NCCA
An Chomhairle Náisiúnta Curaclaim agus Measúnachta
National Council for Curriculum and Assessment

Draft Specification for Leaving Certificate Computer Science

For consultation

Contents

Senior cycle

Senior cycle aims to educate the whole person and contribute to human flourishing. Students' experiences throughout senior cycle enrich their intellectual, social and personal development and their overall health and wellbeing. Senior cycle has 8 guiding principles.

Senior Cycle Guiding Principles	
Wellbeing and relationships	Choice and flexibility
Inclusive education and diversity	Continuity and transitions
Challenge, engagement and creativity	Participation and citizenship
Learning to learn, learning for life	Learning environments and partnerships

These principles are a touchstone for schools and other educational settings, as they design their senior cycle. Senior cycle consists of an optional Transition Year, followed by a two-year course of subjects and modules. Building on junior cycle, learning happens in schools, communities, educational settings, and other sites, where students' increasing independence is recognised. Relationships with teachers are established on a more mature footing and students take more responsibility for their learning.

Senior cycle provides a curriculum which challenges students to aim for the highest level of educational achievement, commensurate with their individual aptitudes and abilities. During senior cycle, students have opportunities to grapple with social, environmental, economic, and technological challenges and to deepen their understanding of human rights, social justice, equity, diversity and sustainability. Students are supported to make informed choices as they choose different pathways through senior cycle and every student has opportunities to experience the joy and satisfaction of reaching significant milestones in their education. Senior cycle should establish firm foundations for students to transition to further, adult and higher education, apprenticeships, traineeships and employment, and participate meaningfully in society, the economy and adult life.

The educational experience in senior cycle should be inclusive of every student, respond to their learning strengths and needs, and celebrate, value, and respect diversity. Students vary in their family and cultural backgrounds, languages, age, ethnic status, beliefs, gender, and sexual identity as well as their strengths, needs, interests, aptitudes and prior knowledge, skills, values and dispositions. Every student's identity should be celebrated, respected, and responded to throughout their time in senior cycle.

At a practical level, senior cycle is supported by enhanced professional development; the involvement of teachers, students, parents, school leaders and other stakeholders; resources; research; clear communication; policy coherence; and a shared vision of what senior cycle seeks to achieve for our young people as they prepare to embark on their adult lives. It is brought to life in schools and other educational settings through:

- effective curriculum planning, development, organisation, reflection and evaluation
- teaching and learning approaches that motivate students and enable them to improve
- a school culture that respects students and promotes a love of learning.

Rationale

Computer science is the foundation of all computing technologies. It involves the study of computing and the design and development of computer systems. Computing technologies have become highly integrated into most aspects of modern life from enhancing patient and medical care and changing how we are entertained, to driving advancements in digital arts and computational sciences. These technologies enable us to communicate instantly across the globe and find new and exciting ways to assist human enterprise. Systems are being designed and developed that are becoming more intelligent, adaptive and autonomous, presenting both benefits and challenges for society.

In this context, the study of computer science is relevant to lives of all students. The Leaving Certificate Computer Science specification is designed to be inclusive, and to accommodate varying levels of previous student experiences. It is a student-centred course that encourages creativity, self-expression, and embraces a human-centred approach to design and development that prioritises the needs of the users. Computational thinking is one of the most fundamental aspects of computer science, through which students learn abstraction, decomposition, pattern recognition, algorithmic thinking and logical reasoning. They learn how to use computing technologies to solve problems and automate processes, creating their own computer programs and artefacts in areas relevant to their own interests and lives.

Students come to understand how computer science impacts the world around us, gaining insights into how algorithms work while also creating their own. The study of computer science deepens the students' awareness of the ethical and social role of computers in society, supporting them to become informed users and creators of technologies. Through Applied Learning Tasks (ALTs), students can choose their own areas of investigation, and in the process develop project management skills and collaborative problem-solving strategies.

Students studying this subject learn to think and create in ways that are valuable and beneficial to them well beyond the computer science classroom.

Aims

Leaving Certificate Computer Science aims to empower and develop students as creators and users of computing technologies. It aims to nurture a life-long engagement with developments in computer science, with students becoming more informed about current and emerging computing technologies.

More specifically, Leaving Certificate Computer Science aims to empower students to:

- develop computational thinking skills to solve problems, and to design and evaluate solutions using computing technologies
- put the principles and concepts of computer science into practice while developing the necessary key competencies
- design and build human-centred computing technologies, independently and collaboratively, in ways that are creative and responsible
- be more critically aware of the ethical, social and environmental impacts of computing technologies on their personal lives and on society.

Continuity and progression

Leaving Certificate Computer Science builds on the learning from early childhood education through to the junior cycle curriculum. When students learn to think computationally they become better able to conceptualise, understand and use computer-based technology, and so are better prepared for today's world and the future.

Junior Cycle

Many of the Statements of Learning at junior cycle relate to Leaving Certificate Computer Science, especially those statements focused on problem solving, creating, communication, and understanding the role and contribution of technology in society. The skills developed during junior cycle are further enhanced in Leaving Certificate Computer Science through opportunities where students can consistently work and learn with others, stimulate their creativity through digital technology, evaluate solutions to issues that are meaningful to their lives, develop resilience and manage their own learning.

Beyond Senior Cycle

Leaving Certificate Computer Science supports students in their understanding of current computer technologies and prepares them for emerging technologies. The learning from this subject is becoming more essential and beneficial to the future pathways of almost all students. It also prepares students for a range of careers directly related to computer science from software engineering to web development. Computer science nurtures a broad range of transferable and trans-disciplinary competences such as problem solving, independent and self-regulated learning, human-centred creative design and collaborative problem-solving. These skills, combined with a technical proficiency and an understanding of Artificial Intelligence (AI), can equip students to embrace the opportunities and challenges ahead, and encourage them to participate meaningfully in society. Exploring the benefits and drawbacks of computing technologies, and their ever-increasing impact on people and societies, develops students as ethical users and creators of technology.

Student learning in senior cycle

Student learning in senior cycle consists of everything students learn **within** all of the subjects and modules they engage with **and** everything students learn which spans and overlaps **across** all of their senior cycle experiences. The overarching goal is for each student to emerge from senior cycle more enriched, more engaged and more competent as a human being than they were when they commenced senior cycle.

For clarity, the learning which spans **across** all of their senior cycle experiences is outlined under the heading ‘key competencies’. The learning which occurs **within** a specific subject or module is outlined under the heading ‘strands and learning outcomes’. However, it is vital to recognise that key competencies and subject or module learning are developed in an integrated way. By design, key competencies are integrated across the rationale, aims, learning outcomes and assessment sections of specifications. In practice, key competencies are developed by students in schools via the pedagogies teachers use and the environment they develop in their classrooms and within their school. Subjects can help students to develop their key competencies; and key competencies can enhance and enable deeper subject learning. When this integration occurs, students stand to benefit:

- during and throughout their senior cycle
- as they transition to diverse futures in further, adult and higher education, apprenticeships, traineeships and employment, and

- in their adult lives as they establish and sustain relationships with a wide range of people in their lives and participate meaningfully in society.

When teachers and students make links between the teaching methods students are experiencing, the competencies they are developing and the ways in which these competencies can deepen their subject specific learning, students become more aware of the myriad ways in which their experiences across senior cycle are contributing towards their holistic development as human beings.

Key competencies

Key competencies is an umbrella term which refers to the knowledge, skills, values and dispositions students develop in an integrated way during senior cycle.

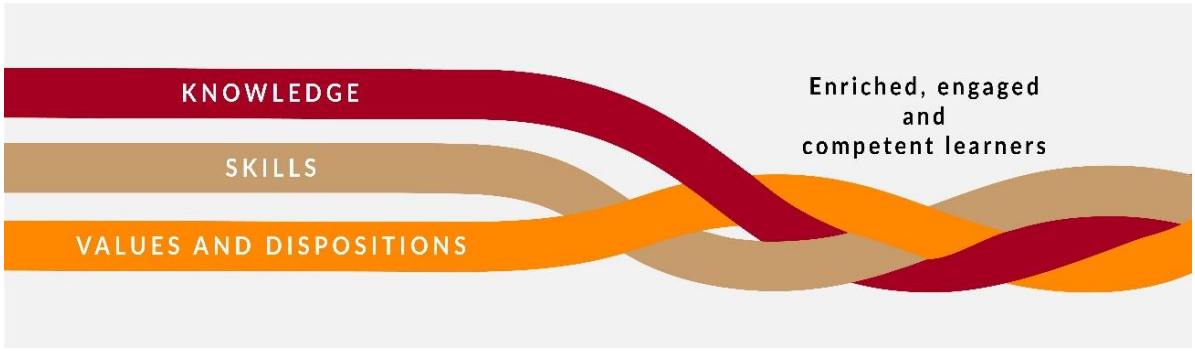

Figure 1 The components of key competencies and their desired impact

The knowledge which is specific to this subject is outlined below under 'strands of study and learning outcomes'. The epistemic knowledge which spans across subjects and modules is incorporated into the key competencies.

Figure 2 Key Competencies in Senior Cycle, supported by literacies and numeracy

These competencies are linked and can be combined; can improve students' overall learning; can help students and teachers to make meaningful connections between and across different areas of learning; and are important across the curriculum.

The development of students' literacies and numeracy contributes to the development of competencies and vice-versa. Key competencies are supported when students' literacies and numeracy are well developed and they can make good use of various tools, including technologies, to support their learning.

The key competencies come to life through the learning experiences and pedagogies teachers choose and through students' responses to them. Students can and should be

helped to develop their key competencies irrespective of their past or present background, circumstances or experiences and should have many opportunities to make their key competencies visible. Further detail in relation to key competencies is available at https://ncca.ie/en/senior-cycle/senior-cycle-redevelopment/student-key-competencies/.

In Leaving Certificate Computer Science, **thinking and solving problems** is supported when students are encouraged to seek challenges, make informed decisions, identify problems and evaluate computational solutions to issues that are relevant and meaningful to their lives. The student agency embedded in the structure of the ALTs enables students to apply computational thinking strategies in ways that range from automation of simple everyday tasks to how they might address complex societal issues and manage uncertainty. Students learn and apply the practices and principles of computer science, and are encouraged to work in ethical and responsible ways. The development of literacies relevant to the tasks and numeracy supports the development of key competencies and vice-versa, and improves the effectiveness of students in applying thinking strategies to their chosen tasks.

Leaving Certificate Computer Science is a collaborative discipline which can readily transfer into other senior cycle subjects and future careers. The specification is designed to provide students with practical opportunities to learn how to work with others while developing project management skills. Through **working with others** on design and development tasks, students can learn to resolve disagreements, celebrate diversity and manage themselves and the emotions of working with others towards a collective goal.

Students learn how to recognise patterns and use abstraction techniques, decompose and solve problems, think algorithmically, manage data, and evaluate digital artefacts. Programming, and the development of their own computing technologies, provide motivation for students and in the process they learn the importance of feedback, and how to respond to feedback, and how to persevere and to be more resilient. The design and construction of their own computing technologies requires an open-mindedness, a sense of playfulness and an ability to incorporate multiple possibilities, perspectives and solutions.

The iterative design process encourages students to see mistakes as feedback where the needs of users can be met in an adaptable and flexible manner. Computer science has vast applications, and **managing learning and self** can be further developed through appropriate open-ended task. Students can improve how they manage their learning, respond to uncertainties in outcomes and build connections to other subjects and future careers. This

versatility of the subject can in turn enhance student **literacies and numeracy**, and strengthen student's digital, data and social media literacies.

The collaborative learning approach requires effective communication and healthy group dynamics. The ALTs, including the additional assessment component (AAC), can often involve students taking account of different perspectives and responding to the feedback from the target user or audience. For example, students design user interfaces, in response to user needs, that are clear and easy to use, and they learn by design how to scaffold their programs with explanatory comments. Designing a computational artefact involves students consistently **communicating** their design process and explaining how their artefact functions.

Leaving Certificate Computer Science is designed to incorporate student agency. This approach, particularly through the ALTs, aims to support students in achieving successful outcomes individually and collaboratively. Students can become more confident in their own abilities, develop internal and external standards and improve self-efficacy through the computational thinking strategies they learn. The opportunity to work on tasks of their own choosing, within a collaborative classroom, nurtures enjoyment and empowers risk-taking which in turn supports students in **being creative** and innovative.

Strands of study and learning outcomes

There are three strands in the Computer Science specification: Practices and principles, Core concepts and Computer science in practice. All three strands are interwoven and are designed to be studied concurrently at different stages of the course. As shown in Figure 3, the strands are not intended to be studied in a linear order. Learning in strands 1 and 2 is applied and developed through collaborative ALTs outlined in strand 3. In that way, the ALTs provide further practical context. Student engagement with the ALTs should increase in complexity and sophistication, thus developing and deepening the learning from strands 1 and 2.

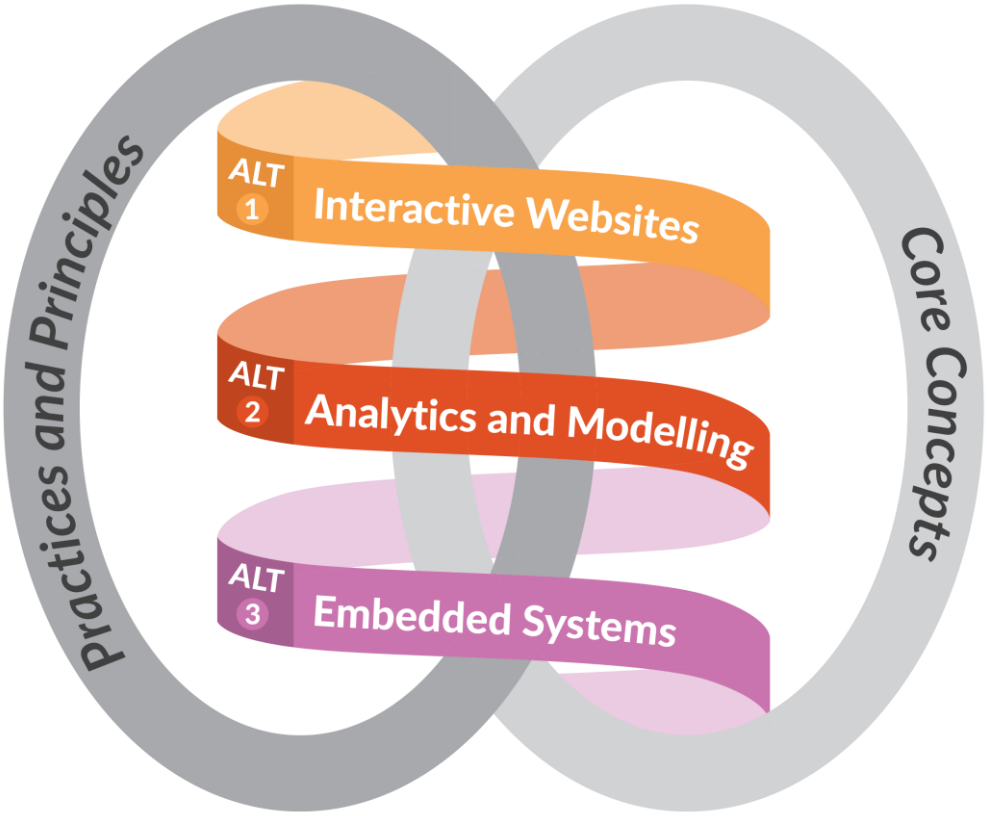

Figure 3 : Structure of Leaving Certificate Computer Science

Strand 1: Practices and principles

The overarching practices and principles of computer science are the behaviours and ways of thinking that computer scientists use. This strand underpins the specification and is fundamental to all learning activities. By becoming familiar with, and fluent in, the practices and principles that underpin good practice, students develop their ability to manage themselves and their learning across the subject.

Strand 2: Core concepts

The core concepts of Leaving Certificate Computer Science represent major areas in the field of computer science: algorithms, data, computer systems, models, machine learning, and testing and evaluation. Students engage with the core concepts theoretically and practically in this strand. As students progress in their learning, they engage in the ALTs outlined in strand 3. Conceptual and practical classroom-based learning are combined with experimental computer-based learning throughout the two years of the course.

Strand 3: Computer science in practice

Computer science in practice provides multiple opportunities for students to apply the practices and principles and the core concepts. Students work in teams to carry out ALTs over the duration of the course, each of which results in the creation of real and/or virtual computational artefacts. These artefacts should be human-centred, and related to the students' lives and interests, while possibly being beneficial to the community and to society in general. Examples of computational artefacts students can create include programs, models, games, web pages, simulations, visualisations, digital animations, embedded systems, and apps.

The three ALTs explore the following contexts: Interactive websites, Analytics and modelling, and Embedded systems. They provide opportunities for students to develop their theoretical and procedural understanding as they grapple with computer science practices, principles and core concepts in increasingly sophisticated applications. The structure is summarised in Table 1.

Table 1: Structure of Leaving Certificate Computer Science

Strand 1: Practices and principles	Strand 2: Core concepts	Strand 3: Computer science in practice
➢ Computational thinking ➢ Computers and society ➢ Designing and developing	➢ Algorithms ➢ Computer systems ➢ Modelling and machine learning ➢ Data ➢ Evaluation and testing	➢ ALT1: Interactive websites ➢ ALT2: Analytics and modelling ➢ ALT3: Embedded systems

The outputs from each ALT are computational artefacts created using the design and development process shown in Figure 4, and a concise report outlining its development. In the report, students outline where and how the core concepts were used. The structure of the reports should reflect the design and development process. Initial reports could be in the form of structured presentations to the whole class. As students progress, their reports should become more detailed and more varied in the format of the reports. Reports and computational artefacts are collected in the student's digital portfolio, which in itself becomes an additional resource over the two years of the course.

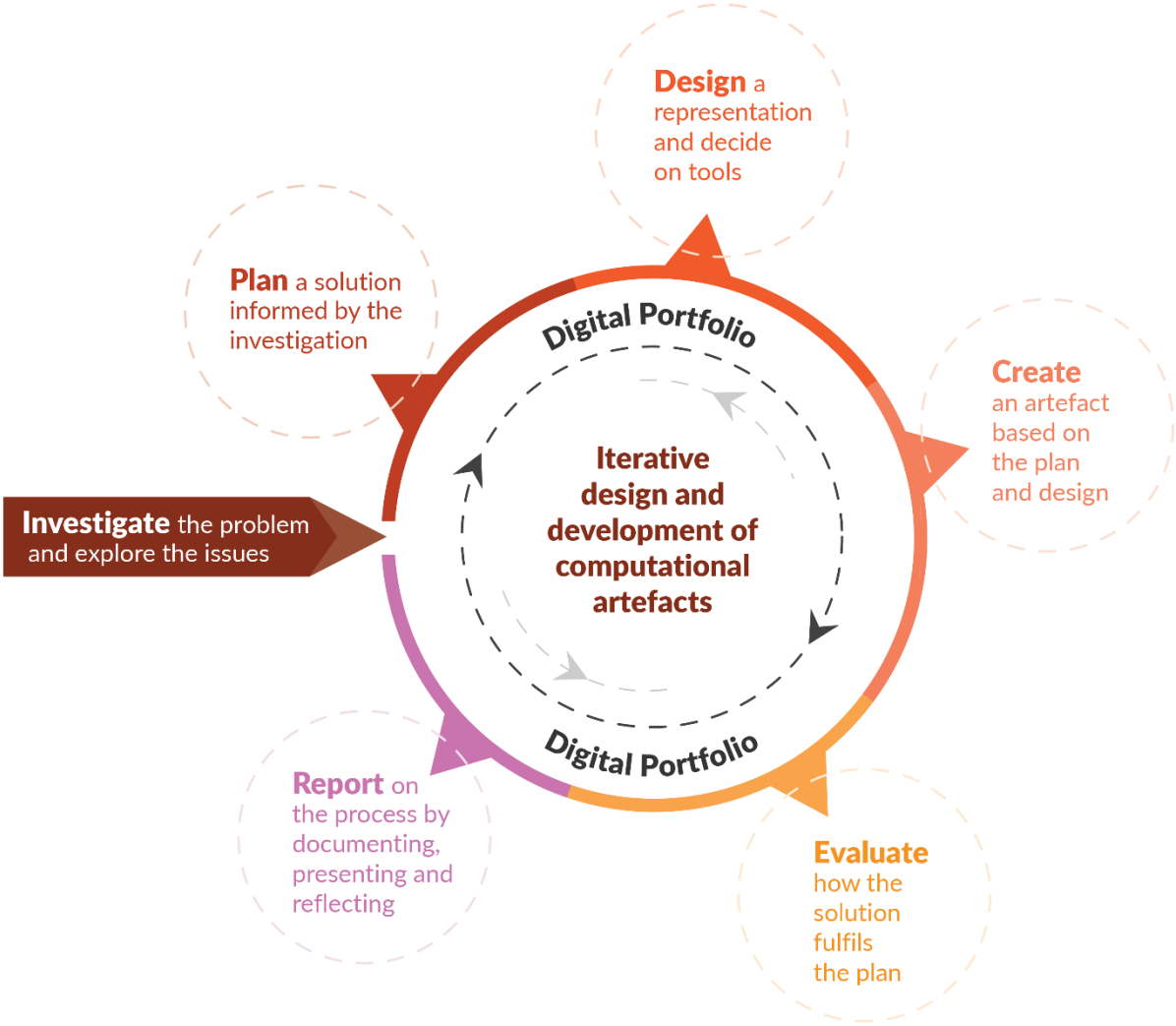

Figure 4: Overview of a design and development process

Leaving Certificate Computer Science is designed for a minimum of 180 hours of class contact time. Table 2 outlines the design of learning outcomes for ordinary and higher level.

Table 2: Design of learning outcomes for Ordinary and Higher level

Ordinary level	Higher level
Only the learning outcomes presented in normal type.	All learning outcomes including those in **bold type.**
Students engage with a broad range of knowledge, mainly concrete in nature, but with some elements of abstraction or theory.	Students engage with a broad range of knowledge, including theoretical concepts and abstract thinking, with significant depth in some areas.
Students demonstrate and use a moderate range of practical and cognitive skills and tools and to plan and develop simple investigative strategies.	Students demonstrate and use a broad range of specialised skills and tools to evaluate and use information, to plan and develop investigative strategies, and to determine solutions to varied, unfamiliar problems.
Students select from a range of procedures and apply known solutions to a variety of problems in both familiar and unfamiliar contexts.	Students identify and apply skills and knowledge in a wide variety of both familiar and unfamiliar contexts.
Students design and produce computational artefacts that serve a useful purpose.	Students design and produce computational artefacts that serve a useful purpose.

Each strand begins with an overview followed by a table containing the learning outcomes. The right-hand column contains learning outcomes which describe the knowledge, skills, values and dispositions students should be able to demonstrate after a period of learning. The left-hand column outlines specific areas that students learn about. Taken together, these provide clarity and coherence with the other sections of the specification.

Strand 1: Practices and principles

The practices and principles of computer science describe the behaviours and ways of thinking that computationally-literate students use to fully engage in a data-rich and interconnected world. Computational thinking, at the heart of computer science practices, is a problem-solving process that involves designing solutions that exploit the power of computers. The practices and principles are encountered in a context-based approach related to social, professional, and scientific contexts. Learning about the role of computers in society broadens the student's understanding of computer science and make it more meaningful and relevant. In learning about designing and developing, students come to appreciate the challenges and fulfilment involved in creating artefacts and in project management.

Strand 1 Learning Outcomes

Students learn about	Students should be able to
Computational thinking Techniques of computational thinking, such as: • Abstraction • Decomposition • Pattern recognition/ Generalisation • Logical reasoning • Algorithmic thinking Programming concepts: input-process-output, variables, operators, conditionals, loops, modularisation Computer programs: How to read, write, modify, design and test	1.1. solve problems using computational thinking techniques 1.2. explain the operation of a variety of algorithms 1.3. create algorithms to implement chosen solutions 1.4. create computer programs using programming concepts
Computing technologies to solve problems and to automate solutions **Heuristics**	1.5. explain how the power of computing enables different solutions to difficult problems 1.6. **discuss when heuristics should and could be used, and outline limitations** 1.7. evaluate alternative computational solutions to problems
Computers and society Impacts on society of computing technologies, including cultural and ethical considerations	1.8. discuss the relationships between computing technologies and society 1.9. describe the role that adaptive and assistive technology can play in the lives of people with additional needs

Students learn about	Students should be able to
The integration of computing technologies into almost all aspects of modern living	1.10. recognise the diverse roles, careers and organisations that use computing technologies
Factors empowering AI that include access to data, computing power, and new algorithms and models Current and emerging AI systems, such as agents, assistants, robotics, natural language processing, that can process and generate language, images, video and other forms of human creativity	1.11. explain factors empowering AI systems 1.12. discuss how machine learning (ML) algorithms and AI systems are used, and could be used, by societies 1.13. illustrate **and describe** a variety of AI systems
Costs and benefits of automating to include algorithmic efficiency, sustainability and decision-making	1.14. evaluate the costs and benefits of the use of computing technology in automating processes
Computing developments: • Turing machine, First electronic computers, Solid state electronics, Integrated circuits, the evolution of programming languages, the personal computer (PC), modern devices, cloud computing • Internet, World wide web (www), cybersecurity, AI including machine and deep learning • Emerging trends that could shape future computing technologies	1.15. outline the importance of developments that have shaped modern computing and consider emerging trends
Designing and developing The design and development process Working in a team, assigning roles and responsibilities, such as: analyst, project manager, designer, developer, tester, user experience Software development: approaches (agile and waterfall), life cycles and design stages	1.16. identify features of both staged and iterative design and development processes 1.17. collaborate, within a team, in a variety of roles and responsibilities, to complete computing tasks 1.18. use modular design to carry out a specific function, in hardware and/or software 1.19. describe systems using abstraction, and explain the relationship between whole and parts
User-centred design Usability and quality features that include: • communication with user	1.20. consider the perspectives of the variety of stakeholders and possible end users 1.21. consider the quality of the user experience

Students learn about	Students should be able to
• consistency • user control • aesthetics • managing errors. Communication and reporting	when interacting with computing technologies, including the role of a user interface and the factors that contribute to its usability 1.22. compare two user interfaces and identify different design decisions that shape the user experience 1.23. reflect on the design and development process

Strand 2: Core concepts

This strand introduces five core concepts that represent major content areas in the field of computer science: algorithms, computer systems, data, modelling and machine learning, and evaluation and testing. The core concepts are developed theoretically and applied practically. In this way, conceptual classroom-based learning is intertwined with experimental computer-based learning throughout the two years of the course.

Strand 2 Learning Outcomes

Students learn about	Students should be able to
Algorithms	
Pseudo code and flowcharts Algorithms: unplugged and plugged Features of algorithms such as sequencing, selection, iteration and non-recursive **and recursive** modularisation	2.1. outline the functionality of an algorithm through pseudo code and flowcharts 2.2. synthesise existing algorithms and create new ones to solve a range of problems and to fulfil specific requirements
Sorting: Selection sort, Bubble sort, **Quicksort** Search: Linear search, **Binary search** **Algorithmic efficiency regarding potential number of operations involved for similar inputs**	2.3. use search and sorting algorithms and compare the limitations and advantages of each algorithm 2.4. compare algorithms on correct functionality **and algorithmic efficiency**
Computer systems	
Components of a computer: basic von Neumann architecture **and operation including CPU-Bus-Memory, Fetch-Execute Cycle, CPU Speed and IO**	2.5. describe the different layers and components of a computer **including the operation of those components**

Students learn about	Students should be able to
Devices Computer layers: Hardware, Operating System, Application, User	2.6. compare digital and analogue inputs and outputs
Units of logic gates: from individual types to **half-adder** Numerical operations: Addition of binary numbers and conversion between binary, decimal and hexadecimal.	2.7. describe the different types of logic gates **and arrange into larger units to perform more complex tasks** 2.8. explain why the binary and hexadecimal number systems are used in digital computing and perform basic numerical operations
Web infrastructure: • the client-server model • communication protocols such as HTTP, HTTPS, TCP/IP stack • **layers: application, transport, network and physical** • basic cloud computing – scalability and flexibility	2.9. explain what is meant by the world wide web, and outline what makes up the web infrastructure
Data Data Types (Python): Numeric (int, float), Text (str), Sequence (list), Boolean (bool) Standard character sets: ASCII and Unicode Simple ciphers: Caesar, substitution **and Vignère** RSA algorithm	2.10. use data types that are common to procedural high-level languages 2.11. consider the importance of having standard character sets 2.12. use a simple cipher to encrypt/decrypt a message and outline how the RSA algorithm works
Data sources including balance, ethical implications of collecting data and bias in datasets Data storage and management: Database management systems (DBMS) • Flat file storage and retrieval • **Relational DBMS**	2.13. use a flat file database to collect, store, clean and sort data 2.14. describe different approaches to data storage and management 2.15. **compare relational with non-relational DBMS**
Modelling and machine learning Computer models and running simulations Model qualities: Purpose, input data, assumptions, and model outputs	2.16. outline the benefits and limitations of modelling and running simulations in relevant situations 2.17. evaluate the qualities of models

Students learn about	Students should be able to
Decision-making algorithms: • classical, rules-based • **supervised ML algorithms and libraries: decision-tree and multiple (linear) regression only**	2.18. use decision-making algorithms 2.19. examine how data can influence the outputs and decisions of models
Evaluation and testing Debugging, fixing and evaluating automated solutions Qualities of programs such as functionality, algorithmic efficiency, modularisation, usability and meeting user needs	2.20. identify warnings and errors in computer code and modify as required 2.21. reflect critically on, and identify limitations in, completed programs and suggest possible improvements
Software Testing: Use case, Unit, **Function, System (alpha and beta)**	2.22. evaluate programs using software testing

Strand 3: Computer science in practice

Computer science in practice provides multiple opportunities for students to use their conceptual understanding in practical applications. Students engage with three team-based ALTs during the course. Student groups design and develop computational artefacts that are personally relevant or beneficial to their community and society in general. Examples of computational artefacts that students can create include programs, models, games, simulations, visualisations, digital animations, embedded systems, and apps. Students are expected to document, reflect and present on each ALT, as part of their ongoing and managed digital portfolio.

Strand 3 Learning Outcomes

Applied Learning Task 1 : Interactive websites

Design is one of the key practices and principles of computer science. As designers and creators of technology, students can be innovative and expressive through the creation of artefacts. Computer science is also an information-intensive discipline that involves the selection, evaluation, recording and presentation of information. Students come to see the richness and complexity of how to communicate with, and provide information about, the world around them. In this ALT, students develop a website which the user can interact with. The students use HTML/CSS to build their website, and they can enhance it using other technologies. Through planning and designing an interactive application that can meet a set of user needs, students

can experience first-hand the design and development process, while enhancing their knowledge of the role of computing systems.

Students learn about	Students should be able to
ALT1: Interactive websites Information systems User-centred and web design Graphical User Interfaces (GUIs), HTML/CSS Design and development process	3.1. understand and list user needs and requirements before defining a solution 3.2. create a user interface taking the quality of the user experience into account 3.3. create an interactive application, using HTML/CSS, that can display information to meet a set of user needs

Applied Learning Task 2: Analytics and modelling

Often with data, it can be challenging to see patterns, spot trends, and understand factors shaping the data. Data analytics and computer models can assist humans in gaining insights. In this ALT, students develop systems to analyse, interpret, and gain insights from data. They identify interdisciplinary topics, pose questions, gather, represent, and analyse data, build models, and test scenarios. Problems or issues that are not amenable to analytics can often be analysed through modelling, and so students could for example create two artefacts designed to explore modelling and analytics separately. Students could also create a single, interconnected artefact that uses the same dataset for both analytics and modelling. This task deepens students' understanding of the practices and principles of computer science while also enabling students to investigate issues of relevance to them.

Students learn about	Students should be able to
ALT2: Analytics and modelling Data preparation process: gather, structure and transform data for analysis Statistical measures such as frequency, averages, spread	3.4. use the data preparation process and represent data graphically 3.5. create a data-based model that can test scenarios and make predictions

Students learn about	Students should be able to
Running simulations and evaluating outcomes Using data to inform and gain insights	3.6. analyse and interpret data and model outputs, in a way that informs decision-making

Applied Learning Task 3: Embedded systems

The design and application of computer hardware and software are a central part of computer science. In this ALT, students will implement a system that uses sensors and controls inputs and outputs as part of an embedded system. By building the component parts of a computer system, students will deepen their understanding of how computers work and how they can be embedded in our everyday environments.

Students learn about	Students should be able to
ALT3: Embedded systems Computer systems Computing and controlling inputs and outputs How to use and manage data – digital and analogue Design and development process	3.7. use inputs and outputs within an embedded system 3.8. create a program that utilises inputs and outputs 3.9. create applications using embedded systems

Teaching for student learning

The three strands of learning are designed to be interwoven and interconnected. The ALTs of strand 3 in particular are further designed to allow for interconnection and they are intended to be the lens through which students experience the course. Student learning in Leaving Certificate Computer Science can be best achieved through teaching approaches aligned to the design and intention of the specification. Teachers are best placed to make professional judgements on how to facilitate an effective balance for the students of theoretical, applied and problem-based learning, project management and authentic collaborative activities. Teaching for student learning therefore requires a corresponding balance of teaching strategies, decided by the teacher as being most beneficial to the students, while developing necessary key competencies.

Through ALTs, students can work together to apply the practices, principles and core concepts of the course. In addition to cumulative learning across the ALTs, and other strands of study, students have opportunities to frequently engage with learning outcomes in a variety of ways. This pedagogical approach opens the course for students to put design and development processes into practice, develop project management skills and collaborate on problem-solving strategies, while also learning how to manage their own progress and learning. Teaching through the lens of ALTs further enables students to make connections between computer science, other subjects, and everyday experiences, as they design and build computational artefacts that are personally relevant to them or their peers, to their community or to society in general.

Teachers supporting self-directed learning and reflection can enable students to plan, monitor, and evaluate their own learning and improve self-efficacy. Reporting and presenting on their artefacts, and managing their digital portfolio, develops communication skills, offers moments of reflection and celebration of achievements and enhances student awareness of more diverse perspectives.

Teachers can work with students at the investigation and planning phase of the design and development process to stimulate ideas and nurture students towards relevant tasks. The problem-based nature of the course, underscored by explicit instruction and inquiry-based approaches, offers genuine opportunities for a variety of summative and formative assessments. In addition, students can be encouraged to move from broad curiosity to a critical understanding of computer science, offering many authentic moments for peer- and self-assessment.

Learning in computer science needs, as far as is practical, to be applied to problem solving and design exercises. Teachers can use their judgement to capture opportunities to develop the theoretical foundations of the students' practice, particularly through the ALTs. Other teaching strategies that can support learning in computer science include pair programming, activity-based learning, scaffolded learning strategies to support students becoming confident programmers, posing questions of varied cognitive load and facilitating peer-to-peer teaching.

Students vary in the amount and type of support they need and the use of inclusive pedagogies, such as differentiated instruction, will provide such support. In addition, strategies such as adjusting the level of skills required for tasks, varying pace and teacher intervention, and increasing opportunities for peer support can help students interact at appropriate levels.

Assessment

Assessment in senior cycle involves gathering, interpreting, using and reporting information about the processes and outcomes of learning. It takes different forms and is used for a variety of purposes. It is used to determine the appropriate route for students through a differentiated curriculum, to identify specific areas of strength or difficulty for a given student and to test and certify achievement. Assessment supports and improves learning by helping students and teachers to identify next steps in the teaching and learning process.

As well as varied teaching strategies, varied assessment strategies will support student learning and provide information to teachers and students that can be used as feedback so that teaching and learning activities can be modified in ways that best suit individual learners. By setting appropriate and engaging tasks, asking questions and giving feedback that promotes learner autonomy, assessment will support learning and promote progression, support the development of student key competencies and summarise achievement.

Assessment for certification

Assessment for certification is based on the rationale, aims and strands of study of this specification. There are two assessment components: a final examination and an additional assessment component (AAC) called the Coursework Project. The final examination will be at higher and ordinary level and the Coursework Project will be based on a common brief. Each assessment component will be set and examined by the State Examination Commission (SEC).

Examination questions will require students to demonstrate learning appropriate to each level. Differentiation at the point of assessment will also be achieved through the stimulus material used, and the extent of the structured support provided for examination students at different levels.

Assessment programming languages

Python will be the programming language assessed in the final examination. There is no restriction in choice of language used in the ALTs and Coursework Project.

Table 3: Overview of assessment for certification

Assessment Component	Weighting	Level
Coursework Project	40%	Common brief
Final examination	60%	Higher and Ordinary

Coursework Project

The Coursework Project provides an opportunity for students to display evidence of their learning and to apply the practices and principles of computer science in ways that cannot be readily assessed by the final examination. It is similar to the structure of the ALTs in strand 3 that students complete during the two years of the course. It is designed to be naturally integrated into the flow of teaching and learning and to exploit its potential to be motivating and relevant for students.

The Coursework Project must be carried out individually. While Leaving Certificate Computer Science is designed to be experienced in a collaborative and supportive environment, evidence of learning is individually submitted and assessed. It provides opportunities for students to pursue their interests in this area and to make their own design and development decisions. It can also further enhance the relevance of computer science to their lives. The Coursework Project is based on learning outcomes from across the strands, with those of strand 3 being particularly relevant. Students will have opportunities to apply, demonstrate and expand the key competencies they have developed through this subject as they complete this assessment.

The Descriptors of Quality in Table 4 are intended to provide insights into the broad expectations for students completing the AAC.

Coursework Project brief

A Coursework Project brief will be published annually by the SEC in term one of year two. It involves students creating an artefact and submitting it for marking to the SEC in term two of year two. The brief will be thematic in nature and require students to apply their learning

from across all strands, with strand 3, Computer science in practice, being of particular relevance. The digital portfolio built up by students during the course, with reports and computational artefacts from the ALTs, will be a useful resource for students carrying out the Coursework Project.

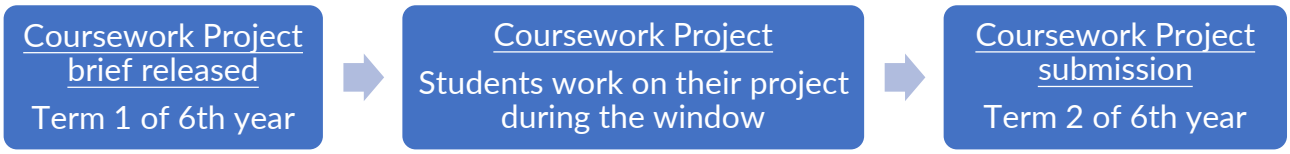

Figure 5: Window for completion of the Coursework Project

In addition to setting out the specific requirements of the Coursework Project, the brief will:

- allow students to develop their thinking and ideas on areas they would like to pursue, related to the brief
- facilitate teachers and students in their planning
- give students opportunities to further deepen their learning in computer science while also applying and expanding key competencies they have developed
- include stimulus material, and the basic and advanced features required of the student's artefact.

The dates for release of the brief and submission of the coursework will be set by the SEC each year.[1] The Coursework Project is designed to naturally integrate into the flow of teaching and learning and the window for completion will be wide enough to allow for flexibility within each classroom. Upon completion, students submit their coursework in a format prescribed by the SEC. This includes a report submitted as an accessible HTML file, where this format is appropriate for a given brief. The overview of the window for completion of the Coursework Project is shown in Figure 5. A separate document, *Guidance to Support the Completion of the Coursework Project in Leaving Certificate Computer Science*, gives detailed guidance on the coursework assessment process, including a range of matters related to the organisation, implementation, and oversight of the Coursework Project.

[1] It is envisaged students will require up to 25 hours to complete the Coursework Project. Further details are provided in the *Guidance to Support the Completion of the Coursework Project in Leaving Certificate Computer Science.*

Descriptors of Quality for the Coursework Project in Computer Science

The Coursework Project will require students to demonstrate proficiency in course content and skills that are not easily assessed by the end-of-course examination. The assessment will require students to create an innovative computational artefact, and to report on the work and process involved. Students must acknowledge, through appropriate citations and references, the source or author of all information or evidence taken from someone else's work, including the use of AI. There are six areas of achievement described in Table 4, which reflect the learning from all strands and are particularly grounded in the practices and principles of computer science. The six areas of achievement are: designing and developing, computational thinking, computer programming, problem solving, appropriate use of computing technologies and awareness of potential societal impacts.

Table 4: Descriptors of Quality: Coursework Project

	Students demonstrating a high level of achievement	Students demonstrating a moderate level of achievement	Students demonstrating a low level of achievement
Designing and developing	iteratively design, model, test, debug and evaluate solutions; choose appropriate ways to represent and evaluate solutions and final products; show considerable evidence of research into a rationale for approaching the brief; evaluate the performance and potential of the final artefact.	iteratively develop, test, and debug solutions; choose limited ways to represent and evaluate solutions and final products; show evidence of research into a rationale for approaching the brief; evaluate the performance of the final artefact.	do not iterate significantly upon solutions or the final product; test, debug and refine solutions in a linear fashion, lacking iterative processes; show limited evidence of research into a rationale for approaching the brief; do not meaningfully evaluate the final artefact.
Computational thinking	consider a variety of alternative potential solutions to the brief; systematically solve problems in the design and development process using a variety of computational thinking techniques;	consider potential solutions to the brief; solve problems in the design and development process using computational thinking techniques	consider limited alternative solutions to the brief; solve problems as part of a process with some evidence of the use of computational thinking techniques; show limited use of innovative thinking and

	use innovative thinking in design and development.	use some innovative thinking in design and development.	tend to avoid challenges that have multiple steps or parts to them.
Computer programming	show considerable evidence of appropriate use of high level data structures; implement a modular approach extensively and maximise opportunities to create well-structured code; minimise duplication and enhances readability with informative, well-placed comments; have fully tested and evaluated their programs for robustness, correct logic, functionality and good UI design.	show some evidence of appropriate use of high level data structures; implement a limited modular approach and avail of opportunities to create well-structured code; minimise duplication and enhances readability with well-placed comments; have partially tested and evaluated their programs for robustness, correct logic, functionality and good UI design.	show limited or no evidence of appropriate use of high level data structures; do not implement a modular approach nor attempt to make programs more structured; duplicate code and do not use comments in an informative way; has not tested nor evaluated their programs, to any meaningful level, for robustness, correct logic, functionality or UI design.
Problem solving	independently identify and act on patterns in problems and solutions; seek out pre-existing solutions, evaluating ideas and/or solutions from one problem context to another.	adapt existing knowledge or solutions to solve new problems; evaluate outcomes systematically from different ideas and solutions.	show limited application of previous learning to new problems; demonstrate a limited systematic approach to solving problems.
Appropriate use of computing technologies	consistently display curiosity and perseverance to investigate and analyse a spectrum of appropriate automated solutions; demonstrate an ability to apply heuristics and workarounds.	investigate a narrow spectrum of alternative automated solutions; display a tendency to stick with a solution, with limited application of heuristics or workarounds	do not deviate from an original plan to use a particular automated solution; display minimal evidence of workarounds when faced with problems.
Awareness of potential societal impacts	celebrate ambiguity and having different interpretations and as creators of artefacts, show a sensitivity to ethical, adaptive and assistive considerations, where appropriate; are aware of the potential social impact of automation in areas aligned to the brief.	show an ability to tolerate ambiguity and as creators of artefacts, demonstrate limited understanding around the ethical, adaptive and assistive implications of automation, where appropriate; are somewhat aware of the potential social impact of automation in areas aligned to the brief.	have difficulty accepting ambiguity in situations; show little or no evidence of ethical, adaptive and assistive considerations in their artefacts; are largely unaware of the potential social impact of automation in areas aligned to the brief.

Final examination

The final examination component will be comprised of computer-based practical examination and a written examination. It will consist of a range of question types. The senior cycle key competencies, developed through the study of Leaving Certificate Computer Science, are embedded in the learning outcomes and will be assessed in the context of the learning outcomes. The final examination component will include a selection of questions that will assess, appropriate to each level, the learning described in the three strands of study.

Reasonable accommodations

This Leaving Certificate Computer Science specification requires that students engage with the nature of the subject on an ongoing basis throughout the course. The assessment for certification in Leaving Certificate Computer Science involves a written and computer-based practical examination worth 60% of the available marks and an additional component worth 40%. In this context, the scheme of Reasonable Accommodations, operated by the SEC, is designed to assist students who would have difficulty in accessing the examination or communicating what they know to an examiner because of a physical, visual, sensory, hearing, or learning difficulty. The scheme assists such students to demonstrate what they know and can do, without compromising the integrity of the assessment. The focus of the scheme is on removing barriers to access, while retaining the need to assess the same underlying knowledge, skills, values, and dispositions as are assessed for all other students and to apply the same standards of achievement as apply to all other students. The Commission makes every effort when implementing this scheme to accommodate individual assessment needs through these accommodations.

More detailed information about the scheme of Reasonable Accommodations in the Certificate Examinations, including the accommodations available and the circumstances in which they may apply, is available from the SEC's Reasonable Accommodations Section.

Before deciding to study Leaving Certificate Computer Science, students, in consultation with their school and parents/guardians, should review the learning outcomes of this specification and the details of the assessment arrangements. They should carefully consider whether or not they can achieve the learning outcomes, or whether they may have a special educational need that may prevent them from demonstrating their achievement of the outcomes, even after reasonable accommodations have been applied. It is essential that if a school believes that a student may not be in a position to engage fully with the assessment for certification arrangements, they contact the SEC.

Leaving Certificate grading

Leaving Certificate Computer Science will be graded using an 8-point grading scale. The highest grade is a Grade 1; the lowest grade is a Grade 8. The highest seven grades (1-7) divide the marks range 100% to 30% into seven equal grade bands 10% wide, with a grade 8 being awarded for percentage marks of less than 30%. The grades at Higher level and Ordinary level are distinguished by prefixing the grade with H or O respectively, giving H1-H8 at Higher level, and O1-O8 at Ordinary level.

Table 5: Leaving Certificate Grading

Grade	% marks
H1/O1	90-100
H2/O2	80<90
H3/O3	70<80
H4/O4	60<70
H5/O5	50<60
H6/O6	40<50
H7/O7	30<40
H8/O8	<30

Appendix 1 Glossary of action verbs

This glossary is designed to clarify the learning outcomes. Each action verb is described in terms of what the learner should be able to do once they have achieved the learning outcome. This glossary will be aligned with the command words used in the assessment.

Action verb	Students should be able to
Analyse	study or examine something in detail, break down in order to bring out the essential elements or structure; identify parts and relationships, and to interpret information to reach conclusions
Collaborate	work jointly with another or others
Compare	give an account of the similarities and/or differences between two (or more) items or situations, referring to both (or all) of them throughout
Consider	describe patterns in data; use knowledge and understanding to interpret patterns, make predictions and check reliability
Create	bring something into existence; to cause something to happen as a result of one's actions
Describe	develop a detailed picture or image of, for example a structure or a process, using words or diagrams where appropriate; produce a plan, simulation or model
Discuss	offer a considered, balanced review that includes a range of arguments, factors or hypotheses; opinions or conclusions should be presented clearly and supported by appropriate evidence
Evaluate (data)	collect and examine data to make judgments and appraisals; describe how evidence supports or does not support a conclusion in an inquiry or investigation; identify the limitations of data in conclusions; make judgments about the ideas, solutions or methods
Evaluate (ethical judgement)	collect and examine evidence to make judgments and appraisals; describe how evidence supports or does not support a judgement; identify the limitations of evidence in conclusions and make judgments about ideas, solutions or methods
Explain	give a detailed account including reasons or causes
Examine	consider an argument or concept in a way that uncovers the assumptions and interrelationships of the issue
Identify	recognise patterns, facts, or details; provide an answer from a number of possibilities; recognise and state briefly a distinguishing fact or feature
Illustrate	use examples to describe something
Investigate	observe, study, or make a detailed and systematic examination, in order to establish facts and reach new conclusions
Interpret	use knowledge and understanding to recognise trends and draw conclusions from given information
List	provide a number of points, with no elaboration
Outline	give the main points; restrict to essentials

Action verb	Students should be able to
Recognise	identify facts, characteristics or concepts that are critical (relevant/appropriate) to the understanding of a situation, event, process or phenomenon
Reflect	give thoughtful consideration to actions, experiences, values and learning in order to gain new insights and make meaning
Solve	find an answer through reasoning
Suggest	propose a solution, hypothesis or other possible answer
Synthesise	combine different ideas to create new or enhanced understanding
Understand	have and apply a well-organised body of knowledge
Use	apply knowledge or rules to put theory into practice

Appendix 2 Glossary of core concepts

Core concept	Meaning
Algorithm	An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices. The words programming, coding and programming language are understood as follows: • Programming is the craft of analysing problems and designing, writing, testing and maintaining programs to solve them • Coding is the act of writing computer programs in a programming language • A programming language is the formal language used to give a computer instruction.
Computer systems	Computer systems consists of hardware, software, computational processes, networks and users.
Computer modelling	Using computing technologies to represent an idea, structure, process or system, and using models to test scenarios, explain, make predictions and run simulations, recognising that all models have limitations.
Data	Data is a collection of any information that can be processed or analysed by a computer. Data can be collected with both computational and non-computational tools and processes.
Machine learning	A subset of AI that includes the use of algorithms and mathematical techniques that enable machines to improve at tasks from experience.
Software evaluation	Software evaluation is the process of determining if the program or combination of programs is the best possible solution to a given problem or task. The evaluation process should include factors such as feasibility, efficiency, and ethical use.
Software testing	Software testing is the process of finding and correcting errors (bugs) in a program or system and ensuring that the program produces the intended output. Debugging includes identifying errors, gaps, and missing

Core concept	Meaning
	requirements.

Appendix 3 Abbreviations

Abbreviation	What it stands for
ASCII	American Standard Code for Information Interchange
CPU	Central Processing Unit
CSS	Cascading Style Sheet
DBMS	Database Management System
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTPS	HyperText Transfer Protocol Secure
IO	Input Output
IP	Internet Protocol
ML	Machine Learning
PC	Personal Computer
RSA algorithm	Rivest Shamir Adleman algorithm
TCP	Transmission Control Protocol
www	World Wide Web