Prifysgol Wrecsam
Wrexham University

Module specification

When printed this becomes an uncontrolled document. Please access the Module Directory for the most up to date version by clicking on the following link: Module directory

Module Code	COM662
Module Title	Software Development and Optimisation
Level	6
Credit value	20
Faculty	FACE
HECoS Code	100956
Cost Code	GACP

Programmes in which module to be offered

Programme title	Is the module core or option for this programme
BSc (Hons) Software Engineering	Core
BSc (Hons) Software Engineering with Industrial Placement	Core

Pre-requisites

N/A

Breakdown of module hours

Learning and teaching hours	12 hrs
Placement tutor support	0 hrs
Supervised learning e.g. practical classes, workshops	12 hrs
Project supervision (level 6 projects and dissertation modules only)	0 hrs
Total active learning and teaching hours	**24** hrs
Placement / work based learning	0 hrs
Guided independent study	176 hrs
Module duration (total hours)	**200** hrs

For office use only	
Initial approval date	08/11/2023
With effect from date	Sept 2026
Date and details of revision	
Version number	1

Module aims

This module provides students programming skills in one or more languages commonly used in software development, enabling the creation of efficient, maintainable, and scalable code. Provide a comprehensive understanding of the software development lifecycle, from requirements analysis and design to implementation, testing, deployment, and maintenance. Various software optimization techniques, algorithms, and data structures, enabling them to improve the performance and efficiency of software applications. Develop students' abilities to identify, diagnose, and fix software defects and performance issues, using appropriate debugging and troubleshooting techniques and tools.

Module Learning Outcomes - at the end of this module, students will be able to:

1	Analyse a range of problems and produce designs and models for algorithmic solutions, with the focus on optimisation.
2	Conduct a critical evaluation of problems and solutions by analysing computational complexity.
3	Develop and implement computational solutions that showcase mastery in diverse data structures, algorithms, and programming techniques.

Assessment

Indicative Assessment Tasks:
This section outlines the type of assessment task the student will be expected to complete as part of the module. More details will be made available in the relevant academic year module handbook.

Another example of the coursework could also be optimizing a given algorithm or developing an optimized version of an existing algorithm. They should provide a detailed analysis of the original algorithm's complexity, identify areas for improvement, implement the optimizations, and compare the performance of the optimized algorithm with the original version. The coursework piece should include code implementation, documentation, and a report outlining the optimization process.

Assessment number	Learning Outcomes to be met	Type of assessment	Weighting (%)
1	1,2,3	Coursework	100%

Derogations

None

Learning and Teaching Strategies

In line with the Active Learning Framework, this module will be blended digitally with both a VLE and online community. Content will be available for students to access synchronously

and asynchronously and may indicatively include first and third-party tutorials and videos, supporting files, online activities any additional content that supports their learning.

As this module progresses, the strategies will change to best support a diverse learning environment. Initially, the module will start with a heavier reliance on engaging tutor-led lectures, demonstrations, and workshops to ensure that the students get the relevant threshold concepts. As the module continues experiential and peer learning strategies will be encouraged as the students' progress with their portfolio work.

Assessment will occur throughout the module to build student confidence and self-efficacy in relation to complex software development problems.

Indicative Syllabus Outline

Yearly content will be updated to represent the most appropriate content for current industry technologies, but a list of indicative topics could include:

- Software optimization concepts and importance
- Performance metrics and benchmarks
- Profiling and Analysis
  - Profiling tools and techniques
  - Performance analysis and bottleneck identification
  - Resource usage analysis (CPU, memory, I/O)
- Analysing algorithm complexity and efficiency
- Data Structures Optimization
  - Choosing efficient data structures for different scenarios
  - Optimizing data access and manipulation operations
- Code Optimization Techniques
  - Optimizing loops and control flow structures
  - Compiler optimizations and code transformation techniques
  - Eliminating unnecessary computations and redundancy
- Optimizing for Specific Platforms or Environments
  - Mobile application optimization
  - Web application optimization
  - Cloud computing and scalability considerations

Indicative Bibliography:

Please note the essential reads and other indicative reading are subject to annual review and update.

Essential Reads

N/A

Other indicative reading

K. Kalpeth, A. Johri, *Combining DataOps, MLOps and DevOps: Outperform Analytics and Software Development with Expert Practices on Process Optimization and Automation*, BPB Publications, 2022.
M. Fischetti, *Introduction to Mathematical Optimization,* Independently published, 2019.
A. Dymo, *Ruby Performance Optimization: Why Ruby Is Slow, and How to Fix It,* O'Reilly, 2015.
J. M. White, *Bandit Algorithms for Website Optimization: Developing, Deploying, and Debugging,* O'Reilly, 2012.
R. Gerber, *The Software Optimization Cookbook*, Intel Press, 2005.