UNIVERSITY OF MUMBAI
No. UG/42 of 2016-17

CIRCULAR:-

A reference is invited to the Syllabi relating to the B.Sc. degree course , **vide** this office Circular No. UG/231 of 2009, dated 16th June, 2009 and the Principals of affiliated Colleges in Science are hereby informed that the recommendation made by Ad-hoc-Board of Studies in Science at its meeting held on 20th May, 2016 has been accepted by the Academic Council meeting held on 23rd May, 2015 **vide** item No. 4.12 and that in accordance therewith, the revised syllabus as per the Credit Based Semester and Grading System for S.Y. B.Sc. Computer Science (Sem.III & IV), which are available on the University's web site (www.mu.ac.in) and that the same has been brought into force with effect from the academic year 2016-17.

MUMBAI – 400 032
5th August, 2016

(Dr.M.A.Khan)
REGISTRAR

To,
The Principals of the affiliated Colleges in Science and the Heads of Recognized Institutions concerned.

A.C/4.12/23/05/2016

****

No. UG/42-A of 2016 MUMBAI-400 032 5th August, 2016

Copy forwarded with Compliments for information to:-

1) The Deans, faculties of Science,
2) The Chairman, Board of Studies in Science,
3) The Professor-cum-Director, Institute of Distance & Open Learning (IDOL)
4) The Director, Board of College and University Development,
5) The Co-Ordinator, University Computerization Centre,
6) The Controller of Examinations.

(Dr.M.A.Khan)
REGISTRAR

PTO..

UNIVERSITY OF MUMBAI

For Information Only

Syllabus for the S.Y.B.Sc.

Program: B.Sc.

Course: Computer Science

(Credit Based Semester and Grading System with effect from the academic year 2016–2017

Preamble

With rapid and steady advances in diverse areas of computing, industry requirements are changing. The three-year B.Sc. Computer Science course is aimed at laying a foundation of software and hardware concepts, supplemented by practical techniques.

The syllabus is designed in such a way that the first year of the course provides core basic concepts of the subject and forms the foundation for further semesters. It attempts to provide technology-oriented students with the knowledge and ability to develop creative solutions, and better understand the effects of future developments of computer technologies.

This syllabus is the continuation to the previous semester's syllabus. It is believed that the syllabus will offer an enriched learning experience and by end of second year, the student will be able to work on several of the prevalent programming platforms.

S.Y.B.Sc. Computer Science Syllabus
Credit Based System and Grading System
Academic year 2016-2017

SEMESTER III

Course	TOPICS	Credits	Lecture/ Week
USCS301	Discrete Mathematics	2	3
USCS302	Object Oriented Design using UML and Python	2	3
USCS303	Data Structures and Algorithms Using Python	2	3
USCSP03	Practical of USCS301 + USCS302 + USCS303	3	9

SEMESTER IV

Course	TOPICS	Credits	Lecture/ Week
USCS401	Operating System and Linux	2	3
USCS402	Java Programming	2	3
USCS403	Web Technologies	2	3
USCSP04	Practical of USCS401 + USCS402 + USCS403	3	9

Semester III – Theory

Course: USCS301	TOPICS (Credits : 02 Lectures/Week:03) Discrete Mathematics	

Objectives : To provide an insight into concepts of discrete mathematics and establish its significance in several areas of computational theories.

Expected Learning Outcomes:

1) To provide overview of theory of discrete objects, starting with relations and partially ordered sets.

2) Study about recurrence relations, generating function and operations on them.

3) Give an understanding of graphs and trees, which are widely used in software.

4) Provide basic knowledge about models of automata theory and the corresponding formal languages.

Unit I	**UNIT I: Relations** Relations: Definitions and examples. Properties of relations, Partial Ordering sets, Linear ordering, Hasse Daigrams, Maximum and Minimum elements, Recurrence Relation: Definition of recurrence relations,Formulating recurrence relations, Solving recurrence relations, Back tracking method, Linear homogeneous recurrence relations with constant coefficients. Solving linear homogeneous recurrence relations with constant coefficients of degree two when characteristic equation has distinct roots and only one root. Particular solutions of non linear homogeneous recurrence relation, Solution of recurrence relation by the method of generating functions, Applications: Formulate and solve recurrence relations for Fibonacci numbers, Tower of Hanoi.	**15 L**
Unit II	Graphs, Trees, Counting (a) Graphs : Definition and elementary results, Adjacency matrix, path matrix, Representing relations using diagraphs. Warshall's algorithm- shortest path , Linked representation of a graph, Operations on graph with algorithms - searching in a graph; Insertion in a graph, Deleting from a graph, (b) Trees: Definition and elementary results. Ordered rooted tree, Binary trees, Complete and extended binary trees, traversing binary trees, binary search tree, Algorithms for searching and inserting in binary search trees, Algorithms for deleting in a binary search tree. (See related topics in Unit 3 of USCS303 Data Structures and Algorithms.) (c) Permutations and Combinations: Partition and Distribution of objects, Permutations and combinations with distinct and indistinct objects, Binomial numbers, Pascal Identity, Vandermonde's Identity, binomial theorem, binomial coefficients and Pascal's triangle.	**15 L**

Unit III	Languages and Automata: Languages, Grammars and Machines: Chomsky hiearchy of type-0, type-1, type-2 and type-3 grammars; and the languages they generate. Regular expressions and finite state macines, context-free languages and pushdown automata, brief mention of context-sensitive languages and linear bounded automata, recursively enumerable languages and turing machines. Universal Turing machine and Turing completeness.	**15 L**

Textbooks:

1. *Elements of Discrete Mathematics*: C.L. Liu , Tata McGraw- Hill Edition .
2. *Discrete Mathematics and its applications*: Kenneth H. Rosen, Third Edition, McGraw- Hill Inc.
3. *Discrete Mathematics*: Y. N Singh, Wiley India
4. *Discrete Mathematics*: Semyour Lipschutz, Marc Lipson, Schaum's out lines, McGraw- Hill Inc.

References:

1. *Discrete Mathematics*: Norman L. Biggs:, Revised Edition, Clarendon Press Oxford 1989.
2. *Concrete Mathematics*: Graham, Knuth, Patashnik Second Edition, Pearson Education.

Course: **USCS302**	**TOPICS (Credits : 02 Lectures/Week: 03)** **Object Oriented Design using UML and Python**	

Objectives: To clearly understand the concepts of object oriented analysis and design and its application in developing software for real world applications. Introduce Unified Modeling Language and explain its importance in the software development life cycle. Concretely understand the concepts with practical implementations in modern programming environment.

Expected Learning Outcomes:

1) Gain knowledge about principles, components and structure of object oriented programming methodology.
2) Understand the behavior and interaction between the software modules through object oriented design.
3) Study about various representations, behavioral and architectural modeling techniques.
4) Explore use of UML in Object Oriented design and implementation of software designs using Python.

(It is understood that implementation of case studies will not be complete by semester-end. What is more important is, students will appreciate the difficulties faced in even relatively simple analysis and design.)

Case study for Units I and II

For practice with UML models and Python coding: Bank branch, with classes - teller, supervisor, cashier, manager (subclasses of the class BankEmployee). Together they daily serve hundreds of customers who visit the bank branch to perform various transactions.

Retail banking becomes an uncomplicated exemplar of OO programming because everyone is familiar

with it. UML constructs (like sequence diagram, class diagram, use cases) are illustrated using the banking case. The implementation or coding is done in Python, in both lecture classes and labs.

For programming purposes, the bank workers have "objects" as their images on the computer network. Each human operates the object that is their image in the computer, from login to timeout and logout.

A customer initiates a transaction by filling in a blank (paper or e-) voucher and handing it to a teller. Once initiated, a a transaction remains pending until it is ended/finished (always by a bank worker).

The teller processes the transaction and if needed, forwards it to the cashier and/or supervisor. As a mark of a pending transaction, a token is issued to the customer, and surrendered to the bank at the end of transaction.

The transactions initiated by the customers are: 1. deposit cash, 2. deposit cheques, 3. withdraw cash (with a cheque or voucher), 4. request account statement. (These form 4 high-level UML use cases.)

A teller accepts the voucher; if paper voucher, enters details into e-voucher; checks what kind of transaction (cash/cheque deposit, cash/DD withdrawal); if cheque deposit, send for clearing; if cash deposit, send to cashier; if withdrawal, verify signature cheque or voucher, verify account balance and forward to supervisor; if withdrawal via DD, initiate the DD issue process, print out the DD, send it to supervisor for signature. Further action by the teller on pending vouchers depends on responses from the supervisor re sent vouchers.

A cashier hands out or receives cash as indicated on transaction vouchers sent by the teller or supervisor. An important part of a bank branch is handling sums of cash, receiving and disbursing it, accounting for it, storing it. A cashier handles cash transactions (deposits and withdrawals), maintains accounts of cash on hand.

A supervisor verifies the correctness of the transaction vouchers and updates accounts. The vouchers are filed away, or returned to the teller, or forwarded to the cashier, or cancelled if invalid.

A manager is the final authority and arbiter in the bank branch, a responsible position with cash involved.

Case Study for Unit II and III: Simplified Order Processing System

The Simplified Order Processing System is a system designed for managing how customers place an order, doing payments after receiving the invoice and the ordered products, the retailer should also verify the availability of the stock.

A retailer checks for the availability of goods in the store. If the stock of goods is less than the reorder level, the retailer places an order for goods. The supplier supplies the goods to the store in the system. Once the ordered goods are received at the store, the retailer then arrange them by product or by price, then retailer makes payment. If the stock of goods is available then he will arrange goods for sale.

The retailer then sells the goods directly to the customer. The customer buys the items from retailer. The retailer prepares the bill for all the goods purchased by the customer, then he receives amount either by credit or by cash from customer soon after the product is delivered to the customer. We will not consider customer returns The overall system is used to manage the goods in the store and does the sales.

The system should comprise of the following set of classes – products, customer, bank, account, order-details, invoice, shipments, etc.

Case study for Unit III: Employee payroll system for the Bank

Build a new payroll system to allow employees to record timecard information electronically and automatically generate paychecks based on the no. of hours worked and the total sales for commissioned employees. It should provide a desktop interface to allow employee to enter timecard information. Some employees work by the hour, and they are paid hourly rate. They submit timecards that record the date & no. of hours worked for a particular charge number. If someone works for more than 8 hours, they should be paid 1.5 times their normal rate for those extra hours. Hourly workers are paid every friday.

Some employees are paid a flat salary but still they should provide their timecards that record the date and hours worked. They are paid on the last working day of the month.

The employee should be able to query the system for no. of hours, days worked, totals of all hours billed for specific task/charge, total amount received till a given date.

Unit I	Imperative vs object-oriented programming; fundamental ideas of OOP: encapsulation, inheritance, abstraction, polymorphism. Python and UML topics given unitwise are to be taught not serially, but together in an integrated manner. **UML**: Use simple examples first, and the case study next, to develop ideas behind class diagrams, actors and use cases, and use case diagrams. Sequence diagrams to capture use cases (CRC cards (class-responsibility-collaboration). Association, dependency, composition. Inheritance and generalization. Activity diagrams, fork, join. Overview of UML, need for a development process for using UML, the Unified Process. **Python**: The __init__ method to initialize newly-created class instances; simple class definitions with and without __init__. Implement classes from the case study: use inheritance for various subclasses of employees, and begin to develop code to implement the sequence diagrams.	**15L**
Unit II	**UML**: Communication diagram – use simple example to show various components involved (such as classes at generic level of communication or collaboration, objects at instance level of communication; messages that are exchange among different objects, and the order of exchanging the messages among the objects, flow of control, the guard conditions, time at which an object is created and destroyed). Activity diagrams – initial and final states, activity and action states, guard conditions, forking, joining and swimlanes, state transitions based on the outcome of guard conditions. **Python**: create objects by instantiation, implement __del__ method to destroy the objects. Based on the conditions that are tested, implement the user-defined method calls along with the necessary parameters to implement the communication among different objects. Multi-threading in order to create multiple flows of control especially when dealing with	**15L**

	forking and joining. Completion of the coding for bank branch case study, initiation and development of a case study for simplified order processing.	
Unit III	**UML:** State change diagrams, Events and signals, State machines, processes and Threads, time and space, state transitions, initial and final state of an object, sub states. Package diagrams – package and elements that are organized in it such as class diagrams, use case diagrams, etc. **Python:** creation of modules, packages, and importing of packages. Implementation of the order processing case study by creating multiple threads to implement concurrent flows of control, modules and packages.	**15L**

Textbook(s):

1) *Object Oriented Modeling and Design with UML,* Michel Blaha, James Rambaug, Second Edition, Pearson
2) *Python 3 Object Oriented Programming,* Dusty Phillips, PACKT Publishing

Reference(s):

3) *Object-Oriented analysis and Design: Understanding system Development with UML 2.0,* Mike O'Docherty, Wiley India
4) *Object Oriented Analysis and Design with UML,* I.K International Publishing
5) *Object Oriented Design and Patterns* - Cay Hortsman, Wiley India
6) *Introduction to Computer Science using Python,* Charles Dierbach,

Course: USCS303	TOPICS (Credits : 02 Lectures/Week: 03) Data Structures and Algorithms Using Python	
Objective: The objective of this course is to make the learner understand the basic concepts and algorithms of different data structures and its applications using Python. **Expected Learning Outcomes:** 1) Students should be able to understand the meaning of data structures and its different types. 2) Students should be able to understand the algorithm of different data structures before implementing it using Python **3)** Students should be able to develop the logic for implementing data structures.		
Unit I	**Algorithm analysis** Problem, size of problem (symbol n); runtime resources time T(n), space S(n); worst case, best case, average case. Measuring running time as a function of n with the wall clock (using function time() of module time); advantages and disadvantages. **7 standard functions:** constant *c, log n, n, n log n,* n^2, n^3, exponential c^n or 2^n; growth of these functions as n grows: for constants c_1 and c2, compare c_1 * f1(n) with c_2 * f_2(n) (for functions f_1 and f_2 from the set of these 7 functions); conclude that one function grows faster than another independent of the values of the constants. **Operation count, unit steps (constant time):** arithmetic operation (or expression evaluation or assignment), comparison (with Boolean operators <, =, >), function call and/or function return, element access (for compound types) can even treat a single loop iteration as a constant-time unit step. **Asymptotic analysis:** upper bounds with A (at most) and O notation; lower bounds with Ω notation, upper and lower bounds with Θ notation. **Problem-solving methods:** greedy method, divide-and-conquer, dynamic programming (briefly, during all 3 units) **Abstract data types** (with associated operations and applications) Define the ADTs as Python classes, and their operations as class methods. **(i) stacks:** operations push(), pop(), is_empty(); stacktop(), len() implementation using lists; applications: reverse a sequence, match parentheses in an expression (or html tags); evaluate a postfix expression.	15 L
Unit II	**(ii) queues:** operations enqueue () and dequeue(), i.e., enter() and exit(), is_empty(), first(), last()); implementation using Python lists; applications: simulation of a single-window queue (uniform, Gaussian and other distributions are available in the Python module random). **(iii) Singly, doubly and circularly linked lists**, with head and optional tail; implementation of list nodes as Python objects; operations: insertion and deletion at the front and the rear of the list, search for a value in a list, delete a value in a list; applications: simulate stack and queue, maintain a set of data in sorted order. Linear search in linked lists. **(iv) trees and binary trees,** definitions and properties; insertion and deletion of a tree node	15 L

Unit III	**(v) trees and binary trees,** implementation of binary trees in lists and in linked structures; applications: preorder, inorder and postorder traversals of binary trees; binary search trees; breadth-first and depth-first tree traversals. **(vi) graphs:** directed and undirected graphs; implementation using adjacency matrix and adjacency list; graph traversal algorithms: depth first and breadth first traversals, application: shortest paths (vii) **map** ADT, Python classes dictionary and set; applications.	15 L

Textbook(s):

1) *Data Structures and Algorithms in Python*, Goodrich, Tamassia, Goldwasser, 2016 J. Wiley
2) *Data Structures and Algorithms Using Python* - Rance D. Necaise, College of William and Mary, 2016, J. Wiley

Reference(s)

1) *Data Structure and Algorithmic Thinking with Python*- Narasimha Karumanchi, 2015, Careermonk Publications
2) *Fundamentals of Python: Data Structures*, Kenneth Lambert, Delmar Cengage Learning

Semester III Practical

USCSP03	**Practical/Tutorial of USCS301 + USCS302 + USCS303** **(Credits: 03, Practical/Week: 09)**
	Discrete Mathematics : 1) Problems based on Boolean Algebra 2) Problems based on Sets and draw Venn diagram 3) Problems based on Relations 4) Use Propositional Logic for representing and solving problems. 5) Use Predicate logic and WFF for representing and solving problems 6) Problems based on Recurrence relations 7) Problems based on generating functions. 8) Problems based on Permutation theory 9) Problems based on Combination theory 10) Problems based on Graphs
	Object Oriented Design using UML and Python : 1. Draw class diagram for the Unit 1 case study (specify the classes, i.e., name, attributes and methods but do not draw the edges between the classes). Start the corresponding Python class definitions (state the stubs for attributes and methods but do not write the code). 2. Develop the important use cases for the case study (specify the actors, the use case initiators and write the use case descriptions). Complete the class

	diagram by drawing the associations. 3. Develop the sequence diagram for a few important scenarios of the case study. Fill in the corresponding Python code. 4-5. Develop the code for the bank branch case study. 6-9. UML diagrams – Use case , class, sequence, package diagrams and develop Python code for the order processing case study. 10-11. Create multiple threads, provide synchronization for the order processing case study using Python 12-16. Draw use case, class, activity, communication and sequence diagrams for employee payroll case study and also create a payroll package with all the necessary classes for the case study and implement the methods of it by calling them in a module called employee_payroll.py
	Data Structures and Algorithms Using Python : 1) Find out space and time complexity for a given non recursive program. 2) Find out space and time complexity for a given recursive (like Fibonacci) program. 3) Write a program to implement stack and its applications. 4) Write a program to implement queue and its applications. 5) Write a program to implement singly linked list and its applications. 6) Write a program to implement doubly linked list and its applications. 7) Write a program to perform insertion and deletion of a node from a tree. 8) Write a program to print pre-order, post-order and in-order traversal of a tree. 9) Write a program to implement a binary tree and its applications. 10) Write a program to implement a graph and its applications. 11) Write a program to implement a map and its applications.

Semester IV - Theory

Course: USCS401	TOPICS (Credits : 02 Lectures/Week: 03) Operating Systems and Linux	

Objectives: To provide a sound understanding of Computer operating system, its structures, functioning and algorithms.

Expected Learning Outcomes:

1) Enable the learner to gain extensive knowledge on principles, functioning and structure of operating systems.
2) Appreciate the importance of Operating system as resource manager and gain insight into how computing resources (such as CPU and memory) are managed by the operating system.

3) Learn about process management, process scheduling, threads, synchronization, memory management, virtual memory concepts, cause and effect of deadlocks, and file system
4) Learn about Linux system, basic shell command, environmental variables, shell script, structured commands.

Unit I	**Introduction to Operating systems, its structure and Process Scheduling :** Definition Operating system, Operating System Services, System Calls and its types, System Programs, Operating system Structure **Process Management**: Concepts, Process Scheduling and Operations, Inter-process Communication, Communication in Client –Server Systems. **Multithreaded Programming :** concepts and benefits, Multithreading Models **Introduction to LINUX System:** The Linux system, Kernel Modules **bash shell, basic commands**: Navigating File system, Listing files and Directories, Handling Files, Managing directories, Viewing File Contents, Monitoring Programs and Disk space, working With data files	**15L**
Unit II	**Process Scheduling :** Scheduling Criteria and algorithms, Thread Scheduling, Multiple processor scheduling, Real-time Scheduling. **Synchronization, Deadlocks ,Memory and Storage Management** **Synchronization** : Background,, Critical-Section Problem, Peterson's Solution, Mutex Locks and Semaphores, Monitors **Deadlocks**: Concept and its characterization, Methods of deadlock handling, Deadlock prevention, Deadlock Avoidance (Safe state and Resource-allocation-graph algorithm), Deadlock Detection, Recovery from Deadlock **Linux bash commands:** Linux environment variables, setting environment variables, Removing environment variables, Variable arrays **bash scripting:** Creating a script file, Displaying messages, Using variables, Redirecting Input and Output, Pipes, Performing math	**15L**
Unit III	**Memory Management Strategie**s **:** Swapping, Contiguous Memory Allocation, Segmentation, Paging and page tables **Virtual Memory Management** : Concept, Demand Paging, Copy-on-Write, Page replacement **File System :** Concept, Access methods, Structure, Directory and disk structure, File system Mounting **bash structured commands**: Working with the if-then, test command, Compound condition testing, Advanced if then features, the case command, for command, until command, while command	**15L**

Textbook(s):

1) *Operating Systems Concepts*- Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne, 9th Edition, John Wiley
2) *Linux Command Line and Shell Scripting BIBLE*, Richard Blum, John Wiley
3) *Official Ubuntu Book 8th Ed Ver 0.2,* by Matthew Helmke & Elizabeth K. Joseph with Jose Antonio Rey and Philips Ballew

Reference(s):

1) *Operating Systems: Internals and Design Principles*, 8th edition, William Stallings; Prentice Hall.
2) *Operating Systems*, 3rd edition, Gary Nutt; Pearson/Addison Wesley.
3) *Modern Operating Systems*, 3rd edition, Andrew S. Tanenbaum; Prentice Hall.
4) https://help.ubuntu.com/community/Java

Course: USCS402	TOPICS (Credits : 02 Lectures/Week: 03) Java Programming	
Objective : The objective of this course is to teach the learner how to implement the code in Object oriented environment and understand the concepts of Core java and to cover-up with the pre-requisites of Core java. **Expected Learning Outcomes :** Through this course there will be an enhancement to 1) Object oriented programming concepts using Java. 2) Knowledge of input, its processing and getting suitable output. 3) Understand, design, implement and evaluate classes and applets. 4) Knowledge and implementation of AWT package.		
Unit I	**Introduction to Java and evolution :** Classes and Objects, data abstraction and encapsulation, inheritance, polymorphism, dynamic binding and message communication **Overview of java language :** writing simple java program, an application with two classes, structure, tokens, statements, implementing java program, JVM, command line arguments, programming style. Constants, variables and data types, declaration of variable, value and scope of variable, symbolic constants, type casting, getting value of variable and standard default value. Operator and expressions **Decision making and branching:** decision making with if statement, simple if, if…else, nesting of if….. else, else if ladder, Switch statement, ? Operator. **Decision making and looping :** WHILE statement, DO statement, FOR statement, JUMP in loops, labeled loops.	**15L**
Unit II	**Working with Classes, objects and methods :** Array Strings, Vectors and Wrapper classes. Interfaces, multiple inheritances, Package and its implementation - putting classes together. JAVA API packages. **Multithreaded programming:** Creating Threads, extending the Thread class, stopping and blocking a Thread, life cycle of a Thread, using Thread methods, Thread exceptions, Thread priority, Synchronization,	**15L**

	implementing “runnable” interface. **Error and Exception Handling:** Managing errors, exceptions and using exception for debugging.	
Unit III	**Applet programming :** Writing Applet, building the code, Applet life cycle, creating an executable Applet and adding Applet to HTML file, running the Applet, passing parameters to an Applet, aligning the display. **Graphics programming:** Introduction to Graphics class, lines, rectangles, circle, ellipses, drawing arcs, polygons, line graphs, using control loops in Applet, drawing bars and charts. **AWT package:** Window fundamentals - Component, Container, Panel, Window, Frame, and Canvas. AWT Controls: Label, Button, TextField, TextArea, CheckBox, CheckBoxGroup, Choice, and List. **Layout Manager:** FlowLayout, BorderLayout, GridLayout. **Tour of Swing:** Introduction to Swing and implementation of its basic components.	**15L**

Text book(s):

1) *Programming with java a primer by E-Balagurusamy*, Tata Mc graw Hill
2) *Java 8 Programming Black Book*, Dt Editorial Services, Dreamtech Press
3) *Java The Complete Reference* 8th Edition, Herb Schildt, Oracle Press, McGraw Hill Education
4) *Object Oriented Programming with Java: Essentials and Applications*, Tata McGraw Hill

Reference(s):

1) *Java 8 in Action*, Mario Fusco , Raoul - Gabriel Urma , Alan Mycroft
2) *Java SE 8 for programmers*, third edition by paul dietel , Harvey daitel (Deitel Developer Series)

Course: USCS403	TOPICS (Credits : 02 Lectures/Week: 03) Web Technologies	
Objectives: To provide insight into technologies used to develop web applications and learn about creating, displaying and managing web contents. Explore about Client Side and Server side using Markup and Scripting Languages. **Expected Learning Outcomes**: 1) To gain knowledge about Markup Languages for developing web applications. 2) Enhance Presentation of contents using Style Sheets. 3) To understand and implement the Client side validations using Scripting language. 4) To understand and implement the Sever side validations using Scripting language.		
Unit I	**Web Programming using HTML** **Introduction to Web Technologies**: HTML Fundamentals, HTML 4.0 elements, and Tags, Attributes, Event Handlers, Document Structure Tags, Working with Text, Formatting Tags, List Tags, links and URLs, Hyperlinks, Image & Image map, color, Table Tags, Form Tags, Frame Tags, Executable Content Tags **Introduction to HTML 5:** Difference between HTML4.0 and HTML5, Features of HTML5, New Tags in HTML5, Working with Multimedia-Use of Audio and Video Tags.	15L
Unit II	**CSS and Java Script** **CSS**: Introduction to CSS, CSS Sectors, CSS in HTML, Inline Styles – Embedding Styles- Linking External Style Sheets, Working with background, color, font and text with CSS, Display and positioning an element, Effects, Frames and controls in CSS **Introduction to Java Script**: Features and Fundamentals, Functions, Events, Objects in Java Script, Browser Objects, Java Script in Web Browser, The Document Object Model, Events and Event Handling, Forms and Form Elements.	15L
Unit III	**Creating and Using PHP Applications** **Introduction to PHP**: Features and advantages, Creating and running PHP Scripts, Handling errors, Using Variables and Constants, Data Types, Operators, Control Structures, String Functions, Array and Array Functions like $_GET, $_POST, $_REQUEST. **Controlling Program flow**: Conditional and Looping Structures, Break, continue and Exit Statements **Forms and Database**: Web Forms, Working with FORM tag, Form processing and Validations, Working with Databases-PHP and MySQL, connection, Adding, altering, Inserting, Modifying and Retrieving Data	15L

Textbook(s):

1) *Using HTML 4, XML & JAVA* by Eric Ladd & Jim O'Donnell. (Platinum Edition) (PHI)
2) *Web Technologies* -Black Book Series, DT Editorial Service, Dream Tech Press.
3) *HTML5 Black Book: Covers CSS3, JAVASCRIPT, XML,XHTML, AJAX, PHP and JQUERY* DreamTech Press.
4) *Beginning PHP6, Apache, MySQL Web Development*, Timothy Boronczyk, Wrox Publication.

Reference(s):

1) *Murach's HTML5 and CSS3* by Zak Ruvalcaba,Anne Bohem, Shroff Publishers and Distributors
2) *Web Technology*, Ralph Moseley, Wiley India
3) *HTML 5 for Beginners*, Firuza Aibra, Shroff Publishers and Distributors
4) Mike Mcgrath, "*PHP & MySQL in Easy Steps*", Tata McGraw Hill
5) *Learning PHP, MySQL, JavaScript, CSS & HTML5*, 3rd Edition by Robin Nixon

Semester IV Practical

USCSP04	Practical/Tutorial of USCS301 + USCS302 + USCS303 (Credits: 03, Pract/Week: 09)

Operating System and Linux :

1) **Installation of Ubuntu Linux operating system**
   a) Booting and Installing from (USB/DVD)
   b) Installing from the Minimal CD
2) **Introduction to bash**: Basic shell commands for directory and file manipulation, like ls, cd, pwd, cp, mv, rm
3) **Finding and Installing Ubuntu Applications**
   a) Using Ubuntu software center
   b) Using Synaptic
   c) Explore useful software packages.
4) **More bash commands:** like echo, history, date, chmod, who, man
5) **Customizing Ubuntu for performance , Accessibility and Fun**
   a) Appearance Tool
   b) Unity Tweak Tool
   c) Compiz Config Setting Manager
   d) Unity Lenses and scope
6) **Shell scripting I**
   a) defining variables, reading user input
   b) Conditions (if - then, case) arithmetic operations
7) **Becoming an Ubuntu power user**
   a) Administering system and User setting
   b) Learning Unity keyboard Shortcuts
   c) Using the Terminal
   d) Working with windows programs
8) Working with data files (sort, grep, linux File compression Utilities –bzip2,

	gzip, zip, Archiving data- tar) 9) **Shell scripting II** Conditions (for loop, until loop and while loop) arithmetic operations **10) Shell scripting III-** Redirecting Output in Scripts , Redirecting Input in Scripts, Creating Your Own Redirection 11) Working and managing with processes: sh, ps, kill, nice, at and batch etc. 12) **Using javac compiler**
	Java Programming : 1) Write java programs to demonstrate following a) A single class b) Multiple classes c) Use of command line arguments d) Read data from keyboard. e) Creating and casting of variables. 2) Write java programs to implement various operators and functions using arithmetic expression a) Arithmetic operators b) Relational operator c) Logical operator d) Assignment operator e) Increment and decrement operator f) Conditional operator g) Bitwise and special operator h) Arithmetic operators i) Mathematical functions. 3) Write Java program that illustrates the concepts of selection statement, looping, nested loops, breaking out of loop. 4) Write Java programs that illustrates the concepts of Java class that includes a) Constructor with and without parameters, destructor b) Overloading methods. c) Creating objects d) Static members 5) Write Java program to demonstrate inheritance by creating suitable classes. 6) Write Java program that illustrates the concepts of one and two dimension arrays and strings. 7) Write Java programs that illustrates the concept of following:- a) Interface in java b) Package in java c) Multithreading in Java 8) Write program that illustrates the error handling using exception handling. 9) Write Java applet to demonstrate graphics, Font and Color classes. 10) Write Java program to illustrate AWT package, Event, classes and listeners and swing package.

	Web Technologies : **Write a program to** 1) Create a web applications-Frame Tags and Image Mapping 2) Create a web applications-List tags, Table tags 3) Create and use a web application-Form Tags 4) Create application using HTML5 Tags -Audio Tags and Video Tags 5) Apply CSS(Internal and External style) to a Web Page 6) Execute different Control structures using JavaScript 7) Execute Client Side Scripts using JavaScript 8) Retrieve data from HTML form using PHP 9) Retrieve Employee Details/ Registration Details from the database using PHP 10) Add, Modify and Delet data from client side into table in MySQL **Note**: Use of Freeware\Open source software for server is recommended

Evaluation Scheme

I. Internal Exam - 25 Marks

(i) Test – 20 Marks

20 marks Test – Duration 40 mins
It will be conducted either using any open source learning management system such as Moodle (Modular object-oriented dynamic learning environment) Or a test based on an equivalent online course on the contents of the concerned course(subject) offered by or build using MOOC (Massive Open Online Course) platform.

(ii) 5 Marks – Active participation in routine class instructional deliveries
Overall conduct as a responsible student, manners, skill in articulation, leadership qualities demonstrated through organizing co-curricular activities, etc.

II. External Exam - 75 Marks

1. Duration - 2.5 Hours.
2. Theory question paper pattern:-

All questions are compulsory.			
Question	Based on		Marks
Q.1	Unit I		[20]
Q.2	Unit II		[20]
Q.3	Unit III		[20]
Q.4	Unit I, II and III		[15]

All questions shall be compulsory with internal choice within the questions.

Each Question may be sub divided into sub questions as a, b, c, d & e, etc & the allocation of Marks depends on the weightage of the topic.

III. Practical Exam –

- Each course carry 50 Marks : 40 marks + 05 marks (journal) + 05 marks (viva)
- Minimum 75 % practical from each paper are required to be completed and written in the journal**.**

(Certified Journal is compulsory for appearing at the time of Practical Exam)