Advt. No. CORP/GRP.A/04/2024
Recruitment to the post of Scientist B (Level 10) against Continuing contract vacancies on payscale

Important Notice for Candidates

Candidates who had applied for the post of Scientist B positions are hereby informed that the originally notified 15 domains have now been grouped into **7 broader domains.**

The revised domains are as follows:

Sr. No.	Notified Domains	Revised domains for the purpose of examination
1	Hardware VLSI design	Hardware – Embedded / VLSI /FPGA Design / IoT / System design
	Cyber Physical systems – Embedded Systems & IOT	
	Hardware – VLSI Design	
	Embedded systems & IoT	
	Hardware – VLSI / FGPA Design	
	Hardware System Design	
2	Dependable & Secure computing (Cyber Security)	Cyber Security
	Cyber Security (R&D)	
3	System Administrator	System Administrator
4	Applied AI & Data Analytics	Applied AI and Data Analytics
5	Applied Computing (e-Gov)	Applied Computing and Enterprise software development
	Enterprise software development	
6	HPC Software	HPC systems and Software Development
	HPC System Software Development	
7	Quantum Computing	Quantum Computing

- Candidates who have applied under multiple domains will now appear for **only one exam for each combined domain**, as per the revised structure, given above.
- The **syllabus for each of the 7 domains** reflects the merged content and is made available hereby.
- **The examinations are tentatively scheduled in 4th week of July / 1st week of August 2025.**
- **Candidates are advised to regularly check their registered email IDs, including the spam/junk folder, for updates regarding call letters and further communication.**

June 12, 2025 Director (HRD), C-DAC

Syllabus

Hardware – Embedded / VLSI /FPGA Design / IoT / System design

Embedded systems

1. **Embedded C Programming and Data structure -** Overview of C Programming language, Introduction to GNU Toolchain and GNU Make utility, Linux environment and vi editor, Tokens of C - Keywords, Data-Types, Variables, Constants, Operators, Identifiers, Storage Class Specifiers, Control Flow Statements, Arrays, Multidimensional arrays, Data Input & Output, Strings, Loops, Functions and Recursion, Pointers - Introduction, Pointer Arithmetic, Pointers and Arrays, Pointers and Functions, Pointers and Strings, Structures, Unions, Enum, Typedef, Bit field operators and pointers with structures, Preprocessors, C and Assembly, Files, I/O, Variable number of arguments, Command Line arguments, Error handling, Debugging and Optimization of C programs, Bit operations, Handling portability issues in C, Hardware, Time, Space and Power aware Programming. Algorithms and Abstract Data Types, Complexity of Algorithms, Linked Lists, Stacks, Queues, Searching and Sorting Algorithms, Hashing, Trees.

2. **Microcontroller programming and peripheral interface -** Overview of Microcontrollers, Microprocessors and SoC, RISC vs CISC, Harvard vs Princeton Architectures, Overview of Computer Architecture, Embedded Memories, Timers/Counters, UART, SPI, PWM, WDT, Input Capture, Output Compare Modes, I2C,CAN, LED, Switches, ADC, DAC, LCD, RTC, Bus Standards (USB, PCI), Programming in Assembly and Embedded C.

3. **ARM**: Overview of ARM Architecture and Organization, Introduction to Cortex-M Architecture, Programming Model and Instruction Set Architecture, Alignment and Endianness, Register access, State, Privilege, Stack, System Control Block, Power Modes, Memory Model, NVIC, Exception Handling, Bit- Banding, Peripheral Programming, SVCall, SysTick, PendSv, MPU, DMA, Mixing Assembly and C programs, Introduction to CMSIS & CMSIS Components, Overview of Cortex A & R architectures.

4. **RISC V**: Why RISC-V processor, RISC-V processor overview, ARM vs RISC-V, Modes in RISC-V, Setting up of necessary tools, RISC-V register set and calling convention, Instruction formats and type, Build Process, Practical examples of instructions, Detail description on Control and Status Registers, Exception handling, Examples in assembly for exception handling, Interrupts, Interrupt Entry and Exit procedure. Introduction to C-DAC VEGA processors.

5. **Power Management**: Power supply requirements for embedded systems, Low-power design techniques, Power modes of microcontrollers (sleep, deep sleep), Energy-efficient software design.

6. **Testing and Debugging in Embedded Systems**: Testing methodologies: unit testing, integration testing, system testing, On-chip debugging techniques: JTAG, SWD, Fault-tolerance and error-handling mechanisms, Testing tools: oscilloscopes, logic analyzers, debuggers.

IoT

1. **Introduction to IoT**

   Fundamentals of IoT: Introduction, Definitions & Characteristics of IoT, IoT Architectures, Physical & Logical Design of IoT, Enabling Technologies in IoT, IoT frameworks, IoT and M2M.

   Sensors Networks: Definition, Types of Sensors, Types of Actuators, IoT Development Boards: Arduino IDE and Board Types, RaspberriPi Development Kit, RFID principles and components, Wireless Sensor Networks: History and Context, The node, Connecting nodes, Networking Nodes, WSN and IoT.

2. **Networking and Communication Protocols -** Overview of Basic Networking Concepts (TCP/IP, OSI Model), MQTT, CoAP, LoRaWAN and Cellular Technologies in IoT, Bluetooth Low Energy (BLE) Network Topologies for IoT, Wireless Technologies for IoT: WPAN Technologies for IoT: IEEE 802.15.4, Zigbee, HART, NFC, Z-Wave, BLE, BACnet, Modbus. IP Based Protocols for IoT IPv6, 6LowPAN, RPL, REST, AMPQ, CoAP, MQTT. Edge connectivity and protocols

3. **IoT Hardware and Embedded Systems -** Overview of Microcontrollers (Arduino, Raspberry Pi, i.MX8), Types of Sensors and Actuators and Their Applications, Power Management Techniques, Embedded Programming Languages (C/C++, Python), Hardware Interfacing Techniques (GPIO, I2C, SPI), Basics of the Python programming language, Programming on the Raspberry Pi. Python on Raspberry Pi, Python Programming Environment, Python Expressions, Strings, Functions and Function arguments, Lists, List Methods, Control Flow.

4. **Data Management and Processing -** Data Acquisition Techniques from Sensors, Signal Processing Methods Role of Cloud Computing in IoT (AWS IoT, Azure IoT) Concepts of Edge Computing Comparison of Data Storage Solutions (SQL vs. NoSQL) Data Analytics in IoT, IoT Physical Servers and Cloud Offerings: Introduction to Cloud Storage models and communication APIs. Webserver – Web server for IoT, Cloud for IoT, Python web application framework, Designing a RESTful web API, Connecting to APIs. Introduction, Bigdata, Types of data, Characteristics of Big data,Data handling Technologies, Flow of data, Data acquisition, Data Storage, Introduction toHadoop. Introduction to data Analytics, Types of Data analytics, Local Analytics, Cloud analytics and applications.

5. **Security and Privacy in IoT -** Common IoT Security Challenges and Threats, Best Practices for Securing IoT devices, Overview of Encryption Methodologies (TLS, End-to-End Encryption),Privacy Concerns in Data Collection and Compliance Standards ,Security in WSN: Challenges of Security in Wireless Sensor Networks, Security Attacks in Sensor Networks, Protocols and Mechanisms for Security, IEEE 802.15.4 and ZigBee Security.

6. **IoT Applications and Use Cases -** Smart Homes, Smart Cities, and Industrial IoT, Healthcare Applications and Wearable Technologies, Environmental Monitoring and Precision Farming, Transportation and Fleet Management Solutions, Applications of IoT: Home Automation, Smart Cities, Energy, Retail Management, Logistics, Agriculture, Health and Lifestyle, Industrial IoT, Legal challenges, IoT design Ethics, IoT in Environmental Protection.

7. **Development Frameworks and Tools**
   • Overview of Popular IoT Platforms (AWS IoT, Azure IoT)

   • Development Tools and Environments (PlatformIO, Arduino IDE)

   • APIs for IoT Integration (REST, GraphQL)

   • IoT Simulation Tools (Cooja, IoTIFY)

8. **Testing and Quality Assurance**
   • Testing Strategies for IoT Devices (Unit, Integration, Performance Testing)

   Importance of Performance Monitoring

   • Reliability Testing Approaches

   • Tools for Testing IoT Applications (API Testing Tools)

9. **Emerging Technologies in IoT -** AI and Machine Learning in IoT, Blockchain Applications in IoT Security Augmented Reality (AR) and Virtual Reality (VR) Digital Twin Technology and its Applications Impact of 5G on IoT Development

<u>Hardware – System Design</u>

1. **Electronics Design Fundamentals:**

   - Introduction to Electronics: Signals, frequency Spectrum of Signals, Analog and Digital Signals, Linear Wave Shaping Circuits: RC LPF, Integrator, RC HPF, Differentiator. Properties of Semiconductors: Intrinsic, Extrinsic Semiconductors, Current Flow in Semiconductors, Diodes: p-n junction theory, Analysis of Diode circuits, Rectifiers,

   - Bipolar junction Transistor (BJTs): Physical Structures & Modes of Operation, Transistor Characteristics, DC analysis, Introduction to Small Signal Analysis, Transistor as an amplifier, The RC coupled amplifier, Introduction to Power Amplifiers, Transistor as switch. Field Effect Transistors (FETs): Physical Structures & Modes of Operation of MOSFETs, MOSFET Characteristics, DC Analysis. Feedback Amplifiers & Oscillators: General Principles, Different types of feedback amplifier. Voltage regulators, Voltage converters, Level Shifters.

   - Operational Amplifiers (OP-Amps): Ideal OP-AMP, Inverting Amplifier, Non-Inverting Amplifier. Adder, Subtractor, Integrator, Differentiator.

- Digital Fundamentals: Binary Numbers, Signed-binary numbers, Hexadecimal Number Systems, Logic Gates. Combinational and sequential logic design, Digital Logic families.

- Programmable Logic Devices: PLD, PGA, PLA, PAL, FPGA etc.

- Measuring and Test equipment: Introduction to Electronic Instruments, such as Oscilloscope, Multi-meter, Signal Generators, Logic Analyzer

2. **Computer Architecture Fundamentals:**

    - Introduction to Computer Architecture and Organization. Von Neuman Architecture, Harvard Architecture, Flynn Classification.

    - Computer Organisation: General register organization, stack organization, Instruction formats, Data transfer and manipulation, program control. RISC, CISC characteristics. Instruction Set Architecture (ISA). Pipeline and Vector processing: Pipeline structure, speedup, efficiency, throughput and bottlenecks. Arithmetic pipeline and Instruction pipeline.

    - Memory Organisation: RAM, ROM, Memory Hierarchy, Organization, Associative memory, Cache memory, and Virtual memory. DDRx memories, flash memories.

    - Input-Output Organization: Input-Output Interface, Modes of Transfer, Priority Interrupt, DMA, IOP processor.

    - Common Bus Architectures such as PCle, LVDS, SPI, I2C, USB etc.

    - Functionality and operation of common networking devices such as network switches, routers.

3. **Embedded System Design:**

    - Overview of Embedded System: Definition, Design Challenges and Characteristics, Categories and Requirements of Embedded Systems. Embedded Hardware and Software Development environment. Difference between microprocessor, microcontroller and DSP. General capability of microcontroller; microcontrollers in embedded systems. Suitability/selection of a microcontroller based on - Cost, Performance, Power dissipation and architecture- 8-bit, 16-bit, 32-bit. Concepts of system-on-chip.

    - Interfacing: I/O interfacing of devices such as LED, LCD, different sensors, ADC, DAC etc.

<u>VLSI</u>

1. **Advanced Digital design -** Combinatorial Logic Design, Sequential Logic Design: State machines, Counter Design, Advanced Design Issues: metastability, noise margins, power, fan-out, design rules, skew, timing considerations, Frequency divide Hazards. Asynchronous State Machine: Cycle stealing using latch in synchronous circuits,

Interfacing Asynchronous data flow, Asynchronous FIFO design, Asynchronous to Synchronous Circuit Interaction

2. **System Architecture -** System Building Blocks: knowledge of Computer Architecture, Memory Architectures, SPI, I2C, UART, eSPI, USB. FPGA Architecture: Architecture study of some popular FPGA families (Ultra Scale architecture), Architecture of Microcontrollers in FPGA (ARM), The backend tools, Integrating non-HDL modules: Building macros, Knowledge of System on Chip (SOC), Multicore Architecture.

3. **Verilog -** Module components, Data types, Operators, Modeling concepts ,Gate level Modeling, Data Flow Modeling, Behavioral modeling, Task and Functions, Compiler Directives, Specify block and Timing checks, Verification and Writing test benches, UDP, VCD, PLI, FSMD

4. **Simulation and Synthesis -** HDL Flow, The concept of Simulation, Types of simulation, HDL Simulation and Modeling, Simulation Vs Synthesis result, The Synthesis Concept, Synthesis of high level constructs, Timing Analysis of Logic circuits, Clock Skew, Clock Jitter, Combinatorial Logic Synthesis, State machine synthesis, Efficient coding styles, Partitioning for synthesis, Pipelining, Resource sharing, Optimizing arithmetic expressions, FPGA synthesis and implementation

5. **CMOS VLSI -** N-MOS, P-MOS and CMOS, Structure of MOS cells, Threshold Voltage, CMOS Inverter Characteristics, Device sizing, CMOS combinational logic design, Design of Basic gates, transmission gates and Design of complex logic circuit, Latch Up effect, Body Effect, Channel Length Modulation, CMOS as a switch, Noise Margin, Rise and fall times, Power dissipation, Knowledge of CMOS fabrication steps, Sequential CMOS logic.

6. **Fin-FET technology.** Application Specific Integrated Circuit (ASIC) Design Flow: Knowldege of Backend VLSI Design Flow – Libraries, Floor planning, Placement, Routing, Verification, Testing. Specifications and Schematic cell Design, Spice simulation, circuit elements, AC and DC analysis, Transfer Characteristics, Transient responses, Noise analysis of current and voltage, Design Rule, Micron Rules, Lambda rules of the design and design rule check, Fabrication methods of circuit elements, Layout design of different cells, Circuit Extraction, Electrical rule check, Layout Vs. Schematic (LVS), Post-layout Simulation and Parasitic extraction, Different design Issues like Antenna effect, Electro migration effect, Body effect, Inductive and capacitive cross talk and Drain punch through, etc., Design format, Timing analysis, Back annotation and Post layout simulation, DFT Guideline, Test Pattern and Built-in Self Test (BIST), ASIC design implementation.

7. **System Verilog (desirable) -** System Verilog Declaration Spaces, Data types, Arrays , structure, union, Procedural Blocks and Statements, Task and function, Verification using SV, Types of verification, Code coverage, task & functions in System Verilog, OOPs Terminology, Implementation of OOPs Concepts in System Verilog, Randomization, Assertions property, Assertions Time, Functional Coverage, FSMD methodologies and working principles, Verilog Regions

8. **Verification (UVM) -** Transaction, Test bench & its component, UVM class, UVM reporting, Device Under Test (DUT) and its connection with environment, Scoreboards, coverage, predictors, monitors, Hierarchy in UVM, Factory Overrides, Interfaces in UVM, Configuration, sequences Multiple Sequences configuration, UVM register Model, RM & its use in verification, RM integration, TLM (Transaction Level Modelling)

9. **Linux Shell scripting, Python (Desirable) -** Linux Commands, Linux File System, Vi editor, The Shell, Shell Programming, Basics of TCL scripting, Python - Operator and Expressions, Numbers, Strings, Lists, tuples, dictionary, standard I/O operations, functions, regex, OOPS concepts

Hardware - VLSI/FPGA Design

- **Digital Design (RTL design): IP design, ASIC/SoC design, FPGA based design from concept to implementation**
- **Digital Verification: Guide development of test plans, test benches and automated test cases**
- **Knowledge of synthesis, timing closure, and formal verification**
- **Knowledge of Physical design and verification**

Digital Design: Number System, Boolean Algebra and Gates, Combinatorial Logic, Sequential Logic

Computer Architecture: CPU Architecture (ARM, RISC-V etc.), Memory Architectures, system bus (PCI- Express), peripheral bus (USB), and LAN (Ethernet) etc.

VLSI Design Flow: RTL to GDS Implementation: Logic Synthesis, Physical Design; Verification and Testing; Post-GDS Processes

Hardware Modeling: Introduction to Verilog, Functional verification using simulation: testbench, coverage, mechanism of simulation in Verilog

FPGA Prototyping :

Architecture popular FPGA families, Xilinx high end FPGA family, Architecture of Microcontrollers in FPGA (ARM), FPGA tools

RTL Synthesis: Verilog Constructs to Hardware Logic Optimization: Definitions, Two-level logic optimization

Logic Optimization: Multi-level logic optimization, FSM Optimization Formal Verification: Introduction, Formal Engines: BDD, SAT Solver

Static Timing Analysis: Synchronous Behavior, Timing Requirements, Timing Graph, Mechanism, Delay Calculation, Graph-based Analysis, Path-based Analysis, Accounting for Variations

Constraints: Clock, I/O, Timing Exceptions Technology Mapping Timing-driven Optimizations

Design for Test: Basics and Fault Models, Scan Design Methodology, ATPG, BIST

Basic Concepts for Physical Design: IC Fabrication, FEOL, BEOL, Interconnects and Parasitics, Signal Integrity, Antenna Effect, LEF files

Chip Planning: Partitioning, Floorplanning, Power Planning

Placement: Global Placement, Wirelength Estimates, Legalization, Detailed Placement, Timing-driven Placement, Scan Cell Reordering, Spare Cell Placement

Clock Tree Synthesis: Terminologies, Clock Distribution Networks, Clock Network Architectures, Useful Skews Routing: Global and Detailed, Optimizations Physical Verification: Extraction, LVS, ERC, DRC, ECO and Sign-off

<u>Embedded Systems and IoT</u>

1. **C Programming Language**
    a. The C Programming Model and Development Environment
    b. Tool chains, Optimization, Libraries, Debugging Tools,
    c. Data Types and Variables
    d. Storage Classes in C
    e. Statements, Loops
    f. Arrays, Structures, Unions, Pointers, Enums
    g. Bit Operations, Registers, Directives
    h. Data Structures in C – Singly Linked Lists, Doubly Linked Lists, Circular Buffers, Trees, Graphs
2. Microcontroller Architecture and Programming
    a. Microcontroller Architectures – Harvard, Von Neuman, CISC, RISC
    b. Memory Architectures – Flash, RAM, NVRAM, Serial Flash, EEPROM
    c. Analog circuits – ADC, Comparators, DAC,
    d. General Purpose IO
    e. Clocks, Timers, Watchdog, Real Time Clock
    f. Embedded Peripheral Interfacing - Serial peripherals: UART, SPI, I2C, CAN
    g. Interrupts and Nested Interrupts, Interrupt Controllers
3. Operating System Concepts and Linux Programming
    a. Process Management, File Management, Device Management, Scheduling, Memory Management
    b. IPC, Synchronization Techniques, Shared Memory
    c. Interrupts and Interrupt Vectors, Handlers and Service Routines
    d. Device Drivers, Kernel Programming, Device Tree Sources, System Calls
    e. Linux System and Application Programming
    f. Filesystem Types, Virtual File Systems - Proc FS, SysFS, Dev FS,
    g. Libraries – Static and Dynamic Libraries,
    h. Bootloader Concepts
    i. Real Time Operating System Concepts – Schedulers, Priority based Scheduling Algorithms, Determinism, Priority Inversion and Inheritance,
4. Embedded Hardware Design Concepts and Power Supplies
    a. Discrete Analog Circuit Design – OpAmps circuits: Amplifiers, Comparators, Integrators, Differentiators, Hysteresis

    b. Microcontroller Board Bring Up – Crystal Oscillators, Power Supply Decoupling, Reset Circuits, Analog and Digital Ground Isolations
    c. Power Supply Circuits – Linear Regulators, Low Drop Out oscillators, Switched Mode Power Supplies – Buck, Boost, Buck Boost, Isolated, Non-Isolated
    d. Input and Output Device Interfacing – Analog Sensors, Serial Peripheral Interfacing, Digital Sensor Interfacing, LCD Interfacing, OLED Interfacing, Memory Chip Interfacing

5. Internet of Things
    a. IoT Communication Topologies – Mesh, Star, Multihop
    b. Wireless IoT Protocols
        i. LPWAN: LoRa, NBIoT, LTE-CAT M1, SigFox,
        ii. Bluetooth and Bluetooth Low Energy
        iii. ZigBee
        iv. WiFi
        v. UWB
        vi. 4G and 5G
    c. Sensors and Types of Sensors – Acoustic Sensors, Climate Sensors, Navigation & Location Sensors, Proximity Sensors,
    d. Actuators and Output Devices
        i. Motors – BLDC, DC, Stepper, Servo
        ii. Displays – LCD, OLED,
        iii. Buzzers,
    e. Battery Chemistries for IoT Use cases
        i. Rechargable Chemistry
        ii. Non-Rechargable Chemistry
        iii. Battery Charging Circuits
    f. Basics of Network Security

<u>Cyber Security</u>

1. Cyber Security Fundamentals

- Definition and Importance of Cyber Security
- Key Principles: Confidentiality, Integrity, Availability (CIA Triad)
- Types of Cyber Security: Information Security, Application Security, Network Security, Cloud Security
- Security Policies and Frameworks: ISO/IEC 27001, NIST Cybersecurity Framework, CIS Controls
- Risk Management: Identifying Risks and Vulnerabilities, Risk Assessment and Mitigation Strategies
- Incident Response: Phases of Incident Response (Preparation, Detection, Response, Recovery), Post-Incident Review and Reporting

2. Network Security

- Network Security Basics: OSI and TCP/IP Models, Types of Networks (LAN, WAN, VPN), secure communication protocols etc

- **Common Network Attacks**: Man-in-the-middle, DoS/DDoS, packet sniffing, IP spoofing etc
- Firewalls: Types (Packet Filtering, Stateful, Application), Configuration and Management
- Intrusion Detection and Prevention Systems (IDPS IPS), XDR: Signature-Based vs. Anomaly-Based Detection, Implementation and Tuning
- VPN Technologies: VPN Protocols (IPSec, SSL, L2TP), Secure Remote Access Solutions
- Wireless Security: Wi-Fi Security Protocols (WPA2, WPA3), and vulnerabilities in wireless communication, Risks and Mitigation Strategies
- Network Monitoring and Logging: Security Information and Event Management (SIEM), Best Practices for Network Monitoring

3. Application Security

- Secure Software Development Lifecycle (SDLC): Security in Development Phases, DevSecOps Principles
- Common Vulnerabilities: OWASP Top Ten (e.g., SQL Injection, XSS, CSRF)
- Secure Coding Practices - **Secure Coding Principles**: Input validation, error handling, data sanitization., **Best Practices**: Use of static analysis tools, code review, secure memory management
- Application Security Testing: Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST)
- API Security: Authentication and Authorization Protocols (OAuth, JWT), Securing REST and SOAP APIs
- Mobile Application Security: Threats to Mobile Apps, Secure Coding Guidelines for iOS and Android, Secure storage, encryption, app sandboxing, mobile-specific threats, etc
- **OS Security:** SELinux, AppArmor, Container etc
- Web Application Firewalls (WAF): Purpose and Implementation, Rules and Policies for WAF
- **OWASP Top 10 Vulnerabilities**: Cross-site scripting (XSS), SQL injection, CSRF, etc.

4. Firmware Security

- **Firmware Basics**: Firmware architecture, types of firmware, and secure boot processes.
- **Firmware Vulnerabilities**: Buffer overflows, memory corruption, hardware backdoors, supply chain risks.
- **Firmware Integrity Checks**: Techniques for secure firmware updates, encryption

5. Cryptography

- Basic Cryptographic Concepts: Symmetric vs. Asymmetric Encryption,
- **Hashing Functions**: SHA, MD5, and their security implications.
- **Digital Signatures and Certificates**: PKI, certificate authorities, and the chain of trust.

- Cryptographic Algorithms: AES, RSA, ECC, DES, Blowfish, symmetric vs. asymmetric encryption
- Key Management: Key Generation, Distribution, and Storage, Public Key Infrastructure (PKI), and secure lifecycle management.
- SSL/TLS: How SSL/TLS Works, Implementing HTTPS
- Cryptanalysis: Techniques Used in Cryptanalysis, Common Vulnerabilities in Cryptography

6. Identity and Access Management (IAM)

- IAM Fundamentals: Concepts of Authentication, Authorization, and Accounting (AAA)
- Access Control Models: Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC)
- Single Sign-On (SSO): Benefits and Implementation Strategies, Protocols (SAML, OAuth, OpenID Connect)
- Multi-Factor Authentication (MFA): Types of MFA, Best Practices for Implementation
- Identity Lifecycle Management: User Provisioning and De-Provisioning, Role Management and Access Reviews
- Privileged Access Management (PAM): Securing Admin Accounts, Implementing the Least Privilege Principle

7. Cloud Security

- Cloud Security Basics: Shared Responsibility Model, Cloud Deployment Models (IaaS, PaaS, SaaS)
- Security Controls in Cloud Environments: Data Encryption and Tokenization, Identity and Access Management in the Cloud
- Compliance and Governance: Relevant Standards (ISO 27017, CSA STAR), Regulatory Compliance (GDPR, HIPAA)
- Incident Response in the Cloud: Cloud-Specific Incident Response Plans, Forensic Investigations in the Cloud
- Security in Cloud Architecture: Secure Application Design in the Cloud, Threat Modeling for Cloud Services

8. Security Operations and Monitoring

- Security Operations Center (SOC): Roles and Responsibilities of a SOC, SOC Tools and Technologies
- Monitoring and Incident Detection: Types of Monitoring (Network, Host, Application), Alerting and Escalation Procedures
- Log Management: Best Practices for Log Collection and Analysis, SIEM Solutions and Usage
- Threat Hunting: Techniques for Proactive Threat Detection, Tools for Threat Hunting
- Incident Response Procedures: Developing and Testing Incident Response Plans, Continuous Improvement of Response Strategies

9. **Governance, Risk, and Compliance (GRC)**

- Introduction to GRC: Importance of Governance in Cyber Security, Risk Management Principles
- Regulatory Compliance: Key Regulations (GDPR, HIPAA, PCI-DSS), Compliance Frameworks and Standards
- Risk Assessment: Risk Assessment Methodologies, Risk Treatment Strategies
- Security Policies and Procedures: Developing and Enforcing Security Policies, Policy Review and Updates
- Auditing and Reporting: Internal and External Audit Processes, Documentation and Reporting Requirements

10. **Security Testing**

- Types of Security Testing: Penetration Testing, Vulnerability Assessment, Red Team vs. Blue Team Exercises
- Tools for Security Testing: Common Tools (Nmap, Metasploit, Burp Suite), Integrating Security Testing in CI/CD Pipelines
- Testing Methodologies: OWASP Testing Guide, NIST SP 800-115
- Reporting and Remediation: Vulnerability Reporting Best Practices, Prioritizing Remediation Efforts

11. **Cybersecurity Threats and Attack Techniques**

- Types of Threats: Malware (Viruses, Trojans, Ransomware, rootkits, adware, spyware), Social Engineering Attacks (Phishing, Spear Phishing, social engineering, insider threats)
- Attack Techniques: Advanced Persistent Threats (APTs) - Definition, behavior, and common examples, Denial of Service (DoS) and Botnets, Distributed Denial of Service (DDoS) - Concepts, attack methods, and mitigation strategies
- Threat Intelligence: Gathering and Analyzing Threat Intelligence, Using Threat Intelligence for Defense
- **Threat Modelling**: STRIDE, DREAD, and risk assessment methodologies.
- Incident Response to Attacks: Incident Response Plans for Different Attack Types, Forensics and Post-Attack Analysis
- **Malware Analysis - Static Analysis**: Signature-based detection, file hashes, binary inspection; **Dynamic Analysis**: Behavioral analysis, sandboxing, and debugging malicious code., **Reverse Engineering Malware**: Tools and techniques for deconstructing malware.
- **Vulnerability Analysis - Common Vulnerabilities**: CVEs, zero-day exploits, memory corruption, race conditions; **Exploitation Techniques**: Buffer overflows, privilege escalation, remote code execution.
- **Penetration Testing - Penetration Testing Phases**: Reconnaissance, scanning, exploitation, reporting; **Reporting and Remediation**: Vulnerability disclosure, patch management, and reporting procedures.

12. Emerging Trends and Technologies in Cyber Security

- Artificial Intelligence and Machine Learning in Cyber Security: Applications of AI in Threat Detection, Challenges and Limitations of AI, behavior analysis, and anomaly detection.
- **AI-based Security Tools**: AI-driven SIEM, intrusion detection, and malware classification systems.
- **Challenges in AI Security**: Adversarial attacks, AI model poisoning, and defense techniques.
- **Cybersecurity of AI - Securing AI Models**: Protecting against data poisoning, evasion, and inference attacks., **Trust and Explainability in AI**: Issues with transparency and accountability in AI-driven systems, **AI Bias and Fairness in Security**: Recognizing and mitigating biases in AI security models.
- Zero Trust Security Model: Principles of Zero Trust, Implementing Zero Trust Architecture
- Internet of Things (IoT) Security: Risks and Challenges in IoT Security, Best Practices for Securing IoT Devices
- Quantum Computing and Cyber Security: Potential Impact of Quantum Computing on Cryptography - Quantum-Safe Cryptography**,** Preparing for Quantum-Resistant Algorithms
- Cybersecurity Frameworks and Standards: Continuous Evolution of Standards, Importance of Adaptability in Cybersecurity Practices
- **Elliptic Curve Cryptography (ECC)**: Use cases, strengths, and weaknesses.
- **Blockchain-based Cryptography**: Merkle trees, hash-based cryptography, zero-knowledge proofs.
  Blockchain Fundamentals: Decentralized ledgers, consensus mechanisms (PoW, PoS) etc.
  Smart Contracts: Security vulnerabilities, formal verification of contracts.
  Blockchain Use in Cybersecurity: Decentralized identity management, supply chain security, data integrity.

System Administrator

1. **Basic Linux Concepts and Linux Operating System Fundamentals** Download, Install and Configurations of Linux Operating System, System Access and File System
2. **Linux System Administration**
3. **Linux File Editors Vi (or Vim) and Nano:** Advantages, Differences, functionality and useability
4. **User accounts and Group management:** Creating and managing user/group accounts, setting up user permissions and access control, and monitoring activity**.**
5. **Users and Sudo access :** Managing the custom permissions for users and Sudoers
6. **Linux Directory Service** - Account Authentication

7. **Linux Commands :** System utility, Processes and schedules, System Monitoring, OS Maintenance, System logs monitor, Changing System Hostname, Finding System Information, Recover root Password, Environment variables
8. **Shell Scripting :** Linux Kernel, what is a Shell, Types of Shells, Basic Shell scripts
9. **Networking :** Networking Servers and System Updates, enabling internet in Linux VM, Network Components, Network files, NIC Information, NIC or port bonding, Download files with URLs, curl and ping commands, File transfer commands, System updates and repositories, System Upgrade/Patch Management, Create Local Repository from CD/DVD, Advance Package Management, Rollback Patches and Updates, SSH and Telnet, DNS, Hostname and IP Lookup, NTP, Apache Web Server, Central Logger, OpenLDAP
10. **Securing Linux Machine** (OS Hardening)
11. **Disk Management and Run Levels :** System run levels, Linux Boot Process, Message of the Day, Storage, Disk partition(Add Disk and Create Standard Partition, Logical Volume Management (LVM), LVM Configuration during Installation, Add Disk and Create LVM Partition, extend disk using LVM, Adding swap space, RAID, File System Check.
12. **System Backup** (dd Command)
13. **Network File System (NFS**)

Applied AI and Data Analytics

1. Python Programming

- Core Python: Data types, control structures, functions, and file handling.
- Object-Oriented Programming (OOP): Classes, inheritance, and polymorphism.
- Libraries: Familiarity with NumPy, Pandas, Scikit-learn, TensorFlow, PyTorch, and FastAPI.
- Data Visualization: Using Matplotlib, Seaborn, and Plotly for insights.
- Performance Optimization: Profiling and improving code efficiency.
- Sorting and Searching: Merge Sort, Quick Sort, Binary Search.
- Data Structures: Arrays, Linked Lists, Stacks, Queues, Trees, Graphs, and Hash Tables.
- Threading: Basics of Python threading and understanding thread safety.
- Multiprocessing: Parallel execution using the multiprocessing module.
- Concurrency vs. Parallelism: Distinguishing and implementing both approaches.

2. Machine Learning & Deep Learning

- Statistical Learning: Basics of model training, optimization, and evaluation.
- Ensemble Methods: Combining multiple models like Boosting, Bagging, and Random Forest.

- Transfer Learning & Meta-Learning: Techniques to reuse or adapt models for new tasks.
- Deep Learning: Understanding CNNs, RNNs, LSTMs, Transformers (like BERT, GPT).
- NLP & Computer Vision: Working with advanced text and image processing models.
- Generative AI: Using LLMs, Diffusion Models, and Retrieval-Augmented Generation (RAG).

3. MLOps & Data Engineering

- Version Control: Tracking experiments and models with tools like MLflow and DVC.
- CI/CD Pipelines: Automating testing and deployment of ML models.
- Containerization: Deploying models with Docker and Kubernetes.
- Data Engineering: Managing large-scale data using Spark and data lakes.
- Real-Time Processing: Tools like Kafka for streaming and Spark Streaming for live data.
- Feature Engineering: Creating and managing useful data features.

Applied Computing and Enterprise software development

A. Applied Computing (e-Gov)

1. Core Java

- **OOP Principles**: Classes, Objects, Inheritance, Polymorphism, Encapsulation, Abstraction
- **Exception Handling**: Checked vs. Unchecked Exceptions, Custom Exceptions, try-catch-finally, Throws/Throw
- **Collections Framework**: Lists, Sets, Maps, Queues, Iterators, Generics
- **Multithreading & Concurrency**: Threads, Executors, Synchronization, Locks, volatile, atomic
- **JVM Internals**: Memory Management, Garbage Collection, Class Loaders, Bytecode
- **I/O Streams & NIO**: File Handling, Byte & Character Streams, Buffering, Channels
- **Lambda Expressions & Streams API**: Functional Programming, Stream Operations, Parallel Streams
- **JDK 8+ Features**: Optional, Default Methods, Stream API, CompletableFuture

2. **Java EE & Spring Framework**

- **Servlets & JSP**: Request-Response Cycle, Session Management, JSP Scripting
- **JPA & Hibernate**: ORM Concepts, Annotations, Criteria API, JPQL, Caching, Entity Lifecycle
- **Spring Core**: Dependency Injection, Inversion of Control, Beans, ApplicationContext
- **Spring MVC**: Controllers, Views (JSP/Thymeleaf), Form Handling, Validation, REST API Development
- **Spring Boot**: Auto-Configuration, Profiles, Embedded Servers, Starters, Properties Configuration
- **Spring Data JPA**: Repositories, Query Methods, Transactions, Paging & Sorting
- **Spring Security**: Authentication, Authorization, JWT, OAuth2, Method Security
- **Spring Cloud**: Microservices, Eureka, Ribbon, Feign, Config Server, Circuit Breakers (Hystrix)
- **Web Services**: RESTful Web Services, SOAP, JSON/XML Marshalling

3. **Database Management & SQL**

- **Relational Databases**: ER Modeling, Normalization (1NF, 2NF, 3NF), ACID Properties, Transactions
- **SQL Queries**: SELECT, INSERT, UPDATE, DELETE, Joins, Subqueries, Aggregations, Group By, Having
- **Indexes & Optimization**: Types of Indexes, Indexing Strategies, Query Optimization, Execution Plans
- **Database Design**: Entity-Relationship Diagrams, Foreign Keys, Primary Keys, Constraints
- **Stored Procedures & Triggers**: Writing Procedures, Functions, Event Triggers, Cursors
- **NoSQL Databases**: Key-Value Stores, Document Stores (e.g., MongoDB), Column Stores (e.g., Cassandra)
- **Data Integrity & Consistency**: Constraints, Transactions, Referential Integrity, Isolation Levels

4. **Web Technologies**

- **HTML & CSS**: HTML5 Elements, CSS3 Layouts, Flexbox/Grid, Responsive Design, Media Queries
- **JavaScript & ES6+**: Variables (let/const), Arrow Functions, Promises, Async/Await, Modules

- **Front-End Frameworks**: React.js, Angular, Vue.js, Component Lifecycle, State Management
- **AJAX & Fetch API**: Asynchronous Requests, XMLHTTPRequest, Fetch API, Promises
- **RESTful APIs**: API Design, CRUD Operations, HTTP Methods, Headers, Status Codes
- **WebSockets & Real-Time Communication**: WebSocket Protocol, Long Polling, Server-Sent Events
- **CSS Preprocessors**: SASS, LESS, Mixins, Variables, Functions
- **Browser DevTools**: Debugging, Performance Analysis, Network Monitoring, Accessibility Testing

5. **Software Architecture & Design Patterns**

- **Software Architecture Styles**: Monolithic, Microservices, Event-Driven, Layered Architecture
- **Design Patterns**: Singleton, Factory, Builder, Prototype, Strategy, Observer, Decorator, Adapter
- **SOLID Principles**: Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion
- **Microservices Communication**: REST, RPC, Message Brokers (Kafka, RabbitMQ), gRPC

f. DevOps & CI/CD

- **Version Control Systems**: Git, Branching Strategies, Merge & Rebase, Pull Requests
- **CI/CD Pipelines**: Jenkins, GitLab CI, CircleCI, Automated Builds, Continuous Deployment
- **Containerization**: Docker, Docker Compose, Container Registry, Image Optimization
- **Orchestration**: Kubernetes, Docker Swarm, Helm Charts, Service Mesh (Istio)
- **Monitoring & Logging**: Prometheus, Grafana, ELK Stack (Elasticsearch, Logstash, Kibana), Fluentd
- **Automated Testing in CI/CD**: Unit Tests, Integration Tests, End-to-End Tests, Code Coverage Tools

g. Software Project Management

- **Project Lifecycle Models**: Waterfall, Agile, Scrum, Kanban, Lean Software Development
- **Agile Frameworks**: Sprint Planning, Daily Standups, Retrospectives, Backlog Grooming, Scrum Roles
- **Task Management & Tracking Tools**: Jira, Trello, Asana, GitHub Issues
- **Risk Management**: Risk Identification, Mitigation Strategies, Risk Logs

- **Stakeholder Communication**: Communication Plans, Reporting, Client Interactions

h. Quality Assurance & Testing

- **Unit Testing**: JUnit, Mockito, TestNG, TDD (Test-Driven Development), BDD (Behavior-Driven Development)
- **Integration Testing**: Testing APIs, Databases, Microservices Communication
- **End-to-End Testing**: Selenium, Cypress, Puppeteer, Postman for API Testing
- **Performance Testing**: JMeter, LoadRunner, Stress Testing, Benchmarking, Bottleneck Analysis
- **Security Testing**: Penetration Testing, Vulnerability Scanning, OWASP Testing Guide
- **Automated Testing**: Continuous Testing, Frameworks for Test Automation (Selenium, Appium)
- **Test Coverage & Metrics**: Code Coverage, Test Reports, SonarQube, Static Code Analysis

i. Security & Compliance

- **Web Security Principles**: XSS (Cross-Site Scripting), SQL Injection, CSRF (Cross-Site Request Forgery)
- **Authentication & Authorization**: OAuth2, JWT, SSO (Single Sign-On), Multi-Factor Authentication
- **Data Encryption**: Symmetric/Asymmetric Encryption, TLS/SSL, HTTPS, Hashing Algorithms (SHA, MD5)
- **Compliance Standards**: GDPR, HIPAA, PCI-DSS, ISO/IEC 27001
- **Security Audits & Penetration Testing**: Vulnerability Assessment, Threat Modeling, Red Team/Blue Team Exercises
- **Secure SDLC (Software Development Life Cycle)**: Security in Design, Secure Coding Practices, Security Testing

j. Microservices

- **Microservices Design**: Decomposition Strategies, Bounded Contexts, Independent Deployment
- **Service Discovery**: Eureka, Consul, Zookeeper, Dynamic Service Registration
- **API Gateway**: Zuul, API Gateway Patterns, Rate Limiting, Circuit Breaking
- **Inter-Service Communication**: REST, Message Brokers (Kafka, RabbitMQ), gRPC, Event-Driven Architecture
- **Resilience Patterns**: Circuit Breaker (Hystrix, Resilience4j), Bulkheads, Retry Patterns, Fallback

- **Data Consistency & Transactions**: Saga Pattern, Eventual Consistency, Two-Phase Commit (2PC)
- **Microservices Security**: OAuth2, JWT, Secure Communication between Services
- **Observability in Microservices**: Distributed Tracing (Zipkin, Jaeger), Metrics (Prometheus), Log Aggregation

B. Enterprise Software Development

1. Algorithms and Data Structures

- Problem Solving & Computational Thinking
- Constructs, Designs, Complexity analysis, OO design, Basic Data Structures

2. Concepts of Operating Systems (OS)

- Operating System concepts with Linux environment
- Shell Programming
- Process Management, Memory Management, Virtual Memory, Deadlock

3. Software Development Methodologies

- Software Development Life Cycles, Models, Tools
- Design and Architectural Engineering - Design approaches, Modularity, Cohesion, Coupling, Layering, Design Models, UML
- DevOps - ecosystem, phases, methodologies, tools, basics of Cloud Native development
- Software testing
- Versioning systems

4. Database technologies

- Database Management Systems – Concepts, Types, Data Models, Database Design
- Relational Database Management systems such as PostgreSQL, MySQL, SQL Programming (database queries, functions, triggers, procedures), NoSQL such as MongoDB

5. Web Programming Technologies

- Web architecture
- HTML, CSS, JavaScript, JSON, Ajax, Node.js, Express.js, React, Angular
- Responsive Web Design

6. Web-based Java Programming

- J2EE, Servlets, Session Management, JSP
- Spring & Spring Boot Frameworks including Microservices Architecture
- RESTful web services
- Web Application Security

7. Mobile app development

- Fundamentals of mobile apps – Types, Mobile app ecosystem (Android, iOS), Design & Development process, Programming fundamentals (Variables, Control structures)
- Mobile IDEs & tools, Design elements, Architecture, Components, Testing tools
- Cross platform Mobile app development frameworks
- Mobile app security

HPC Systems and Software Development

A. HPC System Software Development

1. **Operating Systems -** Process Management, Scheduling, Interprocess Communication & Synchronization, Memory Management, I/O subsystem & File Systems, POSIX Thread Programming, POSIX Semaphores, Mutexes, Conditional Variables, Shared Memory
2. **C programming** – Data-Types, Variables, Constants, Operators, Identifiers, Preprocessors , arrays, pointers, basics of Data Structures, Algorithms and Abstract Data Types, Complexity of Algorithms, Linked Lists, Stacks, Queues, Searching and Sorting Algorithms, Hashing, Trees.
3. **Linux programming -** GNU Toolchain, Linux environment and editors, Debugging and Optimization of C programs, file handling, signal handling, shell commands, scripting, static linking & dynamic linking, cross-compilation
4. **Device driver programming -** Linux Kernel Modules and Module Programming, Char Device Drivers, Kernel Internals: Dynamic memory allocations, Handling Delays, Timers, Synchronization, Locking, I/O Memory and Ports, Interrupts, Deferred Executions, Driver Debugging Techniques
5. **Embedded programming -** Programming in Assembly and Embedded C, Microcontrollers, Microprocessors and SoC, RISC vs CISC, Timers/Counters, UART, SPI, PWM, Input & Output, I2C, CAN, LED, LCD, RTC, Bus Standards (USB, PCI), ARM, RISC-V
6. **Network programming -** OSI layer, Socket Programming, IP addressing
7. **Computer Architecture and Organization -** Instruction Set Architecture, Cache design and coherency, Arithmetic Logic Unit, Floating Point Unit, Instruction Set Pipelining, Parallel Processing Architectures, Distributed systems

B. HPC Software

1. Exposure to x86_64 instruction set, addressing modes, and performance characteristics of x86_64 processors. Understanding the ARMv8/v9

architecture, its features, and its suitability for HPC. Knowledge of the RISC-V instruction set architecture, its modular design, and its potential for HPC.

2. **BIOS/UEFI Firmware - Coreboot, EDKII**
   Understanding of BIOS/UEFI Concepts, UEFI Specifications, Device Tree, Trusted Platform Module, Building and Porting Coreboot, UEFI Driver Development and its stages.
3. **Operating System and Concepts - Linux**
   Understanding Linux kernel architecture, file systems, processes, and memory management. Shell scripts for automating tasks for HPC. Configuring and managing Linux systems for HPC environments.
4. **Programming Languages - C, C++, Fortran, Python**
   Proficiency in C, C++, Fortran and Python programming, data structures, algorithms.
5. **Compilers and Toolchain - GCC, LLVM, GNU Toolchain**
   Using GCC for compiling C, C++, and Fortran programs, and compiler optimizations. Understanding the LLVM compiler infrastructure, its modular design, and its use in development and optimization of compilers. Compiler for GPGPU (NVIDIA/AMD etc.) and AI accelerators (TVM/XLA/Glow etc.). Using the GNU toolchain for profiling, linking, binary analysis and debugging.
6. **Parallel Programming Models - MPI, OpenMP, Hybrid Programming (MPI + OpenMP), CUDA, OpenACC, OpenCL, SyCL**
   MPI concepts, MPI Data types, point-to-point communication, collective operations, and various MPI implementations. OpenMP directives, shared memory parallelism, and its use in multi-core systems and devices. Combining MPI and OpenMP for parallelizing applications with both shared and distributed memory approach.
   CUDA programming, GPU architecture, kernel functions, memory usage and optimization. OpenACC directives for offloading computations to GPUs and managing data transfers. OpenCL for cross-platform device programming and heterogeneous computing and its use in HPC. SyCL as a C++-based abstraction for heterogeneous computing.
7. **Performance Analysis and Debugging - GNU Profiler, GNU Debugger, TAU, HPC Toolkit**
   GNU Profiler to measure program execution time, identify performance bottlenecks, and optimize code. Debugging serial and parallel programs using GNU Debugger, setting breakpoints, inspecting variables, and analyzing memory usage. TAU for performance analysis, profiling, and visualization of HPC applications. Analyzing application performance using HPC Toolkit
8. **Libraries and Benchmarks - Linear algebra (BLAS, LAPACK), NAS Parallel Benchmarks, Floating Point Number System**
   Using BLAS and LAPACK for efficient linear algebra operations in HPC applications. Evaluating the performance of HPC systems using the NAS Parallel Benchmarks suite. IEEE 754, POSIT
9. **HPC Resource Management, Scheduler and Runtime**
   SLURM's architecture, job submission and scheduling, resource allocation, and user management. HPC Scheduling Concepts and Algorithm for efficient

scheduling across heterogenous computing resources. Runtime for enabling program execution on different hardware devices.

- **Computer and HPC Architecture**
- **Operating system (Linux)**
- **C/ Modern C++ and python,**
- **data structure and algorithms**
- **Compiler design**
- **Parallel Programming models- OpenMP,CUDA and SYCL**
- **Message passing techniques**
- **Programming libraries - math, domain and AI based**

Quantum Computing

1. Quantum Mechanics Fundamentals
2. Mathematical Foundation – Linear Algebra
3. Quantum Information Science
4. Basics of Quantum Computing
5. Quantum Algorithms
6. Quantum Hardware and Architectures
7. Control Electronics and Measurement Hardware
8. Quantum Optics
9. Quantum Programming and Simulation Tools – (Qiskit, Cirq)
10. Quantum Error Correction
11. Post-Quantum Cryptography