Module Code
COM00019I

BSc, BEng and MEng Degree Examinations 2022–23

DEPARTMENT OF COMPUTER SCIENCE

Engineering 1 (ENG1)

Open Group Assessment

Issued: Aut/7/Wed, 9 November 2022, 12:00 (noon)
Submission due:

Assessment 1: Spr/4/Wednesday, 1 February 2023 [30%]
Assessment 2: Sum/3/Wednesday, 3 May 2023 [50%] *including*
Assessed Presentation: Sum/3/TBA

Feedback and marks due:

Assessment 1: Spr/8/Wednesday, 1 March 2023
Assessment 2: Sum/7/Wednesday, 31 May 2023

These are department-set dates. Feedback is released as soon as it is ready, so that it can be taken into account in subsequent deliverables.

All teams should submit their answers through the electronic submission system: http://www.cs.york.ac.uk/student/assessment/submit/ by

Assessment 1: Spr/4/Wednesday, 1 February 2023 [30%]
Assessment 2: Sum/3/Wednesday, 3 May 2023 [50%] *including*
Assessed Presentation: Sum/3/TBA

An assessment that has been submitted after this deadline will be marked initially as if it had been handed in on time, but the Board of Examiners will normally apply a lateness penalty.

Your attention is drawn to the section about Academic Misconduct in your Departmental Handbook: https://www.cs.york.ac.uk/student/handbook/.

Any queries on this assessment should be addressed by email to Dimitris Kolovos, Tommy Yuan, Antonio Garcia-Dominguez and Konstantinos Barmpis at dimitris.kolovos@york.ac.uk, tommy.yuan@york.ac.uk, a.garcia-dominguez@york.ac.uk and konstantinos.barmpis@york.ac.uk. Answers that apply to all students will be posted on the VLE.

No questions will be answered in the last 48 hours before the submission deadline.

Rubric:

- The two team assessments account for 80% of the total module mark. The final 20% is for the individual closed examination in the summer common assessment period.
- Answer all questions. Note the page limits for each question. Parts of answers that go beyond the page limit will not be marked. References must be listed at the end of each deliverable, and do not count towards page limits.

The name of all your group members should be on the front cover of your assessment. Do not include exam numbers as these are private to individuals.

1 ENG1 Assessment Structure

This is a **very long** assessment document.

- Read it all, together, as a team: we have not put in anything that is unimportant.
- It is long because the ENG1 assessment forms part of your software engineering training:
  - the things requested, the styles of presentation, and the feedback on the assessments, are designed to help you develop your experience and understanding of team-based software engineering;
  - nothing is there by mistake (we hope!).

The brief describing the product (game) that teams will build will be issued to teams and will appear on the ENG1 VLE site in Autumn week 7.

The **assessed presentation** for Assessment 2 will take place in the Summer term: the details will be published on the ENG1 VLE page and notified by email to all students.

Assessment 2 builds on Assessment 1. Together, the assessments constitute a complete software engineering project. **We require all teams to use Java (version 11) as their programming language.**

1.1 Time

Do not underestimate the individual and team time required for ENG1.

- The module is 20 credits, which is 200 student hours for each member of the team.
- We expect that all teams will spend more than 1000 person-hours on the teamwork.
- Work on the full set of deliverables: do not get carried away with, e.g., coding!
- Work out what needs to be researched and undertaken at the start of the project, and revisit it at the start of every assessment. The best results come in teams that are very strategic in their approach to the overall project.
- Talk to the lecturers whenever you need help or reassurance!

In this document, Section 2 outlines team set-up and team-mark arrangements; Section 3 describes the two team assessments, and outlines the deliverables required for each task; and Section 4 gives guidance on the style and marking of each sort of deliverable, as well as team issues, and various forms of reassessment.

The assessment refers to the “scenario”: this is the product brief that is issued at the

team-forming practical in Autumn week 7, and is subsequently available on the ENG1 VLE site.

1.2 Formatting Requirements

Deliverables should be single-spaced using a sans-serif 11-point font and should include a title page, with the title of the deliverable, the group number and name, and the group member names (**not** examination numbers) clearly stated on it. This title page is not included in the word limit for the report. The font size in figures and tables must be no smaller than 10 points. References must be listed at the end of each deliverable, and do not count towards page limits. Appendices are not permitted; any additional content (e.g. detailed diagrams) should be placed on the team's website and linked from the deliverable.

2 Project Teams and Team Marks

The project is designed to be undertaken by teams of 5, 6 or 7 students.

- It is up to each team to develop its own structures, team leadership and team arrangements. There is material provided, but teams should research team working (there are some notes from previous teams on the VLE site), and consult the module lecturers as soon as team-related problems arise.
- By analogy to project teams in some software development companies, the lead lecturers (Prof. Dimitris Kolovos, Dr Tommy Yuan, Dr Antonio Garcia-Dominguez and Dr Konstantinos Barmpis) form a higher management level, and should be consulted for things beyond the control of the team, for instance where team arrangements do not work effectively, or workload becomes unbalanced and the team cannot reach an equitable adjustment.
- The lead lecturers (Prof. Dimitris Kolovos, Dr Tommy Yuan, Dr Antonio Garcia-Dominguez and Dr Konstantinos Barmpis) are the final arbiters in any dispute arising from the composition, performance or size of teams, and in the distribution of marks to members of a team.

2.1 Team marks and peer-assessment

Each assessment is marked out of 100. The first assessment is weighted at 30% of the module mark, and the second assessment carries 50% of the module mark.

The mark awarded to each student is the sum of the team mark, the peer-assessment mark, any applicable bonus marks, and any individual penalty (section 2.3). This section explains the

peer-assessment mark.

In the context of each assessment, each student must submit a peer assessment for each group member, including themselves. The assessment will consist of a 0-10 mark for the significance of the contribution of the group member as well as an opportunity for open-ended comments regarding group member performance. This information will be submitted in confidence to the lecturers through an online survey mechanism that will be announced on the ENG1 VLE site shortly after each assessment submission deadline. Failure to submit valid peer assessments will result in **a zero mark** for the peer assessment component of the submitting student's mark.

During marking of the assessments, the marking team will examine the peer assessment reports and will also take into consideration the students' practical attendance record. If the marking team is concerned about the level of the contribution of a student to the group project, they may request and examine further evidence (e.g. code commits, contributions to deliverables, group meeting minutes) and/or invite the student for an interview, to determine if an individual penalty is applicable.

If it is deemed from all of this evidence that the student did not participate in the group project in a substantial and meaningful way, there will be a deduction to the student's overall mark on the assessment (see Section 2.3).

2.2 Equitable Work Allocation

As you are planning your work for each assessment, please keep in mind that effort must be distributed in an equitable manner. That is, in a team of 6, each team member must be allocated (and carry out) work that corresponds to $\approx$15 of the 90 marks of the assessment deliverables. For example, a team member should not be allocated to work exclusively on the team's website as the respective deliverable is only worth 3 marks.

If you have any concerns about the amount of work assigned to you by your team, please discuss these with your teammates as a matter of urgency, and if you are not satisfied with the resolution please raise this with the lead lecturers (Prof. Dimitris Kolovos, Dr Tommy Yuan, Dr Antonio Garcia-Dominguez and Dr Konstantinos Barmpis) as management, as soon as possible. Only raising such issues shortly before the hand-in or in your peer-assessment report will not be acceptable justification for a lower contribution to the project.

2.3 Individual Penalties

It is important that **all** problems with participation in team work and assessment activities are discussed in a timely manner with the management (as above). ***Problems can be raised at any time***, not only during an assessment activity.

Whilst the module lecturers can help, they cannot force equal participation. So, a student who does not participate adequately in team work and assessment activities, for whatever reason, or who is awarded a low peer assessment mark for an assessment, or who otherwise misses a substantial part of ENG1, is likely to have marks deducted from one or both team assessments.

Individual penalties are applied by the lead lecturers (Prof. Dimitris Kolovos, Dr Tommy Yuan, Dr Antonio Garcia-Dominguez and Dr Konstantinos Barmpis). It is normal that penalties are only applied after requesting evidence of contribution from the individual concerned. Total non-participation results in **a zero mark** (a zero assessment mark, no team bonus marks and no peer-assessment mark) for the individual student, for all affected team assessments.

3 The Team Assessments

Each team assessment addresses specific software engineering aspects of the game described in the product brief.

Teams must bear their own costs (if any - e.g. if they choose to use commercial software not provided by the Department), and are responsible for all aspects of management of their own work.

Where deliverables of assessment 2 build on deliverables of assessment 1, teams should reuse and extend **the other team's** previous deliverables, including all material provided with a selected software product. Where a deliverable asks for an updated version of an earlier deliverable, teams will be marked on the updates – it is essential that changes are clearly identified.

- Section 4 gives further advice on the content and assessment of each type of deliverable, and complements the requirements for each assessment, below.
- Note that the ENG1 lectures give a general introduction to the software engineering required for the teamwork; teams are expected to research, and to develop their skills through their research, and through reflection on feedback given in practicals and on assessments.

3.1 KISS

If you do not know what this means, find out!

This project requires a non-trivial amount of research, design and implementation effort. A key principle of software engineering is that products should meet their requirements. There is no merit in producing something that goes beyond what is required. You are STRONGLY DISCOURAGED from adding unjustifiable additional features.

3.2 Electronic Submission

Your submission for each team assessment must be one zipfile. The details of what the zipfile should contain for each assessment are given in tables 1 and 2, below. The zipfile must be named using the team number, *team-N.*zip.

- The electronic submission system is configured so that any of you can submit an ENG1 team assessment on behalf of your team. Please note that only **one** nominated team member must submit on behalf of each team.

3.3 Assessment 1: Greenfield Development, due: noon, Spr/4/Wed

Assessment 1 covers the main stages of the software engineering lifecycle – from requirements elicitation to implementation – in a greenfield setting. As part of the assessment, each team will need to elicit requirements, develop an architecture, identify appropriate software engineering methods and techniques, identify risks and their mitigation, and implement a first version of the system specified in the scenario.

3.3.1 Deliverables for Assessment 1

Your team will submit a website plus a single .zip file. The requirements for the deliverables to be included in the zipfile are summarised in Table 1.

Table 1: Assessment 1 zipfile contents

	Deliverable	**Max. mark**	**Page limit**	**File name and format**
1.	Website (submit only the URL)	3	—	url1.txt
2.	Requirements	20	$1 + 3$	Req1.pdf
3.	Architecture	22	$3 + 3$	Arch1.pdf
4.	Method selection and planning	10	$2 + 1 + 2$	Plan1.pdf
5.	Risk assessment and mitigation	10	$1 + 3$	Risk1.pdf
6.	Implementation	25	1	Impl1.pdf + Code + Executable JAR

1. Website [3 marks]

   a) The submitted URL must link to the website that is the “public face” of your team’s project, and will be updated as you proceed.

   b) The website must include links to all the PDF documents listed in Table 1, to the executable JAR of your game, and to the version control repository of your team’s code in a clear and accessible way.

   c) The “management” and other teams can use the website at any time after the submission deadline to access the material above.

   d) In this deliverable, it is the website structure that is marked. You will be penalised if material is not easily locatable and accessible.

2. Requirements [20 marks]:

    a) Write a succinct introduction explaining how requirements were elicited and negotiated, and why they are presented as they are. Your submission should evidence research into requirements specification and presentation (4 marks, $\leq 1\ page$).

    b) Give a systematic and appropriately-formatted statement of user and system requirements (16 marks, $\leq 3\ pages$).

    Note that you will need a requirements referencing system, and may need to update this for subsequent assessment deliverables.

3. Architecture [22 marks]:

    a) Give diagrammatic representations (structural and behavioural diagrams) of the architecture of the team's product, with a brief statement of the specific languages (for instance, relevant parts of UML) and the tool(s) used to create these representations (10 marks, $\leq 3\ pages$).

    b) Give a systematic justification for the architecture and describe how it was initially designed and how it evolved over the course of the project. Provide evidence of the design process followed (e.g. interim versions of architectural diagrams, CRC cards) on your team's website and link to them from your report. Relate the architecture clearly to the requirements, using your requirements referencing for identification, and consistent naming of constructs to provide traceability (12 marks, $\leq 3\ pages$).

4. Method selection and planning [10 marks]:

    a) Give an outline and justification of the team's software engineering methods, and identify any development or collaboration tools that the team has used to support the project or the team working. Justify the fitness of the selected tools with the team's software engineering methods and discuss alternatives considered. (3 marks, $\leq 2\ pages$).

    b) Outline the team's approach to team organisation, and explain why the chosen approach is appropriate for both the team and the project (2 marks, $\leq 1\ page$).

    c) Give a systematic plan for the project. Your plan should lay out the key tasks, their starting and finishing dates, as well as task priorities and dependencies. Provide weekly snapshots of the plan on your team's website and discuss how the plan evolved throughout the duration of the project (5 marks, $\leq 2\ pages$).

5. Risk assessment and mitigation: [10 marks]

    a) Describe and justify the risk management process followed by your team and the format of your team's risk register (3 marks, $\leq 1\ page$).

    b) Give a systematic tabular presentation of risks (risk register) to the project, their likelihood, impact, mitigation and ownership (7 marks, $\leq 3\ pages$).

    ENG1 is a small project, developing non-critical software. Keep your likelihood and impact measures simple.

6. Implementation [25 marks]:

    a) Provide documented code for a working implementation of the part of the game that meets the remit, requirements and architecture for Assessment 1. The code and an executable JAR of the game, that includes all external dependencies, must be included in the zipfile. (20 marks)

    b) List any 3rd-party libraries or assets you may have used in your implementation and the licenses under which they are made available. Briefly discuss the suitability of these licenses for your project. State explicitly any of the features required for Assessment 1 that are not (fully) implemented, using your requirements referencing for identification, and consistent naming of constructs to provide traceability. (5 marks, $\leq 1\ pages$)

See Section 2 for information on peer assessment. [10 marks]

You may use additional pages for a bibliography. Any other content that overruns the page limit will not be considered by the markers and will not receive any marks.

3.4 Assessment 2: Brownfield Development, due: noon, Sum/3/Wed

Assessment 2 requires each team to work on another team's product. There are thus two phases to the assessment, below.

3.4.1 Assessment 2, Phase 1: selection: completed Spr/5/Fri at 17:00

After the submission of Assessment 1, each team has a short period to consider the products of the other teams. In the **Spring Week 4 practical** class that your team attends, each team will present its product, and can then discuss with other teams. On Spr/5/Friday, each team is required to register, by 17:00, their choice of product to work on in Assessment 2, **by emailing the lecturers**.

The only constraint on selection is that a team is not allowed to use its own Assessment 1 software in Assessment 2.

In selecting a software product, criteria that may be considered include: (1) the overall quality of the software product; (2) estimates of effort remaining to complete the implementation; (3) clarity and quality of the requirements specification, architecture and implementation; (4) testability.

A bonus of 2 marks will be added to the team's Assessment 1 mark for each registered selection of the team's software product, with the condition that no individual can attain more than 100% for Assessment 1.

Important: By selecting a team's project for Assessment 2, beyond their implementation, your team will also take over all of their Assessment 1 deliverables (including a copy of their website), which your team must update for Assessment 2. For Assessment 2 you must no longer continue working on your own team's Assessment 1 deliverables.

3.4.2 Assessment 2, Phase 2: extension and integration

Your selected product should have designed and implemented the initial elements of the game, as indicated in the product brief. For Assessment 2, the selected software product must be extended to cover the full product brief.

For this assessment, you are allowed, but not required, to add other features or behaviours, as desired, but these must be fully explained and justified in the deliverables, and must not violate the requirements for the game.

3.4.3 Assessment 2: Assessed Presentation [5 marks]

Each team is required to make a presentation of the finished game as if to an external client. The client, who will be an experienced software engineer or games developer, will question the teams about their games.

The presentation should assume that the client is interested in buying or marketing the product, and is thus aware of the requirements specified for the product (including the changes in Assessment 2). Whilst the client will be interested, in general, in major design decisions, the quality of the software, and the playability of the game, s/he is likely to be also interested in the potential market for, and extensibility of, the product.

The presentation will take place in a meeting room with access to the University computer network: within this constraint, teams can use any appropriate media, and any combination of team members may take part. The presentation should take at most **5 minutes**.

Each team will be marked on the clarity and appropriateness of its presentation and its interaction with the client. Marking is independent of the client's award (the prize!) for what s/he considers the best product.

3.4.4 Other Deliverables for Assessment 2

Your team will submit a website plus a single .zip file. The requirements for the deliverables to be included in the zipfile are summarised in Table 2.

Table 2: Assessment 2 zipfile contents

	Deliverable	**Max. mark**	**Page limit**	**File name and format**
1.	Website (submit only the URL)	3	—	url2.txt
2.	Change report	25	1 + 8	Change2.pdf
3.	Implementation	25	1	Impl2.pdf + Code + Executable JAR
4.	Testing	22	4	Test2.pdf
5.	Continuous Integration	10	2	CI2.pdf

1. The updated Website for your team's project [3 marks].

    a) The website in the submitted URL should now link to:

        i. all the project-related Assessment 2 deliverables, as well as the other team's Assessment 1 versions (please note the specific requirements for updates,

below);

ii. the executable JAR for the game;

iii. the version control repository of your team's code.

b) You are expected to clone the other team's website and add the new Assessment 2 material to it rather than creating a new website from scratch or extending your own team's Assessment 1 website.

2. Change Report [25 marks]:

a) Briefly summarise the processes, tools and conventions your team used to plan, make, keep track of, and review changes to the Assessment 1 deliverables, documentation and code you inherited from the other team. (5 marks, $\leq 1\ page$)

b) For each of the following items, include a brief explanation and justification of any changes made to the other team's Assessment 1 deliverables. Include the precise URLs of the original (i.e. the other team's) and the updated versions of the deliverables. The updated deliverables do not need to adhere to the page limits for Assessment 1. If there is no change to report, please state and justify why no change was necessary.
(Maximum 5 marks per item, to a maximum total of 20 marks; $\leq 8\ pages$ in total with $\leq 3\ pages$ per item)

i. Requirements

ii. Architecture

iii. Method selection and planning

iv. Risk assessment and mitigation

3. Implementation [25 marks]:

a) Provide documented code for a working implementation of the game that meets the remit, requirements and architecture for Assessment 2. Your code comments should highlight new or extended sections of code, and should be consistent with your change report. The complete code and an executable JAR of the game that includes all external dependencies, must be included in the zipfile. (20 marks)

b) List any 3rd-party libraries or assets you may have used in your implementation and the licenses under which they are made available. Briefly discuss the suitability of these licenses for your project. State explicitly any of the features required that are not (fully) implemented, using your requirements referencing for identification, and consistent naming of constructs to provide traceability. (5 marks, $\leq 1\ pages$)

4. Software Testing Report [22 marks]:

    a) Briefly summarise your testing method(s) and approach(es), explaining why these are appropriate for the project. (5 marks, $\leq$ 1 *page*)

    b) Give a brief report on the actual tests, including statistics of what tests were run and what results were achieved, with a clear statement of any tests that are failed by the current implementation. If some tests failed, explain why these do not or cannot be passed and comment on what is needed to enable all tests to be passed. If no tests failed, comment on the completeness and correctness of your tests instead (12 marks, $\leq$ 3 *pages*).

    c) Provide the precise URLs for the testing material on the website: this material should comprise the testing results and coverage report generated by your automated testing tooling, and descriptions of manual test-cases that you designed to test the parts of the code that could not be covered by your automated tests (5 marks).

5. Continuous Integration Report [10 marks]:

    a) Summarise your continuous integration (CI) method(s) and approach(es), explaining why these are appropriate for the project (5 marks, $\leq$ 1 *page*).

    b) Give a brief report on the actual continuous integration infrastructure you have set up for your project (5 marks, $\leq$ 1 *page*).

See Section 2 for information on peer assessment. [10 marks]

You may use additional pages for a bibliography. Any other content that overruns the page limit will not be considered by the markers and will not receive any marks.

4 Assessment notes

To pass ENG1, a student must reach the overall module pass mark. There is no requirement to pass any one of the component assessments.

4.1 General Approach and Style

Teams are expected to *research* their software engineering needs and develop their experience and expertise in relation to team management, software engineering methods and tools, risk management, requirements, design/architecture, change management and version control, coding and GUI design.

- There is a wide range of material on line, and in standard texts such as Sommerville's *Software Engineering* (any recent edition is good: it's also on line).
- Credit will be given for evidence of research and for appropriate consideration of alternatives where explanation or justification is requested.
- The best results are likely to come from regular review and updating of your approaches to the project, rather than leaving these activities til the point when you write each report.
- Good software engineering is not a set of independent activities and documents: you should ensure that your documentation and code are internally and mutually consistent.
- You will **never** be penalised for asking for clarification.

We are looking for *clear, succinct presentation*, not essays. The ENG1 lecturers have to assess, mark and provide feedback on a significant amount of material in a short time (during which they have many other obligations to fulfil!): the easier it is to read your reports, the better chance you have of a good mark.

- As a team, try to understand the markers' position, and work out a clear consistent presentation style.
- You must clearly format your reports so that they match the questions and sections of questions: it must be unambiguous what part of the assessment you think you are addressing in every part of every report.
- You will lose marks for ignoring instructions or for doing what you think we might want rather than what the question asks for.
- Take out superfluous opinions and adjectives: they have no part in an engineering report.
- Use bullet lists, or text bullets (as used in this document), and signal topics clearly, e.g. by subheadings or emboldened lead-words.

- Page limits are limits not targets: if you do not need all the pages, you will **not** be penalised for writing less, well. However, you are likely to lose credit for wordy, rambling, imprecise, or poorly-presented work.
- Spell check AND proof read all documents before submission.

When submitting your assessments, please ensure that all the deliverables are in accordance with the instructions for that element of assessment (Section 3), including zipfile information on naming.

4.2 Website and website deliverables

The website is used to provide a working software-engineering resource for you, other teams, and the markers, giving access to documents (e.g. requirements, architecture, testing materials, etc.) and executables.

- You will be marked on the structure of your website: how easy it is to locate required content.
- Markers will **not** spend time hunting for material on the website, so you need a good site structure, and the route to specific software-engineering resources (including all required product materials) must be clear.

Please note that, if you cannot stay within the assessment page limits for “factual” deliverables such as the requirements or risk assessment, architecture diagrams, test evidence, etc., it is often appropriate to include the full material (suitably introduced, linked and formatted) on the website, and present an appropriately-chosen subset (with explanation and cross-reference links for the full material) in the assessment deliverable.

4.3 Requirements documentation

- You will be marked on the clarity and appropriateness of your approach, and appropriateness and presentation of requirements, **not** the number of requirements that you state.
- Good software engineering pays attention to traceability: (a) from requirements through architecture, to code and tests; and (b) across the development lifespan. Marking will reflect how well your requirements presentation supports traceability, e.g. through its approach to requirement identification.
- Your requirements are assessed on their objectivity: is there a test that can demonstrate whether the requirement is met? For requirements that are inherently difficult to make objective, you may get credit for explaining how, subjectively, a requirement can be shown

to be met.

4.4 Architecture documentation

- You will be marked on how well your architecture describes the essential structure and behaviour of your system: you will lose marks for making the architecture too detailed.
- You will be marked on how clearly you describe your team's design process and the evolution of your architecture over time.
- You will be assessed on the clarity and appropriateness with which you state the language(s) and tools that you use.
- You will be marked on how clearly your architecture justification follows the structure of your architecture, and accounts for unusual features or notations.
- You will be marked on how clearly and comprehensively you trace the architecture back to the requirements of the system.

4.5 Implementation

- You will be marked on the software engineering quality of your code, **not** its cleverness.
- You will be marked on the presentation (e.g. consistent indentation, naming, formatting, commenting, coding style) and on the maintainability (e.g. meaningful class/field/method names, high cohesion, low coupling) of your code.
- Use formatting, naming conventions, etc. to make it easy to trace between your code and all relevant documentation.

The project is primarily about software engineering of the game: there is no teaching on GUI design, and an appropriately *small amount of credit* is available for your GUI.

- It is up to each team to decide how much effort they put into the visuals of the game: this may attract other teams (and the presentation client) to your product, may help to meet requirements, etc., but you do not need to put a lot of work into the GUI write-up!

4.6 Software testing

- Prefer automated testing to manual testing where possible.

- Clearly describe the parts of the system that you were unable to test automatically or for which manual testing was preferable.
- Poor testability of the code you inherited from the other team is not a valid reason for not carrying out automated testing; you are expected to refactor the code and make it testable.
- Tests must be provided for all requirements of the system. You will be marked on traceability between tests and requirements.

4.7 Methods and plans, and their updating

- Make sure that you understand what is meant by the words used in the questions: terminology is used in its software engineering sense, not general English usage.
- Only answer the questions asked: you will not get credit for exploring other aspects of your team work in your answers.
- When you are asked to update or replace a report, you will gain credit for doing what is asked – you will lose credit, for instance, if it is unclear what you changed.

4.8 Risk assessment and mitigation, and its updating

A good risk assessment and mitigation aims to identify things that are risky and to mitigate effects, **not** to try to prevent occurrence of risks.

- Follow the guidance for “Methods and plans”, above.
- Research risk assessment and mitigation: you will get credit for the appropriateness of your approach and presentation to the type of project.
- You will be marked on the clarity and appropriateness of your approach, and appropriateness and presentation of risks and mitigation, **not** the number of risks that you state.

4.9 Change reporting

Change management, and software maintenance, are key activities in practical software engineering. You need to research these, and to plan for change even at the start of your project.

- Try not to report the same changes in different deliverables: you can cross-refer between reports as needed (give URLs if reports are not in the current deliverables).

- You should focus on discussing and justifying major changes, rather than listing minor (uncontentious, no side-effects) changes.
- You will gain credit for the consistency with which large changes are tracked and traceable through your project documentation.

4.10 Continuous Integration

- In deliverable 5a of Assessment 2 you are expected to list the continuous integration (CI) pipelines in your project and discuss their inputs, outputs and how they are triggered. You will be marked on the comprehensiveness of your CI solution and the clarity of its presentation.
- In deliverable 5b of Assessment 2 you are expected to explain the actual implementations of your project's CI pipelines (e.g. GitHub Actions YAML). You will be marked on the clarity with which you present the CI pipelines.

4.11 Team issues, reassessment, and mitigation

As endlessly repeated in the assessment document, and in lectures and practicals, if a team is not functioning or you feel that the work is not being shared in an equitable way, then you are ***strongly encouraged to contact the lead lecturers*** as soon as possible (email dimitris.kolovos@york.ac.uk, tommy.yuan@york.ac.uk, a.garcia-dominguez@york.ac.uk and konstantinos.barmpis@york.ac.uk). There are many ways we can try to help, but we cannot help if we do not know that there is a problem, or if we only find out at or after an assessment submission date.

4.12 Reassessment

If an individual fails ENG1, there are two reassessment elements: there is a resit of the individual examination (a new paper), and there is a reassessment essay on the teamwork. The setters will specify part of the project, and frame the reassessment essay to address the key software engineering learning objectives. The reassessment essay will also require some self-reflection, to assess the candidate's understanding of the teamwork learning objective.

The candidate is expected to contextualise the stated part of the project, and to complete the reassessment on their own (i.e. without interacting with their team or other ENG1 students), using any of their team's materials that they have access to. Reassessment candidates will **not** be able to request or require materials: they must make their own arrangements to access team products.

Remember that ENG1 is a 20-credit module, and that a large number of hours are allocated to the teamwork assessments. It is not going to be a short reassessment or one that is easy to pass.

End of examination paper