



**De Montfort University,
Faculty of Computing
Engineering Media and
Niels Brock, Copenhagen
Business College**

**BSc (Hons)
Computer Science**

G41051

**Curriculum
2023/24**

The aim of the undergraduate degree, Computer Science, is to provide the students with modern approaches to software systems development. A mixture of both theory and practice is covered with an emphasis on "why" as well as "what". Students are encouraged to develop critical thinking and problem-solving skills. A practical element gives students the experience needed to develop software systems using modern languages and environments. An example of this is the final year project which gives students the opportunity to apply all stages of a software development method to produce a software system, guided by a project tutor. As well as all the technical aspects of the course, students learn about the structure of organisations, gaining an insight into the commercial context in which IT systems are commissioned and deployed. Students are also introduced to computing ethics and law.

The first year's syllabus focusses on basic concepts of Computer Science like Computer architecture, Operating systems, Computer networks, Programming, Security etc. The second year's syllabus focuses on more advanced concepts like Object-oriented software design and programming, Data structures and algorithms, Concurrent and parallel algorithms, Agile team project development etc. The third year is a specialisation year covering concepts like Big data and machine learning, Software design methods (including commercial, professional, ethical and legal considerations) and with a possibility to select an elective course covering "up-to-date" concepts like Functional Programming, Web Development, Fuzzy Logic and Data protection. The three years are organised in 4 courses each year, each running in seven-week blocks.

Upon completion of the undergraduate degree, students should be able to:

- Apply the theoretical knowledge and practical experience they have learnt to each phase of the software lifecycle.
- Be able to describe the role of computer technology within business organisations.
- Evaluate and recommend appropriate computer systems.
- Recommend an appropriate implementation strategy.
- Demonstrate knowledge and understanding of areas in computing at the forefront of the discipline.
- Appreciate the relative merits and limitations of different computing environments, paradigms and methodologies.

The subject specific aims are listed in the following module descriptions.

In addition, participants will also gain a number of cognitive skills. These include:

- Capacity for appreciation of the complexity of Computer Science
Critical evaluation and reflection of theoretical and practical issues
- Professional considerations related to both developing systems and implementing these in businesses
- Capacity for independent and self-managed learning
- Ability to draw reasoned conclusions
- Communication skills, be they in writing, as oral presentations etc.

The programme structure is set out in detail below.

Level 4 (Study Year 1)

Core modules:

Module code	Module title	Credit Value
CTEC1701N	Database Design and Implementation	30
CTEC1702N	Fundamental Concepts of Computer Science	30
CTEC1703N	Computer Programming	30
CTEC1704N	Operating Systems and Networks	30

Supplementary (but still mandatory) module:

Module code	Module title	Credit Value
NB001	Mandatory Academic Workshop	0

Please note that all level 4 modules must be passed before you may proceed to level 5

Level 5 (Study Year 2)

Core modules:

Module code	Module title	Credit Value
CTEC2710N	Object Oriented Design and Development	30
CTEC2711N	Data Structures and Algorithms	30
CTEC2712N	Web Application Development	30
CTEC2713N	Agile Development Team Project	30

Please note that all level 5 modules must be passed before you may proceed to level 6

Level 6 (Study Year 3)

Core modules:

Module code	Module title	Credit Value
CTEC3701N	Software Development: Methods and Standards	30
CTEC3702N	Big Data and Machine Learning	30
CTEC3451N	Development Project	30

Please note that the modules CTEC3701N and CTEC3702N must be passed before you hand in the development project (CTEC3451N)

Plus select one of the following:

Module code	Module title	Credit Value
CTEC3704N	Functional Programming	30
CTEC3705N	Advanced Web Development	30
IMAT3722N	Fuzzy Logic and Inference Systems	30
IMAT3711N	Privacy and Data Protection	30

Please note that the module CTEC2712N is a prerequisite for the module CTEC3705N

The modules are described in detail below

Module descriptions Level 4

Database Design and Implementation

Module Code: CTEC1701N **Credit Value:** 30 **Credit Level:** Academic Level 4

Module Description

Structured data, held in relational databases, accessed via SQL, supports the information storage requirements of many companies, organisations, and on-line businesses. In this module the student will learn the fundamentals of how to design the structure of data within a relational database, how to interact with data within the database, and how to protect the data within the database.

Topics include: The relational model, top-down modelling of business requirements, ER model, keys, relationships, traps, normalisation, SQL mapping schema to implementation via DDL, querying data using DML, integrity, transactions, access control and security, introduction to big data, semi-structured and unstructured data. Relevant elements of the Data Protection Act 2018 (DPA 2018) and the General Data Protection Regulations 2016 (GDPR) will be introduced, developing knowledge and understanding of the various 'rights and freedoms' that they provide. This will build to develop knowledge of a Data Protection Impact Assessment (DPIA), which is now a legal requirement whenever a proposed data processing need is perceived as 'risky' such as, inter alia, mass surveillance, use of sensitive data and automated decision making.

Learning outcomes

1. Derive a suitably normalised database design that reflects the business requirements of a typical corporate scenario. (Online Test 1 & Online Test 2)
2. Implement a database design via a Relational Database Management System addressing business use cases, information integrity and information security. (Online Test 1 & Online Test 2)
3. Acquire an understanding of big data and differentiate structured, semi-structured and unstructured data. (Online Test 2)
4. Demonstrate a sound understanding and ability to locate and use primary and secondary sources. (Report)
5. Identify and implement privacy tools in the design of database. (Report)

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't	Minimum Threshold Mark	Essential Component	Anon. Marking Code
Online Test 1			40				AM
Online Test 2			40				AM
Report			20				OPTO2

Anonymous marking exemption codes: OPTO1: Individually distinct work; OPTO2: Reflection on development of own work; OPTO3: Presentation; OPTO4: individually negotiated work; OPTO5: work placement/experience/assessment

Assessment Notes

The online phase tests will be anonymous and accessible. They will provide efficient & reliable marking, immediate feedback, and will be plagiarism limited. The tests will be semi-open book by allowing students to prepare a database design for a given scenario and become familiar with test data in advance of the test and then access and interpret their prepared design and the test data during the test to answer questions. Students are encouraged to work in small groups to prepare the design but must interpret and answer the test questions individually. The students apply their knowledge of the database language by dry running database commands against the test data which they can practice in advance from lab exercises. This more open approach to the test incorporates this coursework style element and leads to more

extended, structured, and responsible learning. To give this further support, formative coursework, scenarios, self-evaluation criteria and solutions are issued in advance of the test. The law related component will be assessed by means of a 1000 word written piece that comprises a business report to a senior manager in your organisation that presents a new DPIA. Students should strive to write an accurate, concise, and cogent Data Protection Impact Assessment that is based on a supplied scenario. One phase test may be re-attempted (optionally) within the module delivery period. If an optional second attempt is taken, then the higher mark of the two attempts for that phase test will be recorded.

Expected Methods of Delivery

Workshops will be used to introduce and demonstrate key practical and theoretical concepts.

Practical programming skill will be gained in regular laboratory sessions. Some sessions may be used for consolidation, revision, and to discuss solutions to practical problems.

Workshop	42 hours
Practical	20 hours
Seminar	4 hours
Self-directed study	76 hours
Consolidation	68 hours
Reading	30 hours
Assessment	60 hours – subsumes estimated time for producing deliverables as per the assessment table

Literature



Data analysis for database design

Book - by D. R. Howe - 2001 - Recommended

[VIEW ONLINE](#)



Introduction to SQL: mastering the relational database language

Book - by Rick F. van der Lans - c2007 - Recommended



An introduction to database systems

Book - by C. J. Date - 2004 - Optional



Database systems: a practical approach to design, implementation, and management

Book - by Thomas M. Connolly; Carolyn E. Begg - 2015 - Optional

[VIEW ONLINE](#)



The essence of databases

Book - by F. D. Rolland - c1998 - Optional

The two recommended books are expected to be the basic literature

Fundamental Concepts of Computer Science

Module Code: CTEC1702N **Credit Value:** 30 **Credit Level:** Academic Level 4

Module Description

This module introduces students to fundamental concepts in computer science in relevant areas of mathematics (including propositional logic, set notation, etc); software modelling; the software lifecycle; requirements capture; user interface design; and the foundations of ethical thinking. These topics can then be applied and further developed as students progress throughout the course.

The module starts by introducing mathematical structures that provide a basis for computer science including logic, set theory, probability, and statistics. More specifically, logic sets, tuples, relations, functions, probability, hypothesis generation and testing, and matrices are covered.

After this, students are taught about fundamental principles in areas of the software life cycle, including requirements capture and user interface design. Developing an understanding of requirements analysis will include how to formulate functional requirements, the various kinds of non-functional requirements, and usability requirements, as well as how to carry out interviews and questionnaires and formulate appropriate questions. Discussions of the software lifecycle include the range of different activities involved in a software development project and some different ways in which these activities can be sequenced, as well as the central importance of designing around user needs and capabilities and integrating user consultation and user testing into the development process.

Students then move onto developing their understanding in topics surrounding the foundations of ethical thinking. There is an introduction to ethical theories and concepts related to information systems, information security, software engineering, computer science and digital forensics. This includes examining the key issues affecting the development of future ICTs (privacy, security, etc). The material is complemented by understanding the responsibilities of the computer professional, including professional codes of conduct and codes of ethics (e.g. IEEE, ACM, BCS).

Learning outcomes

1. Construct and manipulate propositional logic statements. (Online Test)
2. Construct and manipulate set-theoretical mathematical objects. (Online Test)
3. Undertake probabilistic calculations and perform hypothesis testing. (Online Test)
4. Demonstrate an understanding of the different kinds of requirements software systems need to meet, and the ability to formulate appropriate requirements. (Portfolio)
5. Demonstrate an outline understanding of the software development lifecycle including the basic principles of user centered design and awareness of contrasting approaches to structuring the development process. (Portfolio)
6. Demonstrate understanding of key theories and current ethical concerns in information systems and computing, including professionalism and codes of ethics. (Case study analysis)

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't (X = Yes)	Minimum Threshold Mark	Essential Component (X = Yes)	Anon. Marking Code
Online Test 1	HRS	0.7	50				AM
Portfolio	HRS	12	30				OPTO2
Case Study	WRD	1000	20	X			AM

Anonymous marking exemption codes: OPTO1: Individually distinct work; OPTO2: Reflection on development of own work; OPTO3:

Assessment Notes

Online tests are appropriate for the maths work and enable an integrated assessment. Practice tests will be made available in advance so students can prepare fully for the type of assessment. One phase test may be re-attempted (optionally) within the module delivery period. If an optional second attempt is taken, then the higher mark of the two attempts for that phase test will be recorded. The assessment of students' understanding of essential aspects of the software development process will involve the production of design documents based on a case study, that involve the application of analysis and design skills. The assessment for the ethics material is in the form of a case study analysis of a current issue/case surrounding the associated topics delivered on the module.

Expected Methods of Delivery

Workshops will be used to introduce the main topics. To gain full advantage of this module students are expected to hone their skills and understanding by working through progressive exercises. The exercises range from drill to problem solving tasks. The exercises provide the basis of tutorial seminar and laboratory work. Students are expected to complete exercises in their own time. In seminars students receive feedback on their progress and engage in discussions on issues arising from the exercises.

Workshop	42 hours
Seminar	24 hours
Self-directed study	66 hours
Consolidation	58 hours
Reading	30 hours
Revision	20 hours
Assessment	60 hours – subsumes estimated time for producing deliverables as per the assessment table

Literature

There is an extensive reading list for this module because the ethics part includes many articles etc. Students will be guided through the list in the beginning of the module.

The mathematics part:

- Davis, G., & Pecar, B. (2013). Quantitative methods for decision making using Excel. Oxford University Press.
- Giannasi, F., & Low, R. (1995). Maths for computing and information technology. Longman.
- Lipschutz, S., & Lipson, M. (2007). Schaum's outline of discrete mathematics (3rd edition). McGraw-Hill. <https://ebookcentral.proquest.com/lib/dmu/detail.action?docID=6255465>

Software lifecycle part:

- Fowler, M. (2004). UML distilled: a brief guide to the standard object modeling language (3rd ed). Addison-Wesley. <http://www.vlebooks.com/vleweb/product/openreader?id=DeMontfort&isbn=9780134865126>

The Ethics part (an extensive list – subject to change as new cases come up)

- Ball, K. (2005). Organization, Surveillance and the Body: Towards a Politics of Resistance. Organization, 12(1), 89–108. <https://doi.org/10.1177/1350508405048578>

- Barnes, S. B. (2006). A privacy paradox: Social networking in the United States. *First Monday*, 11(9). <https://doi.org/10.5210/fm.v11i9.1394>
- BBC Radio 4 - The Nervous Breakdown of the Internet. (2015, December 1). <http://www.bbc.co.uk/programmes/b06qjzv2>
- BBC Radio 4 - The Public Philosopher - Downloads. <https://www.bbc.co.uk/programmes/b01nmlh2/episodes/downloads>
- Desai, D. (2013). Beyond Location: Data Security in the 21st Century. *Communications of the ACM*, 56(1), 34–36. <https://doi.org/10.1145/2398356.2398368>
- Eriksson, S., & Helgesson, G. (2005). Potential harms, anonymization and the right to withdraw consent to biobank research. *European Journal of Human Genetics*, 13(9), 1071–1076. <https://doi.org/10.1038/sj.ejhg.5201458>
- Haggerty, K. D., & Ericson, R. V. (2000). The surveillant assemblage. *British Journal of Sociology*, 51(4), 605–622. <https://doi.org/10.1080/00071310020015280>
- Hong, J. I., Ng, J. D., Lederer, S., & Landay, J. A. (2004). Privacy risk models for designing privacy-sensitive ubiquitous computing systems. *Proceedings of the 2004 Conference on Designing Interactive Systems Processes, Practices, Methods, and Techniques - DIS '04*. <https://doi.org/10.1145/1013115.1013129>
- Information Commissioner's Office. (2018). ICO. <https://ico.org.uk/>
- Iyad Rahwan: What moral decisions should driverless cars make? | TED Talk | TED.com. https://www.ted.com/talks/iyad_rahwan_what_moral_decisions_should_driverless_cars_make?utm_source=tedcomshare&utm_medium=email&utm_campaign=tedsread-a
- Keshav, S. (2007). How to read a paper. *ACM SIGCOMM Computer Communication Review*, 37(3), 83–84. <https://doi.org/10.1145/1273445.1273458>
- Langheinrich, M. (2001). Privacy by Design — Principles of Privacy-Aware Ubiquitous Systems. In G. D. Abowd, B. Brumitt, & S. Shafer (Eds.), *UbiComp 2001: ubiquitous computing : International Conference, Atlanta, Georgia, USA, September 30-October 2, 2001 : proceedings* (Vol. 2201, pp. 273–291). Springer. <https://ebookcentral.proquest.com/lib/dmu/reader.action?docID=3072498&ppg=286>
- Law, S. (2004). *The philosophy gym: 25 short adventures in thinking*. Review.
- Leenes, R. E., & Koops, B.-J. (2006). 'Code' and Privacy - or How Technology is Slowly Eroding Privacy. In E. J. Dommering & L. F. Asscher (Eds.), *Coding regulation: essays on the normative role of information technology* (Vol. 12). T.M.C. Asser. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=661141
- Leenes, R., & Koops, B.-J. (2005). 'Code': Privacy's death or saviour? *International Review of Law, Computers & Technology*, 19(3), 329–340. <https://doi.org/10.1080/13600860500348572>
- Lyon, D. (2014). Surveillance, Snowden, and Big Data: Capacities, consequences, critique. *Big Data & Society*, 1(2), 1–13. <https://doi.org/10.1177/2053951714541861>

- Mann, S., & Ferenbok, J. (2013). New Media and the power politics of sousveillance in a surveillance-dominated world. *Surveillance & Society*, 11(1/2), 18–34. <https://doi.org/10.24908/ss.v11i1/2.4456>
- Mitrou, L., & Karyda, M. (2006). Employees' privacy vs. employers' security: Can they be balanced? *Telematics and Informatics*, 23(3), 164–178. <https://doi.org/10.1016/j.tele.2005.07.003>
- Mittelstadt, B. D. (2013). Privacy, Risk and Personal Health Monitoring. *ETHICOMP 2013 Conference Proceedings*. <https://scholar.google.co.uk/citations?user=tP685zYAAAAJ&hl=en>
- Moor, J. H. (1997). Towards a theory of privacy in the information age. *ACM SIGCAS Computers and Society*, 27(3), 27–32. <https://doi.org/10.1145/270858.270866>
- Petzold, C. (2008). *Annotated Turing: A Guided Tour Through Alan Turing's Historic Paper on Computability and the Turing Machine*. JohnWiley & Sons, Incorporated.
- Philosophy Bites. <https://philosophybites.com/>
- Surveillance & society. <https://ojs.library.queensu.ca/index.php/surveillance-and-society/>
- The Futuremakers podcast | Research | University of Oxford. <https://podcasts.ox.ac.uk/series/futuremakers>
- Thomas, K. (2014). Privacy implications of online consumer-activity data: an empirical study. *Second Workshop on Society, Privacy and the Semantic Web - Policy and Technology*. <http://oro.open.ac.uk/41308/>
- Thuraisingham, B. (2002). Data mining, national security, privacy and civil liberties. *ACM SIGKDD Explorations Newsletter*, 4(2), 1–5. <https://doi.org/10.1145/772862.772863>
- Dorsey, P. (2000). Top 10 Reasons Why Systems Projects Fail. https://dulcian.com/articles/dorsey_top10reasonssystemspjrojectsfail.pdf
- Wacks, R. (2015). *Privacy: a very short introduction* (Second edition). Oxford University Press.
- Wilton, R. (2009). What's happened to PETs? *Information Security Technical Report*, 14(3), 146–153. <https://doi.org/10.1016/j.istr.2009.10.010>
- Wooldridge, M. J. (2020). *The road to conscious machines: the story of AI*. Pelican Books.
- Zuboff, S. (2019). *The age of surveillance capitalism: the fight for a human future at the new frontier of power*. PublicAffairs. <http://www.vlebooks.com/vleweb/product/openreader?id=DeMontfort&isbn=9781782832744>

Computer Programming

Module Code: CTEC1703N

Credit Value: 30

Credit Level:

Academic Level 4

Module Description

Computer programming requires the analysis of a problem, the production of requirements, and their translation into a design that can be executed on a computer. The design phase in particular requires the identification and combination of appropriate programming abstractions. This module introduces the skills required to develop a computer program to solve a given problem and does so from the perspective of designing trustworthy software with an emphasis on sound coding principles and unit testing.

Outline content:

- Practical program design using the control and data abstractions in a contemporary programming language.
- Highlighting techniques and approaches from different computer programming paradigms.
- The importance of good programming practice and the relevance of coding standards.
- The role of problem analysis and program specification across different computer programming paradigms.
- The use of functions in computer programming design and the production of unit tests.
- The role of testing in the software development process.

During the module students should be made aware of important principles of developing trusted software including, e.g., naming conventions, initialisation of structures and variables, variable scope and lifetime, validation of input. Students should also be made aware of the consequences of poor programming style and technique (i.e. poor maintainability, poor security and vulnerability to attack). Later in the module students will be exposed to the importance and benefit of using functions within their program design and will then utilise a key principle of trustworthy software by using an industry standard unit testing framework. Students will also consider different approaches to identifying and solving a problem in the context of a contemporary programming language.

This module takes its offset in the programming language Scala (www.scala-lang.org/). So why Scala? The landscape of programming is changing. There is still plenty of traditional, imperative and object-oriented code being used in the world which uses control abstractions known as sequence, selection and iteration. This means studying the order of statements (instructions) in a program; how to let a program choose between alternative statements; and how to control statements being executed repeatedly. Object-oriented programming brought in some very powerful ideas including applying functions to collections so that the collections are transformed. Functional programming promotes a similar style of transforming collections by applying functions over them. Crucially, these approaches differ from the imperative one-step-after-another approach. This might all be of theoretical interest only if it were not for the fact that contemporary programming languages including Scala, Java, C#, C++ are all now combining functional and object-oriented concepts. Scala has a relatively clean syntax and a useful interactive compiler which makes it good for teaching the concepts.

Although the student learn Scala on the module, the skills are transferrable to other programming languages.

Learning outcomes

1. Design a program to a given specification using the control and data structure abstractions provided by a contemporary programming language (Online Test 1 & Online Test 2)
2. Deploy trusted software design techniques in the construction of a computer program (Online Test 2)
3. Analyse a problem and produce a program specification (Practical)
4. Apply relevant software testing techniques to verify the components of a computer program and ensure it is trustworthy (Practical)

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't (X = Yes)	Minimum Threshold Mark	Essential Component (X = Yes)	Anon. Marking Code
Practical	HRS	16	40	X			OPT01
Online Test 1	HRS	0.7	30				AM
Online Test 2	HRS	0.7	30				AM

Anonymous marking exemption codes: OPT01: Individually distinct work; OPT02: Reflection on development of own work; OPT03: Presentation; OPT04: individually negotiated work; OPT05: work placement/experience/assessment

Assessment Notes

The module delivers a number of different themes within the subject area with an assessment point coming at the end of each of these. Outcomes are intertwined with different emphasis on each of them as the module evolves. Therefore, the first test may cover LO1 at a basic level but this will also be part of the second test in which certain design techniques have now been learned. The second test also considers trusted software design techniques in the construction of a computer program (LO2). The practical work involves analytical and problem-solving skills (LO3) that will be developed during the module and will allow students to demonstrate software testing techniques and use of functions (LO4).













The assessment model encourages a reflective approach to teaching and learning in which students are encouraged to use feedback gained and re-visit earlier material. The assessment strategy gives students the possibility to demonstrate improved understanding which fits exactly the nature of learning to program in which deeper understanding of earlier concepts can come with experience. Overall, this aligns with an integrated approach to assessment throughout the module where regular practical tasks and formative practice tests directly support assessed work. One phase test may be re-attempted (optionally) within the module delivery period. If an optional second attempt is taken, then the higher mark of the two attempts for that phase test will be recorded.

Expected Methods of Delivery

Workshops will be used to introduce and demonstrate key practical and theoretical concepts. Practical programming skill will be gained in regular laboratory sessions. Some sessions may be used for consolidation, revision, and to discuss solutions to practical problems.

Workshop	24 hours
Practical	42 hours
Self-directed study	76 hours
Consolidation	68 hours
Reading	30 hours
Assessment	60 hours – subsumes estimated time for producing deliverables as per the assessment table

Literature

	The Scala Programming Language Website - Essential	VIEW ONLINE	<input type="radio"/>	⋮
 The main website for the Scala language. You can get software downloads here as well as tutorial information and documentation.				
	Official Scala documentation Webpage - Recommended	VIEW ONLINE	<input type="radio"/>	⋮
 This website contains official documentation for Scala such as the language's style guide, API, etc. Please refer to the notes on Blackboard for details on specific areas that may be of interest.				
	Scala Tutorials by TutorialsPoint Webpage - Recommended	VIEW ONLINE	<input type="radio"/>	⋮
 A great set of online tutorials for Scala.				
	Programming in Scala. Martin Odersky, Lex Spoon, Bill Venners Book - by Martin Odersky - 2016 - Recommended	VIEW ONLINE	<input type="radio"/>	⋮
 This is the book written by the developers of the Scala language itself. It is an excellent resource.				
	Scala cookbook: recipes for object-oriented and functional programming Book - by Alvin Alexander - 2021 - Optional	VIEW ONLINE	<input type="radio"/>	⋮
 Great supporting text on Scala.				
	Introduction to programming and problem-solving using Scala Book - by Mark C. Lewis; Lisa L. Lacher - 2017 - Optional	VIEW ONLINE	<input type="radio"/>	⋮
 One of a number of contemporary text books on Scala. This is an excellent text but requires careful reading.				

Operating systems and Networks

Module Code: CTEC1704N Credit Value: 30 Credit Level: Academic Level 4

Module Description

This module is designed to provide a foundation in computer architecture, operating systems, and computer networks.

Outline content:

1. Theoretical foundations:
 - a. Number systems, integer and real number representation
 - b. Finite state automata; Introduction to data encryption
2. Computer hardware:
 - a. CPU components and operation, instruction sets, Computer architectures, Memory
3. Systems software:
 - a. Operating system fundamentals, processes, file systems, memory management
 - b. Shell scripting
 - c. Authentication and authorisation.
4. Computer networks:
 - a. Network architectures, data communication system fundamentals, Transmission schemes and technologies, error detection and management
 - b. Network components, LAN protocols, Internet protocols.
5. Security issues:
 - a. Information security: confidentiality, integrity and availability (CIA), Network vulnerability and security, threats and attacks
 - b. Operating system vulnerability and security, threats and attacks
 - c. Architectural vulnerability and security

Learning outcomes

1. Demonstrate the theoretical and practical understanding of computer hardware, operating systems and networks. (Phase Test 1, 2 and 3)
2. Relate the abstract concepts of logic and number systems to their concrete representation on real machines. (Phase Test 1, 2 and 3)
3. Identify the security risks in common configurations of computer operating systems and suggest appropriate mitigations. (Phase Test 1, 2 and 3)
4. Identify the security risks in common configurations of computer networks and suggest appropriate mitigations. (Phase Test 1, 2 and 3)

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't (X = Yes)	Minimum Threshold Mark	Essential Component (X = Yes)	Anon. Marking Code
Phase Test 1	HRS	0.7	30				AM
Phase Test 2	HRS	0.7	30				AM
Phase Test 3	HRS	0.7	40				AM

Anonymous marking exemption codes: OPT01: Individually distinct work; OPT02: Reflection on development of own work; OPT03: Presentation; OPT04: individually negotiated work; OPT05: work placement/experience/assessment

Assessment Notes

Assessment 1, 2 and 3 are all online closed-book phase tests to assess factual knowledge in all topics. Students may be provided with a handout containing factual information that will be used to assess the students' skills and their ability to apply their knowledge. The approach to assessment on this module is integrated with the delivery, and rewards students for engaging with the practical content. One phase test may be re-attempted (optionally) within the module delivery period. If an optional second attempt is taken, then the higher mark of the two attempts for that phase test will be recorded.

Expected Methods of Delivery

Lectures will be used to introduce the main theoretical elements. Laboratory sessions will be used for practical application and experimentation.

Workshop	24 hours
Practical	42 hours
Self-directed study	66 hours
Consolidation	68 hours
Reading	40 hours
Assessment	60 hours – subsumes estimated time for producing deliverables as per the assessment table

Literature



Operating system concepts

Book - by Abraham Silberschatz; Peter B. Galvin; Greg Gagne - 2014 - Essential



Computer organization and design: the hardware/software interface

Book - by David A. Patterson; John L. Hennessy - 2018 - Essential

[VIEW ONLINE](#)



Computer networking: a top-down approach

Book - by James F. Kurose; Keith W. Ross - 2017 - Essential

[VIEW ONLINE](#)



Advanced Bash-Scripting Guide

Webpage - Essential

[VIEW ONLINE](#)



The practice of system and network administration

Book - by Tom Limoncelli; Christina J. Hogan; Strata R. Chalup - 2017 - Essential

[VIEW ONLINE](#)



Windows Command Line Beginner's Guide

Book - by Jonathan Moeller - 2019 - Essential

[VIEW ONLINE](#)



LinuxCommand.org: Learn The Linux Command Line. Write Shell Scripts.

Webpage - Essential

[VIEW ONLINE](#)



How Linux works

Book - by Brian Ward - 2021 - Recommended

[VIEW ONLINE](#)



The Linux command line: a complete introduction

Book - by William E. Shotts - 2019 - Recommended

[VIEW ONLINE](#)



Linux command line and Shell scripting bible

Book - by Richard Blum; Christine Bresnahan - 2021 - Essential

[VIEW ONLINE](#)



Shell Scripting: How to Automate Command Line Tasks Using Bash Scripting and Shell Programming

Book - by Jason Cannon - 2015 - Recommended



Module descriptions Level 5

Object Oriented Design and Development

Module Code: CTEC2710N **Credit Value:** 30 **Credit Level:** Academic Level 5

Module Description

This module focuses on Object-Oriented (OO) library and application development. Library development will enable students to design, implement, and test medium scale software systems using an object-oriented approach. Meanwhile, application development will use extensive library packages provided by the Java SDK so that students are comfortable in navigating and making use of a variety of domains such as Collections, Input/Output and Graphical User Interfaces.

The design notation used is the Unified Modelling Language (UML) and the implementation language is Java.

It is essentially a programming module, with the emphasis on implementing OO designs and producing reusable libraries. Students enrolling on this module will have a foundation in programming gained from the study of a level 4 programming module. This module introduces UML and the Java language. Fundamental OO concepts and design issues are discussed. Emphasis is placed on the design and structure of software, and the OO development process. UML is used to document designs, and the concept of software design patterns are introduced.

Contemporary areas of the API will be used to showcase how OO applications can benefit from more recent functional additions such as lambda expressions and stream pipelines. Students will be required to build graphical user interfaces and consider associated features such as layout policies, observable data models, and binding events to properties. The use of advanced areas of the API also allows a variety of design patterns such as composite, strategy and decorator to be discussed and deployed.

Software quality, reliability, and maintainability are integral to the development of software, and are integrated into the delivery by considering different approaches to solving common problems in application design. In particular, encapsulation, decomposition, and decoupling are viewed within the context of the model-view-controller (MVC) architecture. These design issues will commonly link to advised techniques of delivery as outlined in the PAS 754:2014 software trustworthiness specification.

By the end of the module students should have become more independent learners with the ability to adapt their existing knowledge and learn additional software libraries and features.

Outline Content:

- The Java programming language, essential language constructs, and the API
- Principles of the object-oriented paradigm; Objects, Classes, Methods
- OO software design - encapsulation and delegation, polymorphism, etc
- UML notation - both static (e.g. class diagrams) and dynamic (e.g. Object diagrams)
- Implementation of associations – composition, aggregation and inheritance
- User defined classes, data representation decisions, Interfaces and abstraction
- Quality issues in software development
- Unit testing
- Documentation for maintainability and reusability and use of the Javadoc tool
- Identification and application of relevant design patterns, e.g. iterator and strategy
- Navigate and make use of a variety of domains within the SE Java API
- The collections framework and the data structures that underpin them - Nested types, functional interfaces and lambda expressions
- Processing data with stream pipelines - File processing, input and output streams, and exception handling
- Graphical user interfaces and event handling
- Deploying the model-view-controller software architecture

Learning outcomes

1. Use Java to implement standard object-oriented designs given in UML. (Portfolio)
2. Design and develop trustworthy software in the context of an object-oriented language. (Portfolio)
3. Design and develop applications with an emphasis on quality, maintainability, correctness and robustness, enhancing their trustworthiness. (Programming specification)
4. Make effective use of the Java Software Development Kit Application Programming Interfaces. (Programming specification)

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't (X = Yes)	Minimum Threshold Mark	Essential Component (X = Yes)	Anon. Marking Code
Portfolio	HRS	20	50				OPT01
Programming Specification	HRS	20	50	X			OPT01

Anonymous marking exemption codes: OPT01: Individually distinct work; OPT02: Reflection on development of own work; OPT03: Presentation; OPT04: individually negotiated work; OPT05: work placement/experience/assessment

Assessment Notes





The portfolio assessment is based on formative work carried out in lab and tutorial classes. The portfolio assignment is carried out over several weeks and involves using Java effectively to implement a variety of different OO associations given UML designs (outcome 1). The portfolio work also expects consideration of relevant quality issues (outcome 2). The programming specification assessment will require students to use advanced features of the JDK to build a substantial piece of software (outcome 4) using a given specification. There will additionally be an emphasis on the design approaches taken by considering relevant quality issues (outcome 3).

Expected Methods of Delivery

Workshops will be used to introduce and demonstrate key practical and theoretical concepts. Practical programming skill will be gained in regular laboratory sessions. Some sessions may be used for consolidation, revision, and to discuss solutions to practical problems.

Workshop	30 hours
Practical	42 hours
Self-directed study	70 hours
Consolidation	68 hours
Reading	30 hours
Assessment	60 hours – subsumes estimated time for producing deliverables as per the assessment table

Literature

	Java Tutorials hosted by Oracle Website - by Oracle - Essential	VIEW ONLINE	<input type="radio"/>
	Official Java API Specification Website - by Oracle - Essential	VIEW ONLINE	<input type="radio"/>
	Java Fundamentals Tutorials Website - by ProTech - Recommended	VIEW ONLINE	<input type="radio"/>
	Chua Hock-Chuan's Java Programming notes Website - by Chua Hock-Chuan - Recommended	VIEW ONLINE	<input type="radio"/>

Data Structures and Algorithms

Module Code: CTEC2711N **Credit Value:** 30 **Credit Level:** Academic Level 5

Module Description

This module covers a variety of data structures and algorithms for both sequential and parallel execution as core module for BSc Computer Science programme.

The focus of the module is that students should be able to establish the importance of structured representations of data in the memory, the design of various data structures, the rationale behind organising data and how it supersedes the benefits of accessing data with fast retrieval.

The module establishes a strong foundation by employing linear data structures such as array, linked list, stack and queue and then advanced through non-linear data structures such as Hash tables, trees and so on. The sorting and searching algorithms - selection, bubble, insertion, merge, and quick sort and their implementation are covered in detail. The data structure concepts will be further clarified through real-time applications and their implementation in an object-oriented framework. Several applications that implement data structures are represented: the stack of trays in a cafeteria, a stack of plates in a cupboard, maze, parsing arithmetic expressions, backtracking and so on.

The algorithm analysis is done using asymptotic analysis using the time complexities for linear, quadratic, logarithmic, exponential, and so on. The efficiency of the algorithms is measured in terms of theta, omega and Big O notation. The selection of the data structure determines how the operations (such as reading to, writing from, modifying, or computing with the data) can be achieved for the high performance with reduced time complexity.

Later in the module, students will be introduced to concurrent program design in the context of multi-core architectures and distributed applications. With the increasing use of multi-core processors, there is now a greater need to understand how concurrent and parallel programs work. The module will teach the theoretical background to enable students to reason about such programs and determine the possible results a concurrent program could give. It will also show the practical algorithms that can be used to write concurrent programs, focusing on ensuring that such programs will work correctly and provide the expected output.

The module will also introduce the formal notations in the form of temporal logic and will show various methods for using temporal logic to verify the correctness of concurrent programs. While the programming language focus will be on Java, the theoretical concepts are applicable to many languages.

Outline Content

- Sequential algorithms for searching and sorting and their performance characteristics
- Classical data structures (linear, hierarchical, hash tables)
- Implementation issues (genericity, reference and value semantics, mutable and immutable objects)
- Formal notation for functional specification
- Concurrent (and parallel) algorithms
- Program (and program library) support for concurrent (and parallel) algorithm implementation

Learning outcomes

1. Explain and implement a variety of classical data structures (Practical and Phase Test 1)
2. Apply classic sequential algorithms for searching and sorting (Phase Test 1)
3. Analyse specific programming language and library support for the development of sequential and concurrent programs (Practical)
4. Apply standard concurrent algorithm design patterns (Practical and Phase Test 2)
5. Explain formal notation in specific contexts (Phase Test 2)

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't (X = Yes)	Minimum Threshold Mark	Essential Component (X = Yes)	Anon. Marking Code
Phase test 1	HRS	0.7	25				AM
Phase Test 2	HRS	0.7	25				AM
Practical	HRS	20	50	X			AM

Anonymous marking exemption codes: OPTO1: Individually distinct work; OPTO2: Reflection on development of own work; OPTO3: Presentation; OPTO4: individually negotiated work; OPTO5: work placement/experience/assessment

Assessment Notes

Theory and practice in this module are very highly coupled. Practical work will typically involve the implementation and analysis of selected data structures and algorithms, including concurrent and/or parallel algorithms.

The practical assignment will consist of problem-solving tasks, assessing the students' knowledge and implementation skills for a variety of classical data structures (LO1), in addition to standard concurrent algorithm design patterns (LO4). Furthermore, students will be required to analyse programming language and library support for these topics (LO3).

The first phase test will cover the theoretical topics of the module, assessing the students' understanding of a variety of classical data structures (LO1) and their ability to apply classic sequential algorithms for searching and sorting (LO2). The second phase test will assess the theoretical aspects of the module, in particular concurrent algorithm design patterns (LO4) and formal notation in specific contexts (LO5)

One phase test may be re-attempted (optionally) within the module delivery period. If an optional second attempt is taken, then the higher mark of the two attempts for that phase test will be recorded.

Expected Methods of Delivery

Workshops will be used to introduce and demonstrate key practical and theoretical concepts. Practical programming skill will be gained in regular laboratory sessions. Some sessions may be used for consolidation, revision, and to discuss solutions to practical problems.

Workshop	24 hours
Practical	42 hours
Self-directed study	76 hours
Consolidation	68 hours
Reading	30 hours
Assessment	60 hours – subsumes estimated time for producing deliverables as per the assessment table

Literature



Data structures and algorithms in Java

Book - by Michael T. Goodrich; Roberto Tamassia; Michael H. Goldwasser - 2015 - **Recommended**



Problem solving in data structures & algorithms using Java

Book - by Hemant Jain - 2018 - **Optional**



Data structures and problem solving using Java

Book - by Mark Allen Weiss - 2013 - **Optional**

[VIEW ONLINE](#)



Data structures, algorithms, and applications in Java

Book - by Sartaj Sahni - 2005 - **Optional**



Web Application Development

Module Code: CTEC2712N **Credit Value:** 30 **Credit Level:** Academic Level 5

Module Description

This module provides a thorough grounding in the rapidly evolving area of web technologies. With equal focus on user interface design on the 'client-side' or 'front-end' and on security and persistence in 'server-side' or 'back-end' scripting. The module covers crucial design principles, information architecture and usability factors as well as standards compliance, accessibility, authentication/authorisation and security.

This exciting field has been driven by advances in web standards, in particular HTML5 (introduced in 2008 and accepted as a 'living standard' in May 2019), modular CSS (~2012) and the ECMAScript standard (ECMAScript2015 in particular) which together define the modern web platform. The web standards process enables the platform to evolve extremely quickly, and new features are released with increasing regularity.

Modern web applications typically make heavy use of server-side scripting using development languages such as PHP. This pragmatic language is used to great effect by some web developers and with catastrophic naivety by others. Web application solutions are typically implemented by teams of developers. The various roles typically undertaken will be discussed with students.

Outline Content:

- Design principles and user experience
- Information Architecture
- Usability and accessibility
- HTML5, CSS3 and semantic markup
- Modern JavaScript and its context
- Responsive web design
- Version control systems
- Server-side scripting using a web development language such as PHP
- Persistent storage systems
- Common patterns of usage and failure

Learning outcomes

1. Demonstrate a critical understanding of information architecture, user interface design and usability principles (phase test and web project).
2. Effectively apply knowledge of client-side web development technologies (web project).
3. Effectively apply knowledge of server-side web development technologies (web project).
4. Demonstrate a critical understanding of security protocols in web applications (phase test and web project).

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't (X = Yes)	Minimum Threshold Mark	Essential Component (X = Yes)	Anon. Marking Code
Phase test 1	HRS	0.7	40				AM
Web Project	HRS	24	60	X			OPT01

Anonymous marking exemption codes: OPT01: Individually distinct work; OPT02: Reflection on development of own work; OPT03: Presentation; OPT04: individually negotiated work; OPT05: work placement/experience/assessment

Assessment Notes

The web project submission will be assessed on the prototype planning, quality of code, knowledgeable application and integration of the relevant technologies, and adaptation of taught materials. It is to be submitted via a version control

system, with regular commits made during the module.

The phase test will cover all conceptual aspects of the module's content ensuring students are aware of design principles, commercially recognised programming standards and techniques, can apply and evaluate security in the context of web applications, as well as being able to identify common web-application security vulnerabilities.

As part of the required preparation for the phase test, students will work in development teams, where they will typically develop a web application with assigned roles, considering commercially recognised programming standards and techniques.

The phase test may be re-attempted (optionally) within the module delivery period. The higher mark of the two attempts will be recorded.

Expected Methods of Delivery

Workshops will be used to introduce and demonstrate key practical and theoretical concepts. Practical programming skill will be gained in regular laboratory sessions. Some sessions may be used for consolidation, revision, and to discuss solutions to practical problems.

Workshop	24 hours
Practical	42 hours
Self-directed study	76 hours
Consolidation	68 hours
Reading	30 hours
Assessment	60 hours – subsumes estimated time for producing deliverables as per the assessment table

Literature



PHP and MySQL Web Development

Book - by Luke Welling - 2016 - Recommended

[VIEW ONLINE](#)

PHP Pandas: The PHP programming language for everyone

Book - by Dayle Rees - Essential

[VIEW ONLINE](#)

Agile Development Team Project

Module Code: CTEC2713N **Credit Value:** 30 **Credit Level:** Academic Level 5

Module Description

This module is an opportunity for students to engage in a constrained work-place simulation based on agile software development. Students working in teams of 3 to 5 will initially identify a system of sufficient size to be distributed equally among all members. Work allocation will be monitored under the guidance of their tutor/supervisor. For example, each team member might take individual ownership of the development of 2-3 classes from initial inception to completion providing CRUD functionality. In the case of a large system this may mean that some aspects of the system are never built to completion.

By the end of the module each team will be expected to develop an integrated software component based on their individual work. This would typically be an administrative dashboard allowing for maintenance of system data.

No specific language is named for the module; however, it would be wise within a single team to select a family of languages/development environments aligned with the prior experience of members based on their taught programme, ideally this should also align with the team's tutor/supervisor. Projects will need to be carefully sourced to match this range of skills. In house projects might also be available. The assessment will be designed to encourage collaboration, peer learning, and formative feed-forward assessment.

The module will include supporting materials introducing concepts and practice relating to agile development however beyond that there is no formal taught content as this module aims to consolidate learning and skills from prior and concurrent study. It is expected that students will use an appropriate set of tools for collaboration for example git- hub.

The ethical component will focus on software design and development and risk factors associated with the management of these projects e.g. the pitfalls and problems; why do software development projects fail? There will also be a focus on examining ethical approaches for software development (co-creation, collaboration, stakeholder engagement, etc). Additionally, data collection technologies and approaches, processing and re-use, storage, security, and data minimisation will be considered in the context of why protection of personal and business data is an ethical issue.

Learning outcomes

1. Work collaboratively with both tutor and peers (Portfolio)
2. Produce code-based solutions to problems generating suitable documentation (Portfolio)
3. Apply good practice in code design (Portfolio)
4. Find answers to programming problems (Portfolio)
5. Research into an area of computer ethics and integrate different ideas into a coherent analysis of a system, and demonstrate critical and analytical skills in understanding the risks associated with managing software projects (Case study analysis)

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't	Minimum Threshold Mark	Essential Component	Anon. Marking Code
Portfolio			80				OPTO3
Case Study Analysis			20				AM

Anonymous marking exemption codes: OPTO1: Individually distinct work; OPTO2: Reflection on development of own work; OPTO3: Presentation; OPTO4: individually negotiated work; OPTO5: work placement/experience/assessment

Assessment Notes

A sprint is a time limited period of work where students work collaboratively to support each other on their individual projects. Each sprint ends with a sprint meeting initially of five minutes of one-to-one contact between tutor and student. These meetings provide an opportunity for formative feed-forward assessment and feedback such that the tutor may support the student in their learning as they progress through the module. The final ten-minute sprint meeting is summative in nature where the student's work is marked by the tutor and immediate feedback is provided.

Students are encouraged to work in teams to support each other; however, all assessment is individual. There are options available within the assessment to cater for any requirements relating to UDL¹.

The case study analysis will involve utilising codes of ethics to investigate and identify points of failure related to the management of software development projects, and to suggest a suitable ethical approach to address the case being analysed.

Expected Methods of Delivery

The emphasis for assessment will typically be focussed on developing an administrative dashboard allowing for maintenance of system data. The workshops will be used for formative feed forward assessments. The seminars will be used to introduce and discuss ethical issues. Practical programming skills will be gained in regular laboratory sessions. Some workshops and practical laboratory sessions may be used for consolidation and to discuss solutions to practical and ethical problems.

Workshop	42 hours
Practical	20 hours
Seminar	4 hours
Self-directed study	76 hours
Consolidation	78 hours
Reading	20 hours
Assessment	60 hours – subsumes estimated time for producing deliverables as per the assessment table

Literature



Modern systems analysis and design

Book - by Joseph S. Valacich; Joey F. George - 2020 - Recommended

[VIEW ONLINE](#)



Introducing systems development

Book - by Steve Skidmore; Malcolm Eva - 2004 - Optional



Object-oriented systems analysis and design using UML

Book - by Simon Bennett; Steve McRobb; R. Farmer - c2010 - Optional



W3Schools - SQL

Website - Optional

[VIEW ONLINE](#)



Useful website for SQL when creating your stored procedures



W3Schools - C#

Website - Optional

[VIEW ONLINE](#)



¹ **Universal design for learning (UDL)** is a teaching approach that works to accommodate the needs and abilities of all learners and eliminates unnecessary hurdles in the learning process

Module descriptions Level 6 - Core

Software Development: Methods and Standards

Module Code: CTEC3701N **Credit Value:** 30 **Credit Level:** Academic Level 6

Module Description

This module immerses the students in the methodological, regulation environment in which software systems are developed. This is achieved by exploring four types of application development: module, AI, robotic process automation and games systems. These application classes provide the basis for exploring methodological approaches, with a particular emphasis on current agile approaches, in particular Scrum and DevOps. The different needs and emphasis of different applicant classes are considered. Within this context the methodological evolution required in scaling, in embedding, in security by design and in integration are considered.

The module incorporates the context of standards with particular reference to standards in security (ISO27001) and risk management (ISO27005). Practical exercises include an engagement with current agile project management tools (e.g. Jira and Clickup). Students develop an understanding of a range of agile and traditional methodologies and consider the debates, ambiguity and uncertainty in their application.

The module considers legal and ethical aspects of software development. The ethical approach addresses responsible development and embedding sustainability through a considering of green IT issues. Ethics studies include intellectual property (who owns your data?), copyright, patents and trademarks, and the ethics of ownership (NFTs etc). The digital divide is covered (e.g. economic, social, political, cultural, etc) understanding the issues and solutions (including SDG's). Plus, topics surrounding responsible research and innovation (RRI) and how it can begin to address the ethical issues of technological development.

The law component of the module focusses on three key areas – computer misuse, professional negligence, and intellectual property law. Relevant elements of the Computer Misuse Act 1990 (CMA 1990), the civil tort of negligence (and how it applies to professionals) and both the criminal law and civil law aspects of Intellectual Property will be introduced, developing knowledge and understanding of the various obligations.

Outline Content:

- Introduction to methodologies. Overview of agile approaches. Comparing agile and waterfall. Why methodologies are important. Steps and characteristics of developing a mobile application. The development environment in an organisation. Method versus methodology. Ethics of responsibility, accountability and sustainability in software development.
- Developing AI and ML applications. Methodological approaches. Software development as a learning exercise. Systems development standards. Concept of a standard. How standards work. Sustainability consideration in application development.
- Ethics of software development. Intellectual property (who owns your data?), copyright, patents and trademarks, and the ethics of ownership. The digital divide (e.g. economic, social, political, cultural, etc) Understanding the issues and solutions (including SDG's). Responsible research and innovation (RRI). Robotic Process Automaton Scaling, Integration and Security methodologies. Spotify, Scaled Agile Framework. Secure development lifecycle. Security by design. Brief intro to DevOps and DevSecOps. Metrics.
- Games development: methodological approaches. Organisation in a development environments and automation. Design thinking overview. Risk game company. Issues of scaling in multi-player games. Importance of management and risk management standards.
- Criminal Law: The Computer Misuse Act 1990; Elements of Intellectual Property Law. Civil Law: The tort of negligence and how it applies to professional conduct; protection of intellectual property rights using civil actions.
- Revision; assessment completion.

Learning outcomes

1. Evaluate and select a methodology for developing a common application. (Structured Proposal)
2. Assess the consequences of scaling, integration and security in an application development. (Structured Proposal)
3. Devise a plan for the adoption of an appropriate standard within an application development environment. (Structured Proposal)
4. Recognise and evaluate current and future ethical issues surrounding the application of ICT. (Case study analysis)
5. Identify, critically analyse and apply laws, regulations and standards as found in the U.K. thereby demonstrating a sound understanding and ability to locate and use primary and secondary sources. (Report)

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't	Minimum Threshold Mark	Essential Component	Anon. Marking Code
Structured Applications and Methodologie			60				OPTO1
Case Study Analysis			20				AM
Report			20				OPTO2
Anonymous marking exemption codes: OPTO1: Individually distinct work; OPTO2: Reflection on development of own work; OPTO3: Presentation; OPTO4: individually negotiated work; OPTO5: work placement/experience/assessment							

Assessment Notes

The assessment for the ethics material is in the form of a case study analysis of a current issue/case surrounding the associated topics delivered on the module.

The law related material will be assessed by a written component that comprises a business report to a senior manager in their organisation that evaluates the merits of a systems or software business proposal/problem. Students should strive to write an accurate, concise, cogent, and critically analytical business report to their (fictitious) manager that sets out their recommendations in relation to an intellectual property (I.P.) issue that has arisen in a work context.

The software methodologies assignment is based round a client selected by the student. Following an analysis of the client and their current state and future needs, three applications are identified based on the different technologies and contexts explored in the module. Following a concise description of the proposed applications, the student will identify in outline needs for processing, data, hardware and any other requirements are identified. For each application the student will be required to propose two different methodological approaches and evaluation of the methodologies provided in summary which identifies the preferred methodology.

Expected Methods of Delivery

Lectures will be used to introduce and demonstrate key practical and theoretical concepts. Seminars will be used to discuss and further reinforce these. Some sessions may be used for consolidation, revision, and to discuss solutions to practical problems.

Lectures	24 hours
Seminar	42 hours
Self-directed study	86 hours
Consolidation	68 hours
Reading	40 hours

Assessment	60 hours – subsumes estimated time for producing deliverables as per the assessment table
------------	---

Literature

- Farley, D., 2021. Modern Software Engineering: Doing What Works to Build Better Software Faster. Addison-Wesley Professional. ISBN: 978-0137314911 (Recommended)
- Jacobson, I., Ng, P.W., McMahon, P.E. and Goedicke, M., 2019. The essentials of modern software engineering: free the practices from the method prisons!. Morgan & Claypool. ISBN: 978-1947487246 (Optional)
- Sommerville, I., 2015. Software Engineering, Global Edition. Tenth Edition Pearson. ISBN: 978-1292096131 (Optional)

Big Data and Machine Learning

Module Code: CTEC3702N **Credit Value:** 30 **Credit Level:** Academic Level 6

Module Description

The module will focus on machine learning and its application to Big Data in a "taster-like" fashion. That is, ML will be applied to solve analytics problems using appropriate tools e.g., Apache Spark that avail ML libraries. As this is done ML algorithms will be introduced and then applied. The focus is therefore not so much on the technical details of the algorithms - rather, the ability to implement them and use them within analytics. The module covers supervised and unsupervised learning techniques with a specific application to data mining. Selected classification, regression, and clustering approaches will be examined. Algorithm evaluation and evaluation metrics will be explored, and machine learning frameworks and tools introduced. The module also covers Big Data, Big Data analytics, Machine learning frameworks e.g., Apache Spark, machine learning libraries e.g., Spark Machine Learning Library (MLlib), and Hadoop Distributed File System. Additionally, the module considers ethics in relation to AI, big data, and surveillance.

Outline Content:

- Overview of machine learning, concept learning, training data, and inductive bias.
- Experiments, evaluation metrics, and evaluation.
- Introduction to Big Data and Big Data analytics
- Introduction to Big Data storage infrastructure (such as the Hadoop Distributed File System, NoSQL databases, NewSQL databases and so on)
- Machine learning frameworks and tools e.g. WEKA, Apache Spark, etc.
- Machine learning libraries e.g. - Machine Learning Library (MLlib)
- Data exploration, analytics, & visualization
- Linear regression
- Classification: decision tree learning, Bayesian learning, nearest neighbour, and artificial neural networks
- Unsupervised learning and clustering
- Programming and Data Analytics using a machine learning framework e.g. Apache Spark/Graph Frames
- Surveillance and society, examining the development and impact of surveillance e.g. CCTV, Vaccine passports, online tracking etc
- The ethics of AI, its uses, issues, and concerns. To examine the changes in human autonomous decision making and its potential impact and influence on everything from jobs to health or food choices.
- Big data and its influence on organisational decision-making, marketing, and governmental 'nudge' policies.

Learning outcomes

1. Critically evaluate Big Data, where it comes from and what the opportunities and challenges are, including storage infrastructures and their pros and cons (Phase Test)
2. Develop solutions for data exploration, visualization and analytics using suitable machine learning frameworks such as Apache Spark. (Problem Specification)
3. Identify and describe classification and clustering algorithms and their application areas (Phase Test)
4. Select and apply the suite of machine learning algorithms available in a machine learning library e.g., Spark Machine Learning Library, to develop solutions to machine learning problems. (Problem Specification)
5. Research into an area of computer ethics, integrate complex and sometimes conflicting ideas into a coherent analysis that demonstrates integrative, synoptic, and analytical skills. (Case Study Analysis)

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't	Minimum Threshold Mark	Essential Component	Anon. Marking Code
Phase Test			30				AM
Problem Specification			50				OPTO1
Case Study Analysis			20				AM
Anonymous marking exemption codes: OPTO1: Individually distinct work; OPTO2: Reflection on development of own work; OPTO3: Presentation; OPTO4: individually negotiated work; OPTO5: work placement/experience/assessment							

Assessment Notes

Students will receive tutor support on their coursework during the scheduled laboratory sessions. Students will be provided with a problem description or select from a list of problem descriptions. The problem description(s) will be derived from contemporary real-world data analytics problems to give the students a clear feel of the real-world nature of the solutions they should develop. Each student will be expected to develop a solution and document both the process and results.

The phase test will consist of brief essays, multiple choice questions, and instances to fill in spaces. It will cover Big Data, opportunities arising from big data, and machine learning algorithms.

The case study analysis will involve researching into areas such as AI, surveillance, and big data, and considering uses, impacts, concerns, etc, from an ethical perspective.

Expected Methods of Delivery

Lectures will be used to discuss concepts, theories, and applications including machine learning algorithms and data analytics tools. Practical sessions will be used to undertake practical aspects of the module to solve selected data analytics problems from a wide range of areas.

Lecture/Workshop	24 hours
Practical	35 hours
Seminar	7 hours
Self-directed study	70 hours
Consolidation	64 hours
Reading	40 hours
Assessment	60 hours – subsumes estimated time for producing deliverables as per the assessment table

Literature



Big data analytics

Book - by Arvind Sathi - 2012 - Recommended

[VIEW ONLINE](#)



Mastering Apache Spark

Book - by Mike Frampton; Andrew Szymanski - 2015 - Optional

[VIEW ONLINE](#)



Machine learning for absolute beginners: a plain English introduction

Book - by Oliver Theobald - 2017 - Recommended



Machine learning with PySpark: with natural language processing and recommender systems

Book - by Pramod Singh - 2021 - Recommended

[VIEW ONLINE](#)



Big data, data mining and machine learning: value creation for business leaders and practitioners

Book - by Jared Dean - 2014 - Recommended

[VIEW ONLINE](#)



Machine Learning with Python A Practical Beginners Guide

Book - by Oliver Theobald - 2019 - Optional



Development Project (The Bachelor project)

Module Code: CTEC3451N **Credit Value:** 30 **Credit Level:** Academic Level 6

Module Description

The project provides students with the opportunity to carry out a significant piece of work that reflects the aims and outcomes of their specific programme. It provides students with the opportunity to demonstrate practical and analytical skills present in their programme of study; to work innovatively and creatively; to synthesise information, ideas, and practices to provide a quality solution, together with an evaluation of that solution. The project should meet some real need in a wider context.

Students will demonstrate an ability to self-manage a significant piece of work and will undertake a self-evaluation of the process. Students will be expected to demonstrate appropriate and proactive project management and written/verbal presentation skills throughout the period of the project. As well as analysing, designing, delivering and appraising a product of suitable quality, they will be expected to undertake research, analysis, design, implementation, verification, evaluation and reporting pertinent to the project.

Indicative Content: The range of projects will be wide. Projects are obtained from a variety of sources including: internal academic proposals, external organisation suggestions, and a number from students themselves.

The deliverables will include:

1. A main report which will include:
 - a. elucidation of the problem and the objectives of the project
 - b. an in-depth investigation of the context and literature, and where appropriate, other similar products
 - c. where appropriate, a clear description of the stages of the life cycle undertaken
 - d. where appropriate, a description of how verification and validation were applied at these stages
 - e. where appropriate, a description of the use of tools to support the development process
 - f. a critical appraisal of the project, indicating the rationale for any design/implementation decisions, lessons learnt during the course of the project, and evaluation (with hindsight) of the project outcome and the process of its production (including a review of the plan and any deviations from it)
 - g. a description of any research hypothesis
 - h. references
2. A set of appendices that are referred to within the main report, and which contain the substantive work on the project, including product deliverables, such as requirements and design specifications and other project documents (project contract, inform consent, ethics review form etc.).
3. A product demonstration shortly after the submission of the main report.

Learning outcomes

1. Apply theoretical and practical concepts from the programme of study to the construction of a solution to a practical problem. (Initial and Final)
2. Carry out research and analysis to support the project requirements. (Initial)
3. Plan and selfmanage the work. (Initial and Final)
4. Assess the potential global impact of the work (Initial)
5. Present the work using a report in which the process and product are described, analysed, and critically evaluated. (Final)
6. Present and defend the work in a formal demonstration (Final)

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't	Minimum Threshold Mark	Essential Component	Anon. Marking Code
Project Management			10				OPT01
Final Submission			90				OPT01
Anonymous marking exemption codes: OPT01: Individually distinct work; OPT02: Reflection on development of own work; OPT03: Presentation; OPT04: individually negotiated work; OPT05: work placement/experience/assessment							

Assessment Notes

Each project submission will be assessed holistically on specific criteria which cover the report, the product, the management, and the demonstration. There are several means by which evidence for assessing project work may be gathered including but not limited to

- Attendance at supervisory meetings and the outcomes of these meetings
- The written deliverable submissions, in the manner as detailed in the current student project handbook
- Demonstration

An appropriate marking scheme will be available on the module Blackboard shell.

The project demonstration is compulsory and nonattendance will normally result in a mark of zero for the project.

Expected Methods of Delivery

The project is primarily self-directed with guidance and support from an assigned supervisor. Project skills sessions will normally be provided to give students the necessary pre-requisite knowledge and skills for the project that are not covered elsewhere in the taught programme.

A project Blackboard shell is available as a resource for students, which contains all the necessary project documents/forms, the project calendar, project guidance notes, the list of available projects and supervisor allocations, deadline information, lecture notes etc.

A demonstration is given by the student to their supervisor and second reader towards the end of the module. This enables students to show their understanding of the findings of their work, and to defend what they have done and how they have done it.

Workshop	4 hours
Supervisor meetings	5 hours
Self-directed study	291 hours

Literature

There is no set literature list for the Bachelor project. Students are expected to use literature from the courses they have followed so far and/or find literature relevant for the project by themselves.

Module descriptions Level 6 - Electives

Functional Programming

Module Code: CTEC3704N Credit Value: 30 Credit Level: Academic Level 6

Module Description

Functional Programming (FP) is a mature software development paradigm that is for teaching, research, and industrial software development. Throughout the 2010s, and subsequently, the use of FP has grown significantly in industry, and most mainstream programming languages have adapted to include support for fundamental FP concepts. One reason for the growth of FP is that the paradigm makes it easier to develop code for concurrent execution on multiple-core machines. The removal of shared mutable state reduces the dependence on traditional locks and improves scalability; and the replacement of strict, sequential processing by computations that can be distributed automatically across multiple processes, allows for greater levels of optimisation.

(Pure) functional programming is based upon firm mathematical foundations. The use of referentially transparent functions and immutable data makes it possible for techniques such as equational reasoning to be applied to program fragments that facilitate their translation, automatically, into more efficient formulations. A familiarity with the foundations, and some of the techniques that can be used to reason formally about functional programming constructs, will help the programmer to apply these constructs effectively. Programming without the use of mutable state can present a challenge to a programmer who is already familiar with traditional, non-functional programming techniques. However, the benefits that accrue from this style of software development are worthy of serious investigation by any contemporary software engineer.

This module provides the student with the fundamental concepts of FP and looks at how these have been provided within a modern programming language. The student will gain practical experience, using a modern programming language to solve a practical problem using FP techniques. The core principles are transferrable between functional programming languages.

Outline Content:

- Overview of the Functional Programming (FP) paradigm: how FP differs from other paradigms – in particular, procedural, and object- oriented. Explanation of programming with side effects vs. using functions that have referential transparency, immutability.
- Functions: background theory (lambda calculus); curried arguments; higher-order functions; partial application; syntactic shortcuts (e.g. operator sections); recursion; function composition.
- Types: compile-time type checking; generics; type inference; (meaning of covariance and contravariance constraints if using a programming language that supports these concepts).
- Pattern matching:
- deconstructing tuples, lists, etc. in definitions.
- Lists: standard list functions including, e.g., map, filter, concat, flatMap, fold (reduce), take, drop, zip, unzip.
- List comprehensions: syntax, nested comprehensions, relationship to the higher-order functions map, filter, and flatMap.
- List laws: use of equational reasoning to demonstrate equivalences between certain expressions containing lists: e.g. $\text{map } (f \circ g) = \text{map } f \circ \text{map } g$ (This shows how the mathematical underpinning is used to reason
- Data structures: building new user-defined data types - including renaming types, and constructing new abstract data types.
- Laziness: distinction between lazy evaluation and strict evaluation, and showing how the language of instruction supports both these evaluation strategies. Compare the benefits and drawbacks associated with each approach.
- Threading stateful computation: use of monads as a mechanism for performing stateful computation within a functional context.

- Examples of libraries that exploit the role of higher order functions as combinators. E.g. combinator parsing.
- Examples of larger frameworks that are based upon functional concepts. For example, in Scala, an introduction to the actor model Alternatively, concurrent Haskell could be demonstrated if Haskell were (Akka) for concurrent processes would make a suitable case study. the language used for instruction. Any similar framework that is related to the language used for module delivery would be appropriate in this context.

Learning outcomes

1. Describe and evaluate theoretical and programming language constructs that utilise key FP concepts, including functions as first- class values, higher order functions, currying, partial application, referential transparency, immutability, function composition, strict and lazy evaluation. (Online test, Theory paper)
2. Apply formal reasoning to demonstrate specific properties of selected expressions or functions. (Theory paper)
3. Apply methods for constructing and transforming lists using standard higher order functions and list comprehensions. (Practical)
4. Apply functional programming techniques, including, but not limited to, higher order functions, function composition, and immutable state, to construct a solution to a practical problem. (Practical)
5. Justify the use of specific programming techniques with reference to their adherence to the functional model, and their appropriateness and/or efficiency within a specific application area. (Practical)

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't	Minimum Threshold Mark	Essential Component	Anon. Marking Code
Online Test			20				OPTO1
Practical Assignment			50				OPTO1
Theory Paper			30				OPTO1

Anonymous marking exemption codes: OPTO1: Individually distinct work; OPTO2: Reflection on development of own work; OPTO3: Presentation; OPTO4: individually negotiated work; OPTO5: work placement/experience/assessment

Assessment Notes

Three different forms of assessment are used to cover the learning outcomes. These reflect the combination of theory and practice that constitutes the module content. They also offer a range of assessment types within the module.

The Online Test is time-constrained, and to utilise the question types available on a modern VLE such as Blackboard to assess fundamental concepts from both the theoretical and practical aspects of the module. The mathematical nature of the underlying theory, and the syntax and semantics of a programming language, mean that a series of questions that evaluate such understanding can be constructed in this medium. The phase test may be re-attempted (optionally) within the module delivery period. The higher mark of the two attempts will be recorded.

The Theory Paper is a short exercise consisting of, e.g., two or perhaps three questions, that require the student to apply theoretical methods to demonstrate given properties of specific functional constructs. The answers will probably fill about two pages of A4.

The Practical Assignment allows the student to demonstrate that they can develop a functional program or library, to a given specification. The assignment allows the student to demonstrate the use of functional programming techniques that have been covered on the module. It is important that the code is well-commented, and it is encouraged that an analysis of some aspect of the work is required to be included within the code itself – probably as an extended comment block. The latter will allow the student to discuss, in situ, necessary details regarding efficiency, appropriateness, and adherence to the FP model, as required by the assignment specification, and in fulfilment of LO5.

Expected Methods of Delivery



















Workshops will be used to introduce and demonstrate key practical and theoretical concepts. Practical programming skill will be gained in laboratory sessions. Some sessions may be used for consolidation, revision, and to discuss solutions to practical problems.
















Workshop	30 hours
Practical	42 hours
Self-directed study	64 hours
Consolidation	64 hours
Reading	40 hours
Assessment	60 hours – subsumes estimated time for producing deliverables as per the assessment table

Literature

Articles and videos




Some classic authors appear in this list.









	Why Functional Programming Matters Webpage - Recommended	VIEW ONLINE	<input type="radio"/>	
 John Hughes' article is a real eye-opener. It has become a classic. A must-read for any functional programmer.				
	Keynote: Why Functional Programming Matters - John Hughes, Mary Sheeran (Lambda Days 2017) Audio-visual document - 2017 - Recommended	VIEW ONLINE	<input type="radio"/>	
 Please click View Online and select the video link to watch the YouTube video.				
	How functional programming mattered in National Science Review Article - by Zhenjiang Hu; John Hughes; Meng Wang - 01/09/2015 - Recommended	VIEW ONLINE	<input type="radio"/>	
	Can programming be liberated from the von Neumann style?: a functional style and its algebra of programs in Communications of the ACM Article - by John Backus - 08/1978 - Optional	VIEW ONLINE	<input type="radio"/>	
	The emperor's old clothes in Communications of the ACM Article - by Charles Antony Richard Hoare - 1981-2-1 - Recommended	VIEW ONLINE	<input type="radio"/>	
 An article about programming language design from one of the masters of computer science. It is as fresh today as it was in 1981 when Hoare gave this speech at his Turing Award ceremony. Every computer science student should read this article.				
	No Silver Bullet Essence and Accidents of Software Engineering in Computer Article - by F. Brooks - 04/1987 - Optional	VIEW ONLINE	<input type="radio"/>	
 Brooks argues that Object-orientation is not the "silver bullet". Can this argument hold for other paradigms too?				
	Lambda Calculus - Computerphile - YouTube Audio-visual document - by Graham Hutton - 2017 - Optional	VIEW ONLINE	<input type="radio"/>	

	Lambda Calculus - Computerphile - YouTube Audio-visual document - by Graham Hutton - 2017 - Optional	VIEW ONLINE	<input type="radio"/>	
	What is a Monad? - Computerphile - YouTube Audio-visual document - by Graham Hutton - 2017 - Optional	VIEW ONLINE	<input type="radio"/>	
	Functional Parsing - Computerphile - YouTube Audio-visual document - by Graham Hutton - 2020 - Optional	VIEW ONLINE	<input type="radio"/>	
	Programming Loops vs Recursion - Computerphile - YouTube Audio-visual document - 2017 - Recommended  Explains traditional loops and the need for recursion.	VIEW ONLINE	<input type="radio"/>	
	What on Earth is Recursion? - Computerphile - YouTube Audio-visual document - 2014 - Recommended  Great introduction to recursion using the factorial function.	VIEW ONLINE	<input type="radio"/>	
	Expert to Expert: Rich Hickey and Brian Beckman - Inside Clojure Audio-visual document - 2013 - Optional  Fascinating discussion about Clojure with its author. Interesting points on shared immutable state at about 00:30.	VIEW ONLINE	<input type="radio"/>	

The Scala programming language









Books and resources for using Scala




	The Scala Programming Language Website - Recommended  Where to go for all things Scala	VIEW ONLINE	<input type="radio"/>	
--	---	-----------------------------	-----------------------	--

	Programming in Scala. Martin Odersky, Lex Spoon, Bill Venners Book - by Martin Odersky - 2016 - Optional VIEW ONLINE <input type="radio"/> ⋮
 The book on Scala - written by the author of the Scala language. This is the latest edition.	
	Functional programming patterns in Scala and Clojure: write lean programs for the JVM Book - by Michael Bevilacqua-Linn - c2013 - Optional <input type="radio"/> ⋮
 Looks at patterns of computation in functional programming. It doesn't use Java.	
	Introduction to programming and problem-solving using Scala Book - by Mark C. Lewis; Lisa L. Lacher - 2017 - Optional VIEW ONLINE <input type="radio"/> ⋮
 Quite a technical exposition - but this is a solid book and provides plenty of detail.	
	Scala cookbook: recipes for object-oriented and functional programming Book - by Alvin Alexander - 2021 - Optional VIEW ONLINE <input type="radio"/> ⋮
 Very useful Scala resource	

Haskell language








Books and resources for using Haskell

	Haskell Language Website - Recommended VIEW ONLINE <input type="radio"/> ⋮
 Where to go for all things Haskell	
	Introduction to functional programming using Haskell Book - by Richard Bird; Richard Bird - 1998 - Optional  <input type="radio"/> ⋮
 An outstanding book on Haskell - but quite a tough read for some due to its mathematical approach. Still, it needs to be checked out by the serious Haskell developer.	
	Programming in Haskell Book - by Graham Hutton - 2016 - Optional  <input type="radio"/> ⋮
 Introductory Haskell - a fairly easy read and covers all the bases.	

	Real world Haskell Book - by Bryan O'Sullivan; Donald Bruce Stewart; John Goerzen - 2008, c2009 - Optional	VIEW ONLINE	<input type="radio"/>	⋮
	Parallel and concurrent programming in Haskell Book - by Simon Marlow - 2013 - Optional	VIEW ONLINE	<input type="radio"/>	⋮
	 Simon Marlow shows how to use Haskell for concurrent and parallel programming			

Java 8

Books and resources for Java 8

	Java 8 in action: lambdas, streams, and functional-style programming Book - by Raoul-Gabriel Urma; Mario Fusco; Alan Mycroft - 2015 - Recommended		<input type="radio"/>	⋮
	 Very useful reference for the Java-8 features supporting functional programming			
	Functional Java: A Guide to Lambdas and Functional Programming in Java 8 Book - by Nick Maierano - 2014 - Recommended		<input type="radio"/>	⋮
	 This recently published book is promising - but I have no review of it yet.			
	Functional programming in Java: harnessing the power of Java 8 Lambda expressions Book - by Venkat Subramaniam - 2014 - Optional	VIEW ONLINE	<input type="radio"/>	⋮
	 Quite a technical book. It contains the mechanism for implanting tail call optimisation for recursion.			
	Java SE 8 for programmers Book - by Paul J. Deitel; Harvey M. Deitel - March 2014 - Optional		<input type="radio"/>	⋮
	 A resource for Java 8 programming			
	Functional programming in Java: how to improve your Java programs using functional techniques Book - by Pierre Yves Saumont - 2017 - Recommended		<input type="radio"/>	⋮
	 This looks like an in-depth treatment - it is a relatively new title and I've not read it yet.			

Advanced Web Development

Module Code: CTEC3705N **Credit Value:** 30 **Credit Level:** Academic Level 6

Module Description

The module builds on the outcomes of the level 5 module – Web Application Development.

Many modern computer services are now accessed via the ubiquitous web-browser, and users have come to expect instant and secure access to information on a wide range of platforms. Underpinning these web systems is usually a web application, providing a channel to data stored in databases. However, increasingly the website has also become a point of entry for unauthorized access to stored data. This is often the result of poor web application design and/or implementation.

The module considers how a web application may be designed and implemented in such a way as to reduce the likelihood of unauthorised access to information. This also requires an understanding of the more common forms of browser-based attacks and the coding techniques that can be used to defend against these. The module aims to further develop key concepts and techniques for designing, evaluating and implementing interactive web applications. Designing user interfaces that users can understand immediately and learn easily, enabling them to carry out tasks smoothly and efficiently without excessive effort or stress, is a crucial part of software development. Failures of design can cause technically successful systems to fail in practical use. User interface development frequently eats a large chunk of the development budget, and large projects employ many user interface design and user experience specialists - and systems analysts and technical developers need to be able to talk to them - while many non-specialist software developers find themselves needing to tackle interface design problems.

Designing successful interactive systems involves understanding and applying the key principles of designing usable systems, but also understanding the characteristics of the user populations, the nature of their tasks and environments, and which of the many different aspects of usability are important for this system and this task and considering trade-offs between different aspects of usability. But successful interactive system design goes beyond usability: it involves considering the user experience as a whole including how graphic design and system behaviour influences user emotions, and how the interactive system itself is integrated into the user/customer/client's experience of the entire organization.

Design is only one half of the coin - testing and evaluating prototypes of interactive systems is a critical part of building systems without major usability problems and achieving a good user experience. The module therefore provides a thorough grounding in the rapidly evolving area of full-stack web development, incorporating front-end web technologies, back-end server-side scripting, and data persistence techniques. The module also considers how information can be accessed and presented from remote sources via web-service protocols.

Outline Content:

- Designing for usability
- Designing for human cognitive capabilities
- Evaluation
- Designing for good user experience
- Design principles and user experience
- HTML5, CSS3 and semantic markup
- Asynchronous JavaScript and the fetch API
- Web languages for mobile development
- Version control systems
- Web-technology – e.g. web-service protocols, data persistence techniques, security, etc.

Learning outcomes

1. Ability to apply key general principles of usability, both to guide effective design and to evaluate existing systems. (Online Test, Web project)
2. Ability to propose and specify a suitable system design that aligns with the cognitive capabilities of its target human stakeholders and fits the needs of different users for different tasks and environments (Web Project).
3. Ability to critically appraise the suitability of a range of different techniques for evaluating the usability of interactive systems for particular systems, situations and purposes, and apply the evaluation techniques to produce usability evaluations. (Web Project)
4. Can design and implement a fully standards-compliant, responsive and accessible webtechnology-based application that is impervious to the most common web-based attacks. (Web Project, Online Test)
5. Can design and implement a web application that accesses and displays data from remote web services. (Web Project)

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't	Minimum Threshold Mark	Essential Component	Anon. Marking Code
Online Test			40				AM
Web Project			60				OPTO1

Anonymous marking exemption codes: OPTO1: Individually distinct work; OPTO2: Reflection on development of own work; OPTO3: Presentation; OPTO4: individually negotiated work; OPTO5: work placement/experience/assessment

Assessment Notes

The primary assessment will comprise the development of an interactive web application that can run over the web in a web browser, with server-side processing and server-side data persistence.

The web application submission will be assessed on the prototype planning, application of interaction design principles, quality of code, knowledgeable application of all utilised technologies, and adaptation of taught materials. It is to be submitted via a Version Control System, such as GitHub, with regular commits during the module. The web project will allow the student to demonstrate an understanding of the analysis, design and implementation of a secure web application, based upon a commonly used development framework. The web project will be assessed by a mandatory viva voce.

The phase test will be delivered on-line. It will assess the students' knowledge and understanding of current web applications, issues of architecture and security, as well as aspects of human computer interaction.

Expected Methods of Delivery

The module will comprise lectures and reading about different aspects of human computer interaction, and a combination of paper-based tutorial and computer-based lab activities practising the use of a range of different techniques for understanding requirements, developing designs and performing usability evaluations. Intensive teaching through the lectures will be backed up by practical demonstrations and laboratory work.

A series of short tutorial questions will be used to reinforce the lecture content. The web project will require the students to develop a demonstrably secure webapplication, Implementing the techniques they have learned throughout the teaching period.

Students will have access to a localhost development environment (web server, database server, development IDE) upon which their web applications will be implemented. Completed and tested web applications will be uploaded to relevant

public facing servers for assessment. Students will also require access to remote web-service APIs such as the EE M2M SMS server.

Workshops will be used to introduce and demonstrate key practical and theoretical concepts. Practical programming skill will be gained in regular laboratory sessions. Some sessions may be used for consolidation, revision, and to discuss solutions to practical problems.

Workshop	24 hours
Practical	42 hours
Self-directed study	76 hours
Consolidation	68 hours
Reading	30 hours
Assessment	60 hours – subsumes estimated time for producing deliverables as per the assessment table

Literature



The Web application hacker's handbook: finding and exploiting security flaws

Book - by Dafydd Stuttard; Marcus Pinto - c2011 - Essential

[VIEW ONLINE](#)



Seven deadliest Web application attacks

Book - by Mike Shema; Adam Ely - c2010 - Recommended

[VIEW ONLINE](#)



Web Application Security: Exploitation and Countermeasures for Modern Web Applications

Book - by Andrew Hoffman - 2020 - Recommended

[VIEW ONLINE](#)



Hands on hacking

Book - by Matthew Hickey; James McAlonan; Jennifer Arcuri - 2020 - Optional

[VIEW ONLINE](#)



The basics of hacking and penetration testing: ethical hacking and penetration testing made easy

Book - by Pat Engebretson - c2013 - Optional

[VIEW ONLINE](#)



Red team development and operations: A practical guide

Book - by James Tubberville - 2020 - Optional



PHP and MySQL Web Development

Book - by Luke Welling - 2016 - Recommended

[VIEW ONLINE](#)



php [architect] Magazine

Journal - Recommended

[VIEW ONLINE](#)



The library is unable to subscribe to this magazine however students can use the inter-library loan service to obtain materials (where possible). Individuals can also purchase a subscription.

Fuzzy Logic and Inference Systems

Module Code: IMAT3722N Credit Value: 30 Credit Level: Academic Level 6

Module Description

Fuzzy logic is a mathematical model for handling uncertainty. It is able to provide a means in order to successfully inference from abstract and subjective notions. Fuzzy logic adopts the perspective that the world and humanistic understanding are inherently vague and not precise. Concepts like that of; hot; cold; near; far; and other forms of expressive language where precise values are not given, are extremely difficult to model when universal understanding of such concepts are non-existent. What is beautiful to some, may not be beautiful to others; concepts can have different meanings to different people. Fuzzy logic and fuzzy theory provide the tools in order to fuzzify abstract notions so that they can be modelled and inferenced in a humanist manner, such that they can be understood by a larger population.

The utilisation of fuzzy components ultimately allows for the creation of a fuzzy inference system, a system based on the thought and decision making processes adopted by human cognition. The use of fuzzy sets; a fuzzy inference engine; and knowledge base, creates for an incredibly powerful tool. Fuzzy inference systems are extremely versatile and can be deployed on many different domains and have been utilised by industry in many different sectors.

This module will present the core and fundamental concepts of fuzzy logic, from theory to application. The understanding developed will allow for a fuzzy perspective to be adopted, understood and appreciated. The ability to create specialised fuzzy inference systems will be achieved and so too will the ability to articulate on thought processes needed to create such systems. A comprehensive understanding of fuzzy logic, theory and application will also be covered. The module will also investigate the literature on fuzzy and its areas of application to further instil the applicability of a fuzzy approach and the ethical implications of modelling subjective perception based uncertainty.

The module will provide a comprehensive introduction to fuzzy logic in addition to the following:

- The concepts of uncertainty, vagueness and imprecision
- Set theory and the notion of a fuzzy set
- Basic operations on fuzzy sets; intersection; union; complement
- Fuzzy inference systems; Mamdani, TSK, zeroorder, first-order
- Type-2 fuzzy logic; interval type-2 fuzzy logic; generalised type-2 fuzzy logic
- Fuzzy logic applications
- The use of MATLAB for creating fuzzy inference systems
- Ethical considerations when considering cognitive subjective modelling
- Forwards chaining inference; backwards chaining inference
- Knowledge acquisition
- Knowledge representation

Learning outcomes

1. To propose a solution via the creation of a fully functioning fuzzy inference system based on a domain chosen by the student (Coursework)
2. Acquire an in-depth understanding of a fuzzy perspective when modelling abstract notions (Coursework)
3. Demonstrate a sound comprehensive and qualitative understanding and the ability to locate and use appropriate findings from the scientific literature (Coursework)
4. To be able to articulate, critique and evaluate on the decision making processes adopted when justifying the final configured system (Coursework).

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't	Minimum Threshold Mark	Essential Component	Anon. Marking Code
Other Coursework			100				OPTO1
Anonymous marking exemption codes: OPTO1: Individually distinct work; OPTO2: Reflection on development of own work; OPTO3: Presentation; OPTO4: individually negotiated work; OPTO5: work placement/experience/assessment							

Assessment Notes

The coursework is a SINGLE assessed component worth 100% of the module mark. 1) The students will be tasked with creating a fuzzy inference system. 2) The students will be expected to submit a detailed report of their fuzzy inference systems. The coding of the system in terms of syntax or elegance does not contribute to the coursework mark. The coursework mark is dependent on the quality of technical report in accordance to the marking criteria.

The coursework will be used to assess their ability to implement these ideas in a fully functioning fuzzy inference system. The accompanying report will not only document the system in detail, but also provides the student the platform to express their understating of fuzzy logic and its application.

The coursework assessment will be undertaken throughout the various practical lab sessions, where the students can contribute further to their systems and technical reports with contact time with the module tutor. This allow formative feedback from tutor to students on their work to occur throughout the module.

Expected Methods of Delivery

The module will make heavy use of practical lab work, where the students will be able to refine their understanding of the topics covered. The labs will also allow for the students to start on the coursework, which will be evolve after each session, incorporating more fuzzy theory.

Workshop	13 hours
Practical	52 hours
Self-directed study	19 hours
Assessment	216 hours

Literature

- Peckol, J.K., 2021. Introduction to fuzzy logic. John Wiley & Sons. ISBN: 978-1119772613 (Recommended)
- Singh, H. and Lone, Y.A., 2020. Deep neuro-fuzzy systems with python. Apress, Berkeley, ISBN:978-1484253601 (Recommended)
- Keller, J.M., Liu, D. and Fogel, D.B., 2016. Fundamentals of computational intelligence: neural networks, fuzzy systems, and evolutionary computation. John Wiley & Sons. ISBN: 978-1119214342 (Optional)
- Mendel, J., Hagrass, H., Tan, W.W., Melek, W.W. and Ying, H., 2014. Introduction to type-2 fuzzy logic control: theory and applications. John Wiley & Sons. ISBN: 978-1118278390 (Optional)

Privacy and Data Protection

Module Code: IMAT3711N **Credit Value:** 30 **Credit Level:** Academic Level 6

Module Description

There continues to be a growth of databases holding personal and other sensitive information in multiple formats including text, pictures and sound. The scale of data collected, its type and the scale and speed of data exchange have all changed with the advent of ICT. Whilst the potential to breach privacy continues to increase organisations are subjected to a considerable amount of legislation governing privacy and data protection.

This module examines the balance between maintaining business effectiveness, legal compliance and professional practice in the field of IT/IS. The module will:

1. address the legal, social and technological aspects of privacy and data protection,
2. consider privacy enabling technologies and privacy invasive technologies,
3. identify and evaluate the role of the computer professional in providing privacy and data protection.

Care will be taken in ensuring perspectives from different cultures and countries are highlighted and considered in the light of global information systems.

Indicative topics:

- Introduction to the module, privacy and data protection (DP).
- Undertaking literature research.
- Privacy Theory.
- Privacy - the global perspective.
- Privacy as a consumer issue.
- Privacy as a corporate issue.
- Surveillance Theory.
- Surveillance Technology: Types and applications
- Data Protection and its relationship to privacy.
- The current data protection framework.
- Data Protection - Global perspectives.
- Privacy and DP in Information System Development.
- Data Protection and its impact on the IS professional.
- Privacy Enhancing Technologies.
- Technical and Managerial Responses to Privacy and DP.

Learning outcomes

1. Display a systematic understanding of the concepts of privacy and data protection within a multi-cultural and multidisciplinary context.
2. Using appropriate techniques of analysis and enquiry within this discipline recognise and evaluate current and future threats to privacy.
3. Outline and evaluate IS professional practice with regard to perceived professional and social responsibilities to employers and data subjects. Drawing upon current research in the field this will lead to an appreciation of the uncertainty and ambiguity that surrounds the role and responsibility of IS staff in this field.
4. Research into an area of privacy and or data protection; integrate complex and sometimes conflicting ideas into a coherent analysis that demonstrates integrative, synoptic and analytical skills in accordance with the Framework for Higher Education Qualifications guidelines/indicators [a,b,c&d]

Assessment

Assessment	Unit Of Duration	Duration	Individ'n Weighting	Final Assm't	Minimum Threshold Mark	Essential Component	Anon. Marking Code
Other Coursework 1			50				AM
Other Coursework 2			50				OPTO3
Anonymous marking exemption codes: OPTO1: Individually distinct work; OPTO2: Reflection on development of own work; OPTO3: Presentation; OPTO4: individually negotiated work; OPTO5: work placement/experience/assessment							

Assessment Notes

Students will complete weekly reading preparation which develop their reading, writing and critical thinking skills. This, alongside the seminar discussions, will lay the foundation for the coursework. The debate reflection article is to critically analyse and discuss issues and dynamics of the PDP debate topic. The topic for discussion is decided in a poll by students. Students will also undertake a group video presentation.

Expected Methods of Delivery

A range of learning strategies will be used to facilitate student learning.

Sessions will be used to introduce major topics drawing together material from a variety of sources. Students will be encouraged to read widely, using the reading list as a starting point and library resources and the internet for further information.

The assignment will provide the opportunity for students to develop their literature searching skills and their ability to integrate complex and often conflicting ideas into a coherent summary that will form the basis of their essay. The PDP debate is arranged through a democratic process whereby student suggest motions which are then voted upon to decide on the debate topic. The module website will provide essential module information such as a week by week teaching plan, the reading list and assessment information. It will also be used to direct the students to useful sources of information and for essential updates such as drawing the students' attention to recently published relevant papers.

Workshop	40 hours
Seminar	90 hours
Self-directed study	90 hours
Assessment	80 hours

Literature

Web Sites

Trusted Web Resources



The Futuremakers podcast | Research | University of Oxford

Webpage - Recommended

[VIEW ONLINE](#)

🎓 Futuremakers podcast, devoted to issues in AI Ethics



Information Commissioner's Office

Website - 2018 - Recommended

[VIEW ONLINE](#)

🎓 This is the website of the Information commissioner and is a key resource for understanding privacy and data protection.



Philosophy Bites

Website - Recommended

[VIEW ONLINE](#)

🎓 Podcasts with a wide series of discussions with philosophers covering a huge range of topics



Alphabetical List of Resources | Ethics for Artificial Intelligence

Webpage - Recommended

[VIEW ONLINE](#)

🎓 Some very useful links to codes of ethics and other resources



BBC Radio 4 - The Public Philosopher - Downloads

Webpage - Recommended

[VIEW ONLINE](#)

🎓 Podcasts to explore contemporary philosophical issues

Books and Articles



The age of surveillance capitalism: the fight for a human future at the new frontier of power

Book - by Shoshana Zuboff - 2019 - Recommended

[VIEW ONLINE](#)

📖 An important book that highlights how behavioural surplus is utilised by tech companies



Annotated Turing: A Guided Tour Through Alan Turing's Historic Paper on Computability and the Turing Machine

Book - by Charles Petzold - 2008 - Recommended



Ethics: a very short introduction

Book - by Simon Blackburn - 2003 - Recommended

[VIEW ONLINE](#)



The philosophy gym: 25 short adventures in thinking

Book - by Stephen Law - 2004 - Recommended



Privacy: a very short introduction

Book - by Raymond Wacks - 2015 - Recommended



The road to conscious machines: the story of AI











Book - by Michael J. Wooldridge - 2020 - Recommended














Life after privacy: reclaiming democracy in a surveillance society

Book - by Firmin DeBrabander - 2020 - Optional

[VIEW ONLINE](#)

	Privacy by Design — Principles of Privacy-Aware Ubiquitous Systems in Ubicomp 2001: ubiquitous computing : International Conference, Atlanta, Georgia, USA, September 30-October 2, 2001 : proceedings Chapter - by Marc Langheinrich - Recommended	VIEW ONLINE
	Surveillance & society Journal - Recommended  An online peer reviewed journal on surveillance and society	VIEW ONLINE
	How to read a paper in ACM SIGCOMM Computer Communication Review Article - by S. Keshav - 20/07/2007 - Essential	VIEW ONLINE
	Beyond Location: Data Security in the 21st Century. in Communications of the ACM Article - by Devan Desai - Jan 2013 - Recommended	VIEW ONLINE
	'Code': Privacy's death or saviour? in International Review of Law, Computers & Technology Article - by Ronald Leenes; Bert-Jaap Koops - Nov 2005 - Recommended	VIEW ONLINE
	Data mining, national security, privacy and civil liberties in ACM SIGKDD Explorations Newsletter Article - by Bhavani Thuraisingham - 01/12/2002 - Recommended	VIEW ONLINE
	Employees' privacy vs. employers' security: Can they be balanced? in Telematics and Informatics Article - by Lilian Mitrou; Maria Karyda - 2006-8 - Recommended	VIEW ONLINE
	New Media and the power politics of sousveillance in a surveillance-dominated world. in Surveillance & Society Article - by Steve Mann; Joseph Ferenbok - 16/07/2013 - Recommended	VIEW ONLINE
	Organization, Surveillance and the Body: Towards a Politics of Resistance in Organization Article - by Kirstie Ball - 01/2005 - Recommended	VIEW ONLINE

	Potential harms, anonymization and the right to withdraw consent to biobank research in European Journal of Human Genetics Article - by Stefan Eriksson; Gert Helgesson - 2005-9 - Recommended	VIEW ONLINE
	Privacy implications of online consumer-activity data: an empirical study in Second Workshop on Society, Privacy and the Semantic Web - Policy and Technology Article - by Keerthi Thomas - Oct 2014 - Recommended	VIEW ONLINE
	A privacy paradox: Social networking in the United States in First Monday Article - by Susan B. Barnes - 04/09/2006 - Recommended  Please click 'View Online' and then click HTML for full access.	VIEW ONLINE
	Privacy, Risk and Personal Health Monitoring in ETHICOMP 2013 Conference Proceedings Article - by Brent Daniel Mittelstadt - 2013 - Recommended  Please click 'View Online' and scroll down the list until you find the article 'Privacy, risk and personal health monitoring'.	VIEW ONLINE
	Privacy risk models for designing privacy-sensitive ubiquitous computing systems in Proceedings of the 2004 conference on Designing interactive systems processes, practices, methods, and techniques - DIS '04 Article - by Jason I. Hong; Jennifer D. Ng; Scott Lederer; James A. Landay - 2004 - Recommended	VIEW ONLINE
	Surveillance, Snowden, and Big Data: Capacities, consequences, critique in Big Data & Society Article - by David Lyon - 10/07/2014 - Recommended	VIEW ONLINE
	The surveillant assemblage in British Journal of Sociology Article - by Kevin D. Haggerty; Richard V. Ericson - 2000-12-1 - Recommended	VIEW ONLINE
	Towards a theory of privacy in the information age in ACM SIGCAS Computers and Society Article - by James H. Moor - 01/09/1997 - Recommended	VIEW ONLINE
	What's happened to PETs? in Information Security Technical Report Article - by Robin Wilton - 2009-8 - Recommended	VIEW ONLINE

Assessment Methods

Assessment Methods 2023/2024 BSc in Computer Science

Modules are assessed in many different ways as outlined in the module description for each particular module, but here are some of the most common methods of assessment:

Essay	a written assignment based on a set question (or choice of questions) with a word limit.
Case Study	an analysis of a current issue/case surrounding the associated topics delivered on the module.
Report	a structured assignment using headings and sub-headings used to look at a particular problem or issue and make recommendations within a word limit. This could be an individual piece of work or group work.
Portfolio	a written piece of work where students are asked to reflect on their development and experience and what they have learned from it.
Phase Test	a shorter test (usually multi-choice or short answers) which takes place under exam conditions – typically online.
Structured Applications Proposal	a way of presenting a development project – typically in a form used as decision presentation for top-management
Practical	an individually distinct work where the student demonstrates ability to do practical work related to the course subject, e.g., writing working program code in a programming course
Presentation	this can be in groups or done individually and usually takes place in a classroom or lecture theatre using visual aids such as PowerPoint.

Emphasis is on formative feedback when it comes to coursework to enable the student to critically reflect on own work and integrate feedback in future assignments.

Code	Module Title	Learning Outcome	Assessment
CTEC1701N	Database Design and Implementation	<ol style="list-style-type: none">1. Derive a suitably normalised database design that reflects the business requirements of a typical corporate scenario. (Online Test 1 & Online Test 2)2. Implement a database design via a Relational Database Management System addressing business use cases, information integrity and information security. (Online Test 1 & Online Test 2)3. Acquire an understanding of big data and differentiate structured, semi-structured and unstructured data. (Online Test 2)	Phase test 1 (40%) Phase test 2 (40%) Report (20%)

		<ol style="list-style-type: none"> 4. Demonstrate a sound understanding and ability to locate and use primary and secondary sources. (Report) 5. Identify and implement privacy tools in the design of database. (Report) 	
CTEC1702N	Fundamental Concepts of Computer Science	<ol style="list-style-type: none"> 1. Construct and manipulate propositional logic statements. (Online Test) 2. Construct and manipulate set-theoretical mathematical objects. (Online Test) 3. Undertake probabilistic calculations and perform hypothesis testing. (Online Test) 4. Demonstrate an understanding of the different kinds of requirements software systems need to meet, and the ability to formulate appropriate requirements. (Portfolio) 5. Demonstrate an outline understanding of the software development lifecycle including the basic principles of user centered design and awareness of contrasting approaches to structuring the development process. (Portfolio) 6. Demonstrate understanding of key theories and current ethical concerns in information systems and computing, including professionalism and codes of ethics. (Case study analysis) 	Phase test 1 (50%) Portfolio (30%) Case study (20%)
CTEC1703N	Computer Programming	<ol style="list-style-type: none"> 1. Design a program to a given specification using the control and data structure abstractions provided by a contemporary programming language (Online Test 1 & Online Test 2) 2. Deploy trusted software design techniques in the construction of a computer program (Online Test 2) 3. Analyse a problem and produce a program specification (Practical) 4. Apply relevant software testing techniques to verify the components of a computer program and ensure it is trustworthy (Practical) 	Practical (40%) Phase test 1 (30%) Phase test 2 (30%)
CTEC1704N	Operating Systems and Networks	<ol style="list-style-type: none"> 1. Demonstrate the theoretical and practical understanding of computer hardware, operating systems and networks. (Phase Test 1, 2 and 3) 2. Relate the abstract concepts of logic and number systems to their concrete representation on real machines. (Phase Test 1, 2 and 3) 3. Identify the security risks in common configurations of computer operating systems and suggest appropriate mitigations. (Phase Test 1, 2 and 3) 	Phase test 1 (30%) Phase test 2 (30%) Phase test 3 (40%)

		4. Identify the security risks in common configurations of computer networks and suggest appropriate mitigations. (Phase Test 1, 2 and 3)	
CTEC2710N	Object Oriented Design and Development	<ol style="list-style-type: none"> 1. Use Java to implement standard object-oriented designs given in UML. (Portfolio) 2. Design and develop trustworthy software in the context of an object-oriented language. (Portfolio) 3. Design and develop applications with an emphasis on quality, maintainability, correctness and robustness, enhancing their trustworthiness. (Programming specification) 4. Make effective use of the Java Software Development Kit Application Programming Interfaces. (Programming specification) 	Portfolio (50%) Programming specification (50%)
CTEC2711N	Data Structures and Algorithms	<ol style="list-style-type: none"> 1. Explain and implement a variety of classical data structures (Practical and Phase Test 1) 2. Apply classic sequential algorithms for searching and sorting (Phase Test 1) 3. Analyse specific programming language and library support for the development of sequential and concurrent programs (Practical) 4. Apply standard concurrent algorithm design patterns (Practical and Phase Test 2) 5. Explain formal notation in specific contexts (Phase Test 2) 	Phase test 1 (25%) Phase test 2 (25%) Practical (50%)
CTEC2712N	Web Application Development	<ol style="list-style-type: none"> 1. Demonstrate a critical understanding of information architecture, user interface design and usability principles (phase test and web project). 2. Effectively apply knowledge of client-side web development technologies (web project). 3. Effectively apply knowledge of server-side web development technologies (web project). 4. Demonstrate a critical understanding of security protocols in web applications (phase test and web project). 	Phase test 1 (40%) Web Project (60%)
CTEC2713N	Agile Development Team Project	<ol style="list-style-type: none"> 1. Work collaboratively with both tutor and peers (Portfolio) 2. Produce code-based solutions to problems generating suitable documentation (Portfolio) 3. Apply good practice in code design (Portfolio) 4. Find answers to programming problems (Portfolio) 5. Research into an area of computer ethics and integrate different ideas into a coherent analysis of a system, and demonstrate critical and analytical skills in understanding the risks associated with managing software projects (Case study analysis) 	Portfolio (80%) Case study (20%)
CTEC3701N	Software Development: Methods and Standards	<ol style="list-style-type: none"> 1. Evaluate and select a methodology for developing a common application. (Structured Proposal) 2. Assess the consequences of scaling, integration and security in an application development. (Structured 	Structured Applications and Methodologies (60%) Case Study Analysis (20%) Report (20%)

		<p>Proposal)</p> <ol style="list-style-type: none"> Devise a plan for the adoption of an appropriate standard within an application development environment. (Structured Proposal) Recognise and evaluate current and future ethical issues surrounding the application of ICT. (Case study analysis) Identify, critically analyse and apply laws, regulations and standards as found in the U.K. thereby demonstrating a sound understanding and ability to locate and use primary and secondary sources. (Report) 	
CTEC3702N	Big Data and Machine Learning	<ol style="list-style-type: none"> Critically evaluate Big Data, where it comes from and what the opportunities and challenges are, including storage infrastructures and their pros and cons (Phase Test) Develop solutions for data exploration, visualization and analytics using suitable machine learning frameworks such as Apache Spark. (Problem Specification) Identify and describe classification and clustering algorithms and their application areas (Phase Test) Select and apply the suite of machine learning algorithms available in a machine learning library e.g., Spark Machine Learning Library, to develop solutions to machine learning problems. (Problem Specification) Research into an area of computer ethics, integrate complex and sometimes conflicting ideas into a coherent analysis that demonstrates integrative, synoptic, and analytical skills. (Case Study Analysis) 	<p>Phase Test (30%)</p> <p>Problem Specification (50%)</p> <p>Case Study Analysis (20%)</p>
CTEC3451N	Development Project	<ol style="list-style-type: none"> Apply theoretical and practical concepts from the programme of study to the construction of a solution to a practical problem. (Initial and Final) Carry out research and analysis to support the project requirements. (Initial) Plan and selfmanage the work. (Initial and Final) Assess the potential global impact of the work (Initial) Present the work using a report in which the process and product are described, analysed, and critically evaluated. (Final) Present and defend the work in a formal demonstration (Final) 	<p>Project management (10%)</p> <p>Final Submission (90%)</p>
CTEC3704N	Functional Programming	<ol style="list-style-type: none"> Describe and evaluate theoretical and programming language constructs that utilise key FP concepts, including functions as first- class values, higher order functions, currying, partial application, referential transparency, immutability, function composition, strict and lazy evaluation. 	<p>Phase test (20%)</p> <p>Theory paper (30%)</p> <p>Practical assignment (50%)</p>

		<p>(Online test, Theory paper)</p> <ol style="list-style-type: none"> 2. Apply formal reasoning to demonstrate specific properties of selected expressions or functions. (Theory paper) 3. Apply methods for constructing and transforming lists using standard higher order functions and list comprehensions. (Practical) 4. Apply functional programming techniques, including, but not limited to, higher order functions, function composition, and immutable state, to construct a solution to a practical problem. (Practical) 5. Justify the use of specific programming techniques with reference to their adherence to the functional model, and their appropriateness and/or efficiency within a specific application area. (Practical) 	
CTEC3705N	Advanced Web Development	<ol style="list-style-type: none"> 1. Ability to apply key general principles of usability, both to guide effective design and to evaluate existing systems. (Online Test, Web project) 2. Ability to propose and specify a suitable system design that aligns with the cognitive capabilities of its target human stakeholders and fits the needs of different users for different tasks and environments (Web Project). 3. Ability to critically appraise the suitability of a range of different techniques for evaluating the usability of interactive systems for particular systems, situations and purposes, and apply the evaluation techniques to produce usability evaluations. (Web Project) 4. Can design and implement a fully standards-compliant, responsive and accessible webtechnology-based application that is impervious to the most common web-based attacks. (Web Project, Online Test) 5. Can design and implement a web application that accesses and displays data from remote web services. (Web Project) 	Phase test (40%) Web project (60%)
IMAT3722N	Fuzzy Logic and Inference Systems	<ol style="list-style-type: none"> 1. To propose a solution via the creation of a fully functioning fuzzy inference system based on a domain chosen by the student (Coursework) 2. Acquire an in-depth understanding of a fuzzy perspective when modelling abstract notions (Coursework) 3. Demonstrate a sound comprehensive and qualitative understanding and the ability to locate and use appropriate findings from the scientific literature (Coursework) 4. To be able to articulate, critique and evaluate on the decision making processes adopted when 	Other coursework (100%)

		justifying the final configured system (Coursework).	
IMAT3711N	Privacy and Data Protection	<ol style="list-style-type: none"> 1. Display a systematic understanding of the concepts of privacy and data protection within a multi-cultural and multidisciplinary context. 2. Using appropriate techniques of analysis and enquiry within this discipline recognise and evaluate current and future threats to privacy. 3. Outline and evaluate IS professional practice with regard to perceived professional and social responsibilities to employers and data subjects. Drawing upon current research in the field this will lead to an appreciation of the uncertainty and ambiguity that surrounds the role and responsibility of IS staff in this field. 4. Research into an area of privacy and or data protection; integrate complex and sometimes conflicting ideas into a coherent analysis that demonstrates integrative, synoptic and analytical skills in accordance with the Framework for Higher Education's Qualifications guidelines/indicators [a,b,c&d] 	<p>Other coursework 1 (50%)</p> <p>Other coursework 2 (50%)</p>

Overview of academic prerequisites for later modules

Table 1 below shows in a schematic form which modules include learnings from which previous taught modules. So, when it for the level 5 module “Object Oriented Design and Development” states “L4: Fundamental Concepts of Computer Science” and “L4: Computer Programming” it means that the two level 4 modules are part of the fundament for the level 5 module and that students therefore use important knowledge from the level 4 modules in this module.

Level 4	Level 5	Level 6
Database Design and Implementation	Object Oriented Design and Development <ul style="list-style-type: none"> L4: Fundamental Concepts of Computer Science L4: Computer Programming 	Software Development: Methods and Standards <ul style="list-style-type: none"> L4: Fundamental Concepts of Computer Science L5: Object Oriented Design and Development
Fundamental Concepts of Computer Science	Data Structures and Algorithms <ul style="list-style-type: none"> L4: Database Design and Implementation L4: Fundamental Concepts of Computer Science L4: Computer Programming L5: Object Oriented Design and Development 	Big Data and Machine Learning <ul style="list-style-type: none"> L4: Fundamental Concepts of Computer Science L5: Data Structures and Algorithms L6: Software Development: Methods and Standards
Computer Programming	Web Application Development <ul style="list-style-type: none"> L4: Database Design and Implementation L4: Computer Programming L4: Operating Systems and Networks L5: Object Oriented Design and Development 	Development Project <ul style="list-style-type: none"> All relevant previous modules
Operating Systems and Networks	Agile Development Team Project <ul style="list-style-type: none"> In principle all the previous taught modules dependent on the nature of the project 	Functional Programming <ul style="list-style-type: none"> L4: Computer Programming
		Advanced Web Development <ul style="list-style-type: none"> L5: Web Application Development
		Fuzzy Logic and Inference Systems <ul style="list-style-type: none"> L4: Fundamental Concepts of Computer Science
		Privacy and Data Protection <ul style="list-style-type: none"> L4: Fundamental Concepts of Computer Science L6: Big data and Machine Learning

Table 1: How the academic prerequisites for later modules are expected to be fulfilled by the previous modules

The modules at level 4 are all basic modules and are sufficiently self-contained to not be dependent on each other. Therefore, no dependencies are listed at level 4.