

ANNA UNIVERSITY, CHENNAI
NON- AUTONOMOUS COLLEGES
AFFILIATED TO ANNA UNIVERSITY
M.E. BIG DATA ANALYTICS
REGULATIONS 2025

PROGRAMME OUTCOMES (POs):

PO	Programme Outcomes
PO1	An ability to independently carry out research /investigation and development work to solve practical problems
PO2	An ability to write and present a substantial technical report/document.
PO3	Students should be able to demonstrate a degree of mastery over the area as per the specialization of the program. The mastery should be at a level higher than the requirements in the appropriate bachelor program

PROGRAMME SPECIFIC OUTCOMES:

- | |
|---|
| PSO1: Data-Driven Decision Making and Visualization: Apply statistical, machine learning, and deep learning techniques to extract meaningful insights, build predictive models, and create interactive visualizations for informed decision-making in diverse domains. |
| PSO2: Research, Innovation, and Ethical Practice: Conduct research and develop innovative solutions in big data analytics addressing real-world challenges, while ensuring data privacy, security, and ethical use of data in alignment with global standards. |



POSTGRADUATE CURRICULUM (NON-AUTONOMOUS AFFILIATED INSTITUTIONS)

Programme: M.E. Big Data Analytics

Regulations: 2025

Abbreviations:

BS – Basic Science (Mathematics)

L – Laboratory Course

ES – Engineering Science (Programme Core (**PC**),
Programme Elective (**PE**))

T – Theory

SD – Skill Development

LIT – Laboratory Integrated Theory

SL – Self Learning

PW – Project Work

OE – Open Elective

TCP – Total Contact Period(s)

Semester – I

S. No.	Course Code	Course Title	Type	Periods per week			TCP	Credits	Category
				L	T	P			
1.	MA25C07	Advanced Mathematical Methods (CSIE)	T	3	1	0	4	4	BS
2.	CP25C01	Advanced Data Structures and Algorithms	LIT	3	0	4	7	5	ES (PC)
3.	BD25101	Applied Data Science with R Programming	T	3	0	0	3	3	ES (PC)
4.	CP25C03	Advanced Operating Systems	T	3	0	0	3	3	ES (PC)
5.	CP25C04	Advanced Compiler Design	T	3	0	0	3	3	ES (PC)
6.	BD25102	Technical Seminar	-	0	0	2	2	1	SD
Total Credits							22	19	

Semester – II

S. No.	Course Code	Course Title	Type	Periods per week			TCP	Credits	Category
				L	T	P			
1.		Big Data Technologies	LIT	3	0	2	5	4	ES (PC)
2.		Artificial Intelligence and Machine Learning	LIT	3	0	2	5	4	ES (PC)
3.		Python for e-Governance Practices	LIT	3	0	2	5	4	ES (PC)
4.		Programme Elective I	T	3	0	0	3	3	ES (PE)
5.		Quantum Computing	T	2	0	0	2	2	ES (PC)
6.		Industry-Oriented Course I	-	1	0	0	1	1	SD
7.		Industrial Training	-	-	-	-	-	2	SD
8.		Self-Learning Course	-	-	-	-	-	1	-
Total Credits							21	21	

Semester – III

S. No.	Course Code	Course Title	Type	Periods per week			TCP	Credits	Category
				L	T	P			
1.		Programme Elective II	T	3	0	0	3	3	ES (PE)
2.		Programme Elective III	T	3	0	0	3	3	ES (PE)
3.		Programme Elective IV	T	3	0	0	3	3	ES (PE)
4.		Open Elective	--	3	0	0	3	3	-
5.		Industry-Oriented Course II	--	1	0	0	1	1	SD
6.		Project Work I	-	0	0	12	12	6	SD
Total Credits							25	19	

Semester – IV

S. No.	Course Code	Course Title	Type	Periods per week			TCP	Credits	Category
				L	T	P			
1.		Project Work II	-	0	0	24	24	12	SD
Total Credits							24	12	

PROGRAMME ELECTIVE COURSES (PE)

S. No.	Course Code	Course Title	Periods per week			Total Contact Periods	Credits
			L	T	P		
1.		Business Statistics and Data Modelling	3	0	0	3	3
2.		Prompt Engineering	3	0	0	3	3
3.		Digital Marketing Analytics	3	0	0	3	3
4.		Web Analytics and development	3	0	0	3	3
5.		Optimization Techniques for Analytics	3	0	0	3	3
6.		Data Lake Architecture and Applications	3	0	0	3	3
7.		Data Modelling and Integration	3	0	0	3	3
8.		Data Acquisition and Production Management Practices	3	0	0	3	3
9.		Quantum Machine Learning	3	0	0	3	3
10.		Edge and Fog Computing	3	0	0	3	3
11.		Data Intensive Computing	3	0	0	3	3
12.		Deep Learning ecosystem	3	0	0	3	3
13.		Cognitive Computing	3	0	0	3	3
14.		Mixed Reality	3	0	0	3	3
15.		Data Manipulation and Visualization	3	0	0	3	3
16.		Exploratory Data Analysis	3	0	0	3	3
17.		Geospatial analytics	3	0	0	3	3
18.		Federated Learning	3	0	0	3	3
19.		Vibe Coding	3	0	0	3	3
20.		Agentic AI	3	0	0	3	3

Semester I

MA25C07	Advanced Mathematical Methods (CSIE)	L	T	P	C
		3	1	0	4
Course Objectives: • Develop an in-depth understanding of advanced concepts in linear algebra, multivariate analysis, and number theory for computer science applications. • Apply mathematical tools such as eigenvalue decomposition, SVD, and multivariate statistical methods to real-world computing and data-driven problems. • Analyze and implement number-theoretic techniques for cryptography, security, and algorithmic problem-solving in computer science.					
Linear Algebra: Vector spaces, norms, Inner Products, Eigenvalues using QR transformations, QR factorization, generalized eigenvectors, Canonical forms, singular value decomposition and applications, pseudo inverse, least square approximations.					
Multivariate Analysis: Random vectors and matrices, Mean vectors and covariance matrices, Multivariate normal density and its properties, Principal components, Population principal components, Principal components from standardized variables.					
Elementary Number Theory: The division algorithm, Divisibility and the Euclidean algorithm, The fundamental theorem of arithmetic, Modular arithmetic and basic properties of congruences; Principles of mathematical induction and well ordering principle. Primality Testing algorithms, Chinese Remainder Theorem, Quadratic Congruence.					
Advanced Number Theory: Advanced Number Theory, Primality Testing algorithms, Chinese Remainder Theorem, Quadratic Congruence, Discrete Logarithm, Factorization Methods, Side Channel Attacks, Shannon Theory, Perfect Secrecy, Semantic Security.					
Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%.					
Assessment Methodology: Assignments (15), Quiz (10), Virtual Demo (20), Flipped Class Room (10), Review of Gate and IES Questions (25), Project (20).					
References: 1. Gilbert Strang, Linear Algebra and Its Applications, Cengage Learning. 2. Richard A. Johnson & Dean W. Wichern, Applied Multivariate Statistical Analysis, Pearson. 3. Neal Koblitz, A Course in Number Theory and Cryptography, Springer. 4. Victor Shoup, A Computational Introduction to Number Theory and Algebra, Cambridge University Press.					

E-resources:

1. <https://ocw.mit.edu/courses/18-06-linear-algebra>
2. <https://nptel.ac.in/courses/111105041>
3. <https://crypto.stanford.edu/pbc/notes/numbertheory>

CP25C01	Advanced Data Structures and Algorithms	L	T	P	C
		3	0	4	5
Course Objectives: - To explore advanced linear, tree, and graph data structures and their applications. - To design efficient algorithms using appropriate algorithmic paradigms. - To evaluate computational complexity and identify tractable vs. intractable problems.					
Linear Data Structures and Memory Optimization: Advanced arrays: Sparse arrays, dynamic arrays, cache-aware structures, Linked lists: Skip lists, unrolled linked lists, XOR linked lists, Stacks and Queues: Priority queues, double-ended queues, circular buffers, Hashing: Perfect hashing, cuckoo hashing, extendible hashing.					
Practical: - Implement skip lists and measure performance compared with balanced BST. - Experiment with cache-aware data structures and analyze memory utilization.					
Advanced Tree Data Structures: Balanced Trees: AVL, Red-Black Trees, Splay Trees, Treaps, Multi-way Trees: B-Trees, B+ Trees, R-Trees, Segment Trees, Fenwick Trees, Suffix Trees and Tries for string processing, Applications in indexing, text retrieval, computational geometry.					
Practical: - Implement B+ tree for database indexing use-case. - Design a suffix tree-based algorithm for DNA sequence matching.					
Graph Data Structures and Algorithms: Representation: Adjacency list/matrix, incidence matrix, compressed storage, Traversals: DFS, BFS with applications, Shortest Path Algorithms: Dijkstra, Bellman-Ford, Floyd-Warshall, Johnson’s algorithm, Minimum Spanning Trees: Prim’s, Kruskal’s, Borůvka’s algorithm, Network Flow Algorithms: Ford-Fulkerson, Edmonds-Karp, Push-Relabel.					
Practical: - Implement Johnson’s algorithm for sparse graph shortest paths. - Demonstration of Maximum flow in traffic or network routing simulation.					
Algorithm Design and Paradigms: Divide and Conquer: Karatsuba’s multiplication, Strassen’s algorithm, Greedy Methods: Huffman coding, interval scheduling, set cover approximation, Dynamic Programming: Matrix chain multiplication, Floyd-Warshall, knapsack variants, Backtracking and Branch-and-Bound, Randomized Algorithms and Probabilistic Analysis.					
Practical: - Implement Strassen’s algorithm and compare with naive matrix multiplication. - Develop a randomized algorithm for primality testing (Miller–Rabin).					

Computational Complexity and Approximation Algorithms: Complexity Classes: P, NP, NP-Complete, NP-Hard, Reductions: Polynomial-time reductions, Cook-Levin theorem (overview), Approximation Algorithms: Vertex cover, set cover, TSP, k-center problem, Heuristic Algorithms: Local search, simulated annealing, genetic algorithms.

Practical:

- Implement approximation algorithm for vertex cover.
- Complexity analysis of a chosen NP-hard problem and implement a heuristic.

Advanced Topics and Emerging Trends: Randomized Algorithms – Monte Carlo Algorithms, Parallel and Distributed Algorithms – PRAM Model, Divide and Conquer in Parallel, Load Balancing, Streaming Algorithms – Data Stream Models, Sketching and Sampling, Frequency Moments, Advanced String Matching – Suffix Trees, Suffix Arrays, Pattern Matching in Linear Time.

Practical:

- Implement randomized and streaming algorithms on real-world datasets.
- Design of parallel and distributed algorithms.

Weightage: Continuous Assessment: 50%, End Semester Examinations: 50%

Assessment Methodology: Assignments (15), Quiz (10), Virtual Demo (20), Flipped Class Room (10), Review of Gate and IES Questions (25), Project (20)

References:

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms. MIT Press.
2. La Rocca, M. (2021). Advanced algorithms and data structures. Manning Publications.
3. Goodrich, M. T., Tamassia, R., & Mount, D. M. (2011). Data structures and algorithms in C++. John Wiley & Sons, Inc.
4. Weiss, M. A. (2014). Data structures and algorithm analysis in C++. Pearson Education.
5. Drozdek, A. (2013). Data structures and algorithms in C++. Cengage Publications.

E-resources:

1. <https://www.theiotacademy.co/blog/data-structures-and-algorithms-in-c/>
2. https://github.com/afriid18/Data_structures_and_algorithms_in_cpp
3. <https://www.udemy.com/course/introduction-to-algorithms-and-data-structures-in-c/?srsltid=AfmBOorEZlkgV7QzaEh6lqzAaKLjC-lpFU1NGgWFOHMLhOos-uDVKjCK>

	Description of CO	PO	PSO
CO1	Describe data structures and implement algorithmic solutions for complex computational problems.	--	--

	Description of CO	PO	PSO
CO2	Analyze the time complexity and efficiency of algorithms for various computing problems.	PO1(3)	PSO1(3)
CO3	Evaluate algorithmic techniques and data structures to determine their suitability for different applications.	PO3(2)	PSO2(2)
CO4	Design optimized solutions for real-world problems using appropriate algorithms and data structures.	PO2(1)	PSO1(3)

BD25101	Applied Data Science with R Programming	L	T	P	C
		3	0	0	3
Course Objectives: • To equip students with the ability to apply R programming in real-world data science projects. • To enable learners to implement statistical methods, machine learning models, and exploratory data analysis using R. • To develop proficiency in handling datasets and effectively communicating data-driven findings.					
Introduction to R programming: R programming Structures, vectors, matrices and arrays, lists, data frames, reading and writing data, connection Interfaces, subsetting R objects, dplyr package, control structures, Functions, scoping rules of R, Looping functions. Activity: Demonstration of reading data, subsetting using dplyr live in RStudio Cloud.					
Extended Programming in R: Data manipulation techniques - Math and simulations in R, Object-oriented programming, Input and Output, String Manipulation, Debugging, Performance Enhancement, Speed and Memory, Interfacing R to other languages, Parallel R. Activity: Demonstration of R object types with their OOP class systems.					
Linear algebra for data science using R: Linear Combinations - Vector Spaces, Fundamental Theorem of Linear Algebra, Least Squares, Change of Basis, Orthogonality, Eigen Problems. Activity: Visualizations of vector spaces and Orthogonality.					
Statistics for Data Science: Descriptive statistics, Notion of probability, Distributions, Mean, Variance, Covariance, Covariance matrix, understanding Univariate and Multivariate Normal distributions, hypothesis testing, Confidence interval for estimates, Optimization, Typology of data science problems and a solution framework. Activity: Simulate sampling distributions with various sample sizes.					
Statistical Applications using R programming: Time series data handling, power and computation of sample size, advanced data handling, Multiple regression, Survival Analysis, Rates and Poisson Regression, Nonlinear curve fitting, statistical modelling in R, Advanced Non-Linear Modelling and Evaluation: Quantile Regression Method. Activity: Demonstration of Data handling and curve fitting.					
Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%					
Assessment Methodology: Assignments (15), Quiz (10), Virtual Demo (20), Flipped Class Room (10), Review of Gate and IES Questions (25), Project (20).					

References

1. Peng, R. D. (2016). R programming for data science.
2. Mitran, S. Linear algebra for data science. University of North Carolina at Chapel Hill.
3. Bennett, S. R. (2021). Linear algebra for data science with examples in R.
4. Montgomery, D. C., & Runger, G. C. Applied statistics and probability for engineers. John Wiley & Sons.
5. Wickham, H. R for data science. O'Reilly Media.

	Description of CO	PO	PSO
CO1	Apply R programming concepts to manipulate, transform, and visualize data effectively.	--	--
CO2	Analyze linear algebra concepts and statistical methods to solve data science problems using R.	PO1(3)	PSO1(3)
CO3	Evaluate the performance of statistical and machine learning techniques on real-world datasets.	PO3(2)	PSO2(2)
CO4	Design data-driven solutions integrating R programming for complex problems.	PO2(1)	PSO1(3)

CP25C03	Advanced Operating Systems	L	T	P	C
		3	0	0	3
Course Objective: - To analyze the architectures and design issues of advanced operating systems. - To develop the model for process synchronization and recovery in complex environments. - To evaluate algorithms for distributed coordination, resource management, fault tolerance, and security.					
Advanced Process and Thread Management: Multithreading models, thread pools, context switching, Synchronization issues and solutions: semaphores, monitors, lock-free data structures, CPU scheduling in multi-core systems Activity: CPU scheduler simulation for multicore systems.					
Memory and Resource Management in Modern OS: Virtual memory, demand paging, page replacement policies-Huge pages, NUMA-aware memory management-Resource allocation in cloud-native environments Activity: Simulate demand paging and page replacement algorithms.					
Virtualization and Containerization: Hypervisors (Type I & II), KVM, QEMU, Xen-Containers: Docker, LXC, systemd-nspawn-OS-level virtualization and namespaces Activity: Deploy and configure Docker containers with various images.					
Distributed Operating Systems and File Systems: Distributed scheduling, communication, and synchronization-Distributed file systems: NFS, GFS, HDFS-Transparency issues and fault tolerance Activity: Simulate distributed process synchronization.					
Security and Trust in Operating Systems: Access control models: DAC, MAC, RBAC-OS hardening techniques, sandboxing, SELinux, AppArmor-Secure boot, rootkit detection, trusted execution environments Activity: Implement Role-Based Access Control (RBAC) using Linux user and group permissions.					
Real-Time and Embedded Operating Systems: Real-time scheduling algorithms (EDF, RM)-POSIX RT extensions, RTOS architecture-TinyOS, FreeRTOS case studies Activity: Analyze FreeRTOS task scheduling behavior.					
Edge and Cloud OS: Future Paradigms: Serverless OS, unikernels, lightweight OS for edge computing-Mobile OS internals (Android, iOS)-OS for quantum and neuromorphic computing (intro) Activity: Analyze Android’s system architecture using emulator tools.					

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Assignments (15), Quiz (10), Virtual Demo (20), Flipped Class Room (10), Review of Gate and IES Questions (25), Project (20).

References:

1. Tanenbaum, A. S., & Bos, H. (2023). Modern operating systems. Pearson.
2. Buyya, R., et al. (2022). Content delivery networks and emerging operating systems. Springer.
3. Silberschatz, A., Galvin, P. B., & Gagne, G. (2022). Operating system concepts. Wiley.
4. Anderson, T., & Dahlin, M. (2021). Operating systems: Principles and practice. Recursive Books.
5. Arpaci-Dusseau, R. H., & Arpaci-Dusseau, A. C. (2020). Operating systems: Three easy pieces.

E-Resources:

1. Prof. Smruti Ranjan Sarangi, "Advanced Distributed Systems", IIT Delhi, NPTEL, https://onlinecourses.nptel.ac.in/noc22_cs80/preview
2. Prof. Rajiv Misra, "Cloud Computing and Distributed Systems", IIT Patna, NPTEL, <https://nptel.ac.in/courses/106104182>

	Description of CO	PO	PSO
CO1	Describe operating system concepts for memory and resource management.	--	--
CO2	Analyse virtualization and distributed OS mechanisms for scalability and performance.	PO1(3)	PSO1(3)
CO3	Evaluate OS security and resource handling strategies in diverse environments.	PO3(2)	PSO2(2)
CO4	Design innovative OS solutions using modern tools and techniques.	PO2(1)	PSO1(3)

CP25C04	Advanced Compiler Design	L	T	P	C
		3	0	0	3
Course Objectives: - To analyze the theory and principles of modern compiler design and advanced optimization techniques. - To design and implement efficient front-end and back-end compiler components for programming languages. - To evaluate code optimization strategies and runtime environment management in contemporary architectures.					
Intermediate Representations and Control Flow Analysis: Static single assignment (SSA) form- Context-Free Grammer (CFG) construction-dominance relations-Intermediate Representation (IR) design for functional and imperative languages-Static single assignment and def-use chains					
Activities: - Convert source code to SSA form using LLVM IR. - Visualize control flow graphs from SSA using LLVM tools.					
Program Analysis and Transformations: Data flow analysis- live variable analysis-reaching definitions-Alias analysis and dependence analysis-Loop optimizations and transformations					
Activities: - Perform loop unrolling and strength reduction. - Conduct live variable analysis and visualize data flow graphs.					
Advanced Optimizations and Polyhedral Compilation: Polyhedral model for loop nests-Tiling, skewing, fusion, and vectorization-Profile-guided and feedback-directed optimizations					
Activities: - Implement loop tiling and loop skewing on a matrix multiplication program. - Analyze the effect on loop-intensive code with LLVM optimization flags.					
Just-in-Time (JIT) and Runtime Compilation: JIT compilation models: tracing, method-based-GraalVM architecture, Java HotSpot internals-LLVM JIT and dynamic language support					
Activities: - Develop a basic JIT-enabled interpreter with LLVM or GraalVM. - Implement dynamic dispatch using LLVM JIT API.					
Machine Learning in Compiler Design: ML for phase ordering, auto-tuning, and IR prediction-Reinforcement learning for optimization passes-Dataset creation and benchmarking for compiler ML					

Activities:

1. Train an ML model to predict optimization passes.
2. Use reinforcement learning for pass selection in toy compiler.

Domain-Specific Languages (DSLs) and Compiler Extensions: Designing DSLs for AI/ML, DSP, graphics-Code generation for custom accelerators-Integration with TensorFlow XLA and Halide

Activities:

1. Design and test a simple DSL grammar using ANTLR.
2. Integrate a DSL with TensorFlow XLA or Halide.

Security, Verification, and Future Trends: Secure compilation and type-safe intermediate representations-Compiler fuzzing and formal verification (e.g., CompCert)-Quantum compilers, multi-target compilers, and neuromorphic systems

Activities:

1. Use CompCert to verify compilation of simple programs.
2. Apply compiler fuzzing using tools like libFuzzer.

Weightage: Continuous Assessment: 40%, End Semester Examinations: 60%

Assessment Methodology: Assignments (15), Quiz (10), Virtual Demo (20), Flipped Class Room (10), Review of Gate and IES Questions (25), Project (20).

References:

1. Cooper, K. D., & Torczon, L. (2023). Engineering a compiler. Morgan Kaufmann.
2. Grune, D., Bal, H. E., Jacobs, C. J. H., & Langendoen, K. G. (2012). Modern compiler design (2nd ed.). Springer.
3. Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2006). Compilers: Principles, techniques, and tools (2nd ed.). Pearson.
4. Völter, M. (2013). DSL engineering: Designing, implementing and using domain-specific languages. dslbook.org.
5. Sarda, S., & Pandey, M. (2015). LLVM essentials. Packt Publishing.

E-Resources:

1. Prof. AmeyKarkare, IIT Kanpur, "Advanced Compiler Optimizations"
Link: <https://www.cse.iitk.ac.in/users/karkare/Courses/cs738/>
2. Prof. Santanu Chattopadhyay, "Compiler Design", IIT Kharagpur
Link: https://onlinecourses.nptel.ac.in/noc21_cs07/preview

	Description of CO	PO	PSO
CO1	Explain intermediate control flow techniques in compiler design.	--	--

	Description of CO	PO	PSO
CO2	Apply program analysis techniques and advanced optimizations for design of compilers.	PO1(3)	PSO1(3)
CO3	Develop compiler features and machine learning techniques for optimization.	PO3(2)	PSO2(2)
CO4	Evaluate secure compilation strategies for quantum and multi-target compilation.	PO2(1)	PSO1(3)