Ministry of Higher education and Scientific Research

UNIVERSITY OF ANBAR

COLLEGE OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

Department of Computer Science

Department of Computer Science

CATALOUGE

2023/2024

Vision

Our vision is for the department, along with the college, to be an “educational” and “research” edifice in its programs, educational curricula, and scientific research, and seeks to achieve a prominent position among the relevant departments in Iraqi universities by providing and updating distinguished, modern programs that focus on the requirements of the labor market and technological development.

Mission

The department's mission is to prepare, qualify, and supply the labor market with distinguished graduates equipped with the knowledge and skills necessary to solve problems, possess functional intelligence skills, and qualify them to meet the needs of various government institutions and the labor market, as well as the ability to conduct scientific and applied research and provide consulting and training services in the college's specialized fields.

Programme Educational Objectives (PEOs)

1. Building an in-depth and solid theoretical scientific knowledge base through which the student learns theoretical knowledge in computer hardware and software that present scientific concepts.
2. Providing a suitable and comfortable research and teaching environment for teachers to reach high-quality outputs; opening the door to academic research and scientific cooperation with universities and international organizations; and providing consultations and community services in the field of information technology.
3. Building scientific and practical skills in analyzing and building software systems; adopting formulas for building integrated software in laboratory work; and adopting formulas for laboratory and field work
4. Developing the student’s mental abilities through analysis and logical deduction and enabling him to solve programming problems.
5. The necessary development of school curricula to ensure the integration of modern changes in computer science technology and the application of e-learning.

6. Encouraging innovative ideas and projects and developing leadership and creativity skills in the field of information technology by urging students to participate in computer events and forums.

- Career opportunities
- Computer science programmer
- Video game programmer
- Database designer and administrator
- Information security manager
- Manager and software developer
- Computer Center Manager
- Network manager and analyst
- Information systems analyst
- Image analysis and processing

Course Description

Courses are coded as follows:

1. Course code and number
2. Course title
3. Parenthesized numerals, e.g., (4-3-1-3), indicate, in order, the credit hours, the classroom hours (1 hour = 1 credit hour), tutorial hours (credit hour = 0), and the laboratory hours (3 hour = 1 credit hour).

Prerequisites, if any, are indicated at the course description. These have been established to assure an adequate and uniform background for students in advanced classes. Occasionally, students may feel they already have the appropriate background for an advanced course because of previous training, transfer credits, or Credit by examination.

Course Numbering System:

Course code = CS

Total credit hours of courses according to the adopted levels

Courses		Levels				Total
		First	Second	Third	Fourth	
University	Compulsory		4	2	2	8
	Elective					
Total			4	2	2	8
College	Compulsory		14	6	12	32
	Elective					
Total			14	6	12	32
Department	Compulsory		16	22	21	59
	Elective		3	6	3	12
Total						71
Total			37	36	38	111

College requirements for a Master's degree in computer science

N0.	Subject: Semester 1	Units	Hours
1	Advanced Data Warehouse and DSS	3	3
2	Advanced Computer Networks	3	3
3	Advanced Mathematics	3	3
4	Image Processing and Computer Vision	3	3
5	English	1	2
N0.	**Subject: Semester 2**	**Units**	**Hours**
1	Advanced Computer Networks	3	3
2	Advanced Mobile Computing	3	3
3	Evolutionary Computation	3	3
4	Embedded Systems	3	3
5	Research Methodology	0	1

College requirements for obtaining a doctorate degree in computer science

N0.	Subject: Semester 1	Units	Hours
1	Selected Topics in Information Security	3	3
2	Deep Learning	3	3
3	IoT	3	3
4	Metaheuristics	3	3
5	Research Methodology	0	1
N0.	Subject: Semester 2	Units	Hours
1	Network Security	3	3
2	Advanced Data Mining	3	3
3	Big Data	3	3
4	Robotics	3	3
5	English	1	2

University Requirements: 8 credit hours

Course code	Course Title	Credit hours	Weekly hours	Prerequisite
UOA232	English Language-II	2	2	
UOA301	English-III	2	2	
UOA401	English-IIII	2	2	
UOA006	The crimes of the defunct Ba'ath party	2	2	
Total		8	8	

College Requirements: 32

Course Code	Course Title	Credit hours	Weekly hours			Prerequisite
			Lec.	Tut.	Lab	
CCIT060	Data Structures	3	2		2	
CCIT061	Object Oriented Programming I	4	3		2	

CCIT062	Numerical Analysis	3	2		2	
CCIT063	Object Oriented Programming II	4	3		2	
CCIT064	Visual Programming I	3	2		2	
CCIT065	Visual Programming II	3	2		2	
CCIT066	Artificial Intelligence I	3	2		2	
CCIT067	Artificial Intelligence II	3	2		2	
CCIT068	Project in CS	6			12	
Total		**32**	**18**		**26**	

Department Requirements: 71

Course Code	Course Title	Credit hours	Weekly hours			Prerequisite
			Lec.	Tut.	Lab	
CSDC210	Computational Theory 1	2	2			
CSDC209	Database Management Systems I	3	2		2	
CSDC203	Advanced Mathematics	3	3			
CSDE205	Computational Theory II	2	2			
CSDE213	Data Base programming	3	2		2	
CSDE206	Gaming Programming	3	2		2	
CSDC208	Algorithms	3	2		2	
CSDC308	Computer Graphics 2D	3	2		2	
CSDC307	Computer Architecture	2	2			

CSDC305	Computer Networks I	3	2		2	
CSDC310	Compilers I	3	2		2	
CSDC307	Mobile Applications Programming	3	2		2	
CSDE310	Computer Graphics 3D	3	2		2	
CSDE308	Internet of Things	3	2		2	
CSDE311	Computer Networks II	3	2		2	
CSDE307	Compilers II	3	2		2	
CSDC309	Software Engineering	2	2			
CSIT401	Operating Systems I	3	2		2	
CSDC404	Computer Security 1	2	2			
CSDC403	PHP Web Development	3	2		2	
CSDC406	Digital Image Processing	3	2		2	
CSDE407	Research methodology	2	2			
CSDE408	Operating Systems II	3	2		2	
CSDE411	Computer Security	2	2			
CSDE409	Web Programming –Asp	3	2		2	
CSDE412	Computer Vision	3	2		2	
Total		**71**	**53**		**36**	

Elective Courses

Course Code	Course Title	Credit hours	Weekly hours			Prerequisite
			Lec.	Tut.	Lab	
CSDE206	Gaming Programming	3	2		2	
CSDE308	Internet of Things	3	2		2	
CSDC307	Mobile Applications Programming	3	2		2	
CSDE409	Web Programming –Asp	3	2		2	
Total		12	8		8	

Total Credits = 12
Total In-Touch Hours = 12

SECOND LEVEL

Course code	Course Title	Credit Hours	Weekly hours			Prerequisite
			Lec.	Tut.	Lab.	
CSDC210	Computational Theory 1	2	2			
CSDC209	Database Management Systems I	3	2		2	
CCIT061	Object Oriented Programming I	4	3		2	
CCIT060	Data Structures	3	2		2	
CSDC203	Advanced Mathematics	3	3			
UOA006	The crimes of the defunct Ba'ath party	2	2			
CSDE205	Computational Theory II	2	2			
CSDE213	Data Base programming	3	2		2	
CSDE206	Gaming Programming	3	2		2	
CCIT063	Object Oriented Programming II	4	3		2	
CSDC208	Algorithms	3	2		2	
CCIT062	Numerical Analysis	3	2		2	
UOA232	English 2	2	2			
Total		37	30		16	

THIRD LEVEL

Course code	Course Title	Credit Hours	Weekly hours			Prerequisite
			Lec.	Tut.	Lab.	
CCIT064	Visual Programming I	3	2		2	
CSDC308	Computer Graphics 2D	3	2		2	
CSDC307	Computer Architecture	2	2			
CSDC305	Computer Networks I	3	2		2	
CSDC310	Compilers I	3	2		2	
CSDC307	Mobile Applications Programming	3	2		2	
UOA301	English 3	2	2			
CCIT065	Visual Programming II	3	2		2	
CSDE310	Computer Graphics 3D	3	2		2	
CSDE308	Internet of Things	3	2		2	
CSDE311	Computer Networks II	3	2		2	
CSDE307	Compilers II	3	2		2	
CSDC309	Software Engineering	2	2			
Total		36	26		20	

FOURTH LEVEL

Course code	Course Title	Credit Hours	Weekly hours			Prerequisite
			Lec.	Tut.	Lab.	
CSIT401	Operating Systems I	3	2		2	
CSDC404	Computer Security 1	2	2			
CCIT066	Artificial Intelligence I	3	2		2	
CSDC403	PHP Web Development	3	2		2	
CSDC406	Digital Image Processing	3	2		2	
CSDE407	Research methodology	2	2			
CSDE408	Operating Systems II	3	2		2	
CSDE411	Computer Security	2	2			
CCIT067	Artificial Intelligence II	3	2		2	
CSDE409	Web Programming –Asp	3	2		2	

CSDE412	Computer Vision	3	2		2	
CCIT068	Project in CS	6			12	
UOA401	English 4	2	2			
Total		**38**	**24**		**28**	

Curriculum Map of the PEOs with courses according to learning outcomes

LEVEL	Course code	Course title	الاهداف المعرفية			الاهداف المهاراتية الخاصة بالبرنامج			الاهداف الوجدانية والقيمية		
			A1	A2	A	B1	B2	B3	C1	C2	C3
SECOND LEVEL	CSDC21	Computational	✓			✓			✓		
	CSDC20	Database	✓			✓	✓		✓	✓	
	CSDC20	Object Oriented	✓	✓		✓	✓		✓	✓	
	CSIT201	Data Structures	✓			✓	✓		✓		
	CSDC20	Advanced	✓			✓	✓		✓		
	UOA006	The crimes of the	✓			✓			✓		
	CSDE20	Computational	✓			✓	✓		✓		
	CSDE21	Data Base	✓	✓		✓	✓		✓	✓	
	CSDE20	Gaming	✓	✓		✓	✓		✓	✓	
	CSDE21 1	Object Oriented Programming II	✓			✓	✓		✓	✓	
	CSDC20	Algorithms	✓	✓		✓	✓		✓		
	CSDC20	Numerical Analysis	✓	✓		✓			✓		
	UOA232	English 2	✓			✓			✓		
	CSDC30	Visual	✓	✓		✓	✓		✓	✓	
	CSDC30	Computer Graphics	✓			✓	✓		✓		
	CSDC30	Computer		✓	✓	✓			✓		
	CSDC30	Computer		✓	✓	✓			✓	✓	
	CSDC31	Compilers I	✓			✓			✓	✓	
	CSDC30	Mobile	✓	✓		✓			✓		
	UOA301	English 3	✓	✓	✓	✓			✓		
	CSDE31	Visual	✓	✓		✓	✓		✓	✓	
	CSDE31	Computer Graphics	✓			✓	✓		✓		

THIRD LEVEL	CSDE30	Internet of Things	✓	✓		✓	✓		✓		
	CSDE31	Computer	✓	✓		✓			✓		
	CSDE30	Compilers II	✓	✓		✓			✓		
	CSDC30	Software	✓			✓			✓	✓	
FOURTH LEVEL	CSIT401	Operating Systems	✓	✓		✓			✓		
	CSDC40	Computer Security	✓			✓	✓		✓		
	CSDC40	Artificial	✓	✓		✓	✓		✓		
	CSDC40	PHP Web	✓	✓		✓	✓		✓		
	CSDC40	Digital Image	✓			✓	✓		✓	✓	
	CSDE40	Research		✓	✓	✓			✓		
	CSDE40	Operating Systems	✓	✓		✓	✓		✓		
	CSDE41	Computer Security	✓			✓	✓		✓		
	CSDE41	Artificial	✓	✓		✓	✓		✓		
	CSDE40	Web Programming	✓	✓		✓	✓		✓	✓	
	CSDE41	Computer Vision	✓			✓			✓	✓	
	CSDC41	Project in CS	✓	✓		✓			✓		
	UOA401	English 4	✓	✓	✓	✓			✓		

UNIVERSITY REQUIREMENT COURSES
OR
College REQUIREMENT COURSES
OR
Department REQUIREMENT COURSES

CSDC210 Computational Theory 1 (2-2-0-0)

- Designation as a „required‟ or „elective‟ course:

This is a required course for the computer science department .

- **Course Topics :**Set notation, Definitions, Finite Automata (DFA, NFA), Regular Expression, Transition Graph, Kleens Theorem
- **Course Description:** Finite automata are useful models for many important kinds of hardware and software. Here are the most important kinds: Software for designing and checking the behavior of digital circuits; The "lexical analyzer" of a typical complier, that is, the compiler component that breaks the input text into logical units, such as identifiers, keywords, and punctuation; Software for scanning large bodies of text, such as collections of Web pages, to find occurrences of words, phrases, or other patterns; Software for verifying systems of all types that have a finite number of distinct states, such as communication protocols or protocols for secure exchange of information.
- **Course Outcomes:**

- Knowledge and understanding
- Acquire a full understanding and mentality of Automata Theory as the basis of all computer science
- languages design
- Have a clear understanding of the Automata theory concepts such as RE's, DFA's, NFA's, Stack's, Turing machines, and Grammars
    - Cognitive skills (thinking and analysis).
- Be able to design FAs, NFAs, Grammars, languages modelling, small compilers basics
- Be able to design sample automata
    - Communication skills (personal and academic).
- Be able to minimize FA's and Grammars of Context Free Languages

   - Practical and subject specific skills (Transferable Skills).

- **Recommended Textbook(s): Daniel L. A. Cohen, Introduction of the theory of computation.**
- **Prerequisites:** None

CSDC209 Database Management Systems I (3-2-0-2)

- Designation as a „required" or „elective" course:

This is a required course for the computer science department .

- **Course Topics**

- Introduction to Database Concepts
- Database Environment
- Relational Model
- Entity Relationship Model
- Introduction to SQL
- Basic SQL Tables
- DB Creation
- Data Modeling
- Constraints & Data Manipulation
- Database Design (Logical and Conceptual)
- Normalization Database Objects User Creation and Management
- Managing DB tables-Data Integrity
- Single and Multiple table queries
- Advanced Queries, Subqueries & Merge, and Introduction to Regular Expression Functions
- Preparatory week before the final Exam

- **Course Description** :Understand relational data model in terms of data structure, data integrity, and data manipulation.
  1. Understand and create conceptual database models utilizing entity-relationship.
  2. Design data structures that will limit redundancy and enforce data integrity while conforming to organizational requirements utilizing normalization methodology.

3. Understand the theory behind the relational data model as it applies to interactions with current database management systems.
4. Interpret a given data model to query the database and transform the data into information using SQL (Structured Query Language).
5. Implement a data model in a current RDBMS.

- **Course Outcomes:**

1. Apply the basic concepts of Database Systems and Applications.
2. Use the basics of SQL and construct queries using SQL in database creation and interaction.
3. Design a commercial relational database system (Oracle, MySQL) by writing SQL using the system.

Analyze and Select storage and recovery techniques of database system.

- **Recommended Textbook(s):** database System concepts 7th edition
- **Prerequisites:** None
- **Lab. Topics :**

- install and configuration SQL server
- Design a Database and create required tables. For e.g. Bank, College Database
- Write the queries to implement the joins
- Converting ER Model to Relational Model using SQL
- Write the query for implementing the following functions: MAX (), MIN (), AVG (), COUNT ()
- Write the query to implement the concept of Integrity constrains
- Write the query to create the views
- Add more tables to the created DB
- Create foreign keys and primary key
- Implementing Insert queries
- Decompose tables by creating new tables
- Implementing Update queries
- Implement join queries
- Implementing advance join queries
- Review SQL queries

CCIT061 Object Oriented Programming I (4-3-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- Introduction to Object Oriented Programming using C++
- Class and object in OOP
- Class constructor
- Local variable and class variable
- Static and none static method.
- Encapsulation
- Inheritance (Super class and sub class)
- Type of inheritance
- Mid exam
- Fried function
- Polymorphism Based on Overloaded Methods
- Exception Handling
- Using (try, catch, throw and final) with Exception
- Interface
- File
- Preparatory week before the final Exam

- **Course Description** :

- Explain the motivation for and development of object-oriented programming languages.
- Produce a set of use cases given a problem statement.
- Produce class diagrams, object interaction diagrams and object state transition diagrams for a given problem.
- Describe the essential features of an object-oriented programming language.
- Produce and/or debug code fragments that illustrate principles of object-oriented software development.
- Describe the principles for testing object-oriented software and derive sets of test data given a specification.

- **Course Outcomes:**

- Explain the motivation for and development of object-oriented programming languages.
- Produce a set of use cases given a problem statement.
- Produce class diagrams, object interaction diagrams and object state transition diagrams for a given problem.

- Describe the essential features of an object-oriented programming language.
- Produce and/or debug code fragments that illustrate principles of object-oriented software development.
- Describe the principles for testing object-oriented software and derive sets of test data given a specification.

- **Recommended Textbook(s):**
  - Object Oriented Design by Rumbaugh (Pearson publication)
  - Object-oriented programming with C++ by E.Balagurusamy, 2nd Edition, TMH.
- **Prerequisites:** None
- **Lab. Topics :**

Lab 1: develop a program to implement 1 dimension array

Lab 2: develop a program to perform matrix operation using multi-dimensions array

Lab 3: develop program that implement a class and use it with objects

Lab 4: develop program that implement a class and create array of objects

Lab 5: write program for single inherence

Lab 6: write program for hybrid inherence

Lab 7: write program for multi-inheritance

Lab 8: write code for friend function and class

Lab 9: write program for pointer object

Lab 10: write code for polymorphism

Lab 11: write code for exception handling

Lab 12: write code for try, catch, and throw

Lab 13: write program for interface

Lab 14: write code for file

CCIT060 Data Structures (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- Introduction to Data Structures
- Algorithms and Complexity
- Arrays and Pointers
- Linked List 1

- Linked List 2
- First exam
- Stack
- Queue
- Tree 1
- Tree 2
- Graph 1
- Graph 2
- Hashing 1
- Hashing 2
- Second try exam

- **Course Description** :This course covers all data structure types. It starts with defining algorithms and their complexity from the time and space prospection. Then, a list of data structure and their description is presented. The course describes every data structure in detail. In addition to that, it gives the reason to why we need this data structure and where to use it. This course includes many projects that give more understanding to the data structure studied. These projects talks about real life problems that we ask student to use one of the data structure that has been presented in the course to solve it.
- **Course Outcomes:**

- Explain and utilize linked lists, stacks, queues and trees.
- Apply design guidelines to evaluate alternative software designs.
- Basic ability to analyze algorithms and to determine algorithm correctness and time efficiency class.
- Master a variety of advanced abstract data type (ADT) and data structures and their implementations.
- Ability to apply and implement learned algorithm design techniques and data structures to solve problems.

- **Recommended Textbook(s):**

Introduction to Algorithm, third Edition, Thomas H. Cormen Algorithms, fourth edition, Robert Sedgewick and Kevin Wayne

- **Prerequisites:** None

- **Lab. Topics :**
  - Accountant application using arrays
  - Student information system using linked list

- Color cubes games using Stack
- A snake game using queu
- Social Media connections using Graph data structure
- Simple search engine application using hashtable data structure

CSIT203 Advanced Mathematics (3-3-0-0)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- Abstract of differential equation
- Separable equation
- Solve some example
- Homogenous equation
- Exact equation
- Linear equation
- Some example
- Bernoulli equation
- Second order differential equation
- Some example
- Laplace transform
- Power series , Fourier series
- Mid exam
- Review
- Final exam
- Preparatory week before the final Exam

- **Course Description** :

1. Understand the concept of ordinary and partial.
2. Understand the method of solving the first order differential equation.
3. Understand the method of solving second order differential equation.
4. Understand the Laplace transform.
5. Understand the Fourier series.
6. Subject-specific skills:
7. Explain what mean of ordinary and partial.
8. Classify the method of solving.

- **Course Outcomes:**

- Understand the concept of ordinary and partial.
- Understand the method of solving the first order differential equation.
- Understand the method of solving second order differential equation.
- Understand the Laplace transform.
- Understand the Fourier series.
- Subject-specific skills:
- Explain what mean of ordinary and partial.
- Classify the method of solving.
- Classify the differential equation.
- Teaching and Learning Methods By solving many exercises.

- **Recommended Textbook(s):**
- **Prerequisites:** None
- **Lab. Topics :**

UOA006 The crimes of the defunct Ba'ath party (2-2-0-0)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

  - جرائم نظام البعث وفق قانون المحكمة الجنائية العراقية العليا لعام 2005
  - تعريف الجريمة ومصطلحاتها
  - أنواع الجرائم
  - جرائم نظام البعث وفق توثيق قانون المحكمة الجنائية العراقية العليا لعام 2005
  - الجرائم النفسية والاجتماعية وآثارها
  - الجرائم النفسية
  - اليات الجرائم النفسية
  - الجرائم الاجتماعية
  - اثار الجرائم النفسية
  - عسكرة المجتمع
  - انتهاكات القوانين العراقية
  - امتحان
  - قرارات الانتهاكات السياسية
  - الجرائم البيئية
  - تجفيف الاهوار
  - تجفيف البساتين

- امتحان

- **Course Description** :

- تغطي هذه الوحدة القضايا التالية:
- مفهوم الجرائم وأنواعها، تعريف الجريمة ومصطلحاتها، أنواع الجرائم الدولية، القرارات الصادرة من المحكمة الجنائية العليا، الجرائم النفسية والاجتماعية وآثارها، الجرائم البيئية.

- **Course Outcomes:**

- تغطي هذه الوحدة القضايا التالية:
- مفهوم الجرائم وأنواعها، تعريف الجريمة ومصطلحاتها، أنواع الجرائم الدولية، القرارات الصادرة من المحكمة الجنائية العليا، الجرائم النفسية والاجتماعية وآثارها، الجرائم البيئية.

- **Recommended Textbook(s):** https://www.uoanbar.edu.iq/ComputerCollege//catalog/INFO_depart/lectures/infoS_2_1_baathall_compressed.pdf

- **Prerequisites:** None

- **Lab. Topics** :

CSDE205 Computational Theory II (2-2-0-0)

- Designation as a „required‟ or „elective‟ course:
This is a required course for the computer science department .

- **Course Topics** :

  - Basic concepts and definitions; Set operations; partition of a set
  - Equivalence relations; Properties on relation on set.
  - Proving Equivalences about Sets.
  - Central concepts of Automata Theory.
  - Regular Expressions; Operations on Regular expressions
  - Finite Automata and Regular Expressions.
  - Deterministic Finite Automata (DFA)
  - Non-Deterministic Finite Automata (NDFA)
  - Equivalence of Deterministic and Non-Deterministic Finite Automata.
  - Deterministic Finite Automata (DFA); Minimization of DFA.
  - Finite Automata with Epsilon-Transition. Equivalence between DFA, NFA, NFA-Λ
  - Conversion from FA and regular expressions.
  - Mealy and Moore Machines
  - Conversion between mealy and Moore machines
  - Kleen's Theory

- **Course Description** : Finite automata are useful models for many important kinds of hardware and software. Here are the most important kinds: Software for designing and checking the behavior of digital circuits; The "lexical analyzer" of a typical complier, that is, the compiler component that breaks the input text into logical units, such as identifiers, keywords, and punctuation; Software for scanning large bodies of text, such as collections of Web pages, to find occurrences of words, phrases, or other patterns; Software for verifying systems of all types that have a finite number of distinct states, such as communication protocols or protocols for secure exchange of information.
- **Course Outcomes:** - Acquire a full understanding and mentality of Automata Theory as the basis of all computer science

languages design

- Have a clear understanding of the Automata theory concepts such as RE's, DFA's, NFA's, Stack's, Turing machines, and Grammars

• Cognitive skills (thinking and analysis).

- Be able to design FAs, NFAs, Grammars, languages modelling, small compilers basics
- Be able to design sample automata

• Communication skills (personal and academic).

- Be able to minimize FA's and Grammars of Context Free Languages
- • Practical and subject specific skills (Transferable Skills).
- **Recommended Textbook(s):** Daniel L. A. Cohen, Introduction of the theory of computation.
- **Prerequisites:** CSDC210
- **Lab. Topics :**

CSDE213 Data Base programming (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- SQL DDL -Data Definition Language
- SQL Data Manipulation Language
- SQL DCL -Data Control Language
- DQL -Data Query Language
- Decomposition
- Introduction to normalization theory
- Keys and Functional Dependencies
- Lossless Decomposition and Functional Dependencies

- Boyce–Codd Normal Form (BCNF)
- BCNF and Dependency Preservation
- Third Normal Form
- Functional-Dependency Theory
- Dependency preservation
- Canonical cover
- Preparation for final exam

- **Course Description** :
  This course introduces students to the fundamental concepts and skills related to managing data within computer systems. This includes introduction to relational database, modeling, and normalization. Also, this course will introduce the structured query language (SQL), which is the standard language for relational database management systems (RDBMS)

- **Course Outcomes:**

1. Apply the basic concepts of Database Systems and Applications.
2. Use the basics of SQL and construct queries using SQL in database creation and interaction.
3. Design a commercial relational database system (Oracle, MySQL) by writing SQL using the system.
4. Analyze and Select storage and recovery techniques of database system.

- **Recommended Textbook(s):** database System concepts 7th edition
- **Prerequisites:** CSDC209
- **Lab. Topics:**

- Creating schema
- Create, modify, and update schema.
- Control user’s privilege to schema
- Retrieve one or more records from the schema
- Decomposing tables into multiple tables.
- Practicing the join query from the SQL
- Creating primary keys
- Testing select queries on decomposed tables
- Create tables from other tables based on given dependency.

- Exam the new tables using the select query based on given functional dependency
- Create tables from other tables based on given dependency and candidate keys.
- Design GUIs for making user queries
- Link the GUI to the database
- Testing more sql queries through the GUI
- Lab test

CSDE206 Gaming Programming (3-2-0-2)

- Designation as a „required" or „elective" course:
This is elective course for the computer science department .

- **Course Topics :**
  - Introduction to Game Programming
  - How to design a Game
  - Game Physics - The main parts of game physics
  - Fortnite and Battlefield – what are these games, who developed them
  - Minecraft and animal crossing, what are these games, who developed them
  - Game Graphics – the concept of pixel, color, resolutions, and others.
  - Mid-term Exam
  - Game Design – texture mapping, lighting, rasterization, and others.
  - Unity Game Engine - Colliders and Tile maps
  - Unity Engine - Layer-Based Collision Detection and Player Collisions
  - Unity Engine - Health and Inventory
  - Unity Engine - Characters, Coroutines, and Spawn Points
  - Game programming with Artificial Intelligence
  - Artificial Intelligence – Algorithms and Procedures
  - Artificial Intelligence – Smart Games
- **Course Description** :
  Presenting all concepts of game programming. Including an interactive topic that covers important points about game programming. In addition to that doing some reports related to these topics . Game Programming challenges: Showing different sides of game programming abilities such as game graphics, game physics, game sprites and more. Game Programming tools – discussing different types of game engies. How to render images, and how to design games. Learning tools that can accelerate building games.
- **Course Outcomes:**
  Important: Write at least 6 Learning Outcomes, better to be equal to the number of study

weeks.

1. Demonstrate a solid understanding of the fundamentals of game programming and design principles.
2. Apply programming concepts and techniques to develop game mechanics and functionality.
3. Utilize game development tools and engines (e.g., Unity, Unreal Engine, or Godot) to create and prototype games.
4. Implement physics simulations and realistic behaviors in games, including collision detection and response.
5. Design and develop intelligent game characters.

   Create intuitive and visually appealing user interfaces (UI) that enhance the overall player experience.

- **Recommended Textbook(s):** Unity in Action: Multiplatform Game Development in C#" by Joe Hocking
- **Prerequisites:**
- **Lab. Topics:**

- Adding different Objects
- Use Controller to Move Objects
- Build Your Playground
- Project Game
- Follow on Project Game
- Follow on Project Game
- Discussion Project Game

CCIT063 Object Oriented Programming II (4-4-0-2)

- Designation as a „required" or „elective" course:

This is a required course for the computer science department .

- **Course Topics   :**
  - Introduction to Game Programming
  - How to design a Game
  - Game Physics - The main parts of game physics
  - Fortnite and Battlefield – what are these games, who developed them
  - Minecraft and animal crossing, what are these games, who developed them
  - Game Graphics – the concept of pixel, color, resolutions, and others.

- Mid-term Exam
- Game Design – texture mapping, lighting, rasterization, and others.
- Unity Game Engine - Colliders and Tile maps
- Unity Engine - Layer-Based Collision Detection and Player Collisions
- Unity Engine - Health and Inventory
- Unity Engine - Characters, Coroutines, and Spawn Points
- Game programming with Artificial Intelligence
- Artificial Intelligence – Algorithms and Procedures
- Artificial Intelligence – Smart Games

- **Course Description** :
  There are different teaching and learning activities including lectures and laboratories. The concepts, process, and applications of data science will be discussed in lectures. Students will also learn computer programming knowledge and the skills of manipulating, processing, retrieving, storing, and plotting data. Students will develop small programs and learn different in laboratories.
  Object Oriented Design by Rumbaugh (Pearson publication)
- **Course Outcomes:**

- Explain the motivation for and development of object-oriented programming languages.
- Produce a set of use cases given a problem statement.
- Produce class diagrams, object interaction diagrams and object state transition diagrams for a given problem.
- Describe the essential features of an object-oriented programming language.
- Produce and/or debug code fragments that illustrate principles of object-oriented software development.
- Describe the principles for testing object-oriented software and derive sets of test data given a specification.

- **Recommended Textbook(s):** Object-oriented programming with C++ by E.Balagurusamy, 2nd Edition, TMH.

- **Prerequisites: CCIT061**
- **Lab. Topics:**

Lab 1: develop a program to implement 1 dimension array

Lab 2: develop a program to perform matrix operation using multi-dimensions array

Lab 3: develop program that implement a class and use it with objects

Lab 4: develop program that implement a class and create array of objects

Lab 5: write program for single inherence

Lab 6: write program for hybrid inherence

Lab 7: write program for multi-inheritance

Lab 8: write code for friend function and class

Lab 9: write program for pointer object

Lab 10: write code for polymorphism

Lab 11: write code for exception handling

Lab 12: write code for try, catch, and throw

Lab 13: write program for interface

Lab 14: write code for file

CSDC208 Algorithms (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- Program cost and asymptotic analysis
- Sorting and searching
- Graph traversal (DFS, BFS) and applications
- Shortest path
- Hard problems
- Dynamic Programming
- Algorithm correctness
- Time and space complexity
- Asymptotic analysis: Big Oh, Little oh, Theta
- Mid exam
- NP-algorithms
- Greedy algorithms
- Limitations of Algorithmic Power
- Heuristic and Approximate Algorithms
- B-tree
- Preparatory week before the final Exam

- **Course Outcomes**

1. Determine the characteristics of complexity classes and evaluate algorithms in terms of time and space complexity.
2. Choose among the major algorithmic techniques the most appropriate to solve a given problem

including discussion of space and time trade-offs.
3. Develop the appropriate algorithms and relevant data structures for graph processing.

- **Recommended Textbook(s):**
  - Algorithms, 4th Edition, 2011
  - Robert Sedgewick, Princeton University, Kevin Wayne
  - Data Structures and Algorithms in Java™, Sixth Edition, Michael T. Goodrich
- **Prerequisites:**
- **Lab. Topics:**
  - Program cost and asymptotic analysis
  - Sorting and searching
  - Graph traversal (DFS, BFS) and applications
  - Shortest path
  - Hard problems
  - Dynamic Programming
  - Algorithm correctness
  - Time and space complexity
  - Asymptotic analysis: Big Oh, Little oh, Theta
  - Mid exam
  - NP-algorithms
  - Greedy algorithms
  - Limitations of Algorithmic Power
  - Heuristic and Approximate Algorithms
  - B-tree

CCIT062 Numerical Analysis (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**
  - Direct methods for solving linear system of equation

- Simple Gaussian elimination method, gauss elimination method with partial pivoting,
- determinant evaluation, gauss Jordan method,
- L U decompositions Doolittle’s LU decomposition, Doolittle’s method with row interchange
- Finding Matrix Inverse
- Iterative methods for solving linear systems of equations
- Jacobin iteration, gauss – seidel method,
- Successive over relaxation method (sort method
- Mid-term Exam
- Newton-Raphson Method
- Runge-kutta Method
- Interpolation and the Lagrange Polynomial, Data Approximation and Neville’s Method
- Numerical Analysis Methods for Differential Equation
- Final Exam

- **Course Description** :

- Introduction to numerical methods for solving mathematical problems.
- Learn algorithms for approximation, interpolation, and root-finding.
- Explore numerical techniques for integration and differentiation.
- Study numerical solutions for linear and nonlinear systems of equations.
- Understand error analysis and convergence of numerical methods.

- **Course Outcomes**

1. Determine the characteristics of complexity classes and evaluate algorithms in terms of time and space complexity.
2. Choose among the major algorithmic techniques the most appropriate to solve a given problem including discussion of space and time trade-offs.
3. Develop the appropriate algorithms and relevant data structures for graph processing.

- **Recommended Textbook(s):**

- Richard L. Burden and etc." Numerical Analysis ", 9th edition, 2014

- **Prerequisites:**
- **Lab. Topics:**

  Lab 1: Introduction to Agilent VEE and PSPICE

  Lab 2: Thévenin's / Norton's Theorem and Kirchhoff's Laws

  Lab 3: First-Order Transient Responses

  Lab 4: Second-Order Transient Responses

  Lab 5: Frequency Response of RC Circuits

  Lab 6: Frequency Response of RLC Circuits

  Lab 7: Filters

UOA232 English 2 (2-2-0-0)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics** :
  - Getting to know you
  - The way we live
  - It all went wrong
  - Let’s go shopping
  - What do you want to do?
  - Tell me! What’s it like?
  - Fame
  - Do’s and don’ts
  - Going places
  - Scared to death
  - Things that changed the world
  - Dreams and reality
  - Earning a living
  - Family ties
  - Exam

- **Course Description** :

Advanced Reading Skills:

• Analyzing and interpreting complex texts, including literary works, academic articles, and media sources.

• Developing strategies for effective reading comprehension, such as skimming, scanning, and note-taking.

- Identifying main ideas, supporting details, and implicit meanings in texts.
- Evaluating the credibility and validity of sources.

2. Writing Proficiency:

- Developing advanced writing skills, including essay structure, argumentation, and organization.
- Enhancing grammar and sentence structures for clarity and coherence.
- Conducting research and integrating credible sources into written work.
- Refining editing and proofreading techniques for error-free writing.

3. Oral Communication:

- Delivering engaging and persuasive presentations on various topics.
- Participating in debates and discussions, expressing and defending opinions.
- Improving pronunciation, intonation, and fluency in spoken English.
- Enhancing active listening skills and responding appropriately to others.

4. Grammar and Vocabulary:

- Reviewing and reinforcing advanced grammar concepts, such as complex sentence structures, verb forms, and conditional clauses.

- **Course Outcomes**

1. Reviewing the fundamental rules of the English language.
2. Developing the student's skills in formal and informal writing in the English language.
3. Adding new vocabulary from the language.
4. Improving reading skills
5. Writing in English formally and informally.
6. Improving speaking skills in the English language.
7. Enhancing English grammar skills.

- **Recommended Textbook(s):**
- New Headway Plus Intermediate, John and Liz Soars, Oxford University Press, 2006..

Prerequisites:

- **Lab. Topics:**

CCIT064 Visual Programming I (3-2-0-2)

- Designation as a „required‟ or „elective‟ course:
This is a required course for the computer science department .

- **Course Topics :**

  - Introduction to C#:
    Overview of C# programming language
    Setting up the development environment
    Basic syntax and structure of a C# program

- Data types, variables, and operators in C#
- Control structures (loops, conditional statements)
- Classes and objects
  Object-Oriented Programming in C#
- Classes and objects
  Constructors and destructors
  Access modifiers and properties
- Arrays and collections, Classes and objects, Method overloading and overriding
- Mid-term Exam
- File handling and I/O operations
- Exception handling
- Introduction to Windows Forms or WPF (Windows Presentation Foundation)
- Event-driven programming
- Creating and designing GUI elements (buttons, labels, textboxes, etc.)
- Handling user input and validation
- Implementing menus, dialog boxes, and other GUI components
- Introduction to databases and SQL
- Connecting and interacting with databases
- Project Work and Case Studies:
  Hands-on coding exercises and projects
  Implementing real-world scenarios using C#

- Course Description :

1. Demonstrate a solid understanding of the basic concepts of C# programming language, including syntax, data types, variables, control structures, and functions.
2. Apply object-oriented programming principles in C# to design and implement software solutions, including the use of classes, objects, inheritance, encapsulation, and polymorphism.
3. Develop and debug C# programs using appropriate programming techniques and tools, effectively identifying and fixing errors in code.
4. Utilize C# language features and libraries to perform input/output operations, handle exceptions, and manage files and data.
5. Create graphical user interfaces (GUIs) using C# and relevant frameworks, implementing event handling, user input validation, and visual design principles.

- Course Outcomes

1. Demonstrate a solid understanding of the basic concepts of C# programming language, including syntax, data types, variables, control structures, and functions.

2. Apply object-oriented programming principles in C# to design and implement software solutions, including the use of classes, objects, inheritance, encapsulation, and polymorphism.
3. Develop and debug C# programs using appropriate programming techniques and tools, effectively identifying and fixing errors in code.
4. Utilize C# language features and libraries to perform input/output operations, handle exceptions, and manage files and data.
5. Create graphical user interfaces (GUIs) using C# and relevant frameworks, implementing event handling, user input validation, and visual design principles.
6. Employ C# programming techniques to interact with databases, including data retrieval, manipulation, and storage using ADO.NET or other relevant technologies.
7. Develop web applications using C# and frameworks such as ASP.NET, understanding concepts like HTTP requests and responses, session management, and database integration.
8. Apply best practices in coding style, documentation, and software development methodologies to write clean, efficient, and maintainable C# code.
9. Demonstrate an understanding of advanced topics in C# programming, such as multithreading, asynchronous programming, LINQ, and other advanced language features.
10. Analyze and solve programming problems using critical thinking and problem-solving skills, translating requirements into effective C# code solutions.
11. Collaborate effectively in a team environment, demonstrating the ability to communicate and work with others on C# programming projects.

- **Recommended Textbook(s): Agile Principles, Patterns, and Practices in C#,** W3Schools Online Web Tutorials
- **Prerequisites:**
- **Lab. Topics:**

• Introduction to C#:

Overview of C# programming language

Setting up the development environment

Basic syntax and structure of a C# program

• Data types, variables, and operators in C#

• Control structures (loops, conditional statements)

• Classes and objects

Object-Oriented Programming in C#

• Classes and objects

Constructors and destructors

Access modifiers and properties

• Arrays and collections, Classes and objects, Method overloading and overriding

- Mid-term Exam
- File handling and I/O operations
- Exception handling
- Introduction to Windows Forms or WPF (Windows Presentation Foundation)
- Event-driven programming
- Creating and designing GUI elements (buttons, labels, textboxes, etc.)
- Handling user input and validation
- Implementing menus, dialog boxes, and other GUI components
- Introduction to databases and SQL
- Connecting and interacting with databases
- Project Work and Case Studies:

Hands-on coding exercises and projects

Implementing real-world scenarios using C#

CSDC308 Computer Graphics 2D (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**
  - Introduction to Computer Graphics
  - Elements of pictures created in computer graphics
  - Graphics display devices
  - Raster Graphics And
  - Vector Graphics
  - Drawing Algorithms: Plotting Points
  - Line Drawing Algorithms: Naive Line-Drawing Algorithm, and DDA
  - Bresenham Line Drawing Algorithm
  - Mid-term Exam
  - Circle Drawing Algorithms: Direct Algorithm and DDA
  - Bresenham Circle Drawing Algorithm
  - Ellipses Drawing Algorithms
  - Two Dimensional Geometric Transformations:
  - Translation and Scaling with various examples
  - Rotations with various examples

- Shearing and Reflection with various examples
- Final Exam

- **Course Description** : The student's acquisition of the fundamental of computer graphics such as point, pixel, line, polygons, and objects operations such as translation, rotation, scaling and shearing. Then, advanced topic different types of arrays and function are clarified.

- **Course Outcomes**

1. Understanding of fundamental concepts: Demonstrate a solid understanding of the fundamental concepts and principles of computer graphics, including digital image representation, rasterization, vector graphics, and the graphics pipeline.
2. Graphics programming skills: Develop practical programming skills in implementing computer graphics algorithms and rendering techniques using appropriate programming languages or graphics APIs.
3. 2D and 3D transformations: Apply various 2D and 3D transformations to manipulate and animate objects in a virtual scene, including translation, rotation, scaling, and shearing.
4. Rendering techniques: Apply different rendering techniques, such as flat shading, Gouraud shading, and Phong shading, to simulate the behavior of light and achieve realistic rendering of 3D objects.
5. Graphics algorithms: Implement and apply graphics algorithms, such as line-drawing algorithms, polygon filling algorithms, and hidden surface removal techniques, to generate and render computer-generated images efficiently.

Indicative content includes the following.

1.Introduction to 2D Computer Graphics:
- Overview of 2D computer graphics and its applications
- Basic concepts of pixels, coordinates, and color representation
- Graphics programming environment setup

2. 2D Drawing Algorithms:
- Line drawing algorithms (e.g., DDA algorithm, Bresenham's line algorithm)
- Circle drawing algorithms (e.g., midpoint circle algorithm)
- Ellipse drawing algorithms (e.g., midpoint ellipse algorithm)

3. Geometric Transformations:
- 2D translation, rotation, scaling, and shearing transformations
- Matrix representation of transformations
- Composite transformations and hierarchical transformations

4. Clipping and Windowing:
- Line clipping algorithms (e.g., Cohen-Sutherland, Liang-Barsky)
- Polygon clipping algorithms (e.g., Sutherland-Hodgman)
- Windowing and viewport transformations

5. Color and Shading:

- Color models and color spaces
- Color interpolation and shading techniques (e.g., flat shading, Gouraud shading)
- Anti-aliasing techniques for smoother edges

6. 2D Image Manipulation:

- Image representation and file formats
- Image filtering and convolution operations (e.g., blurring, sharpening)
- Image transformations (e.g., rotation, scaling, flipping)

7. Geometric Primitives and Curves:

- Representation and rendering of geometric primitives (e.g., points, lines, polygons)
- Bezier curves and B-spline curves
- Interpolation and approximation techniques for curves

8. Bitmap and Vector Graphics:

- Understanding the differences between bitmap and vector graphics
- Bitmap image manipulation and editing techniques
- Vector graphics representation and manipulation

9. Practical projects involving the implementation of 2D graphics techniques

Case studies of real-world applications of 2D computer graphics

- **Recommended Textbook(s):** Shirley, Peter, Michael Ashikhmin, Steve Marschner. Fundamentals of Computer Graphics. 3rd ed. A K Peters/CRC Press, 2009. ISBN: 9781568814698

- **Prerequisites:**
- **Lab. Topics:**

- Lecture Programs
- Lecture Programs
- Lecture Programs
- Lecture Programs
- Lecture Programs
- Lecture Programs
- Lecture Programs
- Lecture Programs
- Lecture Programs
- Lecture Programs
- Lecture Programs

CSDC307 Computer Architecture(2-2-0-0)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

  - Introduction to computer components and historical review
  - Data representation in computer system
  - Error detection and correction
  - Boolean algebra and digital logic
  - MARIE: an introduction to simple computer
  - MARIE: The Architecture
  - Instruction Set Architecture
  - Instruction Types
  - Memory (1)
  - Memory (2)
  - Input/output storage system
  - System Software
  - Alternative Architecture
  - Embedded System
  - Performance Measurement and Analysis
  - Final Exam

- **Course Description** :

  1.To understand the structure, function and characteristics of computer systems.
  2. To understand the design of the various functional units and components of computers.
  3. To identify the elements of modern instructions sets and their impact on processor design.
  4. To explain the function of each element of a memory hierarchy,
  5. To identify and compare different methods for computer I/O.

- **Course Outcomes :**This course introduces Machine Architecture with coverage of digital logic, machine level data and instruction representation, ALU design, and organization of the processor data path and control. Examines performance analysis, memory system hierarchy, pipelining, and communication.

  - **Recommended Textbook(s):** Shirley, Peter, Michael Ashikhmin, Steve Marschner. Fundamentals of Computer Graphics. 3rd ed. A K Peters/CRC Press, 2009. ISBN: 9781568814698

- **Prerequisites:**

- **Lab. Topics:**

CSDC305 Computer Networks I (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- Chapter: 1 Introduction
  1.1 DATA COMMUNICATIONS
  Components, Data Representation, Data Flow
- 1.2 NETWORKS
  Distributed Processing , Network Criteria, Physical Structures, Network Components: NIC, Repeater HUB, Bridge, Router, BRouter, GATEWAY
- 1.2 NETWORKS
  Distributed Processing , Network Criteria, Physical Structures, Network Components: NIC, Repeater HUB, Bridge, Router, BRouter, GATEWAY

- 1.2 NETWORKS
  Network Models, Categories of Networks,
  Network Classification, LAN, MAN and WAN
  Network topologies: Mesh, Star, Bus and Ring, the advantages and disadvantages of each topology.
  Interconnection of Networks: Internetwork

- 1.2 NETWORKS
  Network Models, Categories of Networks,
  Network Classification, LAN, MAN and WAN
  Network topologies: Mesh, Star, Bus and Ring, the advantages and disadvantages of each topology.
  Interconnection of Networks: Internetwork

- 1.3 THE INTERNET
  A Brief History, The Internet Today
- 1.4 PROTOCOLS AND STANDARDS
  Protocols , Standards, Standards Organizations, Internet Standards

- Chapter: 2 Network Models
  2.1 LAYERED TASKS
  Sender, Receiver, and Carrier , Hierarchy
  - 2.2 THE OSI MODEL
  Layered Architecture, Peer-to-Peer Processes, Encapsulation
  - 2.2.1 LAYERS IN THE OSI MODEL
  Physical Layer, Data Link Layer, Network Layer, Transport Layer, Session

Layer, Presentation Layer, Application Layer, Summary of Layers

- Chapter: 2 Network Models
  2.1 LAYERED TASKS
  Sender, Receiver, and Carrier , Hierarchy
  - 2.2 THE OSI MODEL
  Layered Architecture, Peer-to-Peer Processes, Encapsulation
  - 2.2.1 LAYERS IN THE OSI MODEL
  Physical Layer, Data Link Layer, Network Layer, Transport Layer, Session Layer, Presentation Layer, Application Layer, Summary of Layers

- Chapter: 2 Network Models
  2.1 LAYERED TASKS
  Sender, Receiver, and Carrier , Hierarchy
  - 2.2 THE OSI MODEL
  Layered Architecture, Peer-to-Peer Processes, Encapsulation
  - 2.2.1 LAYERS IN THE OSI MODEL
  Physical Layer, Data Link Layer, Network Layer, Transport Layer, Session Layer, Presentation Layer, Application Layer, Summary of Layers

- 2.3 TCP/IP PROTOCOL SUITE
  Physical and Data Link Layers, Network Layer Transport Layer, Application Layer

- 2.3 TCP/IP PROTOCOL SUITE
  Physical and Data Link Layers, Network Layer Transport Layer, Application Layer

- 2.4 ADDRESSING
  Physical Addresses, Logical Addresses, Port Addresses , Specific Addresses

- PART 2: Physical Layer and Media
  Chapter : 3 Data and Signals
  3.1 ANALOG AND DIGITAL
  Analog and Digital Data, Analog and Digital Signals, Periodic and Nonperiodic Signals
  - 3.2 PERIODIC ANALOG SIGNALS
  Sine Wave, Phase, Wavelength, Time and Frequency Domains, Composite Signals, Bandwidth
  3.3 DIGITAL SIGNALS
  Bit Rate, Bit Length, Digital Signal as a Composite Analog Signal, Transmission of Digital Signals

- 3.4 TRANSMISSION IMPAIRMENT
  Attenuation , Distortion, Noise

3.4.1 DATA RATE LIMITS
Noiseless Channel: Nyquist Bit Rate, Noisy Channel: Shannon Capacity, Using Both Limits

- Chapter: 4 Transmission Media
  4.1 GUIDED MEDIA
  Twisted-Pair Cable, Coaxial Cable, Fiber-Optic Cable
  4.2 UNGUIDED MEDIA: WIRELESS
  Radio Waves, Microwaves, Infrared

- **Course Description** : This course is to provide students with an overview of the concepts and fundamentals of data communication and computer networks. Topics to be covered include: data communication concepts and techniques in a layered network architecture, communications switching and routing, types of communication, network congestion, network topologies, network configuration and Management, network model components, layered network models (OSI reference model, TCP/IP networking architecture) and their protocols, various types of networks (LAN, MAN, WAN and Wireless networks) and their protocols.

- **Course Outcomes :**

1.Understand the fundamental concepts and principles of computer networks, including network architectures, protocols, layers, and networking technologies.
2.Explain the functions and interactions of various network layers, including the physical layer, data link layer, network layer, transport layer, and application layer.
3.Demonstrate knowledge of network addressing and routing, including IP addressing, subnetting, and routing algorithms.
4.Configure and troubleshoot network devices, such as routers, switches, and firewalls.
5.Analyze and evaluate network performance and identify and resolve network-related issues and bottlenecks.
6.Design and implement a local area network (LAN) or a wide area network (WAN), considering factors such as network topology, security, and scalability.
7.Understand the principles and protocols of wireless networking, including Wi-Fi and cellular networks.
8.Evaluate network security risks and implement appropriate security measures, including authentication, encryption, and intrusion detection systems.
9.Demonstrate knowledge of network management and monitoring techniques, including network monitoring tools and protocols.

- **Recommended Textbook(s):**

- Data Communications and Networking, 3, 4 /e, Behrouz A Forouzan

- **Prerequisites:**

- **Lab. Topics:**

Lab1: Comm. Sys.

Lab2: Simulator :Comm. Sys.

Lab2: Simulator :Comm. Sys.

Lab2: Simulator :Comm. Sys

Lab2:Simulator:Comm. Sys

Lab3:Network Components

Lab3:Network Components

Lab3:Network Components

Lab3:Network Components

Lab4:Network Topology

Lab4:Network Topology

Lab4:Network Topology

Lab4:Network Topology

Lab5:Cabling

Lab5:Cabling

CSDC310 Compilers I (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

Introduction to Compilers

- Overview of compilers and their role in software development
- Compilation process and phases
- Types of compilers

Lexical Analysis

- Tokenization and regular expressions
- Lexical analyzer design and implementation
- Scanning techniques

Syntax Analysis (Part 1)

- Context-free grammars and parsing techniques
- Top-down parsing

Syntax Analysis (Part 2)

- Bottom-up parsing
- Parser generators

Semantic Analysis (Part 1)
- Symbol tables and identifier management
- Type systems and type checking

Semantic Analysis (Part 2)
- Attribute grammars and semantic actions
- Static analysis and error detection

Mid-term Exam + Static analysis and error detection

Intermediate Code Generation
- Intermediate representations
- Syntax-directed translation and code generation

Control Flow Analysis and Optimization
- Control flow analysis
- Basic blocks
- Data-flow analysis and optimization

Code Optimization (Part 1)
- Principles of Optimization
- Common optimization techniques
- Local code optimization at the intermediate representation level
- Global Optimization Methods

Code Optimization (Part 2)
- Loop optimization
- Register allocation and instruction scheduling

Code Generation
- Target machine models and instruction sets
- Instruction selection and mapping

Memory Management and Runtime Support
- Addressing modes
- Memory management
- Runtime support for generated code

Compiler Testing and Debugging
- Testing strategies for compilers
- Compiler validation techniques
- Debugging and error handling in compilers
-

Advanced Topics
- Just-in-time (JIT) compilation
- Language-specific optimizations

Preparatory week before the final Exam

- **Course Description** :

1. Understand the fundamental concepts of compiler design: Students should be able to comprehend the basic principles, techniques, and components involved in designing and implementing compilers.

2. Analyze and describe the various phases of a compiler: Students should be able to explain the different phases of a compiler, including lexical analysis, syntax analysis, semantic analysis, intermediate code generation, optimization, and code generation.

3. Implement a compiler: Students should gain practical experience by implementing a simple compiler for a programming language. This may involve designing and developing the lexical analyzer, parser, semantic analyzer, and code generator.

4. Apply formal language theory: Students should understand formal languages, regular expressions, context-free grammars, and automata theory, and be able to apply this knowledge to analyze and manipulate programming languages.

Course Outcomes :

1. Understand the fundamental concepts of compiler design: Students should be able to comprehend the basic principles, techniques, and components involved in designing and implementing compilers.
2. Analyze and describe the various phases of a compiler: Students should be able to explain the different phases of a compiler, including lexical analysis, syntax analysis, semantic analysis, intermediate code generation, optimization, and code generation.
3. Implement a compiler: Students should gain practical experience by implementing a simple compiler for a programming language. This may involve designing and developing the lexical analyzer, parser, semantic analyzer, and code generator.
4. Apply formal language theory: Students should understand formal languages, regular expressions, context-free grammars, and automata theory, and be able to apply this knowledge to analyze and manipulate programming languages.
5. Perform lexical and syntactic analysis: Students should be able to develop lexical analyzers and parsers to break down the source code into meaningful tokens and construct the corresponding parse tree or abstract syntax tree.
6. Conduct semantic analysis: Students should learn how to perform semantic analysis, including type checking, symbol table management, and static analysis techniques to ensure program correctness and identify potential errors.
7. Understand intermediate representations: Students should become familiar with various intermediate representations used in compilers, such as three-address code, abstract syntax trees, and control flow graphs. They should understand how to manipulate and optimize these representations.
8. Apply optimization techniques: Students should learn about common compiler optimization techniques, such as constant folding, common subexpression elimination, loop

optimization, and register allocation. They should be able to apply these techniques to improve the efficiency of generated code.

9. Generate Low Level Language: Students should understand the process of generating machine code or assembly language from the intermediate representation. They should be able to apply code generation algorithms and handle low-level details such as instruction selection and addressing modes.

10. Test and debug compilers: Students should develop skills in testing and debugging compilers. They should be able to identify and fix errors in the compiler implementation and evaluate the correctness and performance of generated code.

11. Stay updated with current compiler trends: Students should be aware of recent developments and trends in the field of compiler design, including just-in-time (JIT) compilation, language-specific optimizations, and parallelizing compilers.

- **Recommended Textbook(s):** A.Aho,R.Sethi,J.D.Ullman," **Compilers- Principles, Techniques and Tools**"Addison-Weseley,2007
- **Prerequisites:**
- **Lab. Topics:**

Introduction to Compiler Tools and Setup
• Introduction to compiler development tools.
Setting up the development environment for compiler labs
Preprocessor Lab
• Implement Macros
• Eliminate Comments
Eliminate White Spaces
Lexical Analysis Lab
• Implementing a lexical analyzer using Lex or a similar tool
Testing and validating the lexical analyzer with sample inputs
Syntax Analysis Lab
• Implementing a recursive descent parser or a bottom-up parser
Constructing and analyzing parse trees for sample inputs
Semantic Analysis Lab
• Building a symbol table and performing type checking
Handling semantic errors and reporting them in the compiler
Intermediate Code Generation Lab
• Generating intermediate code (e.g., three-address code)
Implementing basic optimizations at the intermediate representation level
Code Generation Lab
• Mapping intermediate code to target machine instructions
Handling memory management and addressing modes in code generation

CSDC307 Mobile Applications Programming (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- Introduction to Mobile App Development
  - Mobile Phone Family
  - Reasons for using mobile application programming
- Advantages of Mobile Application Programming
  - Disadvantages of Mobile Application Programming
  - The future trends of mobile application programming
  - Wireless Technologies and Architectures
- Flexible Mobile Phone I
- Flexible Mobile Phone II
- Mobile App Data Management (Local Storage, Databases)
- Short-Range Communication Systems
- Mid-term Exam + Navigation Patterns in Mobile Apps
- Wireless Technologies and Architectures
- Mobile App Multimedia Integration (Images, Audio, Video)
  - Device Sensors and Integration
- Mobile App Security and Privacy
  - User Authentication and Authorization
- Mobile Device Management (MDM)
  - Mobile Device Management Works
- Cross-Platform Development Techniques
- Location-Based Services (LBS)
  - Types of Location-Based Services
- Augmented Reality (AR) and Virtual Reality (VR)
  - AR in Mobile Apps

- Challenges and Considerations
- Future Trends

- Operating Systems in Mobile Application Programming

- **Course Description** :

Mobile Applications Programming is a comprehensive course designed to provide students with the knowledge and skills required to develop mobile applications for various platforms. Throughout the course, students will delve into the intricacies of mobile app development, covering fundamental concepts, platform architecture, user interface design, and the application lifecycle.

The course begins with an exploration of the foundational principles of mobile app development, including understanding platform architectures and the essential components of user interfaces. Students will learn to navigate the complexities of different mobile device families and operating systems, gaining insights into the nuances of each platform to develop applications that cater to diverse user bases.

A key focus of the course is to familiarize students with the motivations for choosing mobile app programming and the advantages it offers, both personally and professionally. By understanding industry demands and trends, students will be better equipped to pursue careers in mobile app development.

As challenges are inevitable in this field, students will learn to identify common issues and develop effective strategies to overcome them. Emphasis will be placed on addressing security concerns, implementing user authentication mechanisms, and ensuring cross-platform compatibility to deliver robust and reliable mobile applications.

Moreover, the course will keep students abreast of the latest trends and technologies in the rapidly evolving landscape of mobile app development. Through hands-on projects and practical exercises, students will gain proficiency in wireless technologies, multimedia integration, data management, and cross-platform development techniques.

By the end of the course, students will have acquired the necessary skills to design, develop, and deploy mobile applications that meet industry standards and user requirements. Whether aspiring to work as independent developers or within established software development teams, students will be well-prepared to contribute effectively to the ever-expanding mobile app ecosystem.

Course Outcomes :

Students will gain a comprehensive understanding of the core concepts and principles of mobile application development, which includes delving into the intricacies of platform architecture, the art of designing user-friendly interfaces, and comprehending the entire lifecycle of mobile applications. This foundation is essential for creating highly effective mobile applications. Moreover, students will become proficient in recognizing and distinguishing among various mobile device families and the operating systems they employ. This proficiency enables them to develop applications that cater to a broad spectrum of user needs, considering the differences and similarities among different platforms.Understanding the motivations and advantages of choosing a career in mobile app programming is emphasized, enabling students to appreciate the personal and professional benefits while recognizing the high demand for mobile applications in the industry.

- **Recommended Textbook(s):** Fitzek, Frank HP, and Frank Reichert, eds. Mobile phone programming: and its Application to Wireless Networking. Springer Science & Business Media, 2007.
- **Prerequisites:**
- **Lab. Topics:**
    - Mobile Operating Systems
    - Setting up Development Environment (Android Studio)
    - Creating a "Hello World" Mobile App
    - User Interface Design in Mobile Apps (Layouts, Views)
    - Building a Simple User Interface
    - Implementing User Interaction and Event Handling
    - Adding Buttons, Text Input, and Image Views
    - Data Storage and Retrieval in Mobile Apps
    - Working with SQLite Databases
    - Networking and Web Services in Mobile Apps I
    - Networking and Web Services in Mobile Apps II
    - Multimedia Integration in Mobile Apps
    - Displaying Images and Playing Audio/Video I
    - Displaying Images and Playing Audio/Video II
    - Implementing User Authentication and Authorization I
    - Implementing User Authentication and Authorization II
    - Login and Registration Functionality

UOA301 English 3 (2-2-0-0)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics** :

  - Getting to know you
  - The way we live
  - It all went wrong
  - Let's go shopping
  - What do you want to do?
  - Tell me! What's it like?
  - Fame
  - Do's and don'ts
  - Going places
  - Scared to death
  - Things that changed the world
  - Dreams and reality
  - Earning a living
  - Family ties
  - Exam

- **Course Description** :

Advanced Reading Skills:

• Analyzing and interpreting complex texts, including literary works, academic articles, and media sources.

• Developing strategies for effective reading comprehension, such as skimming, scanning, and note-taking.

• Identifying main ideas, supporting details, and implicit meanings in texts.

• Evaluating the credibility and validity of sources.

2. Writing Proficiency:

• Developing advanced writing skills, including essay structure, argumentation, and organization.

• Enhancing grammar and sentence structures for clarity and coherence.

• Conducting research and integrating credible sources into written work.

• Refining editing and proofreading techniques for error-free writing.

3. Oral Communication:

• Delivering engaging and persuasive presentations on various topics.

• Participating in debates and discussions, expressing and defending opinions.

• Improving pronunciation, intonation, and fluency in spoken English.

• Enhancing active listening skills and responding appropriately to others.

4. Grammar and Vocabulary:

• Reviewing and reinforcing advanced grammar concepts, such as complex sentence structures, verb forms, and conditional clauses.

- **Course Outcomes**

8. Reviewing the fundamental rules of the English language.
9. Developing the student's skills in formal and informal writing in the English language.
10. Adding new vocabulary from the language.
11. Improving reading skills
12. Writing in English formally and informally.
13. Improving speaking skills in the English language.
14. Enhancing English grammar skills.

- **Recommended Textbook(s):**
- New Headway Plus Intermediate, John and Liz Soars, Oxford University Press, 2006..

Prerequisites:

- **Lab. Topics:**

CCIT065 Visual Programming II (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- Windows Form Application: Form, Buttons, Text Box and LabelBox
- Windows Form Application: Checkbox, RadioButtons, and Message Boxes
- Windows Form Application: ListBox, ComboBox, and PictureBox, Vertical and Horizontal Scrolls and NumericUpDown
- Strings and Characters: Fundamentals of Strings, String Constructors Comparing, Strings Locating Characters and Substrings in strings
- Strings: Extracting Substrings from strings Concatenating strings Miscellaneous string Methods
- Characters: Fundamentals of Characters and Char Methods
- Files: Computer Files, Files Categories Input Files, Outputs Files
- Mid-Term Exam
- Windows Form Application: Timers, Open File Dialog and interaction with Files and images.
- Windows Form Application: Menu Strip, Tool Strip, Status Strip and ProgressBar
- Structures : Fundamentals of Structures, Structures with Constructors
- Structures : Array of Structures, List of Structures
- LINQ Providers : Fundamentals of LINQ, Querying an Array of int Values Using LINQ
- LINQ Providers : LINQ with Structures, Querying an Array of Employee Objects Using LINQ

- Preparatory Week

- **Course Description** : This course explores Visual Programming, with an emphasis on LINQ Principles, Windows Form Applications, String and Character Manipulation, and File Handling. Students will learn how to create interactive graphical user interfaces (GUIs), manage file input/output tasks, manipulate characters and strings, and effectively query data with LINQ. Projects and exercises focus on practical application to enhance competency in these key areas.

- **Course Outcomes**

1. Demonstrate a solid understanding of the basic concepts of C# programming language, including syntax, data types, variables, control structures, and functions.
2. Apply object-oriented programming principles in C# to design and implement software solutions, including the use of classes, objects, inheritance, encapsulation, and polymorphism.
3. Develop and debug C# programs using appropriate programming techniques and tools, effectively identifying and fixing errors in code.
4. Utilize C# language features and libraries to perform input/output operations, handle exceptions, and manage files and data.
5. Create graphical user interfaces (GUIs) using C# and relevant frameworks, implementing event handling, user input validation, and visual design principles.
6. Employ C# programming techniques to interact with databases, including data retrieval, manipulation, and storage using ADO.NET or other relevant technologies.
7. Develop web applications using C# and frameworks such as ASP.NET, understanding concepts like HTTP requests and responses, session management, and database integration.
8. Apply best practices in coding style, documentation, and software development methodologies to write clean, efficient, and maintainable C# code.
9. Demonstrate an understanding of advanced topics in C# programming, such as multithreading, asynchronous programming, LINQ, and other advanced language features.
10. Analyze and solve programming problems using critical thinking and problem-solving skills, translating requirements into effective C# code solutions.
11. Collaborate effectively in a team environment, demonstrating the ability to communicate and work with others on C# programming projects.

- **Recommended Textbook(s):**
- Visual C# How To Program, Paul Deitel and Harvey Deitel, Deitel & Associates, Inc. Pearson, 2018.
- **Prerequisites: CCIT061**
- **Lab. Topics:**
    - Many Programs about Windows Form Application: Form, Buttons, Text Box and LabelBox

- Many Programs about Windows Form Application: Checkbox, RadioButtons, and Message Boxes
- Many Programs about Windows Form Application: ListBox, ComboBox, and PictureBox, Vertical and Horizontal Scrolls and NumericUpDown
- Many Programs about Strings and Characters: String Constructors Comparing, Strings Locating Characters and Substrings in strings
- Many Programs about Strings: Extracting Substrings from strings Concatenating strings Miscellaneous string Methods
- Many Programs about Characters: Char Methods
- Many Programs about Files: Input Files, Outputs Files
- Mid-Term Exam
- Many Programs about Windows Form Application: Timers, Open File Dialog and interaction with Files and images
- Many Programs about Windows Form Application: Menu Strip, Tool Strip, Status Strip and ProgressBar
- Many Programs about Structures : Structures without Constructors Structures with Constructors
- Many Programs about Structures : Array of Structures, List of Structures
- Many Programs about LINQ: Querying an Array of int Values Using LINQ
- Many Programs about LINQ: LINQ with Structures, Querying an Array of Employee Objects Using LINQ
- Preparatory Week

CSDE310 Computer Graphics 3D (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**
  - Introduction to Computer Graphics and 3D Rendering
  - Translation, scaling transformations in 3D
  - Rotation, shearing, and reflection transformations in 3D
  - Implementing 3D transformations in graphics software
  - Projection Transformation : Parallel Projection

- Projection Transformation : Perspective Projection
- Mid-term Exam
- Viewport and window transformations
- Introduction to Clipping: Point Clipping
- Line Clipping
- Cohen–Sutherland Algorithm
- Line Intersections and Clipping
- Polygon Clipping
- Convex and Concave Window
- Final Exam

- **Course Description** : The course aims to introduce students to the fundamental concepts of computer graphics with 3-D, including the modeling, transformation, projection, rasterization, and rendering.

- **Course Outcomes**
  Students will learn about the stages of the graphics pipeline, which involves transforming 3D models into 2D images. This includes understanding concepts such as modeling, transformation, projection, rasterization, and rendering.
- **Recommended Textbook(s):**

Shirley, Peter, Michael Ashikhmin, Steve Marschner. Fundamentals of Computer Graphics.

3rd ed. A K Peters/CRC Press, 2009. ISBN: 9781568814698

Prerequisites: CSDC308

- **Lab. Topics:**

  - Lecture Programs
  - Lecture Programs
  - Lecture Programs
  - Lecture Programs
  - Lecture Programs
  - Lecture Programs
  - Lecture Programs
  - Lecture Programs
  - Lecture Programs
  - Lecture Programs
  - Lecture Programs
  - Lecture Programs
  - Lecture Programs

CSDE308 Internet of Things (3-2-0-2)

- Designation as a „required‟ or „elective‟ course:

This is a required course for the computer science department .

- **Course Topics :**

- Introduction to IoT Concepts
  Overview of IoT
  Evolution and history of IoT
  Key components and technologies in IoT
- IoT architectures: Edge computing vs. cloud computing
  Communication protocols in IoT
  Applications of IoT in different industries
- IoT Security and Privacy
  Security challenges in IoT
  Encryption and secure communication in IoT
- Ethical implications and privacy concerns in IoT
- Strategies for securing IoT systems
- IoT device architecture: Microcontrollers and microprocessors
- Mid-term Exam
- Sensor integration and interfacing
- Sensor networks: Design principles and scalability
- AI-driven IoT solutions
- Blockchain in IoT
- Emerging trends in IoT
- IoT in smart cities and homes
- IoT and 5G networks
- Final project presentations by student groups

- **Course Description** :

- The IoT course provides students with a thorough understanding of IoT principles, technologies, and applications. Through a blend of theoretical learning and hands-on exercises, students explore the complex network of interconnected devices, sensors, actuators, and data analytics central to IoT environments.

- Starting with fundamental concepts such as sensors, actuators, connectivity protocols, and data analytics techniques, students grasp how these elements collaborate to collect, transmit, and analyze data from the physical world.

- Advancing further, students study IoT system architecture, examining diverse deployment models and infrastructure components like edge computing and cloud platforms. Real-world examples and case studies deepen their understanding, enabling them to analyze and compare various architectures while comprehending the roles of edge computing, cloud platforms, and communication protocols in shaping IoT solutions.

- A key focus area is security in IoT systems. Students learn to identify and evaluate security challenges inherent in IoT deployments, covering data integrity, confidentiality, and device authentication. They also explore strategies and best practices to mitigate security risks and safeguard IoT ecosystems from cyber threats.

- In addition to theoretical learning, the course emphasizes practical experience through hands-on exercises and project-based learning. Students design, implement, and troubleshoot end-to-end IoT solutions, integrating hardware selection, communication protocols, and data processing techniques. Proficiency in implementing common IoT protocols such as MQTT and CoAP ensures effective communication and interoperability among devices.

- Upon completion, students possess a deep understanding of IoT principles and technologies, equipped with the knowledge and skills to design, deploy, and secure IoT systems across diverse domains. They are well-prepared to navigate the dynamic challenges of the rapidly evolving IoT landscape, contributing to the advancement of connected technologies in industry and society.

- Course Outcomes

- Introduction to IoT Concepts
- Define and explain fundamental concepts related to the Internet of Things (IoT), including sensors, actuators, connectivity, and data analytics.
- Describe the historical evolution and current trends of IoT technologies and applications.
- Analyze and compare different IoT architectures, identifying the roles of edge computing, cloud platforms, and various communication protocols.
- Security and Privacy
- Assess and articulate security challenges in IoT systems, proposing effective strategies to mitigate risks related to data integrity, confidentiality, and device authentication.
- Evaluate the ethical implications of IoT applications, considering privacy concerns, data ownership, and potential societal impacts.

- Recommended Textbook(s):

Buyya, R., & Dastjerdi, A. V. (Eds.). (2016). Internet of Things: Principles and paradigms. Elsevier.

Kumar, S. (2021). Fundamentals of Internet of Things. CRC Press.

- Prerequisites:

- **Lab. Topics:**

- Packet Tracer – Connecting Devices to Build IoT Topology
- Packet Tracer – Simulating IoT Devices
- Packet Tracer – Simulating IoT Devices I
- Packet Tracer - Sensors and the PT Microcontroller-I
- Packet Tracer - Sensors and the PT Microcontroller-II
- Packet Tracer – SBC Actuate With Python I
- Packet Tracer – SBC Actuate With Python II
- Packet Tracer – Explore the Smart Home I
- Packet Tracer – Explore the Smart Home II
- Packet Tracer – Build a Connected Factory Solution I
- Packet Tracer – Build a Connected Factory Solution II
- Packet Tracer – Securing Cloud Services in the IoT I
- Packet Tracer – Securing Cloud Services in the IoT II
- Packet Tracer – Explore the Smart City I
- Packet Tracer – Explore the Smart City II

CSDE311 Computer Networks II (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- Introduction to Computer Networks:
  Basic concepts of computer networks
  Network architecture and protocols
  Networking standards and organizations
- Network Models and Protocols:
  OSI model and TCP/IP protocol suite
  Data encapsulation and protocol stacks
  Network addressing and subnetting
- Physical Layer and Data Link Layer:
  Transmission media and signaling techniques
  Ethernet LANs and switching

MAC addressing and error detection and correction

- Network Layer
  IP addressing and subnetting
  Routing algorithms and protocols (e.g., RIP, OSPF)
  Introduction to IPv6
- Transport Layer
  Transport protocols (e.g., TCP, UDP)
  Connection-oriented and connectionless communication
  Flow control and congestion control
- Application Layer
  Application layer protocols (e.g., HTTP, FTP, DNS)
  Client-server model and peer-to-peer applications
  Web services and APIs
- Mid-term Exam
- Network Management and Performance
  Network monitoring and troubleshooting
- Quality of Service (QoS) and traffic management
  Network management protocols (e.g., SNMP)
- Virtual Private Networks (VPNs) and Remote Access
  VPN concepts and protocols
- VPN deployment and configuration
- Network Design and Planning
  LAN and WAN design considerations
- Network scalability and redundancy
  Network documentation and project management
- Project
- final Exam

- **Course Description** :Understand the fundamental concepts and principles of computer networks, including network architectures, protocols, layers, and networking technologies. Explain the functions and interactions of various network layers, including the physical layer, data link layer, network layer, transport layer, and application layer. Demonstrate knowledge of network addressing and routing, including IP addressing, subnetting, and routing algorithms. Configure and troubleshoot network devices, such as routers, switches, and firewalls. Analyze and evaluate network performance and identify and resolve network-related issues and bottlenecks.

- **Course Outcomes :**

1.Understanding Network Fundamentals: Introduce students to the basic concepts and components of computer networks, including network architectures, protocols, and network layers.
2.Exploring Network Protocols: Familiarize students with various network protocols, such as TCP/IP, UDP, HTTP, FTP, DNS, and their roles in facilitating communication and data transfer in computer networks.
3.Studying Network Topologies and Technologies: Explore different network topologies, such as bus, star, ring, mesh, and hybrid, and technologies such as Ethernet, Wi-Fi, and cellular networks.
4.Learning Network Design and Implementation: Develop skills in designing and implementing computer networks, including network planning.

- **Recommended Textbook(s):**
  Distributed Systems And TCP/IP Programming In .NET 4.0
  Distributed Systems And TCP/IP Programming In .NET 4.0
- **Prerequisites: CSDC305**
- **Lab. Topics:**

- Introduction to Computer Networks:
  Basic concepts of computer networks
  Network architecture and protocols
  Networking standards and organizations
- Network Models and Protocols:
  OSI model and TCP/IP protocol suite
  Data encapsulation and protocol stacks
  Network addressing and subnetting
- Physical Layer and Data Link Layer:
  Transmission media and signaling techniques
  Ethernet LANs and switching
  MAC addressing and error detection and correction
- Network Layer
  IP addressing and subnetting
  Routing algorithms and protocols (e.g., RIP, OSPF)
  Introduction to IPv6
- Transport Layer
  Transport protocols (e.g., TCP, UDP)
  Connection-oriented and connectionless communication
  Flow control and congestion control
- Application Layer
  Application layer protocols (e.g., HTTP, FTP, DNS)

Client-server model and peer-to-peer applications
Web services and APIs

- Mid-term Exam
- Network Management and Performance
  Network monitoring and troubleshooting
- Quality of Service (QoS) and traffic management
  Network management protocols (e.g., SNMP)
- Virtual Private Networks (VPNs) and Remote Access
  VPN concepts and protocols
- VPN deployment and configuration
- Network Design and Planning
  LAN and WAN design considerations
- Network scalability and redundancy
  Network documentation and project management
- Project

CSDE307 Compilers II (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- Bottom-up parsing (shift-reduce parser)
- Bottom-up parsing (LR parser)
- Bottom-up parsing (LR parser
- Semantic Analysis (Type systems and type checking)
- Semantic Analysis (Static analysis and error detection)
- Mid-term Exam
- Intermediate Code Generation

  Intermediate representations

  Syntax-directed translation and code generation

- Intermediate Code Generation

Control Flow Analysis

Basic blocks

Data-flow analysis

- Code Optimization (Part 1)

  Principles of Optimization

  Common optimization techniques

  Local code optimization

  Global Optimization Methods

- Code Optimization (Part 2)

  Loop optimization

  Register allocation and instruction scheduling

- Code Generation (Part 1)

  Target machine models and instruction sets

  Instruction selection and mapping

- Code Generation (Part 2)

  Memory Management and Runtime Support

  Addressing modes

  Memory management

  Runtime support for generated code

- Compiler Testing and Debugging

  Testing strategies for compilers

  Compiler validation techniques

- Debugging and error handling in compilers

- Advanced Topics:

  Just-in-time (JIT) compilation

- **Course Description** :

- Understand the fundamental concepts of compiler design: Students should be able to comprehend the basic principles, techniques, and components involved in designing and implementing compilers.
- Analyze and describe the various phases of a compiler: Students should be able to explain the different phases of a compiler, including lexical analysis, syntax analysis, semantic analysis, intermediate code generation, optimization, and code generation.
- Implement a compiler: Students should gain practical experience by implementing a simple compiler for a programming language. This may involve designing and developing the lexical analyzer, parser, semantic analyzer, and code generator.
- Apply formal language theory: Students should understand formal languages, regular expressions, context-free grammars, and automata theory, and be able to apply this knowledge to analyze and manipulate programming languages.

- **Course Outcomes :**
  - Understand the fundamental concepts of compiler design: Students should be able to comprehend the basic principles, techniques, and components involved in designing and implementing compilers.
  - Analyze and describe the various phases of a compiler: Students should be able to explain the different phases of a compiler, including lexical analysis, syntax analysis, semantic analysis, intermediate code generation, optimization, and code generation.
  - Implement a compiler: Students should gain practical experience by implementing a simple compiler for a programming language. This may involve designing and developing the lexical analyzer, parser, semantic analyzer, and code generator.
  - Apply formal language theory: Students should understand formal languages, regular expressions, context-free grammars, and automata theory, and be able to apply this knowledge to analyze and manipulate programming languages.
  - Perform lexical and syntactic analysis: Students should be able to develop lexical analyzers and parsers to break down the source code into meaningful tokens and construct the corresponding parse tree or abstract syntax tree.
  - Conduct semantic analysis: Students should learn how to perform semantic analysis,

including type checking, symbol table management, and static analysis techniques to ensure program correctness and identify potential errors.

- Understand intermediate representations: Students should become familiar with various intermediate representations used in compilers, such as three-address code, abstract syntax trees, and control flow graphs. They should understand how to manipulate and optimize these representations.

- Apply optimization techniques: Students should learn about common compiler optimization techniques, such as constant folding, common subexpression elimination, loop optimization, and register allocation. They should be able to apply these techniques to improve the efficiency of generated code.

- Generate Low Level Language: Students should understand the process of generating machine code or assembly language from the intermediate representation. They should be able to apply code generation algorithms and handle low-level details such as instruction selection and addressing modes.

- Test and debug compilers: Students should develop skills in testing and debugging compilers. They should be able to identify and fix errors in the compiler implementation and evaluate the correctness and performance of generated code.

- Stay updated with current compiler trends: Students should be aware of recent developments and trends in the field of compiler design, including just-in-time (JIT) compilation, language-specific optimizations, and parallelizing compilers.

- **Recommended Textbook(s):**
  - A.Aho,R.Sethi,J.D.Ullman," Compilers- Principles, Techniques and Tools"Addison-Weseley,2007
- **Prerequisites: CSDC310**
- **Lab. Topics:**

- Syntax Analysis
- shift-reduce parser
- Syntax Analysis
- (SLR table)
- Syntax Analysis
- (SLR table)
- Syntax Analysis
- (LR program)

- Type checking
- Type checking
- Intermediate Code Generation
- Intermediate Code Implementation
  (3 address code)
- Basic blocks
- Local code optimization
- Control Flow Analysis
- Data-flow analysis
- Global Optimization Methods
- Loop optimization
- Code Generation

CSDC309 Software Engineering (2-2-0-0)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

Introduction to Software Engineering
• Definition and importance of Software Engineering
• Software development life cycle models
• Roles and responsibilities of software engineers
Requirements Engineering
• Software requirements elicitation techniques
• Requirements analysis and documentation
• Requirements validation and verification
Software Design Principles
• Object-oriented design principles
• Design patterns and architectural styles
• Modularity and software component design
Software Testing and Quality Assurance
• Testing techniques and levels (unit testing, integration testing, system testing)
• Test planning and test case design
• Software quality attributes and metrics
Software Project Management
• Project planning and estimation
• Risk management and mitigation

- Project monitoring and control

Software Configuration Management

- Version control systems and practices
- Build management and release processes
- Change management and configuration control

Mid-term Exam + Change management and configuration control

Software Maintenance and Evolution

- Types of software maintenance
- Bug tracking and debugging techniques
- Software reengineering and system evolution

Software Development Tools and Environments

- Integrated Development Environments (IDEs) and software development tools
- Collaboration and communication tools for software teams
- Software documentation and knowledge management tools

Software Ethics and Professional Practices

- Ethical considerations in software engineering
- Professional responsibility and accountability
- Intellectual property and legal issues in software development

Emerging Trends and Technologies in Software Engineering

- Software development for mobile platforms

Emerging Trends and Technologies in Software Engineering

- Cloud computing and Software-as-a-Service (SaaS)

Emerging Trends and Technologies in Software Engineering

- DevOps and continuous integration/continuous delivery (CI/CD)

Group Project Work

- Work on group projects applying software engineering principles
- Project management, communication, and collaboration

Project Presentations and Review

- Group project presentations and demonstrations
- Review and discussion of lessons learned

Preparatory week before the final Exam

- **Course Description** :Introduction to Software Engineering, Requirements Engineering, Software Design Principles, Software Testing and Quality Assurance, Software Project Management, Project planning, estimation, and scheduling, Risk management and mitigation strategies, and Project monitoring and control.

- **Course Outcomes :**

  1.Understand the fundamental principles, concepts, and practices of Software Engineering, including the importance of following a systematic and disciplined approach to software development.

  2. Apply software development methodologies and processes, such as the Software

Development Life Cycle (SDLC), to analyze, design, implement, test, and maintain software systems.
3. Elicit, analyze, document, and manage software requirements effectively, considering stakeholders' needs and system constraints.
4. Design software systems and architectures that are modular, scalable, and maintainable, applying software design principles, architectural styles, and design patterns.
5. Implement and execute software testing techniques to verify and validate software functionality, ensuring the delivery of high-quality software systems.
6. Apply project management principles and practices to plan, estimate, schedule, and monitor software development projects, considering resource allocation and risk management.
7. Understand and apply software configuration management practices, including version control, build management, and change management.
8. Demonstrate the ability to work effectively in a team, collaborating with others in software development projects and communicating ideas and solutions effectively.
9. Understand the challenges and techniques of software maintenance and evolution, including bug fixing, software updates, and system enhancements.
10. Develop a professional and ethical attitude towards software engineering, recognizing the importance of professional responsibility, accountability, and lifelong learning in the field.

- **Recommended Textbook(s):**
- https://www.tutorialspoint.com/software_engineering/software_engineering_tutorial.pdf

Prerequisites:

- **Lab. Topics:**

CSDE408 Operating Systems I (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**
  - Introduction to Operating System
  - Computer Hardware Structure Overview.
  - Operating System Concepts and Structure.
  - Operating System Operation and Functions
  - Process Management: CPU Scheduling
  - First-Come, First-Served Scheduling
  - CPU Scheduling Algorithms
  - Shortest Job First Scheduling

- Exam 1
- CPU Scheduling Algorithms
- Priority Scheduling
- CPU Scheduling Algorithms
- Round-Robin Scheduling
- Interrupt in Operating system
- Protection and Security
- Techniques to improve performance, manage resource in operating system
- Introduction to Memory management
- Exam 2
- Reviewing All Topics

- **Course Description** :

- Enabling students to obtain an understanding and knowledge of the components of an operating system.
- Running and executing programs within the computer.
- Providing the students with the fundamentals and topics related to thinking.
- Problem Solving: Use a range of approaches to critically analyze and evaluate practices of operating systems in identifying, defining, and solving problems by using alternative effective and efficient algorithms.
- Modeling and Design: Use a range of specialist models to model the problems of computer and communication systems, such as deadlock, and design efficient and effective handling procedures.
- Analytic: Critically analyze and evaluate the performance and effectiveness of different algorithms used by different operating systems.
- Creative: Extend knowledge in operating systems to construct specific and effective solution to manage and control computer resources.
- Communication: Show ability to communicate information in appropriate oral and written forms.
- Organizational and Developmental Skills: Demonstrate ability to organize ideas and effectively allocate time in given assignment.

- **Course Outcomes :**

- Enabling students to obtain an understanding and knowledge of the components of an operating system.
- Running and executing programs within the computer.

- Providing the students with the fundamentals and topics related to thinking.
- Problem Solving: Use a range of approaches to critically analyze and evaluate practices of operating systems in identifying, defining, and solving problems by using alternative effective and efficient algorithms.
- Modeling and Design: Use a range of specialist models to model the problems of computer and communication systems, such as deadlock, and design efficient and effective handling procedures.
- Analytic: Critically analyze and evaluate the performance and effectiveness of different algorithms used by different operating systems.
- Creative: Extend knowledge in operating systems to construct specific and effective solution to manage and control computer resources.
- Communication: Show ability to communicate information in appropriate oral and written forms.
- Organizational and Developmental Skills: Demonstrate ability to organize ideas and effectively allocate time in given assignment.

- **Recommended Textbook(s):**

1. Operating Systems: Evolutionary Concepts and Modern Design Principles.
2.Operating system concepts (Ninth Edition).
3. Modern Operating Systems (Fifth Edition).

- **Prerequisites:**
- **Lab. Topics:**

- Introduction to OS
- Operating System types
- Introduction to MS-DOS Instructions
- Implement MS DOS commands
- WAP to implement First Come First Serve (FCFS) Scheduling
- WAP to implement First Come First Serve (FCFS) Scheduling
- WAP to implement shortest job first (SJF) scheduling
- WAP to implement shortest job first (SJF) scheduling
- WAP to implement Priority based scheduling
- WAP to implement Priority based scheduling
- WAP to implement Round Robin (RR) scheduling
- WAP to implement Round Robin (RR) scheduling.
- WAP to implement CPU Scheduling Algorithms

- With Interrupts
- WAP to implement CPU Scheduling Algorithms
- With Interrupts
- Exam

CSDE411 Computer Security I (2-2-0-0)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- Introduction
- Historical Notes
- Classical Encryption Techniques
- Encryption Machines
- Substitution Ciphers:
- Caesar
- Monoalphabetic Cipher
- Polyalphabetic Cipher: Vigener
- Statistical Analysis
- Palyfair
- Hill Cipher
- Transposition Ciphers
- Midterm Exam
- Block Ciphers
- The Data Encryption Standard
- DES Strength
- DES Cryptanalysis
- Using Block and Stream Ciphers
- Modes of Operation
- AES: The Advanced Encryption Standard
- AES Strength

- **Course Description** :This is an introductory undergraduate course on information systems security. This course covers materials related to cryptography and information security. Cryptography, broadly speaking, is about communicating in the presence of an adversary, with goals like preservation of privacy and integrity of communicated data. In the first semester, we will focus on classical and symmetric key cryptography, including block ciphers and their modes of operation. The course will emphasize rigorous mathematical formulations of security goals and aim to train students in spotting weaknesses in designs. This is generally regarded by undergraduates as a challenging course. It is mainly theoretical and mathematical in nature, and calls for ability to understand abstract concepts. Students would be asked to do assignments, solve home works, and implement programming projects in order to develop their skills.

- **Course Outcomes :**

After completing the module, the student should be able to:

1. Describe the basic mathematical and technical issues relating to information security.
2. Learning how to leverage these concepts to protect computers from external threats.
3. Interpret how technology affects the design of symmetrical systems, especially block ciphers.
4. Use rigorous mathematical formulations of symmetric cryptography to spot weaknesses in designs.
5. Demonstrate skills in using classical ciphers for encryption and decryption.
6. Demonstrate skills in using some basic cryptanalysis techniques related to classical cryptography.

- **Recommended Textbook(s):**
- William Stallings, Cryptography and Network Security: Principles and Practice, 7/E, Pearson Education, Inc., 2017. ISBN 978-0-13-444428-4
- **Prerequisites:**
- **Lab. Topics:**

CCIT066 Artificial Intelligence I (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- General Introduction
- The History of AI
- Systematic Search: Basic Graph Concepts; State Space Representation of Problems.
- Depth-First Search and Breadth-First search
- Hybrid Search.
- Propositional Logic and Resolution in Propositional Logic.
- Predicate Logic: Basic Concepts and Definitions.
- Exam
- Horn Clauses; Unification; Skolemization and Clause Normal Form.
- Horn Clauses; Unification; Skolemization and Clause Normal Form.

- Modus-Ponens and Resolution Inference Rules in Predicate Logic.
- Control Strategies for Resolution Inference (Problem Solving).

- **Course Description** :This course aims to make students know about AI and how to solve problems .

- **Course Outcomes :**

- Introducing students to a new scientific subject that enhances their knowledge in the field of computer science, Artificial Intelligence.
- It aims to introduce students to the meaning of the scientific term "Artificial Intelligence" and its applications in computer science, engineering, and other related fields.
- Developing students' computational and mathematical skills.
- Introducing students to the use of logical thinking in problem representation and solving
- The course aims to provide an understanding of the fundamentals and principles of Artificial Intelligence methods, including algorithms and computer programs that simulate human, animal, or other behavioral patterns. These methods enable computers (machines) to acquire the ability to learn, infer, and react to specific situations. One of these intelligent applications is the humanoid robot.
- AI is a rapidly evolving field, and studying AI allows students to understand the underlying principles, algorithms, and techniques that drive AI systems. This knowledge can lead to advancements in AI technology, such as developing more efficient algorithms, improving machine learning models, or creating new AI applications.
- AI has the potential to tackle complex problems that are difficult for traditional computing methods.

- **Recommended Textbook(s):**

- Artificial Intelligence: A Modern Approach, Stuart Russell and Peter Norvig, Pearson Education, 2020.
- Artificial Intelligence: Structures and Strategies for Complex Problem Solving, George F. Luger, Addison-Wesley, 2008

- **Prerequisites:**
- **Lab. Topics:**
    - Introduction to basics of the python language and take the fundamental tools in python (if, else if, nested if, for loop, while , list and functions) and implementing various program in python.

- Introduction to basics of the python language and take the fundamental tools in python (if, else if, nested if, for loop, while , list and functions) and implementing various program in python.
- Introduction to basics of the python language and take the fundamental tools in python (if, else if, nested if, for loop, while , list and functions) and implementing various program in python.
- An introduction to the topic of how to represent a tree in the Python language (Represent a tree in python using list) and explanation the first algorithm (Depth first search) then write and execute the program on the computer by the students.
- An introduction to the topic of how to represent a tree in the Python language (Represent a tree in python using list) and explanation the first algorithm (Depth first search) then write and execute the program on the computer by the students.
- An introduction to the topic of how to represent a tree in the Python language (Represent a tree in python using list) and explanation the first algorithm (Depth first search) then write and execute the program on the computer by the students.
- Explanation (Breadth first search and hybrid algorithms)and explain how to convert the algorithms to a program in python by using (list, functions, and loops) then write and execute the program on the computer by the students.
- Explanation (Breadth first search and hybrid algorithms)and explain how to convert the algorithms to a program in python by using (list, functions, and loops) then write and execute the program on the computer by the students.
- Explanation (Breadth first search and hybrid algorithms)and explain how to convert the algorithms to a program in python by using (list, functions, and loops) then write and execute the program on the computer by the students.

CSDC403 PHP Web Development (3-2-0-2)

- Designation as a „required‟ or „elective‟ course:
This is a required course for the computer science department .

- **Course Topics :**

- Introduction on Web Hosts.
- Introduction to PHP.

- Programming with PHP.
- Functions in PHP.
- Data Validation (Server Side).
- First exam.
- Introduction to MySQL.
- Connecting to the Database.
- MySQL Queries.
- MySQL Queries.
- Operation on MySQL.
- Advanced PHP.
- Second Exam.
- Students team research projects (reports and presentations).
- Students team research projects (reports and presentations)
- Preparatory week before the final Exam

- **Course Description** :

1. Enhanced User Experience: Web applications are designed to provide an intuitive and user-friendly experience for visitors or users.

2. Increased Accessibility: Web applications can be accessed from anywhere with an internet connection, making them highly accessible to users across different devices and platforms.

3. Improved Efficiency and Productivity: Web applications can automate and streamline various business processes, leading to improved efficiency and productivity.

4. Scalability and Flexibility: Web applications can be designed and developed to accommodate growth and changing business needs.

5. Cost-effectiveness: Compared to traditional software applications, web applications can be more cost-effective in terms of development, deployment, and maintenance.

- **Course Outcomes :**

- Enhanced User Experience: Web applications are designed to provide an intuitive and user-friendly experience for visitors or users.
- Increased Accessibility: Web applications can be accessed from anywhere with an internet connection, making them highly accessible to users across different devices and platforms.
- Improved Efficiency and Productivity: Web applications can automate and streamline various business processes, leading to improved efficiency and productivity.

- Scalability and Flexibility: Web applications can be designed and developed to accommodate growth and changing business needs.

- Cost-effectiveness: Compared to traditional software applications, web applications can be more cost-effective in terms of development, deployment, and maintenance methods.

- **Recommended Textbook(s):**
- PHP and MySQL for Dynamic Web Sites 4th Edition.
- **Prerequisites:**
- **Lab. Topics:**

- Creating the HTML files and writing basic scripts.
- Executing HTML tags.
- Creating HTML forms to collect data.
- Inserting CSS to HTML and Dealing with inline style.
- Quiz and in lab script writing.
- Applying methods of Inserting CSS to the HTML and Designing and applying styling tables.
- Inserting JavaScript code and Dealing html by JavaScript

CSDC406 Digital Image Processing (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- Pictures & Images
- What is the digital images?
- The elements of digital image processing system and human visual system
- Electromagnetic spectrum and visible radiation
- Image representation and digital image files formats
- Sampling & Quantization
- Exam
- Gray scale image modification
- Algebraic operations on images
- Image analysis and histogram representation
- Image preprocessing and image enhancement
- Convolution and correlation processes
- The types of 2D filtering compared with 1D filtering
- Review
- Exam

- **Course Description** :Pictures & Images, What is the digital images?., elements of digital image processing system and human visual system, electromagnetic spectrum and visible radiation., image representation and digital image files formats., Sampling & Quantization , gray scale image modification., algebraic operations on images., image analysis and histogram representation., image preprocessing and image enhancement., convolution and correlation processes., types of 2D filtering compared with 1D filtering.

- **Course Outcomes :**

- Understanding the concept of image processing and its various applications.
- Understanding how images are represented and displayed on the screen..
- Understanding and acquiring knowledge of using the MATLAB package in image processing applications.
- Understanding and acquiring knowledge of different methods of image processing.
- Understanding and gaining knowledge of various algorithms used in image processing.
- Providing the student with the skill of representing two-dimensional arrays.

- **Recommended Textbook(s):**
- Gonzalez, Digital Image Processing Using Mtlab, 3 rd Edition, Pearson,2014
- **Prerequisites:**
- **Lab. Topics:**

- Introduction to MATLAB Programming.
- Information Mat lab Desktop.
- Fundamentals of Image Processing.
- Methods of Displaying Image.
- Image Analysis.
- Image Enhancement.
- Color Images

CSDE407 Research methodology (2-2-0-0)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics   :**

Introduction to Research Methodology
- Overview of the research process
- Characteristics of good research
- Ethical considerations in research

Research Design
- Experimental, quasi-experimental, and non-experimental designs
- Cross-sectional and longitudinal designs

Sampling Techniques
- Probability and non-probability sampling methods
- Sample size determination

Data Collection Methods

- Questionnaire design and development
- Interview techniques

Data Analysis - Descriptive Statistics
- Measures of central tendency
- Measures of dispersion

Data Analysis - Inferential Statistics
- Hypothesis testing
- t-tests
- Mid-term Exam + Hypothesis testing

Data Analysis - Inferential Statistics (continued(
- ANOVA
- Regression analysis

Qualitative Research Methods
- Introduction to qualitative research
- Approaches and techniques in qualitative research

Data Analysis - Qualitative Analysis
- Thematic analysis
- Content analysis

Research Proposal Development
- Components of a research proposal
- Research question formulation and objectives

Literature Review
- Strategies for conducting a literature review
- Evaluating and synthesizing research articles

Data Interpretation and Presentation
- Effective presentation skills
- Interpreting and communicating research findings

Research Ethics and Integrity
- Ethical considerations in research involving human participants
- Responsible data management and sharing

Review and Recap
- Recap of key concepts and methodologies covered throughout the course
- Q&A session and preparation for final assessments

- **Course Description** :

- The course aims to familiarize students with the research process, including the different stages involved, from formulating a research question to presenting results.
- Developing research skills: The course aims to develop students' skills in conducting research, including identifying research problems, designing appropriate research methods, collecting and analyzing data, and drawing valid conclusions.
- Familiarity with Research Design: The course focuses on introducing different research designs, such as experimental, correlational, qualitative, and quantitative, and helps students understand their strengths, limitations, and appropriate applications.

- **Course Outcomes :**

Knowledge of research methods: Gain a comprehensive understanding of different research methods, including qualitative and quantitative methods and their applications in various disciplines. Research design skills: Develop the ability to design research studies by formulating appropriate research questions, choosing appropriate methodologies, and designing data collection procedures. Ethical considerations: Understand the ethical principles and guidelines that govern research involving human participants, and ensure that their rights and confidentiality are protected Literature Review Mastery: Gain skills in conducting a comprehensive literature review, identifying relevant sources, evaluating research articles, and synthesizing existing knowledge.

- **Prerequisites:**
- **Lab. Topics:**

CSDE408 Operating Systems II (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- Process Management: Deadlocks
- Process Management: Deadlocks
- Resource-Allocation Graph
- Deadlock Handling
- Deadlock Avoidance (Banker's Algorithm)
- Resource-Request of Banker's Algorithm
- Memory Management.
- Dynamic storage allocation problem
- Exam 1
- Contiguous Memory Allocation Algorithms (First Fit)
- Contiguous Memory Allocation Algorithms (Best Fit)
- Contiguous Memory Allocation Algorithms (Worst Fit)
- Exam 2
- Replacement Algorithms
- Students team reports and presentations

- **Course Description**

- Enabling students to obtain an understanding and knowledge of the components of an operating system.
- Running and executing programs within the computer.
- Providing the students with the fundamentals and topics related to thinking.
- Problem Solving: Use a range of approaches to critically analyze and evaluate practices of operating systems in identifying, defining, and solving problems by using alternative effective and efficient algorithms.

- Modeling and Design: Use a range of specialist models to model the problems of computer and communication systems, such as deadlock, and design efficient and effective handling procedures.

- Analytic: Critically analyze and evaluate the performance and effectiveness of different algorithms used by different operating systems.
- Creative: Extend knowledge in operating systems to construct specific and effective solution to manage and control computer resources.
- Communication: Show ability to communicate information in appropriate oral and written forms.
- Organizational and Developmental Skills: Demonstrate ability to organize ideas and effectively allocate time in given assignment.

- Course Outcomes :

- Enabling students to obtain an understanding and knowledge of the components of an operating system.
- Running and executing programs within the computer.
- Providing the students with the fundamentals and topics related to thinking.
- Problem Solving: Use a range of approaches to critically analyze and evaluate practices of operating systems in identifying, defining, and solving problems by using alternative effective and efficient algorithms.
- Modeling and Design: Use a range of specialist models to model the problems of computer and communication systems, such as deadlock, and design efficient and effective handling procedures.
- Analytic: Critically analyze and evaluate the performance and effectiveness of different algorithms used by different operating systems.
- Creative: Extend knowledge in operating systems to construct specific and effective solution to manage and control computer resources.
- Communication: Show ability to communicate information in appropriate oral and written forms.

- Recommended Textbook(s):

1. Operating Systems: Evolutionary Concepts and Modern Design Principles.
2.Operating system concepts (Ninth Edition)
3. Modern Operating Systems (Fifth Edition).

- Prerequisites: CSIT401

- Lab. Topics:

- Explain Research Project to build OS
- Explain Research Project to build OS
- WAP to implement Resource-Allocation Graph
- Understand Data Structures used to implement the Banker’s Algorithm
- WAP to implement Deadlock Detection. (Banker's Algorithm)

- WAP to implement Deadlock Detection. (Banker's Algorithm)
- WAP to implement Deadlock Detection. (Banker's Algorithm with new resource).
- WAP to implement Deadlock Detection. (Banker's Algorithm with new resource).
- Exam 1
- WAP to implement Algorithms for Allocation First-Fit
- WAP to implement Algorithms for Allocation of Best-Fit
- WAP to implement Algorithms for Allocation of Worst Fit
- Exam 2
- WAP to implement Page Replacement Algorithms
- Student Team Research Projects

CSDE411 Computer Security II (2-2-0-0)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

  - Groups, Rings, and Fields
  - Modular Arithmetic
  - Polynomial Arithmetic
  - Finite Fields of the Form GF(2n)
  - Issues for Symmetric Key Cryptography:
    Key Distribution, Random Number Generation
  - Prime Numbers and Primality Tests
  - Public-Key Cryptography I:
    General Concepts, RSA System, RSA Security
  - Public-Key Cryptography II:
    Exchanging Secret Session Keys
    Diffie-Hellman System
  - Public-Key Cryptography III:
    Constructing Digital Signatures
    El-Gamal System
  - Midterm Exam
  - Hashing for Message Authentication
    Cryptographic Hash Functions
    MACs Schemes
  - Malware: Viruses and Worms
  - Access Control:
    Authentication and Authorization
    CAPTCHA
  - Trusted Operating Systems

- Mounting Targeted Attacks with
- Trojans and Social Engineering

- **Course Description**

In the second semester of the information systems security, our focus will mainly be directed to public key cryptography and system security issues. We will cover topics like hash functions, digital signatures, asymmetric encryption, RSA, public-key infrastructure, key distribution, and various applications. The course aims to train students in spotting weaknesses in designs. Indeed, we will cover topics like access control, viruses, worms, and operating systems security. This is generally regarded by undergraduates as a challenging course. It is mainly theoretical and mathematical in nature, and calls for ability to understand abstract concepts. Students would be asked to do assignments, solve home works, and implement programming projects in order to develop their skills.

- **Course Outcomes :**

After completing the module, the student should be able to:

1. Understand and discuss the mathematical background behind the evolution of public key cryptography.
2. Interpret how technology and theoretical advances can threat existing public key systems.
3. Demonstrate skills in using some public key algorithms for various applications.
4. Demonstrate skills in applying cryptographic hash functions for message authentication.
5. Describe the social and ethical issues relating to viruses and other malicious codes.

- **Recommended Textbook(s):**
- William Stallings, Cryptography and Network Security: Principles and Practice, 7/E, Pearson Education, Inc., 2017. ISBN 978-0-13-444428-4
- **Prerequisites: CSDC404**
- **Lab. Topics:**

CCIT067 Artificial Intelligence II (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- Heuristic Search: Heuristic Functions.
- Hill Climbing Algorithm.
- Best-First Search Algorithm.
- Cost Functions.
- A* Algorithm.
- Properties of Heuristic Functions.
- Search in Games: Introduction.
    - Min-Max Algorithm.
    - Mid Term Exam
- Alpha-Beta Search Procedure; Enhancement to Game Search.

- Expert Systems: Structure; Rule Based Expert Systems
- Control Strategies in Rule Based
- Production Systems: Backward
    - Chaining and its Implementation.
- Pure Forward Chaining and its Implementation; Rule-Cycle Hybrid Control Strategy and its Implementation.
- Uncertainty in Expert Systems: Representing Probabilities in Rules; Combining Evidence.
- Other Approaches to Expert System
    - Design: Decision Lattices; And-OrNot Lattices.

- **Course Description**

This course aims to make students know about AI and how to solve problems by using heuristics search techniques and expert systems.

- **Course Outcomes :**

- Introducing students to a new scientific subject that enhances their knowledge in the field of computer science, Artificial Intelligence.
- It aims to introduce students to the meaning of the scientific term "Artificial Intelligence" and its applications in computer science, engineering, and other related fields.
- Developing students' computational and mathematical skills.
- Introducing students to the use of logical thinking in problem representation and solving
- The course aims to provide an understanding of the fundamentals and principles of Artificial Intelligence methods, including algorithms and computer programs that simulate human, animal, or other behavioral patterns. These methods enable computers (machines) to acquire the ability to learn, infer, and react to specific situations. One of these intelligent applications is the humanoid robot.
- AI is a rapidly evolving field, and studying AI allows students to understand the underlying principles, algorithms, and techniques that drive AI systems. This knowledge can lead to advancements in AI technology, such as developing more efficient algorithms, improving machine learning models, or creating new AI applications.
- AI has the potential to tackle complex problems that are difficult for traditional computing methods.

- **Recommended Textbook(s):**
- Artificial Intelligence: A Modern Approach, Stuart Russell and Peter Norvig, Pearson Education 2020.

Prerequisites: CCIT066

- **Lab. Topics:**

- An introduction to the topic of how to represent a tree in the Python language (Represent a tree in python using list) and explanation the first algorithm (Depth first search) then write and execute the program on the computer by the students.
- Explanation (Breadth first search and hybrid algorithms)and explain how to convert the algorithms to a program in python by using (list, functions, and loops) then write and execute the program on the computer by the students.
- Explanation (Hill climbing and Best - search algorithm) and explain how to convert the algorithm to a program in python by using (list, functions and loops)then write and execute the program on the computer by the students.
- Explanation(A* search algorithm) and explain how to convert the algorithm to a program in python by using (list, functions and loops), write and execute the program on the computer by the students.
- Explanation (Min Max search algorithm) and explain how to convert the algorithm to a program in python by using (list, functions and loops), write and execute the program on the computer by the students.
- Making a comprehensive review of all algorithms and then making a final exam for all the previous topics.

CSDE409 Web Programming –Asp (3-2-0-2)

- Designation as a „required‟ or „elective‟ course:
This is a required course for the computer science department .

- **Course Topics :**

  - Putting ASP.NET Core in Context
  - First ASP.NET Core
  - Application
  - Using the Development Tools
  - Testing ASP.NET Core Applications
  - A Real Application
  - Administration
  - Security and Deployment
  - Exam
  - Understanding the ASP.NET Core Platform
  - Using URL Routing
  - Database design

- Dealing with the database in ASP.NET Core
- Project
- Project
- Exam

- **Course Outcomes :**

1. Enhanced User Experience: Web applications are designed to provide an intuitive and user-friendly experience for visitors or users.
2. Increased Accessibility: Web applications can be accessed from anywhere with an internet connection, making them highly accessible to users across different devices and platforms.
3. Improved Efficiency and Productivity: Web applications can automate and streamline various business processes, leading to improved efficiency and productivity.
4. Scalability and Flexibility: Web applications can be designed and developed to accommodate growth and changing business needs.
5. Cost-effectiveness: Compared to traditional software applications, web applications can be more cost-effective in terms of development, deployment, and maintenance.

- **Recommended Textbook(s):**

Pro ASP.NET Core 6 Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages

Prerequisites:

- **Lab. Topics:**

- Installing Visual Studio
- Creating an ASP.NET Core Project
- Creating the Project
- Managing Packages
- Enabling the MVC Framework
- Creating the Identity Database
- Security and Deployment
- Exam
- Project
- Project
- Dealing with the database
- Dealing with the database
- Project
- Project
- Exam

CSDE412 Computer Vision (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- Computer Imaging
- Why we study computer vision
- How computer dealing with Images
- The relation between Image processing and computer vision
- Add and Remove Noise to images
- Edge Detection
- Solving more examples
- Exam
- Feature Extraction
- The relation between Computer Vision and A.I
- Segmentation
- The skill of preparing the necessary algorithms for filters
- Image Compression
- Solving more examples
- Exam

- **Course Description** :Computer Imaging, Why we study computer vision?, How computer dealing with Images, The relation between Image processing and computer vision, Add and Remove Noise to images, Edge Detection, Feature Extraction, The relation between Computer Vision and A.I, Segmentation, Image Compression
- **Course Outcomes :**

- Understanding how images are represented and displayed on the screen..
- Understanding and acquiring knowledge of using the MATLAB package in image processing applications.
- Understanding and acquiring knowledge of different methods of image processing.
- Understanding and gaining knowledge of various algorithms used in image processing.
- Providing the student with the skill of representing two-dimensional arrays.

- **Recommended Textbook(s):**
    - Scott E Umbaugh :Digital Image Processing and Analysis Applications with MATLAB® and CVIPtools Third Edition (2018 كتاب منهجي

Prerequisites: CSDC406

- **Lab. Topics:**

- Computer Imaging
- Why we study computer vision
- How computer dealing with Images
- The relation between Image processing and computer vision

- Add and Remove Noise to images
- Edge Detection
- Solving more examples
- Feature Extraction
- The relation between Computer Vision and A.I
- Segmentation
- The skill of preparing the necessary algorithms for filters
- Image Compression

CCIT068 Project in CS (6-0-0-12)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- **Course Outcomes :**

1. Research Skills: Demonstrate the ability to conduct independent research, including literature review, data collection, and analysis, to explore a specific topic or problem.
2. Problem-Solving Skills: Identify and define a real-world problem or challenge, develop innovative solutions, and evaluate their feasibility and effectiveness.

3. Critical Thinking: Apply critical thinking skills to evaluate existing knowledge, analyze data, and draw logical conclusions based on evidence.
4. Communication Skills: Effectively communicate project objectives, methodologies, findings, and recommendations in both written and oral formats to diverse audiences.
5. Project Management: Plan, organize, and execute a complex project, including setting goals, managing resources, and meeting deadlines.
6. Analytical Skills: Apply appropriate analytical tools and techniques to interpret and analyze data collected during the project.
7. Technical Competence: Apply knowledge and skills gained from the academic program to design, implement, and evaluate a practical solution or prototype.
8. Professional Ethics: Adhere to ethical guidelines and demonstrate professional integrity throughout the project, including the responsible handling of data and respect for intellectual property.
9. Teamwork and Collaboration: Collaborate effectively with project team members, demonstrate interpersonal skills, and contribute to a positive team dynamic.
10. Self-Management: Take responsibility for self-directed learning, time management, and reflection on personal and professional development throughout the project.

- **Recommended Textbook(s):**

Problem , Objective, methodology and results.

Prerequisites:

- **Lab. Topics:**

UOA401 English 4 (3-2-0-2)

- Designation as a „required" or „elective" course:
This is a required course for the computer science department .

- **Course Topics :**

- No place like home
- Been there, done that!
- What a story!
- Nothing but the truth
- An eye to the future
- Making it big
- Getting on together
- Going to extremes
- Things ain't what they used to be
- Risking life and limb
- In your dreams
- It's never too late
- Quizzes
- Exam
- Review

- **Course Description and Course Outcomes**:

- Advanced Reading Skills:
    - Analyzing and interpreting complex texts, including literary works, academic articles, and media sources.
    - Developing strategies for effective reading comprehension, such as skimming, scanning, and note-taking.
    - Identifying main ideas, supporting details, and implicit meanings in texts.
    - Evaluating the credibility and validity of sources.
- Writing Proficiency:
    - Developing advanced writing skills, including essay structure, argumentation, and organization.
    - Enhancing grammar and sentence structures for clarity and coherence.
    - Conducting research and integrating credible sources into written work.
    - Refining editing and proofreading techniques for error-free writing.
- Oral Communication:
    - Delivering engaging and persuasive presentations on various topics.
    - Participating in debates and discussions, expressing and defending opinions.

    - Improving pronunciation, intonation, and fluency in spoken English.
    - Enhancing active listening skills and responding appropriately to others.
- Grammar and Vocabulary:
    - Reviewing and reinforcing advanced grammar concepts, such as complex sentence structures, verb forms, and conditional clauses.
    - Expanding vocabulary through targeted exercises and activities.
    - Developing word usage skills, including synonyms, antonyms, and idiomatic expressions.
    - Using domain-specific terminology and academic vocabulary accurately.
- Critical Thinking and Analysis:
    - Developing critical thinking skills by evaluating arguments, detecting bias, and identifying logical fallacies.
    - Analyzing and synthesizing information from multiple sources.
    - Engaging in critical discussions and debates on social and ethical issues.
    - Applying critical thinking skills to literary texts for interpretation and analysis.
- Literary Analysis:
    - Exploring a range of literary genres, including novels, short stories, poetry, and drama.
    - Analyzing themes, literary techniques, and character development.
    - Examining the social, historical, and cultural contexts of literary works.
    - Comparing and contrasting different literary texts and authors.
- Cultural Awareness:
    - Investigating diverse cultural perspectives and identities through literature.
    - Examining the impact of cultural, historical, and social contexts on literary works.
    - Developing cultural sensitivity and empathy towards different cultures.
    - Exploring multiculturalism and diversity in literature and society.
- Independent Learning:
    - Developing effective study skills, time management, and goal setting.
    - Engaging in self-directed research projects and presentations.
    - Reflecting on learning progress and setting personal learning goals.
- • Utilizing digital resources and technology tools for independent learning.

- **Recommended Textbook(s):**

New Headway Plus upper Intermediate, Liz and JohnSoars, Oxford University Press, 2009..

Prerequisites: UOA301

- **Lab. Topics**