Master of Science in Artificial Intelligence

Handbook

April 2026

Introduction

Master of Science in Artificial Intelligence

This course is designed for individuals who wish to enhance their knowledge of computer science and artificial intelligence and its various applications used in different fields of employment. It is designed for those that will have responsibility for planning, organising, and directing technological operations as well as individuals seeking to transition into AI-focused roles.

Entry requirements

Education Requirements

Candidates who apply for this course will ordinarily have an EQF Level 6 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 proficiency (minimum B2 level or equivalent) is required.

Instructional design

Teaching: Online delivery provides significant advantages in terms of accessibility — students can more easily reach academic experts across borders and better integrate study within their own life or career. Courses are offered on a platform facilitating synchronous and asynchronous teaching, incorporating videoconferencing and group discussions. Students engage through instructor-led sessions, peer discussions, collaborative projects, asynchronous self-paced learning, and practical laboratory work via cloud-based tools such as GitHub, Google Cloud Platform, AWS, and Replit.

Assessment: Assessment criteria include directness of engagement with the assignment, depth and sophistication of comprehension, coherence and independence of argument, and conceptual precision. Weightings: Coursework 40%, Practical assignments/projects 35%, Examinations 25% (may vary by module).

Degree structure

The degree consists of 51 elective modules totalling 260 ECTS, from which students select 90 ECTS to complete the programme. All modules are at EQF 7.

Module	ECTS	Level
Introduction to Computer Programming: Part 1	5	EQF 7

Relational Databases	5	EQF 7
Introduction to Problem Solving Techniques: Part 1	5	EQF 7
Practical Software Engineering	5	EQF 7
JavaScript	5	EQF 7
Front End Development	5	EQF 7
Back End Development	5	EQF 7
Spreadsheets for Data Understanding	5	EQF 7
Data Visualisation Tools	5	EQF 7
Front End UI/UX Development	5	EQF 7
Numerical Programming in Python	5	EQF 7
Introduction to Machine Learning	5	EQF 7
Introduction to Deep Learning	5	EQF 7
Foundations of Cloud Computing	5	EQF 7
Applied Statistics	5	EQF 7
Introduction to Computer Programming: Part 2	5	EQF 7
Data Engineering	5	EQF 7
Product Analytics	5	EQF 7
Applied Computer Science Project	10	EQF 7

Project Management	5	EQF 7
Linux and Shell Scripting	5	EQF 7
Operating Systems	5	EQF 7
Databases and Computer Networks	5	EQF 7
DevOps Tools Part 1	5	EQF 7
DevOps Tools Part 2	5	EQF 7
Amazon Web Services Part 1	5	EQF 7
Amazon Web Services Part 2	5	EQF 7
Advanced Algorithms Part 2	5	EQF 7
Computational Models	5	EQF 7
Artificial Intelligence Integration Strategies	5	EQF 7
Emerging Artificial Intelligence Technologies	5	EQF 7
Machine Learning Applications	5	EQF 7
Artificial Intelligence for Decision Making	5	EQF 7
Intelligent Data Processing	5	EQF 7
Introduction to Artificial Intelligence	5	EQF 7
Data Science Principles	5	EQF 7
Computational Intelligence	5	EQF 7

Algorithmic Thinking	5	EQF 7
Intelligent Systems	5	EQF 7
Cognitive Computing	5	EQF 7
Predictive Modelling	5	EQF 7
Neural Networks and Deep Learning	5	EQF 7
Applied Data Analytics	5	EQF 7
Robotics and Automation	5	EQF 7
Natural Language Processing	5	EQF 7
Ethical Artificial Intelligence Practices	5	EQF 7
Advanced Artificial Intelligence Concepts	5	EQF 7
Artificial Intelligence in Industry Applications	5	EQF 7
Training Large Language Models	5	EQF 7
Prompt Engineering	5	EQF 7
Artificial Intelligence Product Management	5	EQF 7

Module Descriptions

1. Introduction to Computer Programming: Part 1

This course helps students translate advanced mathematical/statistical/scientific concepts into code. This is a module for writing code to solve real-world problems. It introduces programming concepts (such as control structures, recursion, classes and objects) assuming no prior programming knowledge, to make this course accessible to advanced professionals from scientific fields like Biology, Physics, Medicine,

Chemistry, Civil & Mechanical Engineering etc. After building a strong foundation for converting scientific knowledge into programming concepts, the course advances to dive deeply into Object-Oriented Programming and its methodologies. It also covers when and how to use inbuilt-data structures like 1-Dimensional and 2-Dimensional Arrays before introducing the concepts of computational complexity to help students write optimised code using appropriate data structures and algorithmic design methods.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to converting scientific knowledge into programming concepts, and how to instantiate these using Object-Oriented methods.
2. Apply a professional and scholarly approach to research problems pertaining to computational complexity.
3. Efficiently manage interdisciplinary issues that arise in connection to data structured in 1- and 2-dimensional arrays.
4. Demonstrate self-direction in research and originality in solutions developed for modern programming languages.
5. Act autonomously in identifying research problems and solutions related to Object-Oriented programming.
6. Solve problems and be prepared to take leadership decisions related to the methods and principles of computer programming.
7. Develop a critical understanding of a modern programming language such as Java or Python.
8. Develop a specialised knowledge of key strategies related to Object-Oriented Programming.
9. Acquire knowledge of various methods for structuring data.
10. Critically evaluate diverse scholarly views on computational complexity.
11. Critically assess the relevance of theories for business applications in the domain of technology.
12. Autonomously gather material and organise it into a coherent presentation or essay.
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing.
14. Creatively apply various programming methods to develop critical and original solutions to computational problems.
15. Apply an in-depth domain-specific knowledge and understanding to computer programming.

2. 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. Students would start with diagrammatic tools (ER-diagram) to map a real world data storage problem into entities, relationships and keys. Then, they learn to translate the ER-diagram into a relational model with tables. SQL is then introduced as a de facto tool to create, design and query relational databases. Finally, the course would help students understand data normalisation and optimisation for practical database systems.

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 related to modelling different business scenarios.
4. Demonstrate self-direction in research and originality in designing database systems.

5. Act autonomously in identifying research problems and solutions in relation to Relational Database Systems.
6. Solve problems and take leadership decisions related to database system design.
7. Develop a critical understanding of Entity-Relationship modelling and SQL.
8. Develop a specialised knowledge of database modelling techniques.
9. Acquire knowledge of data normalisation methods and their implications for database systems.
10. Critically evaluate different approaches to solving relational database problems.
11. Critically assess the relevance of database design decisions for business applications.
12. Autonomously gather material and present database design proposals.
13. Employ the standard modern conventions for documentation and referencing.
14. Creatively apply various design methods to develop optimal database systems.
15. Apply an in-depth domain-specific knowledge to relational database systems.

3. Introduction to Problem Solving Techniques: Part 1

The ability to solve problems is a skill, and just like any other skill, the more one practices, the better one gets. So while on this course students will learn about a range of problem solving techniques and will learn to recognise which problem-solving technique is more suitable for a given problem scenario. A distinction is made between algorithmic problem-solving (for structured/understood problems) and heuristic problem solving (for unstructured/ill-defined problems). The course focusses on data structures and algorithms which are the foundational building blocks for computationally solving complex problems.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to problem-solving, and moving from algorithmic to heuristic problem-solving strategies.
2. Apply a professional and scholarly approach to research problems pertaining to data structures.
3. Efficiently manage interdisciplinary issues that arise in problem-solving methodologies.
4. Demonstrate self-direction in research and originality in solutions for problem-solving approaches.
5. Act autonomously in identifying research problems and solutions related to data structures and algorithms.
6. Solve complex problems and be prepared to take leadership decisions related to problem-solving methodologies.
7. Develop a critical understanding of various data structures and their algorithmic properties.
8. Develop a specialised knowledge of when and how to apply different problem-solving techniques.
9. Acquire knowledge of the time and space complexities of various algorithms.
10. Critically evaluate different problem-solving approaches for specific scenarios.
11. Critically assess the appropriateness of algorithmic versus heuristic approaches.
12. Autonomously organise and present solutions for complex problems.
13. Employ the standard modern conventions for algorithmic notation and presentation.
14. Creatively apply various algorithmic approaches to develop efficient solutions.
15. Apply an in-depth domain-specific knowledge to data structures and algorithms.

4. Practical Software Engineering

This course gives the detailed overview on how to approach Low Level Design problems with real-world case studies discussed such as Designing a Pen (Mac/Windows), TicTacToe, BookMyShow (most used event booking app, manages millions of users), Email campaign Management System and detailed design of Splitwise.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to specifying the internal logic of software.
2. Apply a professional and scholarly approach to research problems pertaining to logical and functional design of software components.
3. Efficiently manage interdisciplinary issues that arise in connection to developing hierarchical input process output (HIPO) models.
4. Demonstrate self-direction in research and originality in solutions developed for using Program Design Languages.
5. Act autonomously in identifying research problems and solutions related to refining software designs.
6. Solve problems and be prepared to take leadership decisions related to developing code-ready low-level design documents.
7. Develop a critical understanding of software design and refinement processes.
8. Develop a specialised knowledge of Process Design Languages and flowchart methods for describing desired functions and behaviours.
9. Acquire knowledge of various methods for specifying the logical and functional design of a system.
10. Critically evaluate diverse scholarly views on the appropriateness of various approaches to converting high-level or architectural software design to low-level, component-oriented design.
11. Critically assess the relevance of theories of software design processes for business applications in the realm of software engineering.
12. Autonomously gather material and organise it into a coherent presentation or essay.
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing.
14. Creatively apply various visual and written methods for converting architectural/high-level designs to component-oriented, low-level designs.
15. Apply an in-depth domain-specific knowledge and understanding of the importance of refinement in software design processes.

5. JavaScript

This course is a hands-on course covering JavaScript from basics to advanced concepts in detail using multiple examples. We start with basic programming concepts like variables, control statements, loops, classes and objects. Students also learn basic data-structures like Strings, Arrays and dates. Students also learn to debug our code and handle errors gracefully in code. We learn popular style guides and good coding practices to build readable and reusable code which is also highly performant. We then learn how web browsers execute JavaScript code using V8 engine as an example. We also cover concepts like JIT-compiling which helps JS code to run faster. This is followed by slightly advanced concepts like DOM, Async-functions, Web APIs and AJAX which are very popularly used in modern front end development. We learn how to optimise JavaScript code to run on both mobile apps and mobile browsers along with Desktop browsers and as desktop apps via ElectronJS. Most of this course would be covered via real world examples and by learning from JS code of popular open-source websites and libraries

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to JavaScript.
2. Apply a professional and scholarly approach to research problems pertaining to JavaScript.
3. Efficiently manage interdisciplinary issues that arise in connection to JavaScript.
4. Demonstrate self-direction in research and originality in solutions developed for JavaScript.
5. Act autonomously in identifying research problems and solutions related to JavaScript.
6. Solve problems and be prepared to take leadership decisions related to the methods and principles of JavaScript.
7. Develop a critical knowledge of JavaScript.
8. Develop a specialised knowledge of key strategies related to JavaScript.
9. Acquire knowledge of popular style guides and good coding practices to build readable and reusable code which is also highly performant.
10. Critically evaluate diverse scholarly views on JavaScript.
11. Critically assess the relevance of theories for business applications in the domain of technology.
12. Autonomously gather material and organise into coherent problem sets or presentations.
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing.
14. Creatively apply JavaScript concepts to develop critical and original solutions for computational problems.
15. Apply an in-depth domain-specific knowledge and understanding to JavaScript tools.

6. Front End Development

This course builds upon the introductory JavaScript course to acquaint students of popular and modern frameworks to build the front end. We focus on three very popular frameworks/libraries in use: React.js, jQuery and AngularJS. We start with React.js, one of the most popular and advanced ones amongst the three. students learn various components and data flow to learn to architect real world front end using React.js. This would be achieved via multiple code examples and code-walkthroughs from scratch. We would also dive into React Native which is a cross platform Framework to build native mobile and smart-TV apps using JavaScript. This helps students to build applications for various platforms using only JavaScript. jQuery is one of the oldest and most widely used JavaScript libraries, which students cover in detail. Students specifically focus on how jQuery can simplify event handling, AJAX, HTML DOM tree manipulation and create CSS animations. We also provide a hands-on introduction to AngularJS to architect model-view-controller (MVC) based dynamic web pages.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to front end development.
2. Apply a professional and scholarly approach to research problems pertaining to front end development.
3. Efficiently manage interdisciplinary issues that arise in connection to front end development.
4. Demonstrate self-direction in research and originality in solutions developed for front end development.
5. Act autonomously in identifying research problems and solutions related to front end development.
6. Solve problems and be prepared to take leadership decisions related to the methods and principles of front end development.
7. Develop a critical knowledge of front end development.
8. Develop a specialised knowledge of key strategies related to front end development.
9. Acquire knowledge of popular frameworks/libraries in use: React.js, jQuery and AngularJS.
10. Critically evaluate diverse scholarly views on front end development.
11. Critically assess the relevance of theories for business applications in the domain of technology.

12. Autonomously gather material and organise it into coherent problem sets or presentations.
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing.
14. Creatively apply front end development applications to develop critical and original solutions for computational problems.
15. Apply an in-depth domain-specific knowledge and understanding to front end development solutions.

7. Back End Development

This is a foundational course on building server-side (or backend) applications using popular JavaScript runtime environments like Node.js. Students will learn event driven programming for building scalable backend for web applications. The module teaches various aspects of Node.js like setup, package manager, client-server programming and connecting to various databases and REST APIs. Most of these concepts would be covered in a hands-on manner with real world examples and applications built from scratch using Node.js on Linux servers. This course also provides an introduction to Linux server administration and scripting with special focus on web-development and networking. Students learn to use Linux monitoring tools (like Monit) to track the health of the servers. The module also provides an introduction to Express.js which is a popular light-weight framework for Node.js applications. Given the practical nature of this course, this would involve building actual website backends via assignments/projects for ecommerce, online learning and/or photo-sharing.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to Back End Development.
2. Apply a professional and scholarly approach to research problems pertaining to Back End Development.
3. Efficiently manage interdisciplinary issues that arise in connection to Back End Development.
4. Demonstrate self-direction in research and originality in solutions developed for Back End Development.
5. Act autonomously in identifying research problems and solutions related to Back End Development.
6. Solve problems and be prepared to take leadership decisions related to the methods and principles of Back End Development.
7. Develop a critical knowledge of Back End Development.
8. Develop a specialised knowledge of key strategies related to Back End Development.
9. Acquire knowledge of key aspects of Node.js like setup, package manager, client-server programming and connecting to various databases and REST.
10. Critically evaluate diverse scholarly views on Back End Development.
11. Critically assess the relevance of theories for business applications in the domain of technology.
12. Autonomously gather material and organise it into coherent problem sets or presentations.
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing.
14. Creatively apply Back End Development tools to develop critical and original solutions for computational problems.
15. Apply an in-depth domain-specific knowledge and understanding to Back End Development applications.

8. Spreadsheets for Data Understanding

Spreadsheets for Data Understanding introduces students to the principles and techniques of data cleaning, handling data sets of varying sizes, and visualising data/data storytelling. Students will also learn the basics of predictive modelling from data using spreadsheet applications. This is a foundational module that complements other modules in the data science and data engineering tracks. The module is designed to provide hands-on practice in a business analytics environment. This module also covers advanced spreadsheet features including formulas, functions, pivot tables and macros. Relevant case studies from the finance sector and supply chain will be discussed. The course will be delivered using both MS-Excel and LibreOffice Calc with special focus on scenarios of working with large data sets in a business environment.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to representing and storing data in spreadsheets.
2. Apply a professional and scholarly approach to research problems pertaining to the appropriate use of functions and formulas in a spreadsheet.
3. Efficiently manage interdisciplinary issues that arise in spreadsheet-based data processing and modelling.
4. Demonstrate self-direction in research and originality in solutions developed for spreadsheet applications.
5. Act autonomously in identifying research problems and solutions related to spreadsheet use in real-world scenarios.
6. Solve problems and be prepared to take leadership decisions related to using spreadsheets for business analytics.
7. Develop a critical understanding of data representation in spreadsheets and data cleaning techniques.
8. Develop a specialised knowledge of spreadsheet formulas, functions, and advanced features like pivot tables and macros.
9. Acquire knowledge of data cleaning and preparation techniques for analytics.
10. Critically evaluate different approaches to data representation and analysis in a spreadsheet environment.
11. Critically assess the appropriateness of using spreadsheets for specific business analytics problems.
12. Autonomously organise and present data analysis results and insights.
13. Employ the standard modern conventions for data presentation and documentation.
14. Creatively apply spreadsheet techniques to develop solutions for business analytics problems.
15. Apply an in-depth domain-specific knowledge to data analysis using spreadsheet applications.

9. Data Visualisation Tools

This course is aimed to build a strong foundational knowledge of Data Analytics tools used extensively in the Data Science field. There are now powerful data visualisation tools used in the business analytics industry to process and visualise raw business data in actionable insights. This course focuses on popular industry tools like Tableau and Power BI. Students will also get familiar with open-source visualisation tools like Python Matplotlib and R ggplot2 to create beautiful and informative statistical graphics. Relevant case studies from the finance, retail and supply chain sectors will be discussed. The course covers the entire pipeline from data connection, data cleaning and preparation, creating interactive dashboards and storytelling with data.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to time and space complexity in data science.
2. Apply a professional and scholarly approach to research problems pertaining to data visualisations, including dashboards and storytelling.
3. Efficiently manage interdisciplinary issues that arise in connection to data visualisation workflows.
4. Demonstrate self-direction in research and originality in solutions developed for data visualisation applications.
5. Act autonomously in identifying research problems and solutions related to data visualisation in business contexts.
6. Solve problems and be prepared to take leadership decisions related to designing data visualisations and dashboards.
7. Develop a critical understanding of data visualisation principles and design.
8. Develop a specialised knowledge of popular data visualisation tools like Tableau, Power BI, Matplotlib, and ggplot2.
9. Acquire knowledge of data storytelling and communication strategies.
10. Critically evaluate different approaches to visualising data for specific business problems.
11. Critically assess the effectiveness of visualisation designs in communicating insights.
12. Autonomously design and create interactive dashboards and reports.
13. Employ the standard modern conventions for data visualisation and presentation.
14. Creatively apply visualisation techniques to develop compelling data stories.
15. Apply an in-depth domain-specific knowledge to data analytics and visualisation.

10. Front End UI/UX Development

This is a hands-on course on designing responsive, modern, and lightweight UI for web, mobile, and desktop applications using HTML5, CSS, and Frameworks like Bootstrap 4. This course starts with an introduction to how web browsers, mobile apps, and web servers interact in a typical HTTP web request/response cycle. The course will then dive into the HTML5 standard and its new features. CSS is then covered in detail with focus on responsive design patterns and the new CSS Flexbox and Grid specifications. Bootstrap 4 framework is then used to rapidly prototype and develop real-world user interface. The course also covers user experience design principles like accessibility (WCAG standards), usability, and user testing. This module will include a project where students will design and develop a modern responsive website application for a simulated scenario.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to Front end UI/UX development.
2. Apply a professional and scholarly approach to research problems pertaining to Front end UI/UX development.
3. Efficiently manage interdisciplinary issues that arise in connection to Front end UI/UX development.
4. Demonstrate self-direction in research and originality in solutions developed for Front end UI/UX development.
5. Act autonomously in identifying research problems and solutions related to Front end UI/UX development.
6. Solve problems and be prepared to take leadership decisions related to the methods and principles of Front end UI/UX development.
7. Develop a critical knowledge of Front end UI/UX development.
8. Develop a specialised knowledge of key strategies related to Front end UI/UX development.

9. Acquire knowledge of HTML5, CSS3, and Bootstrap 4 frameworks.
10. Critically evaluate diverse scholarly views on Front end UI/UX development.
11. Critically assess the relevance of theories for business applications in the domain of technology.
12. Autonomously gather material and organise it into coherent problem sets or presentations.
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing.
14. Creatively apply Front end UI/UX development tools to develop critical and original solutions.
15. Apply an in-depth domain-specific knowledge and understanding to Front end UI/UX development.

11. Numerical Programming in Python

This course helps students translate mathematical/statistical/scientific concepts into code. This is a foundational course for writing code to solve Data Science ML & AI problems. It introduces basic programming concepts (like control structures, recursion, classes and objects) from scratch, assuming no prerequisites, to make this course accessible to students from non-computational scientific fields like Biology, Physics, Medicine, Chemistry, Civil & Mechanical Engineering etc. After building a strong foundation, the course advances to dive deep into core Mathematical libraries like NumPy, Scipy and Pandas. Students also learn when and how to use inbuilt-data structures like Lists, Dicts, Sets and Tuples. The module introduces the concepts of computational complexity to help students write optimised code using appropriate data structures and algorithmic design methods. The module does not dive deep into the data structures and algorithm design methods in this course -- that is available in the 'Data Structures and Algorithms' module. This course is valuable for all students specialising in mathematical sub-areas of CS like ML, Data Science, Scientific Computing etc.

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 Numerical programming in Python.
4. Demonstrate self-direction in research and originality in solutions developed for Numerical programming in Python.
5. Act autonomously in identifying research problems and solutions related to Numerical programming in Python.
6. Solve problems and be prepared to take leadership decisions related to the methods and principles of Numerical programming in Python.
7. Develop a critical knowledge of Numerical programming in Python.
8. Develop a specialised knowledge of key strategies related to Numerical programming in Python.
9. Acquire knowledge of core Mathematical libraries like NumPy, Scipy and Pandas.
10. Critically evaluate diverse scholarly views on Numerical programming in Python.
11. Critically assess the relevance of theories for business applications in the domain of technology.
12. Autonomously gather material and organise it into a coherent problem set or presentation.
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing.
14. Create new solutions that are critical to solving computational problems through creatively applying code writing.
15. Apply an in-depth domain-specific knowledge and understanding to numerical programming in Python.

12. Introduction to Machine Learning

This course focuses on building basic classification and regression models and understanding these models rigorously both with a mathematical and an applicative focus. The module starts with a basic introduction to high dimensional geometry of points, distance-metrics, hyperplanes and hyperspheres. We build on top this to introduce the mathematical formulation of logistic regression to find a separating hyperplane. Students learn to solve the optimization problem using vector calculus and gradient descent (GD) based algorithms. The module introduces computational variations of GD like mini-batch and stochastic gradient descent. Students also learn other popular classification and regression methods like k-Nearest Neighbours, NaiveI Bayes, Decision Trees, Linear Regression etc. Students also learn how each of these techniques under various real world situations like the presence of outliers, imbalanced data, multi class classification etc. Students learn bias and variance trade-off and various techniques to avoid overfitting and underfitting. Students also study these algorithms from a Bayesian viewpoint along with geometric intuition. This module is hands-on and students apply all these classical techniques to real world problems.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to machine learning.
2. Apply a professional and scholarly approach to research problems pertaining to machine learning.
3. Efficiently manage interdisciplinary issues that arise in connection to machine learning.
4. Demonstrate self-direction in research and originality in solutions developed for machine learning.
5. Act autonomously in identifying research problems and solutions related to machine learning.
6. Solve problems and be prepared to take leadership decisions related to the methods and principles of machine learning.
7. Develop a critical knowledge of machine learning.
8. Develop a specialised knowledge of key strategies related to machine learning.
9. Acquire knowledge of bias and variance trade-off, and various techniques to avoid overfitting and underfitting.
10. Critically evaluate diverse scholarly views on machine learning.
11. Critically assess the relevance of theories for business applications in the domain of technology.
12. Autonomously gather material and organise it into coherent problem sets and presentation.
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing.
14. Creatively apply regression models to develop critical and original solutions for computational issues.
15. Apply an in-depth domain-specific knowledge and understanding to machine learning solutions.

13. Introduction to Deep Learning

This course provides a strong mathematical and applicative introduction to Deep Learning. The course starts with the perceptron model as an over simplified approximation to a biological neuron. We motivate the need for a network of neurons and how they can be connected to form a Multi Layered Perceptron (MLPs). This is followed by a rigorous understanding of back-propagation algorithms and its limitations from the 1980s. Students study how modern deep learning took off with improved computational tools and data sets. We teach more modern activation units (like ReLU and SeLU) and how they overcome problems with the more classical Sigmoid and Tanh units. Students learn weight initialization methods, regularisation by dropouts, batch normalisation etc., to ensure that deep MLPs can be successfully trained.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to Deep Learning.
2. Apply a professional and scholarly approach to research problems pertaining to Deep Learning.
3. Efficiently manage interdisciplinary issues that arise in connection to Deep Learning.
4. Demonstrate self-direction in research and originality in solutions developed for Deep Learning.
5. Act autonomously in identifying research problems and solutions related to Deep Learning.
6. Solve problems and be prepared to take leadership decisions related to the methods and principles of Deep Learning.
7. Develop a critical knowledge of Deep Learning.
8. Develop a specialised knowledge of key strategies related to Deep Learning.
9. Acquire knowledge of deep learning systems like ADAM, AdaGrad, RMSProp etc. Students also learn AutoEncoders, VAEs and Word2Vec.
10. Critically evaluate diverse scholarly views on Deep Learning.
11. Critically assess the relevance of theories for business applications in the domain of technology.
12. Autonomously gather material and organise it into coherent problem sets or presentation.
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing.
14. Creatively apply Deep Learning techniques to develop critical and original solutions for computational problems.
15. Apply an in-depth domain-specific knowledge and understanding to Deep Learning.

14. Foundations of Cloud Computing

This is a course that focuses both on architectural design and practical hands-on learning of the most used cloud services. The module extensively uses Amazon Web services (AWS) to show real world code examples of various cloud services. It also covers the core concepts and architectures in a platform agnostic manner so that students can easily translate these learnings to other cloud platforms (like Azure, GCP etc.). The course starts with virtualization and how virtualized compute instances are created and configured. Students also learn how to auto-scale applications using load balancers and build fault tolerant applications across a geographically distributed cloud. As relational databases are widely used in most enterprises, students learn how to migrate and scale (both vertically and horizontally) these databases on the cloud while ensuring enterprise grade security. Virtual private clouds enable us to create a logically isolated virtual network of computer resources. Students learn to set up a VPC using virtualized-compute-servers on AWS. The course also covers the basics of networking while setting up a VPC. Students learn of the architecture and practical aspects of distributed object storage and how it enables low latency and high availability data storage on the cloud.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to cloud computing.
2. Apply a professional and scholarly approach to research problems pertaining to cloud computing.
3. Efficiently manage interdisciplinary issues that arise in connection to cloud computing.
4. Demonstrate self-direction in research and originality in solutions developed for cloud computing.
5. Act autonomously in identifying research problems and solutions related to cloud computing.
6. Solve problems and be prepared to take leadership decisions related to the methods and principles of cloud computing.
7. Develop a critical knowledge of cloud computing.
8. Develop a specialised knowledge of key strategies related to cloud computing.
9. Acquire knowledge of virtualization and how virtualized compute instances are created and configured.
10. Critically evaluate diverse scholarly views on cloud computing.

11. Critically assess the relevance of theories for business applications in the domain of technology.
12. Autonomously gather material and organise it into coherent problems sets or presentations.
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing.
14. Creatively apply cloud computing applications to develop critical and original solutions for computational problems.
15. Apply an in-depth domain-specific knowledge and understanding to cloud computing services.

15. Applied Statistics

This course introduces basic probability theory, statistical methods and computational algorithms to perform mathematically rigorous data analysis. The course starts with basic foundational concepts of random variables, histograms, and various plots (PMF, PDF and CDF). Students learn various popular discrete and continuous distributions like Bernoulli, Binomial, Poisson, Gaussian, Exponential, Pareto, log-normal etc., both mathematically and from an applicative perspective. Students learn various measures like mean, median, percentiles, quantiles, variance and interquartile-range. Students learn the pros and cons of each metric and understand when and how to use them in practice. Students will learn conditional probability and Bayes theorem in the applied context of real-world problems in medicine and healthcare. The module teaches the foundations of non-parametric statistics and applies them to solve problems using computational tools. Students learn various methods to determine correlations rigorously in data. This is followed by applied and mathematical understanding of the statistics underlying control-treatment (A/B) experiments and hypothesis testing. The module engages computation tools in modern statics like Bootstrapping, Monte-Carlo methods, RANSAC etc.

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 probability theory to perform mathematically rigorous data analysis.
3. Efficiently manage interdisciplinary issues that arise in connection to Applied Statistics.
4. Demonstrate self-direction in research and originality in solutions developed for Applied Statistics.
5. Act autonomously in identifying research problems and solutions related to Applied Statistics.
6. Solve problems and be prepared to take leadership decisions related to the methods and principles of Applied Statistics.
7. Develop a critical knowledge of Applied Statistics.
8. Develop a specialised knowledge of key strategies related to Applied Statistics.
9. Acquire knowledge of popular discrete and continuous distributions (like Bernoulli, Binomial, Poisson, Gaussian, Exponential, Pareto, and log-normal).
10. Critically evaluate diverse scholarly views on Applied Statistics.
11. Critically assess the relevance of theories for business applications in the domain of technology.
12. Autonomously gather material and organise it into a coherent problem set or presentation.
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing.
14. Creatively apply basic probability theory to develop critical and original solutions for computational problems.
15. Apply an in-depth domain-specific knowledge and understanding of applied statistics.

16. Introduction to Computer Programming: Part 2

This course provides a practical and detailed understanding of popular programming paradigms and data storage types. Students learning this will be able to write and solve programming problems. The course starts from the basics about functions, various built in functions and how to code user defined functions. Then students will learn about various data type storages and learn about lists and how various manipulations can be done lists like list slicing and also go through examples of 2D Lists.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to data storage and how popular programming languages handle this.
2. Apply a professional and scholarly approach to research problems pertaining to functions and data types.
3. Efficiently manage interdisciplinary issues that arise in connection to choosing the best data type for a particular programming need.
4. Demonstrate self-direction in research and originality in handling data in lists.
5. Act autonomously in identifying research problems and solutions related to data storage.
6. Solve problems and be prepared to take leadership decisions related to programming concepts such as lists, sets, tuples, dictionaries, and strings.
7. Develop a critical understanding of product design and development.
8. Develop a specialised knowledge of the various uses and forms of lists in programming, including 2D lists.
9. Acquire knowledge of various methods for storing data in modern programming languages.
10. Critically evaluate diverse scholarly views on functions and algorithms.
11. Critically assess the relevance of theories of data storage for programming.
12. Autonomously gather material and organise it into a coherent presentation or essay.
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing.
14. Creatively apply various visual, written, and code-based methods for manipulating tuples, strings, lists, and similar structures.
15. Apply an in-depth domain-specific knowledge and understanding of computer programming and data management.

17. Data Engineering

Data is the fuel driving all major organisations. This course helps you understand how to process data at scale.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to the data engineering lifecycle.
2. Apply a professional and scholarly approach to research problems pertaining to data warehousing and modelling.
3. Efficiently manage interdisciplinary issues that arise in connection to developing cloud solutions for data engineering problems.
4. Demonstrate self-direction in research and originality in creating advanced SQL queries.
5. Act autonomously in identifying research problems and solutions related to developing for data at scale.
6. Solve problems and be prepared to take leadership decisions related to developing pipelines to handle massive datasets for engineering purposes.
7. Develop a critical understanding of data engineering.

8. Develop a specialised knowledge of standard tools for data processing, such as Apache Kafka, Airflow, and Spark (with PySpark), and the Hadoop Ecosystem.
9. Acquire knowledge of various methods for warehousing data.
10. Critically evaluate diverse scholarly views on best practices in developing data-intensive applications.
11. Critically assess the relevance of theories of data modelling for efficient pipeline creation.
12. Autonomously gather material and organise it into a coherent presentation or essay
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing
14. Creatively apply various visual and written methods for dashboarding data with Grafana/Tableau.
15. Apply an in-depth domain-specific knowledge and understanding of orchestrating complete ETL pipelines.

18. Product Analytics

This course teaches students how to analyse the ways users engage with a service. This method, called product analytics, helps businesses track and analyse user data. Students will learn more deeply what is required to move a product from idea to implementation, through to launch, and then on to iterative improvements. The course teaches how to measure progress, validate or update product hypotheses, and present product learnings.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to product sense, and how to tell whether a product is worth bringing to market.
2. Apply a professional and scholarly approach to research problems pertaining to measuring user engagement.
3. Efficiently manage interdisciplinary issues that arise in connection to designing a product and bringing it to market.
4. Demonstrate self-direction in research and originality in testing and validating hypotheses about a product and its users.
5. Act autonomously in identifying research problems and solutions related to product analytics.
6. Solve problems and be prepared to take leadership decisions related to developing data-informed business cases about bringing products to market and iterating upon them.
7. Develop a critical understanding of product design and development.
8. Develop a specialised knowledge of frameworks for measuring user engagement, such as diagnostics, key performance indicators (KPI), and other metrics.
9. Acquire knowledge of various methods for testing hypotheses about the viability of a product and about how users engage with it.
10. Critically evaluate diverse scholarly views on assessing user behaviours.
11. Critically assess the relevance of theories of user behaviour for product development.
12. Autonomously gather material and organise it into a coherent presentation or essay
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing
14. Creatively apply various visual and written methods for proposing a technical solution to a real-world problem to other technical and managerial-level audiences, and for documenting that solution
15. Apply an in-depth domain-specific knowledge and understanding of system design and implementation in business

19. Applied Computer Science Project

This is a project-based course, with the aim of building the required skills for creating web-based software systems. The course covers the entire lifecycle of building software projects, from requirement gathering and scope definition from a product document, to designing the architecture of the system, and all the way to delivery and maintenance of the software system.

Learning Outcomes

1. Create synthetic contextualised discussions of key issues related to real-world software design, implementation, and deployment situations.
2. Apply a professional and scholarly approach to research problems pertaining to real-world computational complexities.
3. Efficiently manage interdisciplinary issues that arise in connection to deploying a modern, web-based system.
4. Demonstrate self-direction in research and originality in solutions developed for robust and reliable cloud deployments.
5. Act autonomously in identifying research problems and solutions related to modern computational tools and methods.
6. Solve problems and be prepared to take leadership decisions related to developing and deploying cloud-oriented software solutions
7. Develop a critical understanding of modern computational applications.
8. Develop a specialised knowledge of key strategies for designing well-architected information management systems.
9. Acquire knowledge of various methods for version control.
10. Critically evaluate diverse scholarly views on database management.
11. Critically assess the relevance of theories of web security for cloud deployment.
12. Autonomously gather material and organise it into a coherent presentation or essay.
13. Employ the standard modern conventions for the presentation of scholarly work and scholarly referencing.
14. Creatively apply various visual and written methods for proposing a technical solution to a real-world problem to other technical and managerial-level audiences, and for documenting that solution.
15. Apply an in-depth domain-specific knowledge and understanding of system design and implementation in business.

20. Project Management

The course equips students with the essential skills and knowledge to effectively lead and manage AI-focused projects. This course delves into the core principles of project management, including planning, execution, monitoring, and closing projects, with a special emphasis on methodologies that are critical in the fast-paced AI industry. Students will explore both traditional and agile project management approaches, learning to navigate the complexities of AI projects that involve interdisciplinary teams, emerging technologies, and innovative solutions.

Learning Outcomes

1. Lead cross-functional teams in the execution of AI projects, ensuring effective communication, collaboration, and decision-making throughout the project lifecycle.
2. Optimise project processes and workflows by integrating advanced project management tools and AI-driven insights to enhance productivity and project outcomes.

3. Adapt project management strategies in response to emerging AI trends and challenges, demonstrating flexibility and strategic thinking to achieve project goals.
4. Identify the fundamental principles of project management, including scope, time, cost, and quality management, as applied to AI projects.
5. Explain the methodologies and frameworks such as Agile, Scrum, and Waterfall, and how they can be applied to manage AI-based projects effectively.
6. Analyse the risk factors in AI project management, including ethical considerations, data privacy, and algorithmic bias, and their potential impacts on project outcomes.
7. Develop comprehensive project plans for AI initiatives, including timelines, budgets, resource allocation, and risk management strategies.
8. Evaluate project performance using key performance indicators (KPIs) and project management tools, ensuring alignment with AI project objectives.
9. Implement agile project management practices in real-time scenarios, adapting to changes in project scope, resources, and technological advancements.

21. Linux and Shell Scripting

This course equips learners with the essential skills to navigate and manage Linux servers effectively. The course begins by demystifying the Linux command line interface, learning the fundamentals students need to interact with the system. As learners progress, they will explore powerful command-line utilities like grep, sed, awk, and more, allowing them to manipulate and analyse data efficiently. Next, they will delve into the Linux file system structure, gaining a solid understanding of user permissions and access control. This knowledge is crucial for ensuring secure and organised server environments. Finally, the course empowers students to write their own shell scripts. They will learn how to craft conditional statements (if/else) and loops (for loops) to automate repetitive tasks and streamline the workflow. By the end, they will be able to write scripts to solve real-world problems and significantly boost their Linux server administration skills.

Learning Outcomes

1. Design and implement shell scripts to manage user accounts, automate system administration tasks, and perform file processing operations.
2. Evaluate and optimise shell scripts for efficiency and maintainability, adhering to best practices and security considerations.
3. Integrate shell scripts with other tools for continuous integration and deployment pipelines.
4. Define core Linux concepts like process management, file systems, and kernel functionalities.
5. Explain the functionalities and usage of common Linux shell commands for navigation, file manipulation, and user management.
6. Identify and differentiate between various Linux shell scripting languages and their basic syntax.
7. Navigate the Linux directory structure using command-line tools and manipulate files and directories using shell commands.
8. Troubleshoot basic shell script errors using debugging techniques and analyse script outputs.
9. Write basic shell scripts to automate repetitive tasks, including conditional branching and looping constructs.

22. Operating Systems

This fundamental course aims to equip students with knowledge of core Linux operating system concepts. The module will cover essential operating system functionalities, including process management, process synchronisation (concurrency), memory management techniques, and disk scheduling. In process

management, students will explore how Linux manages processes throughout their lifecycle, including starting, pausing, resuming, and allocating resources. The concept of concurrency will be introduced, covering multithreading and multiprocessing. Students will also learn how asynchronous processing facilitates concurrent execution. Memory management is another key topic. The course will delve into how the operating system manages memory on both RAM and hard disk, along with concepts like virtual memory, memory pages, and page caching.

Learning Outcomes

1. Configure and optimise operating system settings for specific workflows, considering factors like security and scalability.
2. Design and implement solutions for process management and synchronisation challenges in a simulated operating system environment.
3. Evaluate resource utilisation metrics and analyse performance bottlenecks within an operating system.
4. Differentiate between various scheduling algorithms and their impact on system performance.
5. Explain the core functionalities of an operating system, including process management, memory management, and I/O subsystem.
6. Identify and describe common synchronisation mechanisms used to control access to shared resources.
7. Apply scheduling algorithms to analyse their behaviour and predict their impact on specific workloads.
8. Compare and contrast different memory management techniques (paging, segmentation) and their suitability for various applications.
9. Troubleshoot deadlock situations in concurrent programming scenarios using techniques like deadlock detection and avoidance.

23. Databases and Computer Networks

This core foundational course equips students with knowledge of Database Management Systems (DBMS) and Computer Networks. The course starts with Entity-Relationship (ER) diagrams, a visual tool for mapping real-world data storage problems. Students learn to translate ER diagrams into a relational model with tables. SQL, the standard language for relational databases, is then introduced. Students will spend significant time building proficiency in writing optimised and complex SQL queries for various data manipulation tasks. Real-world examples will be used to solidify practical knowledge.

Learning Outcomes

1. Design and implement distributed database architectures for high availability and fault tolerance in varied environments.
2. Integrate network security solutions with different tools and workflows to ensure secure communication and data transfer within applications.
3. Select and configure appropriate database management systems for deployment pipelines based on data requirements and scalability needs.
4. Describe different network security threats and basic security principles in network design.
5. Explain the fundamental concepts of computer networks, including network topologies, protocols, and communication models.
6. Identify and differentiate between various database models and their strengths and weaknesses for different applications.
7. Analyze network traffic using tools like Wireshark and identify potential network performance issues.

8. Configure basic network security measures like firewalls and access control lists (ACLs) to mitigate security risks.
9. Design and implement basic database queries using Structured Query Language (SQL) to retrieve and manipulate data.

24. DevOps Tools Part 1

DevOps Tools Part 1 is a comprehensive course designed for students pursuing a Master of Science in Computer Science with a specialisation in DevOps. This course introduces the essential tools and methodologies that form the backbone of modern DevOps practices. Students will gain a solid foundation in version control with Git, continuous integration/continuous deployment (CI/CD) pipelines using Jenkins, and configuration management with Ansible.

Learning Outcomes

1. Design and implement a basic DevOps workflow for a given application scenario, considering factors like version control, infrastructure management, and automated deployment.
2. Select and integrate appropriate DevOps tools within a workflow based on project requirements and team preferences.
3. Troubleshoot and debug issues within a DevOps pipeline, optimising it for efficiency and reliability.
4. Differentiate between popular DevOps tools in different categories based on their features and use cases.
5. Explain the functionalities of various DevOps tools across different stages of the software development lifecycle (SDLC).
6. Identify and describe the core principles of DevOps and its benefits for software development and delivery.
7. Automate infrastructure provisioning and configuration management using tools like Ansible or Chef.
8. Design and implement basic CI/CD pipelines using tools to automate build, test, and deployment processes.
9. Set up and use a version control system (VCS) for code versioning, branching, and collaborative development.

25. DevOps Tools Part 2

This is an advanced course designed for students building on the foundational knowledge from DevOps Tools Part 1. This course delves deeper into advanced DevOps tools and techniques that drive modern software development and operations. Students will explore sophisticated CI/CD pipelines with tools like GitLab CI/CD and Azure DevOps, and master infrastructure as code (IaC) with Terraform and AWS CloudFormation. The course emphasises practical, hands-on experience, enabling students to automate and manage complex cloud environments effectively. In addition to advanced CI/CD and IaC, students will learn about service mesh architectures with Istio, advanced container orchestration with Kubernetes, and continuous monitoring and observability with tools such as Grafana and Jaeger. The course also covers security practices in DevOps, including integrating security tools into the CI/CD pipeline and managing secrets with HashiCorp Vault. By the end of the course, students will have the expertise to design, implement, and manage scalable, secure, and efficient DevOps workflows, preparing them for leadership roles in the field.

Learning Outcomes

1. Design and deploy containerized applications using Docker and orchestrate them using platforms like Kubernetes for scalability and high availability.
2. Develop IaC scripts using tools like Ansible or Terraform to automate complex infrastructure provisioning and management tasks.
3. Integrate advanced monitoring and logging tools within a DevOps pipeline to provide real-time feedback on application health and performance, enabling proactive troubleshooting and incident management.
4. Describe advanced features of DevOps tools (e.g., Git branching strategies, containerization technologies like Docker, container orchestration platforms like Kubernetes)
5. Explain the concepts of infrastructure as code (IaC) and its role in automating infrastructure provisioning and management.
6. Identify and differentiate between continuous monitoring and logging tools used in varied workflows for performance analysis and troubleshooting.
7. Build and manage Docker containers for software deployment and isolate applications and their dependencies.
8. Implement advanced Git workflows, including branching strategies, merging, and conflict resolution.
9. Utilise continuous monitoring and logging tools to analyse application performance data and identify potential issues in production environments.

26. Amazon Web Services Part 1

This course provides a comprehensive overview of Amazon Web Services (AWS), focusing on core services and best practices for deploying applications on the cloud. Students will learn to leverage AWS's extensive portfolio of services including compute (EC2, Lambda), storage (S3, EBS), databases (RDS, DynamoDB), networking (VPC, CloudFront), and monitoring tools (CloudWatch). The course balances theoretical understanding with practical, hands-on exercises, enabling students to build scalable and reliable applications on AWS. Topics include security best practices, cost optimization, and architectural patterns for building robust cloud solutions.

Learning Outcomes

1. Design and deploy a scalable and secure cloud infrastructure for a simple application on AWS, considering factors like cost optimization and fault tolerance.
2. Integrate AWS services with DevOps tools and workflows for automated provisioning, configuration management, and deployment of applications within the cloud.
3. Configure security groups, network access control lists (ACLs), and other security measures to protect AWS infrastructure from unauthorized access.
4. Describe the core AWS services (EC2, S3, RDS, Lambda) and their key features, use cases, and pricing models.
5. Explain AWS architectural patterns and best practices for designing secure, scalable, and cost-efficient cloud solutions.
6. Identify appropriate AWS services for specific application requirements and workload patterns.
7. Deploy and manage EC2 instances, including instance types, security groups, and elastic IP addresses.
8. Create and manage S3 buckets, configure access controls, and implement data lifecycle policies.
9. Set up and monitor AWS resources using CloudWatch, implementing alarms and notifications for proactive issue detection.

27. Amazon Web Services Part 2

This course builds on foundational AWS knowledge, diving deeper into the platform's sophisticated features and services. Students will explore advanced topics including automated deployment with AWS CodeDeploy, infrastructure automation with AWS CloudFormation and Terraform, and serverless architecture patterns. The course covers microservices architecture on AWS using ECS and Kubernetes, advanced networking with private subnets and bastion hosts, and implementing disaster recovery and high availability solutions. Additionally, students will learn about cost optimization strategies, AWS Organizations for multi-account management, and advanced security practices including IAM policies and AWS KMS.

Learning Outcomes

1. Develop and deploy a CI/CD pipeline using AWS CodePipeline to automate the build, test, and deployment process for complex applications within the cloud.
2. Design and implement a serverless architecture for a specific application use case, considering scalability and cost-efficiency.
3. Configure advanced networking features such as private subnets, NAT gateways, and VPN connections to secure and optimize cloud infrastructure.
4. Describe advanced AWS services and features for specialized workloads (e.g., machine learning with SageMaker, analytics with Redshift).
5. Explain serverless architecture patterns and their benefits in terms of scalability, maintenance, and cost.
6. Identify best practices for disaster recovery, high availability, and fault tolerance in AWS deployments.
7. Automate infrastructure provisioning and updates using AWS CloudFormation or Terraform for Infrastructure as Code (IaC).
8. Implement advanced security practices, including IAM role-based access control, encryption, and secrets management.
9. Optimize AWS costs through reserved instances, auto-scaling, and careful resource management.

28. Advanced Algorithms Part 2

This course is designed to deepen students' understanding of advanced algorithmic techniques and problem-solving strategies. Building upon foundational algorithm knowledge, students will explore complex data structures like graphs, trees, and heaps, along with advanced algorithmic paradigms such as dynamic programming, greedy algorithms, and divide-and-conquer. The course covers graph algorithms including depth-first search (DFS), breadth-first search (BFS), and topological sorting. Students will learn advanced optimization techniques like memoization and tabulation for dynamic programming. Additionally, the course introduces advanced concepts such as Disjoint Set Union (DSU), graph colouring, and shortest path algorithms like Bellman-Ford and Floyd-Warshall. These advanced topics will prepare them for tackling complex algorithmic challenges often encountered in technical interviews.

Learning Outcomes

1. Design and implement custom algorithms utilising advanced concepts to solve complex challenges in automation or performance optimization.
2. Evaluate the suitability of different advanced algorithms for specific use cases, considering factors like scalability, resource constraints, and desired level of accuracy.
3. Integrate advanced algorithms and data structures into real-world applications to address performance bottlenecks and scalability requirements.
4. Differentiate between various algorithmic paradigms and select the most appropriate approach for specific problem scenarios.

5. Explain the mathematical foundations and computational complexity of advanced algorithms.
6. Identify suitable data structures (graphs, trees, heaps) for storing and organizing data in various application contexts.
7. Apply dynamic programming techniques to break down complex problems into manageable subproblems and optimize solutions.
8. Implement graph algorithms (DFS, BFS, shortest path) to solve network and relationship problems.
9. Analyse algorithm performance and optimize code for efficiency in memory usage and execution time.

29. Computational Models

This course provides students with a deep understanding of the mathematical and theoretical foundations of computation. It covers finite automata, Turing machines, and formal grammars. Students explore how these models apply to real-world problems and their significance in advanced AI algorithms.

Learning Outcomes

1. Integrate computational models into complex AI systems to solve real-world challenges.
2. Reflect on ethical implications of computational model choices in AI development.
3. Explain fundamental concepts and principles of computational models.
4. Design computational models for AI applications.
5. Evaluate the performance of algorithms within computational models.
6. Implement simulations of computational models using programming languages.
7. Compare computational models and describe their limitations.
8. Identify types and applications of computational models in AI.

30. Artificial Intelligence Integration Strategies

Focused on equipping students with skills to effectively incorporate AI into business and industrial processes, exploring deployment methodologies, system interoperability, and change management.

Learning Outcomes

1. Design scalable and sustainable AI integration solutions.
2. Demonstrate commitment to ethical AI practices.
3. Develop an AI integration plan for industry applications.
4. Assess performance of integrated AI systems.
5. Implement AI integration using relevant tools and platforms.
6. Describe AI integration approaches and their outcomes.
7. Explain principles of AI system architecture and deployment.
8. Identify key AI integration techniques and applications.

31. Emerging Artificial Intelligence Technologies

Designed to immerse students in latest AI advancements including deep learning, neural networks, NLP, computer vision, and reinforcement learning.

Learning Outcomes

1. Innovate by integrating emerging AI technologies.
2. Critically assess ethical implications of deploying emerging technologies.
3. Develop prototypes using emerging AI technologies.
4. Experiment with emerging AI tools and platforms.
5. Identify current and emerging AI technologies.
6. Understand principles of emerging AI technologies.
7. Analyse impact on various industries.
8. Evaluate effectiveness of emerging AI technologies.

32. Machine Learning Applications

Comprehensive course on practical implementation of ML techniques across various industries including healthcare, finance, and marketing.

Learning Outcomes

1. Design and deploy machine learning solutions for industry problems.
2. Critically assess ethical concerns related to ML.
3. Implement ML algorithms using Python.
4. Evaluate model performance on different datasets.
5. Develop and fine-tune ML models for applications.
6. List and describe ML algorithms and use cases.
7. Explain evaluation metrics and model selection.
8. Analyse impact of feature selection on performance.

33. Artificial Intelligence for Decision Making

Equips students with skills to harness AI for enhanced decision-making using predictive analytics, decision trees, and optimization algorithms.

Learning Outcomes

1. Create comprehensive frameworks integrating AI into decision-making.
2. Assess ethical considerations of AI-driven decisions.
3. Design and implement AI models for decision scenarios.
4. Assess quality and reliability of AI-generated decisions.
5. Apply data visualisation to interpret AI-driven decisions.
6. Identify key AI techniques in decision-making.
7. Explain how AI models analyse data and provide recommendations.
8. Evaluate AI effectiveness across different sectors.

34. Intelligent Data Processing

Equips students to efficiently manage and analyse large datasets using advanced AI techniques including data cleaning, integration, and real-time processing.

Learning Outcomes

1. Design end-to-end data processing pipelines.
2. Assess ethical implications of data processing.

3. Implement intelligent data processing techniques.
4. Assess efficiency and accuracy of approaches.
5. Apply processed datasets to train ML models.
6. List and describe key data processing methods.
7. Explain how algorithms transform data into insights.
8. Evaluate data processing effectiveness.

35. Introduction to Artificial Intelligence

Comprehensive overview of AI concepts, techniques, and applications covering history, theories, algorithms, and practical applications in robotics, NLP, and computer vision.

Learning Outcomes

1. Create simple AI systems addressing real-world challenges.
2. Work effectively in groups on AI solutions.
3. Evaluate societal and ethical challenges of AI.
4. Implement and run AI algorithms on datasets.
5. Assess accuracy and performance of AI models.
6. Identify foundational AI concepts.
7. Explain key AI advancements and milestones.
8. Compare narrow, general, and superintelligent AI.
9. Utilise AI tools and frameworks for development.

36. Data Science Principles

Introduces core concepts and methodologies of data science including data collection, cleaning, statistical analysis, and data visualisation.

Learning Outcomes

1. Create comprehensive data science workflows.
2. Work effectively with diverse team members.
3. Assess ethical implications of data science.
4. Apply data cleaning and preprocessing techniques.
5. Create and evaluate statistical models.
6. Assess accuracy and performance metrics.
7. List and describe data science principles.
8. Explain how data science informs business decisions.
9. Analyse different data types and model selection.

37. Computational Intelligence

This course is aimed at providing students with an in-depth understanding of the techniques and algorithms that enable intelligent behaviour in computational systems, covering neural networks, fuzzy logic, evolutionary computation, and swarm intelligence.

Learning Outcomes

1. Create systems that integrate multiple computational intelligence techniques to address complex, multidisciplinary problems.
2. Collaborate on interdisciplinary projects involving computational intelligence demonstrating effective communication and teamwork.
3. Critically evaluate the ethical implications of computational intelligence in decision-making systems.

38. Algorithmic Thinking

This course cultivates a deep understanding of algorithm design and analysis, focusing on principles of algorithmic problem-solving and teaching students to develop efficient algorithms for a wide range of computational problems.

Learning Outcomes

1. Design innovative algorithms for novel problem domains demonstrating creativity and originality in algorithm design.
2. Integrate algorithmic thinking into interdisciplinary projects showcasing the ability to transfer skills across disciplines.
3. Critically assess the societal impact of algorithmic decisions considering ethical implications, fairness, and potential biases.

39. Intelligent Systems

This course provides students with a comprehensive understanding of how to design and implement systems that exhibit intelligent behaviour, exploring expert systems, autonomous agents, knowledge representation, and reasoning.

Learning Outcomes

1. Design intelligent systems with adaptive learning capabilities demonstrating proficiency in adaptive algorithms and real-time learning.
2. Collaborate on the development of multi-agent systems for complex problem-solving and integrate different intelligent agent types.
3. Evaluate the broader implications of deploying intelligent systems, considering issues such as automation, privacy, and decision-making ethics.

40. Cognitive Computing

This course explores the intersection of artificial intelligence and human cognitive processes, examining how cognitive computing systems mimic human thought processes including perception, reasoning, learning, and problem-solving.

Learning Outcomes

1. Demonstrate the ability to lead and manage the implementation of cognitive computing solutions within real-world business environments, ensuring alignment with organisational goals.
2. Show competency in collaborating with professionals from different fields (e.g., data science, software engineering, business) to develop cognitive computing solutions.

3. Exhibit the ability to innovate and propose novel applications of cognitive computing in industries such as healthcare, finance, and education.

41. Predictive Modelling

This course equips students with skills to develop and apply models that forecast future trends and behaviours based on historical data, covering linear and logistic regression, time series analysis, ensemble methods, and advanced machine learning techniques.

Learning Outcomes

1. Demonstrate the ability to design and implement an end-to-end predictive modelling solution, from data collection and preprocessing to model deployment and monitoring.
2. Exhibit the ability to apply predictive modelling techniques creatively in new or emerging domains, addressing specific challenges with tailored solutions.
3. Display competency in working collaboratively in teams to develop, test, and deploy predictive models, leveraging diverse expertise.

42. Neural Networks and Deep Learning

This course focuses on advanced techniques and architectures used to build sophisticated AI systems, providing an in-depth exploration of neural networks including feedforward networks, CNNs, RNNs, and deep learning models.

Learning Outcomes

1. Demonstrate the ability to design and implement novel neural network architectures tailored to specific challenges, pushing the boundaries of current methodologies.
2. Exhibit competency in adapting existing neural network models to address new or complex problems, demonstrating flexibility and innovative thinking.
3. Display proficiency in integrating neural networks with other AI technologies, such as reinforcement learning or symbolic AI, to create hybrid systems.

43. Applied Data Analytics

This course bridges the gap between data theory and real-world applications, focusing on the end-to-end process of data analytics including data collection, cleaning, exploratory analysis, and visualisation.

Learning Outcomes

1. Demonstrate the ability to integrate data analytics into broader business strategies, ensuring that analytical insights align with organisational goals.
2. Exhibit the ability to design and implement data-driven solutions to solve complex, real-world problems, leveraging advanced analytical techniques.
3. Display competency in leading data analytics projects within multidisciplinary teams, managing the entire analytics lifecycle.

44. Robotics and Automation

This course provides students with a comprehensive understanding of the principles and practices of robotics and automated systems, covering robot kinematics, dynamics, control systems, and sensor integration.

Learning Outcomes

1. Demonstrate the ability to integrate robotic systems with automation processes in an industrial setting, ensuring seamless operation and efficiency.
2. Exhibit the ability to lead teams in developing custom robotic solutions tailored to specific challenges, leveraging interdisciplinary knowledge.
3. Demonstrate the competency to apply ethical considerations and sustainable practices in the design and deployment of robotic systems.

45. Natural Language Processing

This course focuses on techniques and technologies that enable computers to understand, interpret, and generate human language, covering tokenization, part-of-speech tagging, named entity recognition, sentiment analysis, and advanced NLP models.

Learning Outcomes

1. Demonstrate the ability to design and deploy complete NLP applications, such as chatbots or language translation systems, from data collection to model deployment.
2. Display ability in integrating NLP techniques into business analytics tools to enhance decision-making processes.
3. Demonstrate the ability to apply ethical considerations when developing NLP systems, ensuring fairness, transparency, and accountability.

46. Ethical Artificial Intelligence Practices

This course explores the ethical, legal, and social implications of artificial intelligence technologies, examining bias in AI algorithms, data privacy, transparency, accountability, and the impact of AI on employment and society.

Learning Outcomes

1. Demonstrate the ability to design AI solutions that prioritise ethical considerations, balancing innovation with responsibility to ensure positive societal impact.
2. Lead and guide multidisciplinary teams in developing and implementing AI systems that adhere to ethical standards, fostering a culture of responsible AI development.
3. Demonstrate the competency to advocate for ethical AI practices in industry and policy discussions, effectively communicating the importance of responsible AI to diverse audiences.

47. Advanced Artificial Intelligence Concepts

This course deepens students' understanding of cutting-edge AI topics, delving into advanced methodologies such as generative adversarial networks (GANs), meta-learning, and advanced reinforcement learning techniques.

Learning Outcomes

1. Demonstrate the ability to integrate advanced AI techniques into existing software systems, ensuring compatibility, scalability, and performance optimisation.
2. Lead and manage innovative AI research projects that explore cutting-edge AI concepts, contributing to the academic and industrial advancement of the field.
3. Demonstrate the competency to adapt advanced AI technologies to address new and unforeseen challenges in various domains.

48. Artificial Intelligence in Industry Applications

This course bridges theoretical AI concepts and their real-world applications across industries, exploring how AI technologies are implemented to solve industry-specific challenges in healthcare, finance, manufacturing, and transportation.

Learning Outcomes

1. Demonstrate the ability to design AI-driven strategies that can transform industry practices, addressing current limitations and leveraging AI for competitive advantage.
2. Lead cross-functional teams in the development and deployment of AI projects within an industry, ensuring collaboration and alignment with business objectives.
3. Demonstrate the ability to adapt existing AI solutions to meet emerging needs and challenges within an industry, ensuring continued relevance and impact.

49. Training Large Language Models

This course focuses on the principles and techniques involved in training large language models (LLMs), providing an in-depth understanding of architectures and training methodologies used to develop powerful language models like GPT-3 and BERT.

Learning Outcomes

1. Develop strategies for reducing biases in LLM outputs, integrating ethical considerations into the model training and deployment process.
2. Collaborate with cross-functional teams to deploy LLMs in production environments, ensuring scalability and efficiency while maintaining performance standards.
3. Lead the creation of documentation and training resources for stakeholders to effectively understand and use LLMs, tailored to various technical backgrounds.

50. Prompt Engineering

This course is dedicated to mastering the art and science of designing effective prompts for large language models, exploring the principles of crafting precise and impactful prompts that elicit desired responses from advanced AI models.

Learning Outcomes

1. Develop and integrate prompt engineering practices into AI development workflows, ensuring that prompts are aligned with overall project goals and technical requirements.

2. Work collaboratively in a team to optimise prompts for multi-faceted AI projects, leveraging collective insights to improve model performance.
3. Demonstrate adaptability by staying updated with the latest trends and research in prompt engineering, applying new techniques and methodologies to improve outcomes.

51. Artificial Intelligence Product Management

This course provides a comprehensive exploration into the strategic and operational aspects of managing artificial intelligence products from inception to market, covering product lifecycle management, market analysis, and the integration of AI capabilities into product roadmaps.

Learning Outcomes

1. Demonstrate the ability to lead cross-functional teams in the development and deployment of AI products, ensuring alignment between technical capabilities and business objectives.
2. Critically assess the performance of AI products post-launch, using key performance indicators and user feedback to drive iterative improvements.
3. Formulate strategies for scaling AI products in global markets, considering cultural, economic, and regulatory differences across regions.

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.

Grading Scheme

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			

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.