# Software Testing 2024-5 Grading Scheme

In this grading scheme you will provide your auditor with the information they need to determine a mark for your coursework. There are opportunities for you to emphasize the areas you believe are excellent and de-emphasize those you believe are less well developed. The aim of this grading scheme is to provide you with a structure to demonstrate that you have achieved the learning outcomes for the course. The awarded grade will vary depending on the quality of the evidence you present but, provided you demonstrate you have met the learning outcomes you will pass the course. This grading scheme is available as a Microsoft Form and this is how you will submit your grading.

This is the grading scheme for the portfolio you develop. You will use it to evaluate your work and your auditor will use it to arrive at a grade for your work. You should use this grading scheme to provide a self- evaluation of your portfolio. The portfolio is a **short** document that presents evidence that you have achieved each of the learning outcomes of the Software Testing course. **The portfolio should be a maximum of 3 pages long of at least 11-point text and normal margins. You are provided with a Word and Latex template for the portfolio.** The portfolio will refer to work you have done throughout the course to provide evidence that your evaluation of your work is sound. This will be checked by the auditor and your self-evaluation grades may be modified to reflect the expert opinion of your auditor.

Before grading you should consider the University-wide common marking scheme. This is constructed to permit sufficient range to allow recognition of variation in ability, particularly the recognition of unusually high ability.

| Honours Class | Mark (%) | Grade | Non-Honours Description |
|---|---|---|---|
| 1st | 90-100 | A1 | Excellent |
| 1st | 80-89 | A2 | Excellent |
| 1st | 70-79 | A3 | Excellent |
| 2.1 | 60-69 | B | Very Good |

| | | | |
|---|---|---|---|
| 2.2 | 50-59 | C | Performance at a level showing the potential to achieve at least a lower second-class honours degree |
| 3rd | 40-49 | D | Pass, may not be sufficient for progression to an honours programme |
| Fail | 30-39 | E | Marginal Fail |
| Fail | 20-29 | F | Clear Fail |
| Fail | 10-19 | G | Bad Fail |
| Fail | 0-9 | H | Bad Fail |

At honours level the University recognizes excellence by awarding a first-class honours degree for a performance that achieves a mark of 70 or more. For the Software Testing course, a mark of 70-79 is awarded for a performance that demonstrates excellence within the confines of the course (I.e. a performance that mostly draws on the material of the course). Marks in the range 80-89 will be awarded for a performance that demonstrates meaningful synergy with material beyond the material of the course. Here "meaningful synergy" means producing work that is markedly better than could be achieved by only drawing on the course material and that improvement is based directly on knowledge and skills derived from outside the course material (e.g. from other courses, reading or other experience). Marks in the range 90-100 will be awarded for a performance that arrives at a result that is better than could be achieved by only drawing on the course material and makes use of novel techniques that are not easily accessible in the literature (such marks are rarely awarded and recognize significant novelty – the 90-100 range is intended to recognize exceptional creativity, it would be unusual to see marks in this range).

## System under test

In the Software Testing coursework, you are aiming to use the testing of software you are already developing or are familiar with as the vehicle to demonstrate you have achieved the learning outcomes of the course. You have a free choice in this but if you do not have a suitable candidate we have supplied a simple JavaScript project that you can work on. Your portfolio should begin with a one paragraph outline of the software. The software itself should be sufficiently documented that a range of requirements are included with the documentation.

## Weighting

There are 5 learning outcomes for the course, and these are listed below. You must demonstrate you have achieved these outcomes, but **you** can choose to vary the weighting for each outcome within limits. The weighting for each outcome can vary between 15% and 25%. The total of the weights must be 100%. To simplify matters, weights must also be a multiple of 5. So, for example, if you think you have done a good job of analyzing

requirements and poorer job of evaluating the limitations of your testing processes you might want to weight learning outcome 1 as 25% and learning outcome 4 as 15%.

## Portfolio Structure and Grading

Your portfolio should be structured into sections, one for each of the learning outcomes. Each section will be quite brief and will point to supporting evidence that you have gathered and stored in your coursework repository. **Recall the max length is 3 pages**.

The grading scheme for the portfolio is provided below and you should submit your selfevaluation of your work along with the portfolio.

Each of the Learning outcomes is broken down into **four grading elements** and each of these can be graded according to six different grades depending on the assessment of the element. These grades are:

- 0: The element is **missing**.
- 1: The element is **poor**. This means that there is some attempt to cover the element, but the portfolio does not provide sufficient evidence that there is a good understanding of the activities covered by the element.
- 2: The element is **fair**: there is an indication that there is some understanding of the element and there is some evidence that the associated activities have been carried out but there is insufficient evidence to support a clear grasp of the activity. A grade 2 across the board results in a mark of 40.
- 3: The element is **sound:** there is an indication that there is good understanding of the element and the activities associated with the element are evidenced in the portfolio. A grade 3 across the board results in a mark of 60.
- 4: The element is **excellent**: there is indication of a very strong grasp of the element and the activities have been carried out thoroughly and this is evidenced in the portfolio. A grade 4 across the board results in a mark of 80.
- 5: The element is **exceptional**: The portfolio provides evidence for the award of a grade 4 and provides evidence that approaches that go beyond those taught in the course have been used and provide significant improvement over what could have been achieved using techniques covered in the course.

1. Analyze requirements to determine appropriate testing strategies [Default Weight: 20%]
   In this section you are evaluating the statement of the requirements on the software you have chosen. These should afford a range of different types of requirement that allow the portfolio to display the techniques covered in the course (1.1). The requirements should be demonstrably applicable at different levels so some should be particular to individual components and on the range up to system level attributes (1.2). The portfolio should identify testing approaches for the requirements identified, this should cover a range of different approaches (1.3). The final component should assess how well the portfolio justifies the choice of testing approach (1.4).
   1.1. Range of requirements, functional requirements, measurable quality attributes, qualitative requirements, …

1.2. Level of requirements, system, integration, unit.
1.3. Identifying test approach for chosen attributes.
1.4. Assess the appropriateness of your chosen testing approach.

2. Design and implement comprehensive test plans with instrumented code [Default Weight: 20%] The course advocates the use of Test-Driven development the assessment should take account of this because the plan is likely to evolve during the development. The portfolio should outline a plan for testing that is consistent with the requirements and test approaches identified in section 1 and ensures adequate testing (2.1). The evaluation of the plan should identify any potential omissions or vulnerabilities of the plan and assess how well it will ensure adequate testing (2.2). In order to test the code adequately there will need to be additional instrumentation. Here you should assess how well the portfolio presents the instrumentation. Is it clear what has been done (2.3)? How good is the instrumentation? Could it be improved to test the requirements more adequately? (2.4)
   2.1. Construction of the test plan.
   2.2. Evaluation of the quality of the test plan.
   2.3. Instrumentation of the code.
   2.4. Evaluation of the instrumentation.
3. Apply a wide variety of testing techniques and compute test coverage and yield according to a variety of criteria [Default Weight: 20%]. This section assesses how well the planned testing has gone in practice. Were all the planned techniques used and how well they were implemented The portfolio should outline how well the implemented tests compare with the planned testing (3.1). The portfolio should contain a brief discussion motivating the choice of evaluation criteria used for each testing method (3.2). The portfolio should also present a brief overview of the results of testing that points to more detailed work this should communicate the results effectively (3.3). The final element in the portfolio should cover the application of the chosen evaluation techniques and this section of the portfolio should effectively communicate the results of the evaluation (3.4).
   3.1. Range of techniques.
   3.2. Evaluation criteria for the adequacy of the testing.
   3.3. Results of testing.
   3.4. Evaluation of the results.
4. Evaluate the limitations of a given testing process, using statistical methods where appropriate, and summarise outcomes. [Default Weight 20%This section of the portfolio should provide an overall evaluation of the testing process. The first part covers any omissions or deficiencies in the testing carried out and how it could be improved (4.1). The next section should identify and discuss the setting of target levels for the tests motivating these adequately (4.2). There should be discussion and explanation of how well testing targets have been met by the testing motivating variations from target (4.3). Finally, the last section should discuss what could be done to achieve or exceed the target levels (4.4).
   4.1. Identifying gaps and omission in the testing process.
   4.2. Identifying target coverage/performance levels for the different testing procedures.

4.3. Discussing how the testing carried out compares with the target levels.
4.4. Discussion of what would be necessary to achieve the target levels.

5. Conduct reviews and inspections and design and implement automated testing processes. [Default Weight: 20%] This section covers review processes and the place of testing in modern software development in a DevOps environment using CI/CD. Software quality depends on may different elements and review is important. The portfolio should identify appropriate review techniques and point to the results (5.1). The remaining three sections relate to the section of the portfolio that describes the CI pipeline constructed (5.2), the embedding of testing in the pipeline (5.3) and evidence that the pipeline behaves as expected (5.4). You should evaluate the quality of the work done in devising, building and operating the pipeline.
   5.1. Identify and apply review criteria to selected parts of the code and identify issues in the code.
   5.2. Construct an appropriate CI pipeline for the software.
   5.3. Automate some aspects of the testing.
   5.4. Demonstrate the CI pipeline functions as expected.

Thus, each of the grading elements is given a score in the range 0-5 and these are summed to give a mark on the range 0-20 for each of the learning outcomes. These five marks are then summed according to the weights you have chosen for each learning outcome. To give a final score out of 100.

### Quizzes

For each of the Learning Outcomes a quiz will be available towards the end of the teaching covering a particular learning outcome. Typically, this will be on the Thursday of weeks 3, 5, 7, 9, and 11. Each quiz will be available for 12 hours 0800-2000 on the day and should take approximately 1 hour to complete. Quizzes are marked on a Pass/Fail basis. Gaining marks of 40% and above on a quiz is deemed to be a Pass, otherwise it is a Fail. The quizzes are not intended fully to assess the Learning Outcome they just indicate an acceptable level of competence in the Learning Outcome. **High marks on a quiz should not be interpreted as indicating high attainment** since they do not cover more challenging aspects. A mark of 40% and 100% have the same interpretation, i.e. Pass.

The quizzes are a means of ensuring basic competence in each of the learning outcomes. So, if you have passed the quiz for a particular learning outcome you can return a mark of no less than 2 out of 5 for each of the sub-criteria for that learning outcome. This means that if you only complete the quizzes and pass each one you will gain a mark of 40% (i.e. a bare pass) overall. Answers to the quizzes will not be released in order to permit those who have missed the quiz through illness etc to take the quizzes later. However, we are happy to discuss issues you may have with the grading. In our experience, keeping up with the reading is sufficient to ensure a passing grade.

## Your Portfolio

In this section we will illustrate the construction of your portfolio. The portfolio is a

summary of the work you have done. We anticipate it will contain many pointers to work you have done but it should be kept short and should be 3 pages in length with normal margins line spacing and font **larger** than 10 point. The portfolio should be structured according to the marking scheme. The portfolio will be submitted to a Turnitin on the course LEARN page. In addition to submitting you should also submit a self-assessment following this form.

We will illustrate the completion of the portfolio using an imaginary project that has the following characteristics:

**OrderBot**: OrderBot is intended to receive orders for items drawn from a catalogue of items (e.g. they could be food items or books, or other consumer goods). It has the following behaviour:

- It receives a mobile number and order message. It parses the message and responds either with a failure or success message. The success message will include a unique order identifier that is associated with the order.
- On request, it generates a summary list of outstanding orders one line per mobile number with the list of items ordered by that mobile number - this should be in 1-1 correspondence with the orders received that have not yet been actioned. The list will include the unique order identifier for each order.
- It receives an order identifier, and authorization code and marks that order as having been fulfilled so it no longer appears in the list of outstanding orders.

1. Analyze requirements to determine appropriate testing strategies
   1.1. Range of requirements, functional requirements, measurable quality attributes, qualitative requirements, … Under this section you would be expected to point to a document that lists the requirements you plan to test for and that this covers a range of different types of requirement e.g., functional requirements, performance attributes and requirements, security requirements or robustness requirements. Your justification for an allocated grade would be that you have reasonable diversity of requirements. Awarding above 4 would require very strong evidence.
   1.2. Level of requirements, system, integration, unit. Here you would need to demonstrate that you have considered a range of requirements that cover different levels of concern. In the case of the bot this might include: some system level requirements, e.g., that it responds correctly to messages, some unit level requirements e.g., that the parsing works correctly, rejecting malformed messages, accepting good ones. You might also conder some integration requirements e.g., that parsing integrates with message reading so you can read and accept or reject messages. Performance requirements could include that responses are fast enough that the parsing is efficient enough. For security, that the system is not susceptible to SQL injection. You might also consider things like robustness to failure of components or failure of an externally provided service.
   1.3. Identifying test approach for chosen attributes. You will learn a range of testing approaches during the course. This part of the grading will assess the range of techniques and how well they match the requirements you are working with. For

OrderBot this might include unit tests that cover the correctness of the parser, integration testing to see if components integrate properly and system level testing that checks the system delivers the service the user requires. You might also consider forms of statistical testing to assess performance characteristics. Timing tests if the system has real time requirements.

1.4. Assess the appropriateness of your chosen testing approach. This is an overall assessment of how well matched your choices of test are to ensuring your requirements. Here you would look at the overall match of the requirements to the types of tests. For OrderBot you might consider each of the chosen test types and identify potential deficiencies in your choice of test for the requirement. To get a good grade here you do not need to have a perfect choice of test type, understanding the limitations of your approach is as important as having a perfect match. For example, you may devise tests for the parser that do not include SQL injection resistance tests because you know the implementation will not use a database. This is a valid choice, but it might be a weakness for parsers that are susceptible to such compromises.

2. Design and implement comprehensive test plans with instrumented code. Section 2.4 asks you to evaluate how adequate you think the instrumentation is. For example, in the OrderBot you might decide that the order tracking should have been timestamped and feel this is a deficiency.

2.1. Construction of the test plan: Here we are recommending a Test-Driven Design approach so, in some sense your test plan will be the approach you take to deciding on how to evolve your test set as the need for functionality evolves in your chosen project. For example, in the OrderBot case, suppose you identify the requirement that a user should be able to check what outstanding orders they have at any given time. How does this change the test set? Does this change the system level tests? How should such tests prompt the consideration of other tests? Are there performance-related tests that should be carried out? The portfolio should point to documentation for your planning and how this has developed the test set through the life of your project.

2.2. Evaluation of the quality of the test plan: In section 2.2 you should identify strengths and weaknesses of the final way you carried out testing. Often you will need to make observations of the behaviour of the code that involve "instrumenting" the code (e.g., with assert statements or by output to a diagnostic stream).

2.3. Instrumentation of the code: You should observe how much instrumentation you implemented and justify this (so if you have done none you should justify this by pointing out it was unnecessary for your requirements). For the OrderBot you might need to add diagnostic output to enable tracking of orders to ensure none are lost.

2.4. Evaluation of the instrumentation: You should evaluate how adequate you think the instrumentation is. For example, in the OrderBot you might decide that the order tracking should have been timestamped and can justify that this is a deficiency.

3. Apply a wide variety of testing techniques and compute test coverage and yield according to a variety of criteria.

3.1. Range of techniques: In this section for OrderBot we would first argue that a selection of functional tests had been implemented and a range of quality attributes tested and argue that these are adequate to give assurance that the requirements have been met. For example, we may have used some sort of stress testing to check it can deal with a particular level of demand.

3.2. Evaluation criteria for the adequacy of the testing: Here you would justify that a proposed strategy of evaluation is adequate. For example, that, amongst other things, a code coverage measures gives us reasonable confidence that the OrderBot parser is acceptable in terms of correctness and resilience to error.

3.3. Results of testing: This points to the results of the test and checks they are presented in a way that indicates they are comprehensive and easily interpretable.

3.4. Evaluation of the results: Here you should provide the evaluation results using the proposed techniques. Your portfolio would assess how complete these results and evaluations are.

4. Evaluate the limitations of a given testing process, using statistical methods where appropriate, and summarise outcomes. This section should be a short discussion of the testing process identifying deficiencies, reviewing the achieved levels of test, how these levels might be increased and how much confidence the testing gives you in the software. For the OrderBot example this might identify that it was impossible adequately to test availability because these was no good characterisation of the operation of the system. Perhaps using mutation testing could provide an estimate of the number of residual faults.

   4.1. Identifying gaps and omission in the testing process

   4.2. Identifying target coverage/performance levels for the different testing procedures 4.3. Discussing how the testing carried out compares with the target levels 4.4. Discussion of what would be necessary to achieve the target levels.

5. Conduct reviews and inspections and design and implement automated testing processes. This section should provide pointers and a brief discussion to the review procedures you adopted along with some commentary on their adequacy. 5.2-5.4 should describe the design of a CI pipeline you have implemented or would have implemented had time been available and assess the extent to which the proposed automated testing would identify issues in your code and some a description of what evidence would be necessary to demonstrate your proposed CI pipeline would operate as expected.

   5.1. Identify and apply review criteria to selected parts of the code and identify issues in the code.

   5.2. Construct an appropriate CI pipeline for the software

   5.3. Automate some aspects of the testing

   5.4. Demonstrate the CI pipeline functions as expected.