Government of **Western Australia**
School Curriculum and Standards Authority

COMPUTER SCIENCE

ATAR course

Year 12 syllabus

Acknowledgement of Country

Kaya. The School Curriculum and Standards Authority (the Authority) acknowledges that our offices are on Whadjuk Noongar boodjar and that we deliver our services on the country of many traditional custodians and language groups throughout Western Australia. The Authority acknowledges the traditional custodians throughout Western Australia and their continuing connection to land, waters and community. We offer our respect to Elders past and present.

Important information

This syllabus is effective from 1 January 2026.

Users of this syllabus are responsible for checking its currency.

Syllabuses are formally reviewed by the School Curriculum and Standards Authority (the Authority) on a cyclical basis, typically every five years.

Disclaimer

Any resources such as texts, websites and so on that may be referred to in this document are provided as examples of resources that teachers can use to support their learning programs. Their inclusion does not imply that they are mandatory or that they are the only resources relevant to the course.

Contents

Rationale **1**

Aims **2**

Organisation **3**

Structure of the syllabus 3

Organisation of content 4

Representation of the general capabilities 6

Representation of the cross-curriculum priorities 9

Unit 3 – Design and development of programming and networking solutions **10**

Unit description 10

Unit content 10

Unit 4 – Design and development of database solutions and cyber security considerations **17**

Unit description 17

Unit content 17

Assessment **22**

School-based assessment 23

Assessment table – Year 12 24

Reporting 26

ATAR course examination **27**

Examination design brief – Year 12 28

Appendix 1 – Grade descriptions Year 12 ..... **29**

Rationale

The Computer Science ATAR course builds on the core principles, concepts and skills developed in the Digital Technologies subject in previous years. Students utilise and enhance established analysis and algorithm design skills to create innovative digital solutions to real-world problems. In the process, students develop computational, algorithmic and systems thinking skills which can be successfully applied to problems across domains outside Information Technology. In addition to the development of software, the essential concepts of networking, data management and cyber security are explored. With the vast amounts of data collected in our increasingly digital world, especially in the information-intensive business and scientific disciplines, data management is becoming essential. Similarly, with more and more devices connecting to the internet, cyber security is a major issue for society and the world continues to look for new, young experts to emerge in this field.

Ethical considerations, security requirements and legal factors affect society as a whole and their influence and impact on the development of digital solutions are explored.

This course provides students with options in a range of post-school pathways. The course has been designed to meet the expectations of tertiary institutions and students will be well prepared for further study in university and TAFE courses. It provides a sound understanding of computer science to support students pursuing further studies and employment in other areas, including Science, Technology, Engineering, Mathematics and Business, all of which are underpinned and driven by advances in Computer Science.

Aims

The Computer Science ATAR course aims to develop students':

- skills in
  - designing, maintaining, adapting and producing relational databases and digital solutions
  - solving problems through the use of algorithms, data structures and programming languages
  - assessing cybersecurity issues within a digital environment to apply appropriate responses
  - understanding of the design, application and interactions of data and software in digital systems through the creation and maintenance of relational databases, network data transmission and programming constructs
- understanding of how to apply a technology process accurately to develop a digital solution
- understanding of the interrelationships between the development and use of digital solutions, for individuals and societies in relation to the legal and ethical implications of software design, data management and cyber threats.

Organisation

This course is organised into a Year 11 syllabus and a Year 12 syllabus. The cognitive complexity of the syllabus content increases from Year 11 to Year 12.

Structure of the syllabus

The Year 12 syllabus is divided into two units which are delivered as a pair. The notional time for the pair of units is 110 class contact hours.

Each unit includes:

- a unit description – a short description of the purpose of the unit
- unit content – the content to be taught and learned.

Technology process

A major focus of the course is the creation of systems and digital solutions to specific problems. In creating solutions, it is expected that students will use a structured development process to guide their approach. This development process is iterative in nature and involves four phases – investigating the problem, developing ideas and designing a solution, developing a solution and evaluating the effectiveness of the solution.

Investigate: students will define the problem. They will deconstruct the problem and develop a set of criteria that can be used to design and evaluate a solution to the problem.

Design: students will use the criteria they developed in their investigation to design a suitable digital solution to the problem they have defined.

Develop: students will use their programming and database skills to develop a suitable digital solution that will meet the criteria they have established to solve the problem.

Evaluate: students will evaluate the effectiveness of their solution in solving the problem that they defined during their analysis. This will allow students to refine their ideas and make partial or minor changes to their solution to improve the user experience and/or the technical quality of their solution.

Unit 3 – Design and development of programming and networking solutions

In this unit, students gain knowledge and skills to create software solutions. They use algorithms, structured programming and object-oriented techniques to design and implement software solutions for a range of problems. They consider the complex interactions between users, developers, the law, ethics and society when computer systems are used and developed. Students learn about network communications and the transfer of data through a network.

Unit 4 – Design and development of database solutions and cyber security considerations

In this unit, students learn about the design concepts and tools used to develop relational database systems. Students gain knowledge and skills to create database solutions and create queries to extract relevant information. Students consider the security of network communications, exploring a range of threats and measures used to keep networks secure. Students examine attitudes and values involved in the creation and use of computer-based systems, and their effect on society. They examine the ethical and legal obligations of the user and developer in the collection and storage of data.

Organisation of content

The unit content includes both theoretical aspects (Knowledge) and practical aspects (Skills).

The course is divided into four content areas.

Unit 3 is divided into two content areas:

- Programming
- Network communications.

Unit 4 is divided into two content areas:

- Cyber security
- Data management.

Programming

Students develop solutions to specific problems using both pseudocode and a programming language. They examine the use of various simple and complex data types with the basic constructs of sequence, selection and iteration. Complex problems are analysed and broken down into small, self-contained units for which students create functions with parameter passing. In this unit, students consider good programming practices and the suitability of an algorithm, including comparing common sort and search algorithms. The application of object-oriented programming to develop working software solutions is studied.

Network communications

Students explore the communication models and protocols underpinning the transfer of data in local networks and the internet. They explore the purpose of layers of a network and the components that operate within them, taking into account factors that affect the design of the network. They investigate design and creation of secure and efficient networks.

Cyber security

Students explore the security of network communications by looking at internal and external threats to a network and potential solutions to those threats. They examine methods of keeping network communications secure over an open connection, such as the internet. The legal requirements and ethical responsibilities of developers, organisations and penetration testers are considered with respect to the management of data and networks.

Data management

Students examine the organisation of data into separate entities using a relational database and the process of normalisation. They explore various methods of representing and organising data and develop a working database solution using SQL. They consider legal and ethical responsibilities with regards to the collection and storage of data.

Programming languages

Python is the prescribed programming language for the Computer Science ATAR course and will be used in ATAR examination questions related to programming. In school based assessments, schools may choose to also use an object-oriented programming language of their choice for the:

- development of a purpose-designed software solution
- design, creation, modification, testing, evaluation and documentation of programs
- writing, interpretation, testing and debugging of code
- use and development of a user interface
- use of control structures, including sequence, selection and iteration
- construction and use of data structures, including arrays and dictionaries
- design and implement data validation techniques
- application of structured programming methods using modularisation.

There is no requirement within this course to create a graphical user interface.

Database management systems

There is no prescribed database management system for the Computer Science ATAR course, but SQLite is recommended. If an alternative to SQLite is chosen, it must meet the assessment requirements for this syllabus that students are required to use a database management system (DBMS) that enables the:

- development of a purpose-designed database solution
- design, creation, modification, testing and evaluation of a database solution
- creation of tables and queries.

Note: the course does not require the development of a graphical user interface for the database system, although some method of allowing the user to interact with the database is required.

The database management systems should provide the student with opportunity to:

- create a working relational database
- construct simple queries using SQL within one or two tables
- construct queries across multiple tables using a database tool.

Representation of the general capabilities

The general capabilities encompass the knowledge, skills, behaviours and dispositions that will assist students to live and work successfully in the twenty-first century. Teachers may find opportunities to incorporate the capabilities into the teaching and learning program for Computer Science. The general capabilities are not assessed unless they are identified within the specified unit content.

Literacy

Students become literate as they develop the knowledge, skills and dispositions to interpret and use language confidently for learning and communicating in and out of school and for participating effectively in society. Literacy involves students listening to, reading, viewing, speaking, writing and creating oral, print, visual and digital texts, and using and modifying language for different purposes in a range of contexts.

In the Computer Science ATAR course, students develop literacy capability as they learn how to communicate ideas, concepts and detailed proposals to a variety of audiences; recognise how language can be used to manipulate meaning; and read and interpret detailed written instructions. They learn to understand and use language to discuss and communicate information, concepts and ideas related to the course.

By gaining literacy in the metalanguage of computer science, students understand that language varies according to context and they increase their ability to use language flexibly. Computer science vocabulary is often technical and includes specific terms for concepts, processes and production. Students learn to understand that much technological information is presented in the form of drawings, diagrams, flow charts, models, tables and graphs. They also learn the importance of listening and talking when learning about technologies processes, especially in articulating, questioning and evaluating ideas.

Numeracy

Students become numerate as they develop the knowledge and skills to use mathematics confidently across other learning areas at school and in their lives more broadly. Numeracy involves students in recognising and understanding the role of mathematics in the world, and having the dispositions and capacities to use mathematical knowledge and skills purposefully.

In the Computer Science ATAR course, students work with the concepts of number, geometry, scale and proportion. They use models, create accurate technical drawings, work with digital models and use computational thinking in decision-making processes when designing and creating best-fit solutions.

Information and communication technology capability

Students develop information and communication technology (ICT) capability as they learn to use ICT effectively and appropriately to access, create and communicate information and ideas, solve problems and work collaboratively, and in their lives beyond school. The capability involves students in learning to make the most of the digital technologies available to them. They adapt to new ways of doing things as technologies evolve, and limit the risks to themselves and others in a digital environment.

In the Computer Science ATAR course, students create solutions that consider social and environmental factors when operating digital systems with digital information. They develop an understanding of the characteristics of data, digital systems, audiences, procedures and computational thinking. They apply this understanding when they investigate, communicate and create purpose-designed digital solutions. Students learn to formulate problems, logically organise and analyse data and represent it in abstract forms. They automate solutions through algorithmic logic. Students decide the best combinations of data, procedures and human and physical resources to generate efficient and effective digital solutions.

Critical and creative thinking

Students develop capability in critical and creative thinking as they learn to generate and evaluate knowledge, clarify concepts and ideas, seek possibilities, consider alternatives and solve problems. Critical and creative thinking is integral to activities that require students to think broadly and deeply using skills, behaviours and dispositions, such as reason, logic, resourcefulness, imagination and innovation in all learning areas at school and in their lives beyond school.

In the Computer Science ATAR course, students develop capability in critical and creative thinking as they imagine, generate, develop, produce and critically evaluate ideas. They develop reasoning and the capacity for abstraction through challenging problems that do not have straightforward solutions. Students analyse problems, refine concepts and reflect on the decision-making process by engaging in systems, design and computational thinking. They identify, explore and clarify technologies, information and use that knowledge in a range of situations. Students think critically and creatively. They consider how data and information systems impact on our lives and how these elements might be better designed and managed.

Personal and social capability

Students develop personal and social capability as they learn to understand themselves and others, and manage their relationships, lives, work and learning more effectively. The capability involves students in a range of practices, including recognising and regulating emotions, developing empathy for others and understanding relationships, establishing and building positive relationships, making responsible decisions, working effectively in teams, handling challenging situations constructively and developing leadership skills.

In the Computer Science ATAR course, students develop personal and social capability as they engage in project management and development in a collaborative workspace. They direct their own learning, plan and carry out investigations, and become independent learners who can apply design thinking, technologies understanding and skills when making decisions. Students develop social and employability skills through working cooperatively in teams, sharing resources, tools, equipment and processes, making group decisions, resolving conflict and showing leadership. Designing and innovating involve a degree of risk-taking and as students work with the uncertainty of sharing new ideas they develop resilience.

The Computer Science ATAR course enhances students' personal and social capability by developing their social awareness. Students develop understanding of diversity by researching and identifying user needs. They develop social responsibility through the understanding of empathy with and respect for others.

Ethical understanding

Students develop ethical understanding as they identify and investigate concepts, values, character traits and principles, and understand how reasoning can help ethical judgement. Ethical understanding involves students in building a strong personal and socially oriented, ethical outlook that helps them to manage context, conflict and uncertainty, and to develop an awareness of the influence that their values and behaviour have on others.

In the Computer Science ATAR course, students develop the capacity to understand and apply ethical and socially responsible principles when collaborating with others and creating, sharing and using technologies, data, processes, tools and equipment. In this course, students consider their own roles and responsibilities as discerning citizens, and learn to detect bias and inaccuracies. Understanding the protection of data, intellectual property and individual privacy in the school environment helps students to be ethical digital citizens.

Intercultural understanding

Students develop intercultural understanding as they learn to value their own cultures, languages and beliefs, and those of others. They come to understand how personal, group and national identities are shaped, and the variable and changing nature of culture. The capability involves students in learning about and engaging with diverse cultures in ways that recognise commonalities and differences, create connections with others and cultivate mutual respect.

In the Computer Science ATAR course, students consider how technologies are used in diverse communities at local, national, regional and global levels, including their impact and potential to transform people's lives. They explore ways in which past and present practices enable people to use technologies to interact with one another across cultural boundaries.

Representation of the cross-curriculum priorities

The cross-curriculum priorities address contemporary issues which students face in a globalised world. Teachers may find opportunities to incorporate the priorities into the teaching and learning program for the Computer Science ATAR course. The cross-curriculum priorities are not assessed unless they are identified within the specified unit content.

Aboriginal and Torres Strait Islander histories and cultures

The Computer Science ATAR course may provide opportunities for students to explore creative, engaging and diverse learning contexts for students to value and appreciate the contribution by the world's oldest continuous living cultures to past, present and emerging technologies.

Asia and Australia's engagement with Asia

The Computer Science ATAR course may provide opportunities for students to explore contemporary and emerging technological achievements that the Asia and Pacific regions have made, and continue to make, to global technological advances, including: innovation in hardware and software design and development; the regions' role in outsourcing of information and communications technology (ICT) services; and globalisation. Students could also consider the contribution of Australia's contemporary and emerging technological achievements to the Asia and Pacific Regions.

Sustainability

The Computer Science ATAR course may provide opportunities for students, within authentic contexts, to choose and evaluate digital technologies and information systems with regard to risks and opportunities they present. They may also evaluate the extent to which digital solutions can embrace and promote sustainable practices.

Unit 3 – Design and development of programming and networking solutions

Unit description

In this unit, students gain knowledge and skills to create software solutions. They use algorithms and structured programming and object-oriented techniques to design and implement software solutions for a range of problems. They consider the complex interactions between users, developers, the law, ethics and society when computer systems are used and developed. Students learn about network communications and the transfer of data through a network.

A major focus of the course is the creation of systems and digital solutions to specific problems. In creating solutions, it is expected that students will use a structured development process to guide their approach. This development process is iterative in nature and involved four phases – investigating the problem, developing ideas and designing a solution, developing a solution and evaluating the effectiveness of the solution.

Unit content

An understanding of the Year 11 content is assumed knowledge for students in Year 12. It is recommended that students studying Unit 3 and Unit 4 have completed Unit 1 and Unit 2.

This unit includes the knowledge, understandings and skills described below. This is the examinable content.

The unit content includes theoretical aspects (Knowledge) and practical aspects (Skills) and these are organised into two content areas:

- Programming
- Network communications.

Programming

Programming skills and concepts

Knowledge

- program control structures, including:
  - sequence
  - selection
  - iteration
    - fixed
    - post-test
    - pre-test
- characteristics of data types used in solutions, including:
  - integer
  - float
  - string
  - Boolean
- modular coding using functions, parameters and arguments

    - scope of variables (Global, Local)
    - parameter passing (value and reference)
- characteristics of the following data structures:
    - arrays
        - one-dimensional arrays
        - two-dimensional arrays
- dictionaries

Skills

- apply, using pseudocode and/or the Python programming language, program control structures in solutions
- apply, using pseudocode and/or the Python programming language, data types used in solutions as variables
- apply, using pseudocode and/or the Python programming language, modular coding using functions, parameters and arguments
- use different types of operators
    - arithmetic operators (+, -, *, /, %)
    - relational operators (==, !=, >, =, <=)
    - logical operators (AND, OR, NOT)
- reading and writing of complex logical expressions, including:
    - Boolean operators (AND, OR, NOT)
    - logical order of precedence
- apply, using pseudocode, the following data structures:
    - arrays
        - one-dimensional arrays
        - two-dimensional arrays
- dictionaries

Good programming practice

Knowledge

- Framework for development
    - investigate
        - problem description
        - define requirements
        - development schedule (including Gantt charts)
    - design
        - design data structures
        - design and test algorithm
    - develop
        - develop and debug code
        - unit testing
    - evaluate
        - user acceptance testing
        - developer retrospective

- compare common development processes
  - linear
  - iterative
- good programming practice, including:
  - validate input before processing
  - a clear and uncluttered mainline
  - one logical task per subroutine
  - use of stubs
  - appropriate use of control structures and data structures
  - writing for subsequent maintenance
  - version control
  - regular backup
  - recognition of relevant social and ethical issues
  - exception handling
  - functions that are able to return a single data structure or value

Skills

- apply common development processes
- apply good programming practice when developing a software solution, including:
  - validate input before processing
  - a clear and uncluttered mainline
  - one logical task per subroutine
  - use of stubs
  - appropriate use of control structures and data structures
  - writing for subsequent maintenance
  - version control
  - regular backup
  - recognition of relevant social and ethical issues
  - exception handling
  - functions return a single data structure or value

Structured algorithms

Knowledge

- effective structure of code
  - use of a modular approach, including functions and classes
  - use of stubs to represent incomplete modules
  - parameter passing
- common algorithms
  - purpose of Big O notation
  - identify Big O notation for search and sort algorithms
  - search algorithms
    - linear search
    - binary search

- sort algorithms
  - bubble sort
  - insertion sort
  - selection sort

Skills

- use pseudocode and/or Python programming language for representing algorithms
- create code with effective structure
  - use of a modular approach, including functions and classes
  - parameter passing
  - use of stubs to represent incomplete functions

Testing

Knowledge

- acceptance testing of functional requirements based on user needs
- the use of live test data to ensure that testing accurately reflects the expected environment in which the new system will operate
  - large file sizes
  - mix of transaction types
  - response times
  - volume of data (load testing)
- validation of the solution with the design specifications
- generating relevant test data for complex solutions
- comparison of actual with expected output
- unit tests to ensure code performs correctly

Skills

- comparison of the solution with the design specifications
- generating relevant test data for complex solutions
- comparison of actual with expected output

Error detection and debugging code

Knowledge

- the process of detecting and correcting errors, including:
  - types of error
    - syntax errors
    - logic errors
    - runtime errors, including:
      - division by zero
      - index out of range
- methods of error detection and correction
  - methodical approach to the isolation of logic errors
  - use of debugging techniques
    - breakpoints
    - print statements
  - desk checking (trace tables)
  - comparison of actual with expected output

Skills

- detecting and correcting errors, including:
  - types of error
  - syntax errors
  - logic errors
  - runtime errors, including:
    - division by zero
    - index out of range
- methodical approach to the isolation of errors
  - use of debugging techniques
  - desk checking (trace tables)
  - comparison of actual with expected output

Object-oriented programming

Knowledge

- characteristics of the following object-oriented concepts:
  - classes
  - objects
  - attributes
  - methods
  - abstraction and polymorphism
  - instantiation
  - inheritance
  - encapsulation

Skills

- apply, using pseudocode and/or the Python programming language, object-oriented programming concepts, including:
  - classes
  - objects
  - attributes
  - methods
  - abstraction
  - instantiation
  - inheritance
- use an object-oriented programming language to:
  - create classes and objects with attributes and methods
  - instantiate objects
  - use inheritance to extend classes
  - use variables and control structures within objects

Ethical and legal implications for developers of software

Knowledge

- impacts of software in society, including:
  - computer malware, such as viruses
  - reliance on software

  - social networking
  - cyber safety
  - large volumes of information (which may be unsupported, unverifiable, misleading or incorrect) available through the internet
- rights and responsibilities of software developers
  - acknowledging the intellectual property of others
  - recognition by others of the developer's intellectual property
  - producing quality software solutions
  - appropriately responding to user-identified problems
  - adhering to code of conduct
  - neither generating nor transmitting malware
  - addressing ergonomic issues in software design
  - ensuring software addresses inclusivity issues
  - ensuring individuals' privacy is not compromised

Network communications

Models of networking

Knowledge

- comparison between the Open Systems Interconnection (OSI) Model and the Department of Defence (DoD) transmission control protocol/internet protocol (TCP/IP) model
- OSI model:
  - purpose of OSI model
  - layers of OSI model
  - role of layers within the model
  - MAC address layer 2 switching
  - IP address layer 3 routing
- DoD TCP/IP model:
  - purpose of DoD model
  - layers of DoD model
    - application
    - transport
    - internet
    - network
  - role of layers within the model
- key protocols and devices associated with layers in models
  - user datagram protocol (UDP)
  - difference between UDP and TCP
  - compare packet architecture between TCP and UDP
- features of TCP/IP
- IPv4 vs IPv6
- IP Addressing
- Private vs public IP addressing
- subnetting
  - subnet masks
  - Classless Inter-Domain Routing (CIDR) notation

- default gateways
- domain name system (DNS)
- ports
- packet architecture
  - data
  - segment
  - packet
  - frame
  - bits

Skills

- identify, design and apply IP addressing scheme
- identify and design subnets, applying subnet masks and CIDR notation

Network components

Knowledge

- role of components at different network layers
  - transmission media
  - modem
  - router
  - gateway
  - switch
  - wireless access point
  - firewalls

Network performance

Knowledge

- network design topology for security and performance
- bandwidth
  - mapping networks using diagrams including intermediary and end devices
  - subnetting and broadcast domains (segmentation)
- tools for network performance management and troubleshooting
  - ping
  - traceroute

Skills

- use ping and traceroute to troubleshoot and evaluate network performance
- interpret and create network diagrams using specified CISCO conventions to represent network topologies, considering addressing, subnets, segmentation, security and performance

Unit 4 – Design and development of database solutions and cyber security considerations

Unit description

In this unit, students learn about the design concepts and tools used to develop relational database systems. Students gain skills to create database solutions and create queries to extract relevant information. Students consider the security of network communications, considering a range of threats and measures used to keep networks secure. Students examine attitudes and values that lead to the creation and use of computer-based systems and their effect on society. They examine the ethical and legal obligations of the user and developer in the collection and storage of data.

This unit focuses on the creation of database systems. Students are expected to follow the technology process in order to produce quality products. The process includes four steps; investigate, design, produce and evaluate. This process is essential for the creation of solutions in the Computer Science course.

Unit content

This unit builds on the content covered in Unit 3.

This unit includes the knowledge, understandings and skills described below. This is the examinable content.

The unit content includes theoretical aspects (Knowledge) and practical aspects (Skills) and these are organised into two content areas:

- Cyber security
- Data management.

Cyber security

Ethics and Law

Knowledge

- the *Privacy Act 1988* as it relates to data security Australian Privacy Principle 11 (APP11)
- the *Privacy Amendment (Notifiable Data Breaches) Act 2017*

Skills

- apply theoretical ethics and law knowledge using supplied case studies

Network threats

Knowledge

- external network threats
  - social engineering (phishing)
  - denial of service, including distributed denial of service
  - back door
  - IP spoofing
  - SQL Injection

    - man-in-the-middle
    - cross-site scripting
    - types of malware
        - viruses
        - worms
        - trojan horses
        - spyware
        - adware
        - ransomware
    - physical network threats
    - zero day vulnerabilities
- internal network threats
    - lost or stolen devices
    - compromised credentials
    - misuse by employees
- security solutions to network threats
    - analysis of log files
    - anti-malware
    - firewall filtering
    - access control lists
    - intrusion prevention systems
    - virtual private networks
    - user training
    - ICT code of conduct
    - physical security
- appropriate solutions to different external network threats

Skills

- identify network security threats and suggest solutions for a given stimulus (mitigation strategies)

Security frameworks

Knowledge

- the CIA triad model of security analysis
    - confidentiality
    - integrity
    - availability
- the AAA framework for securing systems
    - authentication
    - authorisation
    - accounting

Skills

- use the CIA triad to analyse security threats and incidents
- use the AAA framework for security analysis and auditing

Cryptography

Knowledge

- symmetric encryption
- asymmetric encryption (public/private keys)
  - certificate purpose and use
- use of asymmetric encryption to
  - prevent unauthorised access to data
  - authenticate data being sent across network
- secure communication over the internet
  - use of asymmetric encryption to establish connection and symmetric encryption to exchange data
- common methods of encryption
  - early methods and weaknesses
  - current best practice

Ethical hacking

Knowledge

- role of ethical hacking in improving network security
- penetration testing: role of blue team vs red team

Data management

Core concepts

Knowledge

- organisation of a relational database:
  - entities
  - attributes
  - relationships
    - one-to-one
    - one-to-many
    - many-to-many
- tables as the implementation of entities, consisting of fields and records
- data types
  - integer
  - float
  - Boolean
  - text
  - date
- primary and foreign keys to link tables
- composite key
- data anomalies
  - insert
  - update
  - delete
- how databases interact with other systems and link to the programming content

  - open database connectivity
- role of ACID in database transactions – atomicity, consistency, isolation, durability

Data modelling

Knowledge

- purpose of database documentation for the developers
  - data dictionary
  - entity relationship (ER) diagrams using crow's foot notation

Skills

- analyse ER diagrams written in crow's foot notation (up to 10 tables)
- create accurate ER diagrams using crow's foot notation (three to eight tables)
- represent databases using relational notation
- create data dictionaries
- resolve many to many (M:N) relationships

Data integrity

Knowledge

- data integrity
  - referential integrity
  - domain integrity
  - entity integrity
- factors influencing quality of data, including:
  - currency
  - authenticity
  - relevance
  - accuracy
  - outliers (cleaning)

Normalisation

Knowledge

- purpose of normalising data to third normal form (3NF)
  - rules of 1NF
  - rules of 2NF
  - rules of 3NF
- know the process to normalise data to 3NF

Skills

- apply the process to normalise data to 3NF using relational notation

Database creation and manipulation

Knowledge

- know techniques to retrieve required information through querying data sets using SQL
- purpose of cleaning and manipulating data sets to import required data into a database

Skills

- use a relational database management system (RDBMS) to create and manipulate a relational database with a minimum of five tables
- use SQL to create, modify and query a database including:
  - CREATE and MODIFY tables, including:
    - use of constraints to ensure validity of data
    - enforcing referential integrity
  - DROP
  - SELECT
  - INSERT
  - UPDATE
  - DELETE, including cascading deletes
  - aggregate functions (COUNT, SUM, AVG, MAX, MIN)
  - ORDER BY
  - inner join across multiple tables (up to 4 tables)
  - GROUP BY (with aggregate functions)
  - calculated fields
  - concatenated fields
  - use of aliases with calculated and concatenated fields
- clean and manipulate data sets to import required data into a database

Development issues

Knowledge

- ethical issues
  - collecting data about individuals
  - privacy concerns
  - appropriate use of data
  - reliability of data sources
  - acknowledgement of data sources
  - use of data mining
- security issues
  - keeping personal data private
  - backups of organisational data
  - restricting access to data
  - ownership and control of data
- legal issues
  - Australian privacy law in relation to database development
  - Australian Privacy Principles (APP5, APP8, APP10, APP11, APP12)

Assessment

Assessment is an integral part of teaching and learning that at the senior secondary years:

- provides evidence of student achievement
- identifies opportunities for further learning
- connects to the standards described for the course
- contributes to the recognition of student achievement.

Assessment for learning (formative) and assessment of learning (summative) enable teachers to gather evidence to support students and make judgements about student achievement. These are not necessarily discrete approaches and may be used individually or together, and formally or informally.

Formative assessment involves a range of informal and formal assessment procedures used by teachers during the learning process in order to improve student achievement and to guide teaching and learning activities. It often involves qualitative feedback (rather than scores) for both students and teachers, which focuses on the details of specific knowledge and skills that are being learnt.

Summative assessment involves assessment procedures that aim to determine students' learning at a particular time, for example when reporting against the standards, after completion of a unit/s. These assessments should be limited in number and made clear to students through the assessment outline.

Appropriate assessment of student work in this course is underpinned by reference to the set of pre-determined course standards. These standards describe the level of achievement required to achieve each grade, from A to E. Teachers use these standards to determine how well a student has demonstrated their learning.

Where relevant, higher order cognitive skills (e.g. application, analysis, evaluation and synthesis) and the general capabilities should be included in the assessment of student achievement in this course. All assessment should be consistent with the requirements identified in the course assessment table.

Assessment should not generate workload and/or stress that, under fair and reasonable circumstances, would unduly diminish the performance of students.

School-based assessment

The *Western Australian Certificate of Education (WACE) Manual* contains essential information on principles, policies and procedures for school-based assessment that must be read in conjunction with this syllabus.

School-based assessment involves teachers gathering, describing and quantifying information about student achievement.

Teachers design school-based assessment tasks to meet the needs of students. As outlined in the *WACE Manual*, school-based assessment of student achievement in this course must be based on the Principles of Assessment:

- Assessment is an integral part of teaching and learning
- Assessment should be educative
- Assessment should be fair
- Assessment should be designed to meet its specific purpose/s
- Assessment should lead to informative reporting
- Assessment should lead to school-wide evaluation processes
- Assessment should provide significant data for improvement of teaching practices.

Summative assessments in this course must:

- be no more than 8 tasks in total
- allow for the assessment of each assessment type at least once over the year/pair of units
- have a minimum value of 5 per cent weighting of the total school assessment mark
- provide a representative sampling of the syllabus content.

Assessment tasks not administered under test or controlled conditions require appropriate authentication processes.

Assessment table – Year 12

Type of assessment	Weighting
Project The student is required to develop a database and/or software solution by using a structured development process. Students use the skills and theory covered in this course to create working solutions based on provided stimulus materials. Software solutions should make use of object-oriented programming techniques. Database solutions should make use of SQL to create and manipulate a relational database system. Stimulus material can include: diagrams; extracts from newspapers and/or journal articles; data dictionaries; structure charts; trace tables; algorithms and/or algorithm segments (in pseudocode); and/or screen captures or representations of databases and programs. Diagrams can include: entity relationship diagrams, computer system diagrams, and/or network diagrams. The student is required to research ideas; implement a database and/or software system using a relational database management system and/or programming language; explore, develop and evaluate solutions; and manage processes throughout production to produce solutions. Projects should be open-ended in nature to give students the opportunity to apply a structured development process. The project should require students to investigate and deconstruct a problem, design a suitable solution to the problem, develop a solution based on their design and evaluate the effectiveness of their final product. Teachers are encouraged to interrelate topics such as programming and databases where possible to create authentic problems requiring a solution.	40%
Theory test Tests typically consist of a combination of questions requiring short and extended answers. Short answer questions can be a mix of closed and open items that can be scaffolded or sectionalised. The student can be required to: explain concepts, apply knowledge, analyse and/or interpret data and/or refer to stimulus material. Stimulus material can include: diagrams; extracts from newspaper and/or journal articles; data dictionaries; structure charts; trace tables; algorithms and/or algorithm segments (in pseudocode); and/or screen captures or representations of databases and programs. Diagrams can include: entity relationship diagrams and/or network diagrams. Extended answer questions can be a mix of closed and open items that can be scaffolded or sectionalised, typically with an increasing level of complexity. The student can be required to apply knowledge and/or critical thinking skills; analyse and/or interpret data, extended algorithms, relational databases, tables and/or diagrams; devise labelled diagrams, and/or solutions (or parts of solutions). Some questions can require the student to refer to stimulus material. Stimulus material can include: diagrams; extracts from newspaper and/or journal articles; data dictionaries; structure charts; trace tables; algorithms and/or algorithm segments (in pseudocode); and/or screen captures or representations of databases and programs. Diagrams can include: entity relationship diagrams and/or network diagrams.	10%

Type of assessment	Weighting
Practical test Tests typically consist of a set of questions developing and implementing solutions requiring the use of a programming language and/or a relational database management system to meet specified requirements. Programming skills assessed could include: writing code; and/or compiling, testing and/or debugging program code, and/or use object-oriented programming principles to create classes and objects with attributes and methods. Database skills assessed could include: creating and implementing schema, joining tables, applying aggregate functions, ordering and grouping using structured queried language.	10%
Examination Typically conducted at the end of each semester and/or unit and reflecting the examination design brief for this syllabus.	40%

Teachers must use the assessment table to develop an assessment outline for the pair of units.

The assessment outline must:

- include a set of assessment tasks
- include a general description of each task
- indicate the unit content to be assessed
- indicate a weighting for each task and each assessment type
- include the approximate timing of each task (for example, the week the task is conducted, or the issue and submission dates for an extended task).

Note: the Project could be a combined task based on both units.

The set of assessment tasks must provide a representative sampling of the content for Unit 3 and Unit 4.

Assessment tasks not administered under test/controlled conditions require appropriate validation/authentication processes.

Reporting

Schools report student achievement, underpinned by a set of pre-determined standards in terms of the following grades:

Grade	Interpretation
A	Excellent achievement
B	High achievement
C	Satisfactory achievement
D	Limited achievement
E	Very low achievement

The grade descriptions for the Computer Science ATAR Year 12 syllabus are provided in Appendix 1. They are used to support the allocation of a grade. They can also be accessed, together with annotated work samples, on the course page of the Authority website www.scsa.wa.edu.au.

To be assigned a grade, a student must have had the opportunity to complete the education program, including the assessment program (unless the school accepts that there are exceptional and justifiable circumstances).

Refer to the *WACE Manual* for further information about the use of a ranked list in the process of assigning grades.

The grade is determined by reference to the standard, not allocated on the basis of a pre-determined range of marks (cut-offs).

ATAR course examination

All students enrolled in the Computer Science ATAR Year 12 course are required to sit the ATAR course examination. The examination is based on a representative sampling of the content for Unit 3 and Unit 4. Details of the ATAR course examination are prescribed in the examination design brief on the following page.

Refer to the *WACE Manual* for further information.

Examination design brief – Year 12

Time allowed

Reading time before commencing work: ten minutes
Working time for paper: three hours

Permissible items

Standard items: pens (blue/black preferred), pencils (including coloured), sharpener, correction fluid/tape, eraser, ruler, highlighters

Special items: up to three calculators, which do not have the capacity to create or store programmes or text, are permitted in this ATAR course examination, Mathomat and/or Mathaid and/or any system flowchart template

Provided by the supervisor

A source booklet containing stimulus material

Section	Supporting information
Section One Short answer 40% of the total examination 20–30 questions Suggested working time: 70 minutes	Questions can be a mix of closed and open items that can be scaffolded or sectionalised. The candidate can be required to: explain concepts, apply knowledge, analyse and/or interpret data. Some questions can require the candidate to refer to stimulus material. Stimulus material can include: diagrams; extracts from newspaper and/or journal articles; data dictionaries; trace tables; algorithms and/or algorithm segments (in pseudocode); and/or screen captures or representations of databases and programs. Diagrams can include: entity relationship diagrams and/or network diagrams.
Section Two Extended answer 60% of the total examination 4–6 questions Suggested working time: 110 minutes	Questions can be a mix of closed and open items that can be scaffolded or sectionalised, typically with an increasing level of complexity. The candidate can be required to: explain concepts; apply critical thinking skills; analyse and/or interpret complex data, extended algorithms, relational databases, tables and/or diagrams; and/or solutions (or parts of solutions); and/or refer to a case study. Some questions can require the candidate to refer to stimulus materials. Stimulus material can include: diagrams; extracts from newspaper and/or journal articles; data dictionaries; trace tables; algorithms and/or algorithm segments (in pseudocode); and/or screen captures or representations of databases and programs. Diagrams can include: entity relationship diagrams and/or network diagrams.

Appendix 1 – Grade descriptions Year 12

A	**Knowledge and understanding** Accurately uses computer science terminology to describe processes and concepts in context and with justification.
	Cyber security and networking Analyses different network threats, justifying appropriate and alternative security solutions where relevant. Successfully designs complex networks including IP address and subnets whilst accurately applying correct conventions to a diagram. Justifies network design decisions and demonstrates an accurate consideration of network performance and security issues, presenting alternate solutions where relevant. Consistently and accurately interprets and analyses network diagrams.
	Data management skills Consistently creates entity relationship diagrams accurately reflecting system requirements, and accurately representing relationships, cardinality, attributes, keys and diagrammatic conventions. Consistently and accurately applies normalisation to the third normal form to model a complex database solution. Consistently writes SQL queries to create functional multi-table relational databases, accurately reflecting relevant system requirements, consisting of appropriate relationships, keys and data types and using constraints to ensure data validity. Consistently and accurately uses SQL to create complex, functional queries across multiple tables, including the use of calculated fields, concatenated fields and aggregation.
	Programming skills Consistently applies and justifies suitable methodology accurately to assist the software development process. Consistently designs and creates accurate and efficient pseudocode with appropriate use of standards and conventions. Consistently and accurately implements programming control structures in pseudocode and a programming language to develop efficient solutions that reflect software requirements. Consistently and accurately uses a range of appropriate data structures, including two-dimensional arrays and dictionaries, to develop effective software solutions. Consistently and accurately uses effective object-oriented and modular programming techniques, including parameter passing and creating classes with inheritance, abstraction and encapsulation, to develop working software solutions that meet requirements. Uses a broad range of techniques to effectively test and debug software solutions using appropriate test data. Consistently and accurately implements data validation and desk checking techniques. Consistently and accurately explains ethical and legal issues and justifies the relevance to software development.

B	**Knowledge and understanding** Accurately uses computer science terminology to describe processes and concepts in context.
	Cyber security and networking Analyses different network threats and explains appropriate and alternative security solutions where relevant. Designs complex networks including IP addresses and subnets whilst applying correct conventions to a diagram. Explains network design decisions and demonstrates a consideration of network performance and security issues, and identifies alternate solutions where relevant. Consistently interprets and analyses network diagrams.
	Data management skills Creates entity relationship diagrams reflecting system requirements, representing relationships, cardinality, attributes, keys and diagrammatic conventions. Applies normalisation to the third normal form to accurately model a complex database solution. Uses SQL to create functional multi-table relational databases reflecting relevant system requirements and consisting of appropriate relationships, keys and data types. Uses SQL to write queries across multiple tables to extract, insert and update relevant data with data validation. Uses calculated fields concatenated fields and aggregation to extract meaningful data from multiple tables.
	Programming skills Consistently applies and explains suitable methodology to assist the software development process. Consistently designs and creates pseudocode with appropriate use of standards and conventions. Implements programming control structures in pseudocode and a programming language to develop solutions that reflect software requirements. Consistently uses a range of data structures, including two-dimensional arrays and dictionaries, to develop software solutions. Uses modular programming techniques, including parameter passing, to develop working software solutions that meet requirements. Uses object-oriented techniques, including creating classes with inconsistent use of inheritance, abstraction and encapsulation, to develop working software solutions that meet requirements. Uses a range of techniques to effectively test and debug software solutions using appropriate test data. Consistently implements data validation and desk checking techniques. Consistently explains ethical and legal issues related to software development.

C	**Knowledge and understanding** Uses computer science terminology, and describes processes and concepts.
	Cyber security and networking Describes different network threats and describes appropriate security solutions. Designs network diagrams using conventions and showing connections between devices. Describes network design decisions with limited consideration for network performance and security issues. Demonstrates limited interpretation and analysis of network diagrams.
	Data management skills Creates entity relationship diagrams reflecting system requirements, representing relationships, cardinality, attributes, keys and/or diagrammatic conventions with some errors. Applies normalisation to the third normal form to model a simple database solution. Uses SQL to write simple queries to extract, insert and update relevant data, attempting to use aggregate functions and extract data from multiple tables. Creates simple queries across multiple tables, with inconsistent use of calculated and concatenated fields.
	Programming skills Applies and describes suitable methodology for software development. Designs and creates pseudocode with appropriate use of standards and conventions. Implements programming control structures in pseudocode and programming languages, which may include minor errors in logic and conventions, and partially reflects software requirements. Uses a range of data types and structures, such as two-dimensional arrays, to develop software solutions. Uses modular programming techniques, including creating functions with parameter passing, and makes use of pre-existing classes to develop a working software solution. Demonstrates some use of good programming practices. Is able to test software design and solution, including selection of appropriate test data to identify and fix errors. Describes ethical and legal issues related to software development.

D	**Knowledge and understanding** Attempts to use computer science terminology to describe processes and concepts.
	Cyber security and networking Incorrectly lists different network threats, but identifies appropriate security solutions. Designs network diagrams using conventions and showing connections between devices, but with errors. Creates incomplete network design decisions with a lack of consideration for network performance and security issues. Attempts to interpret, but incorrectly analyses network diagrams in a limited fashion.
	Data management skills Creates incomplete entity relationship diagrams, partially reflecting system requirements, including main entities and attributes, but with inaccurate key fields and relationships. Attempts to apply normalisation, incompletely modelling a simple database solution. Creates databases with limited functionality reflecting some aspects of system requirements. Unsuccessfully creates queries across multiple tables.
	Programming skills Inconsistently applies and describes suitable methodology to software development. Creates incomplete or inaccurate designs and creates pseudocode with appropriate use of standards and conventions. Inconsistently implements programming control structures in pseudocode and programming languages, which may include minor errors in logic and conventions and partially reflects software requirements, including sort and search algorithms. Inconsistently uses a range of data types and structures such as multi-dimensional arrays to develop software solutions. Uses incorrect or incomplete object-oriented and modular programming techniques, including parameter passing, to develop a working software solution. Identifies, although does not fix, errors. Demonstrates a limited understanding of ethical and legal issues related to software development.

E	Does not meet the requirements of a D grade and/or has completed insufficient assessment tasks to be assigned a higher grade.