COMPUTER NETWORKS AND SECURITY

Instruction	:	3 Periods / week	Continuous Internal Evaluation	:	30 Marks
Tutorial	:		Semester End Examination	:	70 Marks
Credits	:	3	Semester End Exam Duration	:	3 Hours

Course Objectives:

1. To introduce communication reference models and to understand the characteristics of the transmission media.
2. To reason out the protocol failures during data transmission between two adjacent terminals
3. To solve the optimal route establishment problems for data delivery using relevant metrics
4. To serve data at the end point level and to ensure reliable data delivery mechanisms
5. To model secured exchange of high level data between two applications

Unit I – Introduction
Uses of Computer Networks, Network Hardware, Network Software, Reference Models, Example Networks. Physical Layer - Guided Transmission Media, Wireless Transmission.

Unit II - Data Link Layer
Data Link Layer Design Issues, Error Detection and Correction, Elementary Data Link Protocols, Elementary Data Link Protocols, Sliding Window Protocols. Example of data link protocols. Medium Access Control Sub-Layer **-**The Channel Allocation Problem, Multiple Access Protocols, Ethernet, Wireless LANs.

Unit III - Network Layer
Network Layer Design Issues, Routing Algorithms, Congestion Control Algorithms, Quality of Service. Internetworking, Network Layer in the Internet.

Unit IV - Transport Layer
Transport Service, Elements of Transport Protocols, Internet Transport Protocols- User Datagram Protocol, Transmission Control Protocol. Application Layer **-**Domain Name System, Electronic Mail, World Wide Web. Simple Network Management Protocol.
.
Unit V - Application Layer and security
Cryptography, security services, message confidentiality, message integrity, message authentication, digital signature.

Course Outcomes: At the end of the course, the student should be able to

CO 1 : Appreciate communication reference models along with PDUs and design a physical network based on the topological specifications using various communication media.
CO 2 : Design solutions for logical link layer transmission and control errors, control the medium access patterns through an established methodology.
CO 3 : Solve the optimal routing problems for static and dynamic networks and realizing various QoS parameters.
CO 4 : Demonstrate the significance of various applications using TCP and UDP protocol.
CO 5 : Implement CIA security mechanisms, and Network Security protocols.

Text Books:

1. Andrew S Tanenbaum, *Computer Networks*, 4th Edition, Pearson Education, 2011.
2. Behrouz A. Forouzan, *Data Communications and Networking*, 5th Edition, TMH, 2009.

References:

1. James F. Kurose, and K.W.Ross, *Computer Networking: A Top-Down Approach*, 7th Edition, Pearson Education, 2017.
2. W.Tomasi, *Introduction to Data Communications and Networking*, Pearson Education, 2009.
3. S. Keshav, *Engineering Approach to Computer Networks*, 2nd Edition, Pearson Education, 2008.

SOFTWARE ENGINEERING

(Common to CSE &IT)

Instruction	:	3 Periods / week	Continuous Internal Evaluation	:	30 Marks
Tutorial	:		Semester End Examination	:	70 Marks
Credits	:	3	Semester End Exam Duration	:	3 Hours

Course Objectives:

1. To help the student differentiate between the programming approach and the software engineering approach and introduce the issues while building large programs.
2. To introduce basic concepts of software engineering through – project, product, process models, personal software process, team software process, umbrella activities.
3. To elaborate techniques and processes for software requirements, design methodologies, coding and testing methodologies, software metrics and quality.
4. To make the students understand how the applications of software engineering principles would improve the quality of software and decrease the cost of software development and maintenance.

UNIT I: The Nature of Software, **Software Engineering, The Software Process**

The Nature of Software: Defining Software, Software Application Domains, Legacy Software, The Changing Nature of Software. Software Engineering: Defining the Discipline the Software Process, Software Engineering Practice, Software Development Myths. Process Models**:** Prescriptive Process Models, Specialized Process Models, Component-Based Development, The Unified Process, Personal and Team Process Models, Agile Development: What Is Agility? Agility and the Cost of Change, What Is an Agile Process? Extreme Programming, Other Agile Process Models.

UNIT II: Understanding Requirements, Requirements Modeling: Scenario-Based Methods, Class-Based Methods, Behavioral methods

Understanding Requirements: Requirements Engineering, Establishing the Groundwork, Eliciting Requirements, Developing Use Cases, Building the Analysis Model, Negotiating Requirements, Requirements Monitoring, Validating Requirements, Avoiding Common Mistakes. Scenario-Based Methods: Requirements Analysis, Scenario-Based Modeling, UML Models That Supplement the Use Case. Class-Based Methods: Identifying Analysis Classes, Specifying Attributes, Defining Operations, Class-Responsibility-Collaborator Modeling, Associations and Dependencies, Analysis Packages. Behavioral Methods: Creating a Behavioral Model, Identifying Events with the Use Case, State Representations, Patterns for requirement modeling.

UNIT III: Design Concepts, Architectural Design, Component-Level Design, User Interface Design

Design Concepts: Design within the Context of Software Engineering, The Design Process, Design Concepts, The Design Model. Architectural Design: Software Architecture, , Architectural Genres, Architectural Styles, Architectural Considerations, Architectural Decisions, Architectural Design, Architectural Reviews, Lessons Learned, Pattern-based Architecture Review, Architecture Conformance Checking, Agility and Architecture. Component-Level Design: Definition of Component, Designing Class-Based Components, Conducting Component-Level Design, Designing Traditional Components, Component-Based Development. User Interface Design: The Golden Rules, User Interface Analysis and Design, Interface Analysis, Interface Design Steps.

UNIT IV: Software Testing Strategies, T**esting Conventional Applications, Quality Concepts**

Software Testing Strategies: A Strategic Approach to Software Testing, Strategic Issues, Test Strategies for Conventional Software, Test Strategies for Object-Oriented Software, Context, Validation Testing, System Testing, The Art of Debugging. Testing Conventional Applications: Software Testing Fundamentals, Internal and External Views of Testing, White-Box Testing, Basis Path Testing, Control Structure Testing, Black-Box Testing. Quality Concepts: Definition of Quality, Software Quality, The Software Quality Dilemma, Achieving Software Quality.

UNIT V: Product Metrics, Process and Project Metrics, Risk Management

Product Metrics: A Framework for Product Metrics, Metrics for the Requirements Model, Metrics for the Design Model, Metrics for Source Code, Metrics for Testing, Metrics for Maintenance. Process and Project Metrics: Metrics in the Process and Project Domains, Software Measurement, Metrics for Software Quality, Integrating Metrics within the Software Process, Metrics for Small Organizations. Risk Management: Reactive versus Proactive Risk Strategies, Software Risks, Risk Identification, Risk Refinement, Risk Mitigation, Monitoring, and Management, The RMMM Plan.

Course Outcomes: At the end of the course, the student should be able to

CO 1 : Have a clear understanding of underlying principles of software engineering, software myths and thorough understanding of SE process models
CO 2 : Have exposure to requirements engineering process and related system models.
CO 3 : Appreciate software design process, design quality, design models and will be able to create architectural designs, component designs and UI designs.
CO 4 : Develop a strategic approach to testing and will be able to appreciate the art of debugging.
CO 5 : Understand the importance of software metrics and risk management.

Text Books:

1. Roger S. Pressman, *Software Engineering, A practitioner's Approach*, 8th edition, McGraw-Hill International Edition,2014.
2. Ian Sommerville, *Software Engineering*, 10th edition, Pearson Education,2017.

References:

1. Pankaj Jalote, *Software Engineering A Precise Approach*, 3rd Edition Wiley India, 2010.
2. Waman S Jawadekar, *Software Engineering A Primer*, Tata McGraw-Hill, 2010.
3. Rajib Mall, *Fundamentals of Software Engineering*, 3rd Edition, PHI, 2009.

WEB TECHNOLOGIES

(Common to CSE & IT)

Instruction	:	3 Periods / week	Continuous Internal Evaluation	:	30 Marks
Tutorial	:		Semester End Examination	:	70 Marks
Credits	:	3	Semester End Exam Duration	:	3 Hours

Course Objectives:

1. To learn the basics of HTML elements
2. To learn the basics of java Console and GUI based programming
3. To introduce XML and processing of XML Data with Java
4. To introduce Server-side programming with Java Servlets and JSP
5. To introduce Client-side scripting with JavaScript and AJAX

UNIT I

HTML 5: Semantic Elements, div, span, storage, Components of Web Browser, http status codes.
CSS 3: Syntax structure, using style sheets, box model, Grid, Flexbox. Responsive Web Design using Media Queries, use of viewport, Transition, Animation. CSS Framework: Bootstrap.

UNIT II

JavaScript: Introduction to JavaScript, data types, functions, Arrays, Objects, Regular expressions, Prototypal Inheritance, understanding callbacks, Promise, Closure, XHR request response, Asynchronous Task in JS.
Introduction to Jquery: Syntax, selectors, events, effects.

UNIT III

XML: Syntax, namespaces, DTD, Schema.
Database Technologies: JDBC Drivers and types, JDBC Configuration (Database URLs, registering a driver, connecting to a database), Executing SQL statements (Statement and ResultSet classes), query execution (prepared statements and callable statements), Scrollable and Updatable result sets, row sets, Meta data, transactions.

UNIT IV

Web servers: An introduction to Web Servers, Web application structure and deployment in Tomcat.
Servlet Technology: Servlets, Servlet lifecycle, The Servlet API packages and class and interface hierarchy, Basic servlet program template, Handling requests and responses, Using form parameters, Using ServletContext and ServletConfig objects, Using initialization parameters (both context and config level), Session management (Cookies, Http Session, URL Rewriting, Hidden Form fields), Security issues, Servlet Listeners and Filters.

UNIT V

JSP Technology: The Anatomy of a JSP Page, JSP Lifecycle, Scripting elements: scriplets, expressions, declarations, comments, JSP Directives, JSP Standard actions, JSP Implicit objects, JSP page scope, JSTL Concepts.

Course Outcomes: At the end of the course, the student will be able to

CO 1 : Build a custom website with HTML, CSS, and Bootstrap.
CO 2 : Demonstrate JavaScript, XML, DHTML and related Technologies.
CO 3 : Implement the Database Connectivity and Component Technologies like Beans.
CO 4 : Develop and deploy Servlet based web applications.
CO 5 : Develop Server-side programming using JSP.

Text Books:

1. Jon Duckett, *Beginning HTML, XHTML, CSS, and JavaScript*, Wrox Publications, 2010
2. Bryan Basham, Kathy Sierra and Bert Bates, *Head First Servlets and JSP*, O'Reilly Media,2nd Edition, 2008.

3. Cay Horstmann and Gary Cornell, *Core Java: Volume II – Advanced Features*, Prentice Hall, 9th Edition, 2013 (Only Chapter 4 for Database Programming)

References:

1. Ben Frain, "*Responsive Web Design with HTML5 and CSS3*" , Second Edition, Packt Publishing ,2015
2. E-resource: http://www.w3schools.com/
3. Martin Hall and Larry Brown, *Core Servlets and JSPs Volume I and II*, Pearson.

DESIGN AND ANALYSIS OF ALGORITHMS

(Common to CSE & IT)

Instruction	:	3 Periods / week	Continuous Internal Evaluation	:	30 Marks
Tutorial	:		Semester End Examination	:	70 Marks
Credits	:	3	Semester End Exam Duration	:	3 Hours

Course Objectives:

1. To emphasize upon the demands of real-world problems in engineering solutions
2. To make students conversant with the various paradigms of algorithms
3. To handcraft the performance analysis of designed solutions
4. To take students through various optimization principles of ill-posed problems

UNIT I Fundamentals of algorithm analysis: introduction- Definition of algorithm, algorithmic problem solving, framework for analysis of algorithm- brute force and store and reuse method. Asymptotic notations- o, Ω, and θ notations, properties of asymptotic notations. Recursive algorithms and recurrence relations- toh problem.

UNIT II Algorithm paradigms: Divide and Conquer- control abstraction, binary search algorithm and its complexity, max min search, stable and unstable algorithms, quick sort, its complexity, convex hull problem. Depth first search(dfs), topological sorting, breadth first search (bfs), articulation points

UNIT III Greedy paradigm: control abstraction, fractional knapsack problem, job sequencing problem, minimum spanning tree, Prims algorithm, Disjoint subsets, kruskal's algorithm, Huffman code, Djkstra's algorithm.

UNIT IV Dynamic programming: control abstraction, Multistage Graphs, OBST, Matrix chain multiplication, multiplicative optimization, reliability design, 0/1 knapsack problem

UNIT V Optimization problems as search problems: Back Tracking, n-queens problem, Graph coloring, Subset-sum modeling Branch and Bound- introduction, Data structure support and TSP. Randomized algorithms, p- problems, tractable algorithms, Definition of different NP problems

Course Outcomes: At the end of the course, the student should be able to

CO 1	:	Analyze worst-case running times using asymptotic analysis of algorithms
CO 2	:	Describe the divide-and-conquer paradigm and Synthesize divide-and-conquer algorithms.
CO 3	:	Define optimization problems and solve them through various greedy policies
CO4	:	Describe the dynamic-programming paradigm and synthesize dynamic-programming algorithms, and analyze them.
CO5	:	Reduce size of search space of the optimization problems by applying backtracking and branch and bound tools. Appreciate the Non-Deterministic modeling of algorithms

Text Books:

1. E. Horowitz and S.Sahni, *Fundamentals of algorithms*, 2nd edition Galgotia Publications, 2010
2. T.H.Cormen, C.E.Leiserson, R.L.Rivest, and C.Stein, *Introduction to algorithms*, 2nd edition, PHI/Pearson Education, 2001.

References:

1. Michael T.Goodricoh and Roborto Tamassia, Algorithm Design: Foundations, Analysis and Internet Examples, Wiley India, 2006.
2. Alfred V. *Aho,* John E. *Hopcroft* and Jeffrey D. *Ullman*, *The Design and Analysis of Computer Algorithms*, PHI/Pearson Education, 1974.

COMPILER DESIGN

Instruction	:	3 Periods / week	Continuous Internal Evaluation	:	30 Marks
Tutorial	:		Semester End Examination	:	70 Marks
Credits	:	3	Semester End Exam Duration	:	3 Hours

Course Objectives:

1. To describe the steps and algorithms used by language translators.
2. To discuss the effectiveness of code optimization techniques.
3. To acquire knowledge on various phases of compiler.
4. To explain various tools and techniques on design aspects of Compiler.

UNIT I

Overview of Compilation: Phases of Compilation - Lexical Analysis, Regular Grammar and regular expressions, pass and Phases of translation, interpretation, bootstrapping, data structures in compilation – LEX lexical analyzer generator. Syntax Analyzer, Top down Parsing, Context free grammars, recursive descent parsing, LL (1) parser, Predictive parsing.

UNIT II

Bottom up parsing: Operator precedence parsing, Shift Reduce parsing, LR and LALR parsing, , handling ambiguous grammar, YACC – automatic parser generator. Intermediate forms of source Programs, abstract syntax tree, polish notation and three address codes. Language Constructs into Intermediate code forms.

UNIT III

Semantic analysis: Attributed grammars, Syntax directed translation, Type checker. Symbol Tables: Symbol table format, organization for block structure languages, hashing, tree structure representation of scope information. Block structures and non-block structure storage allocation: Static, Runtime stack and heap storage allocation, storage allocation for arrays, strings and records.

UNIT IV

Code optimization: Scope of Optimization, local optimization, loop optimization, frequency reduction, folding, DAG representation. Data flow analysis: Flow graph, data flow equation, global optimization, redundant sub expression elimination, Induction variable elements, Live variable analysis, Copy propagation.

UNIT V

Object code generation: Object code forms, machine dependent code optimization, Peephole Optimization, register allocation and assignment, generic code generation algorithms, Introduction to Assembly Language Instructions, Addressing Modes, Conversion of three address code to Assembly Language Instructions.

Course Outcomes: At the end of the course, the student should be able to

CO 1 : Understand the design approaches and practical aspects of a compiler.
CO 2 : Implement algorithms on syntax analyzer
CO 3 : Implement algorithms on various parsing techniques
CO4 : Apply various code optimization techniques.
CO5 : Articulate the generation of assembly and intermediate code of programming language.

Text Books:

1. A.V. Aho and J.D.Ullman, Principles *of compiler design*, 2nd Edition, Pearson Publication. 2009

2. C, Andrew and W. Appel, *Modern Compiler Implementation in C*, Cambridge University Press. 2004

References:

1. John R. Levine, Tony Mason, Doug Brown, *lex & yacc*, 2nd Edition, O'reilly 2017
2. Grune D, Van Reeuwijk, et al.. *Modern compiler design*. Springer Science & Business Media; 2012.
3. Keith Cooper and Linda Torczon, *Engineering a Compiler*, 2nd Edition, Morgan Kaufmann 2011

WEB TECHNOLOGIES LAB

(Common to CSE & IT)

Instruction	:	3 Periods / week	Continuous Internal Evaluation	:	30 Marks
Tutorial	:		Semester End Examination	:	70 Marks
Credits	:	1.5	Semester End Exam Duration	:	3 Hours

Course Objectives:

1. To learn the basics of HTML elements
2. To learn the basics of java Console and GUI based programming
3. To introduce XML and processing of XML Data with Java
4. To introduce Server-side programming with Java Servlets and JSP
5. To introduce Client-side scripting with JavaScript and AJAX.

List of Experiments:

1. **Working with Static web pages.**
   Create a web page using the advanced features of CSS: Grid, Flexbox. And apply transition and animations on the contents of the web page

2. Make the web pages created in the above experiment as responsive web page with Bootstrap Framework.

3. **Working with Java Script and Regular Expressions**
   Validate the registration, user login, user profile and payment pages using JavaScript. Make use of any needed JavaScript objects.

4.
   a) Build a scientific calculator.
   b) Working of prototypal inheritance, closure, callbacks, promises and async / await.

5. **Working with XML**
   a) Write an XML file which will display the Book information with the following fields: Title of the book, Author Name, ISBN number, Publisher name, Edition, Price

   Working with DTDs (Validating XML document)
   b) Define a Document Type Definition (DTD) and XML schema to validate the above created XML Documents

6. **Developing JDBC based Applications**
   a) Write a java program to establish a connection to a database and execute simple SQL queries.
   b) Write a java program to demonstrate the usage of JDBC in performing various DML statements. Use prepared statements and callable statements.

7.
   a) Write a java-based application to demonstrate the Updatable and Scrollable result sets.
   b) Write a java program to access meta data of the SQL database.

8.
   a) Write a program to accept request parameters from a form and generate the response.
   b) Write a program to accept ServletConfig and ServletContext parameters.

9. Assume four users user1, user2, user3 and user4 having the passwords pwd1, pwd2, pwd3 and, pwd4 respectively. Write a servlet for doing the following functionalities

a) Create a Cookie and add these four user ids and passwords to this Cookie.
b) Read the user id and password entered into the Login form and authenticate with the values (user id and passwords) available in the cookies. If the person is a valid user (i.e., user-name and password match) you should welcome by name (user-name) else you should display the message "You are not an authenticated user ".

10.

a) Develop a servlet to demonstrate the database access and update from a database.
b) Create a servlet to implement an authentication filter mechanism.
c) Develop a servlet to implement servlet context and session listeners.

11. Working with JSP

Write a JSP which does the following job:

a) Insert the details of the three users who register with the web site by using registration form.
b) Authenticate the user when he submits the login form using the username and password from the database.

12.

a) Write a JSP to demonstrate the usage of JSP standard actions.
b) Write a JSP to show the usage of various scripting elements.

13. Design and use a custom tag library.

14. Design a simple application using both Servlets and JSPs along with database access.

Course Outcomes: At the end of the course, the student should be able to

CO 1 : Build a custom website with HTML, CSS, and Bootstrap.
CO 2 : Demonstrate JavaScript, XML, DHTML and related Technologies.
CO 3 : Implement the Database Connectivity and Component Technologies like Beans.
CO 4 : Deploy the servlet technology & API.
CO 5 : Construct the fundamentals of JSP.

References:

COMPILER DESIGN AND ALGORITHMS LAB

Instruction	:	3 Periods / week	Continuous Internal Evaluation	:	30 Marks
Tutorial	:		Semester End Examination	:	70 Marks
Credits	:	1.5	Semester End Exam Duration	:	3 Hours

List of Experiments:

1. Implement Merge sort algorithm and plot its time complexity with reference to size of the input
2. Implement Quick sort algorithm and plot its time complexity with regard to asymptotic notations (Best, average, and worst)
3. Tiling Problem using Divide and Conquer algorithm: Given a n by n board where n is of form 2k where k >= 1 (Basically n is a power of 2 with minimum value as 2). The board has one missing cell (of size 1 x 1). Fill the board using L shaped tiles. A L shaped tile is a 2 x 2 square with one cell of size 1×1 missing.
4. Write a program that detects the number of islands present in an image
5. Simulate different Framing Algorithms
6. From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm
7. Write a program to identify the articulation points present in a graph
8. From a given vertex in a weighted connected graph, find shortest paths to other vertices using Distance Routing algorithm
9. Implement Priority based Job Scheduling algorithm
10. Given a value N, if we want to make change for N cents, and we have infinite supply of each of S = { S1, S2, .. , Sm} valued coins, how many ways can we make the change? The order of coins doesn't matter.
11. Implement OBST using dynamic programming
12. Write a C program to identify whether a given line is a comment or not.
13. Write a program to find the number of whitespaces & newline characters.
14. Write a C program to test whether a given identifier is valid or not
15. Write a C program to check whether a given string is a keyword or not
16. To write a program to implement the Lexical Analyzer using LEX tool.
17. Write a C program to develop a lexical analyzer to recognize a few patterns in C.
18. Write a C program to Implement a top-down parser (Recursive Descent Parser)
19. Write a C program to implement operator precedence parsing.
20. To write a C program to construct of DAG (Directed Acyclic Graph)
21. Write a C program to generate a three address code for a given expression.
22. Write a C program to generate an assembly language instruction from three address code.

Course Outcomes: At the end of the course, the student should be able to

CO 1	:	Solve Sorting and Searching problems using Divide and Conquer Model.
CO 2	:	Solve shortest path finding problem using Greedy Methods.
CO 3	:	Explore large search spaces using Dynamic programming and Brach and Bound paradigms.
CO 4	:	Construct of DAG (Directed Acyclic Graph).
CO 5	:	Generate assembly language instruction from three address code.

References:

CASE TOOLS LAB

Instruction	:	2 Periods / week	Continuous Internal Evaluation	:	30 Marks
Tutorial	:		Semester End Examination	:	70 Marks
Credits	:	1	Semester End Exam Duration	:	3 Hours

Course Objectives:

1. To understand Unified Modeling language and use it for software requirements analysis and design.
2. To understand and use software configuration management tool GIT.
3. To gain knowledge of agile modeling and use Jenkins for continuous integration.

I. ATM Case Study

1. Use Case diagram for ATM system, with use case specification.
2. Identify the classes for the ATM system and draw the Class Diagram.
3. Draw the Sequence and Collaboration diagrams for each of the use cases identified.
4. Draw the State Chart Diagram for the above system.
5. Draw the Activity Diagram for the system.
6. Draw the Component Diagram for the system.
7. Draw the Deployment Diagram for the system.
8. Forward and Reverse engineering the system

II. Online Railway Reservation System

1. Use Case diagram for Railway Reservation system, with use case specification.
2. Identify the classes for the Railway Reservation system and draw the Class Diagram.
3. Draw the Sequence and Collaboration diagrams for each of the use cases identified.
4. Draw the State Chart Diagram for the above system.
5. Draw the Activity Diagram for the system.
6. Draw the Component Diagram for the system.
7. Draw the Deployment Diagram for the system.
8. Forward and Reverse engineering the system

III. Version Control System, GIT

1. Working Locally with GIT
2. Working Remotely with GITHUB
3. Branching and Merging
4. Resolve merge Conflict
5. GIT reset and Stash operation
6. How to setup Git on Premises Hardware
7. Use Case In Devops Environment

IV. Continuous Integration using Jenkins

1. Introduction of Jenkins
2. Install and setup Jenkins
3. Introduction About Maven project
4. Setup Jenkins with Maven Project
5. Continuous Build and Deployment

Course Outcomes: At the end of the course, the student should be able to

CO 1 : Draw various UML diagrams for analysis and design of the software.
CO 2 : Construct and evaluate hybrid CASE tools by integrating existing tools.
CO 3 : Deliver the product with quality by using GitHub and Jenkins Tools

References: