Outcomes and content for Year 12

Secure software architecture

Outcomes

A student:

- justifies methods used to plan, develop and engineer software solutions **SE-12-01**
- applies structural elements to develop programming code **SE-12-02**
- analyses how current hardware, software and emerging technologies influence the development of software engineering solutions **SE-12-03**
- evaluates practices to safely and securely collect, use and store data **SE-12-04**
- explains the social, ethical and legal implications of software engineering on the individual, society and the environment **SE-12-05**
- justifies the selection and use of tools and resources to design, develop, manage and evaluate software **SE-12-06**
- designs, develops and implements safe and secure programming solutions **SE-12-07**
- tests and evaluates language structures to refine code **SE-12-08**

Content

Designing software

- Describe the benefits of developing secure software

  Including:
  - data protection
  - minimising cyber attacks and vulnerabilities

- Interpret and apply fundamental software development steps to develop secure code

  Including:
  - requirements definition
  - determining specifications
  - design
  - development
  - integration
  - testing and debugging
  - installation
  - maintenance

- Describe how the capabilities and experience of end users influence the secure design features of software

Developing secure code

- Explore fundamental software design security concepts when developing programming code

  Including:
  - confidentiality
  - integrity
  - availability
  - authentication
  - authorisation
  - accountability
- Apply security features incorporated into software including data protection, security, privacy and regulatory compliance
- Use and explain the contribution of cryptography and sandboxing to the ‘security by design’ approach in the development of software solutions
- Use and explain the ‘privacy by design’ approach in the development of software solutions

  Including:
  - proactive not reactive approach
  - embed privacy into design
  - respect for user privacy
- Test and evaluate the security and resilience of software by determining vulnerabilities, hardening systems, handling breaches, maintaining business continuity and conducting disaster recovery
- Apply and evaluate strategies used by software developers to manage the security of programming code

  Including:
  - code review
  - static application security testing (SAST)
  - dynamic application security testing (DAST)
  - vulnerability assessment
  - penetration testing
- Design, develop and implement code using defensive data input handling practices, including input validation, sanitisation and error handling
- Design, develop and implement a safe application programming interface (API) to minimise software vulnerabilities
- Design, develop and implement code considering efficient execution for the user

  Including:
  - memory management
  - session management
  - exception management
- Design, develop and implement secure code to minimise vulnerabilities in user action controls

  Including:
  - broken authentication and session management
  - cross-site scripting (XSS) and cross-site request forgery (CSRF)
  - invalid forwarding and redirecting
  - race conditions
- Design, develop and implement secure code to protect user file and hardware vulnerabilities from file attacks and side channel attacks

Impact of safe and secure software development

- Apply and describe the benefits of collaboration to develop safe and secure software

  Including:
  - considering various points of view
  - delegating tasks based on expertise
  - quality of the solution

- Investigate and explain the benefits to an enterprise of the implementation of safe and secure development practices

  Including:
  - improved products or services
  - influence on future software development
  - improved work practices
  - productivity
  - business interactivity

- Evaluate the social, ethical and legal issues and ramifications that affect people and enterprises resulting from the development and implementation of safe and secure software

  Including:
  - employment
  - data security
  - privacy
  - copyright
  - intellectual property
  - digital disruption

Programming for the web

Outcomes

A student:

- justifies methods used to plan, develop and engineer software solutions **SE-12-01**
- applies structural elements to develop programming code **SE-12-02**
- analyses how current hardware, software and emerging technologies influence the development of software engineering solutions **SE-12-03**
- evaluates practices to safely and securely collect, use and store data **SE-12-04**
- explains the social, ethical and legal implications of software engineering on the individual, society and the environment **SE-12-05**
- justifies the selection and use of tools and resources to design, develop, manage and evaluate software **SE-12-06**
- designs, develops and implements safe and secure programming solutions **SE-12-07**
- tests and evaluates language structures to refine code **SE-12-08**
- applies methods to manage and document the development of a software project **SE-12-09**

Content

Data transmission using the web

- Explore the applications of web programming

  Including:
  - interactive website/webpages
  - e-commerce
  - progressive web apps (PWAs)

- Investigate and practise how data is transferred on the internet

  Including:
  - data packets
  - internet protocol (IP) addresses, including IPv4
  - domain name systems (DNS)

- Investigate and describe the function of web protocols and their ports

  Including:
  - HTTP, HTTPS
  - TCP/IP
  - DNS
  - FTP, SFTP
  - SSL, TLS
  - SMTP, POP 3, IMAP

- Explain the processes for securing the web

  Including:
  - Secure Sockets Layer (SSL) certificates
  - encryption algorithms
  - encryption keys
  - plain text and cipher text
  - authentication and authorisation
  - hash values
  - digital signatures

- Investigate the effect of big data on web architecture

  Including:
  - data mining
  - metadata
  - streaming service management

Designing web applications

- Investigate and explain the role of the World Wide Web Consortium (W3C) in the development of applications for the web

  Including:
  - Web Accessibility Initiative (WAI)
  - internationalisation
  - web security
  - privacy
  - machine-readable data

- Model elements that form a web development system

  Including:
  - client-side (front-end) web programming
  - server-side (back-end) web programming
  - interfacing with databases that are based on Structured Query Language (SQL) or non-SQL

- Explore and explain the influence of a web browser on web development, including the use of developer (dev) tools
- Investigate cascading style sheets (CSS) and its impact on the design of a web application

  Including:
  - consistency of appearance
  - flexibility with browsers or display devices
  - CSS maintenance tools

- Investigate the reasons for version control and apply it when developing web application
- Explore the types and significance of code libraries for front-end web development

  Including:
  - frameworks that control complex web applications
  - template engines
  - predesigned CSS classes

- Explain the use and development of open-source software in relation to web development
- Investigate methods to support and manage the load times of web pages/applications

- Research, experiment with and evaluate the prevalence and use of web content management systems (CMS)
- Assess the contribution of back-end web development to the success of a web application
- Observe and describe the back-end process used to manage a web request

  Including:
  - role of webserver software
  - web framework
  - objects
  - libraries
  - databases
- Develop a web application using an appropriate scripting language with shell scripts to make files and directories, and searching for text in a text file
- Apply a web-based database and construct script that executes SQL

  Including:
  - selecting fields
  - incorporating ‘group by’
  - common SQL queries
  - constraints using WHERE keyword
  - table joins
- Compare Object-Relational Mapping (ORM) to SQL
- Describe how collaborative work practices between front-end and back-end developers improve the development of a web solution
- Design, develop and implement a progressive web app (PWA)

  Including:
  - the application of design and user interface (UI) and user experience (UX) principles of font, colour, audio, video and navigation
  - a UI that considers accessibility and inclusivity

Software automation

Outcomes

A student:

- justifies methods used to plan, develop and engineer software solutions **SE-12-01**
- applies structural elements to develop programming code **SE-12-02**
- analyses how current hardware, software and emerging technologies influence the development of software engineering solutions **SE-12-03**
- evaluates practices to safely and securely collect, use and store data **SE-12-04**
- explains the social, ethical and legal implications of software engineering on the individual, society and the environment **SE-12-05**
- justifies the selection and use of tools and resources to design, develop, manage and evaluate software **SE-12-06**
- designs, develops and implements safe and secure programming solutions **SE-12-07**
- tests and evaluates language structures to refine code **SE-12-08**
- applies methods to manage and document the development of a software project **SE-12-09**

Content

Algorithms in machine learning

- Investigate how machine learning (ML) supports automation through the use of DevOps, robotic process automation (RPA) and business process automation (BPA)
- Distinguish between artificial intelligence (AI) and ML
- Explore models of training ML

  Including:
  - supervised learning
  - unsupervised learning
  - semi-supervised learning
  - reinforcement learning

- Investigate common applications of key ML algorithms

  Including:
  - data analysis and forecasting
  - virtual personal assistants
  - image recognition

- Research models used by software engineers to design and analyse ML

  Including:
  - decision trees
  - neural networks

- Describe types of algorithms associated with ML

  Including:
  - linear regression
  - logistic regression
  - K-nearest neighbour

Programming for automation

- Design, develop and apply ML regression models using an OOP to predict numeric values

  Including:
  - linear regression
  - polynomial regression
  - logistic regression
- Apply neural network models using an OOP to make predictions

Significance and impact of ML and AI

- Assess the impact of automation on the individual, society and the environment

  Including:
  - safety of workers
  - people with disability
  - the nature and skills required for employment
  - production efficiency, waste and the environment
  - the economy and distribution of wealth
- Explore by implementation how patterns in human behaviour influence ML and AI software development

  Including:
  - psychological responses
  - patterns related to acute stress response
  - cultural protocols
  - belief systems
- Investigate the effect of human and dataset source bias in the development of ML and AI solutions

Software engineering project

Outcomes

A student:

- justifies methods used to plan, develop and engineer software solutions **SE-12-01**
- applies structural elements to develop programming code **SE-12-02**
- analyses how current hardware, software and emerging technologies influence the development of software engineering solutions **SE-12-03**
- evaluates practices to safely and securely collect, use and store data **SE-12-04**
- explains the social, ethical and legal implications of software engineering on the individual, society and the environment **SE-12-05**
- justifies the selection and use of tools and resources to design, develop, manage and evaluate software **SE-12-06**
- designs, develops and implements safe and secure programming solutions **SE-12-07**
- tests and evaluates language structures to refine code **SE-12-08**
- applies methods to manage and document the development of a software project **SE-12-09**

Content

Identifying and defining

- Define and analyse the requirements of a problem

  Including:
  - demonstrating need(s) or opportunities
  - assessing scheduling and financial feasibility
  - generating requirements including functionality and performance
  - defining data structures and data types
  - defining boundaries

- Explore tools used to develop ideas and generate solutions

  Including:
  - brainstorming, mind-mapping and storyboards
  - data dictionaries, including selecting appropriate data types
  - algorithm design
  - code generation
  - testing and debugging
  - installation
  - maintenance

- Investigate types of software implementation methods

  Including:
  - direct
  - phased
  - parallel
  - pilot

Research and planning

- Research and use the Waterfall software development approach

  Including:
  - logical progression of steps used throughout the life cycle
  - stages of ‘falling water’
  - advantages and disadvantages
  - scale and types of developments

- Research and use the Agile software development approach

  Including:
  - rate of developing a final solution
  - method tailoring
  - iteration workflow
  - scale and types of developments

- Research the WAgile software development approach

  Including:
  - understanding it is a hybrid model
  - analysis of the ‘when’ and ‘how’ intervention is applied during the development life cycle
  - scale and types of developments

- Apply project management to plan and conduct the development and implementation of a project and software engineering solution

  Including:
  - scheduling and tracking using a software tool, including Gantt charts
  - using collaboration tools

- Explore social and ethical issues associated with project work, including working individually, collaboratively and responding to stakeholders
- Explore communication issues associated with project work

  Including:
  - involving and empowering the client
  - enabling feedback
  - negotiating

- Investigate how software engineering solutions are quality assured

  Including:
  - defining criteria on which quality will be judged
  - ensuring requirements are met using a continual checking process
  - addressing compliance and legislative requirements

- Demonstrate the use of modelling tools

- Explain the contribution of back-end engineering to the success and ease of software development

  Including:
  - technology used
  - error handling
  - interfacing with front end
  - security engineering

Producing and implementing

- Design, construct and implement a solution to a software problem using appropriate development approach(es)
- Present a software engineering solution using presentation software
- Develop, construct and document algorithms
- Allocate resources to support the development of a software engineering solution
- Demonstrate the use of programmed data backup
- Implement version control when developing a software engineering solution
- Explore strategies to respond to difficulties when developing a software engineering solution

  Including:
  - looking for a solution online
  - collaboration with peers
  - outsourcing
- Propose an additional innovative solution using a prototype and user interface (UI) design

Testing and evaluating

- Apply methodologies to test and evaluate code
- Use a language-dependent code optimisation technique
- Analyse and respond to feedback
- Evaluate the effectiveness of a software engineering solution

  Including:
  - developing a report to synthesise feedback
  - developing a test plan
  - testing data used/generated based on path and boundary testing
  - comparing actual output with expected output