# Summative Assignment

| | |
|---|---|
| **Module code and title** | COMP2281 Software Engineering (Team work) |
| **Academic year** | 2025-26 |
| **Coursework title** | Technical Report |
| **Coursework credits** | 6 credits |
| **% of module's final mark** | 30% |
| **Lecturer** | Gokberk Kabacaoglu & Wanqing Tu |
| **Submission date*** | Thursday, March 12, 2026 14:00 |
| **Estimated hours of work** | 12 hours |
| **Submission method** | Ultra |

| | |
|---|---|
| **Additional coursework files** | *None* |
| **Required submission items and formats** | *Technical Report document in PDF format.*<br><br>*You must include the following information the filename: GroupNN(where NN = your group number, zero-padded e.g. Group 1 ->Group01, Group 21 -> Group21, etc.)* |

* This is the deadline for all submissions except where an approved extension is in place.

Late submissions received within 5 working days of the deadline will be capped at 40%.
Late submissions received later than 5 days after the deadline will receive a mark of 0.
It is your responsibility to check that your submission has uploaded successfully and obtain a submission receipt.

Your work must be done by yourself (or your group, if there is an assigned groupwork component) and comply with the university rules about plagiarism and collusion. Students suspected of plagiarism, either of published or unpublished sources, including the work of other students, or of collusion will be dealt with according to University guidelines (https://www.dur.ac.uk/learningandteaching.handbook/6/2/4/).

# Software Engineering (COMP2281) 2025/26

# Technical Report

## Overview

The Technical Report is to be submitted in pdf format via Ultra by ***2pm on 12th March 2025***. We appreciate it can be difficult to submit all the code of your final working solution in a way that makes it possible for staff to run and mark. If your final product is accessible online you should provide an additional text file containing the URL with your submission or outline very clearly at the very beginning of the Technical Report where the system can be found and provide the link, if applicable. Any aspects of your system not accessible online should be zipped and submitted through Ultra too. To successfully mark your Technical Report, staff will need accounts with full access to all parts of your system, which can then be later deleted as required.

## Content

The Technical Report contains two major sub-sections:

1) Documentation
2) Test report

The guidance given below is split into these two sections, please read each carefully before starting the report.

## Form and Content

The Technical Report should be a maximum of 30 A4 pages, using 11-point font, single-spaced, with 2 cm margins on all sides. This includes tables and figures. The length should be appropriate to the complexity of the project.

# Documentation

## Report Structure and Mark Scheme

The quality of the documentation section will be evaluated based on the **clarity, thoroughness**, and **rigour** in addressing the following two key aspects:

- **How the system has been developed**
- **How the system should be used by the client and other stakeholders, including end-users.**

For the first aspect, the technical descriptions should be detailed and self-contained, enabling other developers, not involved in the project, to continue development independently.

For the second aspect, the user instructions should cover all system functionalities and features, which were co-created with the client/stakeholders during the early project phase

(as documented in "Requirements Specification"), tested (as documented in "Test Plan"), and refined throughout development.

The final product should include the submitted code, which must align with the Technical Report. Additionally, the report should address important issues such as system maintenance, future development plans, and the assessment of the system's potential ethical and societal impacts.

Below is the proposed structure for the Documentation Section of the Technical Report. It is strongly recommended that you use this structure and the numbered headings provided to organize your document. You may adapt the structure as needed to address the specific characteristics of your project.

**********

**Cover**

You must include a single cover page indicating:

- The title of the document (i.e. “Technical Report for <Project Name>”)
- The authors of the document (i.e. full names and CIS usernames for each group member)
- The group number of the software engineering team
- The date the document was prepared.

## Section 1: Introduction (subtotal: 20%)

1.1 Present a clear summary of the project to contextualise its main motivation and goals (5%)

1.2 Provide usable access to the developed system, along with clear instructions on how to set it up and run it (5%)

1.3 Present the status of each behavioural requirement in a table. Provide the code (e.g., BR1.2 is “behavioural requirement for feature 1, scenario 2”) and a succinct description (no need to restate the entire user story). Indicate whether the requirement remains unchanged or has been modified (if modified, provide the updated description), and explain the extent to which it is met or not met, with justification if applicable. (10%)

## Section 2: Technical Development (subtotal: 40%)

2.1 Clearly describe the source materials that form the basis for the conceptualisation and development of the system. (5%)

2.2 Provide a clear and appropriately detailed technical description of how each system functionality was developed. Include, where relevant, some or all of the following aspects: (30%):

- A high-level overview of the system architecture.
- The design principles and patterns used.
- The technologies and platforms used (e.g., programming languages, databases, frameworks).
- Relevant diagrams (e.g., UML diagrams, class diagrams, sequence diagrams) to illustrate system design.

- The software development process followed by the team.
- The role of the behaviour-driven development (BDD) approach in the implementation phase.

2.3 Clearly describe how the system's usability and user experience aspects were addressed, providing an appropriate level of detail. (5%).

**Section 3: Use Instructions (subtotal: 20%)**

3.1 Installation: Describe system requirements, including the minimum and recommended hardware requirements (e.g., CPU, RAM, and storage) and operating systems. Provide step-by-step instructions for installing and configuring the software. (5%)

3.2 Deployment: Explain how to deploy the system on a local machine and how to set up the database. If relevant, include instructions for setting up virtual machines, containers or cloud functions. (5%)

3.3 Launching: Provide instructions on creating a user account or logging into the system. Explain different user roles (if applicable). Guide users through any first-time setup steps, such as creating admin accounts or configuring user settings. (5%)

3.4 Troubleshooting: List common error messages and solutions for resolving them. Explain where users can find logs or diagnostic information to troubleshoot issues. (5%)

**Section 4: Maintenance and Implications (subtotal: 20%)**

4.1 Provide useful and usable information how the system can be maintained (5%).

4.2 Provide useful and usable information how the system's possible future development can be implemented (5%).

4.3 Describe potential ethical and societal impacts of the system in its current and possible future status (10%).

# Test Report

The deliverable for this element of the project is a report that describes your testing activities and outcomes. Note that your report should use a vocabulary that is consistent with the one provided in the lectures.

## What you should test

Your report should cover the following forms of testing: *unit* testing, *system* testing and *user acceptance* testing (UAT). If the architecture of your application involves a substantial element of module integration, then you should provide a discussion of how you are managing the process of integration testing.

This section of the report should have the following structure:

## What has been tested (20%)

The first section of the report should identify all of the items being tested for both unit and system testing, and the test oracles that will be used for each one.

## Test Cases (50%)

The second section of your report should provide a minimum of four (4) and a maximum of six (6) sample test cases for each of the forms of testing being used (including integration testing where appropriate). These should be presented in a tabular form, and an example of such a form being used for a test case for the software in a bank ATM is presented below. You can adapt this table as necessary.

| | |
|---|---|
| Test Case ID | sys_test-02 |
| Description of test | Application successfully reads a bank card |
| Related requirement document details | <user story 1> |
| Pre-requisites for test | System is operational and not currently servicing a request.<br>Bank card is available to be inserted. |
| Test procedure | 1. Insert readable card<br>2. System responds with accept or reject message |
| Test material used | Bank card |
| Expected result (test oracle) | Card is accepted and system asks for PIN to be typed in |
| Comments | None |
| Created by | SJ |
| Test environment(s) | Windows 10 |

For the example above, there would need to be a similar test performed with an invalid card.

For each such plan there should be a separate table of the results, and whether or not the test was passed. If a test fails, you should note whether or not the failure is of high or low severity. Where regression testing has been performed, you should provide the test results for each test cycle, indicating whether or not it succeeded for each testing iteration.

Test oracles for system and unit testing should normally be provided by using the requirements and design specifications. For UAT, they will normally be provided directly by the end-users, undertaking the task of applying user stories.

The choice of test cases should be suitably representative, and marks will be deducted for unnecessary repetition or inadequate coverage. Where appropriate, identify equivalence classes that should be used for a particular test.

## Testing context (20%)

This part of your report should indicate whether or not the tests need to use any specific tools or environment. For example, if a test requires use of a web browser, you should indicate which ones are being used (and why). The same applies to operating systems.

You may want to use a simple scale of test failure severity to distinguish between faults that need to be urgently addressed (the system crashes or generates wrong outcomes) and those that are inconvenient. (See the examples in the first lecture on testing.)

You should give reasons for your choices.

## Writing good test cases (10%)

Being professional in your submitted work is expected, use clear formatting like bold text, headings, and bullet points to make your document easy to read. Start with a table of contents so the reader

can easily navigate your work and know what is expected. Provide a short summary of the project at the beginning to explain its purpose and scope. Include a summary of what will be presented in the test report, outlining the key findings and results. Finally, be concise and organized to demonstrate your professionalism throughout.

A test case should be expressed clearly and unambiguously using active voice where possible to emphasise what has been done. Any references to other parts of an application should be clearly referenced. You should also make clear what form of test oracle is used.

Test cases need to be as atomic as possible, so that it is clear what is being tested and what the result should be. As in the example above, test cases should cover both successful use (with a valid bank card) and unsuccessful use (with an invalid bank card) wherever this is appropriate.

For unit testing, you may want to identify appropriate equivalence classes to help determine how many tests are needed for each unit.