Ollscoil
Mhá Nuad
Ollscoil na hÉireann
Má Nuad

Department of Computer Science

Maynooth University

Student Project Handbook

2017 - 2018

BSc Computer Science and Software Engineering
BSc Multimedia, Mobile and Web Development
BSc Computational Thinking
BSc Science
Higher Diploma in Science (Computer Science)
MSc in Computer Science (Applied)

Contents

Introduction

The project is one of the most important aspects of your degree. It provides you with an opportunity to apply the material learned in different modules, and integrate material learned at different stages of the course. There may also be a need for additional, domain-specific knowledge for your project, which will necessitate additional study. **The onus is on you, the student, to contact your supervisor on a regular basis. A schedule should be agreed upon, with your supervisor, at the commencement of your project to agree on the frequency of meetings needed.**

1 OVERVIEW

You will execute a significant project in an area of relevance to Computer Science and/or Software Engineering. Every project is unique, but this will normally involve four activities:

1. Planning your project.
2. Executing your project.
3. Documenting your work.
4. Critically assessing your project.

You are expected to produce something significant in your project: a software implementation, the evaluation of your solution to a research problem, or an evaluation of existing solutions.

2 LEARNING OUTCOMES

On successful completion of the module, you should be able to apply your knowledge and understanding of Computer Science to analysing problems, creating and evaluating solutions, and critically assessing your own work. You should also be able to prepare project plans, give presentations, and write reports.

3 LEARNING METHODS

The final year project is a self-learning exercise, under the guidance of your supervisor. The following six learning elements form the basis of your project: knowledge, understanding, applying, analysing, evaluating, and creating.

During your project, you should use these as follows:

- Develop a **knowledge** and **understanding** of the technical domain on which your project is based
- **Apply** your knowledge and understanding in executing the project
- **Analyse** the problem, your solution, your results, your approach
- **Create,** Design and Implement a solution
- **Evaluate** your work

4 RESPONSIBILITY

It is important to note that you are responsible for your project. Your supervisor will provide you with guidance and feedback, but you must put in the effort to achieve a successful project. So work hard on your project, while balancing your time and effort with your taught modules. You should schedule regular meetings with your supervisor.

5 ASSESSMENT

Your principle assessment is on your project report. In addition, you will be assessed on your project presentation and demonstration.

The deadlines for this year are detailed below. You will give a project presentation and demonstration after submitting the report. The presentation is usually in your supervisor's office. The demonstration is usually in the lab. You are required to submit an interim report half-way through highlighting your progress to date.

Completed final year project reports must be uploaded to Moodle by the due date. You must submit two files:

1. The Report, as a PDF file

   The file name must be: *your_student_number*.pdf (where *your_student_number* is your student number)

   Make sure the report can be opened and read on a standard departmental PC (windows or Linux, in one of the labs) before submitting.

2. The Attachments and Supporting Material, including your source code, as a single ZIP file (named: *your_student_number*.zip):
   a. This file may only contain two folders at the top level, named "*attachments*"

and “*supporting*”.

b. Every file in *attachments* must be in PDF format, with no sub-folders. Appendices should be included here: each appendix must be in a separate file, named appendix_<*N*>.pdf (where <*N*> is the appendix letter).

c. You may use sub-folders in the *supporting* folder, and include any file type as required. In particular, include your full source code here. You must include a file README.TXT describing the folder contents in detail, how to compile sources, how to run any binary programs included, etc. Note that if you use non-standard file types, or require special software tools to use these files, then these files may not be assessed. Ideally, someone else should be able to run your software solution based on the files and instructions you include here.

You are encouraged to use version control systems for their project, and you can use the project git repository to submit your source code (see Moodle for more details).

Important Dates for 2017/18

Undergraduate and HDipCS projects

Date	Event
2nd October 2017	Staff will have assigned students to projects – unassigned students will be paired with supervisors
4th December 2017	Interim reports are due
18th December 2017	Feedback on interim reports will be emailed to students
19th March 2018	Projects are due before 11.59PM
2nd April – 13th April 2018	Demos of projects to take place

MSc in CS (Applied) projects

Date	Event
2nd October 2017	Staff will have assigned students to projects – unassigned students will be paired with supervisors
13th November 2017	Interim reports are due
20th November 2017	Feedback on interim reports will be emailed to students
22nd December 2017	Projects are due before 11.59PM
17th – 26th January 2018	Demos of projects to take place

Project Planning

One of the most important things you will learn when doing your project is the need to manage your time. Final Year Projects are completely different from smaller projects you may have undertaken. They require a considerable amount of time:

- Single Honours Science students should spend 240 hours (accounting for 25% of final year marks)
- CSSE students should spend 240 hours (accounting for 25% of final year marks)
- Double Honours Science students and Computational Thinking students should spend 80 hours, normally in one semester (accounting for 8.3% of final year marks)

Begin your project early, work consistently at it, and track your progress. A project plan helps you to identify all the things you need to do, understand their relationships, and consider the amount of time each will take. It may be useful to keep a daily log of your work.

TASKS

In order to plan your project, you should identify the major tasks you need to complete.

For each task, you should identify the following:

- Title
- Brief description of the work to be done
- Identify what the output or deliverable will be
- Estimate the start and end dates
- Identify the dependencies between tasks
- Identify the key risks to completing the task on time, and what “contingency” action you will take if you are unable to do this

Report Structure

- **Length**
  - Your report, including the title page, should not exceed 20 pages (excluding references and appendices). You must use A4 page size, 11-point font for your main text, with a line spacing of 1.15, and margins of 25mm.
  - Your appendices are not included in this limit – they make take as much space as is required, but you cannot expect markers to read them in the same detail as they do the main body of the report. They must be included in separate files (as detailed above under Assessment).

- **Required Table of Contents:**

Your report must contain a Table of Contents (use the following sections as a guideline):

- Abstract
- Introduction
- Technical Background
- The Problem
- The Solution
- Evaluation
- Conclusions
- References
- Appendices

Report Contents

- **Title Page**
  - This must be as shown in **Report Title Page**
  - You must include your own name, project title and supervisor name on here

- **Abstract** [1]
  - The abstract must be an accurate reflection of what is in your report.
  - Your abstract must be self-contained, without abbreviations, footnotes, or references. It should be a microcosm of the full report.
  - Your abstract **must be 150- 250 words** written as one paragraph, and should not contain displayed mathematical equations or tabular material.
  - Ensure that your abstract reads well and is grammatically correct.
  - The abstract must cover: motivation, the problem statement, the approach, your results, and your conclusions.

- **Introduction**
  - Topic: Introduction to the topic addressed in the project
  - Motivation: Why would one care about the problem and the results? Cite appropriate references in this section. Explain the high-level, abstract problem that your project addresses. Explain what you are trying to achieve in a way that leads naturally into the next section.
  - Problem statement: Describe the technical problem needed to be solved in your project. Note that most projects solve both a more abstract, high-level problem and a specific, technical problem: your problem statement is the detailed technical problem (your motivation should cover the more abstract high-level problem).
  - Approach: summarise how you addressed solving the problem. Provide an overview of how you analysed the problem, how you designed a solution, and how you evaluated your solution. (e.g. use of models, simulation, prototypes, real-world experiments, cases-studies, etc.). What important variables did you control, ignore, or measure in your evaluation [2].

[1] 1 Based on https://www.ieee.org/publications_standards/publications/abstract_description.pdf
[2] 2 Read Fenton and Pfleeger, *Software Metrics*, Chapters 2 & 4

- Metrics: describe how you are going to evaluate your work.

- Project: list, and briefly describe your significant achievements in the project (probably 3-5 of these in a typical project). If you have come up with any contributions to the state-of-the-art, make sure to include them here.

- **Technical Background (all material to be cited correctly)**
  - The purpose of this chapter is to show your depth and breadth of reading and understanding of the problem domain
  - Include two sections:
    - Topic material (research material, if used, from published journals and conference proceedings; less academic publications, if required by the project, from other sources) – for example, what other work researchers have done already in this area, what results they have produced, what work has been done in related areas, what software already exists to solve this or similar problems, etc.
    - Technical material (from any source: including books, websites) – for example, how to write a web server, how to use specific Java features, how to use Ajax, how to use UML to validate your design, etc.
  - Note that material relating to the motivation or non-technical background should **NOT** go here, but rather in the introduction.

- **The Problem (or User Requirements)**
  - The purpose of this chapter is to clearly explain the technical problem and/or identify the user requirements
  - Provide any model(s) of the problem (e.g. equations, ERD's, UML Use Cases & Scenarios, Activity Diagrams, etc.)
  - Provide any analysis of the problem, leading to a greater understanding
  - There should be no decisions made in this chapter

- **The Solution (Design & Implementation)**
  - The purpose of this chapter is to clearly identify, discuss, and justify the decisions you make
  - Depending on your type of project, you may not need to include all of these
    - Analytical Work: E.g. Equations, etc. that describe your solution

- Architectural Level: E.g. Implementation Diagrams
- High-Level Design: E.g. Packages, Class Diagrams, etc.
- Low-Level Design: E.g. Method specifications, Algorithms, etc.

- Implementation: discuss anything interesting here, put full source code in an appendix or attachment

- **Evaluation**
  - **Solution Verification**
    - E.g. use your equations to verify the correctness of your solution
  - **Software Design Verification:** how did you show that your design worked properly?
    - Using a model of your solution
    - E.g. use UML interaction diagrams to verify each scenario
  - **Software Verification:** how did you demonstrate your software worked properly?
    - If you have not tested your software, then you cannot rely on your results. Clearly describe:
      1. Your test approach (i.e. unit testing, sub-system testing, system testing)
      2. Your tests (e.g. scenarios, test cases, test data, etc.)
      3. Your test results
      4. An interpretation of the results

  - **Validation/Measurements:** how did you measure how well your solution solved the problem
    - Results
    - Explanation of Results
    - Analysis of Results
    - Comparison with previous solutions (if relevant)

- **Conclusions**
  - Identify and discuss the implications of your work.
  - If you made a contribution to the state-of-the-art, clearly identify it here.
  - Discuss whether your results are general, potentially generalisable, or specific to a particular case.
  - Identify threats to the validity of your results (e.g. limitations, risks introduced by your approach, etc.)
  - Discuss your project approach
  - Discuss future work, based on what you have done (and not done)

- **References**
  - You must include a list of all cited works here. Use standard guidelines for these (e.g. IEEE guidelines [3], APA guidelines) as advised by your supervisor.
  - The best references are peer-reviewed publications and books.

[3] http://www.ieee.org/documents/ieeecitationref.pdf

    - Note: you must NEVER give just a URL for a reference. At the least it must include an author (even if the company's name), a publisher (may also be the company's name), a title, and a date (if none available, use the date last accessed). If you have a URL for the document, include [<URL>, accessed on: <date last accessed>] in square brackets at the end of the reference.
    - Non-reviewed material (such as blogs, Wikipedia, etc.) does not make a suitable reference, except in the case of technical material have you have been able to verify by checking their implementation (you should state how you did this).
    - Material from other publications (such as journals, newspapers, magazines, official reports, etc.) may be used sparingly, especially if important for motivating your work.

- **Appendices**

    - Include here all extra material, e.g. your source code, project management (optional) including: the task list, Gantt Chart diagrams (or equivalent), discussion of any significant deviations from plan, and how you managed them, discussion of what you would do differently if you repeated the project.

Project Presentation

Your project presentation at the conclusion of your project should consist of five slides. Discuss these with your supervisor in advance. Suggested titles and content are:

1. **Introduction**
    - Give a very brief description of your project goals.
    - Describe the motivation for your project.

2. **Background**
    - Describe the background to your work.
    - Identify previous work your based your work on.
    - Show you understand the “context” of your work.

3. **The Problem**
    - Describe the problem.
    - Include technical details.

4. **The Solution**
    - Describe the solution(s) you developed and/or evaluated.
    - Include technical details.
    - Identify any threats to the validity of the solution.

5. **Evaluation & Conclusions**
    - Summarise how you evaluated the solution.
    - Summarise the results of your evaluation of the solution.
    - Evaluate how well you executed your project - for example:
        - how well you understood new knowledge
        - how well you learned to use new tools
        - how well you evaluated the solution

Marking Scheme

Projects are marked using the following criteria and default weightings.

Marking Criteria	Weighting	Mark (out of 100%)
1.Abstract (style and content) [4]	5%	
2. Motivation	10%	
3. Technical Background/Related Work	10%	
4. The Problem	10%	
5. The Solution & Its Evaluation	45%	
6. Conclusions	10%	
7. Final Project Presentation & Demonstration	10%	

Overall Assessment	Grade
A=Excellent, B=Very Good, C=Good, D=Acceptable, E=Not Up To The Required Standard)	

Markers justification of the overall assessment

[4] See next page: based on “IEEE Abstract Description and Specifications”, 2013

Abstract Style Criteria

Style

- The abstract should be a microcosm of the full report.
- The abstract must be self-contained, without abbreviations, footnotes, or references.
- The abstract must be between 150-250 words.
- The abstract must be written as one paragraph, and should not contain displayed mathematical equations or tabular material.
- The abstract should include three or four different keywords or phrases, as this will help readers to find it.
- Ensure that your abstract reads well and is grammatically correct.

Marking Criteria for a research-oriented project.

1. **Abstract/content**: how well does it state the *primary objective,* offer any tested hypotheses, describe the research design and methodology, describe the *methods and procedures* employed, present the *main outcomes and results*, draw the *conclusions,* and consider any implications for further research or application/practice.
2. **Motivation**: how well both the general topic, and the specific technical issue being addressed, are motivated.
3. **Related Work**: whether the material is relevant, comprehensive, and correct.
4. **The Problem**: how well the research question is presented, how well the problem is described, and how well the problem is analysed.
5. **The Solution**: how well is the problem solved: the abstract solution, the implementation, and the research methodology?
6. **Evaluation**: how well evaluated are, as appropriate to the project:
    a. the solution (for example, through collecting and analysing results from the implementation),
    b. any software developed to evaluate the solution (to ensure that the results are reliable),
    c. and the methodology (i.e. is there a critical analysis of the results, discussing threats to their validity).
7. **Conclusions**: how well summarised are the results, how well founded are the conclusions (i.e. based on the results), how well is the contribution identified, and is future work discussed (based on the results presented and the conclusions).

Marking Criteria for a development-oriented project.

1. **Abstract/content**: how well does it state the motivation, the primary objective, the approach taken, the results, the conclusions, and implications for further work.
2. **Motivation**: how well both the general topic, and the specific software being developed, are motivated.
3. **Technical Background/Related Work**: how well is the technical background, and any relevant related work, described?
4. **The Problem**: how well is it (a) described and (b) analysed. This includes any requirements analysis.
5. **The Solution**: how well is the problem solved (both the end result itself). This includes any software architecture, software design, and software implementation details.
6. **Evaluation**: covering, as appropriate for the project, how well you evaluated your solution, the software architecture, the software design, the software implementation, the final working system, and a critical analysis of technical approach taken.
7. **Conclusions**: summary of results, implications for real-world use, potential future work.

Report Title Page

Project Title – Report Page Template
Optional Subtitle

Put Your Full Name Here

Final Year Project – **Put the Year Here**
B.Sc. Computer Science and Software Engineering
(Amend degree where not applicable)

Department of Computer Science
Maynooth University
Maynooth, Co. Kildare
Ireland

A thesis submitted in partial fulfilment of the requirements for the B.Sc. Computer Science and Software Engineering. **(Amend where not applicable)**

Supervisor: Put your supervisor's name here

Special Note on Plagiarism

Reference the university policy at :

http://examinations.nuim.ie/documents/plagiarism.pdf

Presenting the work of others as your own is not acceptable.

A reader must be able to clearly distinguish between your own work and the work of others.

If you use the work of others, either as a direct copy, or summarising or using their ideas, you must (a) cite their work at every place where you make use of it, and (b) include a full reference to their work.