B.Tech. VI Semester
ELECTRONICS AND COMPUTER ENGINEERING

SEMESTER- VI													
S. No.	**Course Code**	**Course Title**	**Periods**			**Evaluation Scheme**				**End Semester**		**Total**	**Credits**
			L	**T**	**P**	**CT**	**TA**	**Total**	**PS**	**TE**	**PE**		
1	BCS601	Software Engineering	3	1	0	20	10	30		70		100	4
2	BECZ601	Digital Signal Processing	3	1	0	20	10	30		70		100	4
3	BCS603	Computer Networks	3	1	0	20	10	30		70		100	4
4	BECZ061/ BCDS062/ BCS063 / BECZ062	Department Elective–III	3	0	0	20	10	30		70		100	3
5		Open Elective-I	3	0	0	20	10	30		70		100	3
6	BCS651	Software Engineering Lab	0	0	2				50		50	100	1
7	BECZ651	Digital Signal Processing Lab	0	0	2				50		50	100	1
8	BCS653	Computer Networks lab	0	0	2				50		50	100	1
9	BNC601/ BNC602	Constitution of India/ Essence of Indian Traditional Knowledge	2	0	0	20	10	30		70			
		Total	**17**	**3**	**6**							**800**	**21**
Minor Degree/Honors Degree MT-1/HT-1													
***The Mini Project or internship (4 weeks) will be done during summer break after 6th Semester and will be assessed during 7th semester.** ***It is desirable that the students should do their Summer Internship and Minor Project in their specialization area in line with the B.Tech. program.**													

Departmental Elective-I

1. BECZ051 Optical Communication System
2. BCS052 Data Analytics
3. BCS053 Computer Graphics
4. BECZ052 Advance Digital Design using Verilog

Departmental Elective-II

1. BECZ053 VLSI Technology
2. BCIT054 Artificial Intelligence
3. BCS057 Image Processing
4. BECZ054 Integrated Circuit Design

Department Elective-III

1. BECZ061 Microcontroller for Embedded System Design
2. BCDS062 Machine Learning Techniques
3. BCS063 Blockchain Architecture & Design
4. BECZ062 Industrial Electronics

B.Tech 3rd Year VI Semester Syllabus

SOFTWARE ENGINEERING (BCS601)		
Course Outcome (CO)		**Bloom's Knowledge Level (KL)**
At the end of course, the student will be able to		
CO 1	Explain various software characteristics and analyze different software Development Models.	K_1, K_2
CO 2	Demonstrate the contents of a SRS and apply basic software quality assurance practices to ensure that design, development meet or exceed applicable standards.	K_1, K_2
CO 3	Compare and contrast various methods for software design	K_2, K_3
CO 4	Formulate testing strategy for software systems, employ techniques such as unit testing, Test driven development and functional testing.	K_3
CO 5	Manage software development process independently as well as in teams and make use of Various software management tools for development, maintenance and analysis.	K_5
DETAILED SYLLABUS		**3-1-0**
Unit	**Topic**	**Proposed Lecture**
I	**Introduction:** Introduction to Software Engineering, Software Components, Software Characteristics, Software Crisis, Software Engineering Processes, Similarity and Differences from Conventional Engineering Processes, Software Quality Attributes. Software Development Life Cycle (SDLC) Models: Water Fall Model, Prototype Model, Spiral Model, Evolutionary Development Models, Iterative Enhancement Models.	**08**
II	**Software Requirement Specifications (SRS):** Requirement Engineering Process: Elicitation, Analysis, Documentation, Review and Management of User Needs, Feasibility Study, Information Modelling, Data Flow Diagrams, Entity Relationship Diagrams, Decision Tables, SRS Document, IEEE Standards for SRS. Software Quality Assurance (SQA): Verification and Validation, SQA Plans, Software Quality Frameworks, ISO 9000 Models, SEI-CMM Model.	**08**
III	**Software Design:** Basic Concept of Software Design, Architectural Design, Low Level Design: Modularization, Design Structure Charts, Pseudo Codes, Flow Charts, Coupling and Cohesion Measures, Design Strategies: Function Oriented Design, Object Oriented Design, Top-Down and Bottom-Up Design. Software Measurement and Metrics: Various Size Oriented Measures: Halestead's Software Science, Function Point (FP) Based Measures, Cyclomatic Complexity Measures: Control Flow Graphs.	**08**
IV	**Software Testing:** Testing Objectives, Unit Testing, Integration Testing, Acceptance Testing, Regression Testing, Testing for Functionality and Testing for Performance, TopDown and Bottom-Up Testing Strategies: Test Drivers and Test Stubs, Structural Testing (White Box Testing), Functional Testing (Black Box Testing), Test Data Suit Preparation, Alpha and Beta Testing of Products. Static Testing Strategies: Formal Technical Reviews (Peer Reviews), Walk Through, Code Inspection, Compliance with Design and Coding Standards.	**08**

V	**Software Maintenance and Software Project Management:** Software as an Evolutionary Entity, Need for Maintenance, Categories of Maintenance: Preventive, Corrective and Perfective Maintenance, Cost of Maintenance, Software Re- Engineering, Reverse Engineering. Software Configuration Management Activities, Change Control Process, Software Version Control, An Overview of CASE Tools. Estimation of Various Parameters such as Cost, Efforts, Schedule/Duration, Constructive Cost Models (COCOMO), Resource Allocation Models, Software Risk Analysis and Management.	**08**

Text books:

1.RS Pressman, Software Engineering: A Practitioners Approach, McGraw Hill.
2. Pankaj Jalote, Software Engineering, Wiley
3. Rajib Mall, Fundamentals of Software Engineering, PHI Publication.
4. KK Aggarwal and Yogesh Singh, Software Engineering, New Age International Publishers.
5. Ghezzi, M. Jarayeri, D. Manodrioli, Fundamentals of Software Engineering, PHI Publication.
6. Ian Sommerville, Software Engineering, Addison Wesley.
7. Kassem Saleh, "Software Engineering", Cengage Learning.
8. P fleeger, Software Engineering, Macmillan Publication

Software Engineering Lab (BCS651)		
Course Outcome (CO)		**Bloom's Knowledge Level (KL)**
At the end of course, the student will be able to		
CO 1	Identify ambiguities, inconsistencies and incompleteness from a requirements specification and state functional and non-functional requirement	K_2, K_4
CO 2	Identify different actors and use cases from a given problem statement and draw use case diagram to associate use cases with different types of relationship	K_3, K_5
CO 3	Draw a class diagram after identifying classes and association among them	K_4, K_5
CO 4	Graphically represent various UML diagrams, and associations among them and identify the logical sequence of activities undergoing in a system, and represent them pictorially	K_4, K_5
CO 5	Able to use modern engineering tools for specification, design, implementation and testing	K_3, K_4

DETAILED SYLLABUS

For any given case/ problem statement do the following;

1. Prepare a SRS document in line with the IEEE recommended standards.
2. Draw the use case diagram and specify the role of each of the actors. Also state the precondition, post condition and function of each use case.
3. Draw the activity diagram.
4. Identify the classes. Classify them as weak and strong classes and draw the class diagram.
5. Draw the sequence diagram for any two scenarios.
6. Draw the collaboration diagram.
7. Draw the state chart diagram.
8. Draw the component diagram.
9. Perform forward engineering in java. (Model to code conversion)
10. Perform reverse engineering in java. (Code to Model conversion) 11. Draw the deployment diagram.

Note: The Instructor may add/delete/modify/tune experiments, wherever he/she feels in a justified manner
It is also suggested that open source tools should be preferred to conduct the lab (Open Office , Libra , Junit, Open Project , GanttProject , dotProject, AgroUML, StarUML etc.)

Software Engineering Lab (BCS-651): Mapping with Virtual Lab

Name of the Lab	Name of the Experiment
Software Engineering Lab (BCS-651)	Identifying the Requirements from Problem Statements
	Estimation of Project Metrics
	Modeling UML Use Case Diagrams and Capturing Use Case Scenarios
	E-R Modeling from the Problem Statements
	Identifying Domain Classes from the Problem Statements
	Statechart and Activity Modeling
	Modeling UML Class Diagrams and Sequence diagrams
	Modeling Data Flow Diagrams
	Estimation of Test Coverage Metrics and Structural Complexity
	Designing Test Suites