IRP2022: GUIDELINES

IMPORTANT DATES

IRP start	Monday 6 June 2022
Writing Course	Monday 13 June 2022, 10:00-11:30 BST, MS Teams
Project Plan deadline	Friday 1 July 2022, 12:00 BST
Presentation Course	early August (TBC), MS Teams
IRP end	Friday 2 September 2022
Final Report deadline	Friday 2 September 2022, 12:00 BST
Presentations	14, 15, and 16 September 2022, 09:00-18:00 BST

CONTACT

Should you require any information or assistance, please do not hesitate to contact the IRP Team:

- **Email**: ese-msc-irp-EA@groups.imperial.ac.uk
- **MS Teams**: students have access to the **Independent Research Project** Team

Please remember that you can always seek advice from your Personal Tutor.

OBJECTIVES

On successfully completing the Independent Research Project (IRP), students should have:

- developed a scientific/numeric code from scratch, extended the capabilities of an existing code, or built a model to analyse and interpret a substantial dataset;
- contributed to an active research area;
- developed critical analysis techniques and the ability to creatively solve challenging problems;
- defended their research output under critical questioning;
- developed effective communication, writing, and presentation skills;
- demonstrated effective time management.

DESCRIPTION

This module involves each student in the independent analysis of a technical problem for which they will devise, implement, test, modify, validate, and document a practical computational solution. A range of such problems has been provided to the students who either opted to solve one of those, proposed to solve a problem of their own devising, or decided to undertake a computational project as a part of external industry placement. Self-devised and external projects required the agreement and approval of the IRP Team.

Professional computational projects are seldomly performed in isolation. Consequently, students may use whatever resources they can discover to assist them in their project but must fully declare their use (see **Plagiarism** section).

Students should submit a quantitative analysis of the implemented code or the analysed problem. This may include but is not limited to appropriate testing, validation, or optimisation and performance analysis. In addition, students may submit a short appendix in the form of a user and developer guide to their software, with the technical background required to understand the initial problem and the methods used to solve that problem.

DELIVERABLES

There are three IRP deliverables:

1. **Project Plan (5%)**

   - Written Report (Introduction, Literature Review, Problem Description and Objectives, Progress to Date and Future Plan, Bibliography) (70%)
   - Progress to Date (10%)
   - Independence and initiative (20%)
2. **Final Report and Code (75%)**
   - Abstract / Introduction / Problem Statement / Literature Review (20%)
   - Methodology / Software Development Life Cycle / Code metadata (30%)
   - Code / Results (30%)
   - Discussion / Conclusions / Bibliography (20%)
3. **Oral presentation (20%)**

ESSENTIAL GUIDELINES

Title page. You must provide information that identifies you as the author of the code and report. The first page of the project plan and final report includes the following information: university, department, MSc course, module name, first and last name, GitHub username, email, date of submission, project title, and supervisors. If you undertook your IRP as an intern in a company, please also add the company's name and address. Please also add a link to your GitHub repository (if applicable) on the first page. Example Word and LaTeX templates are provided.

Formatting. Written reports should be written in Word or LaTeX, single-spaced, with a font equivalent to 11pt sans-serif font (e.g. Arial) as recommended by the British Dyslexia Association. You must submit each report as a **single PDF file**.

Word limit Each written report has a word limit as well as the maximum number of figures and tables that you must not exceed:

Project Plan	maximum 1,500 words, 5 figures/tables, 8 MB
Final report	maximum 5,000 words, 10 figures/tables, 8 MB

Shorter reports will not be penalised for their length. A small tolerance will be allowed to accommodate the differences in word counting programs. The following will not count towards the word limit: text in figures/tables, captions, algorithms, references, title page, appendices, and acknowledgements. Code snippets, algorithms and pseudo-code embedded in the text will count towards the word limit. However, code snippets, algorithms and pseudo-code part of/presented as figures will not count towards the word limit.

Report structure. The report structure described below is a guideline, and you can, of course, deviate from it.

Submission. The submission deadlines are in **Important dates** section. Each report must be submitted to your GitHub repository to directory `reports/` as a single PDF file, and it must be named as explained later (`COLLEGEUSERNAME-REPORTTYPE.pdf`). The size of a written report PDF file **must not exceed 8 MB**. In addition to GitHub, written reports must be submitted to **Turnitin** for plagiarism checks - details will be provided prior to the submission.

Backups and repository usage. Although infrequent, hardware failures occur, as does human error, rather more often. You are strongly recommended to keep multiple independent backups of your work under a proper versioning system and take full advantage of cloud storage by regularly pushing to your GitHub repository. In particular, mitigating circumstance claims regarding multiple weeks of lost work due to the lack of backup are unlikely to be accepted.

PROJECT PLAN

The project plan is an important part of your project, and it is **not** optional. The project plan will be evaluated, and it will count for 5% of your total mark. Writing a project plan is an essential first step in an independent research project. Your project plan should summarise the problem you are working on and your strategy to address it. You can also add tasks and milestones if appropriate. You should include relevant referencing throughout your document.

- Project plan must follow the rules defined in **Essential Guidelines**.

- Project plan should include the following content: Abstract, Introduction / Literature Review, Problem Description and Objectives, Progress to Date and Future Plan, Bibliography.
- Name project plan PDF file as `COLLEGEUSERNAME-project-plan.pdf` (all lowercase) and push it to your GitHub repository in `reports/` directory (e.g. your plan should be `reports/xyz123-project-plan.pdf`). Submit the same file to Turnitin.

FINAL REPORT

The final report is a document structured as a research paper. Prepare your report following the structure of a paper to be submitted to an academic journal. In the Writing course, you will review the process of writing this type of report.

- Final report must follow the rules defined in **Essential Guidelines**.
- Final report should include the following content: Abstract, Introduction (Literature Review, Problem Statement and Objectives), Methodology, Code metadata, Results, Discussion, Conclusions, and Bibliography. You should also explain your code implementation, validation, testing, and performance.
- If you believe it is necessary to include more information, but the word limit limits you, you can add an Appendix with additional text and images.
- The report will be evaluated by at least two markers, at least one of which will probably be one of your supervisors.
- Name your final report PDF file as `COLLEGEUSERNAME-final-report.pdf` (all lowercase) and push it to your GitHub repository in `reports/` directory (e.g. your report should be `reports/xyz123-final-report.pdf`). Submit the same file to Turnitin.

Abstract. Start with a roughly 200-word abstract.

Introduction. Your introduction should include a summary of the problem that your code/or computational analysis addresses. The introduction should also have a brief and relevant literature review. Describe your objectives and/or hypotheses, and outline the tasks completed during the independent research project. Describe state-of-the-art solutions to the problem, including commercial and academic approaches. Clearly state how your independent research project goes beyond the state-of-the-art and what original work you have done.

Methodology. Describe the technical back-end of your solution. Describe if it was developed as a standalone code or if it is an extension of a pre-existing code. In the latter case, briefly describe the ecosystem in which you developed your solution. List and describe what development and operation tools have used, outline the development methodologies used, and explain why. Add an architectural design diagram of your solution if relevant. Describe your design rationale and implementation strategy, including the description of main data structures, routines, any parallelisation paradigm (shared vs distributed), and their respective verification and validation routines. Add algorithms, pseudo-code, and, if required, code snippets of the most important functionality you implemented. Emphasise the novelty and creativity of your work. This section may be better suited for some types of projects and less suitable for others.

Code metadata. You should succinctly describe the technical platform of your implementation, including any compilation requirements or dependencies - such as programming languages and libraries (open source or proprietary), as well as a description of both the implementation and deployment platforms (operating system, basic hardware requirements, if any). Add the current code version number of your software/computational solution, a link to the code (such as a link to the repository), and links to any developer documentation, user and/or developer support documents. This section can be a table or a short paragraph.

Results. Describe the simulation features and capabilities and the test/study cases investigated during your IRP. Describe the unit or system tests and, where applicable, integration tests that you ran to determine if your program accurately solves the stated problem. Present validation and/or verification results. Describe the implementation of your design, the functionality of your code, and how it may affect the accuracy, efficiency, and scaling of your computational solution. Present the results of your computational analysis and quantify your results.

Discussion & Conclusions. Add a brief discussion section to the report. What were the most difficult tasks to resolve in completing the implementation? What are the strengths and limitations of your solution? What are the next steps? What worked and what did not work? Be formal in your analysis, as if you were writing a journal paper, quantify your statements and base your conclusions on the findings of your study. If applicable, briefly describe any planned future work related to your project.

Bibliography. List all resources (e.g. books, articles, manuals) that you consulted to understand and develop the project; these should be included and properly cited in the text. You should include relevant referencing throughout your document. You should make sure to reference sources of value; for example, papers sourced from indexed journals from sites such as ScienceDirect and Scopus.

Appendices. Appendices containing supplementary information can be added to the final report; keep in mind that the evaluation will be based on the main submitted report, not the appendix.

The report structure described is a guideline, and you can, of course, deviate from it.

PRESENTATION

The presentation/viva is an essential component of the module. It is not optional, and you **must attend**. You will be asked questions about your work and your findings. It is important that you are familiar with your work's details and provide clear and concise answers.

- You should prepare **approximately 10 slides** to give a **10-12 min presentation** of your work.
- Assessment will include evaluating your presentation and your answers during a Q&A session at the end of your presentation.
- Your supervisor(s) may or may not attend your presentation. Other students and members of the public may attend your presentation unless your presentation is confidential.
- Industry project presentations will be confidential by default, whereas academic project presentations will be public by default.
- Name your presentation slides PDF file as `COLLEGEUSERNAME-presentation.pdf` (all lowercase) and push it to your GitHub repository in `reports/` directory (e.g. `reports/xyz123-presentation.pdf`).

CODE

Add your name and GitHub username as a comment to each file you write (when possible, in the first line) and commit it to the repository. Document your code and add a README file that briefly describes the layout of your repository, how it can be compiled/executed, and what the input/output of your program is. Your code will be assessed based on the included documentation (including any installation instructions and user guide), test suites, and any additional software infrastructure that has been submitted to GitHub. In addition, markers will consider the implementation and assessment of your code, as described in your written report. **If your project is confidential, there is no need to submit your code to GitHub.**

PLAGIARISM

Plagiarism is presenting other people's ideas as your own or without giving due credit. Plagiarism is not permitted during the course and is severely penalised by the College. Please refrain from plagiarising/copying text, code, or images/graphs/figures. Since the project plan is part of the same assignment as your final report, you can re-use text from your project plan as long as it is your intellectual property without worrying about self-plagiarism. For further guidelines, please refer to Imperial College Guidelines

EXTENSIONS AND MITIGATING CIRCUMSTANCES

If you have a severe problem affecting your study, coursework, or examinations, you must inform the Senior Tutor as soon as possible. Extensions may be provided to students that have mitigating circumstances once the Senior Tutor is informed. Please inform the IRP coordinator that you have submitted a request for an extension, and please CC ese-msc-irp-EA@groups.imperial.ac.uk. For more details, please refer to Imperial College Guidelines.

GOOD PRACTICES

The following good practices could be helpful for the successful completion of your IRP:

- Be punctual and prepared for meetings with your supervisor(s).
- Consider carefully all advice you receive from your supervisor(s).

- Keep a logbook of all the steps undertaken and their results.
- Start writing your reports early: you do not have to wait to finish all coding before you begin working on your report.
- Ensure drafts of written work are sent to supervisors early to allow sufficient time to receive feedback.
- Manage your time effectively.

BAD PRACTICES

The following bad practices may lead to an unsuccessful project:

- Student spent the entire summer doing something else (for example another internship or job), and put together a submission within the last few weeks;
- Code not written by the student;
- Report written in poor English, which substantially compromises its clarity;
- Not enough code written;
- Findings/results are not clear;
- Analysis is poor, contributions of the student are not clear;
- Student did not follow instructions and advice from their supervisors;
- Code was not validated;
- No quantification of results;

(MODIFIED SWANSON) MARKING CRITERIA

DISTINCTION (70-100%) A project has sufficient quality and originality to be publishable as it stands or with minor modifications to its scientific content, but potentially with substantial improvements required to improve phrasing, spelling, or grammar. The work demonstrates new insights into the topic with an innovative approach, carried out diligently and professionally. It demonstrates significant breadth and depth with sound background research, an exemplary implementation, and a well-structured and well-presented report with the project's background, objectives, and achievements. It largely or wholly overcomes significant technical problems.

- **90 - 100%** Outstanding - making an original contribution by questioning or challenging prevailing paradigms, offering new insights informed by critical evaluation of current research practice, and clearly demonstrating innovative/creative thinking. Also, it must contain a substantial original contribution from the student. The report is publishable in a high-quality journal as is or potentially marketable. There should be no notable weaknesses beyond those reasonably expected given the timescale. The quality of the report must be excellent.
- **80 - 89%** Excellent throughout, showing detailed knowledge and systematic understanding of key aspects, with strong evidence of independent thinking and original insights on the subject. The report contains a significant portion of the results and analysis required for publication in a reputable journal, with minor modifications to the presentation and minimal additional work.
- **70 - 79%** A thorough grasp of the subject, showing the ability to synthesise and criticise, with critical use of existing literature, although occasionally falling below a general level of excellence in original insights and innovative thinking. The project demonstrates technical analysis comparable to that seen in high-quality journals. With some extension and re-organisation, the body of work presented could form the skeleton of a publication in a reputable journal or a professional technical report, with minor modifications to the text and figures and only a small amount of additional work.

MERIT (60-69%) A project displays both breadth and depth. Although the final output might lack novelty, it should demonstrate a high level of individual technical competence and professionalism. The work's importance, originality, or rigour is somewhat below the standard required for publication in a high-quality journal. There should be at least a moderate level of risk in the project's objectives, e.g., if the project was not completely specified at the outset of the work presented some difficult challenges that needed to be overcome. Any implementation should be well designed and correct, although there may be scope for improvements.

- **65 - 69%** The report demonstrates a very good grasp of the subject and evidence of ability to synthesise and criticise, including an understanding of existing literature but falling short of excellence in one or more of these aspects. The software is robust and functional, addresses a challenging problem and demonstrates significant ingenuity.
- **60 - 64%** The report demonstrates a good grasp of the subject and some evidence of the ability to synthesise and criticise. The software is robust and functional but addresses a relatively straightforward problem with some difficult challenges and demonstrates little ingenuity.

PASS (50-59%) A satisfactory project displays an ability to solve well-defined, moderately low-risk problems competently. It shows a reasonable grasp of the relevant concepts and facts but little evidence of the ability to both synthesise and evaluate. The resulting software is broadly functional but may be of limited scope or functionality. The project lacks ambition or fails to overcome the more difficult challenges associated with the problem area.

- **55 - 59%** The report shows clear understanding in most places, but the standard of work is variable. The software is functional but has a few obvious gaps.
- **50 - 54%** The report has noticeable gaps in knowledge or understanding in some key areas or lacks insight. The software is mostly functional but with significant limitations in scope or implementation.

FAIL (0-49%) The project falls below an acceptable standard, showing no significant grasp of the subject's key issues and no more than a rote reimplementation of existing solutions or their application to extremely low-risk problems or failed solutions to more general ones. The resulting software is either highly limited or non-functional.

- **40 - 49%** The report demonstrates a superficial understanding of the key issues of the subject, although lacking in focus. The software is fragile or flawed in places but shows scope to be improved to an acceptable standard.
- **30 - 39%** Although the report is of significant length, it incompletely or incoherently addresses the key issues or shows a serious misunderstanding of the subject. The software is missing significant functionality or has multiple significant errors in implementation
- **20 - 29%** The report shows some attempt to address the subject but has serious flaws throughout, with nothing of substance present. The software is fragmentary, incomplete, or wholly flawed in implementation.
- **0 - 19%** The report has multiple serious errors throughout, largely consists of irrelevant material, is far too brief or is absent. The software shows no more than a skeleton of work towards a potential solution.