Master of Science in Computer Science

Handbook

April 2026

Introduction

Master of Science in Computer Science

This degree is designed for individuals who wish to enhance their knowledge of computer science and its various applications across different fields of employment. It targets those who will have responsibility for planning, organising, and directing technological operations, as well as professionals seeking to develop specialised expertise in areas such as Artificial Intelligence and Machine Learning, Software Engineering, Computer Systems Engineering, or Data Science and Analytics.

Entry requirements

Education Requirements

Candidates who apply for this course will ordinarily have an EQF 6 level degree. Students without a technical background (either in degree or through work experience) may be encouraged to take additional coursework. Candidates can be asked to demonstrate sufficient knowledge and skills through technical certifications or practical experience.

Language Requirements

English language proficiency is required. Where English is not the candidate's first language, they will need to demonstrate proficiency at an appropriate level.

Instructional design

Teaching: Teaching is delivered through synchronous sessions with an instructor, supplemented by supervised activities, scholarly articles and study materials in the VLE, asynchronous lectures, and communication by chat forum.

Assessment: Assessment combines formative assignments (typically 40% of the mark) and summative assignments (typically 60%), providing continuous and final evaluation of student progress throughout each module.

Degree structure

The degree consists of 56 optional modules totalling 310 ECTS credits, all at EQF 7.

Module	ECTS	Level
Data Structures	5	EQF 7
Design and Analysis of Algorithms	5	EQF 7

Relational Databases	5	EQF 7
Numerical Programming in Python	5	EQF 7
Applied Statistics	5	EQF 7
Introduction to Machine Learning	5	EQF 7
High dimensional Data Analysis	5	EQF 7
Advanced Machine Learning	5	EQF 7
Distributed Machine Learning	5	EQF 7
Introduction to Deep Learning	5	EQF 7
Deep Learning for Computer Vision	5	EQF 7
Deep Learning for Natural Language Processing (NLP)	5	EQF 7
Productionization of Machine Learning Systems	5	EQF 7
System Design	5	EQF 7
DevOps	5	EQF 7
Front end UI/UX development	5	EQF 7
JavaScript	5	EQF 7
Front End Development	5	EQF 7
Back End Development	5	EQF 7
Foundations of Cloud Computing	5	EQF 7

Advanced Backend Development	5	EQF 7
Distributed Cloud Computing	5	EQF 7
Advanced Cloud Computing	5	EQF 7
NoSQL Cloud Datastores	5	EQF 7
Design Patterns	5	EQF 7
Capstone: Advanced Applied Computer Science	30	EQF 7
Introduction to Computer Programming: Part 1	5	EQF 7
Introduction to Problem-Solving Techniques: Part 1	5	EQF 7
Introduction to Problem-Solving Techniques: Part 2	5	EQF 7
Mathematics for Computer Science	5	EQF 7
Advanced Algorithms	5	EQF 7
Computer Systems and Their Fundamentals	5	EQF 7
Low-Level Design and Design Patterns	5	EQF 7
Practical Software Engineering	5	EQF 7
Distributed Systems with High-Level System Design	5	EQF 7
Data Visualisation Tools	5	EQF 7
Power BI for Data Analysis and Exploration	5	EQF 7
Statistical Programming	5	EQF 7

Spreadsheets for Data Understanding	5	EQF 7
Advanced Python Programming	5	EQF 7
Foundations of Machine Learning	5	EQF 7
Business Case Studies	5	EQF 7
Studies in Data Science and Data Analytics	5	EQF 7
Further Studies in Data Science and Data Analytics	5	EQF 7
SQL for Data Analytics	5	EQF 7
Product Analytics	5	EQF 7
Data Engineering	5	EQF 7
Product Management for Software Engineers	5	EQF 7
Applied Computer Science Project	10	EQF 7
Introduction to Computer Programming, Part 2	5	EQF 7
Web Design	5	EQF 7
Advanced JavaScript	5	EQF 7
Mobile App Design and Development	5	EQF 7
Software Development and QA	5	EQF 7
UX/UI Design	5	EQF 7
Career Strategies and Soft Skills for IT Professionals	5	EQF 7

Module Descriptions

1. Data Structures

This course is aimed to build a strong foundational knowledge of data structures (DS) used extensively in computing. The module starts with introducing time and space complexity notations and estimating computational complexity. The key data structures covered are: arrays, linked lists, trees, heaps, hash tables, and graphs. After covering each data structure, the module goes on to explain and motivate applications that are based on each data structure.

Learning Outcomes

1. Create synthetic contextualized discussions of key issues related to Data Structures and the different approaches to their implementation
2. Apply a professional and scholarly approach to research problems pertaining to Data Structures and their implementation
3. Efficiently manage interdisciplinary issues that arise in connection to the design and implementation of data structures
4. Demonstrate self-direction in research and originality in solutions developed in the field of data structures
5. Act autonomously in identifying research problems and solutions related to data structures and their implementation
6. Solve problems and be prepared to take leadership decisions related to design choices and their trade-offs in the implementation of data structures

2. Design and Analysis of Algorithms

This is a foundational and mandatory course which aims to build student's ability to apply various algorithmic design methods to provide an optimal solution to computational problems. This course starts by introducing the concept of algorithmic design and efficiency, explains different strategies for algorithm design, compares the pros and cons of different design strategies and then reinforces the concepts by covering popular algorithms from each design paradigm.

Learning Outcomes

1. Create synthetic contextualized discussions of key issues related to design and analysis of algorithms to provide solutions to computational problems
2. Apply a professional and scholarly approach to research problems pertaining to design and analysis of algorithms
3. Efficiently manage interdisciplinary issues that arise in connection to the design and analysis of algorithms and algorithms in general
4. Demonstrate self-direction in research and originality in developing solutions to algorithmic problems
5. Act autonomously in identifying research problems and solutions related to optimal and heuristic algorithms
6. Solve problems and be prepared to take leadership decisions related to the design and analysis of algorithms in the context of software engineering projects

3. Relational Databases

This is a core and foundational course which aims to equip the student with the ability to model, design, implement and query relational database systems for real-world data storage & processing needs. It starts by introducing the relational data model and its mathematical foundations, then builds up practical skills in SQL, query optimisation and database design. The module provides a hands-on experience with database management systems and equips students with the ability to work with relational databases professionally.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to Relational Databases
2. Apply a professional and scholarly approach to research problems pertaining to Relational Databases
3. Efficiently manage interdisciplinary issues that arise in connection to implementation and querying of a relational database management system
4. Demonstrate self-direction in research and originality in solutions developed in the field of database management system design
5. Act autonomously in identifying research problems and solutions related to relational database management systems
6. Solve problems and be prepared to take leadership decisions related to the use of relational database management systems

4. Numerical Programming in Python

This course helps students translate mathematical problems and solutions into Python code. Students will learn to apply statistical and mathematical concepts with minimal abstraction, directly to any number of real-world scientific and engineering problems. Students will gain experience in the numerical solution of differential equations, optimization, and many more basic problems of scientific computing; in a hands-on way, writing code from first principles.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to Numerical Programming in Python
2. Apply a professional and scholarly approach to research problems pertaining to Numerical Programming in Python
3. Efficiently manage interdisciplinary issues that arise in connection to the implementation of various numerical programming algorithms
4. Demonstrate self-direction in research and originality in solutions developed in the field of numerical programming
5. Act autonomously in identifying research problems and solutions related to numerical algorithms
6. Solve problems and be prepared to take leadership decisions related to the implementation of advanced numerical algorithms using Python

5. Applied Statistics

This course introduces basic probability theory, followed by the most common statistical concepts, and different methods of statistical inference. Students then apply these concepts to real-world examples. The course concentrates on application and understanding, rather than mathematical derivation, to give students a solid grounding in the world of statistics without requiring an extensive mathematical background.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to Applied Statistics
2. Apply a professional and scholarly approach to research problems pertaining to Applied Statistics
3. Efficiently manage interdisciplinary issues that arise in connection to applied statistics
4. Demonstrate self-direction in research and originality in solutions developed in the field of applied statistics
5. Act autonomously in identifying research problems and solutions related to statistical methods
6. Solve problems and be prepared to take leadership decisions related to applied statistics and probability theory

6. Introduction to Machine Learning

This course focuses on building basic classical machine learning models. Students will learn theory and implementation of a variety of classical machine learning algorithms, including linear and logistic regression, decision trees, random forests, k-means, knn, SVM, and naive bayes. The course focuses both on the mathematical intuition behind the algorithms, and the practical implementation using sklearn. The course ends with a portfolio-building capstone project.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to machine learning algorithms, data processing, and model evaluation
2. Apply a professional and scholarly approach to research problems pertaining to machine learning
3. Efficiently manage interdisciplinary issues that arise in connection to the implementation of machine learning algorithms
4. Demonstrate self-direction in research and originality in solutions developed in the field of machine learning
5. Act autonomously in identifying research problems and solutions related to building and evaluating machine learning models
6. Solve problems and be prepared to take leadership decisions related to the design and implementation of machine learning solutions

7. High dimensional Data Analysis

This course is aimed to help learners understand issues related to high dimensional data including the curse of dimensionality. Students learn dimensionality reduction methods (PCA, t-SNE, UMAP, Autoencoders) as well as methods to deal with high dimensional datasets, including regularisation and ensemble methods. The module also covers advanced topics including kernel methods and Gaussian processes.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to high dimensional data analysis and the curse of dimensionality
2. Apply a professional and scholarly approach to research problems pertaining to dimensionality reduction methods
3. Efficiently manage interdisciplinary issues that arise in connection to implementing regularisation and ensemble methods
4. Demonstrate self-direction in research and originality in solutions developed in the field of high dimensional data analysis

5. Act autonomously in identifying research problems and solutions related to advanced topics including kernel methods and Gaussian processes
6. Solve problems and be prepared to take leadership decisions related to selection and application of dimensionality reduction and high-dimensional modelling approaches

8. Advanced Machine Learning

This course introduces more advanced ML algorithms, beginning with Bayesian approaches to machine learning, and continuing with Gaussian Processes, Bayesian Optimization and Monte-Carlo methods. Students will learn Hidden Markov Models and Conditional Random Fields as examples of (time-series) structured prediction models. The course ends with an overview of ML fairness and model interpretability methods.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to advanced machine learning algorithms
2. Apply a professional and scholarly approach to research problems pertaining to Bayesian approaches to machine learning
3. Efficiently manage interdisciplinary issues that arise in connection to the implementation of Gaussian Processes and Monte-Carlo methods
4. Demonstrate self-direction in research and originality in solutions developed in the area of structured prediction and time-series modelling
5. Act autonomously in identifying research problems and solutions related to ML fairness and model interpretability
6. Solve problems and be prepared to take leadership decisions related to the application of advanced machine learning approaches

9. Distributed Machine Learning

This course provides an in-depth understanding of the challenges and solutions for running machine learning workloads at scale in distributed environments. Students will learn how to leverage cloud computing and distributed systems to train large-scale models efficiently. Topics include data parallelism, model parallelism, gradient compression, and federated learning. By the end of the module, students will have practical experience with tools and frameworks used in industry for distributed ML, such as Spark and Horovod.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to distributed machine learning and cloud computing
2. Apply a professional and scholarly approach to research problems pertaining to training large-scale models efficiently
3. Efficiently manage interdisciplinary issues that arise in connection to the implementation of data parallelism and model parallelism
4. Demonstrate self-direction in research and originality in solutions developed in the field of federated learning
5. Act autonomously in identifying research problems and solutions related to gradient compression and distributed ML tools

6. Solve problems and be prepared to take leadership decisions related to developing scalable machine learning pipelines

10. Introduction to Deep Learning

This course provides a strong mathematical and practical background in deep learning. Beginning with the history and motivation behind neural networks, students will learn the mathematical details of backpropagation, and build their first neural networks from scratch in numpy. Students will then learn to use Tensorflow/Keras to build CNNs, RNNs, and other modern architectures. The course ends with a capstone project where students will apply their skills in a real-world setting.

Learning Outcomes

1. Create synthetic contextualized discussions of key issues related to the history, motivation and mathematical details behind neural networks
2. Apply a professional and scholarly approach to research problems pertaining to backpropagation and neural network architecture
3. Efficiently manage interdisciplinary issues that arise in connection to building CNNs, RNNs and other architectures using Tensorflow/Keras
4. Demonstrate self-direction in research and originality in solutions developed in the field of deep learning
5. Act autonomously in identifying research problems and solutions related to deep learning model design and training
6. Solve problems and be prepared to take leadership decisions related to applying deep learning to real-world problems

11. Deep Learning for Computer Vision

This course provides a comprehensive overview of the key methods and models of computer vision. Topics covered include traditional CV methods, CNNs, Vision Transformers, Object Detection and Segmentation, Generative models (VAEs, GANs, Diffusion models), and multi-modal models (CLIP, Stable Diffusion). The course includes a final project where students will train their own model from scratch on a real-world dataset.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to traditional computer vision methods and modern CNN architectures
2. Apply a professional and scholarly approach to research problems pertaining to object detection, segmentation and generative models
3. Efficiently manage interdisciplinary issues that arise in connection to training and evaluating models using real-world datasets
4. Demonstrate self-direction in research and originality in solutions developed in the field of multi-modal computer vision models
5. Act autonomously in identifying research problems and solutions related to applying deep learning to computer vision tasks
6. Solve problems and be prepared to take leadership decisions related to the selection and adaptation of computer vision architectures for specific applications

12. Deep Learning for Natural Language Processing (NLP)

This course focuses on modelling sequences and language with deep learning. Students will progress from early sequence models (RNNs, LSTMs), through the revolutionary Transformer architecture, and all the way to modern Large Language Models (LLMs). Topics include fine-tuning, RAG, and LLM agents. The course ends with a final capstone project where students train a language model from scratch.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to modelling sequences using RNNs and LSTMs
2. Apply a professional and scholarly approach to research problems pertaining to the Transformer architecture and its applications
3. Efficiently manage interdisciplinary issues that arise in connection to fine-tuning LLMs and implementing RAG
4. Demonstrate self-direction in research and originality in solutions developed in the field of LLM agents
5. Act autonomously in identifying research problems and solutions related to NLP model design and evaluation
6. Solve problems and be prepared to take leadership decisions related to applying large language models to real-world NLP tasks

13. Productionization of Machine Learning Systems

This course aims to build the core competencies required to serve machine learning models in production. Students will learn the full MLOps workflow: data versioning, experiment tracking, model packaging, CI/CD pipelines for ML, model serving, and monitoring. The course uses industry-standard tools such as MLflow, DVC, Docker, and Kubernetes. Students will end up with a complete production ML project.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to MLOps and the full machine learning production workflow
2. Apply a professional and scholarly approach to research problems pertaining to data versioning, experiment tracking and model packaging
3. Efficiently manage interdisciplinary issues that arise in connection to CI/CD pipelines for ML and model serving
4. Demonstrate self-direction in research and originality in solutions developed using industry tools such as MLflow, DVC and Kubernetes
5. Act autonomously in identifying research problems and solutions related to model monitoring in production
6. Solve problems and be prepared to take leadership decisions related to building and maintaining production ML systems

14. System Design

This course is aimed at equipping students with a thorough understanding of the approaches taken in building large-scale systems that are reliable, maintainable and scalable. It covers key topics of system design such as horizontal/vertical scaling, load balancing, caching, data storage, CDN, message queues,

and microservices. Students will learn how to design systems for real-world use cases, including systems like Twitter, Uber, and YouTube.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to scalable and reliable large-scale system design
2. Apply a professional and scholarly approach to research problems pertaining to horizontal/vertical scaling and load balancing
3. Efficiently manage interdisciplinary issues that arise in connection to caching, CDN and message queue design
4. Demonstrate self-direction in research and originality in solutions developed in the field of microservices architecture
5. Act autonomously in identifying research problems and solutions related to data storage and retrieval at scale
6. Solve problems and be prepared to take leadership decisions related to designing systems for real-world large-scale use cases

15. DevOps

This course provides students with hands-on experience in DevOps tools and practices. Students will learn about version control using Git and GitHub, containerization using Docker, orchestration using Kubernetes, CI/CD pipelines using GitHub Actions, infrastructure as code using Terraform, and monitoring and observability. The course is project-based and prepares students for real-world DevOps roles.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to DevOps practices and the software development lifecycle
2. Apply a professional and scholarly approach to research problems pertaining to containerisation using Docker and orchestration using Kubernetes
3. Efficiently manage interdisciplinary issues that arise in connection to CI/CD pipeline design and infrastructure as code
4. Demonstrate self-direction in research and originality in solutions developed using industry tools such as GitHub Actions and Terraform
5. Act autonomously in identifying research problems and solutions related to monitoring and observability in production systems
6. Solve problems and be prepared to take leadership decisions related to implementing and managing DevOps workflows

16. Front end UI/UX development

This is a hands-on course on designing responsive web interfaces using HTML, CSS, Bootstrap, and Figma. Students will learn about user-centred design principles, wireframing, prototyping, and usability testing. The course includes projects that take students from design concept through to implementation.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to user-centred design and responsive web interface development
2. Apply a professional and scholarly approach to research problems pertaining to wireframing, prototyping and usability testing
3. Efficiently manage interdisciplinary issues that arise in connection to using HTML, CSS, Bootstrap and Figma
4. Demonstrate self-direction in research and originality in solutions developed in the field of UI/UX design
5. Act autonomously in identifying research problems and solutions related to front-end development best practices
6. Solve problems and be prepared to take leadership decisions related to the design and implementation of responsive web interfaces

17. JavaScript

This course is a hands-on course covering all key concepts in JavaScript, including functions, closures, prototypal inheritance, asynchronous programming, and the event loop. Students will learn modern ES6+ syntax, and get an introduction to front-end frameworks. The course is project-based and prepares students for further study in front-end and back-end development using JavaScript.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to JavaScript fundamentals and modern ES6+ syntax
2. Apply a professional and scholarly approach to research problems pertaining to closures, prototypal inheritance and asynchronous programming
3. Efficiently manage interdisciplinary issues that arise in connection to the event loop and front-end framework integration
4. Demonstrate self-direction in research and originality in solutions developed using modern JavaScript
5. Act autonomously in identifying research problems and solutions related to JavaScript design patterns
6. Solve problems and be prepared to take leadership decisions related to building production-ready JavaScript applications

18. Front End Development

This course builds upon the introductory JavaScript course to explore modern front-end development using React. Students will learn component-based architecture, state management with Redux, server-side rendering, and performance optimisation. The course includes projects that replicate real-world front-end engineering workflows.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to component-based architecture and React development
2. Apply a professional and scholarly approach to research problems pertaining to state management using Redux
3. Efficiently manage interdisciplinary issues that arise in connection to server-side rendering and performance optimisation

4. Demonstrate self-direction in research and originality in solutions developed in the field of modern front-end engineering
5. Act autonomously in identifying research problems and solutions related to front-end architecture design
6. Solve problems and be prepared to take leadership decisions related to building scalable React applications

19. Back End Development

This is a foundational course on building RESTful APIs and server-side applications. Students will learn to work with Node.js and Express, design RESTful APIs, use ORMs to interact with databases, and implement authentication using JWT and OAuth. The course includes a final project in which students build a full-stack web application.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to building RESTful APIs and server-side applications
2. Apply a professional and scholarly approach to research problems pertaining to Node.js, Express and ORM usage
3. Efficiently manage interdisciplinary issues that arise in connection to authentication using JWT and OAuth
4. Demonstrate self-direction in research and originality in solutions developed in the field of full-stack web development
5. Act autonomously in identifying research problems and solutions related to back-end architecture design
6. Solve problems and be prepared to take leadership decisions related to building secure and scalable back-end systems

20. Foundations of Cloud Computing

This is a course that focuses both on architecture and hands-on skills in cloud computing. Students will learn about the key concepts and services of cloud computing, and gain practical experience with cloud platforms such as AWS or GCP. Topics include compute, storage, networking, serverless, databases, identity and access management, and security. Students will work towards a cloud certification.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to cloud computing architecture and services
2. Apply a professional and scholarly approach to research problems pertaining to compute, storage and networking in the cloud
3. Efficiently manage interdisciplinary issues that arise in connection to serverless computing and cloud database management
4. Demonstrate self-direction in research and originality in solutions developed in the field of cloud security and IAM

5. Act autonomously in identifying research problems and solutions related to cloud platform selection and architecture
6. Solve problems and be prepared to take leadership decisions related to deploying and managing production cloud infrastructure

21. Advanced Backend Development

This course provides a dive deep into more advanced back-end topics, including microservices architecture, event-driven systems using message queues, GraphQL, gRPC, and advanced database optimisation. Students will work on complex real-world projects that mirror industry practices in modern back-end engineering.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to microservices architecture and event-driven systems
2. Apply a professional and scholarly approach to research problems pertaining to GraphQL, gRPC and advanced database optimisation
3. Efficiently manage interdisciplinary issues that arise in connection to message queue design and implementation
4. Demonstrate self-direction in research and originality in solutions developed in the field of advanced back-end engineering
5. Act autonomously in identifying research problems and solutions related to distributed system design patterns
6. Solve problems and be prepared to take leadership decisions related to building and scaling complex back-end architectures

22. Distributed Cloud Computing

This course provides an in-depth architecture of distributed systems as they apply to cloud computing. Students will learn about consistency models, distributed transactions, consensus algorithms, and fault tolerance. The course covers tools and technologies such as Kafka, Zookeeper, and distributed databases. Students will design and implement distributed systems that meet real-world reliability and scalability requirements.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to distributed systems architecture in cloud computing
2. Apply a professional and scholarly approach to research problems pertaining to consistency models and distributed transactions
3. Efficiently manage interdisciplinary issues that arise in connection to consensus algorithms and fault tolerance
4. Demonstrate self-direction in research and originality in solutions developed using tools such as Kafka and Zookeeper
5. Act autonomously in identifying research problems and solutions related to distributed database design
6. Solve problems and be prepared to take leadership decisions related to building reliable and scalable distributed cloud systems

23. Advanced Cloud Computing

This is a course that focuses both on architecture and hands-on skills in cloud computing. Students will build on the foundations of cloud computing to explore advanced topics including advanced networking, multi-cloud and hybrid cloud strategies, cloud-native application development, and cost optimisation. Students will work on industry-scale cloud architecture projects.

Learning Outcomes

1. deploy advanced cloud computing systems and manage their costs, performance and security at scale
2. Create synthetic contextualised discussions of key issues related to advanced cloud networking and multi-cloud strategies
3. Apply a professional and scholarly approach to research problems pertaining to cloud-native application development
4. Efficiently manage interdisciplinary issues that arise in connection to hybrid cloud deployment and optimisation
5. Demonstrate self-direction in research and originality in solutions developed for advanced cloud architecture
6. Act autonomously in identifying research problems and solutions related to cloud cost optimisation

24. NoSQL Cloud Datastores

This course provides a comprehensive overview of NoSQL databases and their use in cloud environments. Students will learn about different NoSQL data models (document, key-value, wide-column, graph) and when to apply them. Practical work covers MongoDB, DynamoDB, and BigQuery. Students will also learn how to design data architectures that combine relational and NoSQL stores.

Learning Outcomes

1. create synthetic contextualised discussions of key issues related to NoSQL database models and cloud datastore design
2. Apply a professional and scholarly approach to research problems pertaining to document, key-value, wide-column and graph databases
3. Efficiently manage interdisciplinary issues that arise in connection to the use of MongoDB, DynamoDB and BigQuery
4. Demonstrate self-direction in research and originality in solutions developed for hybrid relational and NoSQL data architectures
5. Act autonomously in identifying research problems and solutions related to selecting appropriate data models for cloud applications
6. Solve problems and be prepared to take leadership decisions related to designing scalable cloud data architectures

25. Design Patterns

This course provides a practical understanding of classical software design patterns (Gang of Four) and modern architectural patterns. Students will learn to identify problems in software design and apply appropriate patterns as solutions. The course covers creational, structural, and behavioural patterns, as well as architectural patterns such as MVC, MVP, and MVVM.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to software design patterns and their application
2. Apply a professional and scholarly approach to research problems pertaining to creational, structural and behavioural design patterns
3. Efficiently manage interdisciplinary issues that arise in connection to the application of MVC, MVP and MVVM architectural patterns
4. Demonstrate self-direction in research and originality in solutions developed using classical Gang of Four design patterns
5. Act autonomously in identifying research problems and solutions related to software architecture design
6. Solve problems and be prepared to take leadership decisions related to selecting and applying appropriate design patterns in software engineering projects

26. Capstone: Advanced Applied Computer Science

Advanced Applied Computer Science is a capstone module that allows students to build a substantial real-world project applying concepts from across their programme. Students will work individually or in small teams to design, implement and evaluate a significant software or data project, with regular supervision and milestones. The capstone culminates in a project presentation and written report.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to the design, implementation and evaluation of large-scale software or data projects
2. Apply a professional and scholarly approach to research problems pertaining to project management and software development lifecycle
3. Efficiently manage interdisciplinary issues that arise in connection to the integration of multiple computer science domains in a single project
4. Demonstrate self-direction in research and originality in the design and execution of an advanced applied computer science project
5. Act autonomously in identifying research problems and solutions related to large-scale software engineering
6. Solve problems and be prepared to take leadership decisions related to technical direction and project delivery

27. Introduction to Computer Programming: Part 1

This course helps students translate advanced mathematical and logical concepts into computational solutions using Python. Students will learn the fundamentals of Python programming: data types, control structures, functions, modules, and basic data structures. By the end of the course, students will be able to write Python programs to solve a variety of problems.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to foundational Python programming concepts

2. Apply a professional and scholarly approach to research problems pertaining to data types, control structures and functions in Python
3. Efficiently manage interdisciplinary issues that arise in connection to implementing Python solutions for mathematical and logical problems
4. Demonstrate self-direction in research and originality in solutions developed using Python programming
5. Act autonomously in identifying research problems and solutions related to Python-based computational problem solving
6. Solve problems and be prepared to take leadership decisions related to the design and implementation of Python programs

28. Introduction to Problem-Solving Techniques: Part 1

The ability to solve problems is a skill that spans many areas of life, but this course focuses specifically on the type of problems that arise in computer science, mathematics and business. This first part covers foundational problem-solving strategies, logic puzzles, and pattern recognition, building the analytical skills necessary for study and work in technical fields.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to problem-solving strategies in computer science and mathematics
2. Apply a professional and scholarly approach to research problems pertaining to logic puzzles and pattern recognition
3. Efficiently manage interdisciplinary issues that arise in connection to applying problem-solving frameworks across technical domains
4. Demonstrate self-direction in research and originality in solutions developed using analytical and critical thinking skills
5. Act autonomously in identifying research problems and solutions related to structured problem decomposition
6. Solve problems and be prepared to take leadership decisions related to selecting appropriate problem-solving strategies

29. Introduction to Problem-Solving Techniques: Part 2

This course is a follow-up to Introduction to Problem-Solving Techniques: Part 1 and continues to develop the student's ability to tackle complex problems. This second part introduces more advanced techniques including algorithmic problem-solving, combinatorial reasoning, and introduction to mathematical proofs, preparing students for advanced study in computer science.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to advanced problem-solving strategies
2. Apply a professional and scholarly approach to research problems pertaining to algorithmic problem-solving and combinatorial reasoning
3. Efficiently manage interdisciplinary issues that arise in connection to mathematical proofs and formal reasoning

4. Demonstrate self-direction in research and originality in solutions developed using advanced analytical techniques
5. Act autonomously in identifying research problems and solutions related to complex computational challenges
6. Solve problems and be prepared to take leadership decisions related to applying advanced problem-solving strategies in computer science

30. Mathematics for Computer Science

Mathematics and computer science are closely linked, and many areas of computer science rely heavily on mathematical tools and techniques. This course covers the essential mathematical topics for computer science: discrete mathematics, number theory, linear algebra, and probability theory. Students will develop both the conceptual understanding and the practical ability to apply these tools in computing contexts.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to discrete mathematics and number theory in computer science
2. Apply a professional and scholarly approach to research problems pertaining to linear algebra and probability theory
3. Efficiently manage interdisciplinary issues that arise in connection to the application of mathematical tools in computing
4. Demonstrate self-direction in research and originality in solutions developed using mathematical techniques
5. Act autonomously in identifying research problems and solutions related to mathematical modelling in computer science
6. Solve problems and be prepared to take leadership decisions related to the application of mathematics to complex computing problems

31. Advanced Algorithms

This module covers general approaches to algorithm design with a specific focus on solving real world problems. It covers graph algorithms, dynamic programming and greedy algorithms in depth, as well as approximation algorithms and randomised algorithms. Students will learn how to analyse algorithm correctness and complexity and how to apply these algorithms to solve computational problems efficiently.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to advanced algorithm design strategies
2. Apply a professional and scholarly approach to research problems pertaining to graph algorithms and dynamic programming
3. Efficiently manage interdisciplinary issues that arise in connection to the analysis of algorithm correctness and complexity
4. Demonstrate self-direction in research and originality in solutions developed using approximation and randomised algorithms
5. Act autonomously in identifying research problems and solutions related to efficient algorithm design for computational problems

6. Solve problems and be prepared to take leadership decisions related to selecting and implementing advanced algorithms in real-world contexts

32. Computer Systems and Their Fundamentals

This core course equips the student with knowledge of database management systems and computer architecture, providing a comprehensive understanding of how modern computing systems are structured and function. Topics include memory hierarchies, processor design, operating systems fundamentals, and database architecture.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to the optimisation and efficiency of modern computing systems
2. Apply a professional and scholarly approach to research problems pertaining to computer architecture and database management
3. Efficiently manage interdisciplinary issues that arise in connection to memory hierarchies and operating systems fundamentals
4. Demonstrate self-direction in research and originality in solutions developed in the field of computer systems design
5. Act autonomously in identifying research problems and solutions related to processor design and system architecture
6. Solve problems and be prepared to take leadership decisions related to the design and optimisation of computing systems

33. Low-Level Design and Design Patterns

Low-Level Design & Design Patterns focuses on modularity and reusability in software design. Students will learn how to apply object-oriented principles, SOLID design principles, and classical design patterns to build clean, maintainable, and scalable software systems. The course uses real-world case studies and hands-on projects.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to Object-Oriented Design and classical design patterns
2. Apply a professional and scholarly approach to research problems pertaining to SOLID design principles
3. Efficiently manage interdisciplinary issues that arise in connection to the implementation of creational, structural and behavioural design patterns
4. Demonstrate self-direction in research and originality in solutions developed for building clean and maintainable software systems
5. Act autonomously in identifying research problems and solutions related to modularity and reusability in software design
6. Solve problems and be prepared to take leadership decisions related to applying design patterns in real-world software engineering projects

34. Practical Software Engineering

This course gives the detailed overview on important aspects of practical software engineering and provides hands-on experience in tools and techniques used for building, testing and deploying software systems. Topics include version control, agile methodologies, test-driven development, code review practices, and continuous integration.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to practical software engineering workflows
2. Apply a professional and scholarly approach to research problems pertaining to agile methodologies and test-driven development
3. Efficiently manage interdisciplinary issues that arise in connection to version control, code review and continuous integration practices
4. Demonstrate self-direction in research and originality in solutions developed using industry-standard software engineering tools
5. Act autonomously in identifying research problems and solutions related to software quality and testing
6. Solve problems and be prepared to take leadership decisions related to the design and management of software development workflows

35. Distributed Systems with High-Level System Design

A distributed system is an application that executes a collection of protocols to coordinate the actions of multiple processes on a network, such that all components cooperate together to perform a single or small set of related tasks. This course covers the foundations of distributed systems, and then applies them to high-level system design problems, teaching students to design systems that are fault tolerant, consistent and highly available.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to distributed system architecture and coordination protocols
2. Apply a professional and scholarly approach to research problems pertaining to fault tolerance, consistency and high availability
3. Efficiently manage interdisciplinary issues that arise in connection to the design of multi-process networked applications
4. Demonstrate self-direction in research and originality in solutions developed in the field of distributed systems design
5. Act autonomously in identifying research problems and solutions related to high-level system design challenges
6. Solve problems and be prepared to take leadership decisions related to designing scalable, fault-tolerant distributed systems

36. Data Visualisation Tools

This course is aimed to build a strong foundational knowledge of popular data visualisation tools and techniques used in the industry. Students will learn how to create compelling and informative visualisations using tools such as Matplotlib, Seaborn, Plotly, and Tableau. The course emphasises the principles of effective data communication and storytelling with data.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to data visualisation principles and tools
2. Apply a professional and scholarly approach to research problems pertaining to the use of Matplotlib, Seaborn, Plotly and Tableau
3. Efficiently manage interdisciplinary issues that arise in connection to effective data communication and storytelling
4. Demonstrate self-direction in research and originality in solutions developed for domain-specific data visualisation challenges
5. Act autonomously in identifying research problems and solutions related to visual representation of complex datasets
6. Solve problems and be prepared to take leadership decisions related to selecting and applying data visualisation tools for specific analytical contexts

37. Power BI for Data Analysis and Exploration

Power BI is a Microsoft tool that works to turn data from many sources into interactive dashboards and business intelligence reports. Students will learn how to connect to data sources, create data models, write DAX queries, and design interactive dashboards. The course includes real-world business case studies.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to business intelligence reporting and dashboard design
2. Apply a professional and scholarly approach to research problems pertaining to Power BI data modelling and DAX queries
3. Efficiently manage interdisciplinary issues that arise in connection to connecting multiple data sources and designing interactive reports
4. Demonstrate self-direction in research and originality in solutions developed for business intelligence use cases
5. Act autonomously in identifying research problems and solutions related to data-driven decision making
6. Solve problems and be prepared to take leadership decisions related to the design and deployment of Power BI dashboards

38. Statistical Programming

This module focuses on representing statistical concepts through programming. Students will develop skills in using statistical programming languages and libraries (primarily R and Python) to perform data analysis, statistical modelling, and visualisation. The course covers regression analysis, hypothesis testing, Bayesian statistics, and Monte Carlo simulation.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to statistical programming and data analysis
2. Apply a professional and scholarly approach to research problems pertaining to regression analysis and hypothesis testing

3. Efficiently manage interdisciplinary issues that arise in connection to Bayesian statistics and Monte Carlo simulation
4. Demonstrate self-direction in research and originality in solutions developed using R and Python for statistical modelling
5. Act autonomously in identifying research problems and solutions related to advanced statistical methods
6. Solve problems and be prepared to take leadership decisions related to the application of statistical programming in data-driven contexts

39. Spreadsheets for Data Understanding

Spreadsheets for Data Understanding introduces students to the use of spreadsheet tools (primarily Excel and Google Sheets) for data analysis and business intelligence. Topics include data cleaning, pivot tables, VLOOKUP, conditional formatting, financial modelling, and introduction to VBA macros. The course emphasises practical data skills for business contexts.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to spreadsheet-based data analysis and business intelligence
2. Apply a professional and scholarly approach to research problems pertaining to data cleaning, pivot tables and financial modelling
3. Efficiently manage interdisciplinary issues that arise in connection to the use of Excel and Google Sheets for professional data tasks
4. Demonstrate self-direction in research and originality in solutions developed using spreadsheet tools for business contexts
5. Act autonomously in identifying research problems and solutions related to data organisation and visualisation in spreadsheets
6. Solve problems and be prepared to take leadership decisions related to the design of spreadsheet-based analytical frameworks

40. Advanced Python Programming

Advanced Python Programming builds on introductory Python skills to cover more advanced concepts including object-oriented programming, functional programming, concurrency, and performance optimisation. Students will also learn about Python's scientific and data ecosystem in depth, including NumPy, Pandas, and Scikit-learn. The course includes a substantial final project.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to advanced Python programming paradigms
2. Apply a professional and scholarly approach to research problems pertaining to OOP, functional programming and concurrency in Python
3. Efficiently manage interdisciplinary issues that arise in connection to performance optimisation and the Python scientific ecosystem
4. Demonstrate self-direction in research and originality in solutions developed using NumPy, Pandas and Scikit-learn

5. Act autonomously in identifying research problems and solutions related to advanced Python software design
6. Solve problems and be prepared to take leadership decisions related to building high-performance Python applications

41. Foundations of Machine Learning

This course focuses on building basic classical machine learning models. Students will learn theory and implementation of a variety of classical machine learning algorithms, including linear and logistic regression, decision trees, random forests, k-means, knn, SVM, and naive bayes. The course focuses both on the mathematical intuition behind the algorithms, and the practical implementation using sklearn. The course ends with a portfolio-building capstone project.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to classical machine learning algorithms and their mathematical foundations
2. Apply a professional and scholarly approach to research problems pertaining to model training, evaluation and selection
3. Efficiently manage interdisciplinary issues that arise in connection to the implementation of classification, regression and clustering algorithms
4. Demonstrate self-direction in research and originality in solutions developed in the field of machine learning
5. Act autonomously in identifying research problems and solutions related to machine learning model selection and optimisation
6. Solve problems and be prepared to take leadership decisions related to applying machine learning to real-world problems

42. Business Case Studies

A business case study is a course designed to put practical problem-solving into the real world context of a business. Students will analyse real-world business cases, applying quantitative and qualitative methods to evaluate business decisions, identify problems, and propose data-driven solutions. The course develops critical thinking and communication skills alongside analytical techniques.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to real-world business problem analysis
2. Apply a professional and scholarly approach to research problems pertaining to quantitative and qualitative business analysis methods
3. Efficiently manage interdisciplinary issues that arise in connection to evaluating business decisions and proposing data-driven solutions
4. Demonstrate self-direction in research and originality in solutions developed for business case analysis
5. Act autonomously in identifying research problems and solutions related to business intelligence and strategy
6. Solve problems and be prepared to take leadership decisions related to applying analytical techniques to real-world business contexts

43. Studies in Data Science and Data Analytics

This advanced graduate class addresses a wide range of topics in applied data science. Students will explore advanced machine learning methods, time series analysis, network analysis, and natural language processing from a data science perspective. The course emphasises the application of these methods to real-world datasets across multiple domains.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to advanced applied data science methods
2. Apply a professional and scholarly approach to research problems pertaining to time series analysis and network analysis
3. Efficiently manage interdisciplinary issues that arise in connection to natural language processing in data science contexts
4. Demonstrate self-direction in research and originality in applying machine learning methods to domain-specific datasets
5. Act autonomously in identifying research problems and solutions related to massive data sets
6. Solve problems and be prepared to take leadership decisions related to developing a data-informed approach to a domain-specific problem

44. Further Studies in Data Science and Data Analytics

This advanced graduate class addresses a wide range of topics in applied data science. Building on Studies in Data Science, this course covers further advanced topics including causal inference, reinforcement learning, and advanced deep learning for data science, with an emphasis on cutting-edge methods and real-world application.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to causal inference and reinforcement learning in data science
2. Apply a professional and scholarly approach to research problems pertaining to advanced deep learning methods for data science
3. Efficiently manage interdisciplinary issues that arise in connection to applying cutting-edge data science methods to real-world problems
4. Demonstrate self-direction in research and originality in applying advanced data science techniques to complex datasets
5. Act autonomously in identifying research problems and solutions related to massive data sets
6. Solve problems and be prepared to take leadership decisions related to developing a data-informed approach to a domain-specific problem

45. SQL for Data Analytics

Structured Query Language (SQL) is key to working with data in relational databases, a task at the core of data science and analytics. In this course, students will learn all the major keywords and clauses used to extract data, best practices for formatting SQL queries, and how to generate meaningful insights from the results.

Internships policy

Internships must be a genuine extension of the student's academic programme, providing opportunity to apply theoretical knowledge to substantive projects directly related to their field of study. Internships consisting primarily of administrative or routine tasks will not be approved.

Every internship must have a defined start date, end date, and formal learning plan with objectives agreed in advance by the student, the host organisation, and the relevant college. Responsibilities and task complexity should increase over time. Each student must be assigned a named supervisor within the host organisation who holds relevant expertise and is responsible for providing regular guidance and feedback.

Woolf prioritises paid internships to ensure equitable access regardless of socioeconomic background. Unpaid internships may only be approved where they constitute a genuine learning opportunity and do not displace the work of a paid employee.

Programmatic standards

Day-to-day management sits with the relevant college. Each college must have a designated Woolf contact responsible for vetting and approving all host organisations and placements before any internship may proceed. Colleges are responsible for matching students to approved positions.

Students must complete pre-internship preparation before commencing a placement, which may include CV writing, interview support, and other instruction as necessary. Virtual internships are encouraged to widen access beyond geographical constraints; support systems must address the challenges of remote work, including cross-timezone communication and fostering professional belonging.

Programme effectiveness must be evaluated on an ongoing basis. Formal evaluations will be collected from students, host supervisors, and academic advisors, and will inform curriculum design and programme improvement.

General Marking Criteria and Classification

Marking of student work keeps in view the scale of work that the student can reasonably be expected to have undertaken in order to complete the task.

The assessment of work for the course is defined according to the following rubric of general criteria:

1. **Engagement:**
    - Directness of engagement with the question or task
    - Range of issues addressed or problems solved
    - Depth, complexity, and sophistication of comprehension of issues and implications of the questions or task
    - Effective and appropriate use of imagination and intellectual curiosity
2. **Argument or solution:**
    - Coherence, mastery, control, and independence of work
    - Conceptual and analytical precision
    - Flexibility, i.e., discussion of a variety of views, ability to navigate through challenges in creative ways
    - Completion leading to a conclusion or outcome
    - Performance and success of the solution, where relevant
3. **Evidence (as relevant):**
    - Depth, precision, detail, range and relevance of evidence cited

    - Accuracy of facts
    - Knowledge of first principles and demonstrated ability to reason from them
    - Understanding of theoretical principles and/or historical debate
    - Critical engagement with primary and/or secondary sources
4. **Organisation & Presentation:**
    - Clarity and coherence of structure
    - Clarity and fluency of writing, code, prose, or presentation (as relevant)
    - Correctness of conformity to conventions (code, grammar, spelling, punctuation, or similar relevant conventions)

Definition of marks

97-100

Work will be so outstanding that it could not be better within the scope of the assignment. These grades will be used for work that shows exceptional excellence in the relevant domain; including (as relevant): remarkable sophistication and mastery, originality or creativity, persuasive and well-grounded new methods or ideas, or making unexpected connections or solutions to problems.

94-96

Work will excel against each of the General Criteria. In at least one area, the work will be merely highly competent.

90-93

Work will excel in more than one area, and be at least highly competent in other respects. It must be excellent and contain: a combination of sophisticated engagement with the issues; analytical precision and independence of solution; go beyond paraphrasing or boilerplate code techniques; demonstrating quality of awareness and analysis of both first principles or primary evidence and scholarly debate or practical tradeoffs; and clarity and coherence of presentation. Truly outstanding work measured against some of these criteria may compensate for mere high competence against others.

87-89

Work will be at least very highly competent across the board, and excel in at least one group of the General Criteria. Relative weaknesses in some areas may be compensated by conspicuous strengths in others.

84-86

Work will demonstrate considerable competence across the General Criteria. They must exhibit some essential features of addressing the issue directly and relevantly across a good range of aspects; offer a coherent solution or argument involving (where relevant) consideration of alternative approaches; be substantiated with accurate use of resources (including if relevant, primary evidence) and contextualisation in debate (if relevant); and be clearly presented. Nevertheless, additional strengths (for instance, the range of problems addressed, the sophistication of the arguments or solutions, or the use of first principles) may compensate for other weaknesses.

80-83

Work will be competent and should manifest the essential features described above, in that they must offer direct, coherent, substantiated and clear arguments; but they will do so with less range, depth, precision and perhaps clarity. Again, qualities of a higher order may compensate for some weaknesses.

77-79

Work will show solid competence in solving problems or providing analysis. But it will be marred by weakness under one or more criteria: failure to fully solve the problem or discuss the question directly; some irrelevant use of technologies or citing of information; factual error, or error in selection of technologies; narrowness in the scope of solution or range of issues addressed or evidence adduced; shortage of detailed evidence or engagement with the problem; technical performance issues (but not so much as to prevent operation); poor organisation or presentation, including incorrect conformity to convention or written formatting.

74-76

Work will show evidence of some competence in solving problems or providing analysis. It will also be clearly marred by weakness in multiple General Criteria, including: failure to solve the problem or discuss the question directly; irrelevant use of technologies or citing of information; factual errors or multiple errors in selection of technologies; narrowness in the scope of solution or range of issues addressed or evidence adduced; shortage of detailed evidence or engagement with the problem; significant technical performance issues (but not so much as to prevent operation); poor organisation or presentation, including incorrect conformity to convention or written formatting. They may be characterised by unsubstantiated assertion rather than argument, or by unresolved contradictions in the argument or solution.

70-73

Work will show evidence of competence in solving problems or providing analysis, but this evidence will be limited. It will be clearly marred by weakness in multiple General Criteria. It will still make substantive progress in addressing the primary task or question, but the work will lack a full solution or directly address the task; the work will contain irrelevant material; the work will show multiple errors of fact or judgment; and the work may fail to conform to conventions.

67-69

Work will fall down on a number of criteria, but will exhibit some of the qualities required, such as the ability to grasp the purpose of the assignment, to deploy substantive information or solutions in an effort to complete the assignment; or to offer some coherent analysis or work towards the assignment. Such qualities will not be displayed at a high level, and may be marred by irrelevance, incoherence, major technical performance issues, error and poor organisation and presentation.

64-66

Work will fall down on a multiple General Criteria, but will exhibit some vestiges of the qualities required, such as the ability to see the point of the question, to deploy information, or to offer some coherent work. Such qualities will be substantially marred by irrelevance, incoherence, error and poor organisation and presentation.

60-63

Work will display a modicum of knowledge or understanding of some points, but will display almost none of the higher qualities described in the criteria. They will be marred by high levels of factual or technology error and irrelevance, generalisation or boilerplate code and lack of information, and poor organisation and presentation.

0-60

Work will fail to exhibit any of the required qualities. Candidates who fail to observe rubrics and rules beyond what the grading schemes allow for may also be failed.

Indicative equivalence table

US GPA	US Grade	US Percent	UK Mark	UK UG Classification	UK PG Classification	Malta Grade	Malta Mark	Malta Classification	Swiss Grade
4	A+	97 - 100	70+	First	Distinction	A	80-100%	First class honours	6.0
3.9	A	94-96				B	70-79%	Upper-second class honours	
3.7	A-	90-93							5.5
3.3	B+	87-89	65-69	Upper Second	Merit	C	55-69%	Lower-second class honours	
3	B	84-86	60-64						
2.7	B-	80-83	55-59	Lower Second	Pass				5
2.3	C+	77-79	50-54			D	50-54%	Third-class honours	
2	C	74–76	45-49	Third	Pass				
1.7	C-	70–73	40-44						
1.3	D+	67–69	39-	Fail	Fail				
1	D	64–66							
0.7	D-	60–63							
0	F	Below 60				F			

Synchronous Adjustments Template

Synch discussions may affect the mark on submitted assignments: written work is submitted in advance, and a discussion follows. This provides students an opportunity to clarify and explain their written claims, and it also tests whether the work is a product of the student's own research or has been plagiarised.

The synchronous discussion acts to shift the recorded mark on the submitted assignment according to the following rubric:

+3

Up to three points are added for excellent performance; the student displays a high degree of competence across a range of questions, and excels in at least one group of criteria. Relative weaknesses in some areas may be compensated by conspicuous strengths in others.

+/- 0

The marked assignment is unchanged for fair performance. Answers to questions must show evidence of some solid competence in expounding evidence and analysis. But they will be marred by weakness under one or more criteria: failure to discuss the question directly; appeal to irrelevant information; factual error; narrowness in the range of issues addressed or evidence adduced; shortage of detailed evidence; or poor organisation and presentation, including consistently incorrect grammar. Answers may be characterised by unsubstantiated assertion rather than argument, or by unresolved contradictions in the argument.

- 3 (up to three points)

Up to three are subtracted points for an inability to answer multiple basic questions about themes in the written work. Answers to questions will fall down on a number of criteria, but will exhibit some vestiges of the qualities required, such as the ability to see the point of the question, to deploy information, or to offer some coherent analysis towards an argument. Such qualities will not be displayed at a high level or consistently, and will be marred by irrelevance, incoherence, error and poor organisation and presentation.

0 (fail)

Written work and the oral examination will both be failed if the oral examination clearly demonstrates that the work was plagiarised. The student is unfamiliar with the arguments of the assignment or the sources used for those arguments.

Plagiarism

Plagiarism is the use of someone else's work without correct referencing. The consequence of plagiarism is the presentation of someone else's work as your own work. Plagiarism violates Woolf policy and will result in disciplinary action, but the context and seriousness of plagiarism varies widely. Intentional or reckless plagiarism will result in a penalty grade of zero, and may also entail disciplinary penalties.

Plagiarism can be avoided by citing the works that inform or that are quoted in a written submission. Many students find that it is essential to keep their notes organised in relation to the sources which they summarise or quote. Course instructors will help you to cultivate professional scholarly habits in your academic writing.

Depending on the course, short assignment essays may not require students to submit a bibliography or to use extensive footnotes, and students are encouraged to write their assignments entirely in their own words. However, all essays must acknowledge the sources on which they rely and must provide quotation marks and citation information for verbatim quotes.

There are several forms of plagiarism. They all result in the presentation of someone's prior work as your new creation. Examples include:

- Cutting and pasting (verbatim copying)
- Paraphrasing or rewording
- Unauthorised Collaboration
- Collaboration with other students can result in pervasive similarities – it is important to determine in advance whether group collaboration is allowed, and to acknowledge the contributions or influence of the group members.
- False Authorship (Essay Mills, Friends, and Language Help)

- Paying an essay writing service, or allowing a generous friend to compose your essay, is cheating. Assistance that contributes substantially to the ideas or content of your work must be acknowledged.

Complaints and appeals

Students and faculty should always seek an amicable resolution to matters arising by addressing the issue with the person immediately related to the issue. Students should handle minor misunderstandings or disagreements within a regular teaching session or by direct message, or with their College. If a simple resolution is not possible, or the matter remains unresolved for one party, the steps outlined in this section apply to all groups, colleges, and units of Woolf.

The Red Flag system

An issue with a red flag should be submitted in the case that a member of Woolf seeks to make an allegation of serious misconduct about another member, including matters of cheating, plagiarism, and unfair discrimination or intolerance.

Any member of Woolf, seeking to raise a matter of serious concern, should submit a red flag by emailing redflag@woolf.education. Provide a short, clear description of the issue.

If a student submits an issue with a red flag, or if a faculty member submits an issue about a student, it will trigger a meeting with the student's College Advisor. If the issue is not resolved, the matter will be escalated to the College Dean, or to a committee designated by the College Dean, which will have the power to clear the flag.

If an issue is submitted with a red flag by a faculty member about another faculty member, then the issue is reported directly to the College Dean.

For both students and faculty members, after the Dean's decision, the one who submits the complaint is provided the opportunity to accept or appeal the decision; if the one submitting the issue appeals the decision, it will be assigned to the Quality Assurance, Enhancement, and Technology Alignment Committee, which is a subcommittee of the Faculty Council.

Mitigating circumstances

When serious circumstances ('Mitigating Circumstances'), beyond the control of a student or faculty member, adversely affect academic performance or teaching support, a Mitigating Circumstances report must be submitted using Woolf's red flagging system. Mitigating Circumstances may include but are not limited to serious medical problems, domestic and personal circumstances, major accidents or interruptions of public services, disturbances during examination, or serious administrative or procedural errors with a material effect on outcomes.

Mitigating circumstances do not normally include a member's personal technology problems, including software, hardware, or personal internet connection failures; employment obligations or changes in employment obligations; permanent or sustained medical conditions (unless there is a sudden change of condition); or circumstances where no official evidence has been submitted.

Mitigating circumstances are normally only considered when a red flag has been submitted for the issue before the deadline of an affected written project or assignment, or within one week of a cumulative

examination. Proof of mitigating circumstances may result in an extended deadline or examination period, or the possibility to retake an examination; it will not result in any regrading of existing submissions or exams.

Grade appeals

Students who dissent from the grades they have received should follow the normal procedure for submitting a red flag.