University College Dublin
An Coláiste Ollscoile, Baile Átha Cliath

MSC COMPUTER SCIENCE (CONVERSION) RESEARCH PRACTICUM COMP47360

June 2019

Guidelines

Contents

Research Practicum COMP47360

GUIDELINES

INTRODUCTION

The Research Practicum constitutes a vital component of your degree and as such requires careful consideration and effort. Every MSc Computer Science (Conversion) student is obliged to undertake a team project as part of the research practicum. The purpose of the project is to provide students with an opportunity to learn how to undertake a major project, taking it from problem specification through to a solution while working as part of a team.

The purpose of this document is to outline the nature of the project, to explain the procedures involved in undertaking such a project, and to provide guidelines on the important task of report writing. Please bear in mind that this document is intended only to describe the process in an indicative way, and in appropriate circumstances, certain details may be changed during the term.

The team-based project should be approached and completed in a spirit of collegiality. The purpose of the practicum is to learn, bring together your experience, and demonstrate the skills that you have acquired in the programme to date. The finished project can be used as a showcase for employers and to demonstrate how you have taken a project specification and delivered a solution.

TEAMS

The project for the research practicum is a team-based one. Students will be arranged into teams of approximately four members. The allocation of students to teams is based on students expressing an interest in the specific roles which have been identified as important to the success of the project along with a balancing of abilities. While individuals will take a lead role in a certain area of the project, all team members are expected to contribute to all aspects of the design and development of the project. A description of the roles and the role selection form can be found in Appendix I. Students are required to produce a team agreement in week one of the project. The team agreement should cover the ground rules for communication, participation, meetings, conflicts and decision making.

COMMUNICATION, PEOPLE AND WORKSPACE

We will communicate with you in various ways regarding project guidelines, deadlines, etc. Announcements may be made at meetings, emailed to you, or posted on the Virtual Learning Environment (VLE) page for COMP47360. Please be sure to keep yourself informed of what is going on by looking at the schedules on the VLE as they appear and paying attention to deadlines as extensions will not normally be available. The people involved in the module are listed below. The module coordinators should be the first point of contact for the module.

Project Mentors

Name	Contact
Gavin McArdle (Module Coordinator)	gavin.mcardle@ucd.ie
Colm Ryan (Module Coordinator)	colm.ryan@ucd.ie
Julie Berndsen	julie.berndsen@ucd.ie
Fatemah Golpayegani	fatemeh.golpayegani@ucd.ie
Pádraig Cunningham	padraig.cunningham@ucd.ie

Demonstrator Support

Name	Contact
Barbara De Kegel	barbara.de-kegel@ucdconnect.ie

Career & Skills Consultant

Name	Contact
Edel Caraway	edel.caraway@ucd.ie

Project Mentors

The project mentors play an important role in guiding and advising you as you work on your project. You should be sure to make good use of the mentors and to let them see that your approach to your project work is organised and competent. This can be demonstrated in the informal meetings, during presentations or at team meetings with mentors. In all cases, be professional: if you cannot make a meeting be sure to inform them in plenty of time. Do not come to the mentors with every little problem you have, try to work things out yourself as a team. This will allow you to have an informed discussion with mentors about the problem. At the same time, if you are having serious problems with your project and progress is halted, be sure to let the mentors know as soon as possible.

You will hopefully come to regard the mentors as friendly and supportive. If you have any problems during the term that are impinging on your work, let us know. In circumstances such as illness or bereavement, we are flexible in the allowances we can make.

Demonstrator Support

Dedicated demonstrator is available to support you through the project process. Technical help and guidance on the project is available. They cannot do the work for you but may be able to help you find a solution to specific issues you are having. Given that this is a research and development project, the need for demonstrator help is expected to be minimal. Details of the times which demonstors are available will be posted on the VLE for COMP47360.

Career & Skills Consultant

The career and skills consultant and module coordinators have put together a programme of career planning and speakers from industry. These sessions occur as per the schedule on the VLE for COMP47360. All students will attend these sessions and should contribute to discussions and activities to support their team and class.

Workspace

Classrooms B1.08 and B1.09 in the Computer Science Building are available for use during the summer term. Additionally, there are meeting locations available in the science building if you wish to use those.

Meetings

Each team member will deliver an individual presentation which represents the work of the team on a topic (See Section entitled Project Presentations). This is an opportunity to receive feedback from the project mentors directly but also classmates and other teams. The project mentors will call to the workspaces on Wednesdays (excl. Presentation Wednesdays) to receive updates from teams so it is expected that all team members will be present. Additionally, you can make an appointment with a project mentor to discuss your project or issues and challenges you are facing if you wish.

External Activities

The MSc Computer Science (Conversion) is a full-time programme. Any activity which interferes with your ability to be available during the normal hours of the university is not compatible with the programme/module. COMP47360 is a 30 credit module and so the workload is akin to a typical semester of taught modules.

ALLOCATION OF MARKS

Your project grade comprises a large proportion of your final award in Computer Science. The breakdown of the grade is as follows:

Component	Weighting
Group and Team Presentations	15%
Reflection Essay	15%
Project Report	60%
Team Work and Group Contribution	10%

The grades for your Project Report, Presentation and Reflection Essay are awarded by the mentors. Individual grades for teamwork are decided by team members combined with observations by mentors. Note that each component is graded – using letter grades and not a numeric score. A guide to grades can be found on the UCD website.

PROJECT PRESENTATIONS

Each team member will deliver an individual presentation which represents the work of the team on a topic. These topics and their timings are specified in the table below. Additionally, there is a final presentation to showcase the work produced by the team. The presentations take place before an audience that will normally include project mentors, your team and classmates.

Date	Presentation Title
12-June-2019	Initial Big Picture
26-June-2019	Detailed Requirements and Architecture
10-July-2019	Prototype and Data Analytics
31-July-2019	Pre-final Prototype
21-August-2019	Final Presentation

Presentations and possibly a demonstration (depending on the stage of the project) will typically be 8 minutes, with a further 4 minutes for questions. Please note that all students are obliged to attend the presentations which are a good opportunity for peer learning.

What makes for a good presentation? Firstly, to present your teamwork well, you must take ownership of it but you need input from your team. This means that you describe the work and the motivations. If you do not communicate your own interest in your project, no one else will be interested either. Explain why you did things a certain way, explain the process and be sure to motivate the problem for a general audience. Always build your presentation around a strong working example or use-case. Do not use overcomplicated examples that require too much prior understanding and too much explanation during your talk.

Your presentation should have a narrative structure. It should tell a story with a beginning, a middle and an end. A good presenter is a good story-teller. Set the scene well, motivate your problem, and give your audience enough information to anticipate your next steps throughout. Work on the ending, it is what people will remember most. Story-telling takes practice. It is not about how much you can fit onto your slides, but how you can help your audience to remember. This is definitely a skill worth honing for your future career.

Project Presentations Assessment

Marks will be awarded for your presentation by the project mentors based on the following criteria:

- **Technical Content and Organisation;**
- **Awareness of Audience and Delivery;**
- **Timing and Handling of Questions**;

A little note on answering questions: It is vital that you listen carefully to each question and be sure that you understand it before attempting to answer. Query the questioner if you are not sure what their question is. If you feel the questioning is tangential to your project work, do not be afraid to say something like "That's an interesting question, but my project is focused on the area of…". Do not be afraid of tough questions, or of giving tough answers. Tough questions allow you to shine, and to give the graders one more reason to elevate your project.

PROJECT REPORT & REFLECTIVE ESSAY

There are two written artefacts which you must write as part of your project. The Final Report which is a short thesis and a Reflective Essay is due at the end of the project. These should be produced in a professional manner. The Section entitled 'Preparing your Project Report' provides guidelines and advice on the writing of the final report. You will be provided with a LaTeX template that defines the formatting and style of the report. LaTeX saves you a lot of work and introduces a standard layout to all the reports. These templates must be used.

Reflective Essay

Each individual should produce a reflective essay. This should primarily focus on how you have developed as a computer scientist as a result of the project. You should discuss what you have learned and also what, if anything, you would do differently if you were to undertake a similar project. This discussion should cover three broad areas - technical skills, working as part of a team, and project management - but additional topics can be included. You should discuss anything you found surprising about the software development process and, if appropriate, how it differed from your experience in other areas. The maximum length of the Reflective Essay is 5 pages.

Final Report

The final report consists of two distinct components: A shared component written by the team (35% weighting of total grade) and individual component (25% weighting of total grade) written by each team member. The expected content of the report is outlined below. Apart from a cover page, project specification, abstract, table of contents and references, the maximum number of content pages is 25. These page limits mean that you do not need to "pad out" your report. At the same time, it does not make your task easy. Recall the famous quote: "I am sorry for the length of my letter, but I had not the time to write a short one."[1] The Section entitled 'Preparing your Project Report' provides some guidelines and advice on the writing of your final report.

Final Report Structure and Assessment

The project report is a very important part of your project and its preparation and presentation should be of extremely high quality. Remember that a large portion of the marks for your project are awarded for this report and it is your opportunity to describe the artefact and application that was created. In the assessment of a project and report, the high-level aspects described below may be taken into account. A more detailed breakdown per chapter is described below.

- Problem Comprehension: Does the student clearly grasp the problem that they are trying to solve?
- Technical Content and Overall Approach: Is the technical work of a high standard? Has the student used appropriate techniques in their work? Is the design compact and elegant, or is there clearly a better approach that could have been used?
- Justification: Does the student justify their approach? Where there are a number of options, they should be enumerated, analysed and the most appropriate one chosen.
- Critical: Good science is always self-critical, and this often does not come naturally. A good project report will, in its conclusions, try to be critical of the work done, highlight its weaknesses and failings. This is not a sign of a poor project; quite the contrary. A project report that praises the work done uncritically is lacking in an important respect.
- Report Quality: Is the report well-written and presented? There should be a clear structure to the overall document and a compelling "storyline" running through each chapter. Needless to say, bad spelling, poor punctuation and ill-formed sentences make a report hard to read and will result in a loss of marks.
- Contribution: Did the student make and demonstrate their contribution to the project? Was the extent of this contribution significant?
- The outcome of Project: Was the project (as specified) successful? Did the project go beyond the specification in an innovative way? E.g. if the software was produced, is the code correct? It is expected that demonstrations of your application have been made. An implementation that is hard-to-use or buggy will not score well.

Don't get depressed looking at this list! It is important that you understand what a good report is and show that you are striving towards this ideal. There is an excellent textbook by Christian Dawson on this topic that the School strongly recommends [1]. It is available on Amazon.

Any implementation produced should also be submitted. Not all implementations will be scrutinised, but the factors that may be taken into account in assessing them include:

[1] http://quoteinvestigator.com/2012/04/28/shorter-letter/

- Correctness: The software should perform as described in the final report. Minor deviations are not a problem, but any major aspect described in the report should also be present in the software.
- Robustness: The software should operate reliably and be not subject to frequent abnormal terminations.
- Code Quality: In your years in UCD you have encountered many design and implementation heuristics. It should be apparent in the code you have written that these have guided your work.

Needless to say, a violation under point (1), that is to say, a flagrant misstatement in your report of what your implementation actually does, is a very serious matter.

TEAMWORK

Each team member will be asked to assign a score for each team member's contribution. Comments can also be provided. Observations regarding contributions from the mentors during the practicum will also be used to determine the grade for this component.

SUBMISSIONS

Presentation Submission

Submission of all presentations is via Moodle. In order to facilitate the smooth running of the presentation session, all presentations must be uploaded to Moodle before 5 PM on the day prior to the relevant presentation.

Essay and Report Submission

Submission of all reports is via Moodle. You will be provided with details of this later in the term. Late submission will result in a sanction except where documented extenuating circumstances exist. Late receipt of a project component will normally be penalised by a loss of a full 5% for each day, or part thereof. This means that a report that is deserving of a final mark of 65% will actually be awarded only 60% if it is submitted a single day late (weekends included). Please understand from this that it is simply not acceptable to be late in the submission of any project component. A softcopy in Portable Document Format (PDF) must be submitted. No hard copy submission is required. The project report consists of chapters prepared by individuals and chapters prepared by the team. Each student should submit a report containing all chapters and the team produced chapters must be identical in all reports from a team. **The report must contain an accessible URL where the application can be viewed.**

Project Code Submission

The project involves the development of a piece of software or application. This must also be submitted, along with information on the development environment used and instructions on executing the application if applicable). The exact details regarding how to submit your code will be provided, but the deadline will be in advance of the final report and reflective essay deadline. This is to ensure you have time to complete the report and essay.

Plagiarism

Plagiarism is using other people's work in your project without acknowledging the fact that you are using it. It is most likely to occur in writing the Background chapter, as this will be based on material that you have read.

First of all, here is a tip for avoiding plagiarism. When you are actually writing a chapter, have none of the sources you are using open in front of you. Read the source material first, assimilate it, and write the description in your own words. In this way, the text you write will really be your own understanding of the area. Do provide references to your source material as well.

The School deals very firmly with instances of plagiarism, so please be very careful in this regard. It is also very naive to plagiarise in a field when your supervisor is likely to be very familiar with the source material themselves.

Partly why this is so important is that most students who plagiarise do so without realising what a serious issue it is, or what the consequences will be if it is discovered. Please be extremely wary of how you treat source material in your reports. There are a number of tools available on the web page for the projects to check your content yourself before submission. Ignorance of the rules is never a good defence. You should familiarize yourself with the UCD School of Computer Science Plagiarism Policy which is available on Moodle.

Backups

This is so important that it cannot be overemphasized. Keep several backups of your work in various locations, so in the worst, worst case you still have something to roll back to and are not left with nothing.

To assist you with the management of your project source code and reports, a git repository management system is maintained at https://csigit.ucd.ie. You may use this system creatively to backup, log and share weekly progress reports and project outcomes. It is strongly recommended that you keep your project directory synchronised with your git account, with weekly commits reflecting your progress as a minimum interaction level with the system. Your git repository should be suitably structured e.g. weekly reports, research documents, project implementation source code, outcomes/results. A snapshot of your git repositories for the past week will be kept, should you need to recover lost work. Interaction with a git could be a way to demonstrate your contribution to the project.

PREPARING YOUR PROJECT REPORT

Report writing is an important skill. No matter what field you are engaged in, you will almost certainly find it necessary to be able to write a clear report on your work. If you have a talent for technical writing, you will no doubt find it an easier task. However, it is a skill that can be acquired with practice and it is an essential part of your project work. Be sure to allow yourself enough time to write the report; the process generally takes at least two weeks.

Create a Report Structure

The first step is to produce a draft table of contents, showing how the entire report is to be structured into chapters, sections, and even subsections. Annotate each item with the purpose it is to serve in the overall report and its anticipated length in pages. When you have done this ask yourself the following questions:

- Is there a logical flow through my report? If it does not flow logically at this high-level stage, it certainly will not flow well in the end either. Move sections around until you feel there is a logical thread running through the document.
- Have I written about all the important issues? Pull yourself back from the report and think about the project in general. You should not write about everything you did. This is a report, not a diary but do not omit any vital sections either.
- Are the issues that I have written about important? You have probably written sections that should really be omitted. It is tough to cut out a section that you have laboured over, but dross in a report has a very negative effect on overall quality.

Once you have a sound overall structure, you can start writing sections knowing that they fit into an overall plan. You will know how much preparatory material will have preceded each section, and you will know to what extent it is expected to lay the groundwork for later sections. You may find that you have to change the structure later in the writing of the report. As with software, the later you change the design, the more work it entails.

Chapters and Content

The template you should use is detailed below, the required content for each chapter is provided and this will be used as the basis for assessment.

Title Page, Project Specification, Acknowledgments and Table of Contents

The title page should state at least the project title and your name. As with the Interim Report, the full project specification should be reproduced here. In your Acknowledgments section, give credit to all the people who helped you in your project. A Table of Contents is essential. The order of these is usually Title page, Project Specification, Abstract, Acknowledgments and Table of Contents.

The Abstract (Team)

The abstract should provide a short overview of your project that enables a reader to decide if your report is of interest to them or not. It should be concise, to-the-point and interesting. Avoid making it read like a verbose table of contents! Avoid references, jargon or acronyms, as the reader may not be familiar with them. An abstract usually contains a brief description of:

- The project and its context;

- How the project work was carried out;
- The major findings or results.

One paragraph is plenty. The main thing to remember is the principle that the abstract must be short, and a person reading it should be able to determine if they want to read more.

Team Chapters

Chapter 1: Introduction

- Description of the problem being addressed and aims of the project;
- Discussion of the wider context of the problem;
- Discussion of existing approaches and their limitations;
- Overview of the project;
- A description of the structure of the report, i.e., a roadmap for the reader.

Chapter 2: Description of Final Product

This chapter has the greatest weight for assessment and is the team's opportunity to describe the application. The following are the suggested topics.

- Description of all of the functionality of the application complete with screenshots;
- Explicit description of how the project problem/specification has been addressed;
- Explicit description of innovations beyond the project problem/specification.

Chapter 3: Development Approach

- A detailed account of how the team approached the project, i.e. the strategy you employed (e.g. AGILE). This should be at a high level, separate from design and implementation issues;
- The approach you adopted and used for communication, meetings and conflict resolution.

Chapter 4: Technical Approach

- Description of the architecture of the system;
- Description of the technical stack utilized;
- Discussion of alternative design and technology solutions;
- The justification for design and technology decisions;

Chapter 5: Testing and Evaluation

- Details of the testing strategy for accuracy, efficiency, usability, etc.;

- Results, discussion and critique of the evaluation results.

Individual Chapters

Chapter 6: Major Contributions

- Description of your role on the team with details of how you contributed to the technical, management;
- Details of the development aspects that you took sole responsibility for.

Chapter 7: Background Research

- Literature review which presents the research landscape within which your project is being conducted and considers approaches that have been adopted by other researchers;
- Description of the technologies and programming tools you are using.

Chapter 8: Critical Evaluation

- Conclusions: State the overall conclusions you have come to and what has been achieved. Be very critical in this section. Describe the weaknesses of your approach and avoid making unwarranted conclusions;
- Description of Future Work: Think carefully how your work might be extended or applied to another domain. There will probably be some obvious extensions. If you are able to propose some interesting ideas that are not immediately apparent, this demonstrates that you have a clear understanding of the field.

References

Use one consistent system for citing works in the body of your report. Several such systems are in common use in textbooks and in conference and journal papers. Ensure that any works you cite are listed in the references section, and vice versa. LaTeX will manage the referencing for you and be sure to make use of this facility. It may take more time in the beginning, but at the end of the write-up, it will certainly have saved you a lot of time.

In approximate decreasing order of quality, the best sources to cite are journal papers, international conference papers, national conference papers, books and web pages. Do not just supply a URL if there is an equivalent conference paper you could cite instead. Also, it strengthens your project if at least some of your references are recently published material.

Appendices

Material that you want to include in your report, but that is not directly relevant to the main thread of your report, can be put in an appendix. Possible examples include program/code listings, detailed test results, user guides etc. In most cases, appendices are not necessary and it is only in an exceptional case that it is useful to provide a code listing. Remember that material in the appendix counts towards report length, so do not exceed the page limits.

Order of Writing

The previous section dealt with the logical structure for your report. The order in which you write it all is another issue. There are no fixed rules here. Some people like to write notes throughout the project, so when it comes to writing the final report, they already have a lot of material prepared. This is a very valid idea, but avoid wasting time writing very polished notes during your project work. The notes/sections you write can be quite rough, and only in the final report do you bring them up to full report quality. The reason for this is that you may have to tailor them considerably to fit the context of the report, and this will mean that much of the polishing will have gone to waste.

Assuming you have created a report structure, a good way to continue is to write the introduction in draft form. The reason why you write this in draft form is that you are not yet sure what you are introducing! Only when the later chapters are completed can you return and finish the introduction.

Next, the Background chapter can be included in the final report. Again, you may find that when you write the core chapters later, some of the background work becomes irrelevant and can be removed. This may seem like wasted effort, but if it results in a tighter Background chapter, do it.

Next are the core chapters, followed by the design, testing and individual contribution chapters. When these are complete, you are in a position to write your Conclusion/Critical Evaluation chapter and to return to the Background chapter and bring it to completion. Finally, the team can write the abstract.

The next step depends on how much time you have left. Ideally, you will reach this point where you have a first full draft with at least a week to go. Proofread the report once yourself, and pass it on to other people on your team who can read (part of) it and give you any sort of feedback. Take a rest yourself, so you can return to it in a day or two and re-read the report with a fresh mind.

Note that at this late stage you can only make local improvements to the report. It is too late for major overhauls especially to the team chapters, so at this point the importance of creating a good overall structure becomes clear. If you have started with a good structure, you can aim to create an excellent finished product. However, if your initial structure was awkward, the final report will not read well no matter how you tweak it.

Other Advice

This section contains a number of guidelines that are worth bearing in mind when writing.

Continuity

You may not realise this, but a good academic paper or report, like any good novel you have read, tells a story. It is valuable to keep this in mind when you are writing your report. There should be a storyline running through your report and you should make it easy for the reader to hang on to this storyline:

- At the end of the introduction provide a short description of the layout of the remainder of the report.

- Start every chapter with a very brief recapitulation of the story so far, and an overview of what the chapter is going to add.
- Finish every chapter with a very brief summary of the material in that chapter, and state how it relates to what follows.

In the core chapters, you should take care to make absolutely clear the logical connection between the overall project design and the detailed problems you discuss. If the reader is mired in a detailed description of your solution to some intricate problem, they will be encouraged to persevere if you have clearly indicated its place in the overall project.

This continuity material may sound unnecessary and redundant, but it is useful to the reader. It may help for you to imagine that the reader is coming back to your report after a break of a few days: they will be greatly assisted by occasional reminders of what has already been said.

Presentation Issues

Focus on expressing your ideas clearly. Part of your report is, of course, its physical layout and use of diagrams. Try not to put too much time into this. Simple diagrams are fine and avoid the use of colour unless it really contributes something in particular. Do not bother with tricks like adjusting spacing or margins or fonts in order to make your report seem bigger or smaller. Do include screenshots of the applications and its features.

Textual Matters

Who do you have in mind as you write your report? A good model to use is that you are writing to an educated computer scientist who is not directly involved in the field of your project. Your report is intended to be a technical document, so follow the style you see in the best scientific literature that you read.

Keep your language clear and avoid colloquialisms and abbreviations. Avoid writing in overly “academic” tones. In good academic papers, you will find a simple, clear style that is easy-to-read and not overwrought. The previous sentence could also be written as: “You will find, in academic papers of good quality, a style that is at once both clear and indeed easy-to-read, without being in any sense overwrought.” This style is only suitable if you are making a crowning point that you wish to emphasize heavily.

Do not overly use the word “I”. If at some point you really wish to express a personal opinion, use a phrase like “it is the opinion of this author that. . . ”. Avoid over-using “we” as well, but don’t use the passive voice all the time, or your report will be unreadable.

Avoid overuse of italicisation, underlining, bolding, or other devices for emphasis. Underlining is best avoided, as this was originally a device for informing a typesetter to use italics and not a form of highlighting in its own right at all.

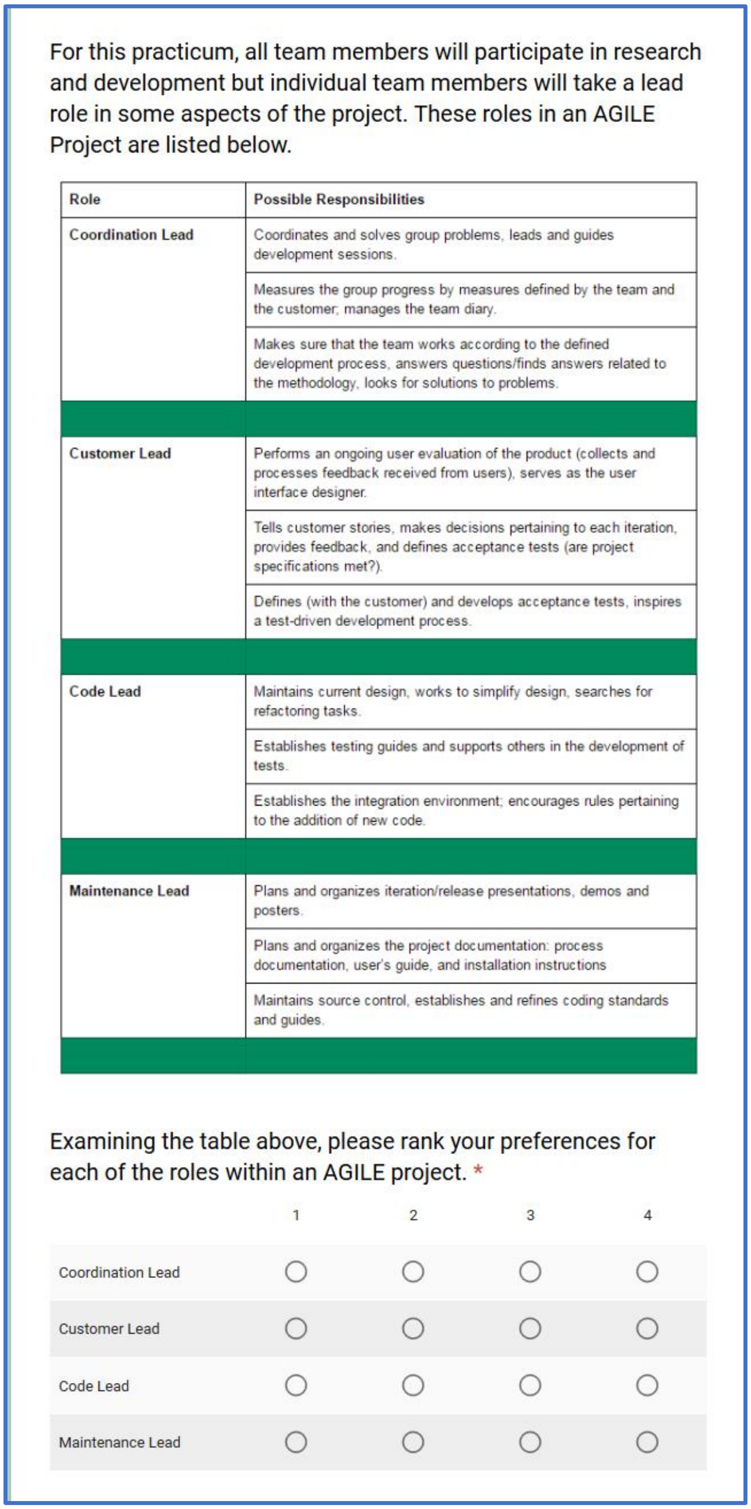
Do not place large blocks of code in your report. If you are considering putting program code in your report at all, ask yourself first if an algorithm written in pseudocode is more appropriate. Any code you do present should earn its place and should be impeccable in construction and layout. Use a suitable font for code, such as courier, and stick to this consistently.

Pay attention to spelling, punctuation and sentence structure. Poor spelling can be very intrusive and is unforgivable given that your word-processing software surely provides a spelling checker. Poor punctuation can destroy the meaning of a sentence. If you are not sure how to use punctuation, use fewer commas rather than more. Long sentences that are difficult to write are usually also difficult to read and may turn out in fact not to be sentences at all. If a sentence feels unwieldy, split it in two.

REFERENCES

[1] Christian Dawson, The Essence of Computing Projects – A Student's Guide, Pearson Education, 2000

APPENDIX I



For this practicum, all team members will participate in research and development but individual team members will take a lead role in some aspects of the project. These roles in an AGILE Project are listed below.

Role	Possible Responsibilities
Coordination Lead	Coordinates and solves group problems, leads and guides development sessions.
	Measures the group progress by measures defined by the team and the customer; manages the team diary.
	Makes sure that the team works according to the defined development process, answers questions/finds answers related to the methodology, looks for solutions to problems.
Customer Lead	Performs an ongoing user evaluation of the product (collects and processes feedback received from users), serves as the user interface designer.
	Tells customer stories, makes decisions pertaining to each iteration, provides feedback, and defines acceptance tests (are project specifications met?).
	Defines (with the customer) and develops acceptance tests, inspires a test-driven development process.
Code Lead	Maintains current design, works to simplify design, searches for refactoring tasks.
	Establishes testing guides and supports others in the development of tests.
	Establishes the integration environment; encourages rules pertaining to the addition of new code.
Maintenance Lead	Plans and organizes iteration/release presentations, demos and posters.
	Plans and organizes the project documentation: process documentation, user's guide, and installation instructions
	Maintains source control, establishes and refines coding standards and guides.

Examining the table above, please rank your preferences for each of the roles within an AGILE project. *

	1	2	3	4
Coordination Lead	○	○	○	○
Customer Lead	○	○	○	○
Code Lead	○	○	○	○
Maintenance Lead	○	○	○	○

Figure 1. Team Role Questionnaire