NATIONAL INSTITUTE OF ELECTRONICS AND INFORMATION TECHNOLOGY

DEEMED TO BE UNIVERSITY UNDER DISTINCT CATEGORY

(AN AUTONOMOUS INSTITUTION UNDER MINISTRY OF ELECTRONICS & IT, GOVT. OF INDIA)

Curriculum and Syllabi for

M.Sc. in Bioinformatics

(2026-27 Scheme)

रा.इ.सू.प्रौ.सं

NIELIT

Deemed to be University

Main Campus at Ropar and Constituent Units at Aizawl, Agartala, Aurangabad, Calicut, Gorakhpur, Imphal, Itanagar, Kekri, Kohima, Patna and Srinagar

Vision

To be a globally recognized center of excellence in Bioinformatics education and research, fostering interdisciplinary expertise at the intersection of biology, computer science, and statistics, to address complex challenges in life sciences and contribute to advancements in healthcare, agriculture, and environmental sustainability.

Mission

To provide a rigorous and interdisciplinary curriculum that equips students with advanced knowledge in molecular biology, genomics, proteomics, and computational methods.

To cultivate strong analytical, programming, and statistical skills for the effective analysis, interpretation, and visualization of large-scale biological data.

To foster innovative research capabilities and problem-solving approaches for real-world applications in drug discovery, personalized medicine, and systems biology.

To instill ethical considerations, professional responsibility, and a commitment to lifelong learning in the rapidly evolving field of bioinformatics.

To prepare graduates for successful careers in academia, industry, and research organizations, contributing to scientific discovery and technological innovation.

Program Education Objectives

Graduates of the M.Sc. in Bioinformatics program will be able to:

PEO1: Interdisciplinary Expertise: Integrate advanced knowledge from molecular biology, computer science, and statistics to comprehend and address complex biological problems.

PEO2: Data Analysis & Interpretation: Apply computational and statistical tools effectively to analyze, interpret, and visualize large-scale biological data (genomic, transcriptomic, proteomic, etc.).

PEO3: Research & Innovation: Design and execute bioinformatics research projects, developing novel algorithms, software, or methodologies to contribute to scientific discovery and technological advancement.

PEO4: Professional & Ethical Conduct: Demonstrate professional integrity, ethical conduct, and an understanding of the societal implications in the application of bioinformatics solutions.

PEO5: Adaptability & Leadership: Adapt to emerging technologies and scientific paradigms in bioinformatics, pursuing lifelong learning and assuming leadership roles in diverse professional environments.

Program Outcomes

Upon successful completion of the M.Sc. in Bioinformatics program, students will be able to:

PO1: Bioinformatics Knowledge: Demonstrate a comprehensive understanding of core concepts in molecular biology, genomics, proteomics, structural biology, and their computational underpinnings.

PO2: Problem Analysis: Identify, formulate, and analyze complex biological problems that require computational solutions, drawing upon principles of biology, computer science, and statistics.

PO3: Computational Design & Development: Design, develop, and implement bioinformatics algorithms, software tools, and pipelines for the analysis and management of biological data.

PO4: Data Handling & Management: Apply advanced programming skills (e.g., Python, R) and utilize specialized

bioinformatics tools and databases for data acquisition, processing, and storage.

PO5: Statistical & Algorithmic Application: Apply appropriate statistical methods, machine learning algorithms, and high-performance computing techniques to extract meaningful insights from biological datasets.

PO6: Research Methodology: Conduct independent research, critically evaluate scientific literature, and contribute to the body of knowledge in bioinformatics through original work.

PO7: Communication: Communicate complex scientific and technical information effectively, both orally and in writing, to diverse audiences.

PO8: Collaboration & Teamwork: Work effectively as an individual and as part of interdisciplinary teams to solve complex problems in bioinformatics.

PO9: Ethics & Social Responsibility: Understand and adhere to ethical principles, data privacy regulations, and responsible conduct in research and application of bioinformatics.

PO10: Lifelong Learning: Recognize the need for, and engage in, continuous professional development and self-directed learning to stay current with advancements in bioinformatics and related fields.

Program Specific Outcomes

Upon completion of the M.Sc. in Bioinformatics program, students will be able to:

PSO1: Omics Data Analysis: Proficiently apply computational tools and statistical methods to analyze and interpret large-scale omics data (genomics, transcriptomics, proteomics) for gene discovery, biomarker identification, and disease understanding.

PSO2: Structural & Drug Design Applications: Utilize structural bioinformatics techniques and molecular modeling software to analyze biomolecular structures, predict protein function, and contribute to computer-aided drug discovery and design.

PSO3: Systems Biology & Network Analysis: Model and analyze complex biological systems using network-based approaches and computational simulations to understand emergent properties and identify key regulatory mechanisms.

CATEGORY WISE CREDIT DISTRIBUTION

Category	Category Code	Credits
Skill / Employment Enhancement Courses (Project/Internship/Seminar/Dissertation/Research)	EE	36
Audit Courses (without grade or credit)	AU	0
Open Elective Courses	OE	8
Professional Elective Courses	PE	16
Program Core Courses	PC	20
Total Credits (Common track)		**80**

EVALUATION AND ASSESSMENT

1. Performance of a student in a semester shall be evaluated through continuous Class assessment, Tutorial/Lab assessment, Mid-Semester Examination (MSE) and End-Semester Examination (ESE). Both the MSE and ESE shall be the University examination and will be conducted as notified by the Controller of Examinations (CoE) of the University.

2. The continuous assessment shall be based on assignments, tutorials, paper presentation / quizzes/ viva-voce / flipped classes, lab work / projects / fieldwork and attendance, etc.

3. The MSE/ESE shall be comprising of written papers.

4. The overall assessment of the students will be done as per the following scheme:

S. No.	Assessment Type	Marks Weightage
1.	Mid Semester Examination	20
2.	End Semester Examination	40
3.	Continuous Assessment - Practical /Lab/ Tutorial	25
4.	Continuous Assessment - Theory	15
Total		**100**

For more details, please refer the applicable ordinance for this programme and Assessment SOPs.

CURRICULUM

Semester 1						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
PC-MSCBIO-101	DASH250018	Introduction to Molecular Biology and Bioinformatics	3	1	0	4
PC-MSCBIO-102	DASH250022	Genomics, Transcriptomics and Proteomics	3	0	2	4
PE-MSCBIO-103	Elective	Program Elective				4
OE-MSCBIO-104	Elective	Open Elective				4
PC-MSCBIO-105	DCSE250080	Structural Bioinformatics	3	1	0	4
AU-MSCBIO-106	Elective	Audit Elective				0

Semester 2						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
PC-MSCBIO-201	DCSE250089	Linux and Python for Biological Data Analysis	3	1	0	4
PC-MSCBIO-202	DASH250043	Statistical Methods in Computational Biology and Problem-Solving using R	2	0	4	4
PE-MSCBIO-203	Elective	Program Elective				4
PE-MSCBIO-204	Elective	Program Elective				4
AU-MSCBIO-205	Elective	Audit Elective				0
EE-MSCBIO-206	EE	Mini Project with Seminar				4

Semester 3						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
PE-MSCBIO-301	Elective	Program Elective				4
OE-MSCBIO-302	Elective	Open Elective				4
EE-MSCBIO-303	EE	Dissertation-I/ Industrial project				12

Semester 4						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
EE-MSCBIO-401	EE	Dissertation-II				20

Elective Courses

Elective Courses for PE Category						
Course Code	**Course Title**	**L**	**T**	**P**	**Cr**	**For Sem./Code**
DOAI260001	AI (Industry)-Applied Ethics for AI	3	0	2	4	
DOAI260002	AI (Industry)-Introduction to Artificial Intelligence	3	0	2	4	
DOAI260003	AI (Industry)-Introduction to Generative AI	3	0	2	4	
DOAI260004	AI (Industry)-Introduction to Machine Learning	3	0	2	4	
DASH250045	Drug Discovery and Molecular Modeling	3	0	2	4	
DASH250051	Systems Biology and Network Analysis	3	1	0	4	
DCSE250041	Fundamentals of Database Systems	3	0	2	4	
DCSE250056	Data Structures and Algorithms	3	0	2	4	
DCSE250079	Fundamentals of Programming in Python	2	0	4	4	
DCSE250086	Health Care and Data Analytics	3	1	0	4	
DCSE250102	Operating Systems	3	0	2	4	
DOAI250005	Artificial Intelligence and Machine Learning	2	0	4	4	
DOAI250009	Big Data Analytics	3	0	2	4	
DOAI250031	Introduction to High Performance Computing	3	1	0	4	

Elective Courses for OE Category
Students may opt courses under MOOC, NPTEL, SWAYAM, NSQF/NCVET aligned courses or other relevant technical subjects not included in their core curriculum. The exercise of option shall be subject to the Course Schedule of the respective Constituent Unit, the credit requirements and availability of seats. Guidelines for choosing MOOC/Online courses are given separately and shall have to be adhered to.

Elective Courses for AU (Audit) Category					
Course Code	**Course Title**	**L**	**T**	**P**	**Cr**
DASH240027	Disaster Management	2	0	0	0
DASH240047	English for Research Paper Writing	2	0	0	0
DASH240052	Environmental Science	3	0	0	0
DASH240085	Pedagogy Studies	2	0	0	0
DASH240086	Personality Development through Life Enlightenment Skills	2	0	0	0
DASH240097	Sanskrit for Technical Knowledge	2	0	0	0
DASH240103	Stress Management by Yoga	2	0	0	0
DASH240104	Technical Presentation and Report Writing	0	0	2	0
DASH240106	Value Education	2	0	0	0
DASH240124	Constitution of India - AU	2	0	0	0
DASH250023	Constitution of India	2	1	0	0
DASH250027	Disaster Management	3	0	0	0
DASH250034	Energy and Environmental Engineering	3	0	0	0
DASH250047	English for Research Paper Writing	2	0	0	0

DASH250054	Essence of Indian Knowledge and Tradition	2	0	0	0
DASH250085	Pedagogy Studies	2	0	0	0
DASH250086	Personality Development	2	0	0	0
DASH250097	Sanskrit for Technical Knowledge	3	0	0	0
DASH250103	Stress Management by Yoga	3	0	0	0
DASH250106	Value Education	2	0	0	0
DASH250117	Innovative Thinking and Entrepreneurship Skills	1	0	0	0
DASH250118	Intellectual Property Rights	2	0	0	0
DCSA240080	Training on Computer Networking	0	0	2	0
DCSA240304	Lab Based on Statistical Package	0	0	2	0
Any other Scheduled Course as per the Course Schedule of the respective Constituent Unit can also be chosen, subject to availability of seats. However, there shall be no assessment and grading for the course chosen as Audit Course.					

The choice of all type of Electives available shall be as per the Course Schedule of the respective Constituent Unit.

Guidelines for Massive Open Online Courses (MOOC) / approved Online Courses

1. Relevant Swayam, MOOC, NPTEL, NIELIT NSQF and any other online courses approved by UGC/AICTE shall also be allowed as Open Electives(OE).

2. One credit of MOOC course should be minimum of 30 hours.

3. A student can choose any approved online course as Open Elective, subject to minimum credit requirements.

4. The students willing to opt OE under MOOC in their next semester, will submit their request to the MOOC Coordinator at their respective campus.

5. Once approved, the campus shall display the list of accepted courses.

6. The student has to submit certificate issued by the institution offering the MOOCs course, along with the number of credits and grades, to get credits transferred into his/her marks certificate issued by NDU.

7. The transfer of credits shall be as adopted by NDU.

Detailed Syllabus

Course Code	Course Name	L	T	P	Cr
DASH250018	Introduction to Molecular Biology and Bioinformatics	3	1	0	4

Course Outcomes:

CO1: Provide a foundational understanding of molecular and cellular biology concepts including biomolecules, gene expression mechanisms, cellular processes, and metabolic regulation.
CO2: Introduce students to core bioinformatics principles, databases, algorithms, and computational tools essential for the analysis and interpretation of biological data.
CO3: Integrate molecular biology with computational approaches to explore biological systems using omics technologies, sequence analysis, and structural bioinformatics.
CO4: Develop skills in handling biological data through database usage, sequence analysis, and molecular modeling tools.
CO5: Apply systems biology and bioinformatics methods for understanding genetic regulation, protein interactions, and metabolic pathways.

Module 1:

Cellular and Molecular Foundations: Prokaryotic vs eukaryotic cells: Structural differences and similarities, Evolutionary implications (endosymbiotic theory), Comparative genomics and minimal genome concepts, Computational tools for visualizing and annotating cell structures; Organelles and cellular compartmentalization: Structure and function of nucleus, ER, Golgi, mitochondria, chloroplasts, lysosomes, peroxisomes; Trafficking of proteins and RNAs; Organelle biogenesis, Subcellular localization prediction tools (e.g., DeepLoc); Cell Cycle and Checkpoints: Phases of the cell cycle: G1, S, G2, M; Cyclins, CDKs, and regulation; Checkpoints: G1/S, G2/M, spindle assembly; Apoptosis and cell senescence, Computational modeling of cell cycle regulation; Historical evolution and significance of Bioinformatics, Definition, Aims and Scope of Bioinformatics, Applications in drug discovery, agriculture, personalized medicine etc., Overview of computational biology, cheminformatics, systems biology.

Module 2:

Genetic Information Flow and Omics Technologies: DNA Replication, Repair, and Recombination: Enzymes of replication: polymerases, helicases, ligases; Origins of replication and replication fork; DNA damage types and repair pathways: BER, NER, MMR, HR, NHEJ; Site-specific recombination and transposons; Tools: DNA repair gene databases, mutation impact prediction; Transcription, RNA Splicing, Editing: RNA polymerases and transcription initiation, Enhancers, silencers, promoters; RNA processing: capping, splicing, polyadenylation; RNA editing and its biological roles, Transcriptomics and computational pipelines; Translation and Post-Translational Modifications: Genetic code and ribosome function; tRNA charging and translation initiation/elongation/termination; Protein targeting and secretion pathways; PTMs: phosphorylation, ubiquitination, acetylation; Proteomics and PTM databases; Genomics: structural and functional genomics, Transcriptomics: RNA-Seq and microarrays, Proteomics and systems-level understanding through omics integration.

Module 3:

Genetics, Heredity, and Epigenetic Regulation: Mendelian Genetics, Chromosomes, Mutations: Laws of inheritance, Dominance, epistasis, penetrance; Chromosomal basis of inheritance; Types of mutations: point, frameshift, CNVs; Genetic simulation tools (e.g., Mendel's Accountant); Genetic Mapping, Linkage, Genetic Disorders: Linkage analysis and recombination frequency, QTL mapping, Human pedigree analysis, Single-gene vs complex disorders, Bioinformatics for disease gene identification (e.g., OMIM, ClinVar); Molecular Genetics & Epigenetics: Gene expression regulation, DNA methylation, histone modifications, Chromatin remodeling, RNA-mediated regulation (miRNA, lncRNA); Epigenomic datasets (e.g., ENCODE) and visualization (UCSC Genome Browser); Role of genome annotation platforms and comparative epigenomics.

Module 4:

Biomolecules, Protein Structure, and Enzymes: Proteins, Carbohydrates, Lipids, Nucleic Acids: Basic chemistry and classification, Functional roles in cells, Computational representation (e.g., SMILES, PDB files); Protein Structure, Folding, and Function: Levels of protein structure, Forces driving protein folding, Molecular chaperones, Experimental methods: X-ray, NMR, Cryo-EM; Introduction to Computational tools: AlphaFold, PyMOL, MD simulations; Enzymes: Active Site, Mechanism, Kinetics: Structure-function relationships, Mechanism of action, catalytic strategies, Michaelis-Menten kinetics, Inhibitors and regulation, Enzyme databases (BRENDA), kinetic modeling with MATLAB or Python; Overview of secondary structure prediction methods (Chou–Fasman, GOR, ANN), tertiary structure modeling (comparative modeling, threading, ab initio), Role of AlphaFold and visualization using PyMOL.

Module 5:

Biological Databases, Sequence Analysis, and Systems Biology: Introduction to Biological data and data types; Importance of biological data management, Biological Data Characteristics; Concept of general databases and database management systems (DBMS); Advantages of relational database management system (RDBMS) over DBMS; Primary databases: GenBank, EMBL, DDBJ; Secondary databases: UniProt, RefSeq, Swiss-Prot; Specialized databases: Pfam, InterPro, KEGG, Reactome; Nucleotide vs Protein databases; Data retrieval tools: Entrez, SRS; File formats: flat files, ASN.1, XML; Sequence formats: FASTA, GenBank, GFF, PDB; Tools for sequence submission and retrieval; Introduction to NCBI tools, UCSC Genome Browser, Ensembl; Sequence Alignment and Similarity Search: Concepts of sequence similarity, identity, homology; Dynamic programming, gap penalties, scoring matrices (PAM, BLOSUM); Pairwise alignment methods: global (Needleman-Wunsch), local (Smith-Waterman); Heuristic methods: BLAST, PSI-BLAST, TBLASTX; Multiple sequence alignment: Clustal Omega, T-Coffee; Phylogenetic analysis: Tree properties and methods: UPGMA, Neighbor joining, Bayesian inference, Maximum parsimony, Maximum likelihood; Introduction to systems biology: Biological networks and computational modeling approaches.

References:

1.Mount, D.W., "Bioinformatics: Sequence and Genome Analysis", Cold Spring Harbor Press
2.Lesk, A.M., "Introduction to Bioinformatics", Oxford Univ. Press
3.Cormen, T.H. et al., "Introduction to Algorithms", MIT Press
4.Leach, A.R., "Molecular Modelling: Principles and Applications", Prentice Hall
5.Baxevanis, A.D., Ouellette, B.F.F., "Bioinformatics: A Practical Guide", Wiley
6.Branden, C. & Tooze, J., "Introduction to Protein Structure"
7.Liebler, D., "Introduction to Proteomics", Humana Press

Course Code	Course Name	L	T	P	Cr
DASH250022	Genomics, Transcriptomics and Proteomics	3	0	2	4

Course Outcomes:

CO1: Provide a comprehensive understanding of genome organization, transcriptome dynamics, and proteome complexity in prokaryotic and eukaryotic systems.
CO2: Equip students with theoretical and practical knowledge of sequencing technologies, transcriptome profiling, proteomic workflows, and associated analytical tools.
CO3: Develop skills to analyze and interpret omics datasets using computational pipelines and statistical methods.
CO4: Utilize public repositories and visualization platforms for genome, transcriptome, and proteome exploration.
CO5: Integrate multi-omics data to perform functional and network-based analyses for biological interpretation, biomarker discovery, and systems-level understanding.

Module 1:

Genome Organization and Annotation: Structural features of prokaryotic and eukaryotic genomes, chromosomal structure, centromeres, telomeres, repetitive sequences, tandem repeats, satellite DNA. Types of genes: housekeeping, tissue-specific, coding vs. non-coding, small RNAs (miRNA, siRNA, lncRNA), gene synteny and conservation. Exon-intron architecture, canonical splice sites, alternative splicing, 5' and 3' UTRs, regulatory RNAs. Regulatory elements including promoters, enhancers, silencers, insulators, TATA box, transcription factor binding motifs, chromatin structure, and epigenetic regulation. Genome annotation methods: ab initio prediction (e.g., Glimmer, GENSCAN), evidence-based annotation (e.g., MAKER, AUGUSTUS), comparative genomics (e.g., OrthoDB). Data formats: GFF3, BED, annotation pipelines, gene model validation, and database references (GENCODE, RefSeq, ENSEMBL).

Module 2:

Sequencing Technologies and Experimental Strategies: Evolution of sequencing: from Maxam–Gilbert and Sanger methods to high-throughput next-generation sequencing (NGS). Illumina technology: bridge amplification, paired-end sequencing, flow cell design, error profiles. Third-generation sequencing (TGS) technologies: PacBio SMRT, Oxford Nanopore; long-read sequencing, methylation detection, real-time sequencing advantages. RNA sequencing protocols: library preparation, fragmentation, strandedness, rRNA depletion vs. polyA selection, unique molecular identifiers (UMIs). Experimental design considerations: replicates, depth, coverage, dynamic range, sequencing biases. Single-cell transcriptomics: droplet-based (10x Genomics), plate-based (Smart-seq), barcoding strategies, unique challenges like dropout events and amplification noise. Introduction to proteomics workflows: experimental approaches in protein extraction, digestion, and peptide labeling (iTRAQ, TMT), sample preparation for LC-MS/MS, shotgun proteomics.

Module 3:

Transcriptome and Proteome Analysis: Read alignment vs. alignment-free quantification; tools such as STAR, HISAT2, Kallisto, Salmon. Transcript assembly with StringTie, Cufflinks; differential transcript usage, transcript-level abundance estimation. Normalization techniques for accurate expression comparison: TPM, RPKM/FPKM, TMM, DESeq2 normalization. Statistical methods for differential expression: negative binomial models, empirical Bayes moderation, shrinkage estimation, hypothesis testing. Tools: DESeq2, edgeR, limma-voom, sleuth. Batch effect correction using ComBat, SVA, PCA, and clustering techniques. Alternative splicing analysis (rMATS, SUPPA2), isoform switching, exon skipping events. Co-expression analysis and clustering, time-series transcriptomics. Protein identification using search engines (Mascot, Sequest, MaxQuant), peptide-spectrum matching, protein inference, protein quantification approaches (label-free, SILAC, SWATH-MS).

Module 4:

Visualization Tools and Omics Repositories: UCSC Genome Browser: custom tracks, track hubs, gene annotations, conservation scores, regulatory elements from ENCODE. Ensembl Genome Browser: transcript variants, comparative genomics, genome alignments, BioMart queries. Integrated Genome Viewer (IGV): real-time read inspection, coverage visualization, multi-sample comparisons, mutation annotation. JBrowse and other

modern genome viewers; handling track configuration and annotations. Public repositories and data portals: NCBI GEO, SRA, EBI ArrayExpress, ENCODE, GTEx, Human Protein Atlas. Proteomics data portals: PRIDE, PeptideAtlas, ProteomeXchange, MassIVE; protein structure databases like UniProt, PDB, and Pfam. Converting, sorting, and indexing alignment files; coordinate manipulation with tools like BEDTools and SAMtools; proteomic data preprocessing and spectral alignment.

Module 5:

Functional Omics and Network-Based Analysis: Biological interpretation of transcriptomic and proteomic data through Gene Ontology (GO): hierarchical structure, enrichment analysis, biological process, molecular function, cellular component. Enrichment tools: DAVID, Enrichr, g:Profiler, WebGestalt, clusterProfiler. Pathway analysis databases: KEGG, Reactome, WikiPathways, BioCarta; mapping gene or protein sets onto pathways. GSEA (Gene Set Enrichment Analysis): preranked analysis, enrichment score, false discovery rate (FDR), leading-edge subset. Protein functional enrichment using tools such as STRING, Panther, and Cytoscape plug-ins. Integrating multi-omics data (gene expression and proteomics) for systems-level insights. Network-based analysis: interaction networks from STRING, Cytoscape visualization, hub gene or protein identification. Case studies on disease-related transcriptomic and proteomic signatures, biomarker discovery, and regulatory module inference.

Labs / Practicals:

1. Accessing genome sequences and annotations from NCBI/Ensembl
2. Sequence quality check and trimming using FastQC and Trimmomatic
3. Genome assembly using SPAdes/Velvet
4. RNA-Seq alignment using HISAT2 or STAR
5. Differential gene expression using DESeq2
6. Visualization of gene expression with IGV
7. BLAST search and multiple sequence alignment of protein sequences
8. Protein domain analysis using Pfam/InterPro
9. 3D protein structure visualization using PyMOL/RasMol
10. Pathway mapping using KEGG and STRING

References:

1. Nelson DL, Cox MM. Lehninger Principles of Biochemistry. New York: W.H. Freeman.
2. Hartwell LH, Goldberg ML, Fischer JA, Hood L, Aquadro CF. Genetics: From Genes to Genomes. "Bioinformatics: Sequence and Genome Analysis", Cold Spring Harbor Press
3. Liebler, D., "Introduction to Proteomics", Humana Press
4. Pevsner, Jonathan. Bioinformatics and Functional Genomics. Hoboken, NJ: Wiley-Blackwell.

Course Code	Course Name	L	T	P	Cr
DCSE250080	Structural Bioinformatics	3	1	0	4

Course Outcomes:

CO1. Explain protein structural organization from primary to quaternary levels, including folding principles, stability factors, and molecular interactions.
CO2. Apply experimental methods such as X-ray crystallography, NMR spectroscopy, and Cryo-EM to determine, refine, and validate protein structures.
CO3. Analyze and compare protein structures using alignment algorithms, RMSD, TM-score, and domain classification for functional inference.
CO4. Construct, refine, and validate homology-based protein models using sequence alignment, template selection, molecular modeling, and energy minimization techniques.
CO5. Evaluate protein–ligand interactions and apply computer-aided drug design strategies, including docking, scoring, and virtual screening approaches.

Module 1:

Protein Structure Fundamentals: Primary structure – amino acid chemical properties, sequence motifs, conserved residues, peptide bond planarity, torsion angles (phi, psi, omega), sequence alignment for structural inference. Secondary structure – alpha helices, beta strands and sheets, beta turns, loops and coils, hydrogen bonding patterns, helix capping, amphipathic helices, Ramachandran plot analysis. Tertiary structure – folding principles, hydrophobic effect, electrostatic interactions, van der Waals forces, disulfide bonds, domain architecture, common folds (TIM barrel, Rossmann fold, beta-propellers), protein stability and folding pathways. Quaternary structure – multimeric assemblies, interface characterization, symmetry and stoichiometry, cooperative binding, allosteric regulation, protein-protein interaction interfaces, supramolecular complexes.

Module 2:

Structure Determination Techniques: X-ray crystallography – principles of crystallization, lattice formation, unit cell and space groups, diffraction and Bragg's law, data collection and processing, phase problem solutions (MIR, MAD, SAD), electron density interpretation, model building using COOT, structure refinement using REFMAC or Phenix, quality assessment using R-work, R-free, B-factors, resolution and map validation. NMR spectroscopy – isotope labeling (15N, 13C), chemical shift assignments, NOE-based distance constraints, 2D and 3D NMR experiments (HSQC, TOCSY, NOESY), structure calculation using CYANA or XPLOR-NIH, limitations and applications to flexible regions and small proteins. Cryo-electron microscopy (Cryo-EM) – sample vitrification, data collection using direct electron detectors, 2D class averaging, 3D reconstruction methods (single-particle analysis), map sharpening, model fitting, local resolution estimation, EMDB and PDB deposition standards. Structure validation tools – Ramachandran plot analysis, MolProbity metrics, clash score, rotamer outliers, real-space correlation, model-to-map validation techniques.

Module 3:

Structural Alignment & Comparison: Structural alignment concepts – rigid and flexible superposition, pairwise alignment, multiple structure alignment. RMSD – coordinate superposition, weighted RMSD, backbone vs. all-atom RMSD, limitations in flexible regions. TM-score and GDT_TS – fold comparison metrics, sensitivity to global structural similarity. Algorithms and tools – DALI for distance matrix comparison, TM-align for optimal alignment path, FATCAT for flexible alignments, MUSTANG for multiple alignments, CE and SSM algorithms. Structural motifs – helix-turn-helix, Greek key, beta-alpha-beta, EF-hand, zinc fingers, Rossmann folds, identification of functional or conserved regions. Domain classification – SCOP and CATH hierarchy, functional domain annotation, detection of structural repeats, family-level inference. Functional inference – conserved structural cores, active site prediction through structural homology, mapping sequence conservation onto structures.

Module 4:

Homology Modeling: Template identification – pairwise and profile-based searches (BLAST, PSI-BLAST, HMMER, HHblits), PDB template selection criteria (sequence identity, resolution, coverage), consideration of

multiple templates. Sequence-structure alignment – alignment accuracy, gap placement, manual refinement using visualization tools, influence on model quality. Model building – backbone modeling using Modeller, Swiss-Model, RosettaCM; loop modeling using ab initio and database-based approaches; side-chain placement using rotamer libraries (Dunbrack, SCWRL). Model validation – statistical potential assessments (DOPE, QMEAN), geometric validation (Ramachandran, rotamers, bond geometry), environmental profile scores (ProSA, VERIFY3D), packing quality checks (ERRAT), model comparison and clustering. Model refinement – energy minimization with molecular mechanics force fields (CHARMM, AMBER), molecular dynamics-based relaxation, application of restraints, hybrid modeling using experimental data, evaluation of local and global model quality.

Module 5:

Protein–Ligand Interactions and Computer-Aided Drug Design: Binding site identification using structural databases and pocket prediction tools (CASTp, Fpocket, SiteMap); analysis of cavity geometry, volume, and hydrophobicity. Ligand preparation—3D conformer generation, ionization states, stereochemistry—via LigPrep, Open Babel, RDKit. Protein preparation—adding missing atoms/residues, assigning protonation states, optimizing H-bond networks. Docking strategies—rigid and flexible docking with sampling algorithms (genetic, Monte Carlo, simulated annealing); tools like AutoDock, AutoDock Vina. Scoring functions—empirical, force-field, knowledge-based; post-docking refinement with MM-GBSA/MM-PBSA. Virtual screening—compound library filtering (drug-likeness, ADMET), high-throughput docking, hit prioritization. Structure-based drug design—pharmacophore modeling, fragment-based design, structure-guided optimization, ligand efficiency, integration with experimental validation; applications in kinase, protease, and antimicrobial drug discovery.

Labs / Practicals:

List to be updated

References:

1.Bourne, Philip E., and Helge Weissig, eds. 2009. Structural Bioinformatics. 2nd ed. Hoboken, NJ: Wiley-Blackwell.
2.Brändén, Carl, and John Tooze. 1999. Introduction to Protein Structure. 2nd ed. New York: Garland Science.
3.Leach, Andrew R. 2001. Molecular Modelling: Principles and Applications. 2nd ed. Harlow, England: Pearson Education.
4.Higgins, Des, and Willie Taylor. 2000. Bioinformatics: Sequence, Structure and Databanks. Oxford: Oxford University Press.
5.Merz, Kenneth M., Dagmar Ringe, and Charles H. Reynolds, eds. 2010. Drug Design: Structure- and Ligand-Based Approaches. Cambridge: Cambridge University Press.

Course Code	Course Name	L	T	P	Cr
DCSE250089	Linux and Python for Biological Data Analysis	3	1	0	4

Course Outcomes:

CO1. Apply Python fundamentals including variables, loops, conditionals, functions, and file I/O to parse and analyze biological sequence data.
CO2. Use regular expressions for pattern recognition, parsing genomic files, and extracting motifs, gene features, and variant information efficiently.
CO3. Employ Biopython modules for sequence manipulation, annotation handling, BLAST parsing, and database queries to support bioinformatics workflows.
CO4. Develop Linux shell scripts with conditional logic, loops, pipelines, and cron jobs to automate genomic data processing tasks.
CO5. Integrate Python scripts with Linux tools to design reproducible, modular, and collaborative workflows for bioinformatics data analysis.

Module 1:

Python Basics: Variables, loops, conditionals, functions: Foundational building blocks of Python programming, beginning with variables (integers, floats, strings, lists, dictionaries, tuples) for handling biological data such as sequences, gene IDs, and annotations. Use of string operations for sequence handling and slicing, list comprehensions for concise filtering, and dictionary-based data storage for structured biological entries. Conditional logic using if, elif, else, and logical operators (and, or, not) for decision-making processes in sequence parsing or filtering. Loops including for and while for repetitive execution across genome files or transcript entries. Function definition using def, argument passing, return values, and modular code writing for reuse in pipelines. File I/O operations using open(), context managers (with), and reading/writing FASTA, GFF, and other formats line-by-line. Practical exercises include parsing DNA/RNA sequence files, extracting open reading frames, and filtering annotation lines based on gene features using conditional blocks.

Module 2:

Regular Expressions: Pattern matching, file parsing: Python's re module for pattern recognition and parsing unstructured or semi-structured biological data. Regex syntax and metacharacters such as ., *, +, ?, ^, $, and character classes for identifying motifs, gene patterns, or IDs in sequence data. Grouping with (), alternation using |, and the use of raw strings (r'') for clarity. Substitution (re.sub), matching (re.match, re.search), and global search (re.findall) for reformatting headers in FASTQ files, parsing GenBank features, or extracting gene names and coordinates. Use of modifiers like re.IGNORECASE, re.MULTILINE for flexible matching. Applications include scanning regulatory motifs, cleaning annotation files, and extracting mutation or variant data from VCFs, GTFs, and GenBank records. Emphasis on using regex inside Python loops and conditionals for scalable pattern-driven parsing in genomics workflows.

Module 3:

Biopython: Sequence handling, annotation, and BLAST parsing: Introduction to Biopython, a comprehensive Python library for biological computation. Installation and overview of module structure. Core modules: Bio.Seq, Bio.SeqIO, Bio.SeqRecord for creating, reading, writing, and manipulating sequence data in formats like FASTA, GenBank, EMBL. Accessing attributes like .id, .description, .seq, .annotations, and .features in SeqRecord objects. Sequence manipulations: slicing, reverse complementing, transcription, and translation. Working with Bio.AlignIO to parse multiple sequence alignments (ClustalW, PHYLIP), extract conserved regions, or compute alignment summaries. Parsing BLAST output using Bio.Blast.NCBIXML and Bio.Blast.Record to retrieve HSPs, query-hit mappings, and identity scores. Database access through Bio.Entrez for remote querying and batch sequence retrieval from NCBI using email authentication. Annotation handling using GenBank features and custom extraction scripts for CDS, gene, and exon annotations. Real-world exercises include generating translation tables, extracting CDS regions, converting between formats, and querying gene entries from online databases.

Module 4:

Linux Scripting: Bash scripting, pipelines, cron jobs: Scripting with bash: using proper shebang (#!/bin/bash), managing execution permissions (chmod +x), and using input parameters $1, $2, $@, $# to handle file input/output dynamically. Declaring and referencing variables, quoting rules (", ', `), arithmetic and conditional evaluations, command substitution with $(...). Conditional branching using if, elif, else, and case for checking file presence, format integrity, or parameter values. Loops (for, while, until) for batch processing genomic files or automating repeated format conversions. Constructing data-processing pipelines using |, combined with sed, awk, cut, sort, uniq, head, tail, and paste to clean and restructure biological datasets. Input/output redirection (>, >>, 2>, &>) and tee for traceable logging. Scheduled automation using cron and crontab syntax (* * * * *) to execute routine jobs such as nightly data backups, genome reindexing, or BLAST updates. Hands-on tasks include bash wrappers for sequence cleaning, alignment log generation, and time-based email reports.'>, >>, 2>, &>) and tee for traceable logging. Scheduled automation using cron and crontab syntax (* * * * *) to execute routine jobs such as nightly data backups, genome reindexing, or BLAST updates. Hands-on tasks include bash wrappers for sequence cleaning, alignment log generation, and time-based email reports.'>Linux Scripting: Bash scripting, pipelines, cron jobs: Scripting with bash: using proper shebang (#!/bin/bash), managing execution permissions (chmod +x), and using input parameters $1, $2, $@, $# to handle file input/output dynamically. Declaring and referencing variables, quoting rules (", ', `), arithmetic and conditional evaluations, command substitution with $(...). Conditional branching using if, elif, else, and case for checking file presence, format integrity, or parameter values. Loops (for, while, until) for batch processing genomic files or automating repeated format conversions. Constructing data-processing pipelines using |, combined with sed, awk, cut, sort, uniq, head, tail, and paste to clean and restructure biological datasets. Input/output redirection (>, >>, 2>, &>) and tee for traceable logging. Scheduled automation using cron and crontab syntax (* * * * *) to execute routine jobs such as nightly data backups, genome reindexing, or BLAST updates. Hands-on tasks include bash wrappers for sequence cleaning, alignment log generation, and time-based email reports.

Module 5:

Script Integration: Creating workflows using Python and Linux tools: Interconnecting Python scripts with Linux command-line tools using os.system, subprocess.run, or shell wrappers. Passing input/output between tools, reading from configuration files, or using command-line arguments (argparse module) for flexible parameterization. Designing reproducible workflows by modularizing scripts, naming conventions, and structured output directories. Integrating Python preprocessing steps (e.g., sequence filtering, format conversion) with Linux tools for downstream processing like alignments, statistical summaries, or plot generation. Case studies on hybrid pipelines: quality filtering using Python, alignment with bwa or hisat2 via bash, and post-processing VCF or expression data using both scripting environments. Logging, error handling using try-except, and notification through automated emails or reports. Introduction to workflow chaining via shell scripts, makefiles, or Python-based task runners (e.g., Snakemake, Luigi). Emphasis on reproducibility, documentation via comments or docstrings, and version control integration with Git for collaborative pipeline development.

Labs / Practicals:

List to be updated

References:

1. Dale D, Burch CH. Python for Bioinformatics. 2nd ed. Cham: Springer; 2021.
2. Cock PJA, Antao T, Chapman BA, Cox CJ, Dalke A, Friedberg I, et al. Biopython Tutorial and Cookbook. Version 1.79. Open Bioinformatics Foundation; 2020.
3. Kind D, Holzschuh A. Linux for Bioinformatics: A Beginner’s Guide. Cham: Springer; 2021.
4. Hedstrom L. Introduction to Linux for Bioinformatics. New York: Oxford University Press; 2022.
5. Baxevanis AD, Ouellette BFF, editors. Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins. 3rd ed. Hoboken (NJ): Wiley-Blackwell; 2005.

Course Code	Course Name	L	T	P	Cr
DASH250043	Statistical Methods in Computational Biology and Problem-Solving using R	2	0	4	4

Course Outcomes:

CO1: Apply basic statistical concepts (data types, distributions, measures of central tendency/dispersion, probability) to biological datasets.
CO2: Perform hypothesis testing and inferential analyses (t-tests, chi-square, ANOVA) for experimental and clinical biological studies.
CO3: Use regression and correlation methods to model biological relationships and interpret association studies.
CO4: Employ multivariate statistical techniques (PCA, clustering, dimensionality reduction) for omics and systems biology data.
CO5: Implement statistical analyses and visualizations in R for real-world biological applications

Module 1:

Basic Statistics:
Concepts of population and sample; types of data in biology: continuous, discrete, nominal, and ordinal variables; frequency distributions and data summarization techniques. Measures of central tendency: mean, median, and mode, with applications in gene expression analysis and sample profiling. Measures of dispersion: range, variance, standard deviation, and coefficient of variation; importance of variability in biological data and interpretation in high-throughput assays. Introduction to probability theory and its applications in stochastic biological processes; axioms of probability and rules of addition and multiplication. Common discrete and continuous probability distributions; biological interpretation and modeling with binomial distribution in gene mutation experiments, Poisson distribution in count-based RNA-seq data, and normal distribution for measurement errors and population statistics. Concepts of skewness, kurtosis, z-scores, and standardization of biological variables.

Module 2:

Inferential Statistics:
Importance of sampling in biology; random sampling strategies and experimental design principles; population parameters vs. sample statistics. Sampling distributions and the Central Limit Theorem; confidence intervals for means, proportions, and variances. Hypothesis testing framework in biological research: null and alternative hypotheses, Type I and II errors, p-values, power of a test, and decision criteria. Parametric hypothesis tests: t-tests; paired and unpaired designs; applications in comparing expression levels between control and treated groups. Non-parametric alternatives and robustness of t-tests under non-normal conditions. Chi-square test for independence in contingency tables; genetic linkage studies and Hardy-Weinberg equilibrium testing. ANOVA for comparing multiple groups means; assumptions of ANOVA and diagnostics; post-hoc tests including Tukey's HSD and Bonferroni correction for multiple comparisons. Biological applications in experimental designs such as microarray and clinical data interpretation.

Module 3:

Regression and Correlation:
Simple linear regression for modeling relationships between continuous biological variables; estimation of regression coefficients using least squares method; interpretation of slope and intercept in biological terms. Assumptions of linear regression and diagnostic checks for residuals, leverage, and multicollinearity. Multiple linear regression models involving multiple predictors; biological applications in modeling phenotypic traits, metabolomic variations, and multi-gene effects. Categorical predictors and use of dummy variables in biological classification. Logistic regression for binary outcomes in case-control studies and classification tasks. Correlation analysis for association studies; Pearson's and Spearman's correlation coefficients; significance testing for correlations and their interpretation in biological co-expression networks. Introduction to partial correlation and its use in identifying indirect relationships in molecular networks.

Module 4:

Multivariate Statistics:

Need for multivariate analysis in systems biology and omics data interpretation; dimensionality reduction techniques. Principal Component Analysis (PCA): mathematical foundations, eigenvalues and eigenvectors, loading scores, scree plot, and biplot interpretation; applications in visualizing population structure, gene expression patterns, and clustering of samples. Standardization and scaling of data prior to PCA. Hierarchical clustering: distance metrics (Euclidean, Manhattan, correlation-based), linkage methods (single, complete, average, Ward's method), dendrogram construction and interpretation. Use of heatmaps representing hierarchical clustering results; applications in transcriptomic and proteomic datasets. Introduction to k-means clustering and its comparison with hierarchical clustering. Advanced methods: multidimensional scaling (MDS) and t-SNE for visualization of complex biological patterns. Practical case studies: clustering of cancer subtypes, tissue classification, and sample stratification.

Module 5:

Introduction to R:
Overview of the R programming environment; installation, interface, packages, and RStudio. Data import from text, Excel, and biological formats such as FASTA and CSV; data types, factors, and data frames; handling missing values and data cleaning. Descriptive statistics and graphical summaries using base R and packages such as ggplot2, dplyr, and tidyr. Statistical tests in R: implementation of t-tests, chi-square tests, ANOVA, correlation, and regression models using base functions and stats package. Model diagnostics and visualization of residuals and fits. Application of PCA and clustering using FactoMineR, cluster, and pheatmap. Writing functions and scripts for biological data analys Case-based applications: microarray data analysis using limma, RNA-seq differential expression with DESeq2, and visualization of multi-omic patterns using ggplot2 and ComplexHeatmap

Labs / Practicals:

Module 1 – Basics of R and Descriptive Statistics
Data types and data frames in R, importing datasets and handling missing values, visualization using ggplot2 including histograms,
boxplots, scatter plots; summary statistics and distribution fitting.
Module 2 – Hypothesis Testing
Implementation of t-test, ANOVA, chi-square test using R; estimation of confidence intervals, interpretation of p-values; application of
statistical tests on gene expression data from sources like GEO.
Module 3 – Regression and Correlation
Building and interpreting linear and logistic regression models using lm() and glm(); plotting regression lines and residuals; computing
correlation matrices; visualizing relationships through heatmaps.
Module 4 – PCA and Clustering
Principal Component Analysis using prcomp(); generation of scree plots and biplots; k-means and hierarchical clustering on gene
expression datasets; generation of heatmaps using pheatmap or ComplexHeatmap.
Module 5 – Bioinformatics Applications in R
RNA-Seq analysis using DESeq2; creation of volcano plots and differential gene expression reports; pathway and protein interaction
analysis using KEGG and STRINGdb; reproducible report generation using R Markdown and basic GitHub integration.

References:

1.Dalgaard P. Introductory Statistics with R.
2.Maindonald J, Braun WJ. Data Analysis and Graphics Using R: An Example-Based Approach. Cambridge: Cambridge University Press.
3.Gentleman R, Carey VJ, Huber W, Irizarry RA, Dudoit S, editors. Bioinformatics and Computational Biology Solutions Using R and Bioconductor.
4.Crawley MJ. Statistics: An Introduction Using R. Chichester: Wiley.
5.Everitt BS, Hothorn T. A Handbook of Statistical Analyses Using R. Boca Raton: Chapman and Hall/CRC.

Course Code	Course Name	L	T	P	Cr
DOAI260001	AI (Industry)-Applied Ethics for AI	3	0	2	4

Course Outcomes:

CO1. Understand and be able to define and explain the terms 'AI for good' and 'responsible AI'. Describe the significance of responsible AI and recognize the potential benefits of AI in society.
CO2. Discuss and examine the impact of ethics on AI decision-making processes. Identify common ethical pitfalls in AI development and deployment. Explore methods for ethical AI practices.
CO3. Understand to apply an ethics-first approach to AI project planning and development. Compare different approaches to ethical AI design which include design of rules based on ethics and virtues. Identify factors influencing ethical decisions in AI development.
CO4. Apply Utilitarian and Kantian ethical principles to AI case studies, scenarios, and dilemmas. Explain the purpose and benefits of AI ethics codes. Develop a code of ethics for AI within an organization.
CO5. Identify the importance of interpretability in AI models. Evaluate the strengths and weaknesses of three different explainable AI tools: SHAP, LIME, and Tree Interpreter.
CO6. Conduct impact assessments for AI solutions. Apply ethical considerations in designing an AI system or solution. Implement measures to mitigate ethical risks and challenges post deployment.

Module 1:

Introduction to AI Ethics – meaning and importance, Role of Ethics in Responsible AI – defining 'AI for good' and 'responsible AI', significance of responsible AI in society, Common Ethical Pitfalls in AI – bias, fairness, accountability, transparency issues in AI development and deployment, Principles of Ethical AI – core principles guiding ethical AI, methods for ethical AI practices in development and deployment.

Module 2:

Ethics-First Approach in the AI Project Cycle – applying ethics at every stage of AI project planning and development, Approaches for Designing Ethical AI – rules-based ethics, values-based design, Introduction to Ethical Frameworks – overview of major ethical frameworks applied to AI, Virtue-Based Ethics – virtue ethics principles and the 4-box method for analysing ethical dilemmas, Applying Bio-Ethics Framework in AI – bioethical principles applied to AI in healthcare and biotechnology.

Module 3:

Utilitarianism for Ethical AI – utilitarian principles applied to AI case studies and dilemmas, Kantianism for Ethical AI – Kantian ethical principles applied to AI scenarios, Code of AI Ethics for Organizations – purpose, benefits and development of an organizational AI code of ethics, Data Protection and AI – risks of data breaches, privacy violations, responsible data handling.

Module 4:

Explainable AI – importance of interpretability in AI models, Intel XAI Toolkit, impact of explainability on decision-making and trust, Explainable AI in Practice – SHAP, LIME and Tree Interpreter tools: strengths, weaknesses and methods for applying them to analyze and interpret AI applications.

Module 5:

Impact Assessment of AI Solutions – conducting impact assessments for AI systems, Mitigating Ethical Issues Post Deployment – strategies to identify and resolve ethical risks after deployment, Project Creation – creating an AI startup with data card, model card, code of ethics, explainability report, impact self-assessment and transparency report.

Labs / Practicals:

1. Develop comprehensive Data Cards for datasets, as outlined in the Data Cards Playbook by Google, to ensure transparency and accountability in AI training data.
2. Design Responsible AI Best Practices.

3. Apply Human-Centered AI Canvas to design AI systems that address user needs and ethical considerations.
4. Learn to use The Box to embed AI principles into organizational workflows and operations.
5. Apply bio-ethics principles to AI case studies, scenarios, and dilemmas; evaluate the impact of virtue-based ethics on AI systems.
6. Develop AI ethics cards for given AI scenarios. Analyse a variety of ethical dilemmas by following the 4-box method from Virtue Ethics.
7. Analyse a variety of ethical dilemmas by walking through the ethical frameworks: Virtue Ethics, Bioethics, Utilitarian, Kantian, and Moral System Agnostic.
8. Develop an AI Code of Ethics for a fictitious company.
9. Use the UXAI Toolkit to brainstorm and plan different explainable interfaces for AI applications. Use the explainable AI library SHAP to create an explainability report for income prediction on the standard adult census income dataset.
10. Conduct impact assessments for AI solutions; create your own AI startup, develop business documents, write documentation (data card, model card, code of ethics, explainability report, impact self-assessment) and create and analyze transparency report.

References:

1. Intel AI for Future Workforce – Applied Ethics for AI Course Content, Intel Digital Readiness Program.
2. Virginia Eubanks – Automating Inequality: How High-Tech Tools Profile, Police, and Punish the Poor, St. Martin's Press.
3. Cathy O'Neil – Weapons of Math Destruction, Crown Publishers.
4. Kate Crawford – Atlas of AI: Power, Politics, and the Planetary Costs of Artificial Intelligence, Yale University Press.
5. Google Data Cards Playbook – https://pair-code.github.io/datacardsplaybook/

Course Code	Course Name	L	T	P	Cr
DOAI260002	AI (Industry)-Introduction to Artificial Intelligence	3	0	2	4

Course Outcomes:

CO1. Describe what AI is and what it is not. Appreciate the history of AI and its evolution over time.
CO2. Identify and analyze current trends in AI by correlating them with other technologies like IoT, Big Data and 5G.
CO3. Identify 3 common domains of AI – Natural Language Processing, Computer Vision and Statistical Data – based on the type of underlying data.
CO4. Examine and appreciate typical steps involved in an AI Project through the AI Project Cycle.
CO5. Examine the immediate impacts of AI practices on society and appreciate the ethical concerns of AI applications.
CO6. Classify Supervised Learning, Unsupervised Learning and Reinforcement Learning with the help of examples. Discuss the roles played by different key stakeholders in a typical AI team.

Module 1:

Demystification of AI – what AI is and what it is not, History and Evolution of AI, Current trends in AI, AI and Technology convergence, Industry 4.0 and Digitalization, Role of IoT, Big Data and 5G in AI, Domains of AI – Natural Language Processing, Computer Vision and Statistical Data.

Module 2:

AI Project Cycle-I – Problem Scoping and Data Acquisition, AI Project Cycle-II – Modelling and Evaluation, Societal Impact of AI – ethical concerns and responsible practices, Elements of ML – Supervised Learning, Unsupervised Learning and Reinforcement Learning with examples.

Module 3:

Elements of AI – types and characteristics, Data as Fuel for AI – structured and unstructured data, methods of data mining and data storage, AI/Data Science Team – roles and responsibilities of key stakeholders.

Module 4:

Tools to implement AI – No-Code tools (Teachable Machine, Orange, Roboflow, Chatfuel) and Code-based tools, Introduction to Neural Networks – working of simple Neural Networks, difference between common Deep Learning models with examples and applications.

Module 5:

AI Project-I – Natural Language Processing use case using no-code tool Chatfuel, AI Project-II – Statistical Data use case using no-code tool Orange Data Mining, AI Project-III – Computer Vision use case using no-code tool Roboflow, Future Possibilities of AI – emerging software and hardware technology trends and their direct impact on development of AI.

Labs / Practicals:

1. Familiarize with AI tools and techniques, including statistical data analysis using raw graphs, computer vision applications like Vision API, and generative AI tools such as DALL·E, Stable Diffusion, Gemini, and ChatGPT.
2. Apply the AI Project Cycle to real-world scenarios by identifying problems, collecting data, developing models, and evaluating their effectiveness to propose impactful AI-based solutions.
3. Deconstruct the working behind simple Neural Networks and outline the difference between some common DL models with the help of examples and applications.
4. Explore and utilize no-code tools, such as Teachable Machine, to build AI projects.
5. Project Building using No-code tool – Orange Data Mining for Statistical Data Use cases.
6. Project Building using No-code tool – Roboflow for Computer Vision Use cases.
7. Project Building using No-code tool – Chatfuel for Natural Language Processing Use cases.
8. Classify the type of data into structured and unstructured and evaluate data quality using real datasets.

9. Explore various AI domains by running Vision API experiments and comparing outcomes from NLP, CV, and Statistical Data use cases.
10. Demonstrate the AI Project Cycle on a chosen real-world problem from an industrial or social impact perspective and present findings.

References:

1. Intel AI for Future Workforce – Introduction to AI Course Content, Intel Digital Readiness Program.
2. Stuart Russell and Peter Norvig – Artificial Intelligence: A Modern Approach, Pearson Education.
3. Andreas Mueller and Sarah Guido – Introduction to Machine Learning with Python, O'Reilly Media.
4. Ian Goodfellow, Yoshua Bengio, and Aaron Courville – Deep Learning, MIT Press.
5. Aurélien Géron – Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly Media.

Course Code	Course Name	L	T	P	Cr
DOAI260003	AI (Industry)-Introduction to Generative AI	3	0	2	4

Course Outcomes:

CO1. Define Gen AI and distinguish its role in the broader context of AI. Describe how the basic model of Gen AI works. Discuss the wider societal impact of Gen AI.
CO2. Understand how NLP has evolved over time and the shift from Traditional AI to Gen AI. Understand Prompt Engineering and its impact on the output of NLP tasks.
CO3. Learn the limitations of current AI generation tools for video and audio. Recognize the challenges in ethical considerations and responsibilities when using generative AI tools.
CO4. Describe the stages of a generative AI project and learn to select and use appropriate AI tools for development.
CO5. Understand the basic structure of neural networks, architecture of transformer models, workings of diffusion models, and the pre-training process for generative AI models.
CO6. Learn to fine-tune generative AI models, understand RLHF, prototype Gen AI applications, and apply principles of responsible and ethical use of LLMs.

Module 1:

Introduction to Gen AI – definition and scope, role of Gen AI in the broader context of AI, how the basic model of Gen AI works, societal impact of Gen AI, Demystifying Gen AI Models – types of generative models (GANs, VAEs, Diffusion Models, LLMs), evolution of NLP to Gen AI, shift from Traditional AI to Gen AI.

Module 2:

Prompt Engineering-I – introduction to prompt engineering, basic applications and impact on NLP task outputs, text generation and conversation initiation, Prompt Engineering-II – advanced prompt engineering techniques for diverse NLP applications, real-world examples such as Mint Mobile Commercial, Basics of Video and Audio for Generation using Gen AI for Productivity-I – introduction to video generation tools (Runway ML, Genmo), text-to-video and image-to-video generation, limitations of current AI generation tools for video and audio.

Module 3:

Basics of Video and Audio for Generation using Gen AI for Productivity-II – audio and music generation using AIVA, code generation using Codeium, text generation using ChatGPT, Gemini and Perplexity, Generative AI-Ethics – ethical considerations, responsibilities and challenges when using generative AI tools, Generative AI Project Lifecycle – stages of a Gen AI project, selection of appropriate AI tools, Gen AI Tools for Development – open-source tools such as Stable Diffusion XL, proprietary tools such as Bing Image Creator powered by DALLE3.

Module 4:

Neural Network Fundamentals – basic structure and functioning of neural networks, how neural networks process data to make predictions, Transformers and the Evaluation of LLMs – architecture of transformer models, methods to evaluate large language models for performance and reliability, walkthrough of OpenAI Playground and parameter exploration, Diffusion Models and the Evaluation of Text to Image Generation – workings of diffusion models, evaluation of quality of text-to-image generation, Pre-Training Gen AI Models-I – pre-training process for generative AI models, importance of pre-training in building effective AI systems, modifying classification labels of Cohere datasets and its impact on fine-tuning and model confidence.

Module 5:

Fine-Tuning Gen AI Models – techniques for fine-tuning generative AI models to adapt them for specific tasks and applications, RLHF – Evaluating and Optimizing Gen AI Models: reinforcement learning with human feedback, evaluation and improvement of performance of generative AI models, Prototyping Gen AI Applications – skills to design, develop and test prototypes of generative AI applications, LangChain components – LLM, Agents, Prompts, Chains, Document loaders and utils with code-based implementation, LLM Application and Responsible use of AI – practical applications of large language models, ethical and responsible use of LLMs, building an AI

Tutor using open-source models such as Dolly 3B and Intel Neural Chat 7B.

Labs / Practicals:

1. Apply prompt engineering skills in basic NLP tasks, such as text generation, conversation initiation, and simple problem-solving.
2. Develop advanced prompt engineering skills for diverse NLP applications. Examine real-world prompt engineering examples, such as the Mint Mobile Commercial, to discern its role in marketing.
3. Explore cutting-edge Gen AI tools, their features, and capabilities for content generation including video and audio.
4. Explore open-source models such as Stable Diffusion XL and proprietary tools such as Bing Image Creator (powered by DALLE3) for image generation.
5. Text-to-video generation and image-to-video generation using tools like Runway ML and Genmo. Generate audio and music files using the tool AIVA.
6. Generate code by prompts using the tool Codeium. Generate text using tools like ChatGPT, Gemini, and Perplexity.
7. Walkthrough of OpenAI Playground. Explore how changing parameters can affect responses generated by different models.
8. Explore how modifying classification labels of Cohere datasets impacts fine-tuning and model confidence.
9. Explore basic components of LangChain like LLM, Agents, Prompts, Chains, and Document loaders and utils, along with a code-based implementation.
10. Utilize open-source text generation models such as Dolly 3B and Intel Neural Chat 7B for building an AI Tutor application.

References:

1. Intel AI for Future Workforce – Introduction to Generative AI Course Content, Intel Digital Readiness Program.
2. Antony Hagiwara – Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play, O'Reilly Media.
3. Ashish Vaswani et al. – Attention is All You Need, Advances in Neural Information Processing Systems.
4. OpenAI – GPT-4 Technical Report, OpenAI.
5. Harrison Chase – LangChain Documentation – https://docs.langchain.com/

Course Code	Course Name	L	T	P	Cr
DOAI260004	AI (Industry)-Introduction to Machine Learning	3	0	2	4

Course Outcomes:

CO1. Describe Machine Learning and how Deep Learning is a subset of the wider field of Machine Learning.
CO2. Understand the importance of dashboards and discuss various applications of dashboards across different industries.
CO3. Explain the mathematical working behind different ML models. Implement different classification and regression ML models using Python.
CO4. Outline the difference between supervised, unsupervised and reinforcement learning algorithms. Examine the working of different reinforcement learning models.
CO5. Describe the working of Neural Networks and contrast it with biological Neurons. Describe various applications of neural networks. Outline different methods to overcome variance and bias in DL models.
CO6. Discuss and interpret the future of ML based on current and upcoming trends. Develop Python-based AI Projects incorporating Supervised Learning, Unsupervised Learning and Deep Neural Networks.

Module 1:

Introduction to Machine Learning – definition, types and applications, Relationship between AI, ML and Deep Learning, Tools to Implement AI (Python & Mathematics) – I: Python basics for ML, NumPy, Pandas, data manipulation.

Module 2:

Tools to Implement AI (Python & Mathematics) – II: mathematical foundations – linear algebra, statistics and probability, No-Code tool for Data Exploration – Tableau dashboard creation, data visualization techniques, statistical concepts implementation using no-code visualization tools.

Module 3:

AI (Modeling) Supervised Learning-I – Linear Regression, Logistic Regression, mathematical working behind models, classification and regression using Python, AI (Modeling) Supervised Learning-II – SVM, Decision Tree, industrial applications of ML models, AI (Modeling) Unsupervised Learning-I – K-Means Clustering, mathematics of clustering algorithms, AI (Modeling) Unsupervised Learning-II – Recommendation Systems, mathematics behind recommendation system, applications of recommendation systems.

Module 4:

Reinforcement Learning – definition, difference from supervised and unsupervised learning, working of reinforcement learning models with applications, Neural Network and Deep Learning-I – biological vs artificial Neurons, working of Neural Networks, applications of neural networks, Deep Learning-II – Convolutional Neural Networks, Recurrent Neural Networks, methods to overcome variance and bias in DL models, implementation using Python and deep learning frameworks, exploring OpenAI's Gym for reinforcement learning.

Module 5:

ML Project – Supervised Learning AI models: end-to-end Python project using classification/regression, ML Project – Unsupervised Learning AI models: end-to-end Python project using clustering, ML Project – Deep Neural Networks: building and training DNN in Python, Future of Machine Learning and Deep Learning – current and upcoming trends in ML and DL.

Labs / Practicals:

1. Interpret and understand the basic tools required for building ML Projects through a selected list of topics from Mathematics and Python Programming.
2. Develop and create a simple dashboard for visualizing data through Tableau.
3. Implement statistical concepts using the no-code visualization tool and create your own dashboard with the help of a no-code visualization tool.

4. Implement supervised machine learning algorithms, such as SVM and Decision Tree, using Python to develop a deeper understanding of their practical applications.
5. Implement unsupervised machine learning algorithms, such as K-Means, using Python to develop a deeper understanding of their practical applications.
6. Implement neural networks and create simple generative AI models using Python and deep learning frameworks, including exploring OpenAI's Gym for hands-on experience with AI and reinforcement learning.
7. Develop Python-based use cases and AI Projects incorporating Supervised Learning methods.
8. Develop Python-based use cases and AI Projects incorporating Unsupervised Learning methods.
9. Develop Python-based use cases and AI Projects incorporating Deep Neural Networks.
10. Implement a recommendation system using Python and evaluate its performance on a real-world dataset.

References:

1. Intel AI for Future Workforce – Introduction to Machine Learning Course Content, Intel Digital Readiness Program.
2. Andreas Mueller and Sarah Guido – Introduction to Machine Learning with Python, O'Reilly Media.
3. Ian Goodfellow, Yoshua Bengio, and Aaron Courville – Deep Learning, MIT Press.
4. Aurélien Géron – Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly Media.
5. Christopher Bishop – Pattern Recognition and Machine Learning, Springer.

Course Code	Course Name	L	T	P	Cr
DASH250045	Drug Discovery and Molecular Modeling	3	0	2	4

Course Outcomes:

CO1: Understand the principles, phases, and strategies of modern drug discovery.
CO2: Apply molecular modeling techniques in structure-based drug design.
CO3: Analyze protein-ligand interactions using computational tools and docking methods.
CO4: Evaluate lead compounds using ADMET and virtual screening pipelines.
CO5: Perform and interpret results from molecular dynamics simulations and cheminformatics approaches.

Module 1:

Introduction to Drug Discovery:
Overview of drug discovery and development pipeline; target identification and validation; hit and lead identification; lead optimization; preclinical and clinical phases; role of bioinformatics and AI in drug discovery; Indian pharma R&D landscape.

Module 2:

Molecular Modeling – Theory and Methods:
Energy minimization, force fields (MMFF, CHARMM, AMBER); molecular mechanics vs. quantum mechanics; conformational analysis; homology modeling; introduction to molecular dynamics and simulation workflows.

Module 3:

Ligand- and Structure-Based Drug Design:
Pharmacophore modeling; de novo drug design; molecular docking – rigid vs. flexible, scoring functions, algorithms (AutoDock, Glide, GOLD); 3D-QSAR, CoMFA, CoMSIA; case studies from antiviral and anticancer drugs.

Module 4:

Virtual Screening and ADMET:
Virtual screening types – ligand-based, structure-based, high-throughput screening; Lipinski's rule of five; ADMET (absorption, distribution, metabolism, excretion, toxicity) profiling tools; databases – PubChem, ChEMBL, ZINC, BindingDB.

Module 5:

Applications and Trends in Drug Discovery:
AI/ML in drug design; peptide drug discovery; biosimilars and biobetters; molecular dynamics (GROMACS, NAMD); case studies – COVID-19, TB, neglected tropical diseases; regulatory aspects; innovation in India – CSIR, DBT, ICMR, Pharma Vision 2020.

Labs / Practicals:

1.Protein structure retrieval and visualization – PyMOL / Chimera
2.Energy minimization and geometry optimization – Avogadro / SwissParam
3.Molecular docking of ligand to target protein – AutoDock / AutoDock Vina
4.Pharmacophore model creation and virtual screening – PharmaGist / ZINCPharmer
5.QSAR modeling and interpretation – PaDEL-Descriptor / R / QSAR tools
6.ADMET prediction of lead molecules – SwissADME / ADMETSAR
7.Homology modeling of target proteins – SWISS-MODEL / MODELLER
8.Molecular dynamics simulation setup (intro only) – GROMACS / NAMD

References:

1. Ashutosh Kar – Medicinal Chemistry, New Age International Publishers.
2. V.K. Ahluwalia & Madhu Chopra – Medicinal Chemistry, Ane Books Pvt. Ltd.

3. G.L. Patrick – An Introduction to Medicinal Chemistry, Oxford University Press.
4. Pandi Veerapandian – Structure-Based Drug Design, CRC Press.
5. R. Gupta & S. Purohit – Molecular Modelling and Drug Design, CBS Publishers.

Course Code	Course Name	L	T	P	Cr
DASH250051	Systems Biology and Network Analysis	3	1	0	4

Course Outcomes:

CO1: Understand systems-level concepts and mathematical frameworks used to study complex biological functions.
CO2: Explore interactions among genes, proteins, and metabolites through the construction and interpretation of biological networks.
CO3: Analyze biological network structures and pathways using computational databases and topological approaches.
CO4: Integrate omics data and utilize pathway resources for functional annotation and disease modeling.
CO5: Apply modeling and simulation techniques to predict, analyze, and visualize system-level behavior in cellular networks.

Module 1:

Introduction to Systems Biology:
Systems-level perspective on biological functions; contrast between reductionist and holistic approaches; understanding emergent behavior in biological systems. Biological complexity—modularity, robustness, feedback regulation, signal amplification, noise filtering. Central dogma and its system-wide modulation; dynamic interplay between DNA, RNA, proteins, metabolites. Omics integration—genomics, transcriptomics, proteomics, metabolomics, epigenomics; significance in decoding phenotype from genotype. Functional genomics and high-throughput technologies; RNA-seq, microarrays, mass spectrometry, next-generation sequencing. Data generation, normalization, preprocessing. Biological networks as systems representations. Mathematical abstraction of biological processes. Interdisciplinary nature—interface of biology, mathematics, engineering, and computer science. Case studies—systems approach in cancer biology, host-pathogen interactions, metabolic diseases.

Module 2:

Biological Networks:
Definition and classification of biological networks. Gene regulatory networks—nodes as genes, edges as transcriptional regulations; role of transcription factors; network motifs—autoregulatory loops, feedforward loops, toggle switches. Protein-protein interaction networks—physical interaction maps; experimental methods (Y2H, Co-IP, MS); functional associations; databases (BioGRID, STRING, IntAct). Metabolic networks—reactions as edges, metabolites as nodes; enzyme-catalyzed transformations; stoichiometry and flux; metabolic maps from KEGG and MetaCyc. Hierarchical organization—core and peripheral modules; network rewiring in diseases. Static vs dynamic networks. Edge directionality and weight. Cross-talk among networks—gene-protein-metabolite layers. Visualization tools—Cytoscape, Gephi.

Module 3:

Network Analysis:
Topological analysis of biological networks. Degree distribution—scale-free vs random networks; biological relevance of scale-freeness. Hubs—high-degree nodes, functional essentiality, drug targets. Motifs—recurrent subgraph patterns, biological logic gates, network building blocks. Network centrality measures—degree, betweenness, closeness, eigenvector; implications in signaling pathways and disease modules. Clustering coefficient, modularity, community structure—identification of functional modules and biological pathways. Network diameter, path length, shortest path algorithms. Network robustness and fragility—effect of node removal. Comparison of real-world biological networks with theoretical models—Erdős–Rényi, Barabási–Albert. Edge weighting, confidence scores. Multi-layer networks and integration across biological levels.

Module 4:

Pathway Databases:
Biological pathways as curated representations of cellular processes. KEGG—Kyoto Encyclopedia of Genes and Genomes; pathway maps, enzyme classifications, ortholog groupings. Reactome—curated human pathways;

reactions, complexes, biological processes; data interoperability, SBML export. BioCyc—comprehensive collection of Pathway/Genome Databases (PGDBs); MetaCyc as a reference; organism-specific pathways. Use of databases for annotation, enrichment analysis, comparative pathway mapping. Visualization and integration tools—Pathway Tools, KEGG Mapper, Cytoscape plug-ins. Mapping omics data onto pathways. Role of pathway databases in disease understanding, biomarker discovery, drug targeting. Limitations and updates; versioning, annotation quality.

Module 5:

Modeling and Simulation:
Modeling approaches in systems biology—qualitative vs quantitative. Boolean models—binary state representation, logic-based simulations, asynchronous/synchronous updating schemes. Application in gene regulatory networks and signal transduction cascades. Dynamic models—differential equations, deterministic vs stochastic simulations, Michaelis–Menten kinetics, mass-action kinetics. Model formulation—state variables, parameters, initial conditions. Parameter estimation and sensitivity analysis. Software tools—COPASI, CellDesigner, BioNetGen, MATLAB, SBML, PySB. Time-course simulations and steady-state analysis. Visualization and interpretation of model outputs. Validation against experimental data. Applications—predictive modeling, in silico perturbations, network rewiring under stress conditions.

Labs / Practicals:

N/A

References:

1. Alon, Uri. An Introduction to Systems Biology: Design Principles of Biological Circuits. Boca Raton: CRC Press.
2. Kitano, Hiroaki. Foundations of Systems Biology. Cambridge, MA: MIT Press.
3. Barabási, Albert-László. Network Science. Cambridge: Cambridge University Press.
4. Palsson, Bernhard O. Systems Biology: Constraint-Based Reconstruction and Analysis. Cambridge: Cambridge University Press.
5. Boccaletti, Stefano, Vito Latora, Yamir Moreno, and Massimo Chavez. Complex Networks: Structure and Dynamics. Oxford: Oxford University Press.
6. Junker, Björn H., and Falk Schreiber. Analysis of Biological Networks. Hoboken, NJ: Wiley-Interscience.

Course Code	Course Name	L	T	P	Cr
DCSE250041	Fundamentals of Database Systems	3	0	2	4

Course Outcomes:

CO1: Explain database concepts, architecture, data models, ER/EER modeling, and construct ER/EER diagrams.
CO2: Apply relational model concepts and relational algebra operations to formulate database queries.
CO3: Develop SQL queries for data definition, manipulation, joins, nested queries, and set operations.
CO4: Perform ER/EER to relational mapping and apply normalization techniques up to BCNF using functional dependencies.
CO5: Analyze transaction management, concurrency control mechanisms, and recovery techniques in database systems.

Module 1:

Introduction to databases: Characteristics of the database approach, advantages over file-based systems. Database system architecture, data independence, and data abstraction. Data models, Entity-Relationship (ER) modeling – entity types, attributes, relationships, constraints, and ER diagrams.
Enhanced ER (EER) model concepts – specialization, generalization, and aggregation.

Module 2:

Relational model concepts: Schema, tuple, attribute, domain, keys, and integrity constraints.
Relational Algebra operations: Select, Project, Join, Union, Intersection, Set Difference, Cartesian Product, Rename.
Understanding expressions and queries using relational algebra.

Module 3:

SQL basics: Data Definition Language (DDL), data types, and constraints (Primary Key, Foreign Key, Not Null, Unique, Check).
Data Manipulation Language (DML): Retrieval using SELECT, conditions, ordering, and grouping.
INSERT, UPDATE, DELETE statements.
Nested queries, joins (inner, outer), and set operations.
Creating and managing tables.

Module 4:

ER/EER to relational mapping: Steps and examples.
Functional dependencies and inference rules.
Normalization: First Normal Form (1NF), Second (2NF), Third (3NF), and Boyce-Codd Normal Form (BCNF).

Module 5:

Introduction to transaction processing: Concept of transaction, ACID properties.
Concurrency control: Locking protocols, deadlock handling, and prevention, Database recovery techniques.

Labs / Practicals:

Consider two tables: Student(StudentID, Name, Age, DeptID, Marks)
Departments(DeptID, Department, HOD)

1. Write an SQL query to display all the records from the Students table.
2. Write a query to display the names and marks of students who scored more than 80 marks.
3. Write a query to display the list of students who belong to the 'IT' department and are younger than 22.

4. Write a query to list all students whose names start with the letter 'A'.
5. Write an SQL query to display students ordered by their marks in descending order.
6. Write a query to find students who scored marks between 70 and 90.
7. Write an SQL query to display student names and corresponding HOD names using a JOIN on the Department column.
8. Write a query to list students from either 'CSE' or 'IT' departments.
9. Write a query to display the average marks scored by students in each department using GROUP BY.
10. Write a query to display departments where the average student marks are greater than 80 using GROUP BY and HAVING clause.

References:

1. A.K. Majumdar, P. Bhattacharyya, Database Management Systems, Tata McGra-Hill, 1996
2. H. Korth, A.Silberschatz, Database System Concepts, McGraw-Hill (Second Edition), 1991
3. R. Elmasri, S. Navathe, Fundamentals of Database System, Benjamin Cummings (Second Edition),. 1994
4. Bipin Desai, An Introduction to Database Systems, Galgotai Publication (West Publishing), 1991
5. F. Mc-Fadden, J. Hoffer, Modern Database Management, Benjamin cummings (Narosa), (Fourth Edition), 1994

Course Code	Course Name	L	T	P	Cr
DCSE250056	Data Structures and Algorithms	3	0	2	4

Course Outcomes:

1)Classify Data structures
2)Describe Linear Data Structures
3)Explain Non-Linear Data Structures
4)Explain basic algorithmic concepts and recursion
5)Apply different Sorting and Searching Algorithms

Module 1:

Introduction to Data Structures:Basic Terminology, Classification of Data Structures, Operations on Data Structures

Module 2:

Linear Data Structures: Arrays: Introduction to Arrays, Representation in Memory, Operations on an Array, Two Dimensional Arrays
Linked Lists: Singly Linked List, Representation in Memory, Operations on a Single Linked List, Circular Linked Lists, Doubly Linked Lists.
Stacks: Introduction to Stacks, Array Representation of Stacks, Operations on a Stack, Linked List Representation and Operations of Stack, Applications of Stacks-Infix-to-Postfix Transformation, evaluating Postfix Expressions.
Queues: Introduction to Queues, Array Representation of Queues, Operations on a Queue,
Linked List Representation and Operations of Queue, Types of Queues-DeQueue, Circular Queue, Applications of Queues-Round Robin Algorithm

Module 3:

Non Linear Data Structures:
Trees: Basic Terminologies, Definition and Concepts of Binary Trees, Representations of a Binary Tree using Arrays and Linked Lists, Operations on a Binary Tree-Insertion, Deletion, Traversals, Types of Binary Trees.
Graphs: Graph Terminologies, Representation of Graphs- Set, List, Matrix, Graph Traversals

Module 4:

Introduction to Algorithms:
Algorithms and flow charts, Time & Space complexity (definition only)
Recursion: Basic concepts and examples of recursion e.g. factorial problem, Fibonacci sequence.

Module 5:

Sorting & Searching Algorithms: Sorting Algorithms: Algorithms and their analysis (time and space) — Bubble sort, Selection Sort, Insertion Sort, Quick Sort, Merge Sort, Heap sort and Radix Sort
Searching Algorithms: Linear search — Binary search –Concept of Hashing.

Labs / Practicals:

1.Write a program to implement array
2.Write a program to add two matrices using two dimensional arrays
3.Write a program using recursive and non-recursive functions to perform search operation in a given list of integers using linear search technique
4.Write a program to implement search operation in a given list of integers using binary search technique

5.Write a program to implement insertion sorting for a given random data
6.Write a program to implement bubble sorting for a given random data
7.Write a program to implement quick sorting for a given random data
8.Write a program to implement selection sorting for a given random data.

References:

1)Data Structures, R.S. Salaria, Khanna Book Publishing, New Delhi
2)Data Structures Using C, Reema Thareja, Oxford University Press India.
3)Classic Data Structures, SamantaDebasis, Prentice Hall of India.
4)Fundamentals of Data Structure in C, Horowitz, Ellis, Sahni, Sartaj, AndersonFreed, Susan, University Press, India.
5)Data Structures: A Pseudo code approach with C, Richard F. Gilberg, Behrouz A. Forouzan, CENGAGE Learning, India.
6)Data Structures and Algorithms: Concepts, Techniques and Applications, G. A. V. Pai, McGraw- Hill Education, India.
7)Introduction to Algorithms, T.H. Cormen, C.E. Leiserson, R. L. Rivest, C. Stein, MIT Press

Course Code	Course Name	L	T	P	Cr
DCSE250079	Fundamentals of Programming in Python	2	0	4	4

Course Outcomes:

CO1: Develop algorithmic solutions to solve simple and well-defined computational problems effectively.
CO2: Write and demonstrate Python programs using basic statements, expressions, and syntax constructs.
CO3: Explain and implement control flow and function concepts in Python to solve programming tasks.
CO4: Apply Python data structures, including lists, tuples, and dictionaries, for managing compound data.
CO5: Describe and use file handling, exceptions, modules, and packages in Python-based problem-solving.

Module 1:

Python Introduction, Technical Strength of Python, Introduction to Python Interpreter and program execution, Using Comments, Literals, Constants, Python's Built-in Data types, Numbers (Integers, Floats, Complex Numbers, Real, Sets), Strings (Slicing, Indexing, Concatenation, other operations on Strings), Accepting input from Console, printing statements, Simple 'Python' programs. Flow Chart Symbols, Basic algorithms/flowcharts for sequential processing, decision based processing and iterative processing.

Module 2:

Assignment statement, expressions, Arithmetic, Relational, Logical, Bitwise operators and their precedence, Conditional statements: if, if-else, if-elif-else; simple programs, Notion of iterative computation and control flow –range function, While Statement, For loop, break statement, Continue Statement, Pass statement, else, assert. Lists, tuples, and dictionaries—slicing, indexing, concatenation, and mutability. Operations such as finding max, min, mean, linear search, and frequency count using dictionaries.

Module 3:

Top-down approach of problem solving, Modular programming and functions, Function parameters, Local variables, the Return statement, DocStrings, global statement, Default argument values, keyword arguments, VarArgs parameters. Library function-input(), eval(),print(), Recursion.

Module 4:

String Functions: count(), find(), rfind(), capitalize(), title(), lower(), upper(), swapcase(), islower(), isupper(), istitle(), replace(), strip(), lstrip(), rstrip(), aplit(), partition(), join(), isspace(), isalpha(), isdigit(), isalnum(), startswith(), endswith(), encode(), decode(), String: Slicing, Membership, Pattern Matching, Numeric Functions: eval(), max(), min(), pow(), round(), int(), random(), ceil(), floor(), sqrt(), Date & Time Functions.

Module 5:

File opening in various modes and closing of a file, Reading from a file, Writing onto a file, File functions-open(), close(), read(), readline(), readlines(),write(), writelines(),tell(),seek(), Command Line arguments.

Labs / Practicals:

1. Write a program to take name and age from user and display it.
2. Write a program illustrating the various operators in python (arithmetic, relational, logical, and bitwise operators).
3. Write a program to check the data type of given variables.
4. Write a program to swap two variables.
5. Write a program to check whether a number is even or odd.
6. Write a program to find the largest among three numbers.
7. Write a program to check if a number is positive, negative or zero.
8. Write a program to check whether a year is a leap year or not.
9. Write a program to calculate grade based on marks.
10. Write a program to print the multiplication table of a number.
11. Write a program to find the factorial of a number using a loop.

12. Write a program to display Fibonacci series up to n terms.
13. Write a program to find the sum of digits of a number.
14. Write a program to print all prime numbers in a given range
15. Write a Python function to produce pyramid triangles.
16. Write a program to count vowels and consonants in a string.
17. Write a program to reverse a string.
18. Write a program to check whether a string is a palindrome.
19. Write a program to find the length of a string without using len().
20. Write a program to remove all punctuation from a string.
21. Write a program to find the largest and smallest number in a list.
22. Write a program to add elements to a list and sort them.
23. Write a program to find the sum and average of list elements.
24. Write a program to find duplicates in a list.
25. Write a program to convert a list to a tuple and vice versa.
26. Write a function to calculate factorial of a number.
27. Write a function to check whether a number is prime.
28. Write a function that returns the square of a number.
29. Write a program with a function to count even and odd numbers in a list.
30. Write a recursive function to find the nth Fibonacci number.
31. Write a function that takes the lengths of three sides: side1, side2 and side3 of the triangle as the input from the user using input function and return the area of the triangle as the output. Also, assert that sum of the length of any two sides is greater than the third side.
32. Write a program to create and write to a text file.
33. Write a program to read contents of a file and display it.
34. Write a program to count the number of lines and words in a file.
35. Write a program to append data to an existing file.
36. Write a program to copy contents from one file to anothe

References:

1. Python Network Programming Cookbook by Pradeeban Kathiravelu, Dr. M. O. Faruque Sarkar, PACKT.
2. Head First Python by Paul Berry, O'Reilly
3. Dive into Python by Mark Pilgrim, APress
4. Beginning Programming with Python Dummies by John Paul Meuller.

Course Code	Course Name	L	T	P	Cr
DCSE250086	Health Care and Data Analytics	3	1	0	4

Course Outcomes:

CO1. Describe the basics of healthcare data analytics.
CO2. Use biomedical image features and perform data analytics.
CO3. Apply natural language processing for data analytics
CO4. Use prediction models for healthcare data analytics.
CO5. Interpret genomic and clinical data for healthcare data analytics.
CO6. Explain data analytics applications for healthcare.

Module 1:

Introduction to Healthcare Data Analytics- Electronic Health Records– Components of EHR- Coding Systems- Benefits of EHR- Barrier to Adopting HER, Challenges- Phenotyping Algorithms.

Module 2:

Biomedical Image Analysis
Biomedical image modalities, Object detection, image segmentation, image registration, feature extraction
Genomic Data Analysis for Personalized Medicine
Genomic data generation, methods for data analysis, types of computational genomics studies towards personalized medicine.

Module 3:

Data Analysis using Natural language Processing for healthcare.
Natural Language Processing, Mining information from Clinical Text, challenges of processing clinical reports, applications.

Module 4:

Clinical Prediction Models and Data Mining for Healthcare data
Basic Statistical Prediction Models, Alternative Clinical Prediction Models, Survival Models. Association Analysis, Temporal Pattern Mining: Sequential Pattern Mining, Time-Interval Pattern Mining.

Module 5:

Visual Analytics and Integrating data for Healthcare
Medical data visualization, visual analytics for public health and population research, clinical workflow, clinicians, patients.

Labs / Practicals:

NA

References:

1. Chandan K. Reddy and Charu C Aggarwal, “Healthcare data analytics”, Taylor & Francis, 2015
2. Hui Yang and Eva K. Lee, “Healthcare Analytics: From Data to Knowledge to Healthcare Improvement, Wiley, 2016.
3. Tinglong Dai, Sridhar Tayur , “Handbook of Healthcare Analytics”, Wiley, 2018.
4. Anand J Kulkarni, Patrick Siarry,”Big Data Analytics in healthcare”,Springer,Studies in Big Data

Course Code	Course Name	L	T	P	Cr
DCSE250102	Operating Systems	3	0	2	4

Course Outcomes:

CO1: Understand the basics of operating systems like kernel, shell, types and services of operating systems.
CO2:Understand the concept of program, process and thread and analyse various CPU scheduling Algorithms.
CO3: Describe and analyse the memory management and its allocation policies.
CO4: Understand the issues related to file system interface and implementation.
CO5: Understand disk management and explain disk scheduling algorithms for better utilization of external memory.
6: Configure OS in an efficient and secure manner.

Module 1:

Introduction: Overview of Operating System, basic concepts, UNIX/LINUX Architecture, Kernel, services and systems calls, system programs.

Module 2:

Process Management and Memory management: Process Management: Process concepts, operations on processes, IPC, Process Scheduling, Multithreaded programming. Memory management: Memory allocation, Swapping, Paging, Segmentation, Virtual Memory, various faults.

Module 3:

File management: Concept of a file, access methods, directory structure, file system mounting, file sharing and protection, file system structure and implementation, directory implementation, free space management, efficiency and performance. Different types of file systems.

Module 4:

I/O System: Mass storage structure - overview, disk structure, disk attachment, disk scheduling algorithms, swap space management, RAID types.

Module 5:

OS Security: Authentication, Access Control, Access Rights, System Logs

Labs / Practicals:

1. Revision practice of various commands like man, cp, mv, ln, rm, unlink, mkdir, rmdir, etc and many more that were learnt in IT Workshop course and later.
2. Implement two way process communication using pipes.
3. Implement message queue form of IPC
4. Implement shared memory and semaphore form of IPC
5. Simulate the CPU scheduling algorithms - Round Robin, SJF, FCFS, priority
6. Simulate all FIFO Page Replacement Algorithm using C program
7. Simulate all LRU Page Replacement Algorithms using C program
8. Simulate Paging Technique of Memory Management
9. Practice various commands/utilitiessuch as catnl, uniq, tee, pg, comm, cmp, diff, tr, tar, cpio, mount, umount, find, umask, ulimit, sort, grep, egrep,fgrep cut, paste, join, du, df , ps, who, etc and many more.

References:

1. "Operating System Concepts (Tenth Edition)" by Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne
https://os-book.com/OS10/index.html

2. "Operating Systems: Internals and Design Principles (Ninth Edition)" by William Stallings
http://williamstallings.com/OperatingSystems

3. "Modern Operating Systems (Fifth Edition)" by Andrew Stuart Tanenbaum and Herbert Bos
https://dl.acm.org/doi/10.5555/2655363

4. "Operating System Design: The Xinu Approach (Second Edition)" by Douglas Earl Comer
https://www.oreilly.com/library/view/operating-system-design/9781498712439

5. "The Design of the UNIX Operating System" by Maurice J. Bach
https://dl.acm.org/doi/10.5555/8570

6. "Advanced Programming in the UNIX Environment (Third Edition)" by William Richard Stevens and Stephen A. Rago
http://www.apuebook.com/apue3e.html

7. "Beej's Guide to Interprocess Communication" by Brian Hall
https://beej.us/guide/bgipc/pdf/bgipc_a4_c_2.pdf

Course Code	Course Name	L	T	P	Cr
DOAI250005	Artificial Intelligence and Machine Learning	2	0	4	4

Course Outcomes:

CO1: Understand AI problem-solving techniques, heuristic search algorithms, and reinforcement learning.
CO2: Apply knowledge representation, logic-based reasoning, and probabilistic models in AI systems.
CO3: Develop and evaluate machine learning models using supervised and unsupervised learning techniques.
CO4: Design and implement deep learning architectures for vision, language, and sequential data applications.
CO5: Analyze advanced AI methods, ethical challenges, and emerging trends in artificial intelligence.

Module 1:

AI Techniques and Search Strategies
History of AI, problem-solving using search, uninformed vs. informed search, heuristic search techniques (Hill Climbing, Simulated Annealing, A* Algorithm), adversarial search (Minimax Algorithm, Alpha-Beta Pruning), AI-driven game strategies (Chess, Chinese Checkers), Reinforcement Learning fundamentals (Markov Decision Processes, Q-learning), heuristic optimization techniques (Genetic Algorithms, Particle Swarm Optimization).

Module 2:

Knowledge Representation, Reasoning, and Probabilistic Models
Propositional logic, inference mechanisms, forward and backward chaining, probabilistic reasoning techniques (Bayesian Networks, Markov Models, Probabilistic Graphical Models), logic-based AI vs. machine learning-driven reasoning, Natural Language Processing (NLP) fundamentals (tokenization, embeddings, language models), AI-driven text processing, machine translation.

Module 3:

Machine Learning Fundamentals and Neural Networks
Supervised and unsupervised learning paradigms, classification and regression techniques (decision trees, support vector machines, ensemble learning methods: Random Forest, XGBoost), artificial neural networks (activation functions, loss functions, back propagation, multilayer perceptions), overfitting, bias-variance tradeoff, hyperparameter tuning, performance evaluation metrics (precision-recall, ROC curves, F1-score).

Module 4:

Deep Learning and Advanced Neural Architectures
Convolutional Neural Networks (CNNs) for image classification and object detection (convolution layers, pooling, fully connected layers), Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks for sequential data processing, Transformer models (BERT, GPT) for NLP applications, optimization techniques (batch normalization, dropout regularization, transfer learning), deep learning applications (medical imaging, self-driving vehicles, generative AI).

Module 5:

Unsupervised Learning, Generative Models, and Ethical AI
Clustering techniques (K-Means, Hierarchical Clustering), Autoencoders, Generative Adversarial Networks (GANs), self-supervised learning, foundation models in AI, AI applications in healthcare, finance, and autonomous systems, AI ethics (fairness, bias mitigation, adversarial attacks, explainable AI), advancements in AI (quantum computing, federated learning, AI governance frameworks).

Labs / Practicals:

1. Implement a basic uninformed search algorithm (e.g., Breadth-First Search, Depth-First Search).
2. Implement informed search algorithms using heuristics (e.g., A* Algorithm for pathfinding).
3. Implement the Hill Climbing algorithm for local search problems.
4. Develop a solution using Simulated Annealing for optimization problems.
5. Implement the Minimax algorithm to solve a simple game (e.g., Tic-Tac-Toe).
6. Implement Alpha-Beta Pruning for optimizing the Minimax algorithm.
7. Build a Reinforcement Learning agent using Q-learning to solve a basic maze problem.
8. Implement Genetic Algorithms to solve optimization problems (e.g., travelling salesman).
9. Build a Particle Swarm Optimization model for function optimization.
10. Implement propositional logic and inference mechanisms (e.g., forward and backward chaining).
11. Develop a basic Bayesian Network for probabilistic reasoning.

References:

1.Tom M. Mitchell: Machine Learning. McGraw Hill
2.Artificial Intelligence: A Modern Approach by Stuart Russell and Peter Norvig
3.Pattern Recognition and Machine Learning by Christopher M. Bishop
4.Machine Learning: A Probabilistic Perspective by Kevin P. Murphy
5.Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto
6.Natural Language Processing with Python by Steven Bird, Ewan Klein, and Edward Loper
7.The Elements of Statistical Learning" by Trevor Hastie, Robert Tibshirani, and Jerome Friedman
8.Michael Nielsen: Neural Networks and Deep Learning (free online book)
9.Stuart Russel & Peter Norvig: Artificial Intelligence– A Modern Approach. Pearson, 4th edition, 2023
10.Deep Learning (Adaptive Computation and Machine Learning series) Illustrated Edition, by Ian Goodfellow, Yoshua Bengio, Aaron Courville, MIT Press, London

Course Code	Course Name	L	T	P	Cr
DOAI250009	Big Data Analytics	3	0	2	4

Course Outcomes:

CO1: Describe big data characteristics, challenges, and applications across industries and domains.

CO2: Evaluate and apply NoSQL databases and distributed storage architectures for big data management.

CO3: Install, configure, and operate Hadoop and HDFS for batch-oriented big data processing.

CO4: Develop and implement data analysis workflows using Hadoop ecosystem tools including Hive, Pig, and HBase.

CO5: Design and implement real-time data processing and analytics using Spark Streaming and PySpark.

CO6: Apply visualization and regression techniques for analytical insights from large-scale data.

Module 1:

Introduction to Big Data Ecosystem

Big Data Characteristics: Volume, Variety, Velocity, Veracity, and Value, Business applications and case studies of big data analytics, Challenges in conventional systems, Big Data Analytics Lifecycle, Tools and Technologies in Big Data, Hadoop vs. Traditional Systems, NoSQL Overview (Key-Value, Document, Columnar, Graph databases), Basics of MongoDB and Cassandra for big data storage.

Module 2:

Hadoop Framework for Big Data

History and Evolution of Hadoop, Hadoop Distributed File System (HDFS): Architecture, Internals, and Operations, Block-level Storage, Fault Tolerance, MapReduce Paradigm: Concepts, Anatomy of a MapReduce Job, Failures, Scheduling, Shuffle, Sort, and Task Execution, Input / Output Formats in MapReduce, Hadoop Streaming and Integrations, Developing MapReduce Applications with Java and Python APIs.

Module 3:

Data Analysis with Hadoop Ecosystem

Hive: Data Modeling, Data Types, File Formats, HiveQL (DDL, DML, DQL), Querying Structured Data, Hive Services and Optimization.
Pig: Data Flow Language, Pig Latin Programming, Grunt Shell, Data Models, Operators, Joins, Developing and Testing Pig Scripts.
HBase: Architecture, Data Model, Integration with Hadoop, Use Cases.
Introduction to Data Lakes using Hadoop.
Hands-on with Hadoop Ecosystem for ETL pipelines.

Module 4:

Real-Time Big Data Analytics and Streaming

Stream Processing Fundamentals, Stream Data Models, Stream Architecture, Concepts of Sampling, Filtering, Counting in Streams, Real-time Sentiment Analysis, Stock Market Analysis using Streams, Introduction to Apache Kafka for Streaming Data Ingestion, Spark Basics: RDD, DataFrames, Spark SQL, In-Memory Computing, PySpark APIs, Spark Streaming for Real-Time Processing, Real-Time Analytics Applications, Case

Studies.

Module 5:

Analytical Models and Visualization for Big Data

Regression Techniques: Simple Linear Regression, Multiple Linear Regression, Model Interpretation for Large-Scale Data, Feature Engineering for Big Data, Visualization Techniques: Interactive Dashboards, Graphical Tools for Big Data (Tableau / PowerBI concepts), Visual Analytics: Trends, Patterns, Correlations, Interaction Models, Real-World Applications of Big Data Visualizations.

Labs / Practicals:

1. Setting up Hadoop, HDFS, and YARN on local and cloud-based environments (AWS / Azure / GCP).

2. Developing basic MapReduce jobs using Java and Python.

3. Implementing ETL pipelines using Hive and Pig.

4. Performing analytics with HBase and Hive.

5. Real-time data streaming with Apache Kafka and Spark Streaming (using PySpark).

6. Implementing basic ML pipelines over Spark (MLlib).

7. Visualizing Big Data using interactive tools (Matplotlib, Plotly, Tableau / PowerBI).

8. Group Project: Building and deploying an end-to-end Big Data Analytics pipeline on a real-world dataset.

References:

1. Tom White, Hadoop: The Definitive Guide, 4th Edition, O'Reilly Media, 2015.

2. Anand Rajaraman, Jeffrey David Ullman, Mining of Massive Datasets, Cambridge University Press, 3rd Edition, 2020.

3. Bill Franks, Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics, John Wiley & Sons, 2012.

4. Jiawei Han, Micheline Kamber, Jian Pei, Data Mining: Concepts and Techniques, 3rd Edition, Morgan Kaufmann / Elsevier, 2011.

5. Holden Karau, Andy Konwinski, Patrick Wendell, Matei Zaharia, Learning Spark: Lightning-Fast Big Data Analysis, 2nd Edition, O'Reilly Media, 2020.

6. Paul Zikopoulos, Dirk deRoos, Krishnan Parasuraman, Thomas Deutsch, James Giles, David Corrigan, Harness the Power of Big Data – The IBM Big Data Platform, Tata McGraw Hill, 2012.

7. Michael Berthold, David J. Hand, Intelligent Data Analysis, Springer, 2007.

8. Da Ruan, Guoqing Chen, Etienne E. Kerre, Geert Wets, Intelligent Data Mining, Springer, 2007.

Official Documentation (Highly Recommended for Labs / Practical Work):

9. Official documentation for Hadoop: https://hadoop.apache.org/

10. Official documentation for Spark: https://spark.apache.org/docs/latest/

11. Official documentation for Hive: https://cwiki.apache.org/confluence/display/Hive/Home

12. Official documentation for Pig: https://pig.apache.org/

13. Official documentation for HBase: https://hbase.apache.org/

14. Official documentation for Kafka: https://kafka.apache.org/documentation/

15. Official documentation for PySpark: https://spark.apache.org/docs/latest/api/python/

Course Code	Course Name	L	T	P	Cr
DOAI250031	Introduction to High Performance Computing	3	1	0	4

Course Outcomes:

CO1: Understand instruction-level parallelism, pipelining techniques, hazards, and compiler/hardware methods for ILP exploitation.
CO2: Analyze multiprocessors, thread-level parallelism, vector architectures, GPU architectures, and CUDA programming concepts.
CO3: Evaluate cache performance, virtual memory, and advanced memory optimization techniques.
CO4: Apply parallel programming platforms (OpenMP, MPI, OpenCL, OpenACC) for performance improvement in AI/ML applications.
CO5: Explain interconnection networks, clusters, SoC interconnects, and NoC architectures for parallel systems.

Module 1:

Introduction to pipelining – Types of pipelining – Hazards in pipelining - Introduction to instruction level parallelism (ILP) – Challenges in ILP - Basic Compiler Techniques for exposing ILP - Reducing Branch costs with prediction - Overcoming Data hazards with Dynamic scheduling - Hardware-based speculation - Exploiting ILP using multiple issue and static scheduling - Exploiting ILP using dynamic scheduling, multiple issue and speculation - Tomasulo's approach, VLIW approach for multi-issue

Module 2:

Introduction to multi processors and thread level parallelism - Characteristics of application domain - Systematic shared memory architecture - Distributed shared – memory architecture – Synchronization
– Multithreading - Multithreading-fined grained and coarse grained, superscalar and super pipelining, hyper threading. Vector architectures; organizations and performance tuning; GPU architecture and internal organization, Elementary concepts in CUDA programming

Module 3:

Introduction to cache performance - Cache Optimizations - Virtual memory - Advanced optimizations of Cache performance - Memory technology and optimizations - Protection: Virtual memory and virtual machines - multi-banked caches, critical word first, early restart approaches, hardware pre-fetching, write buffer merging.

Module 4:

Introduction to parallel computing platforms; (Open MP, MPI, Open CL, Open ACC) with performance improvement analysis done using real-life AI and ML applications

Module 5:

Introduction to inter connection networks and clusters - interconnection network media - practical issues in interconnecting networks- examples - clusters - designing a cluster – System on Chip (SoC) Interconnects – Network on Chip (NOC).

Labs / Practicals:

NA

References:

1. Wang, Endong, Qing Zhang, Bo Shen, Guangyong Zhang, Xiaowei Lu, Qing Wu, and Yajuan Wang. "High-performance computing on the Intel Xeon Phi." Springer 5, 2014.
2. Sanders, Jason, and Edward Kandrot. "CUDA by example: an introduction to general-purpose GPU programming", Addison-Wesley Professional, 2010.
3. Chandra, Rohit, Leo Dagum, David Kohr, Ramesh Menon, Dror Maydan, and Jeff McDonald. Parallel programming in Open MP. Morgan kaufmann, 2001.

4. Kaeli, David R., Perhaad Mistry, Dana Schaa, and Do