PES UNIVERSITY

B.TECH IN COMPUTER SCIENCE AND ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES

- Prepare and train students in theoretical foundations to work with cutting edge computing technologies and design solutions to complex engineering problems, making them ready to work in industrial environment.
- Develop all round skills such as team building, inter-personal skills, and leadership qualities in order to effectively communicate with engineering community and with society at large.
- Promote research culture through internships, research assistantships, research-oriented projects, sponsored and collaborative research and enable them to pursue higher studies in computer science and related fields.
- To inculcate social concern meeting the requirements of prospective employers and to develop an ability to innovate efficient computing solutions for a better society.
- Create professionally superior and ethically strong globally competent employees and entrepreneurs.

PROGRAM OUTCOMES

- Apply mathematical and theoretical principles in the modelling and design of high-quality computer-based systems using state-of-the-art computer technology.
- Conduct in-depth study of research literature in the area of Computer Science, analyse problems in order to arrive at substantiated conclusions using first principles of mathematics, and allied sciences.
- Design, implement and evaluate Computer Systems, programs and processes that meet partial/ complete specifications with concern for society, environment and culture.
- Design and conduct experiments, collect data, analyze and interpret the results to investigate complex engineering problems in the field of Computer Science.
- Apply state-of-the-art techniques and modern computer-based tools in prediction, comparison and modelling of complex engineering activities.
- Have a sound understanding of professional, legal, security and social issues and responsibilities in engineering activities involving Computer Science.
- Understand societal and environmental concerns and demonstrate responsibility in sustainable development of computer-based solutions.
- Be aware of ethical and professional responsibilities in engineering situations; make informed judgments regarding intellectual property and rights in relation to computer-based solutions in global, economic, environmental and societal contexts.
- Able to function effectively in teams to establish goals, plan tasks, meet deadlines, manage risk and produce high-quality technical solutions.
- Contribute and communicate effectively with the society, be able to write effective reports and design documents by adhering to appropriate standards, make effective presentations, give and receive clear instructions.
- Apply skills in clear communication, responsible teamwork and time management by, for example, managing a team or project and communicating with external stakeholders.
- Recognize the need for and demonstrate an ability to engage in continuing professional development in its broadest sense.

B. TECH IN COMPUTER SCIENCE AND ENGINEERING

I SEMESTER (2023-27 BATCH)

Sl. No.	Course Code	Course Title	Hours per week				Credits	Tools / Languages	Course Type
			L	T	P	S	C		
1	UE23CS151A	Python for Computational Problem Solving	4	0	2	5	5	Python	FC- Lab Integrated

II SEMESTER (2023-27 BATCH)

Sl. No.	Course Code	Course Title	Hours per week				Credits	Tools / Languages	Course Type
			L	T	P	S	C		
1	UE23CS151B	Problem Solving with C	4	0	2	5	5	C Programming Language, GCC Compiler, GDB Debugger.	FC- Lab Integrated

III SEMESTER (2023-27 BATCH)

Sl. No.	Course Code	Course Title	Hours per week				Credits	Tools / Languages	Course Type
			L	T	P	S	C		
1	UE23CS251A	Digital Design and Computer Organization	4	0	2	5	5	Icarus Verilog Simulator, GTK Wave waveform viewer. (Open–Source Tool)	CC- Lab Integrated
2	UE23CS252A	Data Structures and its Applications	4	0	2	5	5	C Programming Language, Coding Platforms like HackerRank	CC-Lab Integrated
3	UE23CS241A	Mathematics for Computer Science and Engineering	4	0	0	4	4	Jupyter Notebook**,** Python, Pandas, Matplotlib, Scipy, Seaborn, BeautifulSoup, Numpy, Scikit learn.	CC-Independent
4	UE23CS242A	Web Technologies	4	0	0	4	4	HTML, CSS, JavaScript, MERN Technologies.	CC-Independent
5	UE23CS243A	Automata Formal Languages and Logic	4	0	0	4	4	JFLAP	CC-Independent
6	UE23EC221A	CIE L1	2	0	0	2	2		CC-Independent
7	UE24MA221A*	Bridge Course Mathematics –I (Applicable for	2	0	0	2	0		FC-Independent

		the Lateral Entry Students)							
Total			**22/24**	**0**	**4**	**24/26**	**24**		
* - Audit Course									

IV SEMESTER (2023-27 BATCH)

Sl. No.	Course Code	Course Title	Hours per week				Credits	Tools / Languages	Course Type
			L	T	P	S	C		
1	UE23CS251B	Microprocessor and Computer Architecture%	4	0	2	5	5	ARM Simulator, Arduino Microcontroller kit, ParaCache simulator.	CC-Lab Integrated
2	UE23CS252B	Computer Networks	4	0	2	5	5	Wireshark, Python	CC-Lab Integrated
3	UE23CS241B	Design and Analysis of Algorithms	4	0	0	4	4	C Programming Language, GCC Compiler.	CC-Independent
4	UE23CS242B	Operating Systems@	4	0	0	4	4	C, Linux/Unix OS for system call implementation.	CC-Independent
5	UE23CS243B	Vector Spaces and Linear Algebra	4	0	0	4	4	Python IDE	CC-Independent
6	UE23EC221B	CIE L2	2	0	0	2	2		CC-Independent
7	UE24MA221B *	Bridge Course Mathematics –II (Applicable to Lateral Entry Students)	2	0	0	2	0		FC-Independent
Total			**22/24**	**0**	**4**	**24/26**	**24**		
Note: Desirable Knowledge - %UE23CS251A, @UE23CS252A. * - Audit Course									

UE23CS151A: Python for Computational Problem Solving (4-0-2-5-5)

Python is an easy to learn, general-purpose, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Course Objectives:

- Learn the syntax and semantics of Python programming language.
- Illustrate the process of structuring the data using lists, tuples, sets and dictionaries.
- Demonstrate the use of built-in functions to navigate the file system.
- Learn various paradigms of programming and implement the Object Oriented Programming concepts in Python.

Course Outcomes:

At the end of this course students will be able to,

- Program effectively using the Python language.
- Identify the methods to create and manipulate lists, tuples and dictionaries.
- Discover commonly used operations involving file system.
- Think using different paradigms of programming and interpret the concepts of Object-Oriented Programming as used in Python.

Course Contents:

Unit 1: Introduction

Computation Problem Solving-Limits of Computational Problem Solving - Computer Algorithm - Computer Hardware. Digital Computer - Operating System- Limits of IC technology - Computer Software - Syntax, semantics and program translation ,Introduction to Python Programming Language, IDLE Python Development Environment, Output function - variables, types and id, input function , operators and expressions, Control structures.

12 Hours

Unit 2: Collections & Functions

Lists, Tuples, Dictionaries, Sets, Strings and text file manipulation: reading and writing files. Functions: Definition, call Positional and keyword parameter, Default parameters, Variable number of arguments.

16 Hours

Unit 3: Functions, GUI, Modules, Testing and Debugging

Recursion, Call-backs, Closure, Decorators, generators. Graphical User Interface with Tinkter package- Different geometric methods – Tk, mainloop, Creating simple GUI - buttons, canvas, check button, labels, entry fields, dialogs Widgets - sizes, fonts, colours layouts, nested frames, Modules - import mechanisms Testing- Pytest , Function testing with Doctest, pdb debugger commands.

14 Hours

Unit 4: Functional & Object Oriented Programming

Lambda function, Map, filter, and reduce, max, min, Zip, list comprehension. Classes and objects - inheritance, polymorphism, iterators, Error handling & Exceptions - try, except and raise, exception propagation.

14 Hours

Lab / Hands-on:

1: Programs on Input Output Functions, Operators and Expressions, Usage of Libraries and Control Structures.
2: Programs on Collections (Lists, Tuples, Sets , Dictionaries , Strings).
3: Programs on Files and File manipulations.
4: Programs on Functions.
5: Programs on Functional Programming.
6: Programs on Object Oriented Programming.

Tools / Languages: Python.

Text Book(s):

1: “Introduction to Computer Science Using Python: A Computational Problem- Focus”, Charles Dierbach,Wiley India Edition,John Wiley, 2015.

Reference Book(s):

1: “Learn python Programming”, Fabrizio Romano, 2ndEdition, Packet Publishing,2018.
2: “Fundamentals of Python: First Programs”, Kenneth A. Lambert, Cengage, 2019.
3: “Introduction to Computation and Programming Using Python: With Application to Understanding Data”,
4 John V. Guttag, MIT Press, MIT with Library of Congress Cataloguing- in-PublicationData, 2016.

UE23CS151B – Problem Solving with C (4-0-2-5-5)

In Problem Solving with C Course, will be learning to solve common types of computational problems, by analyzing the given problem statement and developing an algorithm to solve the given problem. Then make use of c language constructs to develop well-structured programs.

Course objectives:

- Acquire knowledge on how to solve relevant and logical problems using computing machine
- Map algorithmic solutions to relevant features of C programming language constructs
- Gain knowledge about C constructs and its associated eco-system
- Appreciate and gain knowledge about the issues with C Standards and its respective behaviours
- Get insights about testing and debugging C Programs

Course outcomes:

At the end of the course, the student will be able to

- Understand and apply algorithmic solutions to counting problems using appropriate C Constructs
- Understand, analyse and apply text processing and string manipulation methods using C Arrays, Pointers and functions
- Understand prioritized scheduling and implement the same using C structures
- Understand and apply sorting techniques using advanced C constructs
- Understand and evaluate portable programming techniques using preprocessor directives and conditional compilation of C Programs

Course Content:

Unit 1 : Introduction

Introduction to Programming, Salient Features of 'C', Program Structure, Variables, Data Types & range of values, Operators and Expressions, Control Structures, Input/ Output Functions, Language specifications-Behaviors.

10 Hours

Unit 2 : Counting, Text Processing and String manipulation

Arrays – 1D and 2D, Functions, Storage classes, Pointers, Strings, String Manipulation Functions & errors.

18 Hours

Unit 3 : Dynamic Memory Management, Structures and File Handling: Dynamic Memory Allocation and Deallocation functions & errors, Structures, #pragma, File IO using redirection, File Handling functions, Searching, Sorting, Combination of Structures, Arrays and Pointers, Call-back, Code Review .

18 Hours

Unit 4 : Queuing and Portable Programming

Lists, Priority Queue, Enums, Unions, Bit Fields, Pre-Processor Directives, Conditional Compilation, Code Review

10 Hours

Tools/ Languages: C Programming Language, GCC Compiler, GDB Debugger.

Lab/ Hands-on: 14 Hours

1: Programs on IO, Operators and Control structures.
2: Programs on Arrays and Pointers, Functions using arrays and pointers
3 : Programs on Strings, Structures and Dynamic Memory Management functions
4 : Programs on Structures and Array of structures
5 : Introduction to GDB and Debugging using GDB.
6 : Programs on Inclusion of files using redirection operator.
7 : Programs using File handling functions in C - Sorting and Searching using Array of pointers to structures.
8 : Implementation of Ordered List.
9 : Implementation of Priority based scheduling.
10 : Programs on unions, enums and Preprocessor directives.

Text Book(s) :

1:“The C Programming Language”, Brian Kernighan and Dennis Ritchie, Prentice Hall PTR, 2nd Edition, 1988.

Reference Book(s):
1: “How To Solve It By Computer”, R G Dromey, Pearson, 2011.
2: “C Programming: A Modern Approach”, 2nd Edition by K.N. King. W. W. Norton & Company.
3: “Learn C the Hard Way”: Zed Shaw’s Hard Way Series, 1st Edition.
4: “C Puzzle Book” by Alan R. Fever, Pearson Education.
5: “Expert C Programming: Deep C Secrets” by Peter Van Der Linden, Pearson Education.

UE23CS251A : Digital Design and Computer Organisation(4-0-2-5-5)

This course focuses on the structure, design and operation of a computer system at different levels of abstraction. The digital design part of the course describes low level digital logic building blocks while the computer organization part explains the structure and operation of microprocessors.

Course Objectives:

- Fundamental (combinational and sequential) building blocks of digital logic circuits.
- Design of more complex logic circuits such as adders, multipliers and register files.
- Design of Finite State Machines based on problem specification.
- Construction, using above logic circuits, of a microprocessor and its functioning at the clock cycle level.

Course Outcomes:

At the end of this course, the student will be able to,

- Perform analysis of given synchronous digital logic circuit.
- Design and implement small to medium scale data path logic circuits from given specification.
- Design and implement control logic using Finite State Machines.
- Understand hardware level microprocessor operation, providing a foundation for the higher layers and utilize the concepts and techniques learnt to implement complex digital systems.

Course Content:

Unit 1: Gate-Level Minimization and Combinational logic-1

Introduction, The map method Four variable K-map, Product of Sums simplification, Don't Care conditions, NAND and NOR implementation, Combinational circuits, Analysis procedure Design Procedure, Combinational logic-1: Binary Combinational logic :Adder- Subtractor, Decimal Adder, Binary multiplier, Magnitude comparator Decoders Encoders, Multiplexers.

Unit 2: Synchronous Sequential Logic-I

Synchronous Sequential Logic: Introduction, Sequential circuits, Storage elements: Latches, Flip flops, Analysis of clocked sequential circuits, State reduction and assignment, Design procedure Registers and counters: Registers, Shift register, Ripple counters, Synchronous counters Other counters.

14 Hours

Unit 3: Basic structure of computers, Standard I/O interface, Interrupts

Computer Types, Functional Units: Input Unit, Memory Unit, ALU, Output Unit, Control Unit, Basic operational concepts ,Number representation and arithmetic Operations, Character representation, Memory locations and addresses, Memory Operations ,Instruction and instruction sequencing ,Addressing modes, Assembly Languages , I/O Operations: Accessing I/O Devices, Interrupts Standard I/O Interfaces. **14 Hours**

Unit 4: Arithmetic Processing Unit and Control Unit Design

Arithmetic: Multiplication of Positive numbers, Signed operand Multiplication, Fast multiplication, Integer division, floating point numbers operation and architecture ,Some fundamental concepts, Execution of a complete instruction, Multiple Bus Organization, Hardwired control ,Single-cycle, Multi-cycle processor data path and control.

14

Hours

Tools / Languages: Icarus Verilog Simulator, GTK Wave waveform viewer. (Open–Source Tool)

Text Book(s):

1: "Digital Design", M Morris Mano, Michael D Ciletti, Pearson, 5th Edition, 2012.

2: "Computer Organization", Carl Hamacher, Zvonko Vranesic, Safwat Zaky, McGraw Hill, 5th Edition,2002.

Reference Book(s):

1: Digital Design & Computer Architecture, David Money Harris, Sarah L. Harris, 2nd Edition, Elsevier, 2013.

2: Computer Organization and Design, David A. Patterson, John L. Hennessey 5th Edition, Elsevier.

Lab/Hands-on: **14 Hours**

1: Verilog Basics- Basic gates, Adder/ Subtractor (One bit & n-bit).

2: Design of an ALU.

3: Design of a Register File.

4: Design of a Data path (Integration of ALU and Register File).

5: Design of a Program Counter.

6: Design of Control Logic.

UE23CS252B: Data Structures and its Applications (4-0-2-5-5)

This course introduces abstract concepts, shows how the concepts are useful for problem solving and then shows how the abstractions can be made concrete by using a programming language. Equal emphasis is placed on both the abstract and the concrete versions of a concept so that the student learns about the concept itself, its implementation and its application.

Course Objectives:

- Basic approaches and mindsets for analyzing and designing data structures and construct essential skills of data structures to store and retrieve data quickly and usefully (efficiently).
- Usage of different data structures that support different set of operations which are suitable for different type of tasks.
- Implement how to insert, delete, search and modify data in any given data structures- Stack, Queue, List, Tree, heap, Graphs.
- Implement a given application using the available data structure.

Course Outcomes:

At the end of this course, the student will be able to,

- Choose relevant data structures for any given application
- Apply the required to implement any data structure.
- Appropriate data structure in competitive programming.
- Design and develop efficient software systems with good knowledge of data structures.

Course Content:

Unit 1: Linked List and Stacks

Review of C , Static and Dynamic Memory Allocation. Linked List: Doubly Linked List, Circular Linked List – Single and Double, Multilist: Introduction to sparse matrix (structure). Skip list Case study: Dictionary implementation using skip list Stacks: Basic structure of a Stack, Implementation of a Stack using Arrays & Linked list. Applications of Stack: Function execution, Nested functions, Recursion: Tower of Hanoi. Conversion & Evaluation of an expression: Infix to postfix, Infix to prefix, Evaluation of an Expression, Matching of Parenthesis. **14 Hours**

Unit 2: Queues and Trees

Queues & Dequeue: Basic Structure of a Simple Queue, Circular Queue, Priority Queue, Dequeue and its implementation using Arrays and Linked List. Applications of Queue: Case Study – Josephus problem, CPU scheduling- Implementation using queue (simple /circular). General: N-ary trees, Binary Trees, Binary Search Trees (BST) and Forest: definition, properties, conversion of an N-ary tree and a Forest to a binary tree. Traversal of trees: Preorder, Inorder and Postorder. **14 Hours**

Unit 3: Application of Trees and Introduction to Graphs

Implementation of BST using arrays and dynamic allocation: Insertion and deletion operations, Implementation of binary expression tree., Threaded binary search tree and its implementation. Heap: Implementation using arrays. Implementation of Priority Queue using heap - min and max heap. Applications of Trees and Heaps: Implementation of a dictionary / decision tree (Words with their meanings). Balanced Trees: definition, AVL Trees, Rotation, Splay Tree, Graphs: Introduction, Properties, Representation of graphs: Adjacency matrix, Adjacency list. Implementation of graphs using adjacency matrix and lists. Graph traversal methods: Depth first search, Breadth first search techniques. Application: Graph representation: Representation of computer network topology.**14 Hours**

Unit 4: Applications of Graphs , B-Trees, Suffix Tree and Hashing

Application of BFS and DFS: Connectivity of graph, finding path in a network. Suffix Trees: Definition, Introduction of Trie Trees, Suffix trees. Implementations of TRIE trees, insert, delete and search operations. Hashing: Simple mapping / Hashing: hash function, hash table, Collision Handling: Separate Chaining & Open Addressing, Double Hashing, and Rehashing. Applications: URLs decoding, Word prediction using TRIE trees / Suffix Trees.

14 Hours

Tool/ Languages: C Programming Languag

Lab / Hands-on: **14 Hours**

Implementation of:

1: Linked List and advanced operations.

2: Stack and applications based on it.

3: Queue and applications based on it.

4: Binary Tree, Binary Search Tree and applications based on it.

5: Graph Data structure and applications based on it.

6: Hashing Techniques.

Text Book(s):

1: "Data Structures using C / C++", Langsum Yedidyah, Moshe J Augenstein, Aaron M Tenenbaum Pearson Education Inc, 2nd edition, 2015.

Reference Book(s):

1: "Data Structures and Program Design in C", Robert Kruse, Bruce Leung, C.L Tondo, Shashi Mogalla, Pearson, 2nd Edition, 2019.

UE23xx: Mathematics for Computer Science and Engineering (4-0-0-4- 4)

This course covers both Descriptive statistics to understand the data and the inferential statistics which seeks to infer something about a population on the basis of a statistical sample and build a simple linear regression model. This course will also cover engineering optimization techniques like unconstrained, constrained and discrete optimization techniques.

Course Objectives:

- Provide insights about the basic roles of a Data Scientist. Develop a greater understanding of the importance of Data Visualization techniques and Random Variables and their Distributions.
- Provide students with knowledge of Confidence Intervals and their importance. Make inferences about the population parameters using sample data and test it to draw meaningful conclusions.
- Provide an understanding on the importance and techniques of predicting a relationship between the two sets of data and determine the goodness of fit model.
- Provide an understanding on the importance and techniques of engineering optimization.

Course Outcomes:

At the end of this course, the student will be able to,

- Use Python and other tools to extract, clean and analyze data from several data sources (files, web) analyze an extremely large dataset and perform exploratory data analysis to extract meaningful insights.
- Analyze a real-world problem and solve the same with the knowledge gained from various distribution studies.
- Develop and test a hypothesis about the population parameters to draw meaningful conclusions and fit a regression model to data and use it for prediction.
- Analyze a real-world problem and solve the same with the knowledge gained from various engineering optimization techniques.

Course Content

Unit 1: Applications of Probability Distributions and Principles of Point Estimation

Introduction, Motivating Examples and Scope. Statistics: Introduction, Types of Statistics, Types of Data, Types of Experiments – Controlled and Observational study, Sampling: Sampling Methods, Sampling Errors, Case Study. Chebyshev's inequality, Normal Probability Plots, Introduction to Generation of Random Variates and mention the types, Acceptance-Rejection method, Sampling Distribution, The Central Limit Theorem and Applications, Principles of Point Estimation - Mean Squared Error for Bernoulli, Binomial, Poisson, Normal, Maximum Likelihood Estimate for Bernoulli, Binomial, Poisson, Normal and Case Study. introduction to multivariate normal distribution, MAP distribution. **16 Hours**

Unit 2: Confidence Intervals and Hypothesis Testing

Confidence Intervals: Interval Estimates for Mean of Large and Small Samples, Student's t Distribution, Interval Estimates for Proportion of Large and Small Samples, Confidence Intervals for the Difference between Two Means, Interval Estimates for Paired Data. Factors affecting Margin of Error, Hypothesis Testing for Population Mean and Population Proportion of Large and Small Samples, Drawing conclusions from the results of Hypothesis tests, Case Study. **12 Hours**

Unit 3: Distribution Free Tests and Multiple Linear Regression

Distribution Free Tests, Chi-squared Test, Fixed Level Testing, Type I and Type II Errors, Power of a Test, Factors Affecting Power of a Test. Simple Linear Regression: Introduction, Correlation, the Least Square Lines, Predictions using regression models - Uncertainties in Regression Coefficients, Checking Assumptions and transforming data, Introduction to the Multiple Regression Model, Case Study. **14 Hours**

Unit 4 Engineering optimization Introduction to Optimization-Based Design, Modelling Concepts, Unconstrained Optimization, Discrete Variable Optimization, Genetic and Evolutionary Optimization, Constrained Optimization.

14 Hours

Tools / Languages/Libraries: Jupyter Notebook**,** Python, Pandas, Matplotlib, Scipy, Seaborn, BeautifulSoup, Numpy, Scikit learn.

Text Book(s):

1. “Statistics for Engineers and Scientists”, William Navidi, McGraw Hill Education, India, 4th Edition, 2015.

2. “Optimization Methods for Engineering Design, Parkinson, A.R., Balling, R., and J.D. Hedengren, Second Edition, Brigham Young University, 2018.

Reference Book(s):

1. “The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modelling”, Raj Jain, Wiley, 2008.

2. “Sampling- Design and Analysis”, Sharon L. Lohr, 2nd edition (stats), Cengage, 2010.

3. “Data Science from Scratch”, Joel Grus, O’Reilly, 1st Edition, 2015.

4. “Statistics for Engineers and Scientists”, William Navidi, 3rd Edition, McGraw Hill, 2010.

UE23CS242A: Web Technologies (4-0-0-4-4)

Web Technologies course demonstrates an in-depth understanding of the technologies necessary for designing and developing a rich web application in an efficient way.

Course Objectives:

- Basic web technologies and building blocks of a website using HTML, CSS, JavaScript and Advanced JavaScript
- The core concepts of HTML5, JQuery and AJAX, MERN (MongoDB, ExpressJS, ReactJS and NodeJS) stack and build an UI of the application using React JS
- Building a multi-tier application by interfacing UI to NodeJS
- Integrate database MongoDB through Express JS Framework and Web services.

Course Outcomes:

At the end of the course, the student will be able to,

- Understand basic web technologies like HTML, CSS and JavaScript
- Achieve rich user experience by implementing HTML5 features and Asynchronous communication using AJAX, JQuery and MERN stack layers (MongoDB, ExpressJS, ReactJS and NodeJS) and Create rich User Interface using React JS
- Understand and Integrate the UI with NodeJS
- Create RESTful Web services using ExpressJS and MongoDB database

Course Content:

Unit 1: HTML, CSS and Client Side Scripting

Introduction to Web Architecture and Web protocols (HTTP Request Response Formats, URLs), Basic Mark-ups & syntax, HTML elements & attributes, Web Form, HTML5 (New Tags, Inputs, Elements and Controls), CSS3.0 - Styles and Style sheets, Selectors, Style properties, Box Model, JavaScript Basics(variables, scope, Builtin Objects), JavaScript objects, DOM Manipulations, Events and Event Handling in JavaScript

14 Hours

Unit 2: HTML5, JQuery and Ajax

HTML5 (APIs), JQuery Introduction, Callbacks and Promises, Introduction to Single Page Application, XML Vs JSON, Asynchronous Communication using AJAX and fetch API. **ReactJS** – MERN Introduction, React Classes and Components, JSX, Rendering of elements **14 Hours**

Unit 3: ReactJS

Properties, State, Context, Component lifecycle methods, Refs & Keys, Event Handling, Stateless components, React Forms, React Hook **NodeJS –** Understanding Node JS Architecture, Set up Node JS app, Node Modules, call-backs, File system Module, HTTP Module, Handling HTTP Requests

14 Hours

Unit 4: MongoDB

MongoDB-Documents, Collections, Reading and Writing to MongoDB, MongoDB NodeJS Driver, Running a react application on NodeJS, React Router. **ExpressJS –** Introduction to Web services and REST API's , Express Framework Overview, Routing and URL building, Error Handling, Express Middleware, Form Data and File Upload.

14 Hours

Tools / Languages: HTML, CSS, JavaScript, MERN Technologies.

Text Book(s) :

1: "Learning PHP, MySQL & JavaScript", Robin Nixon., 5th edition, O'Reilly Media, Inc. ISBN: 9781491978917, 2018.

2: "Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node", Vasan Subramanian,Apress, 2017.

Reference Book(s) :

1: “Beginning Node.js, Express & MongoDB Development”, Greg Lim, July 2019.

2:“Learning React, Functional Web Development with React and Redux”, Alex Banks and Eve Porcello, O’Reilly Media, May 2017.

Reference Link(s);
1: https://reactjs.org/docs/.
2:https://www.kirupa.com/react.

UE23CS243A: Automata Formal Languages and Logic (4-0-0-4-4)

The course introduces fundamental concepts in Automata and Formal Languages and their application to Logic. Thecourse covers the notions of Finite State Automaton, Regular expression, Push down Automaton, Context FreeLanguages and Turing Machines. These abstract and formal models and their usage in Propositional and First Order Predicate Logic, allow for solving problems in Formal language Generation and Recognition.

Course Objectives:

- Teach students to construct basic machines like DFA, NFA which represent Regular Languages, Regular Expressions, and Regular Grammars and to identify Non – Regular Languages.
- To familiarize students to construct Teach students to identify Context Free Languages, to construct Push down Automata which represent Context Free Languages, to convert the given grammar to various normal forms and to make use of Membership Algorithm.
- Teach students to understand closure properties of Context Free Languages, to identify Non – Context Free Languages and to construct Turing Machines and
- To familiarize students with concepts like Recursively Enumerable languages, Recursive Languages, Undecidable Problems. And to familiarize notions of mathematical logic: logical notations (syntax) and how toassign meaning to them (semantics).

Course Outcomes:

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

- Design simple machines like DFA, NFA, convert NFA to DFA and minimize a given DFA, construct regular expressions for different languages, verify that some languages are regular and some are not.
- Analyze the difference between Regular Languages and Context Free Languages, design Push Down automata,construct Context Free Grammars, and convert one form of the grammar to another form.
- Enumerate the properties of Context Free Grammars, verify that some languages are context free and some arenot, design Turing Machines, and analyze the difference between acceptability and decidability and
- Analyze the difference between Recursive and Recursively Enumerable Languages, Decidable Languages,Turing – Recognizable and Co – Turing – Recognizable, some problems that cannot be solved by Turing Machines, reduce one Undesirables Problem to another, Undeniable Problems for Recursively Enumerable Languages. And make use of Propositional Logic and Predicate Logic in knowledge representation and truth verification.

Course Content:

Unit 1: Introduction

Mathematical Preliminaries and Notation, Three Basic Concepts. Finite Automata: Deterministic Finite Accepters, Non-Deterministic Finite Accepters, Equivalence of Deterministic and Non-Deterministic Finite Accepters,Reduction of the number of states in Finite Automata. Regular Expressions, Connection between Regular Expressions and Regular Languages Regular Grammars **14 Hou**

Unit 2: Regular Languages and Context Free Languages

Properties of Regular Languages: Closure Properties of Regular Languages, Elementary Questions about Regular Languages, Identifying Non Regular Languages. Definitions of PDA and CFL, Deterministic Pushdown Automata,Non-Deterministic Pushdown Automata, Pushdown Automata and Context Free Languages, Context Free Grammars.**14 Hours**

Unit 3: Properties of Context Free Languages and Turing Machine

Parsing and Ambiguity. Simplification of Context–Free Grammars and Normal Forms: Methods for Transforming Grammars, Two Important Normal Forms, A Membership algorithm for Context Free Grammar. Properties of Context-Free Languages: Closure Properties and Questions about Context–Free Languages, Pumping Lemma forContext–Free Languages. Turing Machines: The Standard Turing Machine, Constructing Turing Machines, Combining Turing Machines for Complicated Tasks, Turing’s Thesis, Rice theorem. **14 Hours**

Unit 4: Undecidability and design of a declarative language.

Hierarchy of Formal Languages and Automata: Recursive and Recursively Enumerable Languages, the ChomskyHierarchy. Limits of Algorithmic Computation: Some Problems that cannot be solved by Turing Machines,Undecidable Problem for Recursively Enumerable Languages, idea of reduction. A very simple Logic, Syntax, Semantics, A simple knowledge Base, A simple inference procedure. PropositionalTheorem Proving: Inference and Proofs, Proof by Resolution, Conjunctive Normal Form, A resolution algorithm.Syntax and Semantics of First Order Logic: Models for First Order Logic Symbols and interpretations, Terms,Atomic Sentences, Complex Sentences Quantifiers, Equality, Numbers, sets and Lists. Example - The electronic circuits' domains

14 Hours

Tools / Languages: JFLAP.

Text Book(s):

1: "An Introduction to Formal Languages and Automata", Peter Linz, Jones and Bartlett, New Delhi, India, 6th Edition, 2016.

2: "Artificial Intelligence – A Modern Approach", Stuart Russell and Peter Norvig, Pearson, 3rd Edition (Paperback), 2016.

Reference Book(s):

1: "Theory of Computation", Michael Sipser, Cengage Learning, New Delhi, India, 2008.

2: "Introduction to Automata Theory, Languages, and Computation", John E Hopcroft, Rajeev Motwani, Jeffrey D Ullman, Pearson Education, New Delhi, India, 3rd Edition, 2009.

3: "Theory of Computation: A Problem–Solving Approach", Kavi Mahesh, Wiley India, New Delhi, 2012.

UE23CS251B: Microprocessor and Computer Architecture (4-0-2-5-5)

This course will give you an in-depth understanding of the inner-workings of modern digital computer systems and trade-offs present at the hardware-software interface. The course focuses on key topics in microprocessor such as the system architecture, low level programming aspects and interface with other key components. Also, the course will help in understanding the core computer architecture concepts such as multilevel in memory hierarchies, pipelining, and super scalar techniques. A desirable knowledge of Digital Design and Computer organization is required.

Course Objectives:

- Introduce concepts of basic processor architecture and its design.
- Understanding the concept of concepts of pipeline architecture and hazards.
- Study of memory hierarchy, cache memory and its optimizations.
- Introduce advanced concepts in processor architecture like multi-core/ many core processor architectures.

Course Outcomes:

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

- Demonstrate ability to understand the design of different instruction sets like RISC/ CISC and their addressing modes.
- Demonstrate the ability to understand the design of a pipelined processor and its challenges.
- Demonstrate the use of tools to analyse the performance of programs on different architectures. Design alternative memory hierarchy layouts and optimizations.
- Demonstrate and appreciate modern trends in architecture such as multicore architectures.

Desirable Knowledge: UE23CS251A- Digital Design and Computer Organization.

Course Content:

Unit 1 : Architecture

Introduction, ISA Classification - RISC and CISC, Memory Addressing, Operands - Types and Size, Instruction Set - Operations, Control Flow, Instruction Encoding, Case Study - ARM/ MIPS/ x86 Processor. **14 Hours**

Unit 2 : Pipelining

3- Stage Pipelining, 5 - Stage Pipelining, Pipeline Datapath and Control, Data Hazards – Forwarding vs. Stalling, Control Hazards, Branch Prediction Mechanisms and Exceptions, Performance Metrics. **14 Hours**

Unit 3 : Basics of Cache and Cache Optimization

Basics of Caches - Fully Associative, Direct Mapped and Set Associativity, Cache Performance, Basic Cache Optimization- Reduce in Miss Rate. Basic Cache Optimization- Reduce Miss Penalty, Reduce Hit Time. **14 Hours**

Unit 4 : Advances in Architecture

Introduction to Parallel Computing, PC – Applications, Memory architecture, Flynn's taxonomy, parallel programming models, Shared memory programming OpenMP-Introduction, loop-level parallelism, CUDA C Program structure-vector kernel addition, device global memory and Data transfer, Hardware Multi threading, Parallel examples: matrix multiplication, PC-Design Issues, Amdahl's Law, Gustafson Law, Multi-Core Architecture, Introduction to GPU computing

Laboratory :
14 Hours

Course Content:

1. Introduction to ARM Simulator and sample programs
2. Implementation of Data Processing Instructions, Programs on usage of addressing modes
3. Programs on functions and software interrupts, Matrix Operations – Addition and Multiplication
4. Introduction to PARACACHE simulator – Direct Mapping, Associative Mapping
5. Introduction to Plugins interface.
6. Project work

Tools/ Languages: ARM Simulator, Arduino Microcontroller kit, ParaCache simulator.

UE23CS241B - Design and Analysis of Algorithms (4-0-0-4-4)

Algorithms play a key role in science and practice of computing. Learning algorithm design technique is a valuable endeavour from practical standpoint and algorithm design techniques have considerable utility as general problem solving strategies, applicable to problems beyond computing. This course includes classic problems of computer science, application of design techniques and analysis in terms of time and space.

Course Objectives:

- Learn various algorithm design techniques and apply appropriate algorithmic design techniques for specific problems.
- Learn to design and analyze algorithms with an emphasis on resource utilization in terms of time and space
- Learn to trade space for time in algorithmic design using input enhancement and per-structuring.
- Learn the limitations of algorithmic power and techniques to handle these limitations

Course Outcomes:

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

- Identify the design technique used in an algorithm
- Design and implement efficient algorithms for practical and unseen problems and analyze these algorithms using quantitative evaluation.
- Analyse time efficiency over trading space
- Understand the limits of algorithms and the ways to cope with the limitations.

Course Content:

Unit 1: Introduction and Brute Force

Algorithms, Fundamentals of Algorithmic Problem Solving, Important Problem Types. Analysis of Algorithm Efficiency: Analysis Framework, Asymptotic Notations and Basic Efficiency Classes, Mathematical Analysis of Non Recursive and Recursive Algorithms. Brute Force: Selection Sort, Bubble Sort, Sequential Search, Brute Force String Matching, Exhaustive Search. **14 Hours**

Unit 2: Decrease – and – Conquer & Divide-and-Conquer

Decrease-and-Conquer: Decrease by constant number algorithms - Insertion Sort, Topological Sorting, Algorithms for Generating Combinatorial Objects, Decrease-by-a-Constant-Factor Algorithms – Fake Coin Problem, Russian Peasant Multiplication, Josephus problem, Decrease-by-Variable-Size Algorithms – Computing a median and the selection problem. Divide-and-Conquer: Master Theorem, Merge Sort, Quick Sort, Binary Search, Binary Tree Traversals, Complexity analysis for finding the height of BST, Multiplication of Large Integers, Strassen's Matrix Multiplication.

14 Hours

Unit 3: Transform-and-Conquer Space and Time Tradeoffs & Greedy Technique

Transform and Conquer**:** Pre-sorting, Heap Sort, Red-Black Tree Construction and Time complexity Analysis for insert and search operation, 2-3 Trees and B Tree: insertion, deletion, searching, and time complexity analysis. Space and Time Tradeoffs: Sorting by Counting, Input Enhancement in String Matching - Horspool's and Boyer-Moore Algorithms. Greedy Technique: Prim's Algorithm, Kruskal's Algorithm and union-find algorithm, Dijkstra's Algorithm, Huffman Trees **14 Hours**

Unit 4: Limitations, Coping with the Limitations of Algorithm Power & Dynamic Programming,

Dynamic Programming: Computing a Binomial Coefficient, The Knapsack Problem and Memory Functions, Warshall's and Floyd's Algorithms. Limitations of Algorithm Power**:** Lower-Bound Arguments, Decision Trees, P, NP, and NP-Complete, NP-Hard Problems. Coping with the Limitations of Algorithm Power: Backtracking, Branch-and-Bound.

14 Hours

Tools / Languages: C Programming Language, GCC Compiler.

Text Book(s):

1: "Introduction to the Design and Analysis of Algorithms", Anany Levitin, Pearson Education, Delhi (Indian Version), 3rd Edition, 2012.

Reference Book(s):

1: "Introduction to Algorithms", Thomas H Cormen, Charles E Leiserson, Ronald L Rivest and Clifford Stein, Prentice-

Hall India, 3rd Edition, 2009.

2: "Fundamentals of Computer Algorithms", Horowitz, Sahni, Rajasekaran, Universities Press, 2nd Edition, 2007.

3: "Algorithm Design", Jon Kleinberg, Eva Tardos, Pearson Education, 1st Edition, 2006.

UE23CS252B: Computer Networks (4-0-2-5-5)

This is a foundation course on Computer Networking which focuses on building blocks of the Internet. We trace the journey of messages sent over the Internet from an application residing on one host machine (source) to another (Destination) using a top-down layered approach. The course contents are organized based on TCP/IP Protocol stack.

Course Objectives:

- To present a broad overview of computer networking, the Internet and network layered architecture.
- To study the conceptual and implementation aspects of network applications, protocols and socket programming.
- To provide an insight of the Internet's connection–oriented and connectionless end–to–end transport service protocols and TCP’s approach to congestion control, to learn exactly how the network layer can provide its host–to–host communication service.
- To study the IPv4 and IPv6 protocols, to explore several important link–layer concepts and technologies, LAN and Wireless LANs.

Course Outcomes:

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

- Demonstrate in a concise way how the Internet is constructed and functions with respect to TCP/IP or OSI reference models.
- Demonstrate application layer protocols like DNS, HTTP, HTTPS and be able to develop simple client–server applications using socket programming and understand the concept of unreliable data transfer protocols and how UDP implement these concepts.
- Understand the concept of reliable data transfer protocols and how TCP implement these concepts. Implement logical addressing schemes and configure devices using NAT.
- Demonstrate the ability to configure the routers and services such as DHCP, ICMP. Construct and troubleshoot a wired or wireless LAN, and be able to understand wider networking issues.

Course Content:

Unit 1- Computer Networks and the Internet, Application Layer

Introduction to Computer Networks, Internet: A Nuts–and–Bolts Description, A Services Description, Protocol, The Network Edge: Access Networks, Physical Media, Introduction to physical layer devices, The Network Core, Packet Switching, Circuit Switching, A Network of Networks, Delay, Loss, and Throughput in Packet–Switched Networks, Overview of Delay in Packet Switched Networks – Queuing Delay and Packet Loss, End–to–End Delay, Throughput in Computer Networks, The OSI Model and the TCP/IP Protocol Suite, Protocol Layers, The OSI Model, TCP/IP Protocol Suite. Network Application Principles: Network Application Architectures, Processes Communication, Transport Services available to Applications, Transport Services provided by Internet **14 Hours**

Unit 2 - Application Layer, Transport Layer – UDP

The Web, HTTP and HTTPS, Non persistent and persistent connection, HTTP Message Format, User Server Interaction: Cookies, Web Caching. DNS, The Internet’s Directory Service: Services provided by DNS; How

DNS works: DNS Records and messages; Peer to peer Applications; Socket Programming with TCP and UDP; Other Application Layer Protocols: FTP, SMTP, SNMP, Telnet, SSH.

Introduction to Transport Layer Services: Relationship Between Transport and Network Layer, Overview of the Transport layer in the Internet, Multiplexing and Demultiplexing; Connectionless Transport UDP: UDP Segment Structure, UDP Checksum.

14 Hours

Unit 3- Transport Layer – TCP, Network Layer and Internet Protocol

Principles of Reliable Data Transfer: Building a Reliable Data Transfer Protocol, Pipelined Reliable Data Transfer Protocol, Go–Back–N Protocol, Selective–Repeat; Connection Oriented Transport TCP: The TCP Connection, TCP Segment Structure, Flow Control, TCP Connection Management, TCP Congestion Control. Numerical on TCP congestion control mechanisms–TCP Tahoe, Reno.

Overview of Network Layer: Forwarding and routing, what's Inside a Router? The Internet Protocol (IP) IPV4: Datagram Format, Fragmentation, Addressing, NAT.

14 Hours

Unit 4 - Network Layer and Internet Protocol, Link Layer and LAN

Introduction to Network layer Protocols: DHCP, ICMP; IPv6 Protocol: Packet Format, Transition from IPv4 to IPv6; Introduction to Routing Algorithms: Link State: Dijkstra's algorithm and Distance Vector: Bellman–Ford Algorithm.

Link layer – Error–Detection and Correction techniques, Parity checks, Internet Checksum, Cyclic Redundancy Check, and Multiple Access Protocols: CSMA/CD, CSMA/CA; Switched LAN: Link layer addressing and ARP, Ethernet: Link–layer switches. Retrospective: A Day in the Life of a Web Page Request. Physical Layer – Purpose, Signals to Packets, Transmission media. Wireless LANs: IEEE 802.11 LAN architecture, 802.11 MAC Protocol, IEEE 802.11 Frame.

14 Hours

Tools: Wireshark, Python.

Text Book(s):

1: "Computer Networking: A Top – Down Approach", James F. Kurose, Keith W. Ross, 7th Edition, Pearson Publication, 2017.

Reference Book(s):

1: "TCP IP Protocol Suite", Behrouz Forouzan, 4th Edition, McGraw–Hill, 2010.

Lab/ Hands-on: **14Hours**

1. Program on ping, tcpdump and wireshark.

2. Program on Exploring HTTP with wireshark, Web Server setup, FTP/ SMTP and SNMP Clients, Telnet, SSH and DNS

3. Program on Wireshark based TCP congestion window plotting, UDP traffic analysis.

4. Program on Cisco Packet Tracer based Router experiments; IPv4 Fragmentation based Wireshark experiments, Inspection of DHCP, ICMP.5. Program on Wireshark based Link Layer protocol inspection.

UE23CS241B - Design and Analysis of Algorithms (4-0-0-4-4)

Algorithms play a key role in science and practice of computing. Learning algorithm design technique is a valuable endeavour from practical standpoint and algorithm design techniques have considerable utility as general problem solving strategies, applicable to problems beyond computing. This course includes classic problems of computer science, application of design techniques and analysis in terms of time and space.

Course Objectives:

- Learn various algorithm design techniques and apply appropriate algorithmic design techniques for specific problems.
- Learn to design and analyze algorithms with an emphasis on resource utilization in terms of time and space
- Learn to trade space for time in algorithmic design using input enhancement and per-structuring.
- Learn the limitations of algorithmic power and techniques to handle these limitations

Course Outcomes:

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

- Identify the design technique used in an algorithm
- Design and implement efficient algorithms for practical and unseen problems and analyze these algorithms using quantitative evaluation.
- Analyse time efficiency over trading space
- Understand the limits of algorithms and the ways to cope with the limitations.

Course Content:

Unit 1: Introduction and Brute Force

Algorithms, Fundamentals of Algorithmic Problem Solving, Important Problem Types. Analysis of Algorithm Efficiency: Analysis Framework, Asymptotic Notations and Basic Efficiency Classes, Mathematical Analysis of Non Recursive and Recursive Algorithms. Brute Force: Selection Sort, Bubble Sort, Sequential Search, Brute Force String Matching, Exhaustive Search. **14 Hours**

Unit 2: Decrease – and – Conquer & Divide-and-Conquer

Decrease-and-Conquer: Decrease by constant number algorithms - Insertion Sort, Topological Sorting, Algorithms for Generating Combinatorial Objects, Decrease-by-a-Constant-Factor Algorithms – Fake Coin Problem, Russian Peasant Multiplication, Josephus problem, Decrease-by-Variable-Size Algorithms – Computing a median and the selection problem. Divide-and-Conquer: Master Theorem, Merge Sort, Quick Sort, Binary Search, Binary Tree Traversals, Complexity analysis for finding the height of BST, Multiplication of Large Integers, Strassen's Matrix Multiplication.

14 Hours

Unit 3: Transform-and-Conquer Space and Time Tradeoffs & Greedy Technique

Transform and Conquer**:** Pre-sorting, Heap Sort, Red-Black Tree Construction and Time complexity Analysis for insert and search operation, 2-3 Trees and B Tree: insertion, deletion, searching, and time complexity analysis. Space and Time Tradeoffs: Sorting by Counting, Input Enhancement in String Matching - Horspool's and Boyer-Moore Algorithms. Greedy Technique: Prim's Algorithm, Kruskal's Algorithm and union-find algorithm, Dijkstra's Algorithm, Huffman Trees **14 Hours**

Unit 4: Limitations, Coping with the Limitations of Algorithm Power & Dynamic Programming,

Dynamic Programming: Computing a Binomial Coefficient, The Knapsack Problem and Memory Functions, Warshall's and Floyd's Algorithms. Limitations of Algorithm Power**:** Lower-Bound Arguments, Decision Trees, P, NP, and NP-Complete, NP-Hard Problems. Coping with the Limitations of Algorithm Power: Backtracking, Branch-and-Bound. **14 Hours**

Tools / Languages: C Programming Language, GCC Compiler.

Text Book(s):

1: "Introduction to the Design and Analysis of Algorithms", Anany Levitin, Pearson Education, Delhi (Indian Version), 3rd Edition, 2012.

Reference Book(s):

1: "Introduction to Algorithms", Thomas H Cormen, Charles E Leiserson, Ronald L Rivest and Clifford Stein, Prentice-

Hall India, 3rd Edition, 2009.

2: "Fundamentals of Computer Algorithms", Horowitz, Sahni, Rajasekaran, Universities Press, 2nd Edition, 2007.

3: "Algorithm Design", Jon Kleinberg, Eva Tardos, Pearson Education, 1st Edition, 2006.

UE23CS242B: Operating Systems (4-0-0-4-4)

This course focuses on fundamental operating systems concepts including various algorithms and trade-offs for efficient management of resources such as CPU, Memory, Storage and I/O. This course requires the student to have a desirable knowledge of Data Structures and its Applications.

Course Objectives:

- Learn the fundamental Operating System concepts and various algorithms for process management.
- Demonstrate various algorithms and associated trade-offs for efficient resource management such as inter process communication, threads, process synchronization, deadlocks.
- Provide an understanding and apply various memory management techniques.
- Understand file system and secondary storage structures.

Course Outcomes:

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

- Understand the design of various algorithms for scheduling and their relative performance.
- Understand and apply Inter Process Communication, threads, process synchronization and deadlocks
- Apply various memory management techniques.
- Understand file system, its implementation and the various secondary storage structures.

Desirable Knowledge: UE23CS252A - Data Structures and its Applications.

Course Contents:

Unit 1 : Introduction and Process Management

What Operating Systems Do, Operating-System Structure & Operations, Kernel Data Structures, Operating-System Services, Operating System Design and Implementation.

Shell programming: Overview of bash shell programming – variables, control flow

Processes: process concept, Process Scheduling, Operations on Processes, System calls for process management-fork (), vfork (), wait () and exec ().

CPU Scheduling: Basic Concepts, Scheduling Criteria, Scheduling Algorithms. Case Study: Linux Scheduling Policies.

Shell programming – cron.**14 Hours**

Unit 2 : IPC, Threads and Concurrency

IPC: Introduction, Shared Memory systems, Message Passing, Communication in Client–Server Systems-Pipes, ordinary pipes and named pipes, system calls for shared memory, pipes and FIFO's.

Threads: Overview, Multicore Programming, Multithreading Models, Thread Libraries, Thread Scheduling.

Process Synchronization: Background, The Critical-Section Problem, Peterson's Solution, Synchronization Hardware, Mutex Locks, Semaphores, Classic Problems of Synchronization- The Bounded-Buffer Problem, The Readers–Writers Problem, The Dining-Philosophers Problem, Synchronization Examples- Synchronization in Linux. System calls for threads creation and synchronization-POSIX Threads. **Deadlocks:** System Model, Deadlock Characterization, Deadlock avoidance, Banker's Algorithm, Deadlock Detection, Deadlock Recovery.**14 Hours**

Unit 3: Memory Management

Main Memory: Background- Basic Hardware, Address Binding, Logical Versus Physical Address Space, Dynamic Loading, Dynamic Linking and Shared Libraries, Swapping, Contiguous Memory Allocation, Segmentation, Paging, Structure of the Page Table.

Virtual Memory: Background, Demand Paging, Copy-on-Write, Page Replacement Algorithms-FIFO, LRU, Optimal, Allocation of Frames, Thrashing.**14 Hours**

Unit 4 : File and Storage Management

File-System Interface: File Concept, system calls for file operations-open(), read(),write(), lseek(), close() and system call to retrieve file attributes and file types-stat(), lstat(), Access Methods, Directory and Disk Structure, system calls for reading directories, system calls to create hard links (link()) and symbolic links-symlink().

File-System Implementation: File-System Structure, File-System Implementation, Directory Implementation, Allocation Methods, File Sharing, Protection.

Storage management: Overview of Mass-Storage Structure, Disk Scheduling, Swap-Space Management, RAID Structure.

System Protection: Goals, Principles and Domain of Protection, Access Matrix, Implementation of the Access Matrix, Access Control; **System Security**

Shell programming - awk, sed**14 Hours**

Tools/Languages/OS : C, Linux/Unix OS for system call implementation.

Text Book(s):

1: "Operating System Concepts", Abraham Silberschatz, Peter Baer Galvin, Greg Gagne 9th Edition, John Wiley & Sons, India Edition ,2016.

2: "Advanced Programming in the Unix Environment", Richard Stevens and Stephen A Rago, Pearson, 3rd edition, 2017.

3: "Your UNIX/Linux: The Ultimate Guide", Sumitabha Das, 3rd Edition, McGraw-Hill, 2013.

Reference Book(s):

1: "Operating Systems, Internals and Design Principles", William Stallings, 9th Edition, Pearson, 2018.

2: "Modern Operating Systems", Andrew S Tanenbaum, 3rd edition, Pearson, 2007.

3: "Learning the bash shell", Cameron Newham, 3rd edition, O'Reilly, 2005

UE23CS243B - Vector Spaces and Linear Algebra (4-0-0-4-4)

This is a basic subject on matrix theory and linear algebra. Emphasis is given to topics that will be useful in computer science discipline, including systems of equations, vector spaces, eigenvalues, similarity, and positive definite matrices.The course provides hands-on experience in basic programming concepts using PYTHON/R for solving problems relevant to these areas.

Course Objectives:

- The first goal of the course is to teach students how to use linear algebra as a powerful tool for computation.
- The second goal is to show how these computations can be conceptualized in a geometric framework.
- The third goal is to give a gentle introduction to the theory of abstract vector spaces.
- To visualize solution to linear system of equations with different approaches using Python.

Course Outcomes:

After completing this course, students will be able to:

- Solve systems of Linear Equations using Matrix Transformations, Interpret the nature of Solutions, Visualize Consistency of Linear system of Equations and also compute inverse of a Matrix.
- Demonstrate the ability to work within Vector Spaces, distil Vector Space properties and understand the concepts of the four fundamental Subspaces, Linear Span, Linear Independence, Dimension and Basis
- Learn the concepts of Orthogonal Vectors and Orthogonal Subspaces and apply the Gram-Schmidt process to find an Orthonormal Basis in a Subspace, Eigenvalues, Eigenvectors and Diagonalization of a Matrix.
- Apply the concept of Positive Definite Matrices, Singular Value Decomposition into Application problems.

Course Content:

Unit 1: Matrices, Gaussian Elimination and Vector Spaces

Introduction, The Geometry of Linear Equations, Gaussian Elimination, Singular Cases, Elimination Matrices, Triangular Factors -LU decomposition & **Cholesky's method** and Row Exchanges, Inverses and Transposes, Inverse by Gauss -Jordan method, Vector Spaces and Subspaces (definitions only)

Application:

1: Basic operations with matrices in Matlab
2: Matrix operations and image manipulation
3: Matrix multiplication, inversion, and photo filters
4: Solving linear systems

Self-Learning Component: Algebra of Matrices.

14 Hours

Unit 2: Four Fundamental Subspaces & Linear Transformations

Linear Independence, Basis and Dimensions, Row reduced Echelon form, The Four Fundamental Subspaces, Rank-Nullity theorem. Linear Transformations, Algebra of Linear transformations,

Application:

1: Systems of linear equations and college football team ranking (with an example of the Big 12)
2: Convolution, inner product, and image processing revisited
3: Norms, angles, and your movie choices
4: Interpolation, extrapolation, and climate change

Self-Learning Component: Examples of Vector Spaces and Subspaces.

14 Hours

Unit 3: Orthogonalization, Eigenvalues and Eigenvectors

Orthogonal Vectors and Subspaces, Orthogonal Bases, Cosines and Projections onto Lines, Projections and Least Squares. Orthogonalization, The Gram-Schmidt Orthogonalization process, Introduction to Eigenvalues and Eigenvectors, Properties of Eigenvalues and Eigenvectors, Cayley-Hamilton theorem (statement only), Diagonalization of a Matrix

Applications:

1. Orthogonal matrices and 3D graphics
2. Discrete dynamical systems, linear transformations of the plane, and the Chaos Game

3. Projections, eigenvectors
4. Matrix eigenvalues and the Google's PageRank algorithm

14 Hours

Unit 4: Singular Value Decomposition
Symmetric Matrices, Quadratic Forms, Definitions of Positive definite, negative definite, positive semi-definite, negative semi-definite, indefinite forms and matrices, Tests for Positive Definiteness, Singular Values and Singular Vectros, Image Processing by Linear Algebra, Principal Component Analysis(PCA by the SVD), Minimizing a Multivariate Functions, Back Propagation and Stochastic Gradient Descent
Applications:

1. Principal Component Analysis, and face recognition algorithms
2. Social networks, clustering, and eigenvalue problems
3. Singular Value Decomposition and image compression

14 Hours

Tools ' Languages : Python IDE

Text Book(s):
1. Linear Algebra and its Applications, Gilbert Strang, Thomson Brooks/ Cole, 5th Edition, Second Indian Reprint 2007.
2. Application : Applied projects for an introductory linear algebra class by Anna Zemlyanova

Reference Book(s):
1. Linear Algebra and its Applications, David. C lay, Publication by Pearson Education, 5th Edition, 2015
2. Linear Algebra, Schaum's outlines, Seymour Lipschutz and Marc Lipson, Tata McGraw-Hill publications, 4thEdition, 2009.
3**.** Higher Engineering Mathematics, B S Grewal, Khanna Publishers, 44th Edition, 2020,.
4. Practical Linear Algebra, Gerald Farin and Dianne Hansford, CRC Press, Taylor & Francis Group, 3rd Edition, 2013.