Nottingham Trent University

School of Science and Technology

COURSEWORK ASSESSMENT SPECIFICATION

Module Code	SOFT20091
Module Title	Software Design and Implementation
Module Leader	Vishal Thakor
Module Team	Peter Blanchfield Michael Gibbs Amin Safaei Doratha Vinkemeier Pablo Lopez-Custodio Vishal Thakor
Coursework Title	Timetabling System for Nottingham Trent University
Module Learning outcomes assessed	K1: Design and implement software that is robust, portable, and readily maintainable. K3: Select and apply a range of analysis and design techniques to software engineering problems. K4: Effectively communicate and document a software design using UML. S1: Implement appropriate algorithms and data structures and select appropriate containers from the Object Oriented libraries, to develop a system. S2: Communicate effectively via reports. S3: Evaluate requirements and potential solutions. S4: Work in a task/deliverable-oriented way.
Apprentice Learner KSBs evidenced	NA
Contribution to module (include contribution to element if appropriate)	80%
Date work set	03 October 2024
Deadline for submissions	Final Report + Demo video + Final Product **23 April 2025 at 02:30 pm**
	Online Demo via Team week commencing **23 April 2025** and **28 April 2025**
Method of Submission	Dropbox via NOW
Deadline for Feedback	Informative Feedback - During lab After the Summer Exam Period
Method of Feedback	NOW
Previous coursework that may support this submission.	-

Nottingham Trent University

Late submissions and NECs
Work handed in up to five working days late will be given a maximum Grade of Low Third/ Pass whilst work that arrives more than five working days will be given a mark of Zero.

Please note if you are repeating the work and are capped in your grade, you will receive a Zero grade if the work is submitted at all late.

Work will only be accepted beyond the five working day deadline if satisfactory evidence, for example, an NEC is provided. https://www.ntu.ac.uk/studenthub/my-course/student-handbook/submit-a-notification-of-extenuating-circumstances

Presentations/Demo
As a student you may have a statement of access concerning aspects of presentations such as eye contact or reading from notes. While these aspects of presentation will be part of the GBA grid used to assess work, it will not affect your grade.

Breaches of Academic Integrity
To ensure that you are not accused of any breaches of Academic Integrity, look at the NOW page Plagiarism and Academic Integrity at NTU. for guidance.

The University views **plagiarism and collusion** as serious academic irregularities and there are a number of different penalties which may be applied to such offences. The Quality handbook a section on such breaches, which outlines the penalties and states that **plagiarism** includes:
Presenting someone else's ideas as your own (**including text, graph, diagrams, videos etc.**) in a substantial proportion of your work, with or without consent, by incorporating it into assessment without full acknowledgement, including:

Self-plagiarism: reproducing or representing work for assessment without proper attribution and attempting to gain credit for this work where credit has already been received.

Paraphrasing: rephrasing a source's ideas without proper attribution

Mosaic plagiarism/patchworking: weaving phrases and text from several sources into your own work; and/or adjusting sentences without quotation marks or attribution.

Source-based plagiarism: providing inaccurate or incomplete information about sources such that they cannot be found.

Computer code plagiarism: copying or adapting source code without permission from and attribution to the original creator.

Whereas **collusion** includes:

working with other students on an assessment meant for individual submission Sharing your work with other students enabling them to plagiarise your ideas.

Please remember submitting portions of work already assessed for the same learning outcomes is **Self-Plagiarism** and is also a serious academic irregularity.
Penalties for Breaches of academic integrity range from capped or zero grades for elements of modules, to dismissal from the course and termination of studies.

Chat GPT and other AI-powered language models.
It is important to note when using any AI platform that they generate the most common responses to questions, not necessarily the correct ones. They also fabricate evidence.
The material they produce is not your own words. Assessments require you answer questions giving your own view and in your own words. The outputs from platforms such as Chat GPT do not provide that.

By presenting such material as your own words you are violating Academic Integrity policy, a matter that NTU takes very seriously.

The skills you develop during your time with us allow you to interrogate material and evaluate it, important skills in all careers. Generative AI does not allow you to develop these.
If you have utilised such platforms, you must retain any outputs from them to provide evidence your work is your own in the case of suspected breaches of Academic Integrity

I. ASSESSMENT REQUIREMENTS

- The assignment is **individual coursework.**
- Given the problem scenario in Section 2, students are required to write a report on the designed software and develop a software application.
- **The mark for this project will be mainly derived from a demonstration of the project which will include a question-and-answer session where your tutor will ask you about the design and structure of your project. Documentation will be used to help in this process. If you do not attend your demonstration, or, the project shows inadequate ownership of the presented code, your tutor may award a mark of zero.**
- Students **MUST** use the following convention for naming their folders and files: "SDI_Report_Nxxxxxx" (Nxxxxxx is the student number).
- Report and DOCUMENTATION **MUST** be in the PDF/MS Word format and submitted separately into Dropbox.
  - Software Project files (except the report) **MUST** be submitted in a single zip file
  - The zipped file **MUST** have the following structure:
    - SRC containing all the code files (mandatory);
    - TESTS with the tests' source code and README file explaining how to run the tests;
    - BUILD containing the build files

II. ASSESSMENT OVERVIEW

Title: Timetabling System for Nottingham Trent University

Background

Nottingham Trent University (NTU) requires a timetabling system similar to the one available on the NTU student dashboard. This system will facilitate the management and viewing of academic schedules for both students and administrators. The purpose of this assignment is to design and implement a simplified version of the NTU timetabling system.

You are required to develop a **C++ console application** that provides functionality for two types of users: Admin and Student. Admins should be able to manage the timetables, while students will use the system to view and search their schedules.

Core Requirements

Admin User Capabilities (Core Features):

1. Add, edit, and delete modules.
2. Add, edit, and delete student groups.
3. Define and manage types of sessions (e.g., lectures, labs).
4. Register students and assign them to groups.
5. Register lecturers and assign them to modules and sessions.
6. Add, edit, and delete rooms (lecture halls, labs, etc.).
7. Create and update timetables by week number for each academic year (Week 1 to Week 53).
8. Search for timetable conflicts **(Advanced Feature)**.

Student User Capabilities (Core Features):

1. View timetables.
2. Search timetables by week number, module name, room, and lecturer **(Advanced Feature)**.
3. Export the timetable to CSV/Excel file **(Advanced Feature)**.

Scenario Description

Admin User Scenario – John, the University's Scheduler:

John is responsible for setting up the timetable for the new academic year. He logs into the timetabling system with his admin credentials and performs the following tasks:

- **Adding Modules:** John adds modules such as "Introduction to Programming," "Data Structures," and "Database Systems."
- **Creating Student Groups:** He creates student groups "Group A" and "Group B."
- **Defining Session Types:** John defines session types like "Lecture" and "Lab."
- **Registering Students and Lecturers:** John registers students and assigns them to the created groups. For example, Alice is assigned to "Group A," and Bob is assigned to "Group B." He also assigns lecturers, such as Dr Smith, who teaches "Introduction to Programming."
- **Adding Rooms:** John adds rooms such as "John Clare Lecture Theatre 006" for lectures and "MAE 202" for lab sessions.
- **Creating the Timetable:** John creates the timetable by scheduling lectures and lab sessions. For example, "Introduction to Programming" with Dr Smith is scheduled in "MAE 202" on Monday at 10 AM for "Group A." Similarly, he schedules a lab session for "Data Structures" in "MAE 202" on Wednesday at 2 PM for "Group B."

Student User Scenario – Alice, a First-Year Student:

Alice logs into the system with her student credentials. She uses the following features:

- **Viewing the Timetable:** Alice views her timetable for Week 1, which includes the "Introduction to Programming" lecture on Monday at 10 AM in "MAE 202."
- **Searching the Timetable:** Alice searches for her "Data Structures" sessions and finds the lab session scheduled for Wednesday at 2 PM in "MAE 202."

Project Requirements

You are required to develop the following as part of your project deliverables:

1. Working Prototype

- A console application developed using the C++ programming language.
- The application should have two interfaces:
    - **Admin Console Interface**: For managing modules, groups, sessions, students, lecturers, rooms, and timetables.
    - **Student Console Interface**: For viewing and searching timetables.

2. Testing

- Test cases covering each functionality.
- Unit tests for individual features.
- User acceptance testing with sample data for both admin and student users.

3. Documentation

- **Admin User Manual**: Instructions on how to manage the timetable (add, edit, delete entries, and resolve conflicts).
- **Student User Manual**: Guide on how to view and search timetables.
- **Technical Documentation**: Explanation of the system architecture, database schema, code structure, and implementation details.

Development Tools & Frameworks

You may use the following tools and frameworks during development:

- **Operating Systems:** Windows, Linux, or macOS.
- **IDE:** Any (e.g., Visual Studio, Code::Blocks).
- **UML Design Tools:** Any suitable tool.
- **Git Repository:** *https://olympus.ntu.ac.uk/*
  - Ensure proper use of Git for version control; the absence of a GitHub repository may impact your score (see assessment criteria).

Implementation Requirements

Your implementation must meet the following specifications:

- **C++ Programming Language**: The entire application must be developed using C++.
- **Standard Template Library (STL)**: Utilize STL for data structures and algorithms (e.g., containers, sorting, searching).
- **Object-Oriented Programming (OOP) Concepts**: Use classes, inheritance, and other OOP principles as needed.
- **Error and Exception Handling**: Implement validation, error handling, and exceptions.
- **Data Validation**: Ensure user input is validated to prevent incorrect data entry.

Deliverables

1. Analysis and Design Report (Submit as a single PDF or MS Word document):

- List of functional and non-functional requirements.
- Project management documentation, including a task list and Gantt chart.
- Behavioural and structural design diagrams.
- Critical analysis of the software architecture (max. 1 page).
- Critical analysis of the design principles (max. 1 page).
- Explanation of non-self-explanatory parts of the implementation (e.g., design-to-implementation mapping, data structures, relevant classes, sorting, and searching) (max. 2 pages).

- Evaluation of non-functional requirements such as reusability, maintainability, performance, and usability (max. 1 page).
- User manual and testing plan (test case ID, scenario, steps, expected and actual results, pass/fail).

2. Implementation (Submit as a single zip file):

- The implementation must focus on the functional aspects of the system based on the scenario.
- Use OOP features such as inheritance and STL components for data storage, sorting, and searching.
- Submit all source code files and associated documentation.

III. ASSESSMENT CRITERIA (contextualised GBA Grid)

Your overall grade can be understood by referring to the below table.

Detailed Criteria Grid

Software Analysis and Design: 40%

Class/grade Assessment criteria	1st (Excellent)				2.1. (Very Good)			2.2. (Good)			Pass (Fair)			Fail (insufficient)			Zero
	***Exceptional First** 16	High First 15	Mid First 14	Low First 13	High 2.1. 12	Mid 2.1 11	Low 2.1 10	High 2.2 9	Mid 2.2. 8	Low 2.2. 7	High 3 6	Mid 3 5	Low 3 4	Marg Fail 3	Mid Fail 2	Low Fail 1	Zero 0
Behaviour Design 30% - Use case diagrams - Sequence diagrams - State diagrams	A thoroughly convincing design communicated well through the use of all model diagrams. Decisions made during design insight about all the requirements. Some use cases beyond the specs were defined.	An excellent design communicated well through the use of all model diagrams. Decisions made during design show insight about all the requirements			Reasonably good design but may require multiple iterations for implementation. Sufficient detail in use case, sequence and/or state flows and alternative flows added.			A number, though not all, requirements have been identified. Use of UML for identified cases, sequence and state. Some flow diagrams may be missing or incomplete.			Limited requirements have been identified. Use of UML for identified cases, sequence and state. Some flow diagrams may be missing or incomplete.			Major mismatch between requirements and design diagrams. Shows a lack of understanding of the required methodology. Lack of understanding of UML notations			
Structure and architecture design 70% - Class diagrams. - Cohesion and Coupling - Associations - Operations and attributes identified. - Use of UML notation. - Component Diagram - Use of Design principles	The diagrams demonstrate an exceptional understanding of the requirements and the problem. There is a critical evaluation of the key needs for additional functionality in the project. The report includes an outstanding critical analysis of the software architecture, design principles, and patterns utilized in the project.	The diagrams demonstrate an extensive understanding of the requirements and the problem. They are consistent with one another. The report includes a critical analysis of the software architecture, design principles, and patterns used in the project. There is an excellent evaluation of the fulfilment of non-functional requirements, such as reusability, maintainability, performance, and usability.			The diagrams demonstrate a strong ability to understand the requirements and are consistent with one another. There is an thorough understanding of the fulfilment of functional and non-functional requirements.			The diagrams generally communicate the results of the design. The decisions made demonstrate an understanding of most of the requirements, though some inconsistencies are evident.			The diagrams convey the results of the design and show an understanding of some requirements; however, several inconsistencies are evident.			Limited requirements have been identified. UML has been used for the identified use cases, sequence diagrams, and state diagrams. However, some flow diagrams may be missing or incomplete.			

NTU Nottingham Trent University

Software Implementation : 60%

Class/grade Assessment criteria	1st (Excellent)				2.1. (Very Good)			2.2. (Good)			3rd (Fair)			Fail (insufficient)			Zero
	***Exceptional First** 16	High First 15	Mid First 14	Low First 13	High 2.1. 12	Mid 2.1 11	Low 2.1 10	High 2.2 9	Mid 2.2. 8	Low 2.2. 7	High 3 6	Mid 3 5	Low 3 4	Marg Fail 3	Mid Fail 2	Low Fail 1	Zero 0
Functionality: 70%	**All core and advanced requirements have been fully and accurately implemented. Exceptional programming skills and high-quality code.**	**All core and advanced requirements have been completed, with the implementation closely aligned with the design document. The programming skills demonstrated are excellent, and the code is clean, reusable, and of high quality.**			**All core requirements have been completed correctly and without issues, with effective use of exception handling and the STL library.**			All core requirements have been completed without any issues, but the use of the STL library is missing.			**The core requirements have been completed with minor issues.**			**A small part of the program works, but the main functionality is not functioning correctly. The program does not meet the specifications and frequently encounters runtime errors.**			
Object-Oriented Programming: 20%	Exceptional object-oriented design that effectively utilizes most object-oriented programming concepts, binding together the data and the functions that manipulate it, while employing polymorphism.	Excellent object-oriented design and well-thought-out implementation, utilizing sophisticated constructs where appropriate. Demonstrates effort and learning well beyond the taught material. The program successfully achieves its aims or approaches very challenging objectives effectively.			Demonstrates a clear understanding of how to identify, illustrate, and describe all the classes in the program. All classes have been accurately identified and fully implemented.			Demonstrates a good understanding of object-oriented implementation and the ability to identify, illustrate, and describe all the classes in the program.			Sufficient understanding of object-oriented modelling is demonstrated; however, the implementation is incomplete. Many classes, including some key ones, are missing or have not been implemented correctly.			Marginal/insufficient understanding of object-oriented implementation.			
Testing - 10% - Test Plan - Results and screenshots	Exceptional testing strategy and effective use of testing tools.	Excellent testing strategy and effective use of testing to evaluate the functionality of the application, utilizing unit testing frameworks			Very good testing including most of the results/screenshots			Sufficient testing, with adequate results and screenshots included.			Some proof of testing, limited results/screenshots.			No testing documents or insufficient testing			

Nottingham Trent University

IV. Feedback Opportunities

Formative (Whilst you're working on the coursework)
You will be given the opportunity to book appointments to discuss the assessment

Summative (After you've submitted the coursework)
You will receive specific feedback regarding your coursework submission together with your awarded grade when it is returned to you. Clearly, feedback provided with your coursework is only for developmental purposes so that you can improve for the next assessment or subject-related module.

V. Referencing styles:

Referencing styles please use Harvard as detailed here
Guidance for presentations as detailed here and think about what lectures you have liked and why
Guide to planning your time here
General Reference guidance here

VI. Moderation

The Moderation Process
The grades awarded are considered by the module team to check for consistency and fairness across the cohort for the piece of work submitted.

VII. Aspects for Professional Development

The assignment has the benefit of allowing you to practise both design and implementation, as well as their transition in the defined scenarios. You will learn how to create and maintain technical documentation, how to use git based distributed version control system. All those technical skills are normally requested by Software Development Companies.