A busy mid-sized restaurant needs a new management system to address several key requirements. The restaurant has been using outdated methods and is facing issues with efficiency and accuracy. The new system should simplify and improve various aspects of restaurant management.

Firstly, the system must handle reservations efficiently. Customers should be able to book tables online, view real-time availability, and receive confirmation and reminders. Each reservation will be tracked with attributes such as Reservation ID, Customer Name, Table Number, Date and Time, Number of Guests, and Status. Restaurant Staff will need a clear, organized view of all upcoming reservations to manage seating and customer flow effectively.

Secondly, the system needs to streamline order management. Waitstaff should be able to input orders directly into the system from tablets or terminals, which will include attributes such as Order ID, Table Number, Items Ordered, Quantity, and Special Requests. Kitchen Staff will receive these orders in real time, and the system should track order progress, providing updates on the status of each order to notify waitstaff when orders are ready.

Another critical requirement is inventory management. The system should monitor stock levels in real time, with attributes including Item ID, Item Name, Quantity on Hand, Reorder Level, and Supplier Information. It should alert Inventory Staff when items are running low and facilitate easy reordering. The system should also provide reports on inventory usage to help with planning and reducing waste.

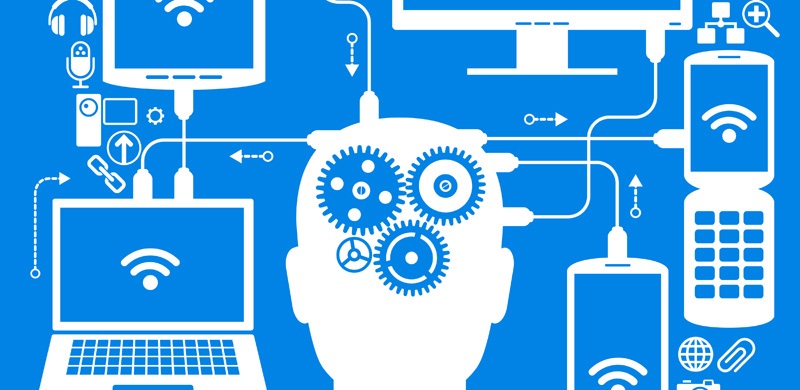
The system must also support staff scheduling. Managers should be able to create and manage staff schedules, with attributes such as Staff ID, Name, Position, Shift Times, and Hours Worked. The system should handle time-off requests and allow Staff Members to view their schedules, request changes, and check their hours.

Additionally, the system should integrate with existing hardware, such as point-of-sale systems, and be secure to protect sensitive customer and business data. It should be user-friendly and accessible, requiring minimal training for Staff to use effectively.

Finally, the system should be reliable, with minimal downtime and strong Customer Support available to address any issues that arise. By addressing these requirements, the new management system will help the restaurant improve its operational efficiency, reduce errors, and enhance the overall dining experience for Customers.

Software Engineering

5CM505

****

Software Engineering Assessment Brief

Wajahat Ali Khan

# Module Leader

* Wajahat Ali Khan
* [w.khan@derby.ac.uk](mailto:w.khan@derby.ac.uk)
* Office Hours: Tuesday – 1400-1600

# Key dates and details

| Assessment Detail | Assessment Information |
| --- | --- |
| Assessment Type: | Software Engineering Portfolio (Group Work) |
| Assessment Weighting: | 100% |
| Word count/Length: | Requirement Analysis: 400~500  Design: 400~500  Reflective Report: 600~700 words |
| Learning Outcomes: | 1. Demonstrate comprehensive understanding of the field of software engineering, including its foundations, key concepts, processes, and theories. 2. Apply software engineering methodologies to the process of managing a software project to successful completion within given resource constraints. |
| Submission Method: | Electronically via Blackboard Learn |
| Submission Date: | Requirement Analysis - 11:59 UK time, Dec 16th, 2024  Design, Reflective Report - 11:59 UK time, May 9th, 2025  Demos (Implementation) – In the week from 12th May to 16th May, 2025 |
| Provisional Feedback Release Date: | After 15 working days |

# Description of the assessment

The students will analyse, design, and implement a software system designed to support an enterprise scenario using software engineering techniques, tools, technologies, and best practices.

The project consists of 4 parts: 1) Requirement Analysis, 2) Design, 3) Implementation, and 4) Reflective Report. Students will design and implement the software system.

This coursework assesses learning outcomes, one and two (LO1 and LO2):

LO1: Demonstrate comprehensive understanding of the field of software engineering, including its foundations, key concepts, processes, and theories.

LO2: Apply software engineering methodologies to the process of managing a software project to successful completion within given resource constraints.

# Relationship to Programme Assessment Strategy

Level 4 modules – 4CM508 (Computer Systems, Data Structures, and Data Management), 4CM523 (Programming); The skillset students develop in these modules will be applied during the design and implementation phases of the Software Engineering module assessment.

Level 5 module – 5CM506 (Data Driven Systems); The skillset students develop will be applied in the design and implementation of databases during the design and implementation phases of Software Engineering module assessment.

The skillset students develop in 5CM505 Software Engineering module can be applied in the Level 5 module, 5CM519 (Team Project), and Level 6 module, 6CM995 (Individual Project).

# Attributes and Skills

The attributes and skills that this assessment is targeting to develop among the students are as follows:

|  | Skills | Links to useful resources |
| --- | --- | --- |
| ☐ | Critical thinking | https://libguides.derby.ac.uk/critical-reading |
| ☐ | Communication | [Communication within Teams](https://www.linkedin.com/learning-login/share?account=236084642&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fcommunication-within-teams%3Ftrk%3Dshare_ent_url%26shareId%3DhMWI%252Fs48Q%252BSRVl%252BPTmGi3g%253D%253D) |
| ☐ | Collaboration | [Essentials of Team Collaboration](https://www.linkedin.com/learning-login/share?account=236084642&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fessentials-of-team-collaboration%3Ftrk%3Dshare_ent_url%26shareId%3DOSFyRcm1QuOY8DDx%252FkkC2Q%253D%253D) |
| ☐ | Creative problem solving | [Problem Solving Techniques](https://www.linkedin.com/learning-login/share?account=236084642&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fproblem-solving-techniques%3Ftrk%3Dshare_ent_url%26shareId%3D7q%252BFOqrvS9GDInxOL%252BtjgQ%253D%253D) |

# Assessment Content

This coursework requires you to complete a software engineering project on a specific scenario which will already be uploaded. The project consists of four main parts:

1. **Requirements Analysis** –

*Core Artifacts – Functional Requirements, Non-Functional Requirements, Use Cases*

A detailed requirements analysis using the holistic method or FURPS, systemic textual analysis, viewpoint analysis, use cases, MoSCoW analysis, object-oriented analysis of the problem, function flow diagrams, context diagrams, standard process models or custom process model, check-off list (showing that the technical requirements meet the characteristics of good requirements). Use identifiers for establishing connections such e.g. FR1, UC1.

The focus should primarily be on the model approach, utilizing the specified tools, techniques, or methods. These are excluded from the word count and has no limit, as it should be tailored to the scenario. Descriptions will be included where necessary with the word count of 400–500 words. Exceeding this limit will not incur a penalty, but only the specified word range will be assessed.

1. **Software Design –**

*Core Artifacts - UML diagrams, Design Patterns, Database Schema/ XML/ JSON/ Flat files or any other appropriate data model, Architectural Model, Wireframes*

Requirements to design transformation, tools usage for UML diagrams, justification for the use of design patterns, and criteria to select architectural model.

The focus should primarily be on the model approach. These are excluded from the word count and has no limit, as it should be tailored to the scenario. Descriptions will be included where necessary with the word count of 400–500 words. Exceeding this limit will not incur a penalty, but only the specified word range will be assessed.

1. **Implementation** –

***Core Artifacts – Source Code, Demo***

**Source code management evidence e.g. a list of GitHub commits, justification for use of a particular source code management tool, agreed coding conventions document, actual source code segments with annotations, evidence of use of design patterns, object-oriented principles, low coupling, high cohesion, refactoring, and code comments.**

1. **Reflective Report -**

This report is expected to be reflective in nature, documenting and critiquing your experiences in the project, and explaining how you applied good software engineering practices to your work. You should chronicle what worked well for your project and what did not work so well and why. You should also provide some commentary regarding what you would do differently if you were tasked with a similar project in the future, and you should justify your choices, summarising the main lessons learned from the Software Engineering approach that you took. The reflective report should consist of about 600~700 words. You should use citations where appropriate and include a reference section at the end of the report, if required.

The focus should primarily be on the model approach, utilizing the specified tools, techniques, or methods. These are excluded from the word count and has no limit, as it should be tailored to the scenario. Descriptions will be included where necessary with the word count of 400–500 words. Exceeding this limit will not incur a penalty, but only the specified word range will be assessed.

1. Start from the fully dressed use case
   1. Spend much time on them (refine them) which will save you time in the sequence diagram
   2. The better more comprehensive sequence diagram will help into class diagaram
   3. Class diagaram results in many partial class diagrams

* Other UML Diagrams
  + Use Case Diagram
  + State Machine Diagram
  + Activity Diagram
* UML Behaviour Diagrams
  + Sequence Diagram
  + Communication Diagram
* UML Structure Diagrams
  + Class diagram
  + Component diagram
  + Deployment Diagram

All components of this coursework assignment MUST be submitted electronically via Course Resources (Blackboard Learn) by the due date and time:

* **Software Engineering project** to be submitted in two parts with Requirement Analysis submitted by Dec 16th, 2024 (Just before Noon) via Course Resources. Similarly, Design and Reflective Report documents should be submitted by May 09th 2025. The Demos of the implementation will take place in the week from 12th May to 16th May, 2025 (Just before Noon) via Course Resources. Submissions must be in either MS Word or PDF format. Below are the details for each component.
* Requirement Analysis Document – Try to use the model approach and provide documentation where necessary.
* Design Document - – Try to use the model approach and provide documentation where necessary.
* Implementation Document – Evidence of the source code with the requirements specified.
* Reflective Report – about 600 to 700 words.

**Note:** Submissions for the software engineering project in formats other than MS Word or PDF (such as rar, zip folders, etc.) will not be accepted, resulting in automatic failure of this assessment. It is imperative to submit all parts/components, as failure to do so will lead to a 25% reduction in the final grade for each missing component.

# Assessment Rubric

The assessment rubric on the next page shows the complete criteria of the CW and how you will be assessed. We will explore the content of the rubric together in Week 1.

When the assessment is returned you will receive a digital version of the rubric showing how you performed against each criterion. You will also receive short individual written feedback that highlights both the strengths of the work and your key areas for development. There will be an opportunity to discuss these with your personal academic tutor.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Criteria | Fail (0-34) | NF (35-39) | 3rd (40-49) | 2:2 (50-59) | 2:1 (60-69) | 1st (70-100) |
| **Requirement Analysis**  **25%** | The requirement analysis does not include most of the requirements from the scenario.  The core artifacts for requirement analysis are incomplete or incorrect. | The requirement analysis does not include the core requirements from the scenario.  The core artifacts have many missing connections.  Requirements doesn’t meet the characteristics of good requirements. | The requirement analysis includes the critical core requirements from the scenario.  The core artifacts demonstrate some connections but have others missing.  The tools and techniques for requirements are used and meet some of the characteristics of good requirements. | The requirement analysis includes most of the core requirements from the scenario.  The core artifacts demonstrate most of the connections among them.  The tools and techniques for requirements are used and many of the characteristics of good requirements are fulfilled. | The requirement analysis includes all the core requirements from the scenario.  The core artifacts demonstrate proper connections.  The tools, techniques and standard process models for requirements used follows all the characteristics of good requirements. | The requirement analysis includes all the core requirements from the scenario.  The core artifacts demonstrate proper connections.  The tools, techniques and standard process models for requirements used follows all the characteristics of good requirements.  The process model has introduced custom process activities that specially target the requirements of the given scenario. |
| **Design**  **25%** | No knowledge of requirements to design transformation.  The core artifacts (excluding design patterns) are incomplete, incorrect representations, and inappropriate design. | The transformation from requirements to design is not clear.  Minimal effort on the core artifacts (excluding design patterns) with many missing connections.  The core artifacts (excluding design patterns) are mostly incomplete, incorrect representations, and inappropriate design. | The transformation process correctly followed from requirements to design.  The core artifacts (excluding design patterns) demonstrate some connections but have others missing.  The core artifacts (excluding design patterns) are somewhat complete, correct with appropriate design.  The tools and techniques used to meet some of the requirements. | The transformation process correctly followed from requirements to design.  The core artifacts (excluding design patterns) demonstrate most of the connections among them.  The core artifacts (excluding design patterns) are mostly complete, correct with appropriate design.  The tools and techniques used meet most of the requirements. | The transformation process correctly followed from requirements to design.  The core artifacts demonstrate proper connections among them.  The core artifacts are properly complete, correct with appropriate design.  The tools and techniques used meet all the requirements. | The transformation process correctly followed from requirements to design.  The core artifacts demonstrate proper connections among them.  The core artifacts are properly complete, correct with appropriate design.  The tools and techniques used meet all the requirements.  The design to the solution has been mapped to suitable architectural patterns.  For 80+ grade, the process model has introduced custom process activities that specially target the requirements transformation into a design that resolve specific problems for the given scenario. |
| **Implementation**  **25%** | The implementation has no concrete connection with the scenario.  No demo provided or is not meaningful. | Very few aspects identified in the design phase are implemented.  Poor code organization and requires significant improvement.  Demo doesn’t demonstrate understanding of the software engineering principles applied for the scenario. | Some aspects identified in the design phase are implemented.  Code is very challenging to understand because the structure is confusing, affecting readability.  Demo demonstrate limited understanding of the software engineering principles applied for the scenario. | Implementation is mapped with many parts of the design.  Code has adequate organization but needs some improvement.  Demo present some understanding of the software engineering principles for the scenario. | The transformation from the design to implementation is done for most parts of the design.  Code is properly written with adequate structure and proper readability.  Demo demonstrate good understanding of the software engineering principles applied for the scenario. | The transformation from the design to implementation is properly done.  Code is exceptionally well-organized, adhering to best practices and design principles.  Code demonstrates foresight in designing for scalability, anticipating future growth and changes.  Demo demonstrate a deep and excellent understanding of the software engineering principles by showcasing key features impressively.  For higher marks (80+), Thorough and comprehensive testing strategy, covering unit, integration, and system testing.  Effective use of automated testing tools, contributing to a robust testing framework. |
| **Reflective Report**  **25%** | The reflective report lacks a meaningful ability to document and analyse experiences in the project, indicating a deficient understanding of good software engineering practices. | The reflective report demonstrates a minimal ability to document and analyse experiences in the project, reflecting a limited understanding of good software engineering practices. | The reflective report presents a satisfactory ability to document and analyse experiences in the project, reflecting a basic understanding of good software engineering practices. | The reflective report showcases a basic ability to document and analyse experiences in the project, demonstrating a satisfactory understanding of good software engineering practices. | The reflective report exhibits a solid ability to document and analyse experiences in the project, displaying a good understanding of good software engineering practices. | The reflective report demonstrates an exceptional ability to critically document and analyse experiences in the project, showcasing a profound understanding of good software engineering practices.  For 80+ grade, provide details about the possible deployment plan and maintenance aspects of the software. |

# Anonymous Marking

You must submit your work using your **student number** to identify yourself, not your name. You must not use your name in the text of the work at any point. When you submit your work in Turnitin you must submit your student number within the assignment document and in the Submission title field in Blackboard. For example, if your student number is 100123456, you should name your file 5CM505-100123456.zip when submitting it on Blackboard.

# Assessment Regulations

The [University’s regulations, policies and procedures](https://www.derby.ac.uk/about/academic-regulations/) for students define the framework within which teaching and assessment are conducted. Please make sure you are familiar with these regulations, policies and procedures.

# Formative

There will be a formative opportunity to support this coursework. You will be provided with formative feedback during the tutorial classes towards the end of the module. This will help you practice and prepare for the final submission.