MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information
معلومات المادة الدراسية

Module Title	**Information Technology Basics**	**Module Delivery**	
Module Type	**Basic**	☒ **Theory** ☒ **Lecture** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**	
Module Code	**NT101**		
ECTS Credits	**6**		
SWL (hr/sem)	**150**		

Module Level	1	**Semester of Delivery**	1
Administering Department	NT	**College**	CSM
Module Leader	Omar Tariq Salih	**e-mail**	E-mail
Module Leader's Acad. Title	Lecturer	**Module Leader's Qualification**	Ph.D.
Module Tutor	Name (if available)	**e-mail**	E-mail
Peer Reviewer Name	Name	**e-mail**	E-mail
Scientific Committee Approval Date	10/6/2024	**Version Number**	1.0

Relation with other Modules
العلاقة مع المواد الدراسية الأخرى

Prerequisite module	None	**Semester**	
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents

أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية

Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	Key learning outcomes: 2. **Computer Systems** – Understanding hardware, software, and system interactions. 3. **Operating Systems** – Installing, configuring, and managing OS with basic administration tasks. 4. **Networks** – Learning network architectures, protocols, IP configuration, and data transmission. 5. **Practical Applications** – Hands-on experience in hardware installation, software setup, and network configuration. 6. **Ethics & Communication** – Understanding IT ethics, privacy, intellectual property, and professional communication.
Indicative Contents **المحتويات الإرشادية**	**This course covers fundamental concepts in Information Technology (IT), providing an understanding of hardware, software, networking, databases, security, and emerging technologies.** ***1. Introduction to IT*** • **Definition, scope, evolution, and societal impact** • **Ethical considerations and challenges** ***2. Computer Systems & Hardware*** • **Computer components (CPU, memory, storage, I/O devices)** • **Organization, architecture, and data representation (binary, hexadecimal)** ***3. Operating Systems*** • **Types and functions (Windows, macOS, Linux)** • **Process management, multitasking, memory management, file systems** ***4. Software & Application Development*** • **Programming languages (Java, Python, C++)** • **Algorithms, problem-solving, and software development methodologies (Waterfall, Agile)** ***5. Networking Fundamentals*** • **Network topologies, protocols (TCP/IP, HTTP, DNS)** • **LANs, WANs, network security, and cybersecurity basics** ***6. Database Systems*** • **DBMS, relational databases, SQL, data integrity, and administration** ***7. Web Technologies*** • **Web development basics (HTML, CSS, JavaScript)** • **Client-server architecture, web applications, CMS** ***8. Information Security*** • **Data protection, security threats, authentication, encryption** ***9. Emerging Technologies*** • **Cloud computing, AI, ML, Big Data, IoT applications** ***10. Ethical & Legal Considerations*** • **IT ethics, intellectual property, cybersecurity laws, and data privacy regulations (GDPR, CCPA)** ○

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	To enhance student engagement and comprehension, the IT Basics course employs a mix of **active learning, practical application, and industry exposure**. 1. **Active Learning** – Use hands-on exercises, group projects, and discussions to apply IT concepts in real-world scenarios. 2. **Practical Exercises** – Assign programming tasks, database projects, and networking simulations with guided feedback. 3. **Real-World Examples** – Illustrate IT applications across industries like business, healthcare, and finance for better understanding. 4. **Multimedia Resources** – Incorporate videos, virtual labs, and interactive tutorials to enhance engagement and cater to different learning styles. 5. **Guest Speakers & Industry Connections** – Invite IT professionals for insights, networking, and industry relevance. 6. **Problem-Solving & Critical Thinking** – Encourage analytical thinking through IT challenges and scenario-based discussions. 7. **Assessment Variety** – Use quizzes, exams, projects, and presentations to evaluate knowledge and teamwork skills. 8. **Current & Emerging Trends** – Introduce students to AI, cloud computing, cybersecurity, and other evolving IT trends. 9. **Ethical & Legal Considerations** – Discuss responsible tech use, privacy, intellectual property, and cybersecurity ethics. 10. **Continuous Feedback & Support** – Provide regular assessments, clarification sessions, and a collaborative learning environment. These strategies ensure an **interactive, practical, and industry-relevant learning experience**, fostering critical thinking, problem-solving, and communication skills essential for IT professionals.

Student Workload (SWL) الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا			
Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	78	**Structured SWL (h/w)** الحمل الدراسي المنتظم للطالب أسبوعيا	6
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	72	**Unstructured SWL (h/w)** الحمل الدراسي غير المنتظم للطالب أسبوعيا	6
Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	**150**		

Module Evaluation تقييم المادة الدراسية					
		Time/Number	**Weight (Marks)**	**Week Due**	**Relevant Learning Outcome**
Formative assessment	**Quizzes**	3	21% (10)	5 and 10	LO #1, #2 and #5, #6
	Assignments	2	6% (10)	2 and 12	LO #3, #4 and #5, #6
	Projects / Lab.	1	10% (10)	Continuous	All
	Report	1	3% (10)	13	All
Summative assessment	**Midterm Exam**	2hr	10% (10)	7	LO #1 - #4
	Final Exam	3hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus) المنهاج الاسبوعي النظري	
	Material Covered
Week 1	Introduction to the basic concepts of Information Technology (IT) and their applications.
Week 2	Understand the structure layers of the infrastructure model of Information technology environment, especially about end-users, operating system, computer network, and storage.
Week 3	Computer Hardware • Components of a computer system • Input and output devices • Computer peripherals and their functions
Week 4	Computer Software • Types of software: system software and application software • Operating systems and their features • Software installation and management
Week 4	Practical Skills - Operating Systems Basic concepts under IT infrastructure model layers • Installation and configuration of operating systems • File management operations • System administration tasks
Week 5	End-users concept underlying infrastructure model
Week 6	Networks and Connectivity-Basic concepts under IT infrastructure model layers • Network architectures: LAN, WAN, WLAN • Network protocols: TCP/IP, Ethernet, Wi-Fi
Week 7	Networks and Connectivity Basic concepts under IT infrastructure model layers • Network architectures: LAN, WAN, WLAN • Network protocols: TCP/IP, Ethernet, Wi-Fi
Week 8	Mid term Examination
Week 9	Storage concept underlying infrastructure model
Week 10	Introduction to Data Management Basic concepts under IT infrastructure model layers

	• Basics of databases and data management systems
Week 11	Introduction to Data Management Basic concepts under IT infrastructure model layers • Structured Query Language (SQL)
Week 12	Introduction to Web Development Basic concepts under IT infrastructure model layers • HTML and CSS fundamentals
Week 13	Introduction to Web Development Basic concepts under IT infrastructure layers • Web page creation and design principles
Week 14	Ethical Considerations Basic concepts under IT infrastructure layers • Ethics in the IT field: privacy, intellectual property, responsible technology use • Professional communication and documentation skills
Week 15	Week 15: Review • Review of key concepts covered throughout the course • Completion of final projects or assignments demonstrating understanding of IT basics
Week 16	**Preparatory week before the final Exam**

Delivery Plan (Weekly Lab. Syllabus) المنهاج الأسبوعي للمختبر	
	Material Covered
Week 1	Operating System Installation and Configuration o Objective: Enable students to install and configure an operating system. o Activities: Students will install an operating system of their choice (e.g., Windows, Linux) on a virtual machine or physical computer. They will configure settings, create user accounts, and explore basic system administration tasks.
Week 2	Introduction to use the terminal of Ubuntu Operating System
Week 3	Introduction to Ubuntu environment.
Week 4	Introduction to the Shell
Week 5	Navigation
Week 6	Navigation
Week 7	Exploring the System
Week 8	Mid term Exanimation
Week 9	Manipulating Files and Directories
Week 10	Permissions
Week 11	Processes
Week 12	Configuration and the Environment
Week 13	Storage Media
Week 14	Networking
Week 15	Networking

Learning and Teaching Resources مصادر التعلم والتدريس		
	Text	**Available in the Library?**
Required Texts		No
Recommended Texts		
Websites		

Grading Scheme مخطط الدرجات				
Group	**Grade**	**التقدير**	**Marks %**	**Definition**
Success Group (50 - 100)	**A** - Excellent	**امتياز**	90 – 100	Outstanding Performance
	B - Very Good	**جيد جدا**	80 – 89	Above average with some errors
	C – Good	**جيد**	70 – 79	Sound work with notable errors
	D - Satisfactory	**متوسط**	60 – 69	Fair but with major shortcomings
	E - Sufficient	**مقبول**	50 – 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX** – Fail	**راسب (قيد المعالجة)**	(45-49)	More work required but credit awarded
	F – Fail	**راسب**	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.

MODULE DESCRIPTION FORM
نموذج وصف المادة الدراسية

Module Information
معلومات المادة الدراسية

Module Title	Problems Solving & Programming 1	**Module Delivery**	
Module Type	**Core**	☒ **Theory** ☒ **Lecture** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**	
Module Code	**NT102**		
ECTS Credits	**8**		
SWL (hr/sem)	**200**		

Module Level	1	**Semester of Delivery**	1
Administering Department	NT	**College**	CSM
Module Leader	Name	**e-mail**	E-mail
Module Leader's Acad. Title	Professor	**Module Leader's Qualification**	Ph.D.
Module Tutor	Name (if available)	**e-mail**	E-mail
Peer Reviewer Name	Name	**e-mail**	E-mail
Scientific Committee Approval Date	10/06/2024	**Version Number**	1.0

Relation with other Modules

العلاقة مع المواد الدراسية الأخرى

Prerequisite module	None	**Semester**	
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents

أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية

Module Learning Outcomes

مخرجات التعلم للمادة الدراسية

Learning Outcomes for "Problem Solving and Programming I" (C++)

By the end of this course, students will be able to:

1. **Understand Problem-Solving Concepts**
   - Describe and apply systematic problem-solving techniques.
   - Break down complex problems into smaller, manageable sub-problems.
   - Develop algorithms to solve given problems efficiently.
2. **Comprehend Fundamental C++ Concepts**
   - Explain the basic syntax and structure of the C++ programming language.
   - Identify and utilize various data types, variables, and operators.
   - Demonstrate proper use of basic input/output operations in C++.
3. **Apply Control Structures**
   - Implement decision-making constructs such as `if`, `if-else`, and `switch`.
   - Use iteration techniques with loops (`for`, `while`, `do-while`).
   - Apply control flow mechanisms to develop structured programs.
4. **Develop Modular Programs Using Functions**
   - Define and invoke functions to improve program modularity.
   - Differentiate between value-returning and void functions.
   - Utilize function parameters and return values effectively.
5. **Enhance Logical Thinking and Algorithmic Design**
   - Translate real-world problems into algorithmic steps.
   - Construct flowcharts and pseudocode for problem-solving.
   - Analyze and optimize basic algorithms for efficiency.
6. **Implement Code Organization and Debugging Techniques**
   - Write clear, readable, and well-documented code.
   - Identify and correct syntax, logic, and runtime errors in C++ programs.
   - Use debugging tools and techniques to troubleshoot issues.
7. **Develop Hands-on Programming Skills**
   - Write, compile, and execute C++ programs using an Integrated Development Environment (IDE).
   - Work on small programming assignments to reinforce learning.
   - Develop simple C++ applications that solve real-world problems.
8. **Prepare for Advanced Programming Concepts**
   - Establish a strong foundation for object-oriented programming and data structures.
   - Demonstrate readiness for more complex C++ programming topics.
   - Understand the significance of programming in software development.

Indicative Contents

المحتويات الإرشادية

Indicative Contents for "Problem Solving and Programming I" (C++)

Module 1: Introduction to Problem Solving and Programming

Understanding problem-solving strategies
Algorithm design and flowcharts
Introduction to programming languages and the role of C++
Setting up a C++ development environment

Module 2: Basics of C++ Programming

Structure of a C++ program
Writing, compiling, and running a simple C++ program
Basic input and output using `cin` and `cout`
Comments and code documentation

Module 3: Variables, Data Types, and Operators

Declaring and initializing variables
Data types: `int`, `float`, `double`, `char`, `bool`

	Constants and symbolic constants (`const`, `#define`) Arithmetic, relational, logical, and bitwise operators Type conversion and casting ***Module 4: Control Structures*** Conditional statements: `if`, `if-else`, `nested if`, `switch-case` Loops: `for`, `while`, `do-while` Nested loops and their applications Break and continue statements ***Module 5: Functions and Modular Programming*** Defining and calling functions Function parameters and return values Scope of variables: local vs. global variables Function overloading and inline functions ***Module 6: Arrays and Strings*** Declaring and using one-dimensional and multidimensional arrays Array initialization and processing Character arrays and string handling functions (`strlen`, `strcpy`, `strcmp`, etc.) Introduction to the C++ string class ***Module 7: Pointers and Memory Management*** Introduction to pointers and pointer arithmetic Dynamic memory allocation (`new` and `delete`) Pointer and array relationship Passing pointers to functions ***Module 8: File Handling in C++*** Introduction to file streams (`ifstream`, `ofstream`, `fstream`) Reading from and writing to files File handling operations (open, close, read, write) Error handling in file operations ***Module 9: Debugging and Code Optimization*** Common programming errors (syntax, logical, runtime errors) Debugging techniques and tools Writing efficient and readable code Code documentation and best practices ***Module 10: Mini Project and Application Development*** Solving real-world problems using C++ Developing small projects (e.g., simple calculator, grade management system, number guessing game) Code review and improvement techniques Presenting and documenting projects This indicative content ensures a strong foundation in C++ programming, focusing on problem-solving skills and hands-on implementation.

Learning and Teaching Strategies

استراتيجيات التعلم والتعليم

Strategies	**Learning Strategies for "Problem Solving and Programming I" (C++)** 1. **Hands-on Practice** – Regular coding exercises and challenges to reinforce concepts. 2. **Incremental Learning** – Gradual introduction of topics, building on previous knowledge. 3. **Algorithmic Thinking** – Emphasis on problem breakdown, flowcharts, and pseudocode. 4. **Collaborative Learning** – Pair programming and group projects for teamwork. 5. **Instructor-Guided Practice** – Step-by-step demonstrations with feedback. 6. **Real-World Applications** – Mini-projects and assignments based on real scenarios. 7. **Debugging Techniques** – Teaching error identification and debugging skills. 8. **Self-Paced Learning** – Supplementary online resources for independent study. 9. **Regular Assessments** – Quizzes, coding assignments, and exams for evaluation. 10. **Computational Thinking** – Logical puzzles and efficiency-focused problem-solving. These strategies ensure a structured, hands-on, and engaging learning experience in C++ programming.

Student Workload (SWL)

الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا

Structured SWL (h/sem) **الحمل الدراسي المنتظم للطالب خلال الفصل**	108	**Structured SWL (h/w)** **الحمل الدراسي المنتظم للطالب أسبوعيا**	7
Unstructured SWL (h/sem) **الحمل الدراسي غير المنتظم للطالب خلال الفصل**	92	**Unstructured SWL (h/w)** **الحمل الدراسي غير المنتظم للطالب أسبوعيا**	6
Total SWL (h/sem) **الحمل الدراسي الكلي للطالب خلال الفصل**	**200**		

Module Evaluation

تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	Quizzes	3	21% (10)	5 and 10	LO #1, #2 and #10, #11
	Assignments	2	6% (10)	2 and 12	LO #3, #4 and #6, #7
	Projects / Lab.	1	10% (10)	Continuous	All
	Report	1	3% (10)	13	LO #5, #8
Summative assessment	Midterm Exam	2hr	10% (10)	7	LO #1 - #4
	Final Exam	3hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus)

المنهاج الاسبوعي النظري

	Material Covered
	Week 1-2: Introduction to Problem Solving & C++ Basics Problem-solving techniques, algorithms, and flowcharts Writing, compiling, and executing C++ programs Basic input/output operations (`cin`, `cout`), data types, and variables *Week 3-4: Control Structures* Decision-making structures: `if-else`, `switch-case` Iteration: `for`, `while`, `do-while` loops Using loops and conditional statements in problem-solving *Week 5-6: Functions and Modular Programming* Function definition, parameters, and return values Scope of variables and function overloading Writing modular, reusable code *Week 7-8: Arrays and Strings* One-dimensional and multi-dimensional arrays String handling and common string functions (`strlen`, `strcpy`, `strcmp`) Array and string manipulation in C++ *Week 9-10: Pointers and File Handling* Introduction to pointers and dynamic memory allocation (`new`, `delete`) File operations: reading/writing text files (`ifstream`, `ofstream`) Using pointers in arrays and functions *Week 11-12: Debugging & Algorithmic Problem-Solving* Common programming errors and debugging techniques Algorithm design and structured problem-solving Writing optimized and readable code *Week 13-14: Mini Project Development* Planning and implementing a real-world C++ project Code optimization, debugging, and documentation Project presentation and review *Week 15: Final Review & Assessment* Summary of key concepts, advanced problem-solving exercises Final coding assessment and course feedback This condensed syllabus ensures a structured, hands-on approach to learning C++ and problem-solving techniques.

Delivery Plan (Weekly Lab. Syllabus)

المنهاج الأسبوعي للمختبر

	Material Covered
	Week 1-2: Basic C++ Programming and I/O Operations Setting up a C++ development environment Writing and executing basic C++ programs Using input/output (`cin`, `cout`) and different data types ***Week 3-4: Control Structures (Decision Making & Loops)*** Implementing `if-else` and `switch-case` conditions Using loops (`for`, `while`, `do-while`) for iterative tasks Pattern printing and factorial calculation ***Week 5-6: Functions and Arrays*** Writing functions with and without return values Implementing function overloading Working with one-dimensional arrays (sorting, searching) ***Week 7-8: Strings, Multi-Dimensional Arrays, and Pointers*** Performing string operations (`strlen`, `strcpy`, `strcmp`) Matrix operations with multi-dimensional arrays Basic pointer operations and dynamic memory allocation ***Week 9-10: File Handling and Debugging*** Reading and writing text files in C++ Implementing a simple file-based student record system Debugging techniques and code optimization ***Week 11-12: Problem-Solving and Mini-Project Implementation*** Solving logic-based problems (prime numbers, Fibonacci series) Planning and structuring a simple C++ project Writing and testing core project functionalities ***Week 13-15: Project Development, Review, and Final Assessment*** Expanding and debugging the mini-project Writing documentation and preparing a project presentation Final coding assessment and course review

Learning and Teaching Resources

مصادر التعلم والتدريس

	Text	Available in the Library?
Required Texts		No
Recommended Texts		
Websites		

Grading Scheme مخطط الدرجات				
Group	**Grade**	**التقدير**	**Marks %**	**Definition**
Success Group (50 - 100)	**A** - Excellent	**امتياز**	90 - 100	Outstanding Performance
	B - Very Good	**جيد جدا**	80 - 89	Above average with some errors
	C - Good	**جيد**	70 - 79	Sound work with notable errors
	D - Satisfactory	**متوسط**	60 - 69	Fair but with major shortcomings
	E - Sufficient	**مقبول**	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX** – Fail	**راسب (قيد المعالجة)**	(45-49)	More work required but credit awarded
	F – Fail	**راسب**	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.

MODULE DESCRIPTION FORM
نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية					
Module Title	**Calculus**			**Module Delivery**	
Module Type	**Support**			☒ **Theory** ☒ **Lecture** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**	
Module Code	**NT103**				
ECTS Credits	**5**				
SWL (hr/sem)	**125**				
Module Level		1	**Semester of Delivery**		1
Administering Department		NT	**College**		CSM

Module Leader	Ms. Merna Adil	**e-mail**	E-mail
Module Leader's Acad. Title	Lecturer	**Module Leader's Qualification**	Ph.D.
Module Tutor	Name (if available)	**e-mail**	E-mail
Peer Reviewer Name	Name	**e-mail**	E-mail
Scientific Committee Approval Date	01/06/2024	**Version Number**	1.0

Relation with other Modules **العلاقة مع المواد الدراسية الأخرى**			
Prerequisite module	None	**Semester**	
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents
أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية

Module Objectives **أهداف المادة الدراسية**	1. **Understanding Limits & Continuity** – Develop a strong foundation in limits, evaluate them algebraically and graphically, and understand continuity. 2. **Calculating Derivatives** – Learn differentiation rules (power, product, quotient, chain) and apply them to optimization, related rates, and approximations. 3. **Analyzing Functions** – Use calculus tools to determine intervals of increase/decrease, find extrema, identify inflection points, and sketch graphs. 4. **Evaluating Integrals** – Master definite/indefinite integrals, antiderivatives, and integration techniques (substitution, integration by parts), applying them to area, average value, and differential equations. 5. **Fundamental Theorem of Calculus** – Understand its implications, evaluate definite integrals, and relate integrals to accumulation functions. 6. **Problem-Solving Skills** – Apply calculus concepts to real-world and mathematical problems, enhancing logical reasoning and analytical thinking. 7. **Mathematical Communication** – Clearly express mathematical reasoning, use proper notation, and present solutions in a structured manner. •
Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	**Summarized Learning Outcomes for Calculus (Networks Department)** Upon successful completion of the course, students should be able to: 1. **Knowledge & Understanding** o Demonstrate a solid grasp of limits, derivatives, and integrals. o Explain the relationships between functions, limits, and continuity. o Apply calculus concepts to solve problems related to rates of change, optimization, and approximation. 2. **Computational Skills** o Perform algebraic manipulations with limits, derivatives, and integrals. o Compute limits and derivatives using rules like power, chain, and product rules. o Solve calculus-based problems in optimization, related rates, and approximations. 3. **Application & Modeling** o Apply calculus to real-world scenarios in physics, economics, engineering, and biology. o Interpret and analyze graphs, tables, and function equations. o Model physical and mathematical phenomena using calculus. 4. **Critical Thinking & Problem-Solving** o Identify relevant calculus concepts for problem-solving. o Develop logical reasoning strategies for complex problems. o Evaluate solution accuracy and interpret real-world implications. 5. **Mathematical Communication** o Clearly express mathematical ideas using proper notation. o Communicate reasoning through written explanations, diagrams, and graphs. o Present solutions in a well-organized and coherent manner.

	6. **Mathematical Reasoning & Proof** o Justify mathematical statements using logical reasoning and proof techniques. o Construct and understand proofs of key calculus theorems. o Emphasize precision and rigor in calculus reasoning. 7. **Technology & Calculus Tools** o Use graphing calculators or software for visualization and analysis. o Apply technology for numerical computations, function graphing, and problem-solving. o Interpret and validate results obtained from computational tools in calculus applications. •
Indicative Contents **المحتويات الإرشادية**	This course covers fundamental **calculus concepts** essential for computer science applications, focusing on limits, differentiation, integration, and their applications. **Introduction to Calculus** Functions: Domain, range, graphing Types: Polynomial, exponential, logarithmic, trigonometric Limits and their properties **Limits & Continuity** Evaluating limits algebraically and graphically o One-sided and infinite limits o Function continuity and its properties 3. **Differentiation** o Definition of the derivative as a rate of change o Derivative rules (power, sum, difference, trigonometric, exponential, logarithmic) o Higher-order derivatives 4. **Applications of Differentiation** o Tangent lines and rates of change o Optimization problems (maxima/minima) o Related rates o Approximation (differentials, linearization) 5. **Techniques of Differentiation** o Product and quotient rules o Chain rule for composite functions o Implicit differentiation o Derivatives of inverse trigonometric functions 6. **Curve Sketching** o Function analysis (increasing/decreasing intervals, local extrema, concavity) o Asymptotes, intercepts, and symmetry o Graphing using critical points 7. **Integration** o Antiderivatives and indefinite integrals o Definite integrals as areas o Basic integration rules o Techniques: Substitution, integration by parts 8. **Applications of Integration** o Area between curves o Average value of a function o Solving basic differential equations 9. **Fundamental Theorem of Calculus** o Statement and application o Evaluating definite integrals o Area under a curve and accumulation functions 10. **Numerical Methods** • Approximating definite integrals (midpoint, trapezoidal rules) • Simpson's rule • Practical applications of numerical integration

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	To enhance student engagement and understanding in **Wireless Sensor Networks (WSN)**, a combination of **active learning, practical applications, and technology integration** is recommended. 1. **Clear Explanation & Examples** o Provide step-by-step explanations of WSN concepts, protocols, and architectures. o Use real-world examples to illustrate applications of sensor networks. 2. **Active Learning** o Engage students in discussions, case studies, and problem-solving exercises. o Encourage hands-on activities like sensor network simulations and coding exercises. 3. **Visual Representations** o Use diagrams, network topology illustrations, and animations to explain concepts. o Demonstrate sensor placements, data flow, and communication protocols visually. 4. **Real-world Applications** o Showcase WSN applications in **healthcare, smart cities, environmental monitoring, and IoT**. o Discuss real-world case studies where WSNs improve efficiency and automation. 5. **Practice & Feedback** o Assign programming exercises, simulations, and network configuration tasks. o Provide constructive feedback on lab work, reports, and problem-solving approaches. 6. **Technology Integration** o Use network simulation tools like **NS3, Contiki, or MATLAB** for hands-on learning. o Demonstrate sensor deployment and data visualization through real or virtual labs. 7. **Conceptual Understanding** o Encourage critical thinking and analysis of **WSN architectures, routing protocols, and energy efficiency**. o Promote understanding of trade-offs in WSN design, such as power consumption vs. communication range. 8. **Collaborative Learning** o Assign **team-based projects** where students design and simulate WSN solutions. o Encourage peer discussions to explore challenges in sensor network deployments. 9. **Formative Assessment** o Use quizzes, in-class discussions, and problem-solving activities to assess understanding. o Provide timely feedback on students' comprehension and problem areas. 10. **Office Hours & Support** • Offer one-on-one guidance on difficult topics, project consultations, and additional tutorials. • Maintain an open communication channel for addressing student concerns.

Student Workload (SWL) الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا			
Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	48	**Structured SWL (h/w)** الحمل الدراسي المنتظم للطالب أسبوعيا	6
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	77	**Unstructured SWL (h/w)** الحمل الدراسي غير المنتظم للطالب أسبوعيا	6
Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	125		

Module Evaluation
تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	Quizzes	2	10% (10)	5 and 10	LO #1, #2 and #10, #11
	Assignments	2	10% (10)	2 and 12	LO #3, #4 and #6, #7
	Projects / Lab.	1	10% (10)	Continuous	All
	Report	1	10% (10)	13	LO #5, #8 and #10
Summative assessment	Midterm Exam	2hr	10% (10)	7	LO #1 - #7
	Final Exam	3hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus)
المنهاج الاسبوعي النظري

	Material Covered
Week 1	• General Review: Real number and their properties, intervals, inequalities, Absolute value with its properties.
Week 2	• The Real Function and its graphs, domain and range.
Week 3	• Limits and continuity: definition, theorems, properties, types of limits.
Week 4	• Derivative of function: Theory of derivative, higher order derivative, Implicit derivative, Chain rule.
Week 5	• The integral: definite and indefinite integrals and applications.
Week 6	• Transcendental Functions: Exponential Function, Logarithmic Function with derivatives and integrals
Week 7	• Trigometric Function, Inverse Trigometric Function with derivatives and integrals

Week 8	• Mid – Term Examination
Week 9	• Matrices: definition, types and their operations
Week 10	• Determinants: definition, properties and applications, The Cofactor and the Inverse of matrix by Cofactor, Grammar's method
Week 11	• Laplace transformation: definition, examples
Week 12	• Inverse Laplace transformation: definition, examples
Week 13	• Series: definition, Taylor and Maclaurin series
Week 14	• Fourier series
Week 15	• Review the Course

Learning and Teaching Resources **مصادر التعلم والتدريس**		
	Text	**Available in the Library?**
Required Texts		No
Recommended Texts		
Websites		

Grading Scheme **مخطط الدرجات**				
Group	**Grade**	**التقدير**	**Marks %**	**Definition**
Success Group (50 - 100)	**A** – Excellent	**امتياز**	90 - 100	Outstanding Performance
	B - Very Good	**جيد جدا**	80 - 89	Above average with some errors
	C – Good	**جيد**	70 - 79	Sound work with notable errors
	D – Satisfactory	**متوسط**	60 - 69	Fair but with major shortcomings
	E – Sufficient	**مقبول**	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX** – Fail	**راسب (قيد المعالجة)**	(45-49)	More work required but credit awarded
	F – Fail	**راسب**	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.

MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information
معلومات المادة الدراسية

Module Title	**Logic Design Fundamentals**	**Module Delivery**	
Module Type	**Core**	☒ **Theory** ☒ **Lecture** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**	
Module Code	**NT104**		
ECTS Credits	**7**		
SWL (hr/sem)	**175**		

Module Level	1	**Semester of Delivery**	1
Administering Department	NT	**College**	CSM
Module Leader	Dr. Zaid Jafer Fadil	**e-mail**	E-mail
Module Leader's Acad. Title	Lecturer	**Module Leader's Qualification**	Ph.D.
Module Tutor	Name (if available)	**e-mail**	E-mail
Peer Reviewer Name	Name	**e-mail**	E-mail
Scientific Committee Approval Date	18/06/2023	**Version Number**	1.0

Relation with other Modules
العلاقة مع المواد الدراسية الأخرى

Prerequisite module	None	**Semester**	
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents
أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية

Module Objectives **أهداف المادة الدراسية**	1. **Understand Digital Logic Basics** o Learn binary number systems, logic gates, and Boolean algebra fundamentals. 2. **Combinational Logic Design** o Design and analyze circuits using logic gates, multiplexers, decoders, and encoders. o Simplify Boolean expressions and implement logic functions. 3. **Sequential Logic Design** o Understand flip-flops, registers, and counters. o Design sequential circuits using state diagrams and transition tables. 4. **Boolean Algebra & Simplification** o Apply Boolean laws, De Morgan's theorem, and Karnaugh maps to simplify expressions. 5. **Circuit Analysis & Simulation** o Analyze and validate digital circuits using simulation tools. 6. **Design Methodologies** o Learn structured design approaches, modular design, and documentation techniques. 7. **Problem-Solving & Critical Thinking** o Apply logical reasoning to solve complex digital logic problems. o Break down problems into smaller, manageable components. 8. **Hands-on Lab Exercises** o Design, implement, and test digital circuits using hardware and simulation software. 9. **Collaboration & Communication** o Work in teams on circuit design projects. o Present and document design solutions effectively.
Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	Upon completing the course, students should be able to: 1. Understand the fundamental concepts of digital logic design, including binary number systems, logic gates, and Boolean algebra. 2. Comprehend the principles and characteristics of combinational and sequential logic circuits. 3. Explain the behavior and operation of various digital components, such as flip-flops, registers, and counters. 4. Understand the different types of memory devices and programmable logic devices. 5. Design and implement combinational logic circuits using logic gates, multiplexers, decoders, and encoders. 6. Simplify Boolean expressions and optimize logic functions using Boolean algebra and logic simplification techniques. 7. Design and analyze sequential logic circuits using state diagrams, transition tables, and timing diagrams. 8. Implement digital circuits using programmable logic devices (PLDs) and understand their programming and configuration. 9. Apply logical reasoning and critical thinking skills to solve problems related to digital logic design. 10. Design, implement, and test digital logic circuits using hardware components and/or digital simulation software. 11. Use appropriate software tools for circuit simulation, validation, and

	analysis. 12. Work effectively in teams to collaboratively design and implement digital logic circuits. 13. Collaborate and contribute to group projects and discussions related to digital logic design.
Indicative Contents **المحتويات الإرشادية**	This course provides fundamental concepts and practical applications in **digital logic and circuit design**, covering number systems, Boolean algebra, combinational and sequential circuits. 1. **Introduction to Digital Logic** o Number systems: Binary, decimal, octal, hexadecimal o Boolean algebra: Logic operators, truth tables, laws o Logic gates: AND, OR, NOT, XOR, NAND, NOR, XNOR 2. **Combinational Logic Design** o Combinational circuit design and analysis o Boolean functions: Expressions, canonical forms, simplification techniques o Karnaugh maps: Truth table conversion, simplification o Implementation of combinational circuits using logic gates 3. **Combinational Circuits** o **Multiplexers & Demultiplexers**: Operation, design, applications o **Encoders & Decoders**: Operation, design, applications 4. **Sequential Logic Design** o Flip-flops: SR, D, JK, and T flip-flops

Learning and Teaching Strategies **استراتيجيات التعلم والتعليم**	
Strategies	1. **Lectures** – Deliver theoretical concepts using multimedia, real-world examples, and interactive presentations. 2. **Hands-on Labs** – Conduct practical sessions on hardware assembly, network configuration, software installation, programming, and troubleshooting. 3. **Group Discussions & Collaborative Learning** – Encourage teamwork through case studies, group projects, and peer discussions. 4. **Guest Speakers & Industry Experts** – Invite professionals to provide industry insights, trends, and career guidance. 5. **Online Resources & Multimedia** – Supplement learning with **video lectures, interactive tutorials, virtual labs, and quizzes**. 6. **Assignments & Projects** – Assign practical tasks that apply networking concepts to real-world scenarios, fostering problem-solving skills. 7. **Assessments & Feedback** – Conduct quizzes, exams, and continuous evaluations with **timely, constructive feedback**. 8. **Online Discussion Forums & Communication** – Establish digital platforms for **student collaboration, resource sharing, and Q&A sessions**.

Student Workload (SWL) الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا			
Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	93	**Structured SWL (h/w)** الحمل الدراسي المنتظم للطالب أسبوعيا	7
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	82	**Unstructured SWL (h/w)** الحمل الدراسي غير المنتظم للطالب أسبوعيا	6
Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	**175**		

Module Evaluation
تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	Quizzes	3	21% (10)	5 and 10	LO #1, #2 and #10, #11
	Assignments	2	6% (10)	2 and 12	LO #3, #4 and #6, #7
	Projects / Lab.	1	10% (10)	Continuous	All
	Report	1	3% (10)	13	LO #5, #8 and #10
Summative assessment	Midterm Exam	2hr	10% (10)	7	LO #1 - #7
	Final Exam	3hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus) المنهاج الاسبوعي النظري	
	Material Covered
Week 1	Introduction to Digital Logic Design - Digital logic levels and signals
Week 2-3	Introduction to Number systems - Binary - Decimal - Octal - Hexadecimal
Week 4-5	Introduction to logic gates and truth table (AND, OR, NOT, NAND, NOT, EX-OR, and EX-NOR)
Week 6	Boolean Algebra - Boolean variables and expressions - Boolean laws and theorems
Week 7	Simplification of Boolean expressions
Week 8-9	Combinational Logic Gate Circuits and truth tables
Week 10	Designing and analyzing combinational circuits
Week 11	Multiplexers and De-multiplexers
Week 12	Karnaugh maps and simplification techniques
Week 13	Arithmetic Circuits - Binary addition and subtraction circuits
Week 14	Binary-coded decimal (BCD) and binary-to-BCD conversion
Week 15	Flip-Flops

Delivery Plan (Weekly Lab. Syllabus) المنهاج الاسبوعي للمختبر	
	Material Covered
Week 1	Introduction to Logic Gates (AND, OR,NOT, NAND, NOR, EX-OR, and EX-NOR)
Week 2	Construct and verify the truth tables for basic logic gates (AND, OR, NOT).
Week 3	Build logic gate circuits using breadboards and test their functionality.
Week 4	Boolean Algebra and Logic Simplification - Simplify Boolean expressions using Boolean algebra laws and theorems.
Week 5	Implement simplified expressions using logic gates and verify the results.
Week 6-7	Combinational Logic Circuits - Design and implement a half-adder circuit using logic gates. - Build a full-adder circuit and test its functionality.
Week 8	- Design and construct a 4-bit binary adder-subtractor circuit. - Build a BCD adder circuit and verify its functionality.
Week 9-10	Combinational Logic Design - Design and build a 4-bit binary-to-BCD converter using combinational logic.
Week 11	- Construct and verify the functionality of a 4-bit magnitude comparator.
Week 12-13	Multiplexers and Decoders 1. Build a 4-to-1 multiplexer and test its operation using different input combinations. 2. Design and construct a 3-to-8 decoder using basic logic gates.
Week 14-15	Sequential Logic Circuits - Construct and verify the functionality of a D flip-flop using basic components.

Learning and Teaching Resources مصادر التعلم والتدريس		
	Text	**Available in the Library?**
Required Texts	Select a comprehensive logic design textbook that covers the fundamental concepts, principles, and techniques of digital logic design. Examples include "Digital Design" by M. Morris Mano and Michael D. Ciletti or "Digital Logic and Computer Design" by M. Morris Mano.	**Yes**
Recommended Texts	"Digital Design" by M. Morris Mano and Michael D. Ciletti **or** "Digital Logic and Computer Design" by M. Morris Mano	
Websites		

Grading Scheme مخطط الدرجات				
Group	**Grade**	**التقدير**	**Marks %**	**Definition**
Success Group (50 - 100)	**A** – Excellent	**امتياز**	90 - 100	Outstanding Performance
	B - Very Good	**جيد جدا**	80 - 89	Above average with some errors
	C – Good	**جيد**	70 - 79	Sound work with notable errors
	D – Satisfactory	**متوسط**	60 - 69	Fair but with major shortcomings
	E – Sufficient	**مقبول**	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX** – Fail	**راسب (قيد المعالجة)**	(45-49)	More work required but credit awarded
	F – Fail	**راسب**	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.

MODULE DESCRIPTION FORM
نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية		
Module Title	**English 1**	**Module Delivery**

Module Type	**Support**	☒ Theory ☒ Lecture ☐ Lab ☐ Tutorial ☐ Practical ☐ Seminar
Module Code	**UOM102**	
ECTS Credits	**2**	
SWL (hr/sem)	**50**	

Module Level	1	**Semester of Delivery**	1
Administering Department	NT	**College**	CSM
Module Leader	Ms. Reem Abduljabar	**e-mail**	E-mail
Module Leader's Acad. Title	Lecturer	**Module Leader's Qualification**	
Module Tutor	Name (if available)	**e-mail**	E-mail
Peer Reviewer Name	Name	**e-mail**	E-mail
Scientific Committee Approval Date		**Version Number**	1.0

Relation with other Modules

العلاقة مع المواد الدراسية الأخرى

Prerequisite module	None	**Semester**	
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Objectives **أهداف المادة الدراسية**	1. **Language Proficiency** – Develop foundational skills in **listening, speaking, reading, and writing** in English. 2. **Grammar** – Understand and apply **basic grammatical structures**, including parts of speech, sentence formation, verb tenses, and subject-verb agreement. 3. **Vocabulary Building** – Expand vocabulary with **common words, synonyms, antonyms, idioms, phrasal verbs, and collocations**. 4. **Reading Comprehension** – Improve the ability to **identify main ideas, supporting details, make inferences, and analyze texts**. 5. **Listening Comprehension** – Enhance skills in understanding **spoken English in conversations, lectures, and presentations**. 6. **Speaking Skills** – Practice **pronunciation, conversations, presentations, and expressing opinions** effectively. 7. **Writing Skills** – Develop writing through **sentence construction, paragraph development, and structured essays**. 8. **Cultural Awareness** – Gain insights into **English-language literature, media, and cultural diversity**. 9. **Study Skills** – Learn **effective study techniques, note-taking strategies, and time management** for language learning. 10. **Assessment** – Demonstrate proficiency through **quizzes, tests, presentations, writing assignments, and class participation**.
Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	Upon completing this course, students should be able to: 1. **Demonstrate Language Proficiency** – Show basic competency in **listening, speaking, reading, and writing** in English. 2. **Apply Grammar Rules** – Use correct **grammatical structures** in written and spoken communication. 3. **Expand Vocabulary** – Effectively use a broader range of **words, phrases, and expressions** in various contexts. 4. **Comprehend & Analyze Texts** – Read and understand **articles, short stories, and essays**, identifying key ideas and themes. 5. **Understand Spoken English** – Follow **conversations, lectures, and presentations** with clarity. 6. **Engage in Verbal Communication** – Express opinions and participate in **discussions, presentations, and dialogues** effectively. 7. **Improve Writing Skills** – Write **clear and structured sentences, paragraphs, and essays** with proper organization. 8. **Develop Cultural Awareness** – Gain insight into **diverse cultural perspectives** through English literature and media. 9. **Apply Study & Time Management Skills** – Utilize **effective note-taking, self-assessment, and time management strategies**. 10. **Demonstrate Proficiency through Assessment** – Show competency in **quizzes, exams, presentations, and writing assignments**.
Indicative Contents	This course focuses on developing **foundational language skills** in **reading, writing, listening, and speaking** while enhancing vocabulary, grammar, and

المحتويات الإرشادية	cultural awareness. 1. **Introduction to English Language** o Basic grammar rules, sentence structure, and punctuation o Parts of speech (nouns, verbs, adjectives, adverbs) o Simple sentence construction 2. **Vocabulary Building** o Commonly used words and expressions o Word formation (prefixes, suffixes, root words) o Synonyms, antonyms, and idiomatic expressions 3. **Reading Comprehension** o Developing reading skills through various texts o Identifying main ideas, supporting details, and making inferences o Practicing skimming and scanning techniques 4. **Writing Skills** o Paragraph structure (topic sentences, supporting details, coherence) o Sentence structure and paragraph development o Basic writing: **descriptive, narrative, and expository writing** 5. **Listening Skills** o Understanding spoken English in different contexts o Note-taking and summarizing spoken content o Listening exercises using **audio materials and dialogues** 6. **Speaking Skills** o Basic conversational skills: **greetings, introductions, and dialogues** o Pronunciation and intonation practice o Participating in **group discussions and oral presentations** 7. **Cultural Awareness** o Exploring **English-speaking cultures** o Understanding **cultural differences and communication norms** 8. **Language Practice & Activities** o **Role plays, group work, and interactive exercises** o **Language games, quizzes, and reinforcement activities**

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	1. **Communicative Approach** o Focus on **real-life communication** through **role plays, pair work, and group activities**. 2. **Task-Based Learning** o Assign **practical tasks and projects** that require English use for **problem-solving and collaboration**. 3. **Multi-Modal Learning** o Use **textbooks, audio recordings, videos, and online materials** to support different learning styles. 4. **Scaffolded Instruction** o Break complex topics into **manageable steps**, progressively increasing difficulty. 5. **Formative Assessment** o Conduct **quizzes, assignments, and in-class activities** with **timely feedback** for improvement. 6. **Technology Integration** o Utilize **language apps, online dictionaries, and multimedia tools** to enhance learning. 7. **Authentic Materials** o Use **news articles, short stories, and videos** to expose students to **real-world English usage**. 8. **Error Correction & Feedback** o Provide **constructive feedback** to improve both **written and spoken English** accuracy and fluency. 9. **Cultural Immersion** o Introduce **cultural discussions, projects, and activities** to promote intercultural awareness.

Student Workload (SWL) الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا			
Structured SWL (h/sem) **الحمل الدراسي المنتظم للطالب خلال الفصل**	32	**Structured SWL (h/w)** **الحمل الدراسي المنتظم للطالب أسبوعيا**	2
Unstructured SWL (h/sem) **الحمل الدراسي غير المنتظم للطالب خلال الفصل**	18	**Unstructured SWL (h/w)** **الحمل الدراسي غير المنتظم للطالب أسبوعيا**	1
Total SWL (h/sem) **الحمل الدراسي الكلي للطالب خلال الفصل**	**75**		

Module Evaluation
تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	Quizzes	2	10% (10)	5 and 10	LO #1, #2 and #10, #11
	Assignments	2	10% (10)	2 and 12	LO #3, #4 and #6, #7
	Projects / Lab.	1	10% (10)	Continuous	All
	Report	1	10% (10)	13	LO #5, #8 and #10
Summative assessment	Midterm Exam	2hr	10% (10)	7	LO #1 - #7
	Final Exam	2hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus)
المنهاج الاسبوعي النظري

	Material Covered
Week 1	Week 1: Introduction to English 1, course overview, and language assessment.
Week 2	Week 2: Grammar: Parts of speech, sentence structure, and basic sentence patterns.
Week 3	Week 3: Vocabulary Building: Basic word formation, synonyms, antonyms, and context clues.
Week 4	Week 4: Reading Comprehension: Developing reading strategies, understanding main ideas, and supporting details.
Week 5	Week 5: Listening Comprehension: Listening for information, note-taking, and understanding spoken dialogues.
Week 6	Week 6: Speaking Skills: Introducing oneself, asking and answering questions, and participating in simple conversations.
Week 7	Week 7: Writing Skills: Sentence construction, paragraph development, and descriptive writing.
Week 8	Week 8: Grammar: Verb tenses, subject-verb agreement, and verb forms.
Week 9	Week 9: Vocabulary Expansion: Idioms, phrasal verbs, and collocations.
Week 10	Week 10: Reading Comprehension: Inferring meaning, making predictions, and analyzing texts.
Week 11	Week 11: Listening Comprehension: Identifying main ideas, understanding specific details, and listening for inference.
Week 12	Week 12: Speaking Skills: Giving opinions, expressing agreement/disagreement, and presenting short talks.
Week 13	Week 13: Writing Skills: Narrative writing, writing emails, and basic essay structure.
Week 14	Week 14: Grammar: Modals, conditionals, and reported speech.
Week 15	Week 15: Review and Assessment: Recap of course topics, practice exercises, and final assessment.
Week 16	**Preparatory week before the final Exam**

Learning and Teaching Resources مصادر التعلم والتدريس		
	Text	**Available in the Library?**
Required Texts		
Recommended Texts		
Websites		

Grading Scheme مخطط الدرجات				
Group	**Grade**	التقدير	**Marks %**	**Definition**
Success Group (50 - 100)	**A** - Excellent	امتياز	90 - 100	Outstanding Performance
	B - Very Good	جيد جدا	80 – 89	Above average with some errors
	C - Good	جيد	70 – 79	Sound work with notable errors
	D - Satisfactory	متوسط	60 – 69	Fair but with major shortcomings
	E - Sufficient	مقبول	50 – 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX** – Fail	راسب (قيد المعالجة)	(45-49)	More work required but credit awarded
	F – Fail	راسب	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.

MODULE DESCRIPTION FORM
نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية			
Module Title	**Democracy and Human Rights**	**Module Delivery**	
Module Type	**S**	☒ **Theory** ☒ **Lecture** ☐ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**	
Module Code	**UOM104**		
ECTS Credits	**2**		
SWL (hr/sem)	**50**		
Module Level	1	**Semester of Delivery**	1
Administering Department	NT	**College**	CSM

Module Leader	Ms. Sahbaa Hikmat	**e-mail**	
Module Leader's Acad. Title	Asst. Lecturer	**Module Leader's Qualification**	
Module Tutor	Name (if available)	**e-mail**	E-mail
Peer Reviewer Name	Name	**e-mail**	E-mail
Scientific Committee Approval Date		**Version Number**	1.0

Relation with other Modules **العلاقة مع المواد الدراسية الأخرى**			
Prerequisite module	None	**Semester**	
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Objectives **أهداف المادة الدراسية**	The course aims to introduce human rights in order to defend human dignity and contribute to changing human life for the better regarding: change in values and feelings - and change in behavior, as well as promoting the idea of social justice and strengthening the link between the individual and the group and the state and its institutions, and developing monitoring skills Violations, dealing with violators, supporting the skills of understanding human rights issues, in addition to enhancing ways to participate in public affairs - citizenship.
Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	1. Human rights are a set of fundamental entitlements and freedoms that are inherent to all individuals, regardless of their nationality, race, gender, religion, or any other characteristic. 2. They are based on the principles of dignity, equality, and respect for the inherent worth and value of every human being.
Indicative Contents **المحتويات الإرشادية**	Human rights are universal, meaning they apply to everyone, everywhere, without discrimination. They encompass civil, political, economic, social, and cultural rights, and are often codified in international and national legal frameworks. Civil and political rights include the right to life, liberty, and security of person; freedom of expression, assembly, and association; the right to a fair trial; and protection against torture, arbitrary arrest, and discrimination.

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	Civil and political rights include the right to life, liberty, and security of person; freedom of expression, assembly, and association; the right to a fair trial; and protection against torture, arbitrary arrest, and discrimination.

Student Workload (SWL) الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا			
Structured SWL (h/sem) **الحمل الدراسي المنتظم للطالب خلال الفصل**	32	**Structured SWL (h/w)** **الحمل الدراسي المنتظم للطالب أسبوعيا**	1
Unstructured SWL (h/sem) **الحمل الدراسي غير المنتظم للطالب خلال الفصل**	18	**Unstructured SWL (h/w)** **الحمل الدراسي غير المنتظم للطالب أسبوعيا**	1
Total SWL (h/sem) **الحمل الدراسي الكلي للطالب خلال الفصل**	**50**		

Module Evaluation
تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	Quizzes	2	10% (10)	5 and 10	LO #1, #2 and #10, #11
	Assignments	2	10% (10)	2 and 12	LO #3, #4 and #6, #7
	Projects / Lab.	1	10% (10)	Continuous	All
	Report	1	10% (10)	13	LO #5, #8 and #10
Summative assessment	Midterm Exam	2hr	10% (10)	7	LO #1 - #7
	Final Exam	3hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus)
المنهاج الاسبوعي النظري

	Material Covered
Week 1	جذور حقوق الانسان وتطورها في التاريخ البشري
Week 2	حقوق الانسان: التحديد والتعريف والضمانات
Week 3	محتوى: الحريات العامة
Week 4	النظرية العامة للحريات العامة
Week 5	النظام القانوني للحريات العامة
Week 6	ضمانات الحرية العامة
Week 7	ضمانات الحرية العامة
Week 8	مفهوم المساواة
Week 9	مبادئ الحريات العامة تفصيليا
Week 10	حرية الأمن والشعور والاطمئنان
Week 11	حريات الفكرية
Week 12	قانون الفصل بين الدولة والكنيسة
Week 13	حرية العمل
Week 14	حرية التجارة والصناعة
Week 15	حرية التجارة والصناعة المبحث الأول: الاحزاب السياسية والحريات العامة المبحث الثاني: الحريات العامة في العالم الثالث المبحث الثالث:التقدم العلمي والتقني والحريات العامة

Learning and Teaching Resources مصادر التعلم والتدريس		
	Text	**Available in the Library?**
Required Texts	د.امير عبد العزيز، حقوق الأنسان في الأسلام	NO
Recommended Texts	نسرين محمد عبده حسونة،2015 ، حقوق الأنسان... المفهوم والخصائص والتصنيفات والمصادر	NO
Websites		

Grading Scheme مخطط الدرجات				
Group	**Grade**	**التقدير**	**Marks %**	**Definition**
Success Group (50 - 100)	**A** - Excellent	**امتياز**	90 - 100	Outstanding Performance
	B - Very Good	**جيد جدا**	80 - 89	Above average with some errors
	C - Good	**جيد**	70 - 79	Sound work with notable errors
	D - Satisfactory	**متوسط**	60 - 69	Fair but with major shortcomings
	E - Sufficient	**مقبول**	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX** – Fail	**راسب (قيد المعالجة)**	(45-49)	More work required but credit awarded
	F – Fail	**راسب**	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.

MODULE DESCRIPTION FORM
نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية		
Module Title	**Problems Solving & Programming II**	**Module Delivery**
Module Type	**Core**	☒ **Theory** ☒ **Lecture** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**
Module Code	**NT107**	
ECTS Credits	**6**	
SWL (hr/sem)	**150**	

Module Level	1	**Semester of Delivery**	2
Administering Department	NT	**College**	CSM
Module Leader	Dr. Tarfa Yaseen Hamd	**e-mail**	E-mail
Module Leader's Acad. Title	Lecturer	**Module Leader's Qualification**	Ph.D.
Module Tutor	Name (if available)	**e-mail**	E-mail
Peer Reviewer Name	Name	**e-mail**	E-mail
Scientific Committee Approval Date		**Version Number**	1.0

Relation with other Modules **العلاقة مع المواد الدراسية الأخرى**			
Prerequisite module	NT102	**Semester**	1
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents
أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية

Module Objectives
أهداف المادة الدراسية

Course Objectives for "Problem-Solving and Programming II in C++"

The objectives of this course are to:

1. **Enhance Problem-Solving Skills**
   - Develop logical and analytical thinking to break down complex problems and formulate algorithmic solutions using C++.
2. **Advance Knowledge of C++ Programming**
   - Build on fundamental programming concepts and introduce more advanced C++ features such as file handling, dynamic memory, and exception handling.
3. **Improve Code Efficiency and Optimization**
   - Teach students how to write efficient, structured, and optimized code by analyzing algorithm complexity and applying best coding practices.
4. **Develop Proficiency in File Handling**
   - Enable students to work with file input/output operations, including reading, writing, and managing different file formats.
5. **Introduce Dynamic Memory Management**
   - Provide hands-on experience in using pointers, memory allocation (`new` and `delete`), and memory deallocation to manage resources efficiently.
6. **Strengthen Understanding of String Manipulation**
   - Teach students various string operations, including searching, modifying, and formatting text using C++ string handling functions.
7. **Implement Exception Handling Mechanisms**
   - Train students to write robust programs by effectively handling runtime errors and preventing unexpected crashes using exception handling.
8. **Encourage Modular and Reusable Code Development**
   - Promote the use of functions, function overloading, and object-oriented programming principles to create modular and maintainable code.
9. **Familiarize Students with the Standard Template Library (STL)**
   - Introduce commonly used STL components such as vectors, lists, stacks, and maps to simplify problem-solving and enhance programming efficiency.
10. **Prepare Students for Real-World Programming Challenges**

- Equip students with the necessary programming skills to develop applications in various fields, including data processing, automation, and simulations.

11. **Improve Debugging and Testing Skills**

- Teach students debugging techniques, error detection methods, and testing strategies to ensure program correctness and reliability.

12. **Encourage Independent Learning and Research**

- Foster self-learning by encouraging students to explore new programming concepts, work on projects, and apply problem-solving techniques beyond classroom examples.

Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	**Learning Outcomes for "Problem-Solving and Programming II in C++"** By the end of this course, students will be able to: 1. **Apply Advanced Control Structures** o Utilize advanced decision-making structures (nested if-else, switch-case) and iterative structures (nested loops) to solve complex problems efficiently. 2. **Implement File Handling in C++** o Read from and write to files using streams, understand different file modes, and implement file error handling. 3. **Manipulate Strings Effectively** o Perform string operations such as concatenation, comparison, searching, tokenization, and transformation using C++ string libraries and functions. 4. **Utilize Dynamic Memory Allocation** o Use pointers for memory management, allocate and deallocate memory dynamically, and prevent memory leaks through proper memory handling techniques. 5. **Handle Exceptions and Errors Gracefully** o Implement exception handling using try-catch blocks, throw exceptions when necessary, and ensure robust error handling in C++ programs. 6. **Design and Implement Modular Programs** o Develop well-structured programs using functions, inline functions, function overloading, and recursion to enhance code reusability and readability. 7. **Apply Object-Oriented Programming (OOP) Principles** o Utilize encapsulation, inheritance, and polymorphism to design and implement object-oriented solutions. 8. **Analyze and Optimize Algorithms** o Evaluate algorithm efficiency using Big-O notation, optimize sorting and searching algorithms, and apply best practices for improving code performance. 9. **Develop Interactive Console-Based Applications** o Design user-friendly console applications that incorporate user input validation, error handling, and interactive menus. 10. **Work with Standard Template Library (STL)** • Use STL components such as vectors, lists, stacks, queues, and maps to simplify data structure implementation and problem-solving. 11. **Debug and Test C++ Programs** • Utilize debugging tools and techniques to identify and fix logical, syntax, and runtime errors in C++ programs. 12. **Solve Real-World Problems Using C++** • Apply problem-solving techniques and programming concepts to develop solutions for real-world applications in various domains such as data processing, automation, and simulations.
Indicative Contents **المحتويات الإرشادية**	**Indicative Contents for "Problem-Solving and Programming II in C++"** The course covers the following key topics: ***1. Review of Basic C++ Concepts*** • Recap of fundamental concepts: data types, operators, control structures (if-else,

loops)
- Functions and scope of variables (local vs. global)
- Parameter passing: pass-by-value, pass-by-reference

2. Advanced Control Structures

- Nested loops and conditional statements
- Switch-case and its applications
- Introduction to recursion and recursive problem-solving

3. File Handling in C++

- File streams and file operations (ifstream, ofstream, fstream)
- Reading from and writing to text and binary files
- Handling file errors and validation techniques

4. String Manipulation

- C++ string class and its methods
- String operations: concatenation, searching, tokenization, substring extraction
- Character arrays vs. string objects

5. Pointers and Dynamic Memory Allocation

- Introduction to pointers and pointer arithmetic
- Dynamic memory allocation and deallocation using `new` and `delete`
- Common issues: memory leaks and dangling pointers

6. Functions and Modular Programming

- Function overloading and inline functions
- Recursive functions and their applications
- Lambda functions (introduction)

7. Exception Handling

- Introduction to exception handling in C++
- Try, catch, throw mechanisms
- Standard exceptions and custom exception handling

8. Object-Oriented Programming (OOP) Concepts

- Classes and objects in C++
- Encapsulation, constructors, and destructors
- Inheritance and polymorphism
- Virtual functions and abstract classes

9. Standard Template Library (STL)

- Introduction to STL and its advantages
- Using STL containers: vectors, lists, queues, stacks, and maps
- STL algorithms and iterators

10. Algorithm Design and Optimization

- Complexity analysis (Big-O notation)

- Sorting algorithms (Bubble Sort, Quick Sort, Merge Sort)
- Searching algorithms (Binary Search, Linear Search)

11. Debugging and Error Handling

- Debugging tools and techniques in C++
- Identifying syntax, logical, and runtime errors
- Code testing strategies and unit testing

12. Developing Console-Based Applications

- Interactive input and output handling
- Menu-driven applications
- Implementing real-world problems using C++

13. Introduction to Multi-File Programs

- Organizing code into multiple files
- Header files and source files
- Using `#include`, `#define`, and namespaces effectively

14. Introduction to Advanced Topics (Optional)

- Basics of threading in C++
- Introduction to GUI programming in C++ (Qt or WxWidgets)
- Introduction to networking in C++

These topics ensure a comprehensive understanding of intermediate-level C++ programming and problem-solving techniques. Let me know if you need modifications or additional details!

-

Learning and Teaching Strategies
استراتيجيات التعلم والتعليم

Strategies	**Learning and Teaching Strategies for "Problem-Solving and Programming II in C++"** To ensure effective learning and engagement, the course will employ a combination of the following strategies: **1. Lectures and Conceptual Teaching** • Structured **theoretical lectures** will introduce key C++ programming concepts, problem-solving strategies, and algorithmic thinking. • Use of **real-world examples** to illustrate programming concepts and best practices. • **Interactive discussions** to encourage students to ask questions and clarify doubts. **2. Hands-on Programming Labs** • **Practical coding sessions** where students implement concepts learned in lectures. • **Supervised lab exercises** focusing on problem-solving techniques, debugging, and code optimization. • **Pair programming and collaborative learning** to encourage teamwork and peer-assisted learning. **3. Problem-Based Learning (PBL)** • Students will be given **real-world problem statements** and asked to develop **C++ solutions**. • Emphasis on **analyzing the problem**, breaking it down into smaller tasks, and implementing step-by-step solutions. • Encouraging students to **explain their thought process** before writing code to enhance problem-solving skills. **4. Algorithm Design and Analysis** • Focus on **stepwise refinement** and designing efficient algorithms before coding. • Introduction to **algorithm complexity analysis (Big-O notation)** to understand performance trade-offs. • Implementing different **sorting, searching, and optimization techniques** in C++. **5. Case Studies and Code Reviews** • **Analysis of well-written C++ code** to learn best practices in software development. • **Code reviews and debugging exercises** to identify errors, improve code quality, and optimize performance. • Encouraging students to **explain their code** to their peers, reinforcing learning through teaching. **6. Active Learning Strategies** • **Think-Pair-Share:** Students discuss solutions with peers before coding. • **Flipped Classroom Approach:** Pre-class materials (videos, readings) allow students to explore topics before in-class discussions. • **Quizzes and short coding challenges** to reinforce learning and assess understanding. **7. Project-Based Learning**

- **Mini-projects throughout the course** to integrate multiple programming concepts.
- A **final project** where students design, develop, and test a real-world application using C++.
- Encouraging the use of **version control tools like Git** for project management.

8. Standardized Assessments and Feedback

- **Formative assessments** (weekly coding assignments, quizzes) to track progress.
- **Summative assessments** (midterm and final exams) to evaluate understanding.
- **Instructor and peer feedback** on coding assignments to improve programming practices.

9. Use of Online Learning Resources

- **Online C++ compilers and IDEs** (e.g., CodeBlocks, Dev-C++, Visual Studio) for coding practice.
- Reference to **open-source C++ libraries** and **official documentation**.
- Access to **online coding platforms** (e.g., LeetCode, Codeforces, HackerRank) for problem-solving practice.

10. Encouraging Independent Learning and Research

- Assigning **self-directed learning tasks** to explore advanced topics.
- Encouraging students to **read technical articles, watch tutorials, and experiment** with new C++ features.
- Guidance on participating in **coding competitions and hackathons** to enhance problem-solving abilities.

Student Workload (SWL) الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا			
Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	78	**Structured SWL (h/w)** الحمل الدراسي المنتظم للطالب أسبوعيا	6
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	72	**Unstructured SWL (h/w)** الحمل الدراسي غير المنتظم للطالب أسبوعيا	8
Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	**150**		

Module Evaluation
تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	**Quizzes**	2	10% (10)	5 and 10	LO #1, #2 and #10, #11
	Assignments	2	10% (10)	2 and 12	LO #3, #4 and #6, #7
	Projects / Lab.	1	10% (10)	Continuous	All
	Report	1	10% (10)	13	LO #5, #8 and #10
Summative assessment	**Midterm Exam**	2hr	10% (10)	7	LO #1 - #7
	Final Exam	3hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus)
المنهاج الاسبوعي النظري

	Material Covered
Week 1	Review to the Introduction to Problem Solving and Programming I
Week 2-3	Structure Compound Data types
Week 4-5	String Manipulation
Week 6	Dynamic Memory Allocation
Week 7-8	Files
Week 9	Mid Term Examination
Week 10	Exception Handling
Week 11-14	Prepare Mini Project
Week 15	Revision and Review

Delivery Plan (Weekly Lab. Syllabus) المنهاج الاسبوعي للمختبر	
	Material Covered
Week 1	Week 1: Review of Introduction to Problem Solving and Programming I • Recap of problem-solving techniques and programming concepts covered in the previous course • Review exercises and discussions to reinforce the foundational knowledge
Week 2-3	Week 2 - 3: Structure Compound Data Types • Introduction to structure data types in programming • Understanding how to define and use structures in C++ • Hands-on exercises to practice working with structures
Week 4-5	Week 4-5: String Manipulation • Exploring string data types and their manipulation in C++ • String functions and operations • Practical exercises and projects involving string manipulation
Week 6	Week 6: Dynamic Memory Allocation • Understanding dynamic memory allocation in C++ • Working with pointers and memory allocation functions (new, delete) • Practical examples and exercises to reinforce the concept
Week 7-8	Week 7-8: Files • Introduction to file handling in C++ • Reading from and writing to files • Exercises and projects involving file input/output operations
Week 9	Week 9: Midterm Examination • Midterm examination covering topics from weeks 1-8 • Review of previous topics and discussion of any questions or concerns

Week 10	Week 10: Exception Handling • Introduction to exception handling in C++ • Handling runtime errors and exceptional situations • Practice exercises and examples to understand exception handling mechanisms
Week 11-14	Week 11-14: Prepare Mini Project • Working on a mini project that integrates concepts learned so far • Planning, designing, and implementing a small-scale application or program • Regular progress check-ins and guidance throughout the project development
Week 15	Week 15: Revision and Review • Recap of all topics covered throughout the course • Review exercises, discussions, and Q&A sessions to solidify understanding • Final exam preparation and guidance

Learning and Teaching Resources **مصادر التعلم والتدريس**		
	Text	**Available in the Library?**
Required Texts	Problem Solving with C++ by Walter Savitch (Author), Kenrick Mock (Author)	
Recommended Texts		
Websites		

Grading Scheme مخطط الدرجات				
Group	**Grade**	**التقدير**	**Marks %**	**Definition**
Success Group (50 - 100)	**A** - Excellent	**امتياز**	90 - 100	Outstanding Performance
	B - Very Good	**جيد جدا**	80 - 89	Above average with some errors
	C - Good	**جيد**	70 - 79	Sound work with notable errors
	D - Satisfactory	**متوسط**	60 - 69	Fair but with major shortcomings
	E - Sufficient	**مقبول**	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX** – Fail	**راسب (قيد المعالجة)**	(45-49)	More work required but credit awarded
	F – Fail	**راسب**	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.

MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية					
Module Title	**Computer Organization**			**Module Delivery**	
Module Type	**Core**			☒ **Theory** ☒ **Lecture** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**	
Module Code	NT108				
ECTS Credits	**5**				
SWL (hr/sem)	**125**				
Module Level	1		**Semester of Delivery**	2	
Administering Department	Type Dept. Code		**College**	Type College Code	
Module Leader	Dr. Ryiadth Zaghlol		**e-mail**		
Module Leader's Acad. Title	Asst. Professor		**Module Leader's Qualification**	Ph.D.	
Module Tutor	Name (if available)		**e-mail**	E-mail	
Peer Reviewer Name	Name		**e-mail**	E-mail	

Scientific Committee Approval Date		**Version Number**	1.0

Relation with other Modules **العلاقة مع المواد الدراسية الأخرى**			
Prerequisite module	NT104	**Semester**	
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Objectives **أهداف المادة الدراسية**	This course provides a **comprehensive understanding of computer architecture**, focusing on **hardware components, memory, instruction sets, processing techniques, and system performance**. 1. **Understanding Computer Architecture** o Learn **computer organization principles** and **component interconnections**. 2. **Data Representation & Arithmetic** o Study **number systems** (binary, decimal, hexadecimal) and conversions. o Understand **data representation** and **binary arithmetic operations**. 3. **Memory Systems** o Explore **memory hierarchy, caching techniques, and performance impact**. o Study **memory management and addressing methods**. 4. **Instruction Set Architecture (ISA)** o Learn **different instruction set architectures** and execution. o Analyze **machine language and ISA relationships**. 5. **Processor Organization** o Study **processor components, instruction pipelining, and control unit design**. 6. **Input/Output (I/O) Systems** o Understand **I/O devices, data transfer techniques, interrupt handling, and DMA**. 7. **Parallel Processing & Multiprocessor Systems** o Explore **parallel processing architectures and multiprocessor system design**. 8. **Performance Evaluation & Optimization** o Learn **performance metrics, system evaluation techniques, and optimization strategies**. 9. **Emerging Trends & Technologies** o Study **modern trends in computer architecture, new technologies, and future challenges**.
Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	Upon completing this course, students should be able to: 1. **Understand Computer Architecture** – Explain **computer components and their interactions** in system organization. 2. **Data Representation & Arithmetic** – Demonstrate proficiency in **number systems, data representation, and arithmetic operations**. 3. **Memory Systems** – Analyze **cache, main memory, and their impact** on performance. 4. **Instruction Set Architectures (ISA)** – Describe **ISAs, instruction formats, and execution processes**. 5. **Processor Organization** – Evaluate **processor design, pipelining techniques, and control unit implementation**. 6. **Input/Output (I/O) Systems** – Understand **I/O devices, interfaces, interrupts, and DMA mechanisms**. 7. **Parallel Processing & Multiprocessor Systems** – Discuss **parallel architectures, interconnectivity, and performance implications**. 8. **Performance Evaluation & Optimization** – Apply **performance metrics and optimization techniques** for system improvement. 9. **Emerging Trends & Technologies** – Stay updated on **new trends and**

	innovations in computer organization and architecture. 10. **Problem-Solving & Analytical Skills** – Develop **critical thinking** and apply **theoretical knowledge** to solve hardware-related challenges. 11. **Effective Communication** – Present and explain **complex computer architecture concepts** in oral and written formats. 12. **Teamwork & Collaboration** – Work effectively in **group projects and activities** related to computer architecture.
Indicative Contents **المحتويات الإرشادية**	**Summarized Indicative Contents for Computer Organization & Architecture** This course provides **a comprehensive understanding of computer organization and architecture**, covering **fundamental concepts, processing techniques, memory management, and emerging technologies**. 1. **Introduction to Computer Organization** o Basic **concepts and terminology** in computer organization. o **Historical evolution** of computer architecture. 2. **Digital Logic & Boolean Algebra** o **Binary representation and arithmetic operations**. o **Logic gates, Boolean functions, truth tables**. o **Combinational and sequential logic circuits**. 3. **Data Representation & Arithmetic** o **Number systems**: binary, decimal, hexadecimal. o **Signed and unsigned integer representation**. o **Floating-point representation and arithmetic**. 4. **Central Processing Unit (CPU)** o **Instruction Set Architecture (ISA) and machine language**. o **CPU organization, components, and control unit**. o **Instruction fetching, decoding, and execution**. 5. **Memory Systems** o **Memory hierarchy and storage technologies**. o **Cache memory organization and mapping techniques**. o **Main memory organization, addressing modes, and virtual memory**. 6. **Input/Output (I/O) Systems** o **I/O devices, interfaces, and data transfer techniques**. o **Interrupt handling and interrupt-driven I/O**. o **Direct Memory Access (DMA) and efficient data transfer**. 7. **Pipeline Processing** o **Instruction pipelining: stages, hazards, and resolution techniques**. o **Performance metrics and pipeline optimizations**. 8. **Parallel Processing & Multiprocessor Systems** o **Parallel processing architectures: SIMD, MIMD, multicore**. o **Interconnection networks and communication techniques**. 9. **Performance Evaluation & Optimization** o **Performance metrics, bottleneck analysis, and optimization techniques**. 10. **Emerging Trends & Advanced Topics** • **Superscalar processors, out-of-order execution, speculative execution**. • **Emerging technologies: quantum computing, neuromorphic computing**.

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	To **effectively teach and learn computer organization**, a **combination of theoretical understanding, hands-on practice, real-world applications, and collaborative learning** is essential. 1. **Understand the Fundamentals** o Master **binary representation, digital logic, and Boolean algebra**. o Build strong **knowledge of number systems, data representation, and arithmetic operations**. 2. **Visualize & Diagram** o Use **diagrams, flowcharts, and visual aids** to represent computer structures. o Illustrate **data flow and control signals** in CPU, memory, and I/O devices. 3. **Hands-on Experience** o Work with **computer hardware, component assembly, and low-level programming**. o Use **simulators or emulators** to observe **instruction execution and data flow**. 4. **Relate to Real-World Examples** o Connect **computer organization concepts to real-world applications**. o Understand **how these principles apply to modern computing devices**. 5. **Analyze & Evaluate Case Studies** o Study **real-world computer architectures** and design trade-offs. o Consider **performance, power consumption, and cost factors**. 6. **Solve Practice Problems** o Engage in **digital circuit analysis, assembly programming, and system optimization** exercises. 7. **Stay Updated with Current Research** o Follow **latest advancements in computer architecture** through research papers and industry trends. o Explore **emerging technologies such as quantum computing and neuromorphic computing**. 8. **Collaborate & Discuss** o Participate in **study groups, peer discussions, and online forums**. o Share knowledge and clarify concepts through interactive learning. 9. **Seek Guidance & Resources** o Use **textbooks, online courses, and academic references** for deeper learning. o Seek help from **instructors, tutors, and industry experts**. 10. **Practice Conceptual Mapping** • Create **conceptual maps** to connect various **topics and components** in computer organization. • Develop a **cohesive understanding of how different system parts work together**..

Student Workload (SWL) الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا			
Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	63	**Structured SWL (h/w)** الحمل الدراسي المنتظم للطالب أسبوعيا	7
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	62	**Unstructured SWL (h/w)** الحمل الدراسي غير المنتظم للطالب أسبوعيا	6
Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	125		

Module Evaluation تقييم المادة الدراسية					
		Time/Number	**Weight (Marks)**	**Week Due**	**Relevant Learning Outcome**
Formative assessment	**Quizzes**	2	10% (10)	5 and 10	LO #1, #2 and #10, #11
	Assignments	2	10% (10)	2 and 12	LO #3, #4 and #6, #7
	Projects / Lab.	1	10% (10)	Continuous	All
	Report	1	10% (10)	13	LO #5, #8 and #10
Summative assessment	**Midterm Exam**	2hr	10% (10)	7	LO #1 - #7
	Final Exam	3hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus) المنهاج الاسبوعي النظري	
	Material Covered
Week 1	Introduction to Computer Organization, Overview of computer systems and their components
Week 2	Digital Logic and Boolean Algebra,
Week 3	Data Representation and Arithmetic
Week 4	Central Processing Unit (CPU)
Week 5	Instruction set architecture (ISA) and machine language
Week 6	CPU organization and components
Week 7	Control unit and instruction execution
Week 8	Memory Hierarchy
Week 9	Memory organization and addressing
Week 10	Cache memory: principles, levels, and mapping techniques
Week 11	Input/Output Systems, Interrupts and DMA (Direct Memory Access), I/O performance and strategies
Week 12	Pipelining and Superscalar Techniques
Week 13	Multiprocessors and Parallel Computer Architecture
Week 14	Performance Evaluation and Benchmarking
Week 15	Review

Delivery Plan (Weekly Lab. Syllabus) المنهاج الاسبوعي للمختبر	
	Material Covered
Week 1	8086 system architecture
Week 2	8086 Instruction Set-1
Week 3	8086 Instruction Set-2
Week 4	8086 Instruction Set-3
Week 5	8086 Instruction Set-4
Week 6	8086 Instruction Set-5
Week 7	8086 Addressing Mode
Week 8	Memories (RAM, ROM)
Week 9	Cache Memory
Week 10	8086 Programming Skills
Week 11	8086 Programming Skills
Week 12	8086 I/O unit
Week 13	Memory Mapped I/O, Isolated Input Output
Week 14	Memory/Input Output Interface
Week 15	Review

Learning and Teaching Resources مصادر التعلم والتدريس		
	Text	**Available in the Library?**
Required Texts	Hwang K., 1993, "Advanced Computer Architecture: Parallelism ,Scalability and Programmability", *McGraw-Hill, Inc. ASIN: 7111067126.*	
Recommended Texts	Barry B. Brey, "The Intel Microprocessors: 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, and Pentium Pro Processor Architecture, Programming, and Interfacing", Pearson Education, 2010	
Websites	https://www.javatpoint.com/8086-microprocessor https://www.tutorialspoint.com/microprocessor/microprocessor_8086_functional_units.htm	

Grading Scheme مخطط الدرجات				
Group	**Grade**	**التقدير**	**Marks %**	**Definition**
Success Group (50 - 100)	**A** - Excellent	**امتياز**	90 - 100	Outstanding Performance
	B - Very Good	**جيد جدا**	80 - 89	Above average with some errors
	C - Good	**جيد**	70 - 79	Sound work with notable errors
	D - Satisfactory	**متوسط**	60 - 69	Fair but with major shortcomings
	E - Sufficient	**مقبول**	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX** – Fail	**راسب (قيد المعالجة)**	(45-49)	More work required but credit awarded
	F – Fail	**راسب**	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.

MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية					
Module Title	**Data Communication and Networking**			**Module Delivery**	
Module Type	**Core**			☒ **Theory** ☒ **Lecture** ☐ **Lab** ☒ **Tutorial** ☐ **Practical** ☐ **Seminar**	
Module Code	NT109				
ECTS Credits	5				
SWL (hr/sem)	125				
Module Level		1	**Semester of Delivery**		2
Administering Department		NT	**College**		CSM
Module Leader	Omar Tariq Salih		**e-mail**		E-mail
Module Leader's Acad. Title		Lecturer	**Module Leader's Qualification**		
Module Tutor	Name (if available)		**e-mail**		E-mail
Peer Reviewer Name		Name	**e-mail**		E-mail
Scientific Committee Approval Date			**Version Number**		1.0

Relation with other Modules العلاقة مع المواد الدراسية الأخرى			
Prerequisite module	NT101	**Semester**	
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Objectives **أهداف المادة الدراسية**	In this course, Networks department aims to achieve the following objectives: 1. Understand the fundamental concepts and principles of data communication and networking. 2. Gain knowledge of communication systems and their components. 3. Familiarize with different communication network types and their characteristics. 4. Comprehend the protocol architecture and the OSI model. 5. Understand the functions and operations of each layer in the OSI model. 6. Gain knowledge of the physical layer, including digital and analog transmission and various transmission media. 7. Understand data and signal concepts, analog, and digital signals, and transmission impairments. 8. Learn about digital transmission techniques, including digital-to-digital conversion and transmission modes. 9. Familiarize with analog transmission techniques, including analog-to-analog conversion and modulation. 10. Gain knowledge of multiplexing techniques such as FDM, TDM, and WDM, and understand synchronization methods. 11. Learn about guided transmission media, including twisted pair, coaxial cable, and fiber-optic. 12. Gain knowledge of unguided transmission media, including wireless, satellite, and microwave. 13. Understand error detection and correction techniques such as parity checking, checksum, and CRC. 14. Learn about multiplexing and multiple access techniques, including FDMA, TDMA, and CDMA. 15. Gain knowledge of wired LANs, with a focus on Ethernet standards, evolution, frame structure, and operation.
Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	Upon successful completion of the this course for the Networks department, students should be able to demonstrate the following learning outcomes: 1. Understand the fundamental concepts and principles of data communication and networking, including communication models, network types, and protocol architecture. 2. Demonstrate knowledge of the OSI model and its layers, and explain the functions and operations of each layer. 3. Describe the characteristics, advantages, and limitations of different transmission media, both guided and unguided. 4. Analyze and interpret data and signals, including analog and digital signals, and understand the factors affecting transmission quality. 5. Apply digital transmission techniques, including digital-to-digital

	conversion and various transmission modes. 6. Apply analog transmission techniques, including analog-to-analog conversion and modulation methods. 7. Demonstrate an understanding of multiplexing techniques, such as FDM, TDM, and WDM, and explain their advantages and applications. 8. Compare and contrast different guided transmission media, such as twisted pair, coaxial cable, and fiber-optic, based on their characteristics and capabilities. 9. Evaluate the characteristics and advantages of unguided transmission media, including wireless, satellite, and microwave technologies. 10. Apply error detection and correction techniques, including parity checking, checksum, and CRC, to ensure data integrity. 11. Analyze multiplexing and multiple access techniques, such as FDMA, TDMA, and CDMA, and understand their applications in communication systems. 12. Explain the Ethernet standard, its evolution, and the frame structure of Ethernet LANs. 13. Evaluate the security considerations and challenges associated with data communication and networking. 14. Demonstrate effective communication and collaboration skills in a networking context. 15. Apply theoretical knowledge to analyze and solve practical problems related to data communication and networking.
Indicative Contents **المحتويات الإرشادية**	The indicative contents of this course for the computer department may include the following topics: 1. Communication Systems • Introduction to communication systems • Communication models and components 2. Network Criteria and Communication Network Types • Network criteria (performance, reliability, security, etc.) • Communication network types 3. Protocol Architecture and OSI Model • Protocol architecture and layered approach • OSI model and its layers • Functions of each OSI layer 4. Physical Layer • Introduction to the physical layer • Digital and analog transmission • Transmission media: Guided and unguided 5. Data and Signals • Data and signal concepts

	• Analog and digital signals • Transmission impairments and noise 6. Digital-to-Digital Conversion • Digital-to-digital conversion techniques 7. Analog Transmission • Analog-to-analog conversion • Analog-to-digital conversion • Modulation techniques (AM, FM, PM) 8. Multiplexing • Multiplexing techniques (FDM, TDM, WDM) • Statistical multiplexing and its advantages • Synchronization and its types 9. Guided Transmission Media • Twisted pair, coaxial cable, and fiber-optic • Characteristics, advantages, and limitations 10. Unguided Transmission Media • Wireless, satellite, and microwave • Characteristics, advantages, and limitations 11. Error Detection and Correction • Introduction to error detection and correction • Parity checking, checksum, and CRC • Forward error correction techniques 12. Multiplexing and Multiple Access • Frequency division multiplexing (FDM) • Time division multiplexing (TDM) • Multiple access techniques (FDMA, TDMA, CDMA) 13. Wired LANs: Ethernet • Introduction to local area networks (LANs) • Ethernet standard and its evolution. • Ethernet frame structure and operation

Learning and Teaching Strategies **استراتيجيات التعلم والتعليم**	
Strategies	Learning and teaching strategies for this course for the Network department can include a combination of the following: 1. Lectures: Engage students through informative lectures that cover theoretical concepts and provide an overview of key topics. Use multimedia resources, visuals, and real-world examples to enhance understanding. 2. Group Discussions and Collaborative Learning: Encourage group discussions and collaborative activities to foster interaction and knowledge sharing among students. Assign group projects or case

	studies that require teamwork and problem-solving. 3. Online Resources and Multimedia: Utilize online resources, interactive tutorials, and multimedia materials to supplement learning. This can include video lectures, online quizzes, virtual labs, and interactive modules. 4. Assignments and Projects: Assign individual and group projects that require students to apply their knowledge and skills to solve real-world problems or complete practical tasks. This promotes critical thinking, problem-solving, and practical application of concepts. 5. Assessments and Feedback: Conduct regular assessments, quizzes, and examinations to evaluate students' understanding of the course material. Provide timely and constructive feedback to help students identify areas of improvement. 6. Industry Visits and Field Trips: Organize visits to IT companies, data centers, or relevant organizations to expose students to real-world IT environments. This provides valuable industry insights and networking opportunities. 7. Online Discussion Forums and Communication Platforms: Establish online discussion forums or communication platforms where students can ask questions, share resources, and engage in discussions outside of the classroom. These strategies promote active learning, practical application of knowledge, and engagement with the subject matter. They cater to different learning styles and encourage students to develop critical thinking, problem-solving, and communication skills necessary for success in this field.

Student Workload (SWL) الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا			
Structured SWL (h/sem) **الحمل الدراسي المنتظم للطالب خلال الفصل**	63	**Structured SWL (h/w)** **الحمل الدراسي المنتظم للطالب أسبوعيا**	6
Unstructured SWL (h/sem) **الحمل الدراسي غير المنتظم للطالب خلال الفصل**	62	**Unstructured SWL (h/w)** **الحمل الدراسي غير المنتظم للطالب أسبوعيا**	6
Total SWL (h/sem) **الحمل الدراسي الكلي للطالب خلال الفصل**	125		

Module Evaluation
تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	Quizzes	2	10% (10)	5 and 10	LO #1, #2 and #10, #11
	Assignments	2	10% (10)	2 and 12	LO #3, #4 and #6, #7
	Projects / Lab.	1	10% (10)	Continuous	All
	Report	1	10% (10)	13	LO #5, #8 and #10
Summative assessment	Midterm Exam	2hr	10% (10)	7	LO #1 - #7
	Final Exam	3hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus)
المنهاج الاسبوعي النظري

	Material Covered
Week 1	• Communication Systems
Week 2	• Network Criteria and Communication Network Types
Week 3	• Protocol Architecture and OSI Model
Week 4	• Physical Layer
Week 5	• Data and Signals
Week 6	• Digital-to-Digital Conversion (Part 1)
Week 7	• Digital-to-Digital Conversion (Part 2)
Week 8	• Analog Transmission
Week 9	• Mid term
Week 10	• Multiplexing
Week 11	• Guided Transmission Media
Week 12	• Unguided Transmission Media
Week 13	• Error Detection and Correction
Week 14	• Wired LANs: Ethernet
Week 15	Week 15: Review

Learning and Teaching Resources مصادر التعلم والتدريس		
	Text	**Available in the Library?**
Required Texts	Data Communication and Networking	
Recommended Texts		
Websites		

Grading Scheme مخطط الدرجات				
Group	**Grade**	التقدير	**Marks %**	**Definition**
Success Group (50 - 100)	**A** - Excellent	امتياز	90 - 100	Outstanding Performance
	B - Very Good	جيد جدا	80 - 89	Above average with some errors
	C - Good	جيد	70 - 79	Sound work with notable errors
	D - Satisfactory	متوسط	60 - 69	Fair but with major shortcomings
	E - Sufficient	مقبول	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX** – Fail	راسب (قيد المعالجة)	(45-49)	More work required but credit awarded
	F – Fail	راسب	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.

MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية					
Module Title	**Probabilities and Statistics**			**Module Delivery**	
Module Type	**SUPPORT**			☒ **Theory** ☒ **Lecture** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**	
Module Code	**NT110**				
ECTS Credits	**5**				
SWL (hr/sem)	**125**				
Module Level		1	**Semester of Delivery**		2
Administering Department		NT	**College**		CSM

Module Leader	Ms. Nada Nazar	**e-mail**	E-mail
Module Leader's Acad. Title	Lecturer	**Module Leader's Qualification**	
Module Tutor	Name (if available)	**e-mail**	E-mail
Peer Reviewer Name	Name	**e-mail**	E-mail
Scientific Committee Approval Date		**Version Number**	1.0

Relation with other Modules

العلاقة مع المواد الدراسية الأخرى

Prerequisite module	None	**Semester**	
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Objectives **أهداف المادة الدراسية**	The Probabilities and statistics basics course for the Networks department aims to achieve the following objectives: 1. Understand the fundamental concepts of probability theory: Students should develop a solid foundation in probability theory, including concepts such as sample spaces, events, probability axioms, conditional probability, and independence. 2. Apply probability concepts to real-world scenarios: Students should be able to apply probability theory to solve problems and analyze real-world situations that involve uncertainty and randomness. This includes calculating probabilities, understanding the concept of expected value, and analyzing random variables. 3. Comprehend statistical concepts and methods: Students should acquire a good understanding of statistical concepts, such as random variables, probability distributions, central tendency, variability, hypothesis testing, confidence intervals, and regression analysis. 4. Perform statistical data analysis: Students should learn how to collect, organize, and analyze data using appropriate statistical techniques. This includes techniques for data exploration, summarization, and inference. 5. Interpret and communicate statistical results: Students should be able to interpret the results of statistical analyses and effectively communicate their findings to others. This involves understanding the limitations of statistical methods and drawing appropriate conclusions from the data. 6. Apply statistical software: Students should gain hands-on experience with statistical software packages commonly used for data analysis, such as R, Python, or SPSS. They should be able to use these tools to perform statistical computations and generate graphical representations of data. 7. Develop critical thinking and problem-solving skills: The course aims to enhance students' critical thinking abilities by challenging them to analyze problems, evaluate evidence, and make informed decisions based on statistical reasoning. 8. Foster a strong mathematical foundation: Probability and Statistics often require a solid understanding of mathematical concepts, so the course aims to strengthen students' mathematical skills, including algebra, calculus, and basic mathematical notation. 9. Prepare for further study in related fields: The course may serve as a prerequisite or provide a foundation for more advanced courses in areas such as machine learning, data science, economics, psychology, or

	engineering, where probabilistic and statistical methods are commonly used.
Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	Upon successful completion of the Probabilities and statistics basics course for the Networks department, students should be able to demonstrate the following learning outcomes: 1. Understand fundamental probability concepts: Students should be able to demonstrate a strong understanding of basic probability concepts, including sample spaces, events, probability axioms, conditional probability, and independence. 2. Apply probability techniques: Students should be able to apply probability techniques to solve problems in various contexts, such as calculating probabilities of events, determining expected values, and understanding concepts like random variables and probability distributions. 3. Analyze statistical data: Students should be able to collect, organize, and analyze data using appropriate statistical methods. This includes understanding descriptive statistics, graphical representations of data, and basic inferential statistics. 4. Interpret statistical results: Students should be able to interpret the results of statistical analyses and draw meaningful conclusions. This involves understanding concepts such as confidence intervals, hypothesis testing, p-values, and statistical significance. 5. Apply statistical software: Students should be proficient in using statistical software packages (e.g., R, Python, SPSS) to perform data analysis and generate graphical representations of data. 6. Critically evaluate statistical claims: Students should be able to critically evaluate statistical claims and arguments presented in various contexts, such as scientific research, news articles, and advertisements. They should be able to identify common fallacies and recognize the importance of sound statistical reasoning. 7. Communicate statistical information: Students should be able to effectively communicate statistical information to both technical and non-technical audiences. This includes presenting findings, using appropriate visualizations, and conveying the limitations and implications of statistical analyses. 8. Apply statistical methods to real-world problems: Students should be able to apply their knowledge of probability and statistics to real-world problems in various fields, such as business, social sciences, engineering, or healthcare. They should be able to identify appropriate statistical methods and apply them to analyze and solve problems. 9. Develop critical thinking and problem-solving skills: The course should

	foster the development of critical thinking skills by engaging students in problem-solving activities that require them to think analytically, reason statistically, and make informed decisions based on data. 10. Prepare for further study or careers: The course should provide a solid foundation for students who wish to pursue further study or careers in fields that require a strong understanding of probability and statistics, such as data science, machine learning, economics, psychology, or research.
Indicative Contents **المحتويات الإرشادية**	The indicative contents of the Probabilities and Statistics basics course for the computer department may include the following topics: 1. Introduction to Probability: o Basic concepts of probability: sample spaces, events, and outcomes. o Probability axioms and properties. o Combinatorics: permutations and combinations. o Conditional probability and independence. 2. Discrete Probability Distributions: o Random variables and probability mass functions. o Common discrete probability distributions: binomial, Poisson, and geometric distributions. o Expected value and variance of discrete random variables. o Joint probability distributions and conditional distributions. 3. Continuous Probability Distributions: o Continuous random variables and probability density functions. o Common continuous probability distributions: uniform, exponential, normal (Gaussian), and gamma distributions. o Expected value and variance of continuous random variables. o Joint probability distributions and conditional distributions. 4. Sampling and Data Description: o Sampling techniques and sampling distributions. o Descriptive statistics: measures of central tendency, measures of dispersion, and graphical representations of data. o Data exploration and visualization. 5. Estimation and Confidence Intervals: o Point estimation: methods for estimating population parameters. o Interval estimation: construction and interpretation of confidence intervals. o Sample size determination for estimation. 6. Hypothesis Testing: o Null and alternative hypotheses. o Test statistics and p-values.

	○ Types of errors and power of tests. ○ Common hypothesis tests: z-tests, t-tests, chi-square tests. 7. Inference for Means and Proportions: ○ Inference for population means: one-sample, independent samples, and paired samples. ○ Inference for population proportions: one-sample and two-sample proportions. 8. Analysis of Variance (ANOVA): ○ One-way ANOVA: comparing means of multiple groups. ○ Post hoc tests and multiple comparisons. ○ Two-way ANOVA: analyzing the effects of two factors. 9. Simple Linear Regression: ○ The simple linear regression model. ○ Least squares estimation and interpretation of coefficients. ○ Assessing model fit and making predictions. 10. Probability and Statistics in Decision Making: ○ Decision theory and utility. ○ Expected value and decision-making under uncertainty. ○ Risk assessment and risk management. 11. Introduction to Bayesian Statistics (optional): ○ Bayesian probability and Bayes' theorem. ○ Prior and posterior distributions. ○ Bayesian inference and decision-making. 12. Introduction to Statistical Software: ○ Hands-on experience with statistical software packages like R, Python, or SPSS. ○ Data manipulation, analysis, and visualization using software tools.

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	Learning and teaching strategies for the Probabilities and statistics basics course for the Network department can include a combination of the following: 1. Active Learning: Encourage active learning by incorporating activities that involve student participation, such as group discussions, problem-solving exercises, case studies, and hands-on data analysis projects. This approach helps students actively engage with the material, apply

concepts, and develop a deeper understanding.

2. Real-World Examples: Use real-world examples and applications to demonstrate the relevance and practicality of probability and statistics. Relating the course content to everyday scenarios, industries, and research fields can enhance students' understanding and motivation.
3. Visual Representations: Utilize visual representations, such as charts, graphs, diagrams, and interactive simulations, to illustrate statistical concepts and relationships. Visual aids can help students visualize abstract concepts, interpret data, and identify patterns more effectively.
4. Technology Integration: Integrate statistical software tools, such as R, Python, or spreadsheet applications, into the course to facilitate data analysis and exploration. This hands-on experience with real-world data and statistical software enhances students' data manipulation and analysis skills.
5. Scaffolding: Break down complex topics into smaller, more manageable subtopics and provide scaffolding support to guide students through the learning process. Start with foundational concepts and gradually introduce more advanced topics, building upon prior knowledge.
6. Formative Assessment: Incorporate formative assessments, such as quizzes, in-class exercises, and homework assignments, to gauge students' understanding and provide feedback. This allows students to identify areas of weakness and reinforces learning throughout the course.
7. Problem-Based Learning: Present students with real-world problems or case studies that require the application of probability and statistical methods. This approach encourages critical thinking, problem-solving skills, and the integration of theoretical knowledge into practical scenarios.
8. Collaborative Learning: Promote collaboration and peer interaction through group activities, discussions, and projects. Working in teams allows students to learn from each other, share perspectives, and develop teamwork and communication skills.
9. Practical Exercises and Experiments: Incorporate practical exercises and experiments that involve collecting and analyzing data. This hands-on approach provides students with firsthand experience in data collection, manipulation, and statistical analysis, reinforcing theoretical concepts.
10. Reflection and Metacognition: Encourage students to reflect on their learning process and develop metacognitive skills. Regularly prompt students to evaluate their understanding, identify areas of improvement, and reflect on their learning strategies.
11. Office Hours and Support: Provide opportunities for individualized

	support, such as office hours or online discussion forums, where students can seek clarification, ask questions, and receive personalized guidance. 12. Engage with Resources: Encourage students to explore additional resources, such as textbooks, online tutorials, academic journals, or educational videos, to deepen their understanding and explore specific topics of interest. By implementing these strategies, instructors can create an engaging and effective learning environment that fosters students' understanding, critical thinking skills, and practical application of probability and statistical concepts.

Student Workload (SWL) الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا			
Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	63	**Structured SWL (h/w)** الحمل الدراسي المنتظم للطالب أسبوعيا	6
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	62	**Unstructured SWL (h/w)** الحمل الدراسي غير المنتظم للطالب أسبوعيا	6
Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	125		

Module Evaluation
تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	Quizzes	2	10% (10)	5 and 10	LO #1, #2 and #10, #11
	Assignments	2	10% (10)	2 and 12	LO #3, #4 and #6, #7
	Projects / Lab.	1	10% (10)	Continuous	All
	Report	1	10% (10)	13	LO #5, #8 and #10
Summative assessment	Midterm Exam	2hr	10% (10)	7	LO #1 - #7
	Final Exam	3hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus) المنهاج الاسبوعي النظري	
	Material Covered
Week 1	Week 1: Introduction to Probabilities and statistics • Basic concepts of probability: sample spaces, events, and outcomes. • Probability axioms and properties. • Combinatorics: permutations and combinations. • Conditional probability and independence.
Week 2	Week 2: Introduction to Probabilities and statistics • Combinatorics: permutations and combinations. • Conditional probability and independence.
Week 3	Week 3: Discrete Probability Distributions • Random variables and probability mass functions. • Common discrete probability distributions: binomial, Poisson, and geometric distributions.
Week 4	Week 4: Discrete Probability Distributions • Expected value and variance of discrete random variables. • Joint probability distributions and conditional distributions.
Week 5	Week 5: Continuous Probability Distributions • Continuous random variables and probability density functions. • Common continuous probability distributions: uniform, exponential, normal (Gaussian), and gamma distributions.
Week 6	Week 6: Continuous Probability Distributions • Expected value and variance of continuous random variables. • Joint probability distributions and conditional distributions.
Week 7	Week 7: Sampling and Data Description • Sampling techniques and sampling distributions. • Descriptive statistics: measures of central tendency, measures of dispersion, and graphical representations of data.
Week 8	Week 8: Sampling and Data Description • Data exploration and visualization.

Week 9	Week 9: Estimation and Confidence Intervals • Point estimation: methods for estimating population parameters. • Interval estimation: construction and interpretation of confidence intervals.
Week 10	Week 10: Estimation and Confidence Intervals • Sample size determination for estimation
Week 11	Week 11: Hypothesis Testing • Null and alternative hypotheses. • Test statistics and p-values.
Week 12	Week 12: Hypothesis Testing • Types of errors and power of tests. • Common hypothesis tests: z-tests, t-tests, chi-square tests.
Week 13	Week 13: Inference for Means and Proportions • Inference for population means: one-sample, independent samples, and paired samples. • Inference for population proportions: one-sample and two-sample proportions.
Week 14	Week 14: Analysis of Variance (ANOVA) • One-way ANOVA: comparing means of multiple groups. • Post hoc tests and multiple comparisons. • Two-way ANOVA: analyzing the effects of two factors.
Week 15	Week 15: Review and Final Projects • Review of key concepts covered throughout the course • Completion of final projects or assignments demonstrating understanding of IT basics
Week 16	**Preparatory week before the final Exam**

Delivery Plan (Weekly Lab. Syllabus) المنهاج الاسبوعي للمختبر	
	Material Covered
Week 1	Introduction to R • Introduction to R environment and RStudio. • Basic R syntax, data types, and objects. • Reading data into R and basic data manipulation.
Week 2	Descriptive Statistics in R 1. Calculating measures of central tendency and dispersion. 2. Creating frequency tables and histograms. 3. Exploratory data analysis with R graphics.
Week 3	Probability Distributions in R 1. Generating random numbers from common probability distributions. 2. Calculating probabilities and percentiles. 3. Plotting probability density functions and cumulative distribution functions.
Week 4	Sampling and Confidence Intervals in R 1. Simple random sampling in R. 2. Estimating population parameters and constructing confidence intervals. 3. Visualizing sampling distributions.
Week 5	Hypothesis Testing in R 1. Performing hypothesis tests for means and proportions. 2. Interpreting p-values and making decisions. 3. Conducting t-tests and chi-square tests in R.
Week 6	Analysis of Variance (ANOVA) in R 1. One-way ANOVA and post hoc tests. 2. Analyzing and interpreting ANOVA results. 3. Visualizing ANOVA data with boxplots and interaction plots.
Week 7	Simple Linear Regression in R • Fitting a simple linear regression model. • Assessing model fit and interpreting coefficients. • Predicting outcomes and evaluating the model.
Week 8	Multiple Linear Regression in R • Extending the simple linear regression to multiple predictors. • Model diagnostics and interpretation of results.

	Handling categorical predictors and interactions
Week 9	Logistic Regression in R • Introduction to logistic regression. • Fitting logistic regression models and interpreting coefficients. • Model assessment and prediction.
Week 10	Time Series Analysis in R • Introduction to time series data. • Time series decomposition and forecasting. • Analyzing and visualizing time series data.
Week 11	: Nonparametric Methods in R • Wilcoxon rank-sum test and Wilcoxon signed-rank test. • Kruskal-Wallis test and Friedman test. • Conducting nonparametric tests in R.
Week 12	Bayesian Statistics in R (optional) • Introduction to Bayesian inference. • Fitting Bayesian models and sampling from posterior distributions. • Interpreting and comparing Bayesian results.
Week 13	: Data Analysis Projects • Students work on data analysis projects applying concepts and techniques learned throughout the course. • Guidance, support, and feedback provided by the instructor during lab sessions.
Week 14	: Data Analysis Projects • Students work on data analysis projects applying concepts and techniques learned throughout the course. • Guidance, support, and feedback provided by the instructor during lab sessions.
Week 15	: Review and Wrap-up • Recap of key concepts and techniques covered throughout the course. • Q&A sessions, review exercises, and additional practice.

Learning and Teaching Resources مصادر التعلم والتدريس		
	Text	**Available in the Library?**
Required Texts		
Recommended Texts		
Websites		

Grading Scheme مخطط الدرجات				
Group	**Grade**	**التقدير**	**Marks %**	**Definition**
Success Group (50 - 100)	**A** - Excellent	**امتياز**	90 – 100	Outstanding Performance
	B - Very Good	**جيد جدا**	80 – 89	Above average with some errors
	C - Good	**جيد**	70 – 79	Sound work with notable errors
	D - Satisfactory	**متوسط**	60 – 69	Fair but with major shortcomings
	E - Sufficient	**مقبول**	50 – 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX** – Fail	**راسب (قيد المعالجة)**	(45-49)	More work required but credit awarded
	F – Fail	**راسب**	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.

MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية		
Module Title	**Arabic Language**	**Module Delivery**
Module Type	**Support**	☒ **Theory** ☒ **Lecture** ☐ **Lab** ☒ **Tutorial** ☐ **Practical** ☐ **Seminar**
Module Code	**UOM101**	
ECTS Credits	**2**	
SWL (hr/sem)	**50**	

Module Level	**1**	**Semester of Delivery**	2
Administering Department	**NT**	**College**	CSM
Module Leader	Dr. Hussam Meshaal	**e-mail**	
Module Leader's Acad. Title	Asst. Prof.	**Module Leader's Qualification**	Ph.D.
Module Tutor		**e-mail**	
Peer Reviewer Name		**e-mail**	
Scientific Committee Approval Date	10/06/2024	**Version Number**	1.0

Relation with other Modules

العلاقة مع المواد الدراسية الأخرى

Prerequisite module	None	**Semester**	
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents
أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية

Module Objectives **أهداف المادة الدراسية**	التعرف على الكلام العربي: من ناحية تعريفة، اقسامة، الى علامات كل قسم منه. معرفة الجملة العربية واقسام الجملة العربية والجمل الاسمية والجمل الفعلية التعرف على حركات الاعراب: سواء كانت اصلية او فرعية معرفة الطالب بالعفل العربي: من حيث الصحة والاعلال معرفة الطالب الفعل العربي من حيث اللزوم والتعدي معرفة لطالب الفعل العربي من حيث الزمن طرق كتابة العدد و تذكرة وتانيثه معرفة علامات الترقيم في الكلام تعلم قواعد رسم الهمزة - التعرف على طريقة كتابة التاء المربوطة، والمبسوطة - قل ولا تقل: الأخطاء الشائعة لدى المتكلمين والكتاب - معرفة ماهو الأسلوب الخبري، 13- معرفة ماهو الأسلوب الانشائي، 14- تعلم مهارات لغوية: تنمية الذوق اللغوي، وتحسين الأسلوب لدى المتعلمين
Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	1- ان يعرف الطالب الكلام العربي: من ناحية تعريفة، اقسامة، الى علامات كل قسم منه. 2- ان يتعلم الطالب الجملة العربية واقسام الجملة العربية والجمل الاسمية والجمل الفعلية 3- التعرف على حركات الاعراب: سواء كانت اصلية او فرعية 4- ان يعرف الطالب العفل العربي: من حيث الصحة والاعلال 5- ان يتعلم الطالب الفعل العربي من حيث اللزوم والتعدي 6- معرفة الطالب الفعل العربي من حيث الزمن 7- معرف الطالب طرق كتابة العدد و تذكرة وتانيثه 8- معرفة الطالب لعلامات الترقيم في الكلام 9- ان يتعلم الطالب قواعد رسم الهمزة 10- معرف الطالب على طريقة كتابة التاء المربوطة، والمبسوطة 11- قل ولا تقل: الأخطاء الشائعة لدى المتكلمين والكتاب 12- التعرف على الأسلوب الخبري، 13- معرفة ماهو الأسلوب الانشائي، 14- التعلم على مهارات لغوية: تنمية الذوق اللغوي، وتحسين الأسلوب لدى المتعلمين
Indicative Contents **المحتويات الإرشادية**	1-التعرف على الكلام العربي: من ناحية تعريفة، اقسامة، الى علامات كل قسم منه[ساعه 2] 2-معرفة الجملة العربية واقسام الجملة العربية والجمل الاسمية والجمل الفعلية، ساعه 2 3-التعرف على حركات الاعراب: سواء كانت اصلية او فرعية، ساعه 2 4-معرفة الطالب بالعفل العربي: من حيث الصحة والاعلال، ساعه 2 5-معرفة الطالب الفعل العربي من حيث اللزوم والتعدي، ساعه 2 6-معرفة لطالب الفعل العربي من حيث الزمن، ساعه 2 7-طرق كتابة العدد و تذكرة وتانيثه، ساعه 2 8-معرفة علامات الترقيم في الكلام، ساعه 2 9-تعلم قواعد رسم الهمزة، ساعه 2 10- التعرف على طريقة كتابة التاء المربوطة، والمبسوطة، ساعه 2 11- قل ولا تقل: الأخطاء الشائعة لدى المتكلمين والكتاب ، ساعه 2 12- معرفة ماهو الأسلوب الخبري، ساعه 2 13- معرفة ماهو الأسلوب الانشائي، ساعه 2 14- التعلم مهارات لغوية: تنمية الذوق اللغوي، وتحسين الأسلوب لدى المتعلمين،

	ساعه 2

Learning and Teaching Strategies
استراتيجيات التعلم والتعليم

Strategies	الإستراتيجية الرئيسية التي سيتم تبنيها في تقديم هذه الوحدة هي تشجيع الطلاب على المشاركة على المشاركه في الكلام الفربي وكتابته بالصورة الصحيحه ، مع تحسين مهارات التفكير النقدي وتوسيعها في نفس الوقت. سيتم تحقيق ذلك من خلال الفصول والبرامج التعليمية التفاعلية ومن خلال النظر في أنواع التجارب البسيطة التي تتضمن بعض أنشطة أخذ العينات التي تهم الطلاب.

Student Workload (SWL)
الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا

Structured SWL (h/sem) **الحمل الدراسي المنتظم للطالب خلال الفصل**	32	**Structured SWL (h/w)** **الحمل الدراسي المنتظم للطالب أسبوعيا**	1
Unstructured SWL (h/sem) **الحمل الدراسي غير المنتظم للطالب خلال الفصل**	18	**Unstructured SWL (h/w)** **الحمل الدراسي غير المنتظم للطالب أسبوعيا**	1
Total SWL (h/sem) **الحمل الدراسي الكلي للطالب خلال الفصل**	50		

Module Evaluation
تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	Quizzes	3	15% (15)	5 and 10	LO #1, #2 and #10, #11
	Assignments	3	15% (15)	2 and 12	LO #3, #4 and #6, #7
	Projects / Lab.				
	Report	1	10% (10)	13	LO #5, #8 and #10
Summative assessment	Midterm Exam	2hr	10% (10)	7	LO #1 - #7
	Final Exam	3hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus) المنهاج الاسبوعي النظري	
	Material Covered
Week 1	الكلام العربي: تعريفة، اقسامة، وعلامات كل قسم.
Week 2	الجملة العربية: تعريفها ، اقسامها : الاسمية والفعلية
Week 3	حركات الاعراب: اصلية، فرعية
Week 4	العفل العربي: من حيث الصحة والاعلال
Week 5	الفعل العربي من حيث اللزوم والتعدي
Week 6	الفعل العربي من حيث الزم
Week 7	امتحان
Week 8	العدد: تذكرة، وتانيثه
Week 9	علامات الترقيم في الكلام
Week 10	قواعد رسم الهمزة
Week 11	التاء المربوطة، والمبسوطة
Week 12	قل ولا تقل: الأخطاء الشائعة لدى المتكلمين والكتاب
Week 13	الأسلوب الخبري،
Week 14	والأسلوب الإنشائي
Week 15	**مهارات لغوية: تنمية الذوق اللغوي، وتحسين الأسلوب لدى المتعلمين**
Week 16	امتحان نهاية الفصل

Learning and Teaching Resources مصادر التعلم والتدريس		
	Text	**Available in the Library?**
Required Texts	جامع الدروس العربية: الشيخ مصطفى الغلاييني	No
Recommended Texts	الجملة العربية: تأليفها وأقسامها د. فاضل السامرائي	No
Websites	https://www.almrsal.com/post/923401	

Grading Scheme مخطط الدرجات				
Group	**Grade**	**التقدير**	**Marks %**	**Definition**
Success Group (50 - 100)	**A** - Excellent	**امتياز**	90 - 100	Outstanding Performance
	B - Very Good	**جيد جدا**	80 - 89	Above average with some errors
	C - Good	**جيد**	70 - 79	Sound work with notable errors
	D - Satisfactory	**متوسط**	60 - 69	Fair but with major shortcomings
	E - Sufficient	**مقبول**	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX** – Fail	**راسب (قيد المعالجة)**	(45-49)	More work required but credit awarded
	F – Fail	**راسب**	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.

MODULE DESCRIPTION FORM
نموذج وصف المادة الدراسية

Module Information
معلومات المادة الدراسية

Module Title	**Object Oriented Programming**	**Module Delivery**	
Module Type	**Core**	☒ **Theory** ☒ **Lecture** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**	
Module Code	**NT201**		
ECTS Credits	**8**		
SWL (hr/sem)	**200**		

Module Level	1	**Semester of Delivery**	1
Administering Department	NT	**College**	CSM
Module Leader	Dr. Firas Mohamad Salih	**e-mail**	E-mail
Module Leader's Acad. Title	Lecturer	**Module Leader's Qualification**	Ph.D.
Module Tutor	Name (if available)	**e-mail**	E-mail
Peer Reviewer Name	Name	**e-mail**	E-mail
Scientific Committee Approval Date	10/6/2024	**Version Number**	1.0

Relation with other Modules العلاقة مع المواد الدراسية الأخرى			
Prerequisite module	Problems Solving & Programming 2	**Semester**	NT107
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Objectives **أهداف المادة الدراسية**	1. Advanced OOP Concepts: • Understand and apply advanced object-oriented programming concepts, such as inheritance, polymorphism, and encapsulation. • Design and implement complex class hierarchies using inheritance and composition. • Utilize advanced OOP techniques to develop modular and reusable code. 2. Design Patterns and Software Architecture: • Explore commonly used design patterns and their application in software development. • Understand architectural patterns and their role in designing scalable and maintainable software systems. • Apply design patterns and software architecture principles to solve real-world problems. 3. Data Structures and Algorithms: • Understand advanced data structures, such as trees, graphs, and hash tables, and their implementation in Python. • Analyze algorithmic complexity and choose appropriate algorithms for different problem-solving scenarios. • Implement and optimize algorithms for searching, sorting, and graph traversal. 4. Exception Handling and Error Management: • Master advanced exception handling techniques, such as exception chaining and custom exception classes. • Understand and apply defensive programming techniques to handle errors and unexpected situations. • Develop error handling strategies for robust and reliable software. 5. Concurrency and Parallel Programming: • Understand the challenges and principles of concurrent programming. • Utilize multithreading and multiprocessing to write concurrent Python programs. • Implement synchronization mechanisms and handle race conditions in concurrent code. 6. Testing and Debugging: • Learn advanced techniques for testing Python code, including unit testing, integration testing, and test-driven development (TDD). • Apply debugging strategies and tools to identify and fix software

	defects. • Develop a comprehensive testing and debugging mindset for producing high-quality code. 7. GUI Development and User Experience: • Gain proficiency in developing graphical user interfaces (GUIs) using Python frameworks such as Tkinter, PyQt, or wxPython. • Understand user experience (UX) principles and design intuitive and user-friendly interfaces. • Incorporate event-driven programming to handle user interactions in GUI applications. 8. Software Development Best Practices: • Learn and apply software development best practices, including code organization, documentation, and version control. • Collaborate effectively in software development teams using version control systems like Git. • Understand the importance of code maintainability, scalability, and reusability.
Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	1. Advanced Understanding of OOP Concepts: • Demonstrate a deep understanding of advanced object-oriented programming concepts, including inheritance, polymorphism, and encapsulation. • Apply advanced OOP techniques to design and implement complex software systems. • Analyze and evaluate different approaches to solve programming problems using OOP principles. 2. Design Patterns and Software Architecture: • Apply various design patterns to solve software design problems effectively. • Design software architectures that are modular, maintainable, and scalable. • Analyze and evaluate different software architectural patterns for different types of applications. 3. Proficiency in Data Structures and Algorithms: • Implement and analyze advanced data structures, such as trees, graphs, and hash tables, using Python. • Design and optimize algorithms for efficient data manipulation and problem-solving. • Apply algorithmic thinking and problem-solving skills to solve complex programming challenges. 4. Robust Exception Handling and Error Management: • Implement advanced exception handling techniques to handle errors and exceptional situations in software. • Design error handling strategies to ensure robustness and reliability of software systems. • Analyze and debug complex software issues related to error management and exception handling. 5. Concurrent and Parallel Programming: • Demonstrate an understanding of concurrent programming principles and techniques. • Utilize multithreading and multiprocessing in Python to write concurrent and parallel programs. • Implement synchronization mechanisms and handle concurrency-

	related issues. 6. Effective Testing and Debugging: • Apply advanced testing techniques, such as unit testing, integration testing, and test-driven development (TDD), to ensure software quality. • Use debugging tools and strategies to identify and resolve complex software defects. • Develop a systematic approach to testing and debugging software systems. 7. Advanced GUI Development and User Experience: • Design and develop sophisticated graphical user interfaces (GUIs) using Python frameworks such as Tkinter, PyQt, or wxPython. • Apply user experience (UX) principles to create intuitive and user-friendly interfaces. • Implement event-driven programming to handle user interactions and enhance user experience. 8. Application of Software Development Best Practices: • Apply software development best practices, including code organization, documentation, and version control, to develop high-quality software. • Collaborate effectively in software development teams, demonstrating good teamwork and communication skills. • Demonstrate an understanding of the importance of code maintainability, scalability, and reusability. 9. Real-World Application Development: • Design, implement, and test larger-scale software projects using object-oriented programming principles. • Apply software engineering principles and techniques to manage project scope, requirements, and timelines. • Demonstrate proficiency in project planning, teamwork, and project management.
Indicative Contents **المحتويات الإرشادية**	1. Advanced OOP Concepts: • Inheritance: Advanced inheritance concepts such as multiple inheritance, method resolution order (MRO), and mixins. • Polymorphism: Advanced polymorphism techniques including method overriding, abstract base classes (ABCs), and interfaces. • Composition: Utilizing composition over inheritance to design and implement complex class relationships. 2. Design Patterns and Software Architecture: • Introduction to design patterns: Understanding and applying common design patterns such as Singleton, Factory, Observer, and Strategy. • Software architecture principles: Exploring architectural patterns like Model-View-Controller (MVC) and understanding their application in software development. • Component-based architecture: Designing and implementing software using component-based architectural patterns. 3. Data Structures and Algorithms: • Advanced data structures: Implementation and application of advanced data structures like balanced search trees, heaps, and graphs. • Algorithm analysis: Analyzing the time and space complexity of algorithms and choosing the appropriate algorithmic solutions for different problem domains. • Sorting and searching algorithms: Implementing and analyzing various sorting and searching algorithms, including quicksort, mergesort, binary

search, and more.

4. Exception Handling and Error Management:
- Advanced exception handling: Handling and propagating exceptions, exception chaining, and creating custom exception classes.
- Defensive programming: Implementing defensive programming techniques to handle errors and edge cases in software.
- Error management strategies: Designing error handling strategies to ensure fault tolerance and reliability in software systems.

5. Concurrency and Parallel Programming:
- Introduction to concurrency: Understanding the challenges and principles of concurrent programming.
- Threading and multiprocessing: Implementing multithreading and multiprocessing techniques in Python for concurrent and parallel programming.
- Synchronization and coordination: Utilizing synchronization mechanisms like locks, semaphores, and condition variables to handle shared resources and coordinate concurrent tasks.

6. Testing and Debugging:
- Advanced testing techniques: Implementing unit tests, integration tests, and test-driven development (TDD) approaches for robust software testing.
- Debugging strategies: Applying advanced debugging techniques and tools to identify and fix software defects.
- Test coverage and code quality: Understanding the importance of code coverage and maintaining high code quality through testing and debugging.

7. GUI Development and User Experience:
- GUI frameworks: Exploring advanced GUI frameworks in Python, such as PyQt, wxPython, or Kivy, for developing interactive graphical user interfaces.
- User experience design: Incorporating user-centered design principles to create intuitive and visually appealing user interfaces.
- Event-driven programming: Utilizing event-driven programming to handle user interactions and create responsive GUI applications.

8. Software Development Best Practices:
- Code organization and modularity: Applying modular design principles and organizing code into reusable and maintainable components.
- Documentation and commenting: Writing clear and comprehensive documentation and comments to enhance code readability and understandability.
- Version control: Utilizing version control systems, such as Git, for collaborative software development and code management.

9. Real-World Application Development:
- Large-scale project development: Working on larger-scale projects that involve designing, implementing, and testing complex software systems.
- Project planning and management: Understanding project management methodologies and applying them to effectively plan and manage software development projects.
- Team collaboration: Collaborating with peers in a team environment, demonstrating effective communication and teamwork skills.

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	1. Active Learning and Problem-Solving Approach: • Emphasize active learning strategies, such as hands-on coding exercises, group discussions, and problem-solving activities. • Encourage students to apply their knowledge of OOP principles and Python programming to real-world scenarios and projects. • Provide opportunities for students to actively engage with the material through coding challenges, case studies, and practical assignments. 2. Project-Based Learning: • Implement a project-based approach where students work on larger-scale programming projects that require the application of advanced OOP concepts. • Assign projects that involve designing, implementing, and testing software systems using Python and OOP principles. • Encourage students to collaborate in teams, simulate real-world development environments, and manage project requirements and deadlines. 3. Practical Coding Exercises and Assignments: • Provide a variety of coding exercises and assignments that focus on advanced OOP topics, design patterns, algorithms, and software architecture. • Include programming assignments that require students to implement complex OOP concepts, solve algorithmic problems, and design efficient data structures. • Provide feedback and guidance on students' code to promote good programming practices and enhance their understanding of OOP in Python. 4. Code Reviews and Peer Collaboration: • Incorporate code review sessions where students review and provide feedback on each other's code, promoting code quality and best practices. • Encourage peer collaboration and teamwork, fostering communication and problem-solving skills in a professional software development context. • Utilize version control systems (e.g., Git) to facilitate code sharing, collaboration, and tracking of project development. 5. Integration of Design Patterns and Software Architecture: • Introduce design patterns and software architecture principles in a practical context, demonstrating their relevance and benefits in software development. • Guide students to identify and apply appropriate design patterns and architectural patterns in their projects. • Discuss case studies and examples of real-world applications where design patterns and software architecture have been effectively used. 6. Practical Testing and Debugging Techniques: • Teach advanced testing techniques, such as unit testing, integration testing, and test-driven development (TDD), to ensure software quality and reliability.

	• Emphasize the importance of systematic debugging approaches and the use of debugging tools to identify and resolve software defects. • Provide practical examples and exercises that require students to write comprehensive test cases and debug complex code scenarios. 7. Practical Application of GUI Development: • Provide practical assignments and projects that require students to develop GUI applications using Python frameworks like Tkinter, PyQt, or wxPython. • Incorporate user experience (UX) principles and usability testing to enhance the design and functionality of GUI applications. • Encourage students to incorporate event-driven programming concepts to handle user interactions and create interactive interfaces. 8. Exposure to Real-World Software Development Practices: • Introduce students to software development best practices, including code organization, documentation, and version control. • Familiarize students with collaborative software development tools and techniques, such as code repositories and issue tracking systems. • Discuss industry trends, emerging technologies, and the importance of continuous learning in the field of object-oriented programming.

Delivery Plan (Weekly Syllabus)

المنهاج الاسبوعي النظري

	Material Covered
Week 1	Introduction to Object Oriented Programing and Structural Programming
Week 2-3	Introduction to Classes and Objects
Week 4	Encapsulation and Access modifiers
Week 5	Abstraction
Week 6-7	Inheritance
Week 8-9	Polymorphism
Week 10	Mid Term Examination
Week 11	Operator Overloading
Week 12	Operator Overriding
Week 13-15	Project and Presentation

Delivery Plan (Weekly Lab. Syllabus)

المنهاج الاسبوعي للمختبر

	Material Covered
Week 1	Week 1: Introduction to Object-Oriented Programming and Structural Programming • Overview of the principles and concepts of object-oriented programming (OOP) • Introduction to the fundamentals of structural programming • Discussion on the advantages and characteristics of OOP

Week 2-3	Week 2 - 3: Introduction to Classes and Objects • Understanding the concept of classes and objects in OOP • Creating and defining classes in C++ • Exploring object instantiation and member functions
Week 4	Week 4: Encapsulation and Access Modifiers • Understanding encapsulation and its importance in OOP • Exploring access modifiers (public, private, protected) • Discussion on data hiding and encapsulation principles
Week 5	Week 5: Abstraction • Introduction to abstraction in OOP • Understanding abstract classes and interfaces • Implementing abstraction in C++ using pure virtual functions
Week 6-7	Week 6-7: Inheritance • Exploring the concept of inheritance in OOP • Implementing inheritance in C++ through derived classes • Discussing the different types of inheritance (single, multiple, multilevel, hierarchical)
Week 8-9	Week 8-9: Polymorphism • Understanding polymorphism and its significance in OOP • Exploring function overloading and overriding • Implementing polymorphism through virtual functions in C++
Week 10	Week 10: Midterm Examination • Midterm examination covering topics from weeks 1-9 • Review of previous topics and discussion of any questions or concerns
Week 11	Week 11: Operator Overloading • Introduction to operator overloading in C++ • Overloading unary and binary operators • Exploring the use of friend functions for operator overloading
Week 12	Week 12: Operator Overriding • Understanding the concept of operator overriding

	• Overriding base class operators in derived classes • Discussion on the limitations and best practices of operator overriding
Week 13-15	Week 13-15: Project and Presentation • Working on a project that incorporates the principles and concepts covered in the course • Planning, designing, and implementing an object-oriented program • Preparing a presentation to showcase the project and its features

Learning and Teaching Resources **مصادر التعلم والتدريس**		
	Text	**Available in the Library?**
Required Texts	C++ Primer (5th Edition) 5th Edition , by Stanley Lippman (Author), Josée Lajoie (Author), Barbara Moo (Author)	
Recommended Texts		
Websites		

MODULE DESCRIPTION FORM
نموذج وصف المادة الدراسية

Module Information **معلومات المادة الدراسية**		
Module Title	Website Development I	**Module Delivery**
Module Type	**Core**	☒ **Theory** ☒ **Lecture** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**
Module Code	**NT202**	
ECTS Credits	**6**	
SWL (hr/sem)	**150**	

Module Level	1	**Semester of Delivery**	1
Administering Department	NT	**College**	CSM
Module Leader	Zaid Dawood Salem	**e-mail**	E-mail
Module Leader's Acad. Title	Asst. Lecturer	**Module Leader's Qualification**	Ph.D.
Module Tutor	Name (if available)	**e-mail**	E-mail
Peer Reviewer Name	Name	**e-mail**	E-mail
Scientific Committee Approval Date	10/6/2024	**Version Number**	1.0

Relation with other Modules

العلاقة مع المواد الدراسية الأخرى

Prerequisite module	Problems Solving & Programming 2	**Semester**	NT107
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents

أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية

Module Objectives **أهداف المادة الدراسية**	1. Understanding Web Technologies: To provide students with a foundational understanding of web technologies, including HTML, CSS, and JavaScript, and their role in building websites and web applications. 2. Website Structure and Design: To teach students how to create well-structured and visually appealing websites using HTML and CSS, focusing on concepts such as layout, typography, color schemes, and responsive design. 3. Interactive Web Elements: To enable students to incorporate interactive elements into websites using JavaScript, such as form validation, event handling, and dynamic content manipulation. 4. Client-Server Communication: To introduce students to the basics of client-server communication in web development, including sending and receiving data from a server using HTTP requests and APIs. 5. Web Development Tools: To familiarize students with popular web

	development tools, such as text editors, version control systems, and debugging tools, and teach them how to use these tools effectively in their development workflow. 6. Web Accessibility: To emphasize the importance of web accessibility and teach students how to design and develop websites that are inclusive and accessible to users with disabilities. 7. Project Development: To provide students with hands-on experience in developing a complete web project, from conceptualization to deployment, while applying the concepts and techniques learned throughout the course. 8. Collaboration and Communication: To promote teamwork and effective communication skills by encouraging students to work collaboratively on group projects, communicate project requirements and progress, and provide constructive feedback to their peers. 9. Problem-Solving and Debugging: To enhance students' problem-solving skills and teach them how to debug and troubleshoot common issues in web development, fostering a systematic and analytical approach to solving technical problems. 10. Professionalism and Ethical Considerations: To instill professional ethics and good practices in web development, including copyright and intellectual property considerations, respecting user privacy, and adhering to industry standards and best practices.
Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	• Knowledge of Web Technologies: Gain a solid understanding of web technologies, including HTML, CSS, and JavaScript, and their role in web development. • Website Creation: Design and develop well-structured websites using HTML and CSS, considering factors such as layout, typography, color schemes, and responsive design. • Interactive Elements: Implement interactive features on websites using JavaScript, such as form validation, event handling, and dynamic content manipulation. • Client-Server Communication: Understand the basics of client-server communication in web development, including making HTTP requests and working with APIs to retrieve and send data. • Use of Web Development Tools: Utilize popular web development tools, such as text editors, version control systems, and debugging tools, to enhance productivity and efficiency in web development projects. • Web Accessibility: Apply principles of web accessibility to ensure websites are inclusive and accessible to users with disabilities, following accessibility guidelines and best practices.

	• Project Development: Develop a complete web project, applying the knowledge and skills acquired throughout the course, from planning and design to implementation and deployment. • Collaboration and Communication: Collaborate effectively with team members, communicate project requirements and progress, and provide constructive feedback to peers in group projects. • Problem-Solving and Debugging: Demonstrate problem-solving skills and the ability to debug and troubleshoot issues in web development projects, using a systematic and analytical approach. • Professionalism and Ethical Considerations: Understand and adhere to professional ethics and considerations in web development, including respecting copyright and intellectual property, protecting user privacy, and following industry standards and best practices
Indicative Contents **المحتويات الإرشادية**	The indicative contents for Web Development 1 may include: 1. Introduction to Web Technologies: • Overview of web development concepts and technologies • Understanding the client-server architecture and how the web works 2. HTML Fundamentals: • HTML syntax and structure • Working with tags, attributes, and elements • Creating hyperlinks, lists, tables, and forms 3. CSS Basics: • Introduction to Cascading Style Sheets (CSS) • Applying styles to HTML elements • Working with selectors, properties, and values • Managing layout, typography, and colors 4. JavaScript Fundamentals: • Introduction to JavaScript programming language • Variables, data types, and operators • Conditional statements and loops • Functions and event handling 5. Responsive Web Design: • Designing websites that adapt to different screen sizes and devices • Using media queries and viewport meta tags • Implementing responsive layouts and navigation menus

	6. Web Accessibility: • Understanding the importance of web accessibility • Applying accessibility principles and techniques • Testing and optimizing websites for accessibility 7. Introduction to Server-side Technologies: • Overview of server-side programming languages and frameworks • Introduction to databases and server-side scripting • Basics of server-side development and interacting with databases 8. Introduction to Version Control: • Understanding the concept of version control and its importance in web development • Using Git for version control and collaboration • Branching, merging, and resolving conflicts 9. Web Project Development: • Planning and organizing a web development project • Creating wireframes and mockups • Implementing the project using HTML, CSS, and JavaScript 10. Deployment and Maintenance: • Uploading and hosting a website on a server • Performing maintenance tasks and updates • Testing and troubleshooting common issues Please note that the above contents are indicative and may vary depending on the specific curriculum and institution offering the course

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	1. Lectures: In-class lectures can be used to introduce and explain key concepts, programming languages, and techniques related to web development. The instructor can provide examples and demonstrations to illustrate the concepts. 2. Hands-on Practice: Hands-on practice is essential for web development. Students can engage in practical exercises and coding activities during the class or in dedicated lab sessions. This allows them to apply the knowledge gained and practice coding HTML, CSS, and JavaScript. 3. Project-based Learning: Assigning small projects or tasks related to web development can provide students with real-world scenarios to apply

	their skills and knowledge. Working on projects helps them develop problem-solving abilities, collaboration skills, and practical experience in building websites. 4. Group Discussions and Peer Learning: Encouraging group discussions and peer learning can enhance understanding and knowledge retention. Students can discuss challenges, exchange ideas, and collaborate on problem-solving. This fosters a collaborative learning environment and allows students to learn from each other's experiences. 5. Online Resources and Tutorials: Providing students with online resources, tutorials, and documentation can supplement classroom learning. These resources can include video tutorials, coding exercises, interactive websites, and documentation of programming languages and frameworks. 6. Code Reviews and Feedback: Conducting code reviews and providing feedback on student projects or assignments can help improve their coding skills. Feedback can be provided by the instructor or through peer code reviews. This helps students understand best practices, identify areas for improvement, and learn from their mistakes. 7. Guest Speakers and Industry Insights: Inviting guest speakers from the industry or web development professionals can provide valuable insights and real-world experiences to students. They can share their expertise, industry trends, and challenges in web development, inspiring students and bridging the gap between academia and industry. 8. Assessment and Evaluation: Assessments can include quizzes, assignments, projects, and exams to evaluate students' understanding and progress. This allows the instructor to gauge their knowledge and provide constructive feedback for improvement. 9. Continuous Learning and Updates: Web development is a rapidly evolving field. It is important to emphasize the need for continuous learning and staying updated with the latest technologies, frameworks, and best practices. Encouraging students to explore online resources, attend workshops, and engage in self-directed learning can help them keep up with industry trends. It is worth noting that the selection and implementation of these strategies may vary based on the specific educational institution, class size, resources available, and the preferences of the instructor.

Delivery Plan (Weekly Syllabus)

المنهاج الاسبوعي النظري

	Material Covered
Week 1	Introduction to Web Development • Introduction to HTML and its structure

	• Creating a basic HTML webpage • Understanding CSS and styling web pages
Week 2-3	HTML and CSS Fundamentals • Working with text, links, images, and lists in HTML • Applying CSS styles to HTML elements • Introduction to responsive design principles
Week 4-5	Web Design Principles • Understanding color theory and typography in web design • Creating layouts using CSS positioning and flexbox • Introduction to CSS frameworks like Bootstrap
Week 6-7	Multimedia and Forms • Adding images, videos, and audio to web pages • Creating forms and handling user input • Validating form data using HTML5 and JavaScript
Week 8-9	: Introduction to JavaScript • Fundamentals of JavaScript programming • Working with variables, data types, and operators • Writing JavaScript functions and control structures
Week 10-11	JavaScript DOM Manipulation • Accessing and modifying HTML elements using JavaScript • Handling events and creating interactive web pages • Introduction to JavaScript libraries (e.g., jQuery) •
Week 12-13	: Introduction to Front-End Frameworks • Exploring popular front-end frameworks like React or Vue.js • Building dynamic and interactive web pages using frameworks • Understanding component-based development
Week 14-15	Responsive Design and Deployment • Designing responsive websites for different devices and screen sizes • Optimizing web pages for performance • Deploying a website to a web server •

Delivery Plan (Weekly Lab. Syllabus) المنهاج الاسبوعي للمختبر	
	Material Covered
Week 1	: Introduction to HTML • Creating a basic HTML page • Adding headings, paragraphs, and lists • Working with links and images
Week 2	CSS Styling • Applying CSS styles to HTML elements • Creating and styling navigation menus • Implementing CSS layouts and positioning
Week 3	Forms and Input Validation • Creating HTML forms with various input types • Implementing client-side form validation using JavaScript • Handling form submission and processing user input
Week 4	Responsive Web Design • Designing a responsive web layout using CSS media queries • Adapting the website for different screen sizes and devices • Testing and optimizing the website for mobile devices
Week 5	Introduction to JavaScript 4. Writing basic JavaScript code 5. Manipulating the DOM using JavaScript 6. Implementing simple interactive features on a web page
Week 6	: JavaScript Functions and Events • Defining and calling JavaScript functions • Handling different types of events (e.g., click, mouseover) • Implementing event handlers and callback functions
Week 7	Introduction to Bootstrap Framework • Using Bootstrap CSS classes and components for rapid web development • Building responsive and mobile-friendly web layouts with Bootstrap • Customizing and extending Bootstrap components

Week 8	JavaScript Libraries and Frameworks • Introduction to popular JavaScript libraries (e.g., jQuery) • Exploring the features and functionalities of JavaScript frameworks (e.g., Angular, React) • Building interactive web applications using libraries and frameworks
Week 9	Working with APIs and JSON • Making AJAX requests to retrieve data from external APIs • Parsing and manipulating JSON data • Displaying API data on a web page
Week 10	: Introduction to Server-Side Development • Setting up a local development environment (e.g., Apache, PHP) • Writing server-side scripts using PHP • Implementing basic server-side functionality (e.g., form handling, database connection)
Week 11-14	11-14: Web Project Development • Applying the concepts learned to develop a complete web project • Planning, designing, and implementing a website or web application • Testing, debugging, and optimizing the web project
Week 15	Project Presentation and Evaluation

Learning and Teaching Resources

مصادر التعلم والتدريس

	Text	**Available in the Library?**
Required Texts		No
Recommended Texts		
Websites		

MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية		
Module Title	**Network Protocol -1-**	**Module Delivery**

Module Information معلومات المادة الدراسية		
Module Title	**Network Protocol -1-**	**Module Delivery**
Module Type	**Basic**	☒ **Theory** ☒ **Lecture** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**
Module Code	**NT203**	
ECTS Credits	**4**	
SWL (hr/sem)	**100**	

Module Level	1	**Semester of Delivery**	1
Administering Department	NT	**College**	CSM
Module Leader	Omar Tariq Salih	**e-mail**	E-mail
Module Leader's Acad. Title	Lecturer	**Module Leader's Qualification**	Ph.D.
Module Tutor	Name (if available)	**e-mail**	E-mail
Peer Reviewer Name	Name	**e-mail**	E-mail
Scientific Committee Approval Date	10/6/2024	**Version Number**	1.0

Relation with other Modules العلاقة مع المواد الدراسية الأخرى			
Prerequisite module	Data Communication & Networking	**Semester**	NT109
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Objectives **أهداف المادة الدراسية**	In this course, Networks department aims to achieve the following objectives: 1. Understand the fundamental concepts and principles of the TCP/IP protocol suite. 2. Explain the layered structure of the TCP/IP model and the functions of each layer. 3. Identify and describe the key protocols and components of the TCP/IP architecture. 4. Demonstrate knowledge of network infrastructures, including different topologies and connecting devices. 5. Perform IP address calculations using classful and classless addressing techniques. 6. Apply subnetting and supernetting methods to efficiently allocate IP addresses. 7. Understand the process of data delivery and routing in TCP/IP networks. 8. Configure and troubleshoot basic network connectivity using TCP/IP protocols. 9. Analyze network connectivity issues and apply appropriate solutions. 10. Apply critical thinking and problem-solving skills to address real-world networking scenarios.
Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	Upon successful completion of this course for the Networks department, students should be able to demonstrate the following learning outcomes: 1. Comprehensive Understanding: • Demonstrate a comprehensive understanding of the TCP/IP protocol suite and its components. • Explain the functions and responsibilities of each layer in the TCP/IP model. 2. IP Addressing and Subnetting: • Apply classful and classless addressing techniques for IP address allocation. • Perform subnetting and supernetting calculations to efficiently manage network segments.

	3. Network Infrastructure and Connectivity: • Identify network topologies and understand the role of connecting devices. • Configure and troubleshoot basic network connectivity using TCP/IP protocols. 4. Data Delivery and Routing: • Explain the process of data delivery, including packet encapsulation, routing, and forwarding. • Analyze and troubleshoot network connectivity issues using routing protocols. 5. Critical Thinking and Problem-Solving: • Apply critical thinking skills to solve real-world networking scenarios. • Evaluate and select appropriate IP addressing strategies based on network requirements. 6. Communication and Collaboration: • Communicate effectively about TCP/IP concepts and network-related issues. • Collaborate with peers to solve problems and share knowledge. 7. Practical Application: • Apply acquired knowledge to design, configure, and troubleshoot TCP/IP networks. • Implement effective IP addressing schemes and network segmentation strategies. 8. Lifelong Learning: • Recognize the importance of continuous learning in the field of networking. • Stay updated with new protocols and technologies related to TCP/IP networking.
Indicative Contents **المحتويات الإرشادية**	The indicative contents of this course for the computer department may include the following topics: 1. Internet Introduction • History and evolution of the Internet • Key milestones and developments • Internet architecture and key protocols • Internet governance and organizations 2. TCP/IP Protocol Suite Layers • Introduction to the TCP/IP protocol suite • Overview of the TCP/IP layers (Application, Transport, Internet, Link) • Functions and responsibilities of each layer • Protocols and services associated with each layer 3. Infrastructure Network and Connecting Devices

	• Overview of network infrastructures • Network topologies: bus, star, ring, mesh, etc. • Common connecting devices: switches, routers, hubs • Benefits and limitations of different network infrastructures 4. Classful Addressing and Special Addressing • Introduction to classful addressing • IP address classes and ranges (Class A, B, C, D, E) • Special IP addresses: loopback, broadcast, multicast • Address exhaustion issues with classful addressing 5. Subnetting and Supernetting • Introduction to subnetting and its need • Subnet masks and subnet addressing • Calculating subnet addresses and broadcast addresses • Supernetting and route aggregation for efficient addressing 6. Classless Addressing • Introduction to classless addressing • Classless Inter-Domain Routing (CIDR) • Variable Length Subnet Masking (VLSM) • Address allocation and route summarization in classless addressing 7. Delivery and Routing • Data encapsulation and decapsulation process • Overview of packet delivery and routing • Static routing and dynamic routing protocols (RIP, OSPF) • Routing table configuration and routing decision process

Learning and Teaching Strategies **استراتيجيات التعلم والتعليم**	
Strategies	Learning and teaching strategies for this course for the Network department can include a combination of the following: 8. Lectures: Engage students through informative lectures that cover theoretical concepts and provide an overview of key topics. Use multimedia resources, visuals, and real-world examples to enhance understanding. 9. Group Discussions and Collaborative Learning: Encourage group discussions and collaborative activities to foster interaction and knowledge sharing among students. Assign group projects or case studies that require teamwork and problem-solving. 10. Online Resources and Multimedia: Utilize online resources, interactive tutorials, and multimedia materials to supplement learning. This can include video lectures, online quizzes, virtual labs, and interactive modules. 11. Assignments and Projects: Assign individual and group projects that

	require students to apply their knowledge and skills to solve real-world problems or complete practical tasks. This promotes critical thinking, problem-solving, and practical application of concepts. 12. Assessments and Feedback: Conduct regular assessments, quizzes, and examinations to evaluate students' understanding of the course material. Provide timely and constructive feedback to help students identify areas of improvement. 13. Industry Visits and Field Trips: Organize visits to IT companies, data centers, or relevant organizations to expose students to real-world IT environments. This provides valuable industry insights and networking opportunities. 14. Online Discussion Forums and Communication Platforms: Establish online discussion forums or communication platforms where students can ask questions, share resources, and engage in discussions outside of the classroom. These strategies promote active learning, practical application of knowledge, and engagement with the subject matter. They cater to different learning styles and encourage students to develop critical thinking, problem-solving, and communication skills necessary for success in this field.

Delivery Plan (Weekly Syllabus) **المنهاج الاسبوعي النظري**	
	Material Covered
Week 1	• Introduction to TCP/IP and the Internet
Week 2	• TCP/IP Protocol Suite Layers
Week 3	• Infrastructure Network and Connecting Devices
Week 4-6	• Classful Addressing and Special Addressing
Week 7-9	• Subnetting and Supernetting
Week 10	• Mid Term
Week 11-12	• Classless Addressing
Week 13-14	• Delivery and Routing
Week 15	• Review
Week 16	**Preparatory week before the final Exam**

Learning and Teaching Resources مصادر التعلم والتدريس		
	Text	**Available in the Library?**
Required Texts	TCP/IP PROTOCOL SUITE	
Recommended Texts		
Websites		

MODULE DESCRIPTION FORM
نموذج وصف المادة الدراسية

Module Information
معلومات المادة الدراسية

Module Title	**Data Structures**	**Module Delivery**	
Module Type	**Core**	☒ **Theory** ☒ **Lecture** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**	
Module Code	**NT204**		
ECTS Credits	**6**		
SWL (hr/sem)	**150**		

Module Level	1	**Semester of Delivery**	1
Administering Department	NT	**College**	CSM
Module Leader	Dr. Radhawn Mohammed	**e-mail**	E-mail
Module Leader's Acad. Title	Asst. Prof.	**Module Leader's Qualification**	Ph.D.
Module Tutor	Name (if available)	**e-mail**	E-mail
Peer Reviewer Name	Name	**e-mail**	E-mail
Scientific Committee Approval Date	10/6/2024	**Version Number**	1.0

Relation with other Modules
العلاقة مع المواد الدراسية الأخرى

Prerequisite module	Problems Solving & Programming 2	**Semester**	NT107
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents **أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية**	
Module Objectives **أهداف المادة الدراسية**	The objectives of the "Data Structures" course are: 1. Understanding Fundamental Data Structures: To introduce students to the fundamental data structures such as arrays, linked lists, stacks, queues, trees, and graphs, and their characteristics, operations, and applications. 2. Implementing Data Structures: To provide students with hands-on experience in implementing data structures using programming languages, allowing them to understand the internal workings and mechanisms of these structures. 3. Algorithmic Problem-Solving: To develop students' problem-solving skills by designing and implementing algorithms using appropriate data structures, and to apply these algorithms to solve real-world problems efficiently. 4. Performance Optimization: To teach students techniques for optimizing the performance of data structures and algorithms, such as choosing the most suitable data structure for a given problem and employing efficient algorithms for common operations. 5. Teamwork and Collaboration: To encourage teamwork and collaboration through group projects and assignments, enabling students to work effectively in teams and learn from each other's perspectives and approaches. 6. Critical Thinking and Analysis: To foster critical thinking and analytical skills by challenging students to evaluate, modify, and improve existing data structures and algorithms, and to adapt them to new problem scenarios. 7. Practical Application: To demonstrate the practical application of data structures in various domains such as software development, database management, networking, and artificial intelligence, emphasizing their relevance in real-world scenarios. By achieving these objectives, students will develop a strong foundation in data structures and acquire the skills necessary to design, implement, and analyze efficient algorithms and data structures for solving complex problems
Module Learning Outcomes	Upon completion of the "Data Structures" course, students will be able to: 1. Understand and Identify Data Structures: Identify and differentiate

مخرجات التعلم للمادة الدراسية	various data structures such as arrays, linked lists, stacks, queues, trees, and graphs, and understand their characteristics, advantages, and limitations. 2. Implement Data Structures: Implement data structures using programming languages, demonstrating proficiency in coding and understanding the internal workings and mechanisms of data structures. 3. Design and Implement Algorithms: Design and implement algorithms to solve problems efficiently using appropriate data structures, considering factors such as time complexity, space complexity, and code readability. 4. Apply Data Structures to Real-World Problems: Apply data structures to real-world scenarios, such as database management, network routing, and algorithmic problem-solving, effectively solving complex problems using the appropriate data structure and algorithmic approach. 5. Evaluate and Optimize Performance: Evaluate the performance of data structures and algorithms, identify bottlenecks, and optimize their efficiency through algorithmic improvements or selecting more suitable data structures. 6. Collaborate in Team Projects: Work effectively in teams to design and implement data structure-related projects, collaborating with team members to achieve project objectives and deliver high-quality solutions. 7. Apply Critical Thinking and Problem-Solving Skills: Apply critical thinking and problem-solving skills to analyze problems, break them down into smaller subproblems, and devise effective data structure-based solutions. 8. Communicate Data Structure Concepts: Communicate data structure concepts and solutions effectively, both orally and in written form, using appropriate terminology and visual representations. 9. Continuously Learn and Adapt: Recognize the dynamic nature of data structures and algorithms, and demonstrate the ability to learn and adapt to new data structures and algorithmic techniques as they emerge in the evolving field of computer science.
Indicative Contents **المحتويات الإرشادية**	The indicative contents of the "Data Structures" course may include: 1. Introduction to Data Structures: o Overview of data structures and their significance in problem-solving. o Basic terminology and concepts related to data structures. o Abstract data types and their implementation. 2. Arrays and Linked Lists: o Array representation and operations (insertion, deletion, searching). o Singly linked lists, doubly linked lists, and circular linked lists. o Linked list operations (insertion, deletion, searching, traversal). 3. Stacks and Queues: o Stack data structure and its operations (push, pop, peek). o Queue data structure and its operations (enqueue, dequeue). o Applications of stacks and queues. 4. Trees and Binary Trees: o Tree terminology, concepts, and properties. o Binary tree representation and traversal algorithms (preorder,

	inorder, postorder). o Binary search trees and their operations (insertion, deletion, searching). 5. Heaps and Priority Queues: o Heap data structure and its properties. o Priority queue implementation using heaps. o Heap operations (insertion, deletion, heapify). 6. Graphs: o Graph terminology, types, and representations. o Graph traversal algorithms (depth-first search, breadth-first search). o Shortest path algorithms (Dijkstra's algorithm, Bellman-Ford algorithm). 7. Hashing and Hash Tables: o Hashing concepts and techniques. o Hash functions and collision resolution strategies. o Hash table implementation and operations (insertion, deletion, searching). 8. Advanced Data Structures: o Advanced topics such as balanced search trees (AVL trees, Red-Black trees), B-trees, and tries. o Advanced graph algorithms (minimum spanning trees, topological sorting). o Advanced hashing techniques (dynamic hashing, cuckoo hashing). 9. Applications and Case Studies: o Real-world applications of data structures in software development, database management, networking, and other domains. o Case studies highlighting the selection and utilization of appropriate data structures for specific problems. The above indicative contents provide a broad overview of the topics typically covered in a "Data Structures" course. The actual course content may vary depending on the specific curriculum and instructor.

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	The "Data Structures" course can be effectively taught using a combination of learning and teaching strategies, including: 1. Lectures: Engaging lectures delivered by the instructor to introduce and explain key concepts, theories, and techniques related to data structures. The lectures can include real-world examples and demonstrations to enhance understanding. 2. Hands-on Programming Exercises: Providing students with programming exercises to implement and manipulate data structures using programming languages. This allows students to gain practical experience in implementing and working with data structures. 3. Group Discussions and Peer Learning: Encouraging group discussions

	and peer learning activities where students can collaborate, share their knowledge, and solve problems together. This promotes active learning and helps reinforce understanding of data structure concepts. 4. Case Studies and Real-World Applications: Presenting case studies and real-world examples that demonstrate the practical applications of data structures. This helps students understand how data structures are used in various domains, such as software development, networking, and database management. 5. Problem-Solving Sessions: Conducting problem-solving sessions where students are given challenging problems to solve using appropriate data structures and algorithms. This helps develop their problem-solving skills and strengthens their understanding of data structure concepts. 6. Visualizations and Interactive Tools: Utilizing visualizations and interactive tools, such as animations and simulations, to illustrate the internal workings of data structures and algorithms. This visual approach enhances comprehension and helps students grasp complex concepts more easily. 7. Assessments and Feedback: Administering regular assessments, such as quizzes and assignments, to evaluate students' understanding and progress. Providing timely and constructive feedback on their work helps them identify areas of improvement and reinforces their learning. 8. Guest Lectures and Industry Experts: Inviting guest lecturers and industry experts to share their insights and experiences related to data structures. This provides students with a broader perspective and exposes them to real-world applications and challenges. 9. Online Resources and Self-Study: Recommending online resources, textbooks, and tutorials for self-study. This allows students to explore additional materials at their own pace and deepen their understanding of data structures. 10. Project-based Learning: Assigning projects that require students to design and implement solutions using data structures. This encourages creativity, problem-solving, and practical application of learned concepts. By employing these strategies, students can actively engage with the course material, develop a solid understanding of data structures, and acquire the necessary skills to apply them effectively in various contexts.

Delivery Plan (Weekly Syllabus)

المنهاج الاسبوعي النظري

	Material Covered
Week 1	Introduction to Data Structures and Algorithms
Week 2-3	Stacks and Queues
Week 4-5	Trees and Binary Trees
Week 6	Heaps and Priority Queues
Week 7-8	Graphs

9	Mid-Term Exam
Week 10-11	Hashing and Hash Tables
Week 12-13	Advanced Data Structures
Week 14	Advanced Graph Algorithms
Week 15	Review

Delivery Plan (Weekly Lab. Syllabus) **المنهاج الاسبوعي للمختبر**	
	Material Covered
Week 1	Introduction to Programming Environment 2. Setting up the programming environment (IDE, compiler, etc.) 3. Writing and executing a simple program in a programming language
Week 2	Array Manipulation 4. Implementing basic array operations (insertion, deletion, searching) 5. Analyzing the time complexity of array operations
Week 3	Linked List Implementation • Implementing a linked list data structure • Performing operations on a linked list (insertion, deletion, traversal)
Week 4	Stack and Queue Implementation 4. Implementing a stack using arrays and linked lists 5. Implementing a queue using arrays and linked lists 6. Performing stack and queue operations
Week 5	Tree Traversals 7. Implementing tree data structures (binary tree, binary search tree) 8. Performing tree traversals (pre-order, in-order, post-order)
Week 6	Heap Operations 4. Implementing a heap data structure 5. Performing heap operations (insertion, deletion, heapify)
Week 7-8	Graph Traversals • Implementing a graph data structure (adjacency matrix, adjacency list)

	• Performing graph traversals (depth-first search, breadth-first search)
Week 9	Midterm Exam
Week 10-11	Hash Table Implementation • Implementing a hash table data structure • Handling collisions using separate chaining or open addressing
Week 12-13	Balanced Binary Search Tree • Implementing a balanced binary search tree (AVL tree, red-black tree) • Performing operations on the balanced binary search tree (insertion, deletion, search)
Week 14	Graph Algorithms • Implementing graph algorithms (Dijkstra's algorithm, Kruskal's algorithm) • Analyzing the time complexity of graph algorithms
Week 15	Review

Learning and Teaching Resources
مصادر التعلم والتدريس

	Text	Available in the Library?
Required Texts	Data Structure and Program Design in C++, by Robert Kruse	Yes
Recommended Texts	Data Structure and Algorithm Analysis in C++, by Mark Allen Weiss	No
Websites		

MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

MODULE DESCRIPTION FORM
نموذج وصف المادة الدراسية

Module Information
معلومات المادة الدراسية

Module Title	**Visual Programming**	**Module Delivery**	
Module Type	**Basic**	☒ **Theory** ☒ **Lecture** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**	
Module Code	**NT207**		
ECTS Credits	**6**		
SWL (hr/sem)	**150**		

Module Level	1	**Semester of Delivery**	1
Administering Department	NT	**College**	CSM
Module Leader	Dr.Firas Mohammed Salih	**e-mail**	E-mail
Module Leader's Acad. Title	Lecturer	**Module Leader's Qualification**	Ph.D.
Module Tutor	Name (if available)	**e-mail**	E-mail
Peer Reviewer Name	Name	**e-mail**	E-mail
Scientific Committee Approval Date	10/6/2024	**Version Number**	1.0

Relation with other Modules
العلاقة مع المواد الدراسية الأخرى

Prerequisite module	Object Oriented Programming	**Semester**	NT201
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Objectives **أهداف المادة الدراسية**	The Visual programming course for the Networks department aims to achieve the following objectives: 1. Understanding Visual Programming Concepts: Gain a solid understanding of the fundamental concepts and principles of visual programming, including visual representations, event-driven programming, and graphical user interfaces (GUIs). 2. Proficiency in Visual Programming Tools: Develop proficiency in using popular visual programming tools and environments such as Scratch, Blockly, or visual programming languages like Visual Basic, Python with Tkinter, or App Inventor. 3. GUI Design and Development: Learn to design and develop user-friendly graphical user interfaces (GUIs) using visual programming tools, including layout design, component placement, and interactive elements. 4. Event-Driven Programming: Understand the concepts of event-driven programming and learn how to create event handlers and respond to user input and system events in visual programming environments. 5. Algorithmic Thinking and Problem Solving: Enhance algorithmic thinking and problem-solving skills by developing logical and computational thinking through visual programming challenges and projects. 6. Integration of Multimedia and Sensors: Explore the integration of multimedia elements such as images, audio, and video, as well as sensor inputs like motion, sound, and touch, into visual programming projects. 7. Collaboration and Teamwork: Foster collaboration and teamwork skills through group projects that involve designing, developing, and presenting visual programming applications. 8. Debugging and Troubleshooting: Develop the ability to identify and resolve errors and bugs in visual programming code through effective debugging and troubleshooting techniques. 9. Creativity and Innovation: Encourage creativity and innovation by allowing students to explore and create interactive and visually appealing applications using visual programming tools. 10. Ethical and Responsible Use of Visual Programming: Promote ethical and responsible use of visual programming by emphasizing issues such as privacy, security, intellectual property, and social impact. 11. Project Management and Documentation: Gain experience in project management by planning, organizing, and documenting visual programming projects, including requirements gathering, design, implementation, testing, and documentation. 12. Continuous Learning and Adaptation: Develop a mindset of

	continuous learning and adaptation to keep up with emerging trends and advancements in visual programming and related technologies.
Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	Upon successful completion of the Visual programming course for the Networks department, students should be able to demonstrate the following learning outcomes: 1. Proficiency in Visual Programming Tools: Students should be able to effectively use visual programming tools and environments to create functional and visually appealing applications with user-friendly interfaces. 2. GUI Design and Development: Students should be capable of designing and developing graphical user interfaces (GUIs) using visual programming techniques, including layout design, component placement, and interactive features. 3. Event-Driven Programming: Students should understand the concepts of event-driven programming and be able to create event handlers and respond to user input and system events in visual programming environments. 4. Algorithmic Thinking and Problem Solving: Students should demonstrate the ability to apply algorithmic thinking and problem-solving skills to develop logical and computational solutions to programming challenges within the visual programming paradigm. 5. Integration of Multimedia and Sensors: Students should be able to integrate multimedia elements, such as images, audio, and video, as well as sensor inputs like motion, sound, and touch, into their visual programming projects. 6. Collaboration and Teamwork: Students should have experience working collaboratively in teams to design, develop, and present visual programming applications, demonstrating effective communication, cooperation, and shared responsibility. 7. Debugging and Troubleshooting: Students should possess the skills to identify and resolve errors and bugs in visual programming code through effective debugging and troubleshooting techniques. 8. Creativity and Innovation: Students should showcase creativity and innovation by creating unique and interactive applications that go beyond basic requirements, incorporating novel ideas, design elements, or features. 9. Ethical and Responsible Use of Visual Programming: Students should exhibit an understanding of ethical considerations related to visual programming, including privacy, security, intellectual property, and the social impact of their applications. 10. Project Management and Documentation: Students should demonstrate the ability to plan, organize, and document visual programming projects, including requirements gathering, design documentation, code commenting, and user instructions. 11. Continuous Learning and Adaptation: Students should exhibit a mindset of continuous learning and adaptation, being aware of emerging trends and advancements in visual programming and related technologies and being capable of independently learning and exploring new tools and concepts.

Indicative Contents **المحتويات الإرشادية**	The indicative contents of the Visual programming course for the computer department may include the following topics: 13. Introduction to Visual Programming o Overview of visual programming concepts and benefits o Introduction to visual programming tools and environments o Basic elements and features of visual programming interfaces 14. GUI Design and Layout o Principles of graphical user interface (GUI) design o Layout managers and component placement o Styling and customization of GUI elements 15. Event-Driven Programming o Introduction to event-driven programming paradigm o Handling user input events (e.g., button clicks, mouse movements) o Responding to system events (e.g., window events, timer events) 16. Data Manipulation and Visualization o Working with data structures and variables in visual programming o Displaying data using charts, graphs, and other visualization techniques o Interacting with data through input forms and user controls 17. Multimedia Integration o Incorporating images, audio, video, and animations into visual programming projects o Manipulating multimedia elements using visual programming tools o Creating interactive multimedia applications 18. Animation and Game Development o Introduction to animation concepts in visual programming o Creating animations and transitions o Developing simple games using visual programming techniques 19. Database Integration o Connecting visual programming applications to databases o Retrieving and manipulating data from a database using visual programming tools o Creating interactive forms for data entry and retrieval 20. Web and Mobile Application Development o Introduction to web and mobile application development using visual programming o Creating web pages or mobile app interfaces with visual programming tools o Integrating web services or mobile device features into visual programming applications 21. Advanced Visual Programming Concepts o Advanced GUI design techniques (e.g., drag-and-drop, custom controls) o Multi-threading and concurrency in visual programming o Extending visual programming functionality with plugins or extensions 22. Project Development o Planning, designing, and implementing a visual programming

	project ○ Applying learned concepts and techniques to develop a substantial application ○ Iterative development, testing, and debugging of the project 23. User Experience (UX) Design and Usability ○ Introduction to UX design principles and usability considerations ○ Conducting user testing and incorporating user feedback into visual programming projects ○ Enhancing the user experience through effective design choices 24. Deployment and Distribution ○ Packaging and distributing visual programming applications for different platforms ○ Considerations for deployment on web, desktop, or mobile platforms ○ App store guidelines and submission processes

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	Learning and teaching strategies for the Visual programming course for the 1. Hands-on Projects: Encourage students to actively engage in hands-on projects throughout the course. Assign programming tasks and projects that allow students to apply the visual programming concepts they have learned. Provide them with real-world scenarios and challenges to solve using visual programming tools. 2. Interactive Demos and Examples: Use interactive demos and examples to demonstrate the application of visual programming concepts. Show step-by-step development of applications, highlighting key features and techniques. Encourage students to experiment with the provided examples and modify them to gain a deeper understanding. 3. Collaborative Learning: Promote collaborative learning by assigning group projects or pairing students for programming tasks. Encourage students to discuss and share ideas, troubleshoot issues together, and provide feedback to their peers. This fosters teamwork and enhances problem-solving skills through collective effort. 4. Practice and Experimentation: Assign regular coding exercises and practice sessions to reinforce learning. Provide a variety of problems and challenges to tackle, allowing students to practice different aspects of visual programming. Encourage experimentation and creativity by giving them the freedom to explore different approaches and solutions. 5. Code Review and Feedback: Incorporate code review sessions where students can present their projects and receive feedback from the instructor and their peers. Provide constructive criticism to help students improve their coding style, design choices, and adherence to best practices in visual programming. 6. Online Resources and Tutorials: Share supplementary online resources, tutorials, and documentation related to visual programming tools and

	concepts. Point students to helpful websites, forums, and video tutorials where they can find additional learning materials and examples to deepen their understanding. 7. Mini-Projects and Challenges: Introduce mini-projects and coding challenges that require students to think creatively and solve specific problems using visual programming. These smaller-scale projects allow students to focus on specific skills or concepts and provide opportunities for self-assessment and self-improvement. 8. Reflective Learning: Encourage students to reflect on their learning progress and experiences. Assign periodic reflection papers or journal entries where they can express their thoughts, challenges faced, and lessons learned while working on visual programming projects. This promotes metacognition and helps students identify areas of improvement. 9. Guest Speakers and Industry Insights: Invite guest speakers from the industry who have expertise in visual programming or have utilized visual programming tools in their work. They can share real-world applications and insights, providing students with a broader perspective on the relevance and practical applications of visual programming. 10. Documentation and Documentation Review: Emphasize the importance of documenting code and maintaining clear, well-structured project documentation. Teach students how to write effective comments, documentation, and README files. Conduct documentation review sessions to assess their ability to communicate their code and project effectively.

Delivery Plan (Weekly Syllabus) المنهاج الاسبوعي النظري	
	Material Covered
Week 1	Week 1: Introduction to Visual Programming • Overview of visual programming concepts and tools • Getting familiar with the visual programming environment
Week 2	Week 2: GUI Design and Layout • Exploring layout managers and component placement • Creating simple user interfaces
Week 3	Week 3: Event-Driven Programming • Understanding event-driven programming paradigm • Handling user input events and system events
Week 4	Week 4: Data Manipulation and Visualization

	• Working with data structures and variables in visual programming • Displaying data using charts, graphs, or other visualization techniques
Week 5	Week 5: Multimedia Integration • Incorporating images, audio, video, and animations into projects • Manipulating multimedia elements using visual programming tools
Week 6	Week 6: Animation and Game Development • Creating animations and transitions • Developing simple games using visual programming techniques
Week 7	Week 7: Database Integration • Connecting visual programming applications to databases • Retrieving and manipulating data using visual programming tools
Week 8	Week 8: Web and Mobile Application Development • Introduction to web and mobile app development with visual programming • Creating web pages or mobile app interfaces
Week 9	Week 9: Advanced Visual Programming Concepts • Exploring advanced GUI design techniques • Multithreading and concurrency in visual programming
Week 10	Week 10: Project Development (Part 1) • Planning and designing a visual programming project • Implementing core features of the project
Week 11	Week 11: Project Development (Part 2) • Continuing project implementation • Testing and debugging the project
Week 12	Week 12: User Experience (UX) Design and Usability • Introduction to UX design principles and usability considerations • Enhancing the user experience of visual programming projects
Week 13	Week 13: Deployment and Distribution • Packaging and distributing visual programming applications • Considerations for deploying on different platforms
Week 14	Week 14: Project Refinement and Review • Finalizing project implementation

	• Conducting code review and project review sessions
Week 15	Week 15: Final Project Presentation and Evaluation • Presenting visual programming projects to the class • Evaluation and feedback on the projects
Week 16	**Preparatory week before the final Exam**

Delivery Plan (Weekly Lab. Syllabus) **المنهاج الاسبوعي للمختبر**	
	Material Covered
Week 1	Lab 1: Introduction to Visual Programming Tools o Familiarization with the visual programming environment o Exploring the basic features and functionalities of the chosen visual programming tool o Creating a simple "Hello World" application
Week 2	Lab 2: GUI Design and Layout o Applying layout managers to create a responsive user interface o Adding components and arranging them within the interface o Customizing the appearance and styling of GUI elements
Week 3	Lab 3: Event-Driven Programming o Implementing event handlers for user interactions (e.g., button clicks, mouse events) o Responding to system events (e.g., window events, timer events) o Creating interactive applications with event-driven programming
Week 4	Lab 4: Data Manipulation and Visualization o Working with data structures (e.g., arrays, lists) in visual programming o Displaying data using charts, graphs, or other visualization techniques o Creating interactive data-driven applications
Week 5	Lab 5: Multimedia Integration o Incorporating multimedia elements (e.g., images, audio, video) into projects o Manipulating multimedia assets using visual programming tools

	○ Developing interactive multimedia applications
Week 6	Lab 6: Animation and Game Development ○ Creating animations and transitions using visual programming tools ○ Developing a simple game using visual programming techniques ○ Implementing game mechanics and user interactions
Week 7	Lab 7: Project Development and Review ○ Applying learned concepts to develop a small-scale visual programming project ○ Testing and debugging the project ○ Presenting the project and receiving feedback from peers and the instructor

Learning and Teaching Resources **مصادر التعلم والتدريس**		
	Text	**Available in the Library?**
Required Texts	Microsoft Visual C# Step by Step, 10th Edition, By John Sharp Microsoft Press, 2022	
Recommended Texts		
Websites		

MODULE DESCRIPTION FORM
نموذج وصف المادة الدراسية

Module Information **معلومات المادة الدراسية**		
Module Title	Website Development 2	**Module Delivery**
Module Type	**Core**	☒ **Theory** ☒ **Lecture** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**
Module Code	**NT210**	
ECTS Credits	**6**	
SWL (hr/sem)	**150**	

Module Level	1	**Semester of Delivery**	1
Administering Department	NT	**College**	CSM
Module Leader	Zaid Dawood Salim	**e-mail**	E-mail
Module Leader's Acad. Title	Asst. Lecturer	**Module Leader's Qualification**	Ph.D.
Module Tutor	Name (if available)	**e-mail**	E-mail
Peer Reviewer Name	Name	**e-mail**	E-mail
Scientific Committee Approval Date	10/6/2024	**Version Number**	1.0

Relation with other Modules

العلاقة مع المواد الدراسية الأخرى

Prerequisite module	Web Development 1	**Semester**	NT202
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents

أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية

Module Objectives
أهداف المادة الدراسية

The course objectives for Web Development 2 may include:

1. Advanced Web Technologies: Introduce students to advanced web technologies and frameworks, such as server-side scripting languages (e.g., PHP, Python), content management systems (e.g., WordPress), and client-side frameworks (e.g., React, Angular).
2. Database Integration: Teach students how to integrate databases into web applications, including database design, querying, and data manipulation using SQL. Focus on concepts such as data modeling, normalization, and database connectivity.
3. Dynamic Web Development: Enable students to build dynamic web applications by integrating server-side scripting languages with client-side technologies. Cover topics like session management, form handling, user authentication, and data validation.
4. Web Security: Raise awareness about common web security vulnerabilities and techniques to secure web applications. Teach students about secure coding practices, input validation, authentication

	mechanisms, and protection against common attacks like cross-site scripting (XSS) and SQL injection. 5. Web Performance Optimization: Explore techniques to improve the performance and efficiency of web applications. Cover topics like caching, code minification, image optimization, and front-end optimization techniques to enhance the user experience. 6. Responsive Web Design: Introduce students to the principles of responsive web design and teach them how to create websites that adapt and display well on different devices and screen sizes. Cover CSS frameworks, media queries, and techniques for creating responsive layouts. 7. Web Accessibility: Emphasize the importance of creating web applications that are accessible to users with disabilities. Teach students about accessibility standards, techniques for implementing accessible features, and the use of assistive technologies. 8. Project Development: Provide opportunities for students to work on larger-scale web development projects. Encourage collaborative project work, where students can apply their knowledge and skills to build real-world web applications. 9. Industry Practices and Emerging Trends: Keep students updated with current industry practices and emerging trends in web development. Introduce them to topics like progressive web apps, single-page applications, API integrations, and the use of modern tools and frameworks. 10. Professional Development: Foster professional skills by promoting effective communication, teamwork, project management, and problem-solving abilities within the context of web development projects. These objectives aim to equip students with the necessary knowledge, skills, and techniques to become proficient web developers capable of building dynamic and secure web applications using the latest technologies and industry best practices.
Module Learning Outcomes **مخرجات التعلم للمادة الدراسية**	The learning outcomes for Web Development 2 may include: 1. Advanced Web Technologies: Demonstrate proficiency in using advanced web technologies and frameworks, such as server-side scripting languages, content management systems, and client-side frameworks, to develop robust and scalable web applications. 2. Database Integration: Apply database integration techniques to create dynamic web applications, including database design, querying, and data manipulation. Develop skills in working with databases and understanding the importance of efficient data management. 3. Dynamic Web Development: Build dynamic web applications by integrating server-side scripting languages with client-side technologies. Implement features like session management, form handling, user authentication, and data validation to create interactive and responsive web experiences. 4. Web Security: Identify and mitigate common web security vulnerabilities. Implement secure coding practices, employ authentication mechanisms, and protect against common attacks like cross-site scripting (XSS) and SQL injection to ensure the security of web applications. 5. Web Performance Optimization: Optimize the performance of web

	applications by implementing techniques such as caching, code minification, and front-end optimization. Improve website loading speed and user experience through efficient resource management. 6. Responsive Web Design: Create responsive web designs that adapt to different devices and screen sizes. Develop skills in using CSS frameworks, media queries, and responsive layout techniques to ensure consistent and visually appealing experiences across multiple platforms. 7. Web Accessibility: Design and develop web applications that are accessible to users with disabilities. Apply accessibility standards and techniques to ensure equal access and usability for all users, considering factors such as screen readers, keyboard navigation, and alternative text for images. 8. Project Development: Collaborate with a team to plan, design, and implement larger-scale web development projects. Apply project management principles, communicate effectively, and work collaboratively to deliver high-quality web applications. 9. Industry Practices and Emerging Trends: Stay informed about current industry practices and emerging trends in web development. Demonstrate awareness of technologies, tools, and frameworks used in the industry, and adapt to changing demands and advancements in the field. 10. Professional Growth: Demonstrate professionalism in web development by effectively communicating ideas, solving problems, and adapting to new challenges. Continuously improve skills and stay updated with industry advancements through self-directed learning and professional development opportunities. These learning outcomes aim to equip students with the knowledge, skills, and abilities to become proficient web developers who can create dynamic, secure, and user-friendly web applications using advanced technologies and industry best practices.
Indicative Contents **المحتويات الإرشادية**	The indicative contents for Web Development 2 may include: 1. Introduction to Server-Side Scripting: Overview of server-side scripting languages such as PHP, Python, or Node.js. Understanding the server-side architecture and the role of server-side languages in web development. 2. Database Integration: Exploring advanced database integration techniques using SQL or NoSQL databases. Topics may include database design, advanced querying, data manipulation, and database security. 3. Content Management Systems (CMS): Introduction to popular CMS platforms like WordPress, Drupal, or Joomla. Understanding the architecture, theme development, plugin customization, and content management using CMS. 4. Web Frameworks: Introduction to popular web frameworks such as Ruby on Rails, Django, or Laravel. Exploring the features, MVC architecture, routing, database integration, and rapid development using web frameworks. 5. RESTful API Development: Understanding the concepts of RESTful APIs and their role in web development. Building and consuming RESTful APIs using popular frameworks or libraries like Express.js or

	Flask. 6. Authentication and Authorization: Implementing user authentication and authorization mechanisms in web applications. Topics may include user registration, login/logout functionality, password hashing, and role-based access control. 7. Web Security Best Practices: Exploring advanced web security concepts and best practices. Topics may include handling user input securely, preventing common vulnerabilities like CSRF and XSS attacks, and implementing secure coding practices. 8. Web Performance Optimization: Techniques for optimizing the performance of web applications. Topics may include caching, asynchronous loading, minification, image optimization, and front-end performance best practices. 9. Responsive Web Design: Advanced concepts in responsive web design. Exploring responsive frameworks, media queries, responsive images, and techniques for creating mobile-friendly and adaptive web layouts. 10. Testing and Debugging: Strategies for testing and debugging web applications. Topics may include unit testing, integration testing, browser debugging tools, and error handling techniques. 11. Version Control and Collaboration: Introduction to version control systems like Git and their role in collaborative web development. Understanding branching, merging, resolving conflicts, and collaborative development workflows. 12. Project Development: Working on a larger-scale web development project in a team environment. Applying project management principles, agile development methodologies, and effective communication and collaboration skills.

Learning and Teaching Strategies **استراتيجيات التعلم والتعليم**	
Strategies	The learning and teaching strategies for Web Development 2 can include: 1. Lectures: The instructor delivers lectures to introduce new concepts, explain theoretical aspects, and provide examples and case studies related to web development. 2. Hands-on Coding: Students engage in hands-on coding exercises and projects to apply their knowledge and skills in building web applications. They can work individually or in groups to develop real-world projects, implementing the concepts learned during the course. 3. Code Review and Feedback: Students participate in code reviews where they share their code with peers and receive feedback. This promotes collaboration, peer learning, and improvement of coding practices. 4. Practical Examples and Demonstrations: The instructor demonstrates practical examples and showcases real-world applications to illustrate the concepts and techniques in web development. This helps students visualize the application of the learned concepts. 5. Discussion and Debate: Students engage in discussions and debates on topics related to web development, such as emerging trends, best practices, and ethical considerations. This encourages critical thinking,

	problem-solving, and the exploration of different perspectives. 6. Guest Speakers and Industry Experts: Inviting guest speakers and industry experts to share their experiences and insights in web development can provide students with valuable industry perspectives and practical knowledge. 7. Workshops and Tutorials: Conducting workshops and tutorials where students can work on specific web development tasks, solve problems, and learn new tools and technologies. These sessions can be interactive and allow students to receive guidance and support from the instructor. 8. Online Resources and Self-Study: Providing access to online resources, tutorials, documentation, and coding exercises to encourage self-study and exploration. Students can deepen their understanding of web development concepts and technologies at their own pace. 9. Project-based Learning: Assigning individual or group projects that require students to design, develop, and deploy web applications. This allows them to apply their knowledge, practice problem-solving skills, and gain hands-on experience in real-world scenarios. 10. Assessments and Feedback: Conducting regular assessments, quizzes, and assignments to evaluate students' understanding of the concepts and their ability to apply them. Providing timely feedback helps students identify areas of improvement and reinforce their learning. 11. Collaborative Learning: Encouraging collaboration among students through group work, pair programming, or collaborative coding sessions. This fosters teamwork, communication skills, and the sharing of knowledge and expertise. 12. Industry Case Studies: Presenting case studies of successful web development projects and applications in various industries. This helps students understand the practical application of web development skills and the challenges faced in real-world scenarios. These strategies aim to create an engaging and interactive learning environment that promotes active participation, practical application of knowledge, and continuous learning in the field of web development.

Delivery Plan (Weekly Syllabus)

المنهاج الاسبوعي النظري

	Material Covered
Week 1	: Introduction to Server-Side Programming • Overview of server-side programming languages (e.g., PHP, Python, Node.js) • Setting up a development environment with a server-side language and a web server
Week 2-3	Database Integration • Introduction to database systems (e.g., MySQL, PostgreSQL) • Connecting a web application to a database

	• Querying and manipulating data using SQL
Week 4-5	Advanced JavaScript and DOM Manipulation • JavaScript libraries and frameworks for front-end development (e.g., React, Angular) • Advanced DOM manipulation techniques • Handling asynchronous operations using AJAX and promises
Week 6-7	Web Application Security • Common web vulnerabilities (e.g., Cross-Site Scripting, SQL injection) • Techniques for securing web applications • Implementing user authentication and authorization
Week 8-9	Server-Side Frameworks • Introduction to popular server-side frameworks (e.g., Laravel, Django, Express.js) • Building dynamic web applications using a framework • Implementing RESTful APIs
Week 10-11	Version Control and Collaboration • Introduction to version control systems (e.g., Git) • Collaborative web development using Git and GitHub • Deployment strategies for web applications
Week 12-13	Testing and Debugging • Writing unit tests for web applications • Debugging techniques for identifying and fixing issues • Performance optimization and code profiling
Week 14-15	Project Development and Presentation • Applying the learned concepts to develop a complete web application • Project planning, development, and documentation • Presenting and demonstrating the web application

Delivery Plan (Weekly Lab. Syllabus) **المنهاج الاسبوعي للمختبر**	
	Material Covered
Week 1	Review of Web Development Basics • Refreshing HTML, CSS, and JavaScript concepts • Recap of responsive web design principles

Week 2	Advanced CSS Techniques • Implementing CSS animations and transitions • Using CSS preprocessors (e.g., Sass, Less) • Creating CSS frameworks and libraries
Week 3	Advanced JavaScript Concepts • Exploring advanced JavaScript topics (e.g., closures, prototypes) • Working with JavaScript libraries (e.g., jQuery, Lodash) • Building modular JavaScript code using modules and namespaces
Week 4	Single-Page Applications (SPA) • Introduction to SPA architecture and frameworks (e.g., Angular, React, Vue) • Building a simple SPA using a chosen framework • Routing and navigation in SPAs
Week 5	Server-Side Scripting with Node.js 9. Setting up a Node.js development environment 10. Writing server-side JavaScript code using Node.js 11. Implementing server-side functionality and APIs
Week 6	Database Integration • Introduction to database management systems (e.g., MySQL, MongoDB) • Interacting with databases using server-side scripting (e.g., CRUD operations) • Implementing data persistence in web applications
Week 7	Authentication and Authorization • Implementing user registration and login functionality • Exploring authentication and authorization techniques (e.g., sessions, tokens) • Securing web applications against common vulnerabilities (e.g., cross-site scripting, SQL injection)
Week 8	API Development • Designing and implementing RESTful APIs • Handling API requests and responses • Documentation and testing of APIs
Week 9	Web Performance Optimization • Techniques for optimizing web page load times • Implementing caching strategies • Analyzing and improving website performance using tools (e.g., PageSpeed Insights, Lighthouse)
Week 10	Advanced Front-End Frameworks

	• Exploring advanced features and components of front-end frameworks (e.g., Angular, React) • Building complex web applications with front-end frameworks •
Week 11-14	Web Project Development • Applying the concepts learned to develop a complex web project • Planning, designing, and implementing a dynamic web application • Testing, debugging, and optimizing the web project
Week 15	Project Presentation and Evaluation

Learning and Teaching Resources
مصادر التعلم والتدريس

	Text	Available in the Library?
Required Texts		No
Recommended Texts		
Websites		