B.Sc (Hons.) in Computer Science with Specialization in AI & Data Science Syllabus

Four-Year Undergraduate Program (FYUGP)

Department of Computer Science and Engineering
Bhattadev University, Bajali

August 2024

Introduction

The University Grants Commission (UGC) has initiated several measures to bring equity, efficiency, and excellence in the Higher Education System of the country. The important measures taken to enhance academic standards and quality in higher education include innovation and improvements in curriculum, teaching-learning process, examination, and evaluation systems, besides governance and other matters. Due to the various diversities present in the system of higher education, there are multiple approaches followed by universities towards examination, evaluation, and grading systems. However, the academic reforms recommended by the UGC in the recent past have led to an overall improvement in the higher education system. Based on these recommendations, apart from flexibility and freedom in designing the examination, there is a need to devise a sensible system for awarding grades based on students' performance.

The NEP2020 based Four-Year Undergraduate Programme (FYUGP), being adopted by Bhattadev University, is an 8-semester (4-year) programme of 160 credits with multiple exit and entry options at the successful completion of courses assigned at the end of each year.

- Students who opt to exit after completing the first year and have secured 40 credits will be awarded a certificate if, in addition, they complete one vocational course of 4 credits during the summer vacation of the first year.
- Students who opt to exit after completing the second year and have secured 80 credits will be awarded the diploma if, in addition, they complete one vocational course of 4 credits during the summer vacation of the second year.
- Students who opt to exit after completing the third year and have secured 120 credits will be eligible for the Bachelor's degree in the Major discipline without Honours.
- Students after completing the fourth year and have secured 160 credits will be eligible for the Bachelor's degree with Honours in the Major discipline.
- Students are allowed to re-enter the degree programme within three years and complete the degree programme within the stipulated maximum period of seven years.

Exit and Re-entry Options

- **Certificate:** Students exiting after the first year with 40 credits, including a vocational course of 4 credits during the summer vacation of the first year, receive a certificate. Students are allowed to re-enter the degree program within three years and must complete it within a maximum of seven years.
- **Diploma:** Exiting after the second year with 80 credits and an additional vocational course of 4 credits during the second year's summer vacation of the second year earns a diploma. Students are allowed to re-enter the degree program within three years and must complete it within a maximum of seven years.

Table 1: Minimum Credit Requirements to Award Degree under Each Category

S.No.	Broad Category of Course	Minimum Credit Requirement	
		3-year UG	4-year UG
1	Major (Core)	60	80
2	Minor Stream	24	32
3	Multidisciplinary	09	09
4	Ability Enhancement Courses (AEC)	08	08
5	Skill Enhancement Courses (SEC)	09	09
6	Value Added Courses common for all UG	06 - 08	06 – 08
7	Summer Internship	02 - 04	02 – 04
8	Research Project / Dissertation	-	12
Total		**120**	**160**

- **Bachelor's Degree (3-year):** Completion of three years in the major discipline and 120 credits awards a bachelor's degree in Computer Science (AI and Data Science). For more details see Table 1.
- **Bachelor's Degree with Honours (4-year):** On completing four years in the major discipline and securing 160 credits, students earn a bachelor's degree with honours in Computer Science (AI and Data Science). For more details see Table 1.
- **4-year UG Degree (Honours with Research):** Students who secure 75% marks and above in the first six semesters and wish to undertake research at the undergraduate level can choose a research stream in the fourth year. They should do a research project or dissertation under the guidance of a faculty member of the University/College. The research project/dissertation will be in the major discipline. The students who secure 160 credits, including 12 credits from a research project/dissertation, are awarded UG Degree (Honours with Research).

Outline of Courses

The broad categories of courses and minimum credits required for the 4-year Honours degrees as per the UGC document are as follows:

- **Major (Core) course/paper:** 80 credits
- **Minor course/paper:** 32 credits
- **Interdisciplinary course/paper (IDC):** 9 credits
- **Ability Enhancement Course/paper (AEC):** 8 credits
- **Skill Enhancement Course/paper (SEC):** 9 credits
- **Value Added Course/paper (VAC):** 8 credits
- **Summer Internship:** 2 credits
- **Research Project/Dissertation:** 12 credits (for Honours with Research degree)

The following points may be noted:

- In lieu of the Research Project, a student may study 3 courses each of 4 credits (i.e., total 12 credits), leading to an Honours degree (without Research).
- For the 4-year Honours degrees, the Major subject/discipline requires 80 credits, and the Minor subject/discipline requires 32 credits.
- For a Double Major, the minimum credit requirements are 48 (3-year degree) and 60 (4-year Honours degree) respectively in a subject/discipline other than the original Major.

Definition of Keywords

In FYUGP the terminlogies those are relevant to the B.Sc. curricula have been briefly described below.

- `Academic Year:` Two consecutive (one odd + one even) semesters constitute one academic year.
- `Semester:` Each semester will consist of 15 weeks of regular academic work. The odd semester may be scheduled from July to December and even semester from January to June under normal circumstance.
- `Programme:` An educational programme leading to award of a Certificate, Diploma or Degree (B.Sc., B.A., etc.)
- `Discipline:` This means a particular subject.
- `Course:` Each programme is equipped with a number courses of various disciplines/subjects. The course of a particular discipline/subject refers to the content of the papers the students have to study in that discipline/subject required in obtaining a degree. The courses should define learning objectives and learning outcomes. A course may be designed to comprise lectures /tutorials/laboratory work/ field work/outreach activities/project work /seminars /assignments / presentations etc. or a combination of any of these.
- `Honours:` A particular discipline/subject that a student opts as major subject. (e.g. honours in Computer Science)
- `Core Course (CC):` A discipline/subject specific compulsory basic course.
- `Skill Enhancement Course (SEC):` A course designed by a department for enhancement of skill of the students in a particular discipline/subject.
- `Minor Course (M):` A course in a discipline/subject corresponding to a subject other than the major subject.
- `Value Added Course (VAC):` Value-based education to include managment of biological resources and biodiversity for the development of humanistic, ethical, sustainable development and living, constitutional, and universal human values of truth, righteous conduct, peace, love, nonviolence, scientific temper, citizenship values, and life skills.

- `Ability Enhancement Compulsory Course (AECC):` These are compulsory courses. For science programme there will be two of them. AECC-1 is Communicative English & AECC-2 is Environmental Science.

- `Vocational Course (VOC):` A vocational course is focused on practical work, preparing students for a particular trade or skilled profession. These courses are best for students who have a good idea of their career path and want to gain the knowledge to get there.

- `Levels of Courses:`
  100 - 199 : Foundation or introductory courses.
  200 - 299 : Intermediate level courses.
  300 - 399 : Higher level courses.
  400 - 499 : Advanced courses.

- `Credit:` A unit by which the course work is measured. It determines the number of hours of instructions required per week. *Theory/Tutorial classes:* 1 credit = 1 hour / week and *Practical classes:* 1 credit = 2 hours / week

- `Credit Point:` It is the product of grade point and number of credits for a course.

- `Letter Grade:` It is an index of the performance of students in a said course.

- `Grade Point:` It is a numerical weight allotted to each letter grade on a certain point scale. The following Table 2 explains the above two points

Table 2: Grade Point Table

Letter Grade	Grade Point	Performance	Letter Grade	Grade Point	Performance
O	10	Outstanding	C+	5	Average
A+	9	Excellent	C	4	Pass
A	8	Very Good	F	0	Fail
B+	7	Good	I	0	Absent/Incomplete
B	6	Above Average			

Semester Grade Point Average (SGPA)

This is an average score that shows how well a student has performed in a single semester, calculated by dividing the total credit points by the total credits taken in that semester.

SGPA (Semester Grade Point Average)

- Measures a student's performance in a single semester.
- Calculated using the formula:

$$\text{SGPA}(S_i) = \frac{\sum(C_i \times G_i)}{\sum C_i}$$

Where:

- C_i = Number of credits for the i^{th} course
- G_i = Grade points obtained in the i^{th} course
- $\sum$ (sum over all the courses taken by the student during the semester)

Cumulative Grade Point Average (CGPA)

This score reflects the overall academic performance of a student over all semesters, calculated by dividing the total credit points earned in all semesters by the total number of credits taken across these semesters.

CGPA (Cumulative Grade Point Average)

- Reflects the overall academic performance of a student over all semesters.
- Calculated using the formula:

$$\text{CGPA} = \frac{\sum (S_i \times C_i)}{\sum C_i}$$

Where:

- S_i = SGPA for the i^{th} semester
- C_i = Total number of credits in the i^{th} semester
- $\sum$ (sum over all semesters attended by the student)

Grade Sheet/Report

Based on the grades earned, a grade certificate shall be issued to all the registered students after every semester. The grade certificate will display the course details (code, title, number of credits, grade secured) along with SGPA of that semester and CGPA earned till that semester.

CC :: Core Course/Papers

Level 100-199

- **CSE1104C:** Computer Fundamentals and Applications
- **CSE2104C:** Programming in C and C++

Level 200-299

- **CSE3104C:** Digital Systems and Computer Architecture
- **CSE3204C:** Mathematical Foundations of Computing [*Discrete Maths, TOC*]
- **CSE4104C:** Data Structures and Algorithms I
- **CSE4204C:** Foundations of Computer Networks and Security
- **CSE4304C:** Software and Database Engineering

Level 300-399

- **CSE5104C:** Data Structures and Algorithms II
- **CSE5204C:** Artificial Intelligence
- **CSE5304C:** Operating Systems
- **CSE5404C:** Computational Theory and Application
- **CSE6104C:** Relational and NoSQL Database Systems
- **CSE6204C:** Mathematical and Statistical Foundations for AI/ML
- **CSE6304C:** Machine Learning
- **CSE6404C:** Computer Networking and Cyber Security

Level 400-499 (For CS Honours)

- **CSE7104C:** Deep Learning
- **CSE7204C:** Data Science and Big Data Analytics
- **CSE7304C:** Data Visualization and Predictive Analytics
- **CSE7404C:** Complex Network Analysis
- **CSE8104C:** Advanced Topics on Deep Learning

SEC:: Skill Enhancement Courses

Level 100-199

- **CSE1103SE:** Web Technology (HTML, CSS, JavaScript)
- **CSE2103SE:** System Software and Open Source Systems
- **CSE3103SE:** Programming in Python and R

IDC:: Interdisciplinary Course/Papers [Offered to students of other disciplines]

Level 100-199

- **CSE1103ID:** Introduction to Computer Systems
- **CSE2103ID:** Introduction to Web Technology
- **CSE3103ID:** Introduction to Computer Programming (Python/R)

Minor Courses (CS-Minor) [Offered to students of other disciplines]

Level 100-199

- **CSE1104M:** Introduction to Computer Systems and Applications
- **CSE2104M:** Introduction to C and C++

Level 200-299

- **CSE3104M:** Fundamentals of Web Development Technologies
- **CSE4104M:** Introduction to Computer Programming and Data Structure

Level 300-399

- **CSE5104M:** Fundamentals of Digital Systems and Computer Architecture
- **CSE6104M:** Introduction to Operating Systems

Level 400-499

- **CSE7104M:** Introduction to Database Management System
- **CSE8104M:** Foundations of Computer Networks and Security

CC:: Core Courses/Papers [Additional Core Courses for degree with Computer Science (Honours with Research)]

Level 400-499

- **CSE8204C:** Soft Computing Techniques
- **CSE8304C:** DevOps and Cloud Computing
- **CSE8404C:** Modern Full Stack and Mobile Development

Research Project

- **CSE8512C:** In lieu of CSE8204C: Soft Computing Techniques, CSE8304C: DevOps and Cloud Computing, CSE8404C: Modern Full Stack and Mobile Development, a Research Project of 12 credits has to be chosen for the degree in Computer Science (Honours with Research).

Course Structure for B.Sc (Hons.) in Computer Science with Specialization in AI & Data Science Under FYUGP

First Year Course Structure

Year	Semester	Course	Course Title	Credit
1	1st	Core	CSE1104C: Computer Fundamentals and Applications	4
		Minor	To be opted from other department Like Statistics, Mathematics etc.	4
		AEC	Common course	2
		SEC	CSE1103SE: Web Technology (HTML, CSS, JavaScript)	3
		IDC	To be opted from other departments	3
		VAC	Common course	4
		Internship	NA	
		Total Credit		**20**
	2nd	Core	CSE2104C: Programming in C and C++	4
		Minor	To be opted from other department Like Statistics, Mathematics etc.	4
		AEC	Common course	2
		SEC	CSE2103SE: System Software and Open Source Systems	3
		IDC	To be opted from other departments	3
		VAC	Common course	4
		Internship	NA	
		Total Credit		**20**
For Lateral EXIT students need to complete one VOC course of 4 credits and he/she will be awarded UG certificate.				

Note:

- For computer science students it is suggestive that the other core course should be taken as either Mathematics or Statistics.
- Whatever is the other core course opted in Sem I & Sem II that course is to be carried on subsequent semesters as minor.

Semester I

Core Course/Papers(CC)

CSE1104C: Computer Fundamentals and Applications[4-0-0-4]

Course Description:

This course provides an essential introduction to the core concepts of computing, encompassing both theoretical knowledge and practical skills. It covers the basic architecture and functions of computer systems, including hardware and software components, operating systems, and basic networking concepts. Students will also gain hands-on experience with office productivity tools, such as word processors, spreadsheets, and presentation software. Additionally, the course introduces basic programming principles and internet applications, emphasizing cybersecurity awareness and data privacy.

Learning Outcomes:

Upon successful completion of this course, students will be able to

- Grasp the basic components and functions of computer systems, including hardware, software, and operating systems.
- Understand key concepts in computer networking, including network types, topologies, and basic data communication principles.
- Develop basic programming skills and problem-solving techniques using a high-level programming language.
- Understand the fundamentals of cloud computing, including service models, deployment models, and key concepts like virtualization.
- Recognize common cybersecurity threats and implement basic protective measures, while understanding fundamental concepts of the Internet and web technologies.

Course Content:

Unit 1: Introduction to Computers and Basic Concepts: Overview of Computers- Definition, characteristics, and applications of computers, Types of computers: Analog, Digital, Hybrid, Microcomputers, Mainframes, and Supercomputers. Computer System Components: Hardware: Input devices, output devices, storage devices (HDD, SSD), memory Hierarchy, peripheral devices and processing units. Software: System software, application software, and utility programs. Number Systems and Data Representation: Binary, octal, decimal, and hexadecimal number systems. Representation of data (characters, numbers, images, sound).

Unit 2: Refresh basic knowledge about how computers operate and key components such as the CPU, I/O, memory, etc,. Instruction set architecture (ISA). Control Unit, Arithmetic Logic Unit (ALU), and control flow. Performance factors (clock speed, cache memory). Motherboard components, buses, Types of buses (address bus, data bus, control bus), ports. Software Concepts: Introduction to programming languages (Machine language, Assembly language, High-level languages). Overview of system software: Compilers, assemblers, interpreters, and loaders. Introduction to Networking: Network types (LAN, WAN, MAN, PAN) and topologies (Star, Ring, Bus, Mesh). Basic concepts of the Internet, intranet, and network protocols (TCP/IP, HTTP, FTP). Data Communication: Basic principles of data transmission, modes of transmission (simplex, half-duplex, full-duplex), and transmission media.

Unit 3: Overview of Cloud Computing: Definition, characteristics, and service models (IaaS, PaaS, SaaS). Benefits and Challenges of Cloud Computing: Cost efficiency, scalability, reliability, and security concerns. Cloud Service Providers: Overview of popular cloud platforms (e.g., AWS, Google Cloud, Microsoft Azure). Cloud Deployment Models: Public, private, hybrid, and community clouds. Basics of Cloud Storage: Understanding cloud storage solutions (e.g., Google Drive, Dropbox, Amazon S3). Introduction to Virtualization: Concepts of virtual machines, containers, and serverless computing. Cloud Computing Applications: Use cases in business, education, and personal use. Future Trends in Cloud Computing: Edge computing, AI integration, and blockchain in the cloud.

Unit 4: Basics of Programming and Internet Applications: Introduction to Programming-Concepts of algorithms, flowcharts, and pseudocode. Introduction to programming languages (e.g. C, python). Writing simple programs for arithmetic operations, control structures (if, loop). Fundamental concepts of Arrays and functions. Introduction to search engines, email, social media, and online security. Cybersecurity Fundamentals: Introduction to cybersecurity threats (viruses, malware, phishing). Basic cybersecurity practices (password management, antivirus software). Understanding data privacy and protection laws.

Text Books:

1. Computer Fundamentals by Priti Sinha, Pradeep K., Sinha.
2. Computer Fundamentals by Goel.

Reference Books:

1. Fundamentals of Computers by Rajaraman V and Adabala N.
2. Fundamentals of Computers by Reema Thareja.

Skill Enhancement Courses

CSE1103SE: Web Technology (HTML, CSS, JavaScript)[2-0-2-3]

Course Description

This course equips students with the fundamental skills to create interactive web pages using HTML, CSS, and JavaScript. Students will learn how to structure web content, style it visually, and add dynamic behavior to web applications.

Learning Outcomes

- Understand the core principles of web development and the roles of HTML, CSS, and JavaScript.
- Create well-structured and semantic HTML documents.
- Apply CSS styles to control the visual appearance of web pages.
- Utilize JavaScript to add interactivity and dynamic behavior to web pages.

Course Content

Unit 1: Introduction to Web Development: History of the web, Understanding the client-server model. Different web development roles (front-end, back-end). What is HTML? (HyperText Markup Language). Basic HTML structure (elements, attributes). Creating simple HTML documents. Common HTML elements for structure and content (headings, paragraphs, lists, images, links). Nesting elements and creating complex page layouts. Semantic HTML (using elements for their intended meaning). Adding attributes to elements to provide additional information. Common attributes for various elements (e.g., id, class, src, href). Inline formatting elements (bold, italic, underline). Block-level elements for layout (paragraphs, headings, lists). Hyperlinks (internal and external), Anchors and navigation within web pages.

Unit 2: Styling with CSS: What is CSS? (Cascading Style Sheets). Separation of concerns (HTML for content, CSS for style). Applying basic CSS styles (colors, fonts, margins). Targeting specific elements with CSS selectors (by ID, class, tag name). Combining selectors for more precise styling. How CSS styles are applied and overridden. Understanding inheritance and specificity rules. Common CSS properties for styling text, backgrounds, borders, positioning, etc. Applying appropriate values to achieve desired visual effects.

Unit 3: Advanced CSS and Responsive Design: Understanding the box model for element layout (content, padding, border, margin). Positioning: Static, relative, absolute, and fixed positioning for elements. Advanced layout properties: Display, Float, Clear, and the Z-index. Principles of Responsive Web Design (RWD): making sites work across different devices. Use of media queries for responsive design. Introduction to CSS Flexbox for efficient flexible layouts. Understanding the CSS Grid layout: basic terminology and usage. Building complex layouts with Grid: properties of grid containers and grid items. CSS Transitions and animations for dynamic effects.

Unit 4: Adding Interactivity with JavaScript: What is JavaScript? (Client-side scripting language) Running JavaScript code in web pages, understanding basic syntax and data types Variables, data types (numbers, strings, booleans). Operators (arithmetic, comparison, logical). Control flow statements (if/else, loops). Understanding JS Functions: Definitions, Parameters, Invocation, and Closures. Exploring JS Objects: Properties, Methods, Prototypes, and Object-Oriented Concepts. Arrays and Array Methods: Iteration, Sorting, Searching. Advanced Data Handling: Sets, Maps, and using Spread and Rest operators. DOM Manipulation: Selecting, updating, and managing HTML/CSS with JS. Event Handling: Adding interactivity to web pages. JavaScript BOM (Browser Object Model) Handling: Working with Window, Location, History, Navigator. Introduction to Web APIs: Fetch API for server communication. AJAX Concepts: Making asynchronous requests to update web pages dynamically.

Text Books

1. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 by Robin Nixon.
2. A beginner's guide to HTML, CSS, Javascript, and Web Graphics, by Jennifer Niederst Robbins

References Books

1. Html5: The Missing Manual by Matthew MacDonald.
2. HTML and CSS: Design and Build Websites, by Jon Duckett.
3. Ivan Bayross, *Web Enabled Commercial Application Development Using Html, Dhtml, JavaScript, Perl, Cgi*, BPB Publications, 2009.

Lab Exercises

Unit 1: HTML Fundamentals

Lab Exercises:

1. **Basic HTML Structure:** Create a simple HTML document with headings, paragraphs, lists, images, and links. Practice using semantic HTML tags like `<header>`, `<footer>`, `<article>`, and `<section>`.
2. **Advanced HTML Features:** Practice adding attributes like `id`, `class`, `src`, and `href`. Create a complex page layout using nested elements and various inline and block-level elements.
3. **Navigation and Linking:** Build a multi-page site with internal and external hyperlinks, including anchor tags for in-page navigation.

Unit 2: CSS Styling

Lab Exercises:

1. **CSS Basics:** Apply basic CSS styles to HTML elements including colors, fonts, and margins. Use class and ID selectors to style specific elements.
2. **CSS Selectors and Inheritance:** Combine multiple selectors for precise styling. Understand and demonstrate the concept of CSS inheritance and specificity.
3. **Styling with CSS Properties:** Use CSS to style text, backgrounds, borders, and positioning. Create exercises that require students to use CSS for real-world webpage layouts.

Unit 3: Advanced CSS and Responsive Design

Lab Exercises:

1. **Box Model and Positioning:** Explore the CSS box model in detail. Practice with different positioning schemes: static, relative, absolute, and fixed.
2. **Responsive Web Design:** Implement responsive design using media queries. Convert a fixed layout into a responsive layout that adapts to various screen sizes.
3. **Flexbox and Grid:** Use CSS Flexbox and Grid systems to build complex layouts. Create layouts that require nested flex containers or grid-based designs.
4. **CSS Transitions and Animations:** Create interactive visual effects using CSS transitions and animations. Design exercises that enhance user experience through dynamic CSS.

Unit 4: JavaScript Interactivity

Lab Exercises:

1. **JavaScript Basics:** Write scripts that utilize variables, data types, operators, and control flow (if/else, loops). Implement functions with parameters and return values.
2. **DOM Manipulation and Event Handling:** Practice selecting, updating, and managing HTML elements with JavaScript. Add event listeners to HTML elements for interactive responses.
3. **Advanced JavaScript Concepts:** Explore arrays, object properties, methods, prototypes, and object-oriented concepts. Use advanced data handling with Sets, Maps, Spread, and Rest operators.
4. **Asynchronous JavaScript:** Make asynchronous requests using the Fetch API and AJAX. Create exercises that involve fetching data from a server and dynamically updating the webpage.

Project-Based Learning

S. No	Project	Objective	Skills
1	Basic Personal Blog	Build a simple personal blog with posts and a homepage.	Basic HTML, CSS, minimal JavaScript
2	Simple Recipe Website	Create a website displaying recipes with ingredients and instructions.	HTML, CSS
3	Community Event Calendar	Develop a calendar listing local events with clickable details.	HTML, CSS, basic JavaScript
4	Portfolio Showcase	Construct a portfolio site with projects, an about page, and a contact form.	Semantic HTML, CSS Flexbox, HTML form handling
5	Fitness Tracker	Design a tracker to log activities and display weekly totals.	HTML forms, CSS, basic JavaScript
6	Restaurant Menu Display	Build a website displaying a restaurant menu categorized by course.	HTML, CSS
7	Local Library Catalog	Create a library catalog with books categorized by genre.	HTML, CSS, basic JavaScript
8	Simple Contact Directory	Develop a contact directory with clickable details.	Basic HTML, CSS, JavaScript
9	One-Page Restaurant Website	Create a one-page restaurant website with sections and navigation.	HTML, CSS, anchor links
10	Digital Clock	Create a digital clock that displays the current time in real-time.	DOM manipulation, JavaScript Date object
11	Pulse Search Bar	Design a visually appealing search bar with a pulsing animation.	CSS animations
12	Simple Calculator	Build a basic calculator that performs arithmetic operations.	JavaScript functions, event handling

S. No	Project	Objective	Skills
13	Dark/Light Mode Switch	Implement a toggle for switching between dark and light themes.	CSS variables, JavaScript event handling
14	Drag and Drop File Upload	Create a drag-and-drop interface for file uploads.	File handling, JavaScript events
15	Real-Time Weather App	Develop a weather app fetching data from an external API.	API integration, data manipulation
16	Product Showcase Carousel	Build an interactive carousel for showcasing products.	Dynamic content rendering, JavaScript
17	Interactive Quiz App	Create a quiz app with timed questions and feedback.	State management, JavaScript
18	To-Do List Application	Develop a to-do list with task management features.	User input handling, local storage

Interdisciplinary Courses (IDC)

CSE1103ID: Introduction to Computer Systems[3-0-0-3]

Course Description

This course provides a comprehensive foundation for understanding the inner workings of computer systems. Students will delve into the hardware components, software layers, and essential concepts that make computers function.

Learning Outcomes

- Understand the fundamental building blocks of a computer system (hardware and software).
- Explain the interaction between hardware components and software programs.
- Gain a grasp of computer architecture concepts like instruction sets, memory hierarchy, and CPU performance factors.
- Analyze the core functionalities of operating systems and their role in resource management.
- Comprehend data representation and manipulation within computer systems.
- Explore basic computer networking principles and communication protocols.

Course Content

Unit 1: Introduction to Computer Systems: Overview of computer systems (hardware, software, and their interaction). Historical development of computers. Classification of computer systems (personal computers, servers, embedded systems). Von Neumann Architecture (basic principles).

Unit 2: Hardware Fundamentals: Central Processing Unit (CPU). Instruction set architecture (ISA). Control Unit, Arithmetic Logic Unit (ALU), and control flow. Performance factors (clock speed, cache memory). Memory Hierarchy: Primary memory (RAM) and secondary storage (hard disk drives, SSDs). Cache memory and its role in performance. Input/Output (I/O) devices and communication protocols. Buses: Data transfer pathways within a computer system. Types of buses (address bus, data bus, control bus).

Unit 3: Operating Systems: Introduction to operating systems (OS). Process management (scheduling, synchronization). Memory management (virtual memory). Device management (I/O handling). File systems for data organization. Types of Operating Systems (Desktop OS, Mobile OS, Embedded OS). User interface (UI) and user interaction with the OS.

Unit 4: Data Representation and Manipulation: Number systems (binary, decimal, hexadecimal). Data types (integers, floating-point numbers, characters). Boolean algebra and logic gates. Computer arithmetic (addition, subtraction, multiplication, division). Instruction encoding and decoding (basic concepts).

Text Books:

1. Fundamentals of Computers by E Balagurusamy.

References Books:

1. Computer Fundamentals by Priti Sinha, Pradeep K. Sinha
2. Computer Systems: A Programmer's Perspective by Randal E. Bryant, David R. O'Hallaron

Minor Courses (CS-Minor)

CSE1104M: Introduction to Computer Systems and Applications[4-0-0-4]

Course Description:

This course provides an essential introduction to the core concepts of computing, encompassing both theoretical knowledge and practical skills. It covers the basic architecture and functions of computer systems, including hardware and software components, operating systems, and basic networking concepts. Students will also gain hands-on experience with office productivity tools, such as word processors, spreadsheets, and presentation software. Additionally, the course introduces basic programming principles and internet applications, emphasizing cybersecurity awareness and data privacy.

Learning Outcomes:

Upon successful completion of this course, students will be able to

- Grasp the basic components and functions of computer systems, including hardware, software, and operating systems.
- Demonstrate the ability to use word processors, spreadsheets, and presentation software effectively for various applications.
- Understand key concepts in computer networking, including network types, topologies, and basic data communication principles.
- Develop basic programming skills and problem-solving techniques using a high-level programming language.
- Recognize common cybersecurity threats and implement basic protective measures, while understanding fundamental concepts of the Internet and web technologies.

Course Content:

Unit 1: Introduction to Computers and Basic Concepts: Definition, characteristics, and common applications of computers. Types of computers: Basic classification (Analog, Digital, Hybrid, Microcomputers, Mainframes, Supercomputers). Overview of system software (operating systems), application software, and utility programs. Introduction to Binary, Octal, Decimal, and Hexadecimal number systems. Basic representation of data (characters, numbers). Basic functions of an operating system (OS). Brief overview of Batch, Multitasking, Multiprocessing, and Real-time systems.

Unit 2: Basics of Computer Hardware and Software: Basic components including CPU, memory (RAM, ROM), storage devices, and peripheral devices. Instruction set architecture (ISA). Control Unit, Arithmetic Logic Unit (ALU), and control flow. Performance factors (clock speed, cache memory). Memory Hierarchy, Cache memory and its role in performance. Introduction to motherboard components and ports. Introduction to programming languages (Machine language, Assembly language, High-level languages). Overview of system software, including compilers and interpreters. Basic understanding of network types (LAN, WAN) and basic network protocols (TCP/IP).

Unit 3: Office Productivity Tools and Applications: Basic functionalities of word processors (e.g., Microsoft Word, Google Docs) for document creation and formatting. Introduction to spreadsheet software (e.g., Microsoft Excel, Google Sheets) for data entry and basic calculations. Creating basic presentations using tools like Microsoft PowerPoint and Google Slides.

Unit 4: Basics of Programming and Internet Applications: Simplified introduction to algorithms, flowcharts, and basic programming constructs using a beginner-friendly language like Python. Basic overview of internet usage, including search engines, email, and social media. Basic introduction to common cybersecurity threats (viruses, malware) and simple security practices.

Text Books:

1. Computer Fundamentals by Priti Sinha, Pradeep K., Sinha.
2. Computer Fundamentals by Goel.

Reference Books:

1. Fundamentals of Computers by Rajaraman V and Adabala N.
2. Fundamentals of Computers by Reema Thareja.

Semester II

Core Course/Papers(CC)

CSE2104C: Programming in C and C++[3-0-2-4]

Course Description

This course introduces the fundamental concepts of programming using C/C++. Students will gain hands-on experience in writing, compiling, debugging, and executing C/C++ programs. The course covers core programming concepts, data types, control flow, functions, arrays, pointers, and object-oriented programming basics in C++.

Course Learning Outcomes

By the end of this course students will be able to:

- Explain the fundamental concepts of computer programming, including the problem-solving process, algorithms, and data structures.
- Write, compile, debug, and execute basic C/C++ programs using a development environment.
- Work with arrays to store and manipulate collections of data elements.
- Grasp the concept of pointers for memory management and advanced data manipulation in C.
- Demonstrate the principles of object-oriented programming (OOP) in C++ using classes and objects.
- Apply problem-solving techniques to design, develop, and test C/C++ programs for solving various problems.

Course Content

Unit 1: History of C/C++: Overview of Procedural and Object-Oriented Programming. Basic Programming Elements: Using main() function, compiling, and executing simple C programs. Data Types and Variables: Declaring, defining, and initializing variables, scope, and constants. Operators and I/O: Arithmetic, logical, bitwise operators, comments, character I/O (getc, getchar, putc, putchar), formatted I/O (printf(), scanf(), cin, cout). Basic Header Files: Usage of stdio.h, iostream.h, conio.h. Expressions and Control Flow: Simple expressions, operator precedence, conditional statements (if, switch-case), iterative statements (while, do-while, for loops), and nested statements.

Unit 2: Utility, call by value/reference, functions returning values, void functions, inline functions, declaration vs. definition, command-line arguments, variable number of arguments. Arrays: One-dimensional arrays (declaring, defining, initializing, accessing, and manipulating elements), types of arrays (integer, float, character/strings), two-dimensional arrays, multi-dimensional arrays. Structures and Unions: Declaring, initializing, and using structures and unions, array of structures, passing and returning structures from functions.

Unit 3: Pointers: Understanding and using pointers, pointers to pointers, pointers to structures, passing pointers to functions, returning pointers from functions, arrays as pointers, and pointers vs. references. Memory Allocation: Static vs. dynamic memory allocation, usage of malloc, calloc, free, new, and delete operators. File I/O and Preprocessor Directives: Opening/closing files (using fstream, ifstream, ofstream classes), reading/writing text files (put(), get(), read(), write()), random file access, preprocessor directives (#include, #define, #error, #if, #else, #elif, #endif, #ifdef, #ifndef, #undef), and macros.

Unit 4: Principles of Object-Oriented Programming, Defining & Using Classes, Class Constructors, Function overloading in classes. Overview of Function Overloading and Operator Overloading, Inheritance, Polymorphism, and Exception Handling. Problem solving, Introduction, Problem solving techniques, Algorithms. Debugging techniques, Testing and code verification.

Lab Practice

- Implement fundamental domain knowledge of the course for developing effective computing solutions by incorporating creativity and logical reasoning.
- Understand and learn how a big program can be broken up into independent modules and define functions and call them with appropriate parameters.
- Learn the use of programming language for solving real-world problems on a computer.

Text Books

1. *The C Programming language* is written by Brian W.Kernigham and Dennis M.Ritchie.
2. *Programming in ANSI C* written by E Balagurusamy.
3. E Balaguruswamy, *Object Oriented Programming with C++*, Tata McGraw-Hill Education, 2008.

Reference Books

1. *Let Us C* written by Yashavant Kanetkar.
2. *Head First C* is written by David Griffiths.
3. *Programming: Principles and Practice Using C++* by Bjarne Stroustroup
4. *C++ Primer* (5th Edition) By Stanley B. Lippman, Josée Lajoie, and Barbara E. Moo

Skill Enhancement Courses

CSE2103SE: System Software and Open Source Systems[2-0-2-3]

Course Description:

This course offers a foundational understanding of system software and open-source systems explores the components and functions of system software, with a focus on operating systems, system calls, and open-source operating systems, particularly Linux. The course integrates theoretical knowledge with practical skills, emphasizing the role of open-source communities and the significance of security in system software.

Learning Outcomes:

By the end of this course students will be able to:

- Acquire a solid understanding of system software, including the role of operating systems and system calls in managing hardware and software resources.
- Gain practical experience in using Linux-based operating systems, including installation, configuration, and system administration.
- Understand the principles and practices of open-source systems, including licensing and community contributions.
- Develop skills in system programming using system calls and automating tasks with shell scripts.
- Learn and apply basic security practices in managing Linux systems and open-source software

Course Content:

Unit 1: Operating Systems Fundamentals: Introduction to operating systems and their roles in computer systems. Types of operating systems: Batch, time-sharing, distributed, real-time, and embedded systems. Process management: Concepts of processes, threads, and multitasking. Fundamental concepts of memory management: Paging, segmentation, virtual memory, and memory allocation strategies. System Calls: Definition and importance of system calls as the interface between user programs and the operating system. Categories of system calls: Process control, file management, device management, information maintenance, and communication. Examples and usage of system calls in Unix/Linux systems, such as fork(), exec(), read(), write(), open(), and close(). Process and Thread Management: CPU scheduling algorithms. Fundamentals of process synchronization: Critical sections, semaphores, and deadlock.

Unit 2: Overview of System Software: Definition, categories, and significance of system software. Relationship between system software and operating systems. Assemblers and Compilers: Basic concepts and functionalities of assemblers, assembly language, and machine language. Differences between interpreters and compilers, and their respective use cases. Linkers and Loaders: Functions and types of linkers and loaders. Static and dynamic linking, and the process of loading programs into memory.

Unit 3: Linux and Open Source Operating Systems: Overview of Linux as a Unix-like, open-source operating system. Key features of Linux, and an introduction to popular Linux distributions (Ubuntu, Fedora, CentOS, etc.). Linux Kernel and Architecture: Structure of the Linux kernel: Monolithic architecture, kernel modules, and device drivers. Filesystem hierarchy and management: Understanding Ext4, XFS, Btrfs, and others. Linux System Calls and Programming: Exploration of Linux-specific system calls and their practical applications. Basic shell scripting. Text editor (e.g., Vim, Nano, or Emacs). Different types of shells (bash, zsh, etc.). Declaring and assigning variables, Using variables in expressions, Environment variables. Strings, Numbers, Arithmetic operators, Comparison operators, Logical operators, Conditional statements (if-else, case) Loops (for, while, until). Navigation, file manipulation, and directory management.Defining and calling functions, Passing arguments to functions, Creating, opening, and closing files Reading from and writing to files.
Unit 4: Practical Applications and Security in Open Source Systems: Installation procedures and configuration of Linux systems. System administration tasks: User management, file permissions, package management, and system updates. Shell Scripting and Automation: Fundamentals of shell scripting in Bash: Variables, control structures, loops, and functions. Automating administrative tasks using shell scripts.

Text Books:

1. Operating System Concepts by Abraham Silberschatz, Peter B. Galvin, Greg Gagne.
2. System Software An Introduction To Systems Programming By Leland L. Beck
3. UNIX : Concepts and Applications by Sumitabha Das
4. UNIX and Linux System Administration Handbook by Evi Nemeth, Garth Snyder, Trent Hein, Ben Whaley, Dan Mackin.

Reference Books:

1. Modern Operating Systems by Tanenbaum Andrew S.
2. Systems programming and operating systems by Dhamdhere, D. M
3. The Linux Command Line – A Complete Introduction by Williams E. Shotts
4. LINUX BIBLE by Christopher Negus

Interdisciplinary Courses (IDC)

CSE2103ID: Introduction to Web Technology[2-0-2-3]

Course Description

This course equips students with the fundamental skills to create interactive web pages using HTML, CSS, and JavaScript. Students will learn how to structure web content, style it visually, and add dynamic behavior to web applications.

Learning Outcomes

- Understand the core principles of web development and the roles of HTML, CSS, and JavaScript.
- Create well-structured and semantic HTML documents.
- Apply CSS styles to control the visual appearance of web pages.
- Utilize JavaScript to add interactivity and dynamic behavior to web pages.

Course Content

Unit 1: Introduction to Web Development. Brief history and evolution of the web. Understanding the client-server model and the roles in web development (front-end, back-end). Basics of HTML (HyperText Markup Language): Introduction to HTML and its purpose. Basic HTML structure: Elements and attributes. Creating simple HTML documents. Common HTML elements: Headings, paragraphs, lists, images, and links. Introduction to semantic HTML and basic layout.

Unit 2: Introduction to CSS (Cascading Style Sheets) Purpose and Basics of CSS. What is CSS and why it's used. Basic CSS syntax: Selectors, properties, and values. Applying styles to HTML elements (colors, fonts, and margins). Basic CSS selectors and simple styling techniques. Styling and Layout:Introduction to the box model (content, padding, border, margin). Simple layout techniques: Block vs. inline elements. Basic text and background styling.

Unit 3: Responsive Design and Advanced CSS Responsive Web Design:Importance of making websites work on different devices. Introduction to media queries for responsive design. Basics of CSS Flexbox for layout adjustments. Advanced CSS Concepts:Basic understanding of CSS Grid layout. Simple transitions and animations for visual effects.

Unit 4: Basic JavaScript for Interactivity Introduction to JavaScript:What is JavaScript and its role in web development. Basic syntax and data types (numbers, strings, booleans). Simple control flow (if/else statements, loops). Basic JavaScript Functions and DOM Manipulation: Introduction to functions and their use. Basic DOM manipulation: Selecting and modifying HTML elements. Simple event handling: Adding basic interactivity.

Text Books

1. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 by Robin Nixon.

2. A beginner's guide to HTML, CSS, Javascript, and Web Graphics, by Jennifer Niederst Robbins

References Books

1. Html5: The Missing Manual by Matthew MacDonald.

2. HTML and CSS: Design and Build Websites, by Jon Duckett.

Minor Courses (CS-Minor)

CSE2104M: Introduction to C and C++[3-0-2-4]

Course Description

This course introduces the fundamental concepts of programming using C/C++. Students will gain hands-on experience in writing, compiling, debugging, and executing C/C++ programs. The course covers core programming concepts, data types, control flow, functions, arrays, pointers, and object-oriented programming basics in C++.

Course Learning Outcomes

By the end of this course students will be able to:

- Explain the fundamental concepts of computer programming, including the problem-solving process, algorithms, and data structures.
- Write, compile, debug, and execute basic C/C++ programs using a development environment.
- Work with arrays to store and manipulate collections of data elements.
- Grasp the concept of pointers for memory management and advanced data manipulation in C.
- Demonstrate the principles of object-oriented programming (OOP) in C++ using classes and objects.
- Apply problem-solving techniques to design, develop, and test C/C++ programs for solving various problems.
- Use programming language for solving real-world problems on a computer.

Course Content

Unit 1: Brief history and key differences between Procedural and Object-Oriented Programming. Introduction to compiling and running simple C programs. Basic Programming Elements: Main Function: Understanding the main() function. Data Types and Variables: Declaring and initializing basic data types and variables, understanding scope and constants. Basic Operators: Arithmetic, logical, and basic bitwise operators. Simple character I/O (getc, getchar, putc, putchar) and formatted I/O (printf(), scanf()). Control Flow: Conditional Statements: Basic usage of if and switch-case statements. Loops: Simple loops using while, do-while, and for loops.

Unit 2: Utility, call by value/reference, functions returning values, void functions, inline functions, declaration vs. definition, command-line arguments, variable number of arguments. Arrays: One-dimensional arrays (declaring, defining, initializing, accessing, and manipulating elements), types of arrays (integer, float, character/strings), two-dimensional arrays, multi-dimensional arrays. Structures and Unions: Declaring, initializing, and using structures and unions, array of structures, passing and returning structures from functions.

Unit 3: Pointers: Understanding and using pointers, pointers to pointers, pointers to structures, passing pointers to functions, returning pointers from functions, arrays as pointers, and pointers vs. references. Memory Allocation: Static vs. dynamic memory allocation, usage of malloc, calloc, free, new, and delete operators. Basic File Operations: Opening, reading, writing, and closing text files.

Unit 4: Understanding the principles of Object-Oriented Programming. Classes and Objects: Basic definition and usage of classes. Simple OOP Features: Class Constructors: Introduction to constructors in classes. Function Overloading: Basic understanding of function overloading in C++.

Lab Practice

- Implement fundamental domain knowledge of the course for developing effective computing solutions by incorporating creativity and logical reasoning.
- Understand and learn how a big program can be broken up into independent modules and define functions and call them with appropriate parameters.
- Learn the use of programming language for solving real-world problems on a computer.

Text Books

1. *The C Programming language* is written by Brian W.Kernigham and Dennis M.Ritchie.
2. *Programming in ANSI C* written by E Balagurusamy.
3. E Balaguruswamy, *Object Oriented Programming with C++*, Tata McGraw-Hill Education, 2008.

Reference Books

1. *Let Us C* written by Yashavant Kanetkar.
2. *Head First C* is written by David Griffiths.
3. *Programming: Principles and Practice Using C++* by Bjarne Stroustroup
4. *C++ Primer* (5th Edition) By Stanley B. Lippman, Josée Lajoie, and Barbara E. Moo

Second Year Course Structure

Year	Semester	Course	Course Title	Credit
2	3rd	Core	CSE3104C: Digital Systems and Computer Architecture.	4
		Core	CSE3204C: Mathematical Foundations of Computing	4
		Minor	To be opted from other departments. (*For double major +1=2*)	4(*+4*)
		AEC	Common course	2
		SEC	CSE3103SE: Programming in Python and R	3
		IDC	To be opted from other departments	3
		VAC	NA	
		Internship	NA	
		Total Credit		**20(*+4*)**
	4th	Core	CSE4104C: Data Structures and Algorithms I	4
		Core	CSE4204C: Foundations of Computer Networks and Security	4
		Core	CSE4304C: Software and Database Engineering	4
		Minor	To be opted from other department. (*For double major +1=2*)	4(*+4*)
		AEC	Common course	2
		SEC	NA	
		IDC	NA	
		VAC	NA	
		Internship	Has to be engaged in a summer internship	2
		Total Credit		**20(*+4*)**
For Lateral EXIT students need to complete one VOC course of 4 credits and he/she will be awarded UG Diploma.				

Third Year Course Structure

Year	Semester	Course	Course Title	Credit
3	5th	Core	CSE5104C: Data Structures and Algorithms II.	4
		Core	CSE5204C: Artificial Intelligence	4
		Core	CSE5304C: Operating Systems	4
		Core	CSE5404C: Computational Theory and Application	4
		Minor	To be opted from other departments. (*For double major +2=3*)	4(+8)
		AEC	NA	
		SEC	NA	
		IDC	NA	
		VAC	NA	
		Internship	NA	
			Total Credit	**20(+8)**
	6th	Core	CSE6104C: Relational and NoSQL Database Systems.	4
		Core	CSE6204C: Mathematical and Statistical Foundations for AI/ML	4
		Core	CSE6304C: Machine Learning	4
		Core	CSE6404C: Computer Networking and Cyber Security	4
		Minor	To be opted from other departments. (*For double major +2=3*)	4(+8)
		AEC	NA	
		SEC	NA	
		IDC	NA	
		VAC	NA	
		Internship	NA	
			Total Credit	**20(+8)**
Total Credit after 3 Years: 120(+24) After 3 years student will be awarded UG degree with one major and one minor (Total Credit 120). For double major (Total Credit 144).				

Fourth Year Course Structure

Year	Semester	Course	Course Title	Credit
4	7th	Core	CSE7104C: Deep Learning	4
		Core	CSE7204C: Data Science and Big Data Analytics	4
		Core	CSE7304C: Data Visualization and Predictive Analytics.	4
		Core	CSE7404C: Complex Network Analysis	4
		Minor	To be opted from other departments. (*For double major +1=2*)	4(*+4*)
		AEC	NA	
		SEC	NA	
		IDC	NA	
		VAC	NA	
		Internship	NA	
		Total Credit		**20(*+4*)**
	8th	Core	CSE8104C: Advanced Topics on Deep Learning	4
		Minor	To be opted from other departments. (*For double major +1=2*)	4(*+4*)
		AEC	NA	
		SEC	NA	
		IDC	NA	
		VAC	NA	
		Internship	NA	
		Total Credit		**20(*+4*)**

Total Credit after 4 Years: 160(*+32*)
After 4 years student will be awarded UG degree (Honours/Honours with research) Total Credit 160.
After 4 years student will be awarded UG degree (Honours/Honours with research) with double major (Total credit 192).