UNIVERSITY SCHOOL
OF
INFORMATION AND COMMUNICATION TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAMME STRUCTURE

M.TECH. COMPUTER SCIENCE AND ENGINEERING

SPECIALIZATION : SOFTWARE ENGINEERING

2025-2027

GAUTAM BUDDHA UNIVERSITY
GAUTAM BUDH NAGAR, GREATER NOIDA, UP, INDIA

SEMESTER I

S.No.	Course Code	Course Name	L	T	P	Credits	Types
1	CS521	Advanced Data Base Management System	3	0	0	3	CC1
2	CS523	Advanced Data Structure and Algorithm	3	1	0	4	CC2
3	CS525	Research Techniques in ICT	3	0	0	3	CC3
4	CS527	Python Programming	3	0	0	3	CC4
5	ES415	Energy and Environment	3	0	0	3	OE1 /AECC
6	CD581	Advanced Data Base Management System Lab	0	0	4	2	CC-L1
7	CD583	Python Programming Lab	0	0	4	2	CC-L2 / SEC
8	GP	General Proficiency	Non Credit				
		Total Hours and Credits	**15**	**1**	**8**	**20**	

SEMESTER II

S.No.	Course Code	Course Name	L	T	P	Credits	Types
1	CS522	Advanced Software Engineering	3	0	0	3	CC5
2	CS524	Deep Learning	3	0	0	3	CC6/ SEC
3	CS526	DevOps Engineering	3	0	0	3	CC7
5	CS528	Blockchain Technology	3	1	0	4	CC8
		Elective-1	3	0	0	3	E1/ DSE
8	CS582	Advanced Software Engineering Lab	0	0	4	2	CC-L3
9	CS584	Deep Learning Lab	0	0	4	2	CC-L4 / SEC
10	GP	General Proficiency	Non Credit				
		Total Hours and Credits	**15**	**1**	**8**	**20**	

ELECTIVES FROM DCSE

S.No.	Course Code	Course Name	L	T	P	Credits	Types
1	CS530	Software Reliability Engineering	3	0	0	3	E1
2	CS532	Software Quality Assurance	3	0	0	3	
3	CS534	Software Project Management	3	0	0	3	
4	CS536	Open Source Software Engineering	3	0	0	3	
5	CS538	Parallel and High Performance Computing	3	0	0	3	

* Semester 1 is approved in 33rd BoS and Semester 2 is approved in 35th BoS.

SEMESTER 1

ADVANCED DATABASE MANAGEMENT SYSTEM			
Course Code:	**CS521**	**Course Credits:**	**3**
Course Category:	**CC1**	**Course (U / P)**	**P**
Course Year (U / P):	**1P**	**Course Semester (U / P):**	**1P**
No. of Lectures + Tutorials (Hrs/Week):	**03 + 00**	**Mid Sem. Exam Hours:**	**1**
Total No. of Lectures (L + T):	**45 + 00**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES
1 Knowledge of database design
2 A general understanding of database ,design and dependency
3 Understanding of different types of databases
4 Knowledge of databases on the internet
5 Application on enhanced database

COURSE OUTCOMES
At the end of the course the students should be able to:
1 Basic knowledge and understanding of ER diagram and UML class diagram.
2 Ability to apply functionality and Normalization on relational database.
3 Understand and fetch data from object oriented, parallel and distributed databases.
4 Use XML and understand unstructured data
5 Implement concept and deduction of enhanced database on different applications

UNIT I INTRODUCTION TO DATABASE DESIGN

Entities, Attributes, Entity Sets, Relationships, Key Constraints, Participation Constraints, Weak Entities, UML Class Diagrams, Subclasses, Super classes, Inheritance, Specialization, Generalization, Constraints and Characteristics of Specialization and Generalization Hierarchies, Modeling of UNION Types Using Categories, Representing Specialization and Generalization In UML Class Diagrams, Data Abstraction, Knowledge Representation and Ontology Concepts.

UNIT II DATABASES DESIGN THEORY

Problems Caused by Redundancy, Decompositions, Problems Related to Decomposition, Reasoning About FD's, FIRST, SECOND, THIRD Normal Form, BCNF, Forth Normal Form, Lossless Join Decomposition, Dependency Preserving Decomposition, Schema Refinement in DataBase Design, Multi Valued Dependencies.

UNIT III OBJECT- ORIENTED, PARALLEL AND DISTRIBUTED DATABASES

Overview of Object-Oriented Concepts, Object Identity, Object Structure, Type Constructor, Encapsulation of Operations, Methods and Persistence; Architectures For Parallel Databases, Parallel Query Evaluation, Parallelizing Individual Operations, Sorting Joins, Distributed Database Concepts, Data Fragmentation, Replication and Allocation Techniques for Distributed Database Design, Query Processing in Distributed Databases, Concurrency Control and Recovery in Distributed Databases.

UNIT IV DATABASES ON THE WEB AND SEMI–STRUCTURED DATA

Web interface, XML, structure of XML data, querying XML data, storage of XML data, XML applications, semi-structured data model, indexes for text data.

UNIT V ENHANCED DATA MODELS FOR ADVANCED APPLICATIONS

Active database concepts, temporal database concepts, spatial databases: concept and architecture, deductive databases and query processing, mobile databases, Geographic Information Systems (GIS).

Text Books:

1. Elmasri and Navathe, Fundamentals of Database Systems,
2. Ramakrishnan and Gehrke, Database Management Systems,

References Books:

1. Korth, Silberschatz, Sudarshan, Database System Concepts,
2. Rob and Coronel, Database Systems: Design, Implementation and Management,
3. Date and Longman, Introduction to Database Systems,

ADVANCED DATA STRUCTURE AND ALGORITHM			
Course Code:	**CS523**	**Course Credits:**	**4**
Course Category:	**CC2**	**Course (U / P)**	**P**
Course Year (U / P):	**1P**	**Course Semester (U / P):**	**1P**
No. of Lectures + Tutorials (Hrs/Week):	**03 + 01**	**Mid Sem. Exam Hours:**	**1**
Total No. of Lectures (L + T):	**45 + 00**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES
1 Understand the appropriate data structures, ADT libraries, and use it to design algorithms for a specific problem.
2 Be capable of solving problems using abstraction techniques.
3 Be able to choose appropriate algorithms for a specific problem.
4 Be able to analyze algorithms in terms of their efficiency and correctness.
5 Understanding the recent developments in the area of algorithm design.

COURSE OUTCOMES
At the end of the course the students should be able to:
1 Design and analyze programming problem statements.
2 Choose appropriate data structures and algorithms for a specific problem.
3 Understand the necessary mathematical abstraction to solve problems.
4 Come up with analysis of efficiency and proofs of correctness.
5 Comprehend and select algorithm design approaches in a problem specific manner.

UNIT I INTRODUCTION

Review of Basic Concepts: Abstract data types, Data structures, Algorithms, Big-Oh, Small-Oh, Omega, Small-Omega and Theta Notations, finding time complexity of programs, **Recurrence Relations:** Solving Recurrence Relations, Substitution Method, Master Theorem.

UNIT II HASHING

Hashing: Review of Hashing, Hash Function, Collision Resolution Techniques in Hashing, Separate Chaining, Open Addressing, Linear Probing, Quadratic Probing, Double Hashing, Rehashing, Extendible Hashing, Recent Trends in Hashing.

UNIT III TREES & GRAPH

Trees: Binary Search Trees, AVL Trees, Red Black Trees, 2-3 Trees, B-Trees, Splay Trees, Minimum Spanning Tree (MST), Kruskal's Algorithm and Prim's Algorithm, Applications to MST.

Graph: Graph, Breadth First Search, Depth First Search, Shortest path in edge-weighted case (Dijkstra's), Bellman Ford Algorithms, Topological Sorting.

UNIT IV SELECTED TOPICS

Strassen's Matrix Multiplication, Greedy method VS Dynamic Programming, Job sequencing with deadlines, Fractional Knapsack Problem, 0/1 Knapsack Problem, Travelling Salesman Problem, Huffman coding, Pre order, Post order, Inorder traversal, Postfix to infix notation, Infix to Postfix notation.

UNIT V LINEAR PROGRAMMING & RECENT TRENDS

Linear Programming: Geometry of the feasibility region and Simplex algorithm
NP-completeness: Examples, proof of NP-hardness and NP-completeness.
Recent Trends: Recent Trends in problem solving paradigms using recent searching and sorting techniques by applying recently proposed data structures.

Text Books:

1. Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.
2. Algorithms Unlocked: Thomas H. Cormen.
3. The Algorithm Design Manual, Steven S. Skiena.

References Books:

1. Algorithms: Robert Sedgewick and Kevin Wayne.
2. Advanced Data Structures: Peter Brass.

RESEARCH TECHNIQUES IN ICT			
Course Code:	**CS 525**	**Course Credits:**	**3**
Course Category:	**CC3**	**Course (U / P)**	**P**
Course Year (U / P):	**1P**	**Course Semester (U / P):**	**1P**
No. of Lectures + Tutorials (Hrs/Week):	**03 + 00**	**Mid Sem. Exam Hours:**	**1**
Total No. of Lectures (L + T):	**45 + 00**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES
1 To get the understanding about what is research.
2 To have the knowledge about research methodology.
3 Awareness of methodology to be followed for report and paper writing.
4 Familiarization to different models and algorithms during research.
5 Conversant with various simulation and soft computing techniques.

COURSE OUTCOMES
At the end of the course the students should be able to:
1 Acquainted with procedure for carrying out the research and its importance.
2 Can identify and apply the appropriate research techniques according to domain of research.
3 Recognize models and algorithms required for their study.
4 Know about research paper writing and publication procedure.
5 Will get the direction about different simulation packages.

UNIT I INTRODUCTION TO RESEARCH TECHNIQUES

Meaning of research, objectives of research, motivation in research, types of research, characteristics and prerequisites of research, significance of research, research process, sources of research problem, criteria of identifying the problem, necessity of defining the problem, errors in selecting research problem, technique involved in defining the problem, report and paper writing.

UNIT II DATA ANALYSIS AND STATISTICAL TECHNIQUES

Data and their analyses, quantitative methods and techniques, Measure of central tendency, measures of variation, frequency distribution, analysis of variance, methods, Correlation analysis, regression analysis, time series and forecasting, introduction to discriminant analysis, factor analysis, cluster analysis, conjoint analysis, probability distribution, binomial distribution, poisson distribution, uniform distribution, exponential distribution, and normal distribution, sampling methods, test of hypothesis.

UNIT III MATHEMATICAL MODELING

Steps of modeling, operations research models like queuing theory, stochastic processes, application of models, conceptual framework development and validation techniques, optimization techniques.

UNIT IV ALGORITHMIC RESEARCH

Algorithmic research problems, types of algorithmic research, types of solution procedure, steps of algorithm development, steps of algorithmic research, design of experiments.

UNIT V SIMULATION AND SOFT COMPUTING TECHNIQUES

Introduction to soft computing, artificial neural network, genetic algorithm, fuzzy logic and their applications, tools of soft computing, need for simulation, types of simulation, simulation language, fitting the problem to simulation study, simulation models, output analysis, data simulation packages like MATLAB, NS2, ANSYS, Cadence.

Text books:

1. Research Methodology: Methods and Techniques, C.R. Kothari

Reference Books:

1. Research Methodologies, R. Panneerselvam, Prentice Hall, 2007.
2. Research in Education, Best John V. and James V Kahn, Wiley eastern, 2005.
3. Elements of Educational Research, Sukhia, S.P., P.V. Mehrotra, and R.N. Mehrotra, PHI publication, 2003.
4. Methodology of Research Education, K. Setia, IEEE publication, 2004.
5. Research methodology, Methods and Techniques, Kothari, C.R., 2000.

PYTHON PROGRAMMING			
Course Code:	**CS527**	**Course Credits:**	**3**
Course Category:	**CC4**	**Course (U / P)**	**P**
Course Year (U / P):	**1P**	**Course Semester (U / P):**	**2P**
No. of Lectures + Tutorials (Hrs/Week):	**03 + 00**	**Mid Sem. Exam Hours:**	**1**
Total No. of Lectures (L + T):	**45 + 00**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES
1. To learn and understand Python programming basics and paradigm
2. To learn and understand python looping, control statements and string manipulations.
3. Students should be made familiar with the concepts of GUI controls and designing GUI applications.
4. To learn and know the concepts of file handling, exception handling and database connectivity.
5. To learn and understand database connectivity in the python programming language.

COURSE OUTCOMES
At the end of the course the students should be able to:
1. To read and write simple Python programs.
2. To develop Python programs with conditionals and loops.
3. To define Python functions and to use Python data structures –- lists, tuples, dictionaries
4. To do input/output with files in Python
5. To do searching ,sorting and merging in Python

UNIT I Introduction:

The Programming Cycle for Python , Python IDE, Interacting with Python Programs, Elements of Python, Type Conversion. Basics: Expressions, Assignment Statement, Arithmetic Operators, Operator Precedence, Boolean Expression.

UNIT II Conditionals:

Conditional statement in Python (if-else statement, its working and execution), Nested-if statement and Elif statement in Python, Expression Evaluation & Float Representation. Loops: Purpose and working of loops , While loop including its working, For Loop , Nested Loops , Break and Continue.

UNIT III Function:

Parts of A Function , Execution of A Function , Keyword and Default Arguments ,Scope Rules. Strings : Length of the string and perform Concatenation and Repeat operations in it. Indexing and Slicing of Strings. Python Data Structure : Tuples , Unpacking Sequences , Lists , Mutable Sequences, List Comprehension, Sets, Dictionaries Higher Order Functions: Treat functions as first-class Objects, Lambda Expressions

UNIT IV File I/O :

File input and output operations in Python Programming Exceptions and Assertions Modules : Introduction, Importing Modules, Abstract Data Types: Abstract data types and ADT interface in Python Programming. Classes : Class definition and other operations in the classes , Special Methods (such as _init_, _str_, comparison methods and Arithmetic methods etc.) , Class Example, Inheritance, Inheritance and OOP.

UNIT V Iterators & Recursion:

Recursive Fibonacci , Tower Of Hanoi Search : Simple Search and Estimating Search Time , Binary Search and Estimating Binary Search Time Sorting & Merging: Selection Sort , Merge List , Merge Sort , Higher Order Sort

Text Books:

1. Allen B. Downey, ``Think Python: How to Think Like a Computer Scientist'', 2nd edition, Updated for Python 3, Shroff/O'Reilly Publishers, 2016

2. Guido van Rossum and Fred L. Drake Jr, —An Introduction to Python – Revised and updated for Python 3.2, Network Theory Ltd., 2011.

ENERGY AND ENVIRONMENT			
Course Code:	**ES415**	**Course Credits:**	**3**
Course Category:	**OE1**	**Course (U / P)**	**P**
Course Year (U / P):	**1P**	**Course Semester (U / P):**	**1P**
No. of Lectures + Tutorials (Hrs/Week):	**03 + 00**	**Mid Sem. Exam Hours:**	**1**
Total No. of Lectures (L + T):	**40+00**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES
To provide in-depth knowledge of renewable and non-renewable energy resources, their harnessing techniques and energy-environment issues.
COURSE OUTCOMES
The knowledge so gathered could be utilized to meet the challenges of energy vis-a-vis environmental security.

Sun as Source of Energy- Nature of its radiation, solar radiation and its spectral characteristics; Conventional energy sources (coal, oil, biomass and natural gas), Non-conventional energy sources (hydro-electric power, tidal, wind, geothermal, solar, nuclear magneto-hydrodynamic power MHD); Energy use pattern in India and parts of world, Energy security

Fossil Fuels: Classification, composition, physico-chemical characteristics; Calorific value – gross and net; Energy content of coal, petroleum and natural gas, shale oil, coal bed methane, gas hydrates

Concept of Green Energy; Principles of generation of hydro-power, tidal energy, ocean thermal energy conversion, wind power, geothermal energy, solar energy (solar collectors, photo-voltaic modules, solar ponds)

Nuclear Energy - Fission and fusion, Nuclear fuels, Nuclear reactor - principles and types; Mechanism of radiation action on living organisms - Stochastic and Non-stochastic effects, delayed effects; Radioactivity from nuclear reactors, fuel processing and radioactive waste, hazards related to power plants

Bioenergy: Types, importance, methods of energy production from biomass

Environmental Implications of Energy Use; CO2 emission and atmosphere -scenario in developed and developing world (and India), Global warming, Radiative forcing, Impacts of large scale exploitation of solar, wind, hydro, nuclear and bio-energy sources; National Solar Mission, National Mission for Enhanced Energy Efficiency, case studies.

Text Books:

1. Fay, J.A. and Golomb, D.S. 2011. Energy and the Environment, Oxford University Press, New Delhi.
2. Iqbal, M. 1983. An Introduction to Solar Radiation. Academic Press, New York.
3. Kaushika, N.D. and Kaushik, K. 2004. Energy, Ecology and Environment: A Technological

ADVANCED DATABASE MANAGEMENT SYSTEM LAB			
Course Code:	**CS581**	**Course Credits:**	**2**
Course Category:	**CC-L1**	**Course (U / P)**	**P**
Course Year (U / P):	**1P**	**Course Semester (U / P):**	**1P**
No. of Lab (Hrs/Week) / Total No. of Lab	**03 / 10**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES
. To explore the features of a Database Management Systems
. To interface a database with front end tools
. To understand the internals of a database system
. To provide a strong foundation in advanced database concepts from an industry perspective.
. To learn query processing and transaction management concepts for object-relational database and distributed database

COURSE OUTCOME
At the end of the course the students should be able to:
1. Develop and apply critical thinking skills.
2. Design and present Lab as well as project reports
3. Apply appropriate methods for the analysis of raw data
4. Perform logical troubleshooting as and when required.
5. Verify and implement the concepts and theory learnt in class.

List of Experiments:

1. Introduction to MySQL, PostgreSql, Microsoft Sqlsoftwares.
2. An exercise of data types in PostGresql& Data Definition Language Commands
3. Exercise on Data Manipulation Language and Transaction Control Commands using PostgreSql.
4. Exercise on Types of Data Constraints using PostgreSql.
5. Exercise on JOINS (Single-Table) Using Normalization
6. Exercise on JOINS (Multiple-Table) Using Normalization
7. Exercise on GROUP BY/ORDER BY Clause and Date Arithmetic using PostgreSql.
8. Exercise on different Functions (Aggregate, Math and String)
9. Exercise on different types of sub queries
10. Procedures, View and Triggers

PYTHON PROGRAMMING LAB			
Course Code:	**CS583**	**Course Credits:**	**2**
Course Category:	**CC-L2**	**Course (U / P)**	**U / P**
Course Year (U / P):	**1P**	**Course Semester (U / P):**	**2P**
No. of Lab (Hrs/Week) / Total No. of Lab	**03 / 10**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES
1. Interpret the use of procedural statements like assignments, conditional statements, loops and function calls.
2. Infer the supported data structures like lists, dictionaries and tuples in Python
3. Illustrate the application of matrices and regular expressions in building the Python programs.
4.Discover the use of external modules in creating excel files and navigating the file systems.
5. Describe the need for Object-oriented programming concepts in Python.

COURSE OUTCOME
At the end of the course the students should be able to:
1 To write, test, and debug simple Python programs.
2 To implement Python programs with conditionals and loops.
3 Use functions for structuring Python programs.
4 Represent compound data using Python lists, tuples, dictionaries
5 Read and write data from/to files in Python.

List of Experiments:

1. Write a python program find the maximum of a list of numbers.
2. Write a python program to perform Matrix Multiplication.
3. Write a python program first n prime number
4. Write a python program selection sort.
5. Write a python program to compute the GCD of two numbers.
6. Write a python program to find the most frequent words in a text file.
7. Write a Python program to create a scientific calculator
8. Write a Python program to print all the Disarium numbers between 1 and 100.
9. Write a Python program to encrypt the text using Caesar Cipher technique. Display the encrypted text. Prompt the user for input and the shift pattern.
10. Write a Python program to construct a linked list. Prompt the user for input. Remove any duplicate numbers from the linked list.

SEMESTER 2

ADVANCED SOFTWARE ENGINEERING			
Course Code:	**CS522**	**Course Credits:**	**3**
Course Category:	**CC**	**Course (U / P)**	**IP / P**
Course Year (U / P):	**4 IP / 1P**	**Course Semester (U / P):**	**8IP / 2P**
No. of Lectures + Tutorials (Hrs/Week):	**03**	**Mid Sem. Exam Hours:**	**1.5**
Total No. of Lectures (L + T):	**45**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES

1. To provide an idea of using various process models in the software industry according to given circumstances
2. To gain the knowledge of how Analysis, Design, Implementation, Testing and Maintenance processes are conducted in a software project
3. To analyze, and document user needs using techniques like prototyping and interviews to create Software Requirement Specifications (SRS).
4. To apply design principles and create UML diagrams (Use Case, Class, Sequence) for robust software architecture.
5. To provide the idea of decomposing the given problem into Analysis, Designing, Implementation, Testing and Maintenance phases

COURSE OUTCOMES

At the end of the course the students should be able to:

1. Implement reusability mechanisms for producing application systems.
2. Explore maintenance and reengineering approaches for legacy systems.
3. Understand the properties and methods of designing reliable and safe system.
4. Analyze and apply CASE in SDLC phases.
5. Study of advance concepts in various domains to produce an effective software system

UNIT I INTRODUCTION

Introducing to Software Reuse: What is Software Reuse?, Reuse types, Reuse Approaches, Reuse Technology, Reuse benefits & barriers, Reuse success & failure Factors, CBSE Process, Reuse Driven Software Engineering is a business.

UNIT II ARCHITECTURAL CONCEPTS FOR SOFTWARE REUSE

Architectural Concepts in Reuse: Application and component systems, Application families allow significant reuse, Developing Application Systems from Reusable Components: Reuse Variability, Facades for Component System Internals and Externals. Organizing a system in Layered Architecture.

UNIT III CONCEPTS OF SOFTWARE CHANGE AND REENGINEERING

Software change, Software Evolution, Software maintenance: Models and Metrics, Reengineering: Reengineering Process and Activities: Program Comprehension, Reverse Engineering, Restructuring, Forward Engineering, Re-documentation. Software Aging.

UNIT IV SOFTWARE DEPENDABILITY, QUALITY, AND CASE TECHNOLOGY

Software Dependability, Software Safety, Software Availability, Software Reliability: Metrics, Approaches and Models. Software Quality: Quality Factors, Verification & Validation (V&V), SQA. Computer-Aided Software Engineering (CASE): Scope and Technology, CASE support in SDLC, Second Generation CASE Tools, Architecture of a CASE Environment.

UNIT V USABILITY ENGINEEERING

Usability Engineering: HCI, Types of UI, Component-Based GUI Development, Usability Engineering Process and Methods. Aspect-Oriented Software Engineering. Cleanroom Software Engineering and Crowdsourcing. Artificial Intelligence & Machine Learning in SDLC.

Text Books:

1. Software Reuse Architecture, Process and Organization for Business Success- by Ivar Jacobson, Martin Griss, Patrick Johnson, First Edition, Pearson Education, 2000.
2. Software Engineering: Concepts & Practices- Ugrasen Suman, Cengage Learning Publications, Second Ed. 2022.
3. Software Engineering- Ian Somerville, Pearson Education Asia, 10th Edition, 2016.
4. Software Engineering-A practitioner's approach- R. S. Pressman, Tata McGraw-Hill International Editions, New York.

Reference Book:

1. Software Engineering: An Engineering Approach, by J.F.Peters and W. Pedrycz, Publisher: John Wiley and Sons.
2. Software Engineering: A Practitioner's Approach by Roger Pressman, Publisher: McGraw-Hill.
3. Fundamentals of Software Engineering by Ghezzi, Jayazeri, and Mandrioli, Publisher: Prentice-Hall.
4. Software Engineering Fundamentals by Ali Behforooz, and Frederick J.Hudson, Publisher: Oxford University Press.

DEEP LEARNING			
Course Code:	CS524	Course Credits:	3
Course Category:	CC	Course (U / P)	IP / P
Course Year (U / P):	4IP / 1P	Course Semester (U / P):	8IP / 2P
No. of Lectures + Tutorials (Hrs/Week):	3	Mid Sem. Exam Hours:	1.5
Total No. of Lectures (L + T):	45	End Sem. Exam Hours:	3

COURSE OBJECTIVES

1. Gain a historical perspective of Deep Learning and its foundations. Become familiar with basic principles of AI toward problem-solving, inference, perception, knowledge representation and learning.
2. Experience Deep Learning Algorithms and development various AI tools.
3. Explore the current scope, potential, limitations, and implications of Deep learning.
4. Analysis on the effect of deep learning practices in software systems.
5. Evaluation of deep learning algorithms

COURSE OUTCOME

At the end of the course the students should be able to:

1. Demonstrate knowledge of the building blocks of Deep Learning system.
2. Analyze and formalize the problem as a state space, graph, design heuristics, and select different search or game-based techniques to solve them.
3. Develop intelligent algorithms for constraint satisfaction problems and also design intelligent systems for Game Playing
4. Attain the capability to represent various real-life problem domains using logic-based techniques and use this to perform inference or planning.
5. To evaluate and compare deep learning models across different application domain.

UNIT I INTRODUCTION TO DEEP LEARNING

Introduction to machine learning - Linear models (SVMs and Perceptron's, logistic regression)- Introduction to Neural Nets: What are a shallow network computes- Training a network: loss functions, back propagation and stochastic gradient descent- Neural networks as universal function approximates.

UNIT II INTRODUCTION TO DEEP LEARNING

History of Deep Learning- A Probabilistic Theory of Deep Learning- Back propagation and regularization, batch normalization- VC Dimension and Neural Nets-Deep Vs Shallow Networks Convolutional Networks- Generative Adversarial Networks (GAN), Semi-supervised Learning.

UNIT III DIMENTIONALITY REDUCTION

Linear (PCA, LDA) and manifolds, metric learning - Auto encoders and dimensionality reduction in networks - Introduction to Convnet - Architectures – AlexNet, VGG, Inception, ResNet - Training a Convnet: weights initialization, batch normalization, hyper parameter optimization.

UNIT IV OPTIMIZATION AND GENERALIZATION IN DEEP LEARNING

Optimization in deep learning– Non-convex optimization for deep networks- Stochastic Optimization Generalization in neural networks- Spatial Transformer Networks- Recurrent networks, LSTM Recurrent Neural Network Language Models- Word-Level RNNs & Deep Reinforcement Learning - Computational & Artificial Neuroscience.

UNIT V APPLICATIONS OF DEEP LEARNING

Image-net- Detection-Audio WaveNet-Natural Language Processing Word2Vec - Joint Detection Bio-Informatics- Face Recognition- Scene Understanding- Gathering Image Captions.

Textbooks:

1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.
2. Deng & Yu, Deep Learning: Methods and Applications, Now Publishers, 2013.
3. Michael Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.

DevOps ENGINEERING			
Course Code:	**CS526**	**Course Credits:**	**3**
Course Category:	**CC**	**Course (U / P)**	**IP/P**
Course Year (U / P):	**4 IP / 1P**	**Course Semester (U / P):**	**8IP / 2P**
No. of Lectures + Tutorials (Hrs/Week):	**03**	**Mid Sem. Exam Hours:**	**1.5**
Total No. of Lectures (L + T):	**45**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES
The main objectives of this course are to:
1. Describe the agile relationship between development and IT operations.
2. Understand the skill sets and high-functioning teams involved in DevOps and related methods to reach a continuous delivery capability.
3. Implement automated system update and DevOps lifecycle.
4. Discussion on DevOps Systems
5. Evaluation of Sytem design lifecycle in DevOps Engineeering
COURSE OUTCOMES
On successful completion of this course, students will be able to:
1. Identify components of Devops environment.
2. Describe Software development models and architectures of DevOps.
3. Apply different project management, integration, testing and code deployment tool.
4. Investigate different DevOps Software development models.
5. Assess various Devops practices.
6. Collaborate and adopt Devops in real-time projects.

UNIT-I INTRODUCTION

Introduction, Agile development model, DevOps, and ITIL. DevOps process and Continuous Delivery, Release management, Scrum, Kanban, delivery pipeline, bottlenecks, examples.

UNIT – II SOFTWARE DEVELOPMENT MODELS AND DEV OPS

DevOps Lifecycle ,DevOps, and Continuous Testing. DevOps influence on Architecture: Introducing software architecture , Architecture rules of thumb, The separation of concerns, Handling database migrations, Microservices, and the data tier, DevOps, architecture .

UNIT – III INTRODUCTION TO PROJECT MANAGEMENT

The history of source ,code management, Roles and code, source code management system and migrations, Shared authentication, Hosted Git servers, Different Git server , Docker intermission, The pull request model.

UNIT – IV INTEGRATING THE SYSTEM

Build systems, Managing build dependencies, Jenkins plug, and file system layout, The host server, Build slaves, Software on the host, Triggers, Job changing.

UNIT–V MONITORING & CLOUD IN DEVOPS

Monitoring Tools ,Performance Monitoring ,Cloud Computing , AWS Basics , DevOps in Cloud , Handling data base ,CI pipelines, Automated testing .

Textbooks:

1. The DevOps Handbook – Gene Kim, Jez Humble, Patrick Debois, John Willis. Publisher: IT Revolution Press, Year: 2016 (first edition).
2. Continuous Delivery – Reliable Software Releases through Build, Test and Deployment Automation – Jez Humble, David Farley .Publisher: Addison-Wesley (or Addison Wesley Professional), Year: 2010.

BLOCKCHAIN TECHNOLOGY			
Course Code:	**CS528**	**Course Credits:**	**4**
Course Category:	**CC**	**Course (U / P)**	**IP / P**
Course Year (U / P):	**4IP / 1P**	**Course Semester (U / P):**	**8IP / 2P**
No. of Lectures + Tutorials (Hrs/Week):	**3 +1**	**Mid Sem. Exam Hours:**	**1.5**
Total No. of Lectures (L + T):	**45+15**	**End Sem. Exam Hours:**	**3**
COURSE OBJECTIVES			
1. To get acquainted with the concept of Distributed ledger system and Blockchain.			
2. To learn the concepts of consensus and mining in Blockchain through the Bitcoin network.			
3. To understand Ethereum and develop-deploy smart contracts using different tools and frameworks.			
4. understand permissioned Blockchain and explore Hyperledger Fabric.			
5. To understand different types of crypto assets.			
COURSE OUTCOME			
At the end of the course the students should be able to:			
1. Describe the basic and fundamental concept of Blockchain and Distributed Ledger Technology.			
2. Interpret the knowledge of the Bitcoin network, nodes, keys, wallets and transactions			
3. Implement smart contracts in Ethereum using different development frameworks.			
4. Develop simple applications using Solidity language on Ethereum platform.			
5. Interpret different Crypto assets and Crypto currencies			

UNIT I FUNDAMENTALS OF BLOCKCHAIN TECHNOLOGY

Blockchain – Definition, architecture, elements of blockchain, benefits and limitations, types of blockchain. Consensus – definition, types, consensus in blockchain.

Decentralization – Decentralization using blockchain, Methods of decentralization, Routes to decentralization, Blockchain and full ecosystem decentralization.

UNIT II FUNDAMENTALS OF CRYPTOGRAPHY

Introduction to Cryptography, Symmetric cryptography – AES. Asymmetric cryptography – RSA. Elliptic curve cryptography, Digital signatures – RSA digital signature algorithms. Secure Hash Algorithms – SHA-256. Applications of cryptographic hash functions – Merkle trees, Distributed hash tables.

UNIT III CONSENSUS ALGORITHMS AND BITCOIN

Consensus Algorithms, Crash fault-tolerance (CFT) algorithms – Paxos, Raft. Byzantine fault-tolerance (BFT) algorithms – Practical Byzantine Fault Tolerance (PBFT), Proof of work (PoW), Proof of stake (PoS), Types of PoS. Bitcoin – Definition, Cryptographic keys – Private keys, public keys, addresses. Transactions – Lifecycle, coinbase transactions, transaction validation. Blockchain – The genesis block. Mining – Tasks of miners, mining algorithm, hash rate. Wallets – Types of wallets.

UNIT IV SMART CONTRACTS AND USE CASES

Smart Contracts – Definition, Smart contract templates, Oracles, Types of oracles, Deploying smart contracts. Decentralization terminology – Decentralized applications, Decentralized Autonomous Organizations. Use cases of Blockchain technology – Government, Health care, Finance, Supply chain management. Blockchain and allied technologies – Blockchain and Cloud Computing, Blockchain and Artificial Intelligence.

UNIT V ETHEREUM AND SOLIDITY

Ethereum – The Ethereum network. Components of the Ethereum ecosystem – Keys and addresses, Accounts, Transactions and messages. The Ethereum Virtual Machine, Blocks and blockchain. The Solidity language – The layout of a Solidity source code, Structure of a smart contract, variables, data

types, control structures, events, inheritance, libraries, functions, error handling. Smart contracts Case study: Voting, Auction.

Text Book

1. Imran Bashir, Mastering Blockchain: A deep dive into distributed ledgers, consensus protocols, smart contracts, DApps, cryptocurrencies, Ethereum, and more, Packt Publishing, Third edition, 2020.
Mastering Bitcoin Unlocking Digital Cryptocurrencies, Andreas M. Antonopoulos, O'Reilly Media

Reference Books:

1. Ritesh Modi, Solidity Programming Essentials: A beginner's guide to build smart contracts for Ethereum and blockchain, Packt Publishing, First edition, 2018.

2. Kumar Saurabh, Ashutosh Saxena, Blockchain Technology: Concepts and Applications, First Edition, Wiley Publications, First edition, 2020.

3. Chandramouli Subramanian, Asha A George, et al, Blockchain Technology, Universities Press (India) Pvt. Ltd, First edition, August 2020.

4. Blockchain Technology: Concepts and Applications, Kumar Saurabh and Ashutosh Saxena, Wiley.

ADVANCED SOFTWARE ENGINEERING LAB			
Course Code:	**CS582**	**Course Credits:**	**2**
Course Category:	**CC-L**	**Course (U / P)**	**IP/P**
Course Year (U / P):	**4IP/1P**	**Course Semester (U / P):**	**8IP/2P**
No. of Labs (Hrs/Week):	**4Hrs**	**Mid Sem. Exam Hours:**	**NIL**
Total No. of Labs:	**10**	**End Sem. Exam Hours:**	**2**

COURSE OBJECTIVES
1. Student will have successful career in technology or managerial functions.
2. Student will have solid technical and professional foundation to continue higher studies.
3 Students will have skills to develop products, offer services and create new knowledge.
4 Students will have fundamental awareness of Industry processes, tools and technologies.
COURSE OUTCOMES
At the end of the course the students should be able to:
1 Problem analysis:- Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
2 Design/development of solutions:- Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
3 Conduct investigations of complex problems:- Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions..
4 Project management and finance:- Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

LIST OF EXPERIMENTS:

1. Study the GitHub control.
2. Write down the problem statement for a suggested system of relevance
3. Do requirement analysis and develop Software Requirement Specification Sheet (SRS) for suggested system.
4. To perform the function oriented diagram: Data Flow Diagram (DFD) and Structured chart.
5. To perform the user's view analysis for the suggested system: Use case diagram.
6. Problem Analysis and Project Planning -Thorough study of the problem–Identify Project scope, Objectives and Infrastructure.
7. Software Requirement Analysis –Describe the individual Phases/modules of the project and Identify.
8. Data Modelling –Use work products –data dictionary..
9. Software Designing -Develop use case diagrams and activity diagrams, build and test class diagrams, sequence diagrams and add interface to class diagrams..
10. Prototype model –Develop the prototype of the product.

DEEP LEARNING LAB			
Course Code:	**CS584**	**Course Credits:**	**2**
Course Category:	**CC-L**	**Course (U / P)**	**IP / P**
Course Year (U / P):	**4 IP / 1P**	**Course Semester (U / P):**	**8IP / 2P**
No. of Labs(Hrs/Week):	**04**	**Mid Sem. Exam Hours:**	**NIL**
Total No. of Labs	**10**	**End Sem. Exam Hours:**	**2**
COURSE OBJECTIVES			
1. Provide hands-on experience with deep learning frameworks and tools.			
2. Implement and experiment with neural network models for real-world problems			
3. Understand training, optimization, and evaluation of deep learning models.			
4. Apply deep learning techniques in domains such as computer vision, NLP, and time-series analysis.			
5. Encourage research-oriented thinking and reproducible experimentation.			
COURSE OUTCOMES			
At the end of the course the students should be able to understand:			
1. Implement basic to advanced neural network architectures using Python-based frameworks.			
2. Train, tune, and evaluate deep learning models on benchmark datasets.			
3. Apply CNNs, RNNs, and Transformer-based models to domain-specific problems.			
4. Analyze model performance using appropriate metrics and visualization tools			
5. Develop mini-projects and research prototypes using deep learning techniques.			

Software and Tools:

- Programming Language: Python
- Frameworks/Libraries: TensorFlow, Keras, PyTorch
- Supporting Libraries: NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn, OpenCV, NLTK / SpaCy
- Platforms: Jupyter Notebook, Google Colab (optional)

List of Practical:

1. Python & DL Environment Setup: NumPy, Pandas, Matplotlib, TensorFlow/Keras installation, dataset preprocessing.
2. Single Layer Perceptron: Implementation from scratch and logical gate classification.
3. Multi-Layer Perceptron (MLP): Implementation from scratch and logical gate classification.
4. Backpropagation Algorithm: ANN model training and performance visualization.
5. Activation & Loss Functions: Comparison of Sigmoid, ReLU, Tanh, Softmax, MSE, Cross-Entropy.
6. Convolutional Neural Network (CNN): MNIST digit classification.
7. Image Classification using CNN: CIFAR-10 dataset with Dropout and Batch Normalization.
8. Sequence modeling or text classification using RNN.
9. Hyperparameter Tuning: Learning rate and optimizer comparison (SGD, Adam, RMSprop).
10. Recurrent Neural Network (RNN): Sequence prediction implementation.
11. LSTM Implementation: Time series forecasting and future value prediction.
12. Autoencoder: Dimensionality reduction and reconstruction analysis.
13. Transfer Learning: Fine-tuning pre-trained models (ResNet) on a custom dataset.

ELECTIVES

SOFTWARE RELIABILITY ENGINEERING			
Course Code:	**CS530**	**Course Credits:**	**3**
Course Category:	**E1**	**Course (U / P)**	**IP / P**
Course Year (U / P):	**4 IP / 1P**	**Course Semester (U / P):**	**8IP / 2P**
No. of Lectures + Tutorials (Hrs/Week):	**3**	**Mid Sem. Exam Hours:**	**1.5**
Total No. of Lectures (L + T):	**45**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES
1. To provide fundamental concepts of software reliability.
2. To provide knowledge how to develop reliable software.
3. To give knowledge on faults classification and fault tolerance attribute and systems.
4. Discussion on the software and hardware fault tolerance with failure intensity functions.
5. Discussion on concept of N-version programming.
COURSE OUTCOME
At the end of the course the students should be able to:
1. Explain and classify the reliable and non reliable software.
2. Differentiate the error, fault and failure during development of reliable software.
3. Knowledge of fault classification and fault prevention.
4. Classify various fault tolerance software and hardware.
5. Knowledge of N-Version programming with code coverage and coding technique.

UNIT I SOFTWARE RELIABILITY

Measures of software reliability, Mean Time To Failure (MTTF), Mean Time Between Failure (MTBF), Mean Time To Recovery (MTTR), availability, maintainability, Musa's operational profiles and type-1 uncertainty, defect removal and type-2 uncertainty, reliability stability and reliability growth, hardware reliability vs. software reliability, failure probability density function and reliability function, Reliability prediction , reliability metrics.

UNIT II DEVELOPMENT OF RELIABLE SOFTWARE

Reliable software, defect prevention, detection and removal, design for robustness, verification & validation, stabilization of requirements, design, code and test artifacts, active and passive fault detection, fault handling and correction, exceptions, survivability, reliability models, software availability model.

UNIT III FAULT TOLERANCE IN HARDWARE SYSTEMS

Fault classification, fault tolerance attributes and system structure, fault prevention, anticipated and unanticipated fault, test generation for digital systems, combinational logic network, Boolean difference method, test generation for sequential circuits, fault simulation, application of hardware fault tolerance in developing fault tolerant software systems.

UNIT IV SOFTWARE AND HARDWARE FAULT TOLERANCE

Software and hardware faults, failure intensity function, characterization of fault injection, detection and correction, techniques for prediction of remaining faults and fault injection, classification tree analysis, code coverage, coding technique, fault tolerant & self-checking, fail safe circuits, synchronous and asynchronous fail-safe circuits.

UNIT V FAULT TOLERANT SOFTWARE

Concept of N-version programming (NVP) and methods, recovery block, acceptance tests, fault trees, validation of fault tolerant systems, security, fault tolerance in wireless/mobile networks and Internet.

Text Books :

1. Software Reliability Engineering, John D. Musa, Tata McGRAW Hill, 2005.
2. Fault-Tolerant Computer System Design, D.K. Pradhan, 2003.
3. Design and Analysis of Fault-Tolerant Digital Systems, B. W. Johnson, Addison-Wesley, 1989.
4. Fault-Tolerant Computing, Theory & Techniques, D.K. Pradhan, Prentice Hall, 1986.
5. Reliable Computer Systems: Design and Evaluation, D. P. Siewiorek and R. S. Swartz, Digital Press, 1992.
6. Probability and Statistics with Reliability, Queueing and Computer Science application, K.S.Trivedi, Prentice Hall, 1982.
7. Fault Tolerant Principles and Practice, Anderson and Lee, PHI, 1989.

SOFTWARE QUALITY ASSURANCE			
Course Code:	**CS532**	**Course Credits:**	**3**
Course Category:	**E1**	**Course (U / P)**	**IP / P**
Course Year (U / P):	**4 IP / 1P**	**Course Semester (U / P):**	**8IP / 2P**
No. of Lectures + Tutorials (Hrs/Week):	**03**	**Mid Sem. Exam Hours:**	**1.5**
Total No. of Lectures (L + T):	**45**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES
1. Understand basic concepts of software quality and SQA principles.
2. Analyze quality factors and pre-project SQA activities.
3. Apply SQA techniques like reviews, V&V, and testing.
4. Evaluate software quality infrastructure and tools.
5. Assess quality metrics, standards, and management practices.
COURSE OUTCOMES
At the end of the course the students should be able to:
1. Understand basic software quality and SQA principles.
2. Analyze software quality factors and SQA plans.
3. Apply reviews, verification, validation, and testing techniques.
4. Evaluate software quality infrastructure and tools.
5. Assess quality metrics, standards, and management practices.

UNIT I INTRODUCTION TO SOFTWARE QUALITY ASSURANCE

Software Quality Challenge, Uniqueness of Software Quality Assurance, Software Errors, Faults and Failures, Definition of Software Quality, Software Quality Assurance: Definition, Objectives and Scope, Relationship between Software Engineering and SQA, Software Quality Factors: Product Operation Factors, Product Revision Factors, Product Transition Factors, Software Quality Models, Software Quality Assurance vs Quality Control.

UNIT II SQA SYSTEM AND PRE-PROJECT QUALITY COMPONENTS

Components of the Software Quality Assurance System, SQA Architecture and Framework, Pre-Project SQA Components, Contract Review Process, Objectives, Stages and Implementation, Development Plan and Quality Plan, Risk Management in Software Projects, SQA Considerations for Small and Internal Projects, Quality Factor Model.

UNIT III SQA ACTIVITIES IN THE SOFTWARE DEVELOPMENT LIFE CYCLE

Integrating SQA into the Project Life Cycle, Verification, Validation and Qualification, Software Reviews: Formal Design Reviews, Peer Reviews and Inspections, Software Testing Strategies: White Box Testing, Black Box Testing, Test Planning and Test Case Design, Automated Testing, Alpha and Beta Testing Quality Assurance vs Testing.

UNIT IV SOFTWARE QUALITY INFRASTRUCTURE COMPONENTS

Quality Assurance in Software Maintenance, Quality Assurance for External Participants and Outsourcing, CASE Tools and Their Impact on Software Quality, Procedures and Work Instructions, Supporting Quality Devices: Templates and Checklists, Staff Training and Certification, Corrective and Preventive Actions Software Configuration Management, Documentation Control.

UNIT V SOFTWARE QUALITY MANAGEMENT, METRICS AND STANDARDS

Project Progress Control, Software Quality Metrics: Process Metrics, Product Metrics, Cost of Software Quality, Quality Management Standards: ISO 9001 and ISO 9000-3, CMM, CMMI, SPICE, IEEE Software Engineering Standards, Role of Management in SQA, SQA Organization and SQA Units, Future Trends and Challenges in Software Quality Assurance

Text Books:

1. Daniel Galin, Software Quality Assurance: From Theory to Implementation, Pearson Education.
2. Software Quality Assurance by Jeff Tian, Wiley-IEEE Press, 2005.
3. Software Engineering by K. K. Agarwal & Yogesh Singh, New Age International Publishers, 2013.
4. Software Testing: Principles, Techniques and Tools by Yogesh Singh, Oxford University Press, 2008.

SOFTWARE PROJECT MANAGMENT			
Course Code:	**CS534**	**Course Credits:**	**3**
Course Category:	**E1**	**Course (U / P)**	**IP / P**
Course Year (U / P):	**4 IP / 1P**	**Course Semester (U / P):**	**8IP / 2P**
No. of Lectures + Tutorials (Hrs/Week):	**3 + 00**	**Mid Sem. Exam Hours:**	**1.5**
Total No. of Lectures (L + T):	**45 + 00**	**End Sem. Exam Hours:**	**3**

COURSE OBJECTIVES
1. To provide fundamental skills of software Project management emphasizing on challenges.
2. To provide project management concepts through working in a group as team leader.
3. To give knowledge on different techniques of project monitoring, control and review.
4. Discussion on the effect of project management practices in an organization.
5. Discussion on Project monitoring, risk and quality control

COURSE OUTCOME
At the end of the course the students should be able to:
1. Define the principles of project management for developing software.
2. Explain various project management scheduling techniques.
3. Classify various project management tools and estimate the risks involved in project activities.
4. Assess issues related to project quality and staffing.
5. Knowledge of different tools used in Software Project Management.

UNIT I SOFTWARE PROJECT MANAGEMENT, PLANNING AND ESTIMATION

Fundamentals of Software Project Management (SPM), Project Management Cycle, SPM Objectives, SPM Framework, Software Project Planning, Planning Objectives, Project Plan, Types of Project Plan, Structure of a Software Project Management Plan, Software Project Estimation, Estimation Methods, Estimation Models, Decision Process.

UNIT II PROJECT ORGANIZATION AND SCHEDULING

Work Breakdown Structure (WBS), Types of WBS, Functions, Activities and Tasks, Project Life Cycle and Product Life Cycle, Ways to Organize Personnel, Project Schedule, Scheduling Objectives, Building the Project Schedule, Scheduling Terminology and Techniques, Network Diagrams: PERT, CPM, Bar Charts: Milestone Charts, Gantt Charts

UNIT III PROJECT MONITORING AND CONTROL

Dimensions of Project Monitoring & Control, Earned Value Analysis, Earned Value Indicators: Budgeted Cost for Work Scheduled (BCWS), Cost Variance (CV), Schedule Variance (SV), Cost Performance Index (CPI), Schedule Performance Index (SPI), Software Reviews, Types of Review: Inspections, Deskchecks, Walkthroughs, Code Reviews

UNIT IV SOFTWARE CONFIGURATION AND RISK MANAGEMENT

Software Configuration Items and Tasks, Baselines, Plan for Change, Change Control, Change Requests Management, Version Control, Risk Management: Risks and Risk Types, Risk Breakdown Structure (RBS), Risk Management Process: Risk Identification, Risk Analysis, Risk Planning, Risk Monitoring, Cost Benefit Analysis, Software Project Management Tools: CASE Tools, MS-Project

UNIT V PROJECT QUALITY MANAGEMENT

Quality of project, Stages of software quality management, Software Quality Attributes, Software Quality Metrics and Indicators, Process vs Product Quality Management, Tools and techniques for quality control and management.

Text Books:

1. Software Project Management, Bob Hughes and Mike Cotterell, McGraw Hill
2. Information Technology Project Management" Kathy schwalbe, International student edition,THOMSON course Technology, 2003.
3. Microsoft office Project 2003 Bible", Elaine Marmel, Wiley publishing Inc.

Reference Books:

1. Software Project Management A Unified Framework, Walker Royce, Addison-Wesley
2. A practitioner's Guide to Software Engineering, Roger Pressman, Tata McGraw Hill 2014 8th edition.
3. Basics of Software Project Management, NIIT, Prentice-Hall India, Latest Edition. Dan Woods, Open Source for the Enterprise: Managing Risks, Reaping Rewards, O'Reilly, 2005.

OPEN SOURCE SOFTWARE ENGINEERING			
Course Code:	CS536	Course Credits:	3
Course Category:	E1	Course (U / P)	IP / P
Course Year (U / P):	4 IP / 1P	Course Semester (U / P):	8IP / 2P
No. of Lectures+Tutorials (Hrs/Week):	03 + 00	Mid Sem. Exam Hours:	1.5
Total No. of Lectures (L + T):	45 + 00	End Sem. Exam Hours:	3

COURSE OBJECTIVES
1 Knowledge of open source software.
2 A General understanding of open-source technology.
3 Understanding of open-source language.
4 Understanding of open-source software applications and framework.
5 Understanding of open-source in the enterprise.

COURSE OUTCOME
At the end of the course the students should be able to:
1 Basic to advanced knowledge of open-source software.
2 Ability to apply open-source technology.
3 Ability to design, develop, maintain and evaluate open source software application and framework.
4 Ability to apply open source language.
5 Ability to apply open-source software in the enterprise.

UNIT I OPEN SOURCE SOFTWARE

Open Source Software (OSS), history, philosophy and ethics of open source software, open source software development methodology, open source vs. closed source, open source software vs. free software, open source software vs. source available, Windows and Linux, open source development environment, methods and models, standards, open source standards, benefits of open standards, standard setting organizations and processes, project management via open source and open standard.

UNIT II OPEN SOURCE TECHNOLOGY

Open source technology and platform, Operating system: Linux, Berkeley Software Distribution; web server: Apache; communication servers: send mail, jabber, application and messaging server: JBoss, Zope, Zend.

UNIT III OPEN SOURCE LANGUAGES

Ruby, Ruby and object-orientation, data, expressions and flow control, class, object and modules, project and libraries, developing a basic Ruby application, PHP, configure environment, PHP basic, functions, arrays, object-oriented PHP, MYSQL, PostgreSQL.

UNIT IV OPEN SOURCE SOFTWARE APPLICATIONS AND FRAMEWORK

Open source desktop applications, Wiki software, LAMP application, web server and database server application, OSS management tools: taskjuggler, dotProject.net, rapid web application development framework: Ruby on Rail, Model-View-Controller model, Don't Repeat Yourself principle.

UNIT V OPEN SOURCE SOFTWARE ENGINEERING

Nature of open source, leadership in open source software life cycle, designing an open source strategy, open source licenses, comparison of open source licenses, open source empowerment, Requirements in OSS project

Text Books:

1. Paul Kavanagh, Open Source Software: Implementation and Management, Digital Press, 2004.
2. W. Jason Gilmore, Beginning PHP and MySQL, Apress, 2010.
3. Timothy Fisher, Ruby on Rail, Apress, 2009.

Reference Books:

1. Dan Woods, Open Source for the Enterprise: Managing Risks, Reaping Rewards, O'Reilly, 2005.
2. James Lee, Brent Ware,Open Source Web Development with LAMP, Pearson Education, 2008.
3. Steven Weber, The Success of Open Source, HarvardUniversity Press, 2004.
4. Peter Cooper, Beginning Ruby, Apress, 2007.

PARALLEL AND HIGH PERFORMANCE COMPUTING			
ourse Code:	CS538	Course Credits:	3
Course Category:	E1	Course (U/ P)	IP /P
Course Year (U / P):	4 IP / 1P	Course Semester (U / P):	8IP / 2P
No. of Lectures + Tutorials (Hrs/Week):	03 + 00	Mid Sem. Exam Hours:	1.5
Total No. of Lectures (L +T):	45+ 00	End Sem. Exam Hours:	3

COURSE OBJECTIVES
1. Introduce the fundamental concepts of parallel and high-performance computing, including concurrency, parallelism, scalability, and performance limitations.
2. To develop the ability to analyse performance bottlenecks in parallel programs using profiling tools, memory hierarchy concepts, and efficiency metrics.
3. To enable students to identify parallelism in applications and design efficient parallel algorithms using CPU-based optimization and vectorization techniques.
4. To provide hands-on experience in developing and optimizing shared-memory parallel programs using the OpenMP programming model with appropriate scheduling and synchronization strategies.
5. To familiarize students with distributed-memory and accelerator-based computing by implementing parallel programs using MPI and GPU programming models and evaluating their scalability and performance.
COURSE OUTCOMES
At the end of the course the students should be able to:
1. Explain the fundamental concepts of parallel and high-performance computing, including concurrency, scalability, and performance limitations.
2. Analyse performance bottlenecks in parallel programs using profiling techniques, memory hierarchy concepts, and efficiency metrics.
3. Design parallel solutions by identifying parallelism in applications and applying suitable parallel algorithms and CPU-based optimization techniques.
4. Develop and optimize shared-memory parallel programs using OpenMP, applying scheduling, synchronization, and vectorization strategies for improved performance.
5. Implement distributed and accelerator-based parallel programs using MPI and GPU programming models, and evaluate their scalability and performance.

UNIT 1 INTRODUCTION TO PARALLEL AND HIGH PERFORMANCE COMPUTING

Need for parallel computing, real-world applications of parallel and high-performance computing, concurrency vs parallelism, scalability basics, Amdahl's Law and motivation for high-performance computing

UNIT II PERFORMANCE ANALYSIS

Performance bottlenecks in parallel programs, profiling techniques for performance analysis, performance limits and efficiency metrics, data layout and memory hierarchy, cache effects on program performance

UNIT III PARALLEL PROGRAM DESIGN AND CPU PARALLELISM

Planning parallel computing projects, identifying parallel opportunities in programs, parallel algorithms and common computation patterns, vectorization concepts, SIMD architecture and speedup

UNIT IV SHARED MEMORY PROGRAMMING

Loop transformations for performance optimization, compiler support for vectorization, OpenMP programming model, OpenMP performance optimization techniques, scheduling, reduction, synchronization overhead

UNIT V DISTRIBUTED MEMORY PARALLELISM AND GPU COMPUTING

Practical multicore performance tuning, need for MPI and distributed high-performance computing, MPI programming basics, point-to-point communication, collective communication operations, performance considerations in MPI, communication overhead, scalability, GPU architecture and computational advantages, GPU programming model, GPU execution concepts

Text Books:

1. Parallel and High Performance Computing, Robert Robey and Yuliana Zamora, 2021.
2. Introduction to High Performance Computing for Scientists and Engineers, Georg Hager, Gerhard Wellein, 2011.
3. Introduction to parallel computing (2nd edition). Pearson Addison Wesley, 2003.