). The ideal outcome is that the video will force you to identify the key messages that you need to highlight in your report, and effective visuals to explain them, and both things will make writing your report a lot easier.

Secondly, your video will act as a way to disseminate your work. It will be stored in our department's media archives, as well as made available to be published on our comms channels and our upcoming CS Fair. You are also encouraged to use it in your personal webpages and social media, as this can be very valuable for job-hunting to showcase both your technical and presentation skills.

Finally, the video should act as a point of reflection during your project. Even if you are doing it briefly, you are explaining all aspects in your project, even bits that you might have not completed yet (e.g. testing, studies). Does the plan still seem feasible? What are the key remaining tasks/steps that you definitely need to take to support the goals of your project? Which ones are optional? Are you missing any steps (e.g. tests) to actually demonstrate a key aspect of your project? Use the video as a reflection point to identify key aspects of your project, evaluate your progress and plan for the time ahead. Sharing it with your supervisor will also help you make the most of your video, as your supervisor will be able to give you feedback, honing your key messages and figures.

This is not assessed as part of your final project mark, but you are strongly encouraged to complete it. Your supervisor should check the video and once they have approved it as valid, complete the required form (link on Moodle).

4 The Project Report

The Project Report documents the results of your project work and is an important deliverable as it will be read by all the examiners that mark your project. Further, to achieve a good mark for your project, your report must be of the appropriate quality. Excellent design, programming or the equivalent, are not enough on their own to gain high marks, you must also submit a matching high quality project report.

Examiners give credit for good quality writing, so aim for a clear and concise writing style. You should use relevant notations, terminology and demonstrate relevant computer science knowledge. The proper evaluation and critical assessment of your work is also important. Above all your report should be both readable and interesting to read. Beware of creating a report that is too long, tedious to write, and tedious to read. Write something that your examiners will want to read!

Before starting to write your report you should plan its structure by creating a contents outline and getting your supervisor to review the outline. Then, as you write your report, particularly during the second half of Term 2, you should be showing your supervisor draft chapters to get feedback. Note, however, that it is *not* your supervisor's job to be your editor or proof-reader, so don't expect your supervisor to fully proof-read everything and comment in minute detail on your drafts.

It is strongly recommended that you plan your contents outline early in term 2 and also begin to do some writing in order to get started – getting started is often the hardest part of writing the report so don't keep putting it off! In the second half of Term 2 writing your report should become the main project activity. Don't underestimate how long it takes to write *and* edit a good report. The worst mistake to make is to believe you can write your report quickly right at the end of the project; you can't, and won't have time do a good job. Be ruthless when editing to get the text into the best shape possible.

4.1 What to write about

1. Write about the *interesting parts* of your project. No-one wants to read about every single detail of how you implemented your project! Devote appropriate space and time to this aspect, addressing issues such as:

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

- What did you learn?

2. Write about the context in which your work fits. You should provide enough background to the reader for them to understand what the project is all about. For example:

   - What problem are you solving?
   - Why are you solving it?
   - How does this relate to other work in this area?
   - What work does it build on?
   - What is the scope of your work?
   - What is included in the scope and what is outside the scope?

   You should assume your reader is a computer scientist, but may not be an expert in the area of your project or of any tools or building blocks you've used, so help them understand.

3. Describe any preparatory work you needed to do. This includes researching ideas, concepts and tools, that will be needed to do the project work. A literature survey is an important way of mapping out the relevant subject knowledge of the area your project is in.

4. Capture the requirements clearly. Some projects involve detailed requirements capture, and for others the requirements are essentially handed to you. If you had to do detailed requirements capture (e.g., a survey of potential users, etc.), then this might be described in some detail. If you didn't do detailed requirements capture, then just summarise the requirements.

5. Give a good overview of your development work. If your project involved designing a system, give a good high-level overview of your design before going into specific details. In many projects, the initial design and the final design differ somewhat. If the differences are interesting, write about them, and why the changes were made.

   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 (e.g., it turned out to be difficult for unexpected reasons), write about it.

6. Describe how you evaluated your work. Most projects involve the creation of software. Building software that works well is difficult, and you may have spent a lot of time on testing. In most cases your reader does not want to know about all the detailed tests you ran, but they will be interested in your testing strategy and philosophy. Some software is particularly hard to test,

and in such cases you might need to go into quite some detail on why it's hard to test, and how you overcame this obstacle.

Some projects involve designing systems whose final correct behaviour is not known in advance. Usually such projects would be classed as "research" projects. Typically such a project report will devote a lot of time to evaluation of how the final system behaved, and where it worked well and where it didn't. No system behaves well in all circumstances, so your reader will be interested in both how well it works, and in the circumstances in which it works less well.

7. Provide a critical assessment of your work. Your reader wants to know that you understand the advantages, disadvantages, strengths and limitations of your work. No project is perfect, there's never enough time for that! Give a good critique of your work.

   Some examples:

   - Did the design do the job you intended, or were there problems?
   - Is your solution fit for purpose?
   - What is the quality of your work, and how do you measure it?
   - How does the resulting system compare against the competition?
   - If you didn't finish implementing all the features, how hard do you think it would be to do so?
   - What advice would you have for someone who wished to take your system and use it or extend it?
   - What would you do differently if you could start all over again?

8. Provide a User Manual. For many software projects you need to include enough information to show how to use the software and what it looks like. A user manual is a good way of doing this, probably as an appendix section. Some projects are heavily user-centered, and so the user interface is a key part of the project; in these cases a user manual would not just be an appendix, but a core part of the project report. It may be the case, though, that a user manual is not be the most effective way to present your ideas, so use your discretion.

4.2 Report Content Structure

The structure of a typical report is outlined below. Your report structure can and should vary to suit your specific project. Don't feel compelled to write text to fill out each section described below, especially if there isn't anything interesting to write about. Add sections that are relevant to your project.

A report should be broken down into a series of chapters, with each chapter consisting of a sequence of numbered sections and sub-sections. A typical list of chapters for a design and programming project is described here but, remember, you can adapt or replace this to fit the needs of your project. You can also trust your supervisor if she/he suggests a different structure, more suitable for the specifics of your project.

- Chapter 1 Introduction
  - Outline the problem you are working on, why it is interesting and what the challenges are.
  - List your aims and goals. An aim is something you intend to achieve (e.g., learn a new programming language and apply it in solving the problem), while a goal is something specific you expect to deliver (e.g., a working application with a particular set of features).
  - Give an overview of how you carried out the project (e.g., an iterative approach).
  - A brief overview of the rest of the chapters in the report (a guide to the reader of the overall structure of the report).

  This chapter is relatively short (2-4 pages) and must leave the reader very clear on what the project is about and what your goals are.

- Chapter 2 Context (or choose a chapter title that fits the material best)

  This chapter should cover background information, related work, research done, and tools or software selected for use in the project.

  - Provide necessary context and background information to describe how your project relates to what is already known or available.
  - A description of the research carried out to learn out about the nature of the problem(s) being investigated and potential solutions. The form of the research will vary widely depending on the kind of project. For example, it might involve searching through research publications and online resources, or might involve an exploration of design possibilities for a user interface or program structure.
  - The sources of information you are drawing on (papers, books, websites, etc.) should all be cited or referenced clearly. In addition, state how each source relates to your work and avoid the temptation to pad out the chapter by including sources that you didn't make use of during the project.
  - If relevant, a survey of similar solutions, programs or applications to yours, and how yours is differentiated.

- Introduce the software, programming languages, library code, frameworks and other tools that you are using. Discuss choices and make clear which you made use of and why.

You should not include well known things (e.g., HTML or Java) or try to give tutorials on how to use a tool or code library (use references to books and websites for that information). Everything you include should be directly relevant to your work and the relationship made clear. This chapter is likely to be fairly substantial, perhaps 8-10 pages.

- Chapter 3 Requirements and Analysis

  - Give the detailed problem statement. This elaborates on what you may have included in the introduction chapter, and represents the starting point from which requirements were derived.
  - A structured list of requirements.
  - Use cases (a use diagram and list of use case titles, with the full use cases appearing in the appendix).
  - Results of analysing the requirements to extract information. For example, data modelling to find the data to be stored (ER diagram), views/web pages needed and so on.

  The level of detail of the requirements and use cases will depend on the nature of your project. If you are doing a Software Engineering based design and implementation project, then they will need to be done thoroughly. If there is a substantial body of requirements and use cases, then a summary should be given in the chapter, with the full set included in an appendix section.

  If your project is not Software Engineering oriented, then you still need to describe the requirements you are working to and relevant analysis information. Use cases may not be needed or be relevant.

  The analysis part of the chapter is what you did to map the requirements information into the first pass design. You can think of analysis as the first stage of design, and the purpose is to show how the requirements were used to inform the design.

  The length of this chapter depends on the kind of project, but you are typically looking at 5-6 pages.

- Chapter 4 Design and Implementation

  - Describe the design of what you have created.

- Start with the application architecture, giving its overall structure and the components that make up that structure.
- Give a description of the design of each of the the components that make up the architecture.
- Include the database or storage representation.
- Provide implementation details as necessary.

As with other chapters, the structure and contents of this chapter will depend on the nature of your project, so the list above is only a suggestion not a fixed requirement.

Find an ordering and form of words so that the design is clear, focusing on the interesting design decisions. For example, what were the alternatives, why select one particular solution? You have a limited number of pages so be selective about details. Also remember that someone (your examiners!) has to read this so don't overwhelm them with intricate descriptions of everything that only you can follow – but do make sure the key details of the solution are in place. Use appropriate terminology and demonstrate that you have a good understanding of the Computer Science principles involved.

You can use diagrams and screen shots to help explain the design but don't overuse them. Diagrams and screen shots should add information, not duplicate what is written in the text, and definitely avoid page after page of diagrams as this will disrupt the flow of your text. Where relevant, UML diagrams can certainly be used but, again, don't flood the chapter with diagrams. Additional diagrams can always be included in an appendix section. It is not the case that a full set of UML diagrams must be provided for a software development project, and they shouldn't be added in the belief that there must be UML diagrams to do a good project. Think about what you need to communicate and use UML diagrams if and when they fit the need.

It may be useful to include sections of code to highlight how a particular algorithm is implemented or how an interesting programming problem was solved. However, avoid lengthy sections of code, as this can also disrupt the flow of the text. Also make sure that your code fragments are readable, easy to follow and properly laid out. It may be better to use pseudo-code rather than actual code, especially when describing an algorithm. If you need to make use of longer sections of code, you can put the code in the appendix and reference it from the text.

An alternative way to organise the content of both this chapter and the preceding one, suitable for some projects, is to have a sequence of chapters or sections for each major iteration of the project. This allows the progression of the project to be shown, with each iteration building on the last, and the opportunity for interesting discussion about the decisions that needed to be made.

This is a core chapter in your report and will usually be quite substantial, 10 pages or more.

- Chapter 5 Testing (and/or Results Evaluation)
  - Describe your testing strategy (unit, functional, acceptance testing and how they are carried out). How were test cases selected?
  - Examples of specific tests and how they were carried out (e.g., using mock objects to break dependencies). Focus on the interesting cases.
  - A summary of the test results and what coverage was achieved. Detailed test reports should appear in the appendix, if they add useful information or you want to demonstrate the kinds of tests and coverage achieved.

  If your project requires substantial evaluation of data and results, evaluation of algorithms, or other forms of testing that are not code-based, then adapt this chapter to suit.

  This chapter will typically be 2-4 pages in length but could be more depending on the depth of testing done. If you need to do a detailed evaluation for a more mathematical or theory-based project, then this chapter could well be more substantial.

- Chapter 6 Conclusions and Evaluation
  - A summary of what the project has achieved. Make sure that you address each goal set out in the Introduction chapter, to show that you have achieved what you claimed you would. Don't leave any loose ends.
  - A critical evaluation of the results of the project (e.g., how well were the goals met, is the application fit for purpose, has good design and implementation practice been followed, was the right implementation technology chosen and so on).
  - Future work. How could the project be developed if you had another 6 months. Take care to differentiate between what you have done to

satisfy your stated project goals, and work that could be done to meet extended goals.

- Wrap-up and final thoughts on your project.

This chapter is typically 2-4 pages long but could be longer if the project work requires more extensive evaluation.

- List of references. Give publication details for all the items referred to by references you have made in main text of the report.
- Bibliography. This lists all the sources of information that you made use of during the project but are *not* referenced in the text. The items in the list must be relevant to your project, so don't just list everything you may have looked at or read.

  The list of references and bibliography are often combined into one section labelled Bibliography.

You are not limited to the chapters suggested above but do remember that the report documents your *results* and is not meant to be a diary of progress or a novel. For some projects it may make sense to have a separate evaluation chapter before the conclusions and the way that requirements are specified can also vary as techniques like developing use cases may not be appropriate for your project.

4.3 Appendices of the Report

You should add each of the following appendix sections if they are relevant to your project. A source code listing should always be included, or an equivalent listing if your project has other kinds of outputs. In addition, there may be other appendix sections relevant to your project.

The appendices should provide additional *relevant* information to your examiners that ought to be present in the report but not as part of the main chapters. Be selective over what you include and avoid using the appendices as a dumping ground for unimportant or trivial information.

1. *System Manual* – This should include all the technical details that would enable a student to continue your project next year, and be able to amend or extend your code. For example, Where is the code?, What do you do to compile and install it?
2. *User manual* – If relevant to your project, this should give enough information for someone to use what you have designed and implemented. It is a good place to include screen shots of the application or output.

3. *Supporting documentation or data* – If you have additional data, diagrams or things like a set of use case specifications that are relevant to the documentation of your work then they can be included in an appendix section. Don't just include everything, only items that are directly relevant to the documentation or description of your work.

4. *Test results and test reports* – If you have test results that add to the value of the report, but which would not fit within the page limit of the main report, you can include then as an appendix. Don't add them just to pad the report, though.

5. *Evaluation data and results* – If your project involves activities such as generating, analysing and evaluating data of some sort, then sample data, descriptions of analysis methods and detailed results can be included here.

6. *Project Plan and Interim Report* – Put copies of them into an appendix section.

7. *Code Listing* – Your code should be properly presented and formatted neatly. Don't let long lines of code arbitrarily wrap round to the next line, as this looks very messy. In order not to use up too many pages you can switch the page orientation to landscape and print the code in two columns.

   In general, don't add more than about 25 pages of code listing to the report. If your code does not fit within 25 pages, you can provide a listing of the most interesting parts our your code (but include around 20-25 pages) or parts of the code you may need to reference from the main chapters. If you don't include everything, it must be clear that this is not the complete listing. State which parts you've included, add a brief explanation of why you've included these particular parts, and provide a brief summary of which code you have omitted.

   A full copy of your source code will be uploaded when you submit your project.

4.4 Alternative Research MEng Project Format

MEng projects can be quite research-oriented, in which case it is possible to submit the main part of the report in the form of a research paper or papers. The report should still include the correctly formatted front cover, the abstract, an introductory chapter describing the aims and goals, a conclusions and evaluation chapter, and appropriate appendices.

If you opt for this approach you must first obtain the agreement of your supervisor and then carefully follow their advice on how to proceed. Your supervisor will

guide you through the process of writing an academic paper. You will also be encouraged to give a research group seminar on your work towards the end of the project, which can be counted as one of the project deliverables.

4.5 Report Formatting

This section gives the basic formatting requirements that everyone should use. Your report will actually be submitted on-line and doesn't need to be printed on paper for submission. Nonetheless it should be formatted so it can be printed double-sided on A4 size paper, with proper page headers and footers.

The format requirements are not overly restrictive, for example there is no requirement for you to use a particular typeface. However, do not use too many different typefaces in your report, or in general spend too much time developing an elaborate visual presentation. It is better to keep the look of your project report simple and straightforward. An over-elaborate presentation can in fact create a negative impression, suggesting that the author thought the material was rather thin and felt that an eye-catching style might disguise this!

By all means use plotting/drawing packages to create graphs and figures, but if, for example, it is going to take you most of a week to learn to use a drawing package, you would be better advised to hand-draw your figures neatly and get on with something else.

The pages at the start of the report should follow this sequence:

- Title Page
- Abstract Page
- Contents List
- List of figures or diagrams (optional)
- Acknowledgements (optional)
- Chapter 1
- Rest of content...

4.5.1 Report Length

The absolute maximum length allowed for the entire report is 120 pages, where a page is defined as a side of an A4 sheet of paper. With double-sided printing this would mean a maximum of 60 physical sheets of A4 paper, as a single sheet holds two pages, back and front.

The main chapters, excluding the appendices, should not really be more than 45 pages in length, ideally around 40 pages (where, again, a page is defined as one

side of an A4 sheet of paper, so 45 pages is roughly 23 physical sheets of paper if your report were printed double-sided). The goal is to be *concise* and to the point. Examiners do *not* give credit for writing a longer report with every possible detail included. Good writing matters!

As there is some variation in the report content for different kinds of project, we don't specify an absolute limit for the number of pages the main chapters take up, or specify a word count. If you find yourself going over 45 pages then you most likely have too much content, but consult your internal supervisor for advice. If you go over 50 pages then it really is too long.

4.5.2 Title Page

The first or front page of your report is the title page. As well as the title of the project, the year of submission, the title page should also include the name(s) of your supervisor(s) and your degree programme (BSc, MEng, etc). Make sure you retain this page anonymous for marking (i.e. use your student ID instead of your name, and make sure the ID is easy to find!).

If you include the UCL logo on the title page, make sure you use the new logo as seen on most web pages and not the old logo with the large dome icon.

4.5.3 Title Page Disclaimer

On the title page, towards the bottom below the other items, you must also include a disclaimer in the words given below.

"This report is submitted as part requirement for the [BSc or MEng] Degree in [your degree title, 'Computer Science', etc.] at UCL. It is substantially the result of my own work except where explicitly indicated in the text." Then follow this with the words:

"The report may be freely copied and distributed provided the source is explicitly acknowledged."

or, if your project includes information that prevents it being more widely circulated, by

"The report will be distributed to the internal and external examiners, but thereafter may not be copied or distributed except with permission from the author."

You should consult the Projects Co-ordinator if you intend to use this statement.

You are reminded that the project is an individual project and that the work submitted should be substantially your own as stated in the disclaimer. Within your report you should identify clearly:

- Which work you have completed by yourself and represents your own individual contribution.

- Which work you have completed in conjunction with other people with whom you have collaborated (such as an external supervisor).
- Which work you have incorporated from other sources, such as from previous years' students, from external sources (e.g., third party algorithms, methods, publications, source code or code libraries), or from Research Projects with which you have been allied with during your project work (e.g., those of your Supervisor or of external companies).

4.5.4 Example Title Page

An example title page can be seen on the Moodle site.

Do make sure that the title page is clearly readable and that your name is easy to find and read. If you want to include the UCL logo then use the latest version as seen on many UCL web pages.

4.5.5 Abstract

On the page immediately following the title page you must have a short abstract giving a descriptive summary of your project. The abstract should be no more than half a page and typically consists of three short paragraphs:

- What the project is about, the primary aims and goals, and specific challenges.
- How you carried out the project and what work it involved.
- The results and achievements of the project.

The abstract needs to be read quickly by various people to get an overview of what your project is about, so make sure it doesn't get too long.

You will also need to upload one separate copy of the abstract when you submit your project report, just a single page. This copy should include your name, your supervisor's name, the project title and submission date.

4.5.6 Contents List Page

A contents list should follow the abstract. The contents list at the start of this document illustrates how it should be formatted.

You may also have a separate list of figures or diagrams, if that is suitable for your report.

4.5.7 References and the Reference List

It is very important that all sources of external information and ideas referred to in your report are properly referenced. Such information includes (this is not an exhaustive list):

- Published research papers, from publications such as journals and conference proceedings.
- Books, book chapters and specific sections or pages in books.
- Web sites, pages and other online material.
- Recorded seminars, talks and presentations.
- Source code, libraries and tools used by your project.

A complete reference consists of a flag or tag in the your text associated with the referenced material and an entry in the Reference List. Most importantly the Reference List entry must contain sufficient information for the reader to locate the referenced material if they want to read it.

There are a number of different styles for making references, several of which will be outlined below. You should check with your supervisor for their advice on which style to use for your project.

- Parenthesised or Harvard style. Parentheses are used to denote the reference, for example Bloggs (2002). If the reference forms a natural part of the text and directly refers to someone's work, the name or names are not put inside the parentheses. For example, "The algorithm developed by Smith & Jones (1997) is faster than ...", "The paper by Dent (2010) argues that ..." or "Simpson (1999, 2002, 2006) identifies ...".

  If the text refers to an idea or concept and just uses a reference to point to an example in passing, then the name and date are put inside the parentheses. For example, "It has been claimed that the algorithm is slow (Patel 2003) but ..." or "Earlier work (Smith 1993, Shah 1997) supports the claim that ...".

- Alpha style. A label is placed in square brackets, for example [Simp01] or [KNUTH87]. The label is formed from the surname of the first author, often shortened to three to five characters, followed by the last two digits of the year of publication.

- IEEE style numbered references in square brackets. The reference is a number in square brackets, for example [1], [3].

Whatever style you use, you should use it consistently and not use any other style.

The Reference List gives the full details of each reference, and appears immediately after the conclusions chapter as described in the report structure outline earlier. Each reference starts with the tag or label used in the main text and is followed by the reference details. For example:
Bloggs, J (2002), "The title of the paper", Journal of Computer Science, vol.

45, no. 2, pp. 207-226 or
[BLOG02] Bloggs, J (2002), "The title of the paper", Journal of Computer Science, vol. 45, no. 2, pp. 207-226 or
[3] Bloggs, J (2002), "The title of the paper", Journal of Computer Science, vol. 45, no. 2, pp. 207-226

The information included in the full reference should include volume and issue numbers, page numbers and any other information needed to locate the referenced text. References to information on the web should include the URL:

Shah, A (2011), "The title of the article", http://www.cs.ucl.ac.uk/info/a123.html

If you specify a URL you should have confidence that it will remain valid for some time (at least until after the examiners have read your report!). If not, then you may need to reference the site rather than the specific page, providing additional information, such as a search term, to help locate the information. However, if possible avoid providing references to unstable URLs.

Your program source code should also include references, for example to point to information about an algorithm you have implemented based on one described in a paper, or where a library being used comes from. If you have copied and pasted someone else's section of source code into yours, then it too must be referenced. For source code you should include the full reference, as it appears in the Reference List, in a comment in the code.

For further information about references, citations and avoiding plagiarism problems see the UCL web pages on 'What is a Citation?'. [3]

4.5.8 References in Source Code

A common question is 'Should source code include references?'. The basic answer is 'Yes' and this sub-section gives guidelines on what to do. Code and design reuse is normal practice in software development, and is certainly allowed in project work, but the principles of making clear where information and ideas come from still applies.

The approach to follow is:

- Use comments in the code and/or documentation to make it explicit where sections of code or algorithms that are not designed or written by yourself come from. You can use reference tags or more likely URL's. The same applies to making use of algorithms described elsewhere even if you coded a working version of the algorithm yourself.
- If you copy and paste sections of source code into your source code then the origin of the pasted code should be made clear. This requires some thought

[3] http://www.ucl.ac.uk/current-students/guidelines/plagiarism citation

as it is now very common to copy short sections of code found via a web search when you are discovering how to get something working. Anything more than 2–3 lines should be referenced, for smaller sections use your judgement. Does it gives a critical solution to a problem you needed to solve? If yes, then add a reference in a comment.

- Standard libraries that come with a programming language, for example the Java class libraries, do not need to be referenced.
- Frameworks, such as Android, iOS, Django, Rails, etc. should be referenced in one or two key places but not every time every feature is used.
- Other libraries and sections of code should be referenced as they are used.
- The same principle applies to design level structures, such as the use of design patterns and UML diagrams.
- It is not acceptable to use code, a library, or framework that automatically solves most or all of the problem being investigated (i.e., with only limited intellectual input from you). If it is the case that a ready made solution exists, then the project goals need to be reviewed. Projects can make use of whatever library code or frameworks are needed to support the design and implementation of the main focus of the project, serving as infrastructure that allows you to work on the actual interesting problems.

4.5.9 Other Format Requirements

As specified earlier in 4.5.1, but worth repeating here, the main body of your report (excluding appendices and code listing) should normally not be more than about 40-45 pages, and should include information about the most interesting aspects of your project. If you write a lot more than this, getting above 50 pages, your report is too long. The extra material risks being regarded as padding, and will not be viewed favourably by your examiners.

However, each project is unique and has its own natural length, and you need to make a careful judgement over when you have written everything that you think needs to be written. If in doubt, of course, ask for your supervisor's advice. Also note that examiners are assessing your ability to describe your work concisely and efficiently, and construct a good quality report. They will take note of your academic writing ability, and how well you are able to choose what to include and how to describe it.

You must use 1.5 line spacing and are strongly recommended to use 12 point type. On no account should you use a typeface less than 10 points – it is unreadable!

Pages should be numbered starting from page one on the first page after the abstract. Chapters should also be numbered and sections and sub-sections should

use hierarchical numbering as used in this document. You should use the standard A4 paper size throughout, don't include other pages sizes even for large diagrams.

4.5.10 Word Processing Tools

You are free to use any word processor or text processing tools you like. The main advice is to learn how to use your chosen tool effectively well before you start writing your report. Understanding styles, and getting page, chapter, section and sub-section numbering working automatically will save a lot of time, as will getting the contents list generated automatically.

We would encourage you to learn and use LaTeX to format your report. LaTeX is a document markup language and processing system widely used in academia and has many advantages for writing structured reports and scientific documents, such as ease of typesetting mathematical formulas and quick adaptation to publisher style files.

Keep frequent backups of what you write!

4.6 Style and Grammar

It is important to get your grammar correct and use punctuation correctly. Poor usage will give examiners a negative impression of your work.

Some common problems with punctuation include:
Full stops (.)
A full stop never has a space before it and is always followed by a space, except when followed by a closing bracket. A full stop used as a decimal point should not have spaces on either side of it.

Commas (,)
A comma never has a space before it and always has a space after it. The only exception is when it is used as a separator in large numbers, such as 5,789,567. Commas – like brackets and hyphens – separate out subordinate clauses or phrases that merely add information. Within a sentence you can tell if commas are being used correctly if you can lift out the words involved and have a sentence that still makes sense.

Colons and semicolons (: and ;)
These are 'almost end of sentence' markers that follow the same rules as a full stop. Semicolons are useful when a full stop feels too abrupt, but a comma would seem to link two succeeding sentences too strongly. However, many people never use them; if you are unsure about their use it is probably best to stick to full stops and commas.

Slash (/)
A forward slash (used as in 'his/hers') should not have a space on either side of it.

Hyphens (-)
When these are used as a pair within a sentence, in a similar way to a pair of brackets, then both hyphens have a space immediately before and after them (you should really use an en-dash (–) rather than a hyphen). However, when a single hyphen is part of a word (as in 'criss-cross') then there are no spaces to either side of it.

Use of brackets
There are several different kinds of brackets; parentheses (round brackets), braces and square brackets. Parentheses are most commonly used in normal writing, but braces and square brackets will be needed for mathematical expressions and program code. In normal text an opening bracket always has a space before it and never has a space after it. Conversely a closing bracket never has a space before it and always has one after it, unless followed by a punctuation mark such as a full stop or comma. Just as in programming languages, in English text brackets have to come in pairs (.....). If the items between a pair of brackets form a sentence, then there must be a full stop immediately preceding the closing bracket. Quotation marks, showing what some speaker actually said, behave just like brackets with regard to full stops.

Apostrophes (')
These are used in two ways, to indicate possession, as in 'John's book' and and in contracted forms, such as 'it'll' as a shortened form of 'it will'. For possessives the rule in relation to singular and plural nouns is:

If the noun is singular, there is an apostrophe followed by an 's', as in the 'The College's buildings...'. This rule is followed whether or not the singular noun ends in an 's', for example in 'The princess's clothes...'.

If the noun is plural, there is an apostrophe after the 's'; for example in writing about UCL then 'the colleges' buildings' is appropriate in referring to the buildings making up the campus. Note, however, that there are some words that denote possession that do not have an apostrophe: his, hers, its, ours, theirs, yours.

No-one ever writes 'her's' but the corresponding misuse of the apostrophe in 'it's' is probably one of the commonest of all grammatical errors. Using 'it's' always means 'it is', and is an example of a contraction. When using 'it's' read the sentence out loud pronouncing 'it's' as 'it is' to see if it makes sense. If not then use 'its', for example the possessive form of 'its' in this sentence 'The dog fetched its squeaky toy.' doesn't make sense using 'it is'.

Contractions such at 'they'll' for 'they will' and 'it's' for 'it is' reflect the rhythms of natural speech. Many people hear a kind of inner voice as they write and it is natural for most of us to write what we speak. Nevertheless, contractions are not recommended in a formal report. Try not to use them.

4.7 Use of First-Person Pronouns

The common first-person pronouns include 'I', 'me', 'my', 'our', 'us' and 'we'. Other than 'we', the use of these pronouns is largely discouraged in scientific writing as they are seen as undermining the scientific objectivity of the work. You will see 'we' used widely in academic papers and books, however, to refer to the authors or their work, as this is a widely accepted convention.

In general, you should avoid using the first-person pronouns in your report but be careful not to end up with rather stilted text written in the past tense. For example, you might want to write "I decided to use this algorithm to ..." and end up with "It was decided to use this algorithm to ...", where it would be better to write "The code uses this algorithm to ...". This tries to avoid the past tense to be more direct and readable. Use 'we' only if there are multiple people involved, not just to refer to yourself (the so-called royal 'we'!).

5 Project Submission

The ***provisional*** submission deadline is published on Moodle. This is an absolute deadline. Extensions are granted only for unavoidable circumstances that have had a substantial impact on your project, such as serious illness or bereavement. If you have a case for an extension due to such circumstances, talk to your supervisor as soon as possible. An Extenuating Circumstances application form, with good supporting documentation, must be submitted as soon as is practical, following the specified UCL procedure. An extension must be applied for and there is no guarantee that one will be granted.

If the circumstances warrant it, a short extension (up to a week or so) can be given by the extenuating circumstances panel, providing there will be no problems with completing marking on time. A longer extension will not allow enough time for marking to be completed before the exam board, which might end up delaying the award of your degree until the following year.

It is unusual for the submission deadline to change; if a change is necessary, this will be announced by email or on Moodle forum. You are required to check your UCL email regularly (at least once every day, more regularly as deadlines and exams approach); not reading your email is not an acceptable excuse for missing a deadline.

Also, keep in mind that there might be a significant load to the server, if you try to upload your report too close to the deadline. Plan ahead and do not let this become a problem. Failure to upload your report due to an overloaded server will not be considered as an excuse.

5.1 What to Submit

Your project is submitted via the upload links that will be provided on the Moodle site "COMP0029/COMP0138 Computer Science Undergraduate Final Year Individual Projects". You should be automatically registered on the Moodle site, if not contact the Undergraduate Administrator via the email address cs.undergraduate@ucl.ac.uk.

You should upload:

- A complete copy of your project report, fully formatted following the requirements described earlier. A single pdf file should be submitted. Make sure that the pdf displays properly on several different machines, in case of issues with fonts or images. Submit in pdf format only.
- One separate copy of your abstract (descriptive summary) of your project. This should be headed by the project title, your own name and that of your supervisor(s), and the submission date. This should be submitted via the online form available at the Moodle page.
- A single zip file containing a complete copy of your project source code and related files (e.g., build files or scripts). Don't submit binary or executable versions of your code. Do make sure that you use the zip format and don't password protect the zip file. Alternatively, you can submit a link to your GitHub repository, and option typically required by external projects.

6 Plagiarism

Detailed information about plagiarism can be found on the relevant web pages on the main UCL Web Site [4]. Make sure you read through all the information on these pages and understand the full implications.

In addition, please take note of this statement:

"You are reminded that all work submitted as part of the requirements for any examination or assessment at UCL must be expressed in your own words and incorporate your own ideas and judgements. Plagiarism – that is, the presentation of another person's thoughts or words as though they were your own – must be avoided, with particular care in course-work or essays written in your own time. Direct quotations from the published or unpublished work of others must always be clearly identified as such by being placed inside quotation marks, and a full reference to their source must be provided in the proper form. Remember that a series of short quotations from several different sources, if not clearly identified as such, constitutes plagiarism just as much as does a single unacknowledged long quotation from a single source. Equally, if you summarise another person's ideas or judgements, you must refer to that person in your text, and include the work referred to in your reference list. Failure to observe these rules may result in an allegation of cheating.

[4] http://www.ucl.ac.uk/current-students/guidelines/plagiarism

You should therefore consult your tutor or programme director if you are in any doubt about what is permissible"

As the project is such an important component of your degree, plagiarism in project work is taken very seriously, and when discovered has potentially very severe consequences for the culprit's degree and possibly for their entire future career. It is NOT worth it!

6.1 Get Referencing Correct

The most common form of plagiarism occurs when material created by others is not properly referenced in your project report. The underlying rule is:

The examiner reading your report must be left in no doubt whatsoever which are your words and ideas, and which are the words and ideas of others.

Any content in your report not tagged by reference is assumed to have been written or created by yourself. Further, and of critical importance, when you include a reference tag in your report (see 4.5.7) it must be clear what part of your text the tag applies to. In particular, if you are quoting someone else's words then those words need to be clearly identified as written by that person. Usually this is done by putting the words or sentence in double quotes, for example:
Bloggs (2003) states that "A sentence quoted from a paper that someone has written". or

"A short paragraph defining a concept relevant to your work, quoted from a research paper", Bloggs (2004).

Sometimes it is useful to put the quoted text in italics as well. The same need to reference also applies to diagrams and images included in your report, with the reference tag appearing in the figure or diagram label, for example: Figure 2 A diagram showing something, source Bloggs (2005)

What is absolutely not acceptable is to copy and paste some text, maybe up to several paragraphs, and just scatter in one or two reference tags with no quotes or other formatting. This fails to identify clearly which are your words and which are being referenced. Even worse is to intermingle some text of your own or to make edits to the copied text so it is not identical to the original but not your own words either.

Don't forget that the need to reference also applies to source code as well, or any other additional material you include with your report.

6.2 Assessment UCL

Your report will be submitted **via Assessment UCL (link available in Moodle)**, and this is the official submission point.

However, you can use Turnitin (also available via Moodle) prior to submission in order to get a detailed analysis of your work to find text that matches entries in its extensive database, giving you an indication of whether you might have an issue with plagiarised content.

Turnitin will always find some matches in your work due to its very rigorous processing algorithms. Many of these matches are likely to be false positives and it is important to recognise this. However, if Turnitin starts finding sentences and paragraphs that match, and you see larger areas of text being highlighted, then you have a problem.

Note that Turnitin imposes a twenty-four hour delay between generating each report so, even if you can submit several times, you need to leave some time between submitting 2 versions.

6.3 Why It Happens...

Plagiarism in projects, on those occasions when it has occurred, has more often been a desperate response to a looming deadline rather than a cynical strategy implemented from the very start of the work. This is largely the result of bad planning and time management. Things shouldn't have got so bad that plagiarism seems like the only possible option. Please look at the sections on project time planning. The most important aspect of this is that you see your supervisor regularly. Between thirty minutes to an hour per week, at a regular time, is recommended.

If you feel that you are slipping behind, that your project has had to be neglected in favour of coursework, that you have had family or other difficulties that have impacted on your project work, that you don't understand some of what your project entails, then whatever the problem is talk to your supervisor about it. Remember also that for more serious and/or wide-ranging problems, or if you feel you cannot talk to your supervisor, the Departmental Tutor is always available to see you.

7 Project Marking

The pass mark for the BSc project module is 40% and for the MEng module is 50%. MEng projects are at Master's level and are expected to demonstrate the greater depth and understanding obtained from having an extra year of study. Copies of the BSc and MEng marksheets completed by the examiners are available on the Moodle site. Reading these will help you understand the criteria better.

Note that the final overall BSc module mark includes the Change the World Challenge mark from the end of the second year. The challenge is worth 10% of the overall mark, and the project 90%. The BSc marksheet does not show the challenge mark and you need to get at least 40% on the project alone in order to

pass the module (i.e., you cannot turn a failed project into a module pass by adding in the challenge mark).

The stages of the marking process are as follows:

1. Marking by the First and Second Examiners
   All projects are initially marked independently by two members of UCL CS academic staff (often referred to as the First Examiner and the Second Examiner, or First Marker and Second Marker). The First Examiner is normally your internal supervisor. If you have had an external supervisor too, your internal supervisor may consult with them when deciding on a mark, but the external supervisor is not an examiner. The Second Examiner will be someone who has the relevant knowledge to mark your project.

   The following criteria are taken into account (this is not an exhaustive list):

   - Report organisation and structure.
   - Quality of writing and clarity of expression.
   - Suitable research and background reading.
   - Demonstration of a good understanding of the subject area.
   - Key problems identified and solved.
   - Reasonable and well-justified conclusions.
   - Critical analysis and appraisal of the work done and results produced.
   - Completeness (objectives achieved fully).
   - Overall quality of the final result.
   - Practicality of the design.
   - Requirements and objectives well understood and presented.
   - Appropriate use of data structures and algorithms.
   - Appropriate use of process and design methodology.
   - Appropriate use of tools, libraries, existing code and other artefacts.
   - Well structured and readable implementation.
   - Suitable and thorough evaluation and/or verification of the results achieved.
   - Appropriate system testing and/or verification.

   The First the First and Second Examiners discuss your project with the aim of coming to an agreed mark. It is usually the case that the two examiners can agree on a mark, but in those rare cases where this cannot be achieved the situation is resolved by appointing a Third Examiner and possibly seeking the advice the External Examiners (see below). If the First and

Second examiners do agree a mark but their initial marks differ by more than 15% then a Third Examiner will be asked to review the project and marking to confirm that the agreed mark is valid.

2. Project Moderation: Project marks are reviewed to confirm that there is a consistent standard of marking, that projects of similar quality get consistent marks, that borderline marks are reviewed, and that any issues from earlier in the marking process have been correctly addressed.

3. External Examiners: The External Examiners are experienced senior academics from computer science departments at other universities in the UK. They are part of the examining process for all subjects, but their role in relation to projects is to confirm that the assessment process is being run properly, that projects are of appropriate quality, and to confirm that all marks have been properly agreed if there were disagreements during earlier stages of the project marking cycle. They do not mark projects and do not look at all the projects, only a subset to satisfy themselves that everything has been carried out correctly. The actual mark awarded is the decision of the Exam Board, taking into account the External Examiner's comments, and is final.

8 Issues and who to contact

Issues concerning or affecting your project should always be shared with your supervisor in the first instance; they are best placed to deal with them or will know who to bring in to assist. Administration of projects is managed by the undergraduate teaching and learning team in the department (cs.undergraduate@ucl.ac.uk). All contacts including administrators and the project co-ordinator (the module leader) are listed on Moodle.