Low-Level Design of Software	Code & No:	CS 431
	Credits:	3 (3,0,1)
	Pre-requisite:	CS 360
	Co-requisite:	None
	Level:	9 or 10

Course Description:

This course is designed to teach the disciplined process of software development, from formal specification through to working systems. Topics include:

- Fundamentals of Software Design-Principles and Rules
- Software Design-Practices
- Object-Oriented Programming in C++
- Program Style and Structure, Selection of Data Structures
- Algorithms-Categories, Design Methodologies
- Modularization, Detailed Design and Implementation

Course Aims:

1. Knowledge of Software Design-Principles and Rules
2. Understanding of Software Design-Practices
3. Quick overview of Object-Oriented Programming in C++
4. Have an understanding of Program Style and Structure, Selection of Data Structures
5. To know about Algorithms-Categories, Design Methodologies
6. Practice Modularization, Detailed Design and Implementation

Student Outcomes (SOs):

☐(a) An ability to apply knowledge of computing and mathematics appropriate to the program's student outcomes and to the discipline

☒(b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

☒(c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs

☐(d) An ability to function effectively on teams to accomplish a common goal

☐(e) An understanding of professional, ethical, legal, security and social issues and responsibilities

☐(f) An ability to communicate effectively with a range of audiences

☐(g) An ability to analyze the local and global impact of computing on individuals, organizations, and society

☐(h) Recognition of the need for and an ability to engage in continuing professional development

☒(i) An ability to use current techniques, skills, and tools necessary for computing practice.

☐(j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices. [CS]

☒(k) An ability to apply design and development principles in the construction of software systems of varying complexity. [CS]

☐(j) An ability to use and apply current technical concepts and practices in the core information technologies of human computer interaction, information management, programming, networking, and web systems and technologies. [IT]

☐(k) An ability to identify and analyze user needs and take them into account in the selection, creation, evaluation, and administration of computer-based systems. [IT]

☐(l) An ability to effectively integrate IT-based solutions into the user environment. [IT]

☐(m) An understanding of best practices and standards and their application. [IT]

☐(n) An ability to assist in the creation of an effective project plan. [IT]

Course Learning Outcomes (CLOs):

1. Learn low-level design principles important for any software developer
2. Be conversant in the language of program design patterns
3. Understand and plan development of larger software systems.
4. Learn about good OO design as well as good design in procedural languages..
5. Apply modularization principle in software development

SOs and CLOs Mapping:

CLO/SO	a	b	c	d	e	f	g	h	i	j	k	l	m	n
CLO1			√											
CLO2		√												
CLO3									√					
CLO4											√			
CLO5											√			

No.	Topics	Weeks	Teaching hours
1	Fundamentals of Software Design-Principles and Rules	2	6
2	Software Design-Practices	2	6
3	Object-Oriented Programming in C++	1	3
4	Program Style and Structure, Selection of Data Structures	3	9
5	Algorithms-Categories, Design Methodologies	3	9
6	Modularization, Detailed Design and Implementation	3	9
Total		**14**	**42**

Textbook:

- Software Design for Engineers and Scientists, John A. Robinson,Elsevier,2004

Essential references:

- Software Engineering, by Sommerville, 9th Edition, 2011, Addison Wesley
- Fundamentals of Software Engineering, 2nd Edition, by Ghezzi, Jazayeri and Mandrioli, 2002, Prentice Hall;
- Software Engineering: Practitioner's Approach, 7th Edition, Roger S. Pressman, 2010, McGrawHill